

Speaker: Andreas Lipowsky /CEO
Chat Moderator: Leonard Immel (Creator of GUI Editor)

Agenda

- Welcome
- GUI editor concept
- Introduction to the essential steps for creating a Baby-LIN GUI
- Description of widget including new features
- Question and answers

Starting point

- SDF solutions for LIN- and CAN- applications
- Usage of bus signals, virtual signals, protocols, macros and events
- SDF execution with SimpleMenu frontend or stand alone.

Depending on the application and device, also with a user interface. The GUI elements were previously available for this purpose.

SessionConf v2.36.1 - [C:/documents/presentations/Gui-Editor-Webinar/Demo-Gui-Elements-EKB00xx.sdf]

File Edit View Tools Help

Hide expert settings Required SDF version: v3.20 FID:

SDF Version 3	Type	Name	Target	Comment
0	Monitored signal	SDF-Project	SDF-Project	
1	Monitored signal	SDF-Version	SDF-Version	
2	Edit signal	Cfg-PositionTolerance	Cfg-PositionTolerance	
3	Macro	DutConfig-670100-EKB0001	DutConfig--670100-EKB0001	
4	Macro	DutConfig-670200-EKB0001	DutConfig--670200-EKB0001	
5	Macro	DutConfig-670100-EKB0012	DutConfig-670100-EKB0012	
6	Macro	DutConfig-670200-EKB0012	DutConfig-670200-EKB0012	
7	Macro	ConfigAccess	ConfigAccess	
8	Macro	ConfigParams	ConfigParams	
9	Macro	ConfigAux	ConfigAux	
10	Macro	Reference	Reference	
11	Macro	RunToPos	RunToPos	
12	Macro	WritePartNum	WritePartNum	
13	Macro	WriteFunctionId	WriteFunctionId	
14	Macro	WriteHwVersion	WriteHwVersion	
15	Macro	SavePosition	SavePosition	
16	Macro	VerifyAll	VerifyAll	

14-LIN: SDF-Diag ToolBox V. 1.10

Section properties

- Bus description
- Emulation
- Tables
- Virtual signals
- Signalfunctions
- Protocols
- GUI-Elements (SimpleMenu/HARP etc)**
- Macros
 - [0] DutConfig--670100-EKB0001
 - [1] DutConfig--670200-EKB0001
 - [2] DutConfig-670100-EKB0012
 - [3] DutConfig-670200-EKB0012
 - [4] ConfigAccess(Nad, PidReq, Pid...
 - [5] ConfigParams(FsTime, FsPositi...
 - [6] ConfigAux(Rollback, SafeDista...
 - [7] _resetCycleFlags
 - [8] Reference(PosReq, TimeoutSec)
 - [9] RunToPos(TargetPos, TimeoutS...

Signals Macros Macroselections

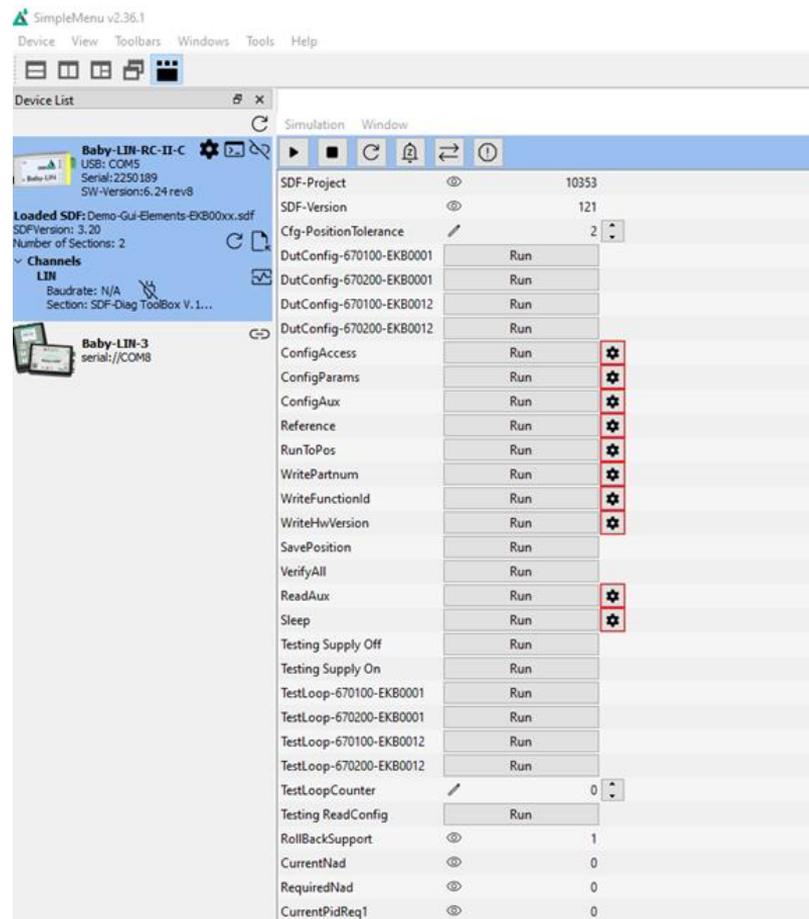
Add signal by drag and drop or double click

Filter:

SignalNr	Signalname	Frame	
0	VA_3W_InfVar_Posn_Rq	VA_3W_Rq_Frm	LIN_Master (master)
1	VA_3W_InfVar_RefDrive_EnbL_Rq	VA_3W_Rq_Frm	LIN_Master (master)
2	VA_3W_InfVar_PosnActSav_Rq	VA_3W_Rq_Frm	LIN_Master (master)
3	VA_3W_Posn_Stat	Stat_Frm_0x1F	Valve
4	VA_3W_Move_Stat	Stat_Frm_0x1F	Valve
5	VA_3W_Err_Stat	Stat_Frm_0x1F	Valve
6	VA_3W_Err_ElectricFault_Stat	Stat_Frm_0x1F	Valve
7	VA_3W_Err_UnderVolt_Stat	Stat_Frm_0x1F	Valve
8	VA_3W_Err_OverVolt_Stat	Stat_Frm_0x1F	Valve
9	VA_3W_Err_OverTemp_Stat	Stat_Frm_0x1F	Valve
10	VA_3W_Err_MechBreak_Stat	Stat_Frm_0x1F	Valve
11	VA_3W_Err_UnexpBlock_Stat	Stat_Frm_0x1F	Valve
12	VA_3W_Err_RefFault_Stat	Stat_Frm_0x1F	Valve
13	VA_3W_Err_Group_ERR_Stat	Stat_Frm_0x1F	Valve
14	VA_3W_Err_Group_SNA_Stat	Stat_Frm_0x1F	Valve

Disadvantages of the previous GUI elements

- Displays only via Simple Menu (exception HARP), therefore no solution for stand-alone applications.
- There were no design options when using the GUI elements in the SimpleMenu.
- Always everything arranged in a vertical list, always everything visible, always everything in the same text size, text color etc.
- User-friendly GUI could not be realized by this means.



Development goals GUI editor

1. Gui editor is part of session Conf
2. Use of the SimpleMenu option also for existing Generation 2 devices
3. Use as a stand-alone GUI on the new Baby-LIN-3 devices with their own display (Baby-LIN-3-RC/Baby-LIN-3-RCplus)
4. Use as a stand-alone GUI via the web interface of the Baby-LIN-3-MB.

Concept of the new GUI solution

- Creation of GUI by LINWorks SessionConf
- Saving the GUI definitions in the SDF
- Functional replacement for the previous GUI elements (Signal, Macro, etc)
- Additional design elements (widgets)
text label, image, line, frame, meter and more.
- Free positioning of the widgets
- Extensive setting options for the widget properties
- Different layouts for the various playback paths (SimpleMenu, DeviceDisplay)
- Distribution of information across several panels
- Control widget visibility dynamically from the SDF.

