2015

SGUI Version 1.3

Quellcode "C" Librarys und Beispiele für ATMega und STM32F4 CPUs

Simple GUI-Library (STM32F429) von Uwe Becker

Diese Library ermöglicht es eine “einfache” Graphische Benutzeroberfläche mit dem STM32F429 und einem Grafik-Display (mit Touch) zu realisieren.

- Michael Steinsträßer DD4MS
- Uwe Becker

- Quellcode "C" Librarys und Beispiele für ATMega und STM32F4 CPUs
- 16.06.2015

1 Inhaltsverzeichnis

SGUI (Version : 1.3)

2 Projekt: SGUI

```
  SGUI_PictureSetHandler(picture, pic_fkt);
}
```

2.1 Beschreibung :

Die *"simple-GUI"* ermöglicht es eine *"einfache"* grafische Benutzeroberfläche zu erstellen.

Es können verschiedene *"Fenster"* angelegt werden auf denen unterschiedliche *"Objekte"* platziert werden können.

Einige Objekte dienen zur Darstellung von Zahlen & Texten andere Objekte können vom User per Touch bedient werden.

Die Anordnung, das Aussehen und das Verhalten der Windows und Objekte muss der User zur Laufzeit vom Programm festlegen.

Für Objekte die per Touch betätigt werden, können Call-Back Funktionen angelegt werden, die dann automatisch aufgerufen werden wenn ein Event (z.B. Button-Click) auftritt.

Die SGUI übernimmt die Aufgaben den Touch abzufragen und die Windows &Objekte entsprechend zu zeichnen.

2.2 Funktionen:

```
void SGUI_Init(void)
```

- init-Funktion für die SGUI
- muss zum start einmal aufgerufen werden
- löscht das komplette Display mit einer Farbe

```
void SGUI_Do(void)
```

- Arbeits-Funktion der SGUI
- muss vom User-Programm aus zyklisch aufgerufen werden

```
void SGUI_ReDraw(void)
```

- zeichnet das gerade aktive Fenster nochmal auf den Screen
- dient im Moment nur zu Debug-Zwecken

```
void SGUI_Pause_ms(uint32_t delay)
```

- erzeugt eine Pause vom *"delay"* ms

```
uint32_t SGUI_GetUsedRam(void)
```

- übergibt den aktuellen Speicherplatzverbrauch im RAM (in Bytes)
- jedes Objekt reserviert beim *"Create"* eine bestimmte Anzahl an Bytes

2.3 Minimal-Beispiel:

Initialisiert die SGUI und ruft die Do-Funktion zyklisch auf

```
#include "stm32_ub_sgui.h"

void main(void) {
  SGUI_Init();

  while(1) {
    SGUI_Do();
  }
}
```

2.3.1 Beispiel-1:

erzeugt ein Window mit einem Button darauf (der Button löst keine Funktion aus)

```
#include "stm32_ub_sgui.h"

void main(void) {
  SGUI_Init();

  SGUI_WindowCreateMain(1);
  SGUI_ButtonCreate(10,10,100,50);

  while(1) {
    SGUI_Do();
  }
}
```

3 Objekt: Window

Abbildung 1 SGUI

3.1 Beschreibung:

Windows dienen als *"Hintergrund"* für alle anderen Objekte (Ausnahme andere Windows)

Es gibt zwei verschiedene Typen:

1. = Main-Window (feste Größe von 320 x 240 Pixel)
2. = Child-Window (einstellbare Größe < 320 x 240 Pixel)

Jedes Window besitzt eine eindeutige *"window_nr"* die beim *"create"* mit angegeben werden muss.
Mit dieser Nummer kann das Window per *"show"* angezeigt werden.

Es kann immer nur ein Window zu einer Zeit *"aktiv"* sein. (per *"show"* wird ein Window auf dem LCD angezeigt)

Ein Child-Window liegt (da es kleiner als ein Main-Window ist) im Vordergrund und verdeckt das Main-Window nur zum Teil.

Jedes andere Objekt das per *"create"* auf einem Window platziert wird, übernimmt die aktuelle *"window_nr"* in seinen Parametersatz.