GUI playback paths

- **Standalone solution**
Gui definition from SDF is loaded into the device and executed locally there.
- **SimpleMenu solution**
The Gui definitions are only loaded into the DLL (not into the device) and executed on the PC, so Generation 2 devices can also use this playback path with all features.
- **BL-3-MB web interface solution**
The SDF runs on the device and a contained GUI definition is delivered to a connected WebBrowser and executed there.

Technical background:

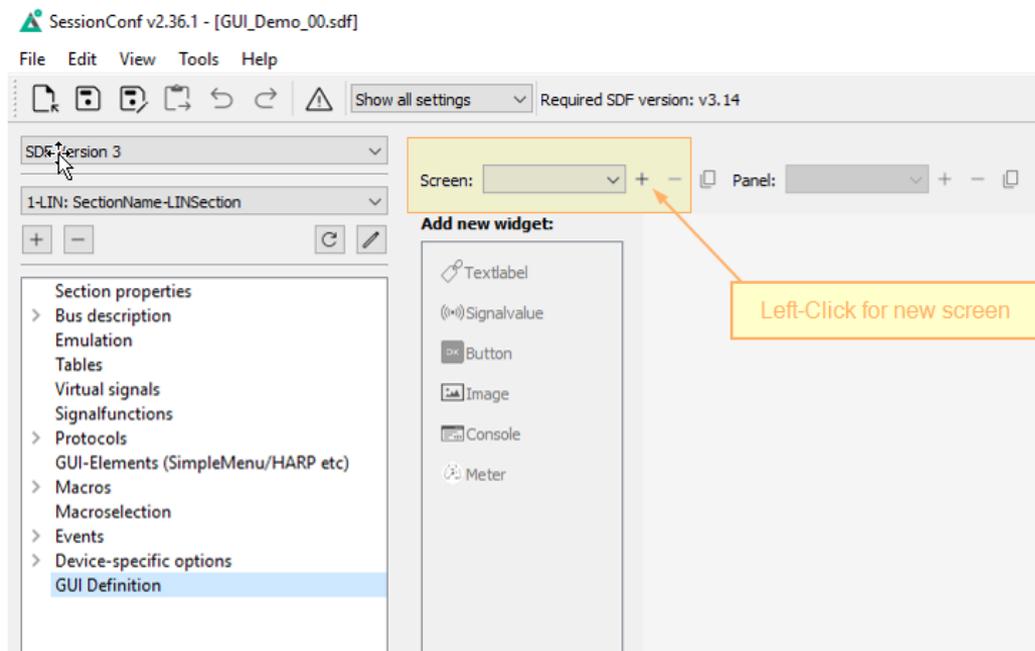
As all three delivery methods use the same code base, simple further development and maintenance of these software components is guaranteed.

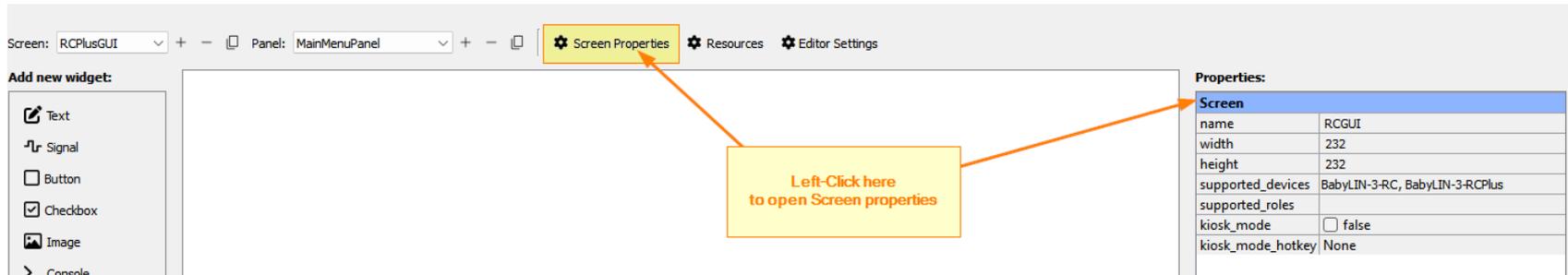
Basic elements of GUI Editor

- **Screen**
Is the basic element and defines the resolution of the display device and the possible playback paths
Multiple screens for different devices possible in one SDF.
- **Panel**
One or more panels per screen
Grouping of visual information on one panel
Panel selection/switching possible at runtime
- **Ressourcen**
Resources that can be used for all panels of an SDF, such as fonts, images
- **Widgets**
Individual display elements that are positioned on a panel.

Screen

- First element when creating a new GUI.
- Is the basic element and defines the resolution.
- Screens for different devices possible in one SDF.
- Simple Menu and Web-Gui screens do not require any memory on the USB Baby-LIN-3 devices!



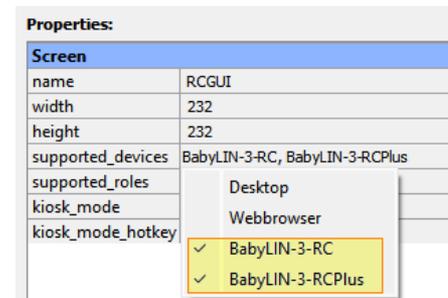


Supported devices

decides on which devices this screen should be executed

Important hint:

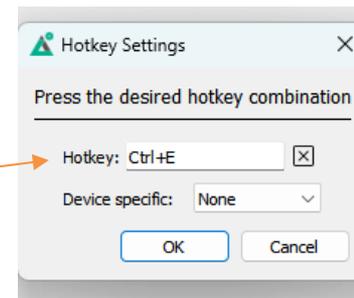
To run a GUI on Baby-LIN-3-RC and Baby-LIN-3-RCplus make sure both device are selected.



Kiosk Mode

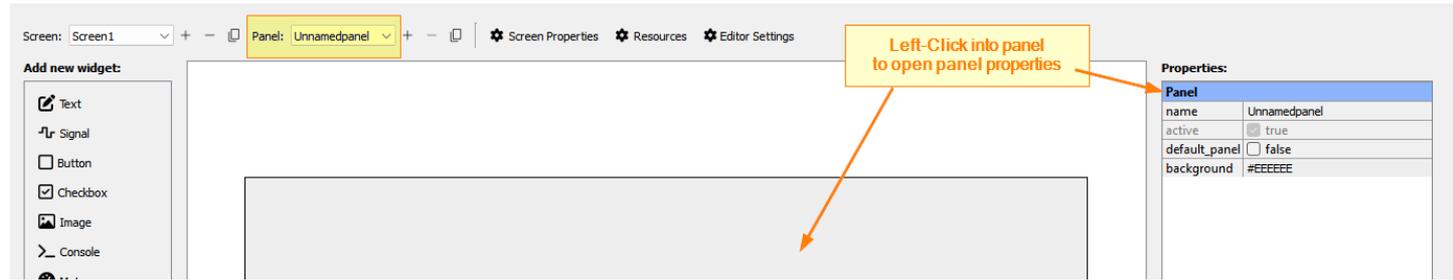
If this is checked, the SDF will open in SimpleMenu without showing the device List and old GUI definitions.

A hotkey can be assigned to stop KioskMode. To assign a hotkey for SimpleMenu, just enter key or key combination on PC keyboard.



Panel

- One or more panels per screen.
- Definition of visual information by arranging corresponding widgets on a panel.
- A first panel is automatically created when a screen is defined.
- The panel name, the background color and the default_panel property can be set in the panel properties
- As long as there is only one panel, the default_panel property doesn't matter.
- If there are several panels, the default_panel property controls which panel is displayed at the start (property "active"). If the default_panel property is not set for any panel, the 1st panel is set to "active"="true".
- The property "active" is not editable in SessionConf as it will be defined during runtime, to mark the actual displayed panel. This is an exclusive property per screen, so only one panel can be active at the same time.
- In GUI-editor, each currently shown panel will have the active property set.



Ressources

These are resources that can be shared by all panels.

- 4 fixed defined text fonts
- Import of Images (PNG)
- Import of custom fonts (ttf)

Further resource types might be added in the future.

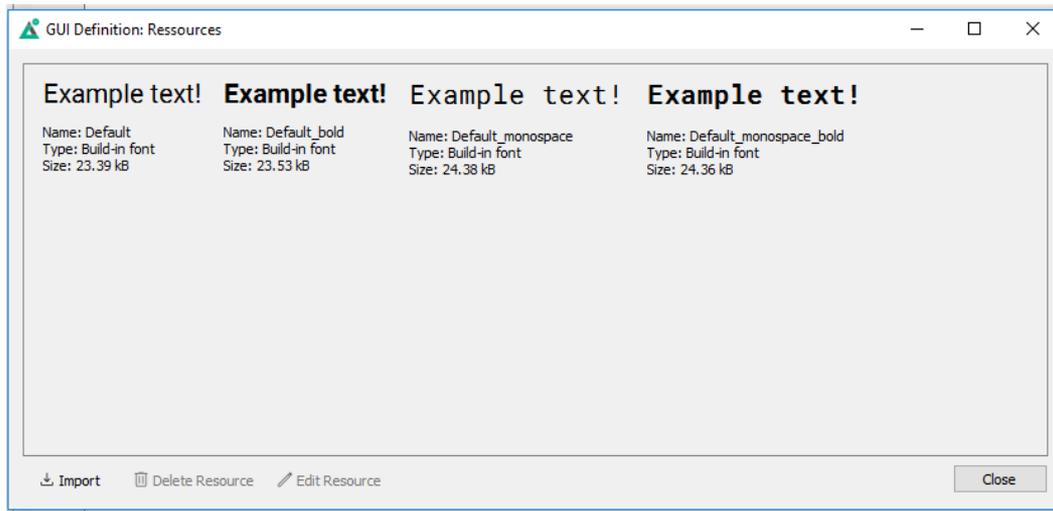


Image Import

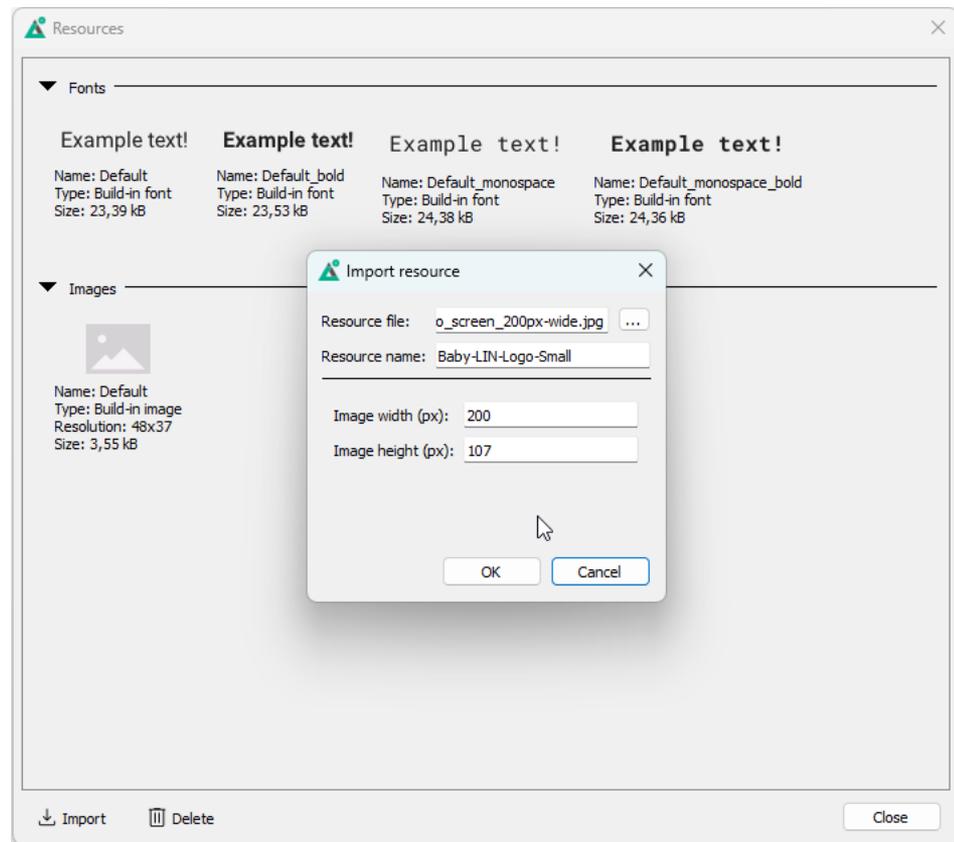
Actually pictures must be imported in the resolution suitable for the selected screen resolution.

Resizing during positioning an image widget on a panel is not yet supported.

So, resize you image with an external picture editing program so have the right size.

Keep in mind that the Baby-LIN-3-RC/Rcplus display has a total reolutopm of 232 x 232 pixel.

So, it can make sense to hold 2 versions of a picture in your resources, in case you GUI provides screens for desktop and device displays.



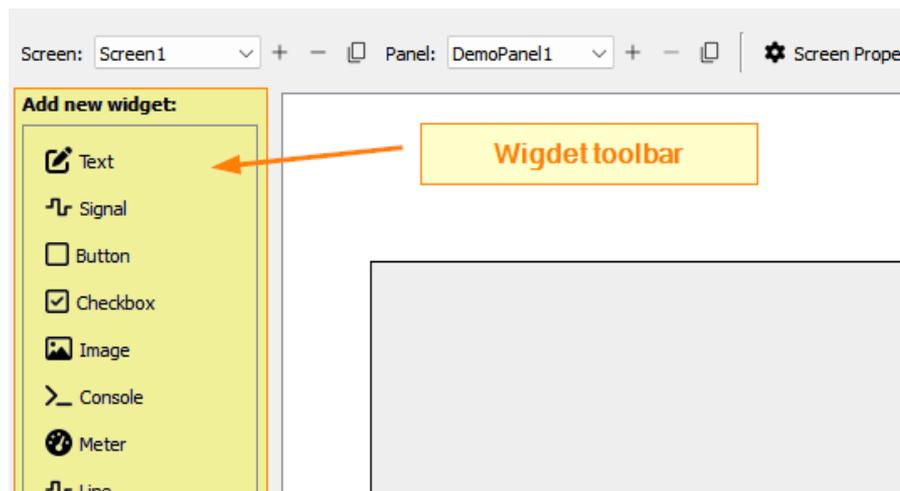
Widgets

After defining the screen and panel, widgets can be placed on the actual select panel.

By clicking on a widget in the widget toolbar on the left, it is placed on the panel and can be moved to the desired position.

Clicking the same widget in the toolbar multiple times adds a “Stacked Widget indicator” to the widget indicating the number of existing overlaid widgets. Here the 4 indicated that 5 text labels had been added.

A properties menu with the specific setting options is displayed on the right for each selected widget.



Back

Font type test:

Default Font <>

Default Font Bold

Default Mono Font

Default Mono Font Bold

CUSTOM FONT RESOURCE

Text color test:

Default Color (Black)

White Text

Red Text

Green Text

Blue Text

Text alignment test:

Default Alignment (Left)

Left Aligned

Centered

Right Aligned

Text wrap test:

Wrap Mode: Clip (XXXXXXXXXX)

Wrap Mode: Dots (XXXXXXXXXX...)