Aus diesem Grund muss zumindest ein Window per *"create"* erstellt worden sein, bevor ein anderes Objekt per *"create"* erstellt wird.

Beim Darstellen von einem Window auf dem LCD werden zusätzlich alle Objekte mit der gleichen *"window_nr"* gezeichnet und nur diese sind per Touch steuerbar.

Das letzte Window das per *"create"* erstellt wurde, ist das gerade aktive Window und wird auf dem LCD dargestellt. (Per *"show"* kann jedes andere Window angezeigt werden)

3.2 Funktionen:

```
SWINDOW_t* SGUI_WindowCreateMain(uint16_t window_nr)
```

- erzeugt ein Main-Window
- *"window_nr"* = Eindeutige Nummer vom Window

```
SWINDOW_t* SGUI_WindowCreateChild(uint16_t window_nr, uint16_t x, uint16_t
y, uint16_t w, uint16_t h)
```

- erzeugt ein Child-Window
- *window_nr* = Eindeutige Nummer vom Window
- *x, y* = Position vom Window auf dem Screen
- *w, h* = Breite und Höhe vom Window

```
void SGUI_WindowSetStyle(SWINDOW_t* ptr, STYLE_TYP_t style)
```

- stellt bei einem Child-Window den Style vom Rahmen ein
- *style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED*

```
void SGUI_WindowSetFrameSize(SWINDOW_t* ptr, uint8_t px)
```

- stellt bei einem Child-Window die dicke vom Rahmen ein (wenn *STYLE = STYLE_FLAT*)

```
void SGUI_WindowSetColor(SWINDOW_t* ptr, uint16_t c1, uint16_t c2)
```

- stellt die Farben vom Window ein
- c1 = Rahmenfarbe (nur bei Child-Window und wenn *STYLE = STYLE_FLAT*)
- c2 = Hintergrundfarbe

```
void SGUI_WindowShow(uint16_t window_nr)
```

- zeigt ein Window (und alle zugehörenden Objekte) auf dem LCD an
- *window_nr* = Nummer vom Window das angezeigt werden soll
- falls Window-Typ = MAIN das Window füllt den kompletten Screen
- falls Window-Typ = CHILD das letzte Main-Window ist noch zum Teil sichtbar

```
void SGUI_WindowShowPrev(void)
```

- zeigt das zuletzt angezeigt Window an
- diese Funktion ist bei Child-Windows nützlich, wenn mit einen "OK" oder "CANCEL" Button das darunter liegende Main-Window wieder aktiviert werden soll.

```
uint16_t SGUI_WindowGetActivNr(void)
```

- übergibt die *„window_nr“* vom geraden aktiven Window

3.3 Minimal-Beispiel:

Erzeugen von einem Main-Window

```
void foo(void) {
  SGUI_WindowCreateMain(1);          // NR=1
}
```

3.3.1 Beispiel-1:

Farbe vom Window einstellen
(Die Rahmenfarbe bei einem Main-Window ist egal**)**

```
void foo(void) {
  SWINDOW_t *win;

  win=SGUI_WindowCreateMain(1);                 // NR=1
  SGUI_WindowSetColor(win, 0, RGB_COL_RED);     // col=Rot
}
```

3.3.2 Beispiel-2:

Zwei Windows **(**Nr.1 und Nr.2**)** erstellen und Window Nr.1 anzeigen

```
void foo(void) {
  SWINDOW_t *win;

  win=SGUI_WindowCreateMain(1);                         // NR=1
  SGUI_WindowSetColor(win, 0, RGB_COL_RED);             // col=Rot
  win=SGUI_WindowCreateMain(2);                         // NR=2
  SGUI_WindowSetColor(win, 0, RGB_COL_BLUE);            // col=Blau

  SGUI_WindowShow(1);                                   // NR 1 anzeigen
}
```

3.3.3 Beispiel-3:

Ein Child-Window erstellen mit der *„window_nr“* **=** 10

```
void foo(void) {
  SGUI_WindowCreateChild(10, 5, 20, 100, 200);
                                              // NR=10,x=5,y=20,w=100,h=200
}
```

3.3.4 Beispiel-4:

Farbe**,** Rahmen und Style vom Child-Window einstellen

```
void foo(void) {
  SWINDOW_t *win;

  win=SGUI_WindowCreateChild(10,5,20,100,200);
                                              // NR=10,x=5,y=20,w=100,h=200
  SGUI_WindowSetColor(win, RGB_COL_GREEN, RGB_COL_YELLOW);
                                                    // col = Grün + Gelb
  SGUI_WindowSetFrameSize(win,5);                   // 5 Pixel Rahmendicke
  SGUI_WindowSetStyle(STYLE_FLAT);                  // kein 3D-Rahmen
}
```

3.3.5 Beispiel-5:

Abfragen welche Window-Nr gerade aktiv ist

```
void foo(void) {
  uint16_t akt_win_nr;

  akt_win_nr=SGUI_WindowGetActivNr();
}
```

4 Objekt: Text

Abbildung 2 Objekt Text

4.1 Beschreibung:

Texte dienen zum Darstellen von Strings, Integer- oder Float-Zahlen auf einem Window.

Es gibt zwei verschiedene Arten Texte darzustellen

1. Per *"Print"* wird ein Text direkt auf das gerade sichtbare Window gezeichnet. Der Text wird aber nicht gespeichert und verschwindet somit wieder, wenn ein anderes Window den Text überzeichnet.

2. Per *"Create"* wird ein Text-Objekt auf dem gerade aktiven Window abgelegt. Der Text ist damit fest mit dem Window verbunden und wird auch nach dem überzeichnen von einem anderen Window wieder dargestellt.

Die Texte die per *"Create"* angezeigt werden können auch nachträglich noch verändert werden (*Farbe, Font*)

Der Text wird an der aktuellen Cursor Position angezeigt. Beim Child-Windows bezieht sich die Cursor Position auf die Position vom Child-Window. Beim Schreiben vom Text wird die Cursorposition aktualisiert.

Beim Schreiben wird der *„Default-Font“* und die *„Default-Farbe“* benutzt.

4.2 Funktionen:

```
void SGUI_TextSetCursor(uint16_t x, uint16_t y)
```

- setzt den Cursor an *x, y Position* vom Window

```
void SGUI_TextCursorLinefeed(void)
```

- setzt den Cursor an den Anfang der nächsten Zeile

```
void SGUI_TextSetDefFont(UB_Font *font)
```

- stellt die *Default-Schriftart* ein

```
void SGUI_TextSetDefColor(uint16_t c1, uint16_t c2)
```

- stellt die *Default-Farben* ein
- c1 = Textfarbe
- c2 = Hintergrundfarbe

```
void SGUI_TextPrintString(char *txt_ptr)
```

- schreibt einen String an die aktuelle Cursorposition
- verschiebt danach den Cursor ans Ende vom String

```
void SGUI_TextPrintInt(int32_t value, uint8_t digits, bool padding)
```

- schreibt eine Integer-Zahl an die aktuelle Cursorposition
- verschiebt danach den Cursor ans Ende der Zahl
- value = Wert der Zahl
- digits = Anzahl der Ziffern der Zahl
- padding = *true*, wenn führende Nullen angezeigt werden sollen

```
void SGUI_TextPrintFloat(float value, uint8_t digits, bool padding)
```

- schreibt eine Float-Zahl an die aktuelle Cursorposition
- verschiebt danach den Cursor ans Ende der Zahl
- value = Wert der Zahl
- digits = Anzahl der Ziffern der Zahl
- padding = *true*, wenn führende Nullen angezeigt werden sollen

```
STEXT_t* SGUI_TextCreateString(char *txt_ptr)
```

- erzeugt ein String-Objekt an der aktuellen Cursorposition
- verschiebt danach den Cursor ans Ende vom String

```
void SGUI_TextSetString(STEXT_t* ptr, char *txt_ptr)
```

- ändert den Text von einem String-Objekt

```
STEXT_t* SGUI_TextCreateInt(int32_t value, uint8_t digits, bool padding)
```

- erzeugt ein Integer-Zahlen-Objekt an der aktuellen Cursorposition
- verschiebt danach den Cursor ans Ende der Zahl
- value = Wert der Zahl
- digits = Anzahl der Ziffern der Zahl
- padding = *true*, wenn führende Nullen angezeigt werden sollen

```
void SGUI_TextSetInt(STEXT_t* ptr, int32_t value)
```