Wrap Mode: Scroll (XXXXXXXXXX)

(XXXXXXXXXX) Wrap Mode: Scro

Wrap Mode: Expand (XXXXXXXXXX)

Properties:

Widget	
name	label
x	356
y	238
width	99
height	27
visible	<input checked="" type="checkbox"/> true
z_index	13
opacity	100%
bg_color	#FFFFFF
bg_opacity	0%
Text properties	
font	Default
font_size	24
text_color	#00FF00
text_alignment	Default
text_wrap_mode	Clip
Label	
text	Green Text
text_recolor	<input checked="" type="checkbox"/> true

- New Properties
- Background color
- Background opacity
- Extended label text editor
- text_recolor to change color within text

Back

This is a multiline message.
Showing color change in text
Now **textcolor blue**
and now **textcolor red**
and now Black again

Edit Text

Enter text:

This is a multiline message.
Showing color change in text
Now #0000FF textcolor blue #
and now #FF0000 textcolor red #
and now #000000 Black again #

Predefined symbols

OK **Cancel**

Properties:

Widget

name	label
x	152
y	84
width	436
height	162
visible	<input checked="" type="checkbox"/> true
z_index	0
opacity	100%
bg_color	#FFFFFF
bg_opacity	0%

Text properties

font	Default
font_size	24
text_color	#000000
text_alignment	Default
text_wrap_mode	Clip

Label

text	now #000000 Black again # ...
text_recolor	<input checked="" type="checkbox"/> true

Description:

Allows to change the color of parts of the text using a in-text formatter.

Format: "#RRGGBB #"
Example: "#FF0000 RedText # NormalText "
 or when changing text by macro print command from SDF:
 [GUI]
 setprop widgetname "text"="#FF0000 RedText # NormalText"
 If a text must contain a '#' character, just put '##'

Zoom: 100%

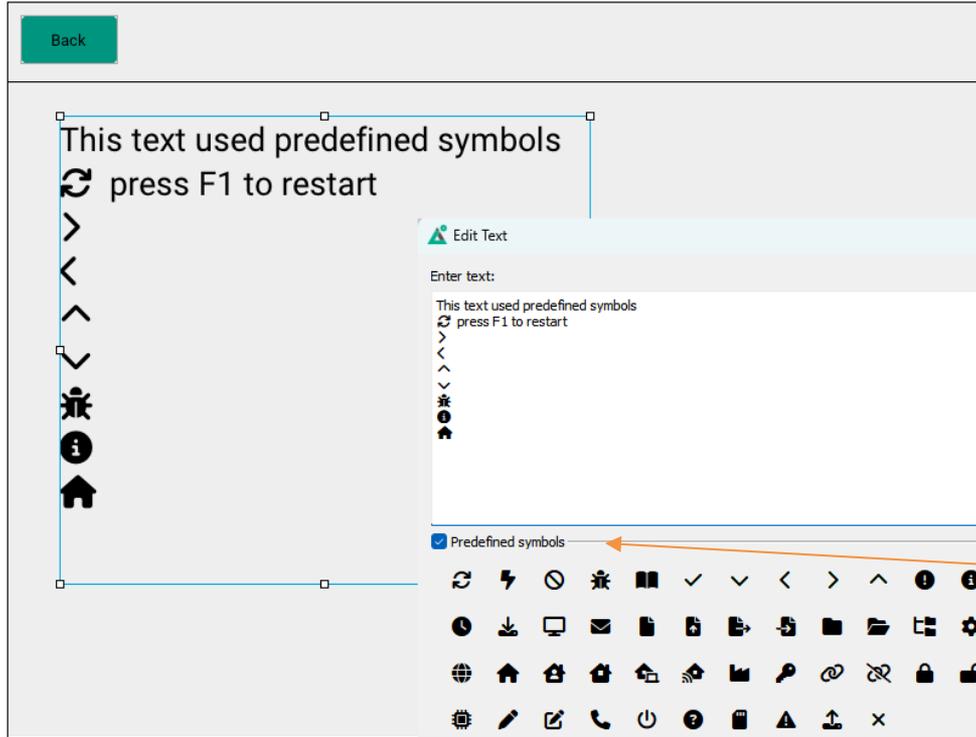
Property text_recolor to change color within text.

Color changes are entered via Text editor

Check this to allow recoloring

Left-Click here shows this help text

If recoloring is enabled, same syntax can be used in setprop ... "text"= commands from within SDF.



Properties:

Widget	
name	label
x	42
y	92
width	436
height	383
visible	<input checked="" type="checkbox"/> true
z_index	0
opacity	100%
bg_color	#FFFFFF
bg_opacity	0%
Text properties	
font	Default
font_size	32
text_color	#000000
text_alignment	Default
text_wrap_mode	Clip
Label	
text	> \n<\n^<\n>\n#<\n@<\n!< ...
text_recolor	<input checked="" type="checkbox"/> true

Extended label text editor
with predefined symbols

Property change on multiple elements

Change the properties of several elements at the same time by selecting them and setting the property to the new common value they all should have.

Here the width of all selected text labels will be changed to 100.



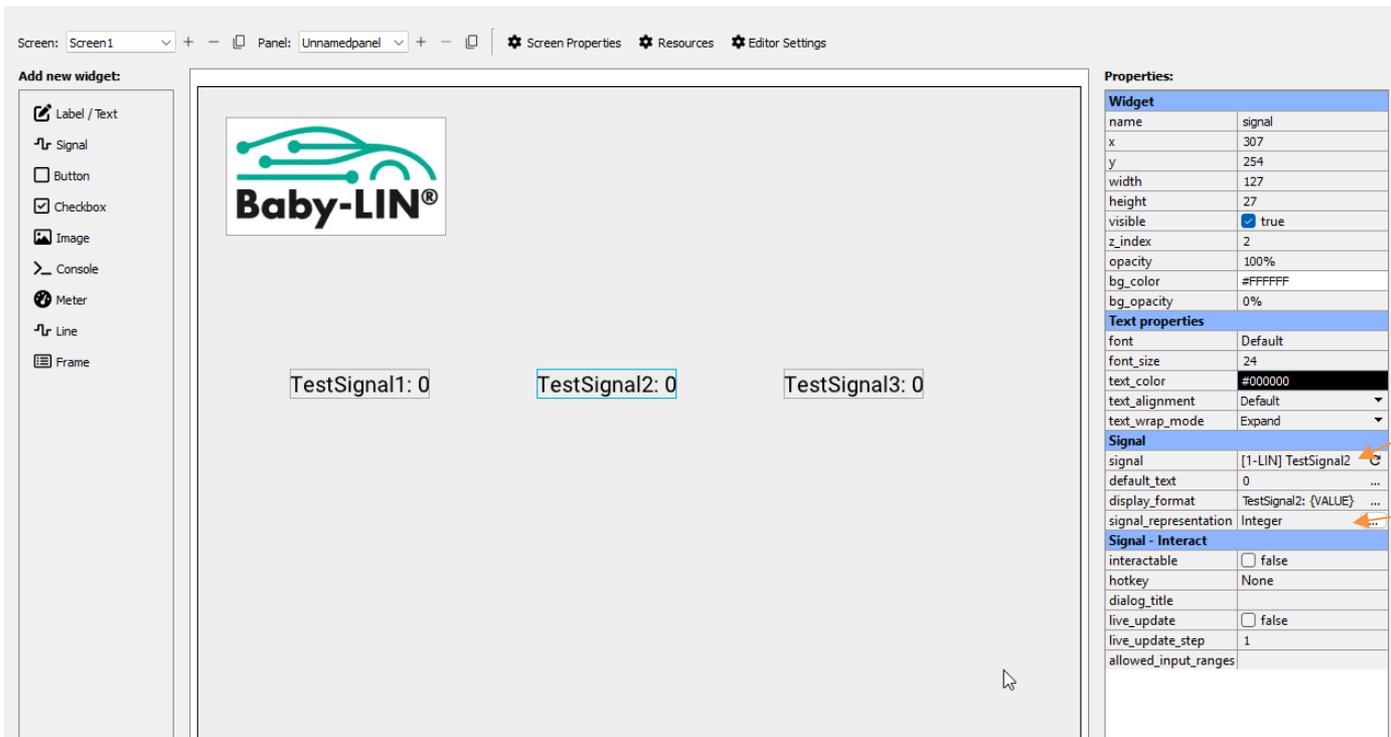
The screenshot shows a software interface with a list of five 'Textlab' widgets on the left. An orange arrow points from the selected widgets to the 'width' property field in the 'Properties' panel on the right. The 'width' field is highlighted in blue and contains the value '100'. The 'Properties' panel also shows other widget and text properties.

Widget	
name	label
x	63
y	[...]
width	100
height	30
visible	<input checked="" type="checkbox"/> true
z_index	[...]
opacity	100%
bg_color	#FFFFFFF
bg_opacity	0%
Text properties	
font	Default
font_size	24
text_color	#000000
text_alignment	Default
text_wrap_mode	Clip
Label	
text	Textlabel ...
text_recolor	<input checked="" type="checkbox"/> true

Screen: Screen1 + - Panel: Unnamedpanel + - Screen Properties Resources Editor Settings

Add new widget:

- Label / Text
- Signal
- Button
- Checkbox
- Image
- Console
- Meter
- Line
- Frame



Properties:

Widget	
name	signal
x	307
y	254
width	127
height	27
visible	<input checked="" type="checkbox"/> true
z_index	2
opacity	100%
bg_color	#FFFFFF
bg_opacity	0%
Text properties	
font	Default
font_size	24
text_color	#000000
text_alignment	Default
text_wrap_mode	Expand
Signal	
signal	[1-LIN] TestSignal2
default_text	0
display_format	TestSignal2: {VALUE} ...
signal_representation	Integer
Signal - Interact	
interactable	<input type="checkbox"/> false
hotkey	None
dialog_title	
live_update	<input type="checkbox"/> false
live_update_step	1
allowed_input_ranges	

Signal linkage into SDF

Signal representation

Interactive section

Note: In contrast to the previous GUI elements, the new GUI is global for all bus sections contained in the SDF, allowing signals from different bus sections to be displayed in same panel.

Signal values can be displayed as an

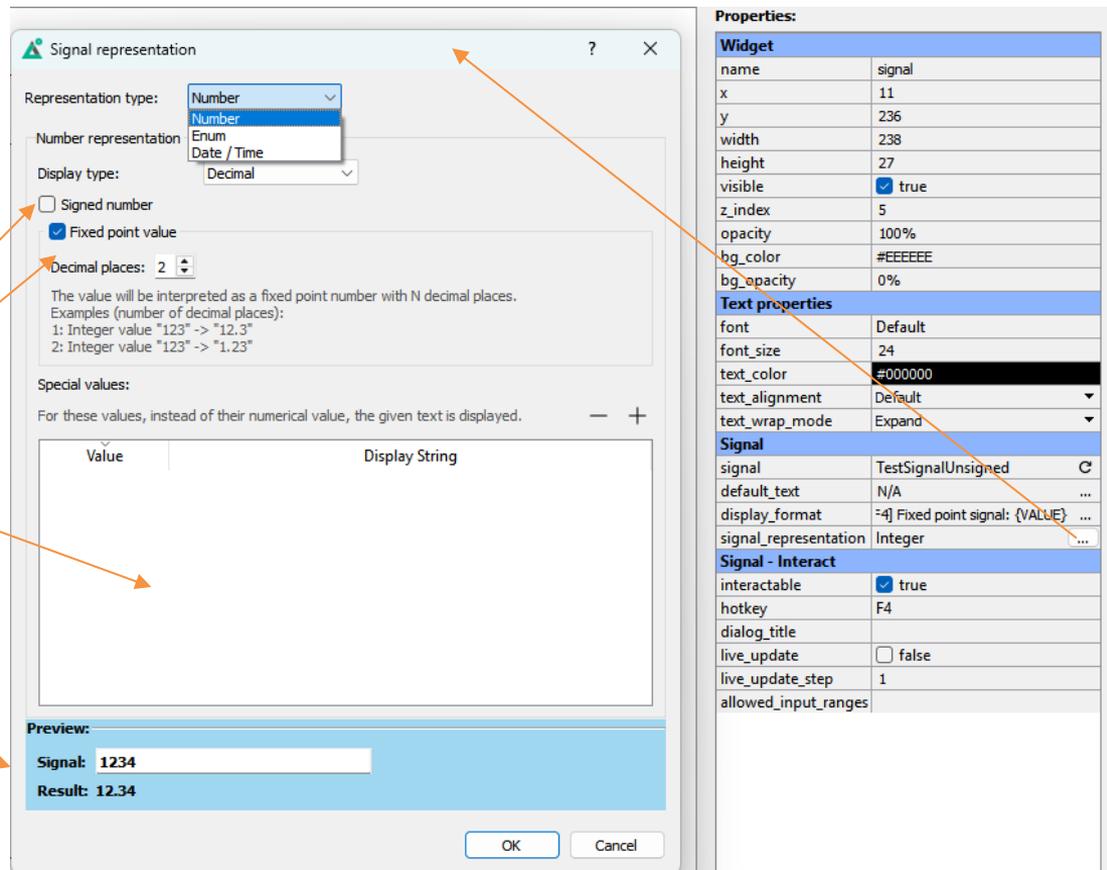
- number
- enum or
- converted time.

For number presentation the interpretation as signed integer and an optional fixed point representation can be chosen.

Now representation type number you can also define special value, which will then be shown instead of value.

Eg. 0 for OFF etc

The preview shows the displayed value for the given LIN signal value.



Signal representation

Representation type: **Number**

Number representation

Display type: **Decimal**

Signed number

Fixed point value

Decimal places: **2**

The value will be interpreted as a fixed point number with N decimal places.
Examples (number of decimal places):
1: Integer value "123" -> "12.3"
2: Integer value "123" -> "1.23"

Special values:

For these values, instead of their numerical value, the given text is displayed.

Value	Display String

Preview:

Signal: **1234**

Result: **12.34**

Properties:

Widget	
name	signal
x	11
y	236
width	238
height	27
visible	<input checked="" type="checkbox"/> true
z_index	5
opacity	100%
bg_color	#EEEEEE
bg_opacity	0%
Text properties	
font	Default
font_size	24
text_color	#000000
text_alignment	Default
text_wrap_mode	Expand
Signal	
signal	TestSignalUnsigned
default_text	N/A
display_format	[-4] Fixed point signal: {VALUE}
signal_representation	Integer
Signal - Interact	
interactable	<input checked="" type="checkbox"/> true
hotkey	F4
dialog_title	
live_update	<input type="checkbox"/> false
live_update_step	1
allowed_input_ranges	

Back

No Signal: N/A

Default: N/A

[F2] Unsigned signal: N/A

[F3] Signed signal: N/A

[F4] Fixed point signal: N/A

[F5] 16 Bit Hex signal: N/A

[F6] Enum signal: N/A

Time signal: N/A

Date signal: N/A

Date & Time signal: N/A

Live update: N/A

Live update Enum: N/A

Set 0

Set 1

Set 2

Set Invalid

Widget	
name	signal
x	11
y	316
width	191
height	27
visible	<input checked="" type="checkbox"/> true
z_index	8
opacity	100%
bg_color	#FF0000
bg_opacity	0%
Text properties	
font	Default
font_size	24
text_color	#000000
text_alignment	Default
text_wrap_mode	Expand
Signal	
signal	DeviceLabelExpandState
default_text	N/A
display_format	[F6] Enum signal: {VALUE}
signal_representation	Enum
Signal - Interact	
interactable	<input checked="" type="checkbox"/> true
hotkey	F6
dialog_title	
live_update	<input type="checkbox"/> false
live_update_step	1
allowed_input_ranges	

Property section Interact

Enables value editor in GUI.

Value Editor can either be opened by

- Mouse click in SimpleMenu or WebGui
- Hotkey press
- By focus & select method on device

Per default value editors pass the edited value to the SDF only on closing the editor.
 So changed signal value gets active only after closing the editor, e.g. sent on the LIN bus.
 Sometimes it is better to have a kind of immediately signal change update.
 This is enabled by `live_update` feature.

In this case the signal editor looks like this.

For every left-click on the up/down arrows (SimpleMenu or Web Gui) or every press on up/down buttons on the device keypad, the value will be incremented or decremented by the given step size.

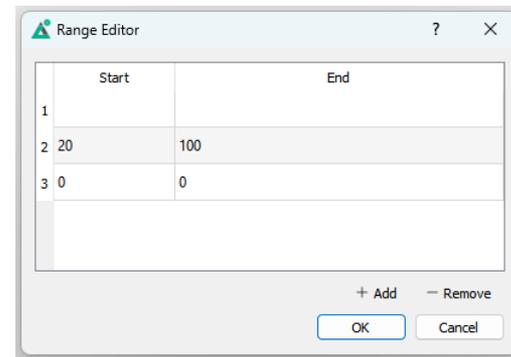
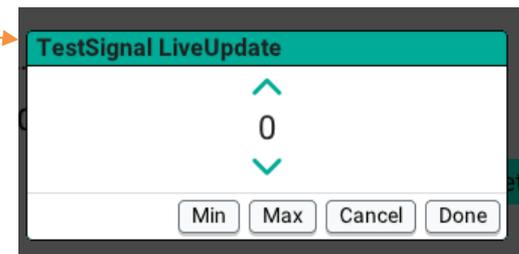
Values are limited by property allowed range.
 If no range is defined the signal size defines the minimal and maximal values.

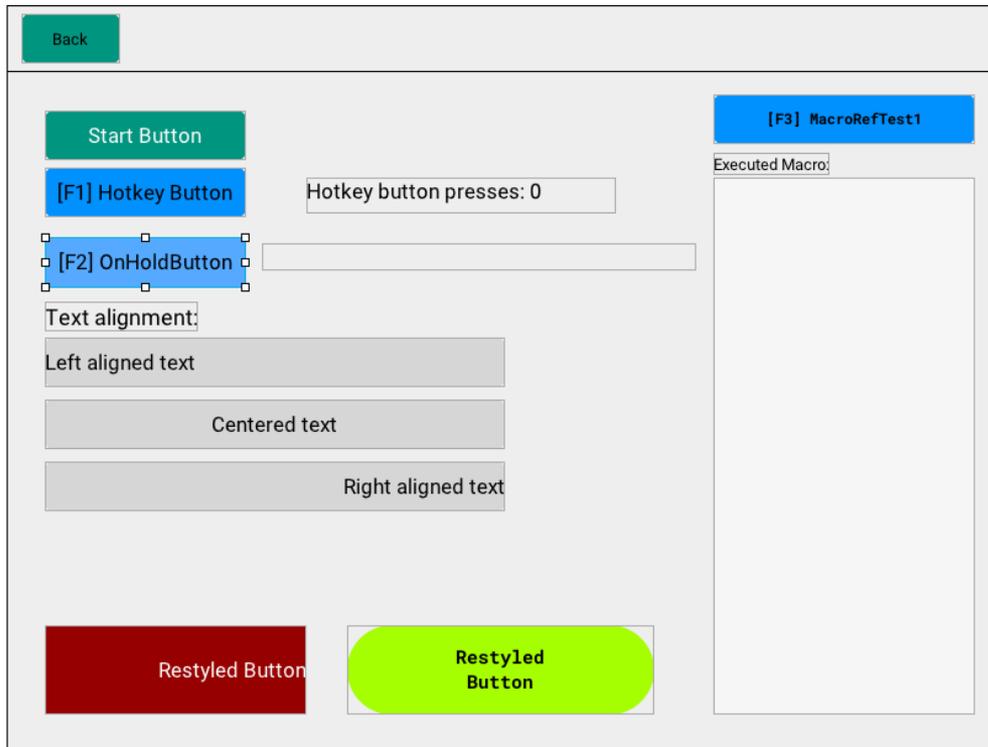
The Min and Max buttons allow for quick setup of minimal and maximal value of allowed range.

Allowed range can be configured by combining several ranges.
 Here an example for a combined range allowig values for `{0, 20, 21, 22,...100}`

For fine tuning a signal value, `live_ude` offers a more convenient user experience.

Signal - Interact	
interactable	<input checked="" type="checkbox"/> true
hotkey	None
dialog_title	TestSignal LiveUpdate
live_update	<input checked="" type="checkbox"/> true
live_update_step	10
allowed_input_ranges	0,20-100





The screenshot displays a GUI editor interface with several interactive elements:

- Back**: A green button in the top left corner.
- Start Button**: A green button.
- [F1] Hotkey Button**: A blue button with a hotkey indicator.
- [F2] OnHoldButton**: A blue button with a hotkey indicator and a small square icon next to it.
- Text alignment:** A section showing three text boxes: "Left aligned text", "Centered text", and "Right aligned text".
- Hotkey button presses: 0**: A text box next to the [F2] button.
- [F3] MacroRefTest1**: A blue button with a hotkey indicator.
- Executed Macro:**: A large white text area below the [F3] button.
- Restyled Button**: A red rectangular button.
- Restyled Button**: A bright green rounded rectangular button.

Properties:

Widget	
name	button
x	30
y	186
width	163
height	41
visible	<input checked="" type="checkbox"/> true
z_index	5
opacity	100%
Text properties	
font	Default
font_size	20
text_color	#000000
text_alignment	Center
text_wrap_mode	Clip
Button	
background	#55AAFF
text	[F2] OnHoldButton
corner_radius	7
border_width	0
border_color	#000000
Button - Interact	
interactable	<input checked="" type="checkbox"/> true
panel_id	[None]
hotkey	F2
on_press_macro	[1-LIN] OnHoldButtonPressed
on_release_macro	[1-LIN] OnHoldButtonReleased

Position and size

Text font, size, color and alignment

Interact Section
With 2 linkages with macros for button press and release

To allow simulation of mouse clicks on device display the Focus & Select mode was introduced.

If a panel with at least one interactable widget with Hotkey assignment is displayed, Focus & Select mode can be activated by either

- Long press on Menu Button or
- Short press on Menu button and selection of first entry Focus & Select by Enter (F3) key.

In Focus & Select mode the F1 ...F6 keys are switched to their secondary functions (green labeling)

F2/F4,F5 and F6 are used to focus the next interactable element on screen.

F3 key will trigger the element like open a value editor or execute a macro.

Focus & Select mode is finished by ESC(F1) key.

While Focus&Select mode is activated, the key actions are not forwarded to SDF for processing (system signals SYSDIGIN1—6, digital input events etc.)

Checkbox value is mainly intended to display or input binary values.

Back

Unchecked

Referenced signal: 0

Styled Checkbox

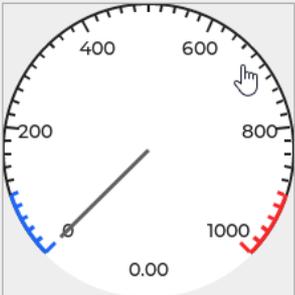
Properties:

Widget	
name	checkbox
x	49
y	130
width	94
height	22
visible	<input checked="" type="checkbox"/> true
z_index	3
opacity	100%
Text properties	
font	Default
font_size	16
text_color	#000000
text_alignment	Default
text_wrap_mode	Clip
Checkbox	
signal	CheckboxSignal ⊞
checked_text	Checked ...
unchecked_text	Unchecked ...
checked_range_list	1-65535
check_set_value	1
uncheck_set_value	0
Checkbox - Interact	
interactable	<input checked="" type="checkbox"/> true
hotkey	None
Checkbox - Style	
unchecked_bg_color	#FFFFFF
checked_bg_color	#FFFFFF
tick_color	#00957F
border_color	#000000

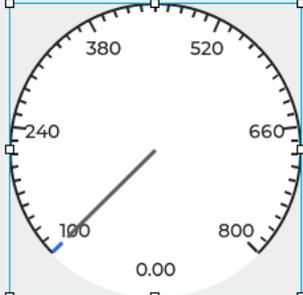
Meter widget allows to show a signal value in a meter representation.
Some new properties had been added to this widget.

Back
[F1] Start Signals
[F2] Stop Signals

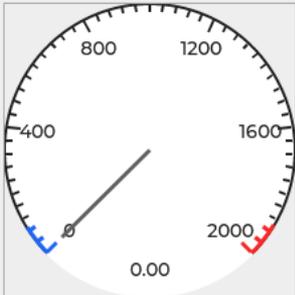
Default Meter:



Offset Meter (+100):



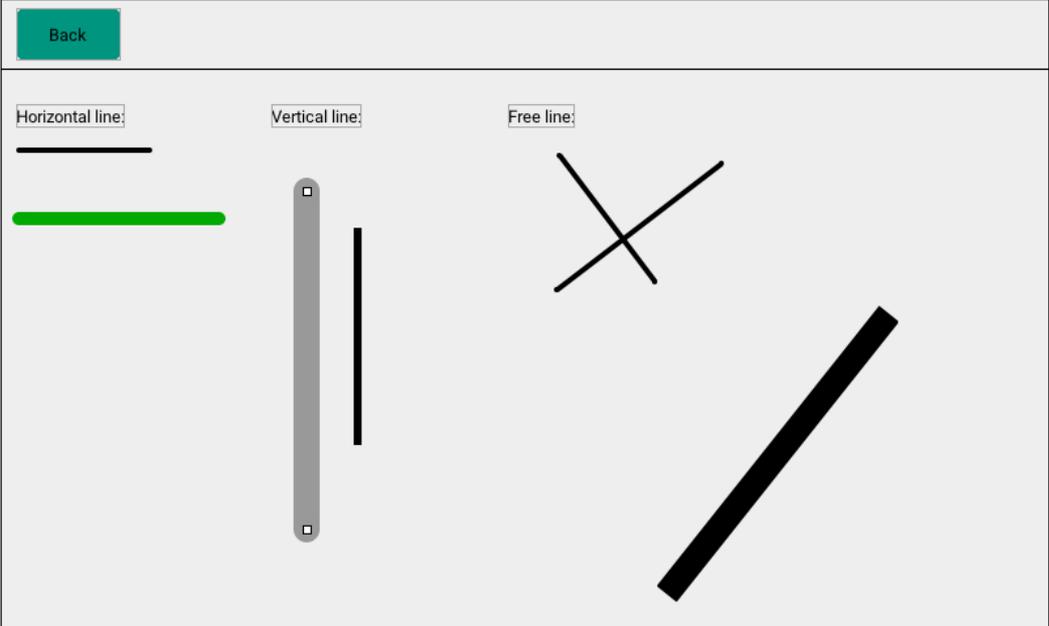
Factor Meter (x2):



Properties:

Widget	
name	signal_meter
x	289
y	115
width	220
height	220
visible	<input checked="" type="checkbox"/> true
z_index	7
opacity	100%
SignalMeter	
signal	MeterTestSignal C
signal_representation	Integer ...
unit	...
scale_start	100
scale_end	800
lower_limit	100
upper_limit	900
factor	1.000000
offset	100.000000
SignalMeter - Interact	
interactable	<input type="checkbox"/> false
hotkey	None
dialog_title	
live_update	<input type="checkbox"/> false
live_update_step	1
allowed_input_ranges	

Line widget can be used to decorate the GUI and to separate areas on the panel.
There is no runtime function or SDF action related to this widget.



Properties:

Widget	
name	line
x	223
y	135
visible	<input checked="" type="checkbox"/> true
z_index	9
opacity	100%
Line	
orientation	Vertical
start_x	233
start_y	145
end_x	233
end_y	402
line_width	20
line_color	#999999
line_rounded	<input checked="" type="checkbox"/> true

Frame widget can be used to decorate the GUI and to separate areas on the panel.
There is no runtime function or SDF action related to this widget.

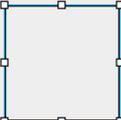
Back

Default

Border

Rounded

Circle











Infront Frame

Behind Frame

Properties:

Widget	
name	frame
x	185
y	111
width	100
height	100
visible	<input checked="" type="checkbox"/> true
z_index	3
opacity	100%
bg_color	#FFFFFFF
bg_opacity	0%
Frame	
border_width	2
border_color	#000000
border_radius	0

Control GUI elements at runtime

There are several ways to change the GUI in the context of SDF execution:

Signal, Meter and checkbox widgets

These widgets typically have a dynamic link to the SDF, and display the current signal value

Signal and Button Widget

with `interactable = true` will allow signal change or macro execution at runtime

Console Widget

shows the the macroprint output on the debug report channel in a scrollable list window.

Widget Properties changes

All properties of a widget can be changed at runtime: text, text colour, visibility, etc.

MessageBox

This mechanism allows a message to be transferred from the SDF to the GUI.

Furthermore a user feedback is optionally available.

Widget Property changes

There is a special syntax for the Macro Print command to change the widget properties

[GUI]

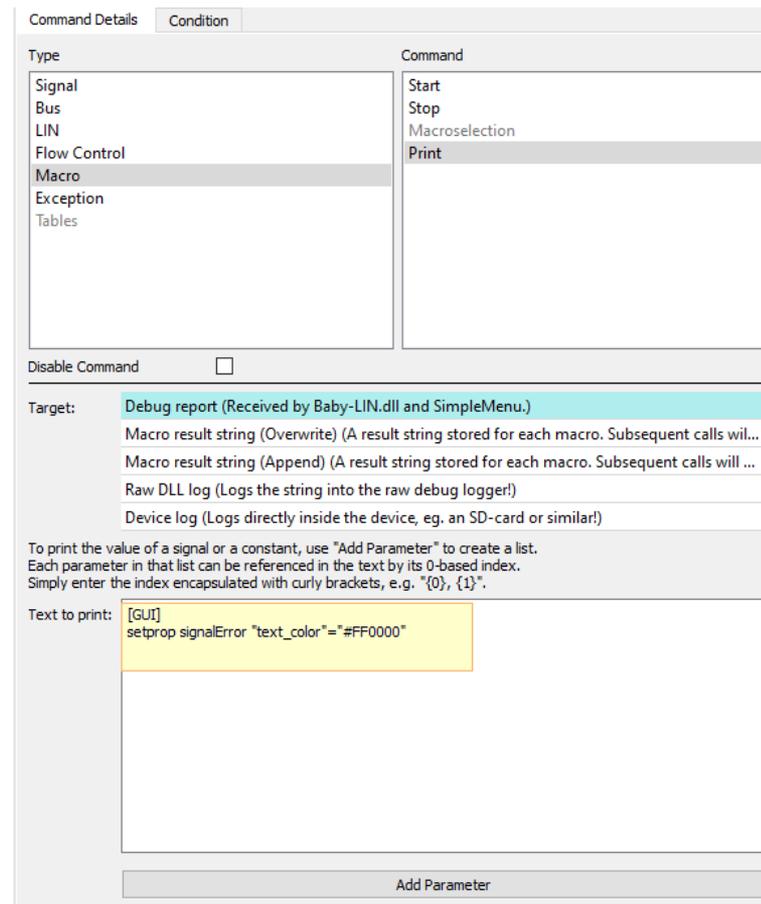
```
setprop signalError "text_color"="#FF0000"
```

[GUI] and setprop are reserved keywords.

signalError is the widget name here. (Case sensitive!)

The widget name is followed by the name and new value of the property, both enclosed in quotation marks, and separated by an '=' character.

If the widget name contains blanks, it must also be enclosed in quotation marks.



Command Details Condition

Type

- Signal
- Bus
- LIN
- Flow Control
- Macro
- Exception
- Tables

Command

- Start
- Stop
- Macroselection
- Print

Disable Command

Target:

- Debug report (Received by Baby-LIN.dll and SimpleMenu.)
- Macro result string (Overwrite) (A result string stored for each macro. Subsequent calls will ...)
- Macro result string (Append) (A result string stored for each macro. Subsequent calls will ...)
- Raw DLL log (Logs the string into the raw debug logger!)
- Device log (Logs directly inside the device, eg. an SD-card or similar!)

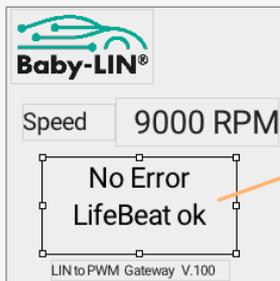
To print the value of a signal or a constant, use "Add Parameter" to create a list. Each parameter in that list can be referenced in the text by its 0-based index. Simply enter the index encapsulated with curly brackets, e.g. "{0}", "{1}".

Text to print:

```
[GUI]
setprop signalError "text_color"="#FF0000"
```

Add Parameter

Properties Change Beispiel:
 [GUI]
 setprop signalError "text_color"="#FF0000"



Properties:

Widget	
name	signalError
x	30
y	125
width	164
height	82
visible	<input checked="" type="checkbox"/> true
z_index	1
opacity	255
Font	
text_font	Default
text_size	28
text_color	... #000000
text_align	Center
text_wrap	Wrap
Signal	
signal	... [1] HELLA_LIN2PWM_co
default_text	... No Error\nLifeBeat ok
display_format	... {VALUE}
editable	<input type="checkbox"/> false
signal_representation	... Enum

More examples

[GUI]
 setprop signalError "visible" = "0"

Or
 [GUI]
 setprop signalError "visible" = "1"
 setprop signalError "text_color" = "#00FF00"

Multiple property changes can be in includes in the same Macro Print command .

For screens with multiple panels, there are 2 methods of switching between panels:

- Via button widget with assigned panel ID.
On device you have to select Focus & Select modus first to trigger button in display. Alternatively, you can assign a hotkey on the Baby-LIN keyboard to the button.

- Via MacroPrint command from within the SDF

[GUI]

setprop Panel1 "active" = "1"

or

[GUI]

setprop Panel2 "active" = "1"

It is sufficient to set only the desired panel to active = 1, all others are automatically deactivated.



Properties:	
Widget	
name	buttonSwitchToPanel2
x	602
y	27
width	122
height	40
visible	<input checked="" type="checkbox"/> true
z_index	8
opacity	100%
Text properties	
font	Default
font_size	16
text_color	#FFFFFF
text_alignment	Center
text_wrap_mode	Wrap
Button	
background	#00957F
text	PanelSwitch per Button ...
corner_radius	7
border_width	0
border_color	#000000
Button - Interact	
interactable	<input checked="" type="checkbox"/> true
panel_id	Panel2
hotkey	None
on_press_macro	[None]
on_release_macro	[None]

SessionConf v2.36.1 - [C:/documents/presentations/Gui-Editor-Webinar/GUI_Demo_PanelSwitchFromSDF.sdf]

File Edit View Tools Help

Hide expert settings Required SDF version: v3.24 FID: 0x0 PID: Hex 0x80

SDF Version 3

Macro number 0

Name PanelSwitch

Parameter count 0

Comment

Label	Condition	Command	Comment
0		CurrentPanel = CurrentPanel + 1	
1	If Signal CurrentPanel > 2	CurrentPanel = 1 (0x01)	
2		Print on Debug report: "[GUI] setprop Panel(0) "active"="1" ", Parameter: [0] = CurrentPanel	

Section properties

- Bus description
- Emulation
- Tables
- Virtual signals
- Signalfunctions
- Protocols
- GUI-Elements (SimpleMenu/HARP etc)
- Macros
 - [0] PanelSwitch
 - Macroselection
 - Events
 - Device-specific options
 - GUI Definition

Command Details

Type

- Signal
- Bus
- LIN
- Flow Control
- Macro
- Exception
- Tables

Command

- Start
- Stop
- Macroselection
- Print

Disable Command

Target:

- Debug report (Received by Baby-LIN.dll and SimpleMenu.)
- Macro result string (Overwrite) (A result string stored for each macro. ...)
- Macro result string (Append) (A result string stored for each macro. ...)
- Raw DLL log (Logs the string into the raw debug logger!)
- Device log (Logs directly inside the device, eg. an SD-card or similar!)

To print the value of a signal or a constant, use "Add Parameter" to create a list. Each parameter in that list can be referenced in the text by its 0-based index. Simply enter the index encapsulated with curly brackets, e.g. "{0}", "{1}".

Text to print:

[GUI]
setprop Panel(0) "active"="1"

Delete Parameter 0 Signal

Signal CurrentPanel

See also
GUI_Demo_panelSwitch.sdf

Messagebox Function

For displays from the SDF in the sense of a message box, there is another special syntax for the Macro Print command

[GUI]

```
msgbox open "Das ist der Anzeigetext!" 1:23 "Select Yes" "Select No"
```

1:23 defines the section and signal index for a feedback signal, where the index (1..n) of the pressed selection is stored and can be evaluated in the SDF.

The two text strings "Select Yes" and "Select No" are the labels of the two buttons.

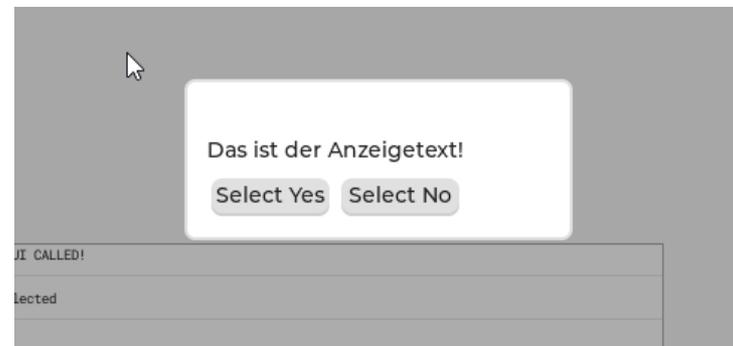
You can currently define up to 10 buttons.

Target:

- Debug report (Received by Baby-LIN.dll and SimpleMenu.)
- Macro result string (Overwrite) (A result string stored for each macro. Subsequent...
- Macro result string (Append) (A result string stored for each macro. Subsequent ...
- Raw DLL log (Logs the string into the raw debug logger!)
- Device log (Logs directly inside the device, eg. an SD-card or similar!)

To print the value of a signal or a constant, use "Add Parameter" to create a list. Each parameter in that list can be referenced in the text by its 0-based index. Simply enter the index encapsulated with curly brackets, e.g. "{0}", "{1}".

Text to print: [GUI]
msgbox open "Das ist der Anzeigetext!" 1:23 "Select Yes" "Select No"



The screenshot shows a white message box with a grey border. The text inside reads "Das ist der Anzeigetext!". Below the text are two buttons: "Select Yes" and "Select No". The buttons are rounded rectangles with a light grey background and dark grey text. The message box is centered on a dark grey background. Below the message box, there is a grey bar with the text "JI CALLED!" and "lected".

Variants of Messagebox syntax

Open Message Box:

[GUI]

msgbox open "Message Box will disappear in 5s"

Update Message Box text:

[GUI]

msgbox update "Message Box will be closed after {0}s"

Close Message Box:

[GUI]

msgbox close

Macro number	2		
Name	Msgbox Update		
Parameter count	0		
Comment			
	Label	Condition	Command
0			Print on Debug report: " [GUI] msgbox open "Message Box closes in 5s" "
1			popupcloseTimer = 5 (0x05)
2	loop		Delay 1000ms
3			popupcloseTimer = popupcloseTimer - 1
4			Print on Debug report: " [GUI] msgbox update " Message Box closes in {0}s" ", Parameter: {0} = popupcloseTimer
5		If Signal popupcloseTimer > 0	Jump to "loop"
6			Print on Debug report: " [GUI] msgbox close "

- Allow picture resizing when positioning on panel
- Support for additional image formats
- Optimization of image storage
- Signal plotter
- Your ideas and requests