- ändert den Integer-Zahlenwert von einem Integer-Zahlen-Objekt
- value = Wert der Zahl

```
STEXT_t* SGUI_TextCreateFloat(float value, uint8_t digits, bool padding)
```

- erzeugt ein Float-Zahlen-Objekt an der aktuellen Cursorposition
- verschiebt danach den Cursor ans Ende der Zahl
- value = Wert der Zahl
- digits = Anzahl der Ziffern der Zahl

padding = *true*, wenn führende Nullen angezeigt werden sollen

```
void SGUI_TextSetFloat(STEXT_t* ptr, float value)
```

- ändert den Float-Zahlenwert von einem Float-Zahlen-Objekt
- value = Wert der Zahl

```
void SGUI_TextSetColor(STEXT_t* ptr, uint16_t c1, uint16_t c2)
```

- stellt die Farben von einem Text-Objekt ein
- *c1* = Farbe vom Font
- *c2* = Farbe vom Hintergrund

```
void SGUI_TextSetFont(STEXT_t* ptr, UB_Font* font)
```

- stellt die Schriftart von einem Text-Objekt ein

4.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einem Text

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_TextCreateString("Hallo");              // Anzeige = "Hallo"
}
```

4.3.1 Beispiel-1:

Text an einer bestimmten Position anzeigen

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_TextSetCursor(10,20);                   // x=10, y=20
  SGUI_TextCreateString("Hallo");              // Anzeige="Hallo"
}
```

4.3.2 Beispiel-2:

Farbe und Font einstellen

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_TextSetDefFont(&Arial_10x15);           // Font=Arial_10x15
  SGUI_TextSetDefColor(RGB_COL_RED, REG_COL_BLUE);  // col=red + blue
  SGUI_TextCreateString("Hallo");              // Anzeige="Hallo"
}
```

4.3.3 Beispiel-3:

Texte und Zahlen darstellen

```
void foo(void) {
  int a=12;

  SGUI_WindowCreateMain(1);                // NR=1
  SGUI_TextCreateString("Messwert=");
  SGUI_TextCreateInt(a, 4, true);          // mit 4 stellen mit führende Null
  SGUI_TextCreateString("mV");             // Anzeige= "Messwert=0012mV"
}
```

4.3.4 Beispiel-4:

Wert von einer Zahl ändern

```
void foo(void) {
  int a=12;
  STEXT_t *txt;

  SGUI_WindowCreateMain(1);                // NR=1
  SGUI_TextCreateString("Messwert=");
  txt=SGUI_TextCreateInt(a, 4, true);      // mit 4 stellen mit führende Null
  SGUI_TextCreateString("mV");             // Anzeige = "Messwert=0012mV"

  a+=13;
  SGUI_TextSetInt(txt, a);                 // neuen Zahlenwert darstellen
                                           // Anzeige = "Messwert=0025mV"
}
```

5 Objekt: Panel

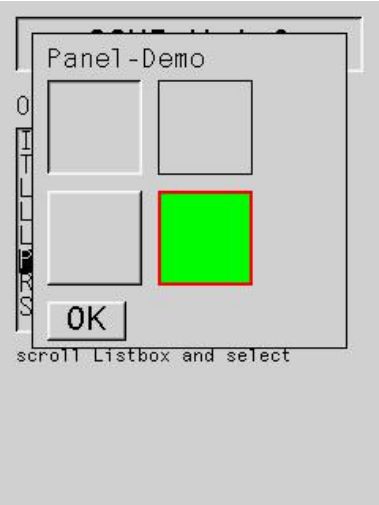

Abbildung 3 Objekt Panel

5.1 Beschreibung:

Panels sind nur grafische Objekte und haben keine eigene Funktion. Panels sind rechteckige Flächen deren Farbe eingestellt werden kann. Panels werden beim Zeichnen von einem Window als erstes dargestellt und liegen somit hinter allen anderen Objekten.

5.2 Funktionen

```
SPANEL_t* SGUI_PanelCreate(uint16_t x, uint16_t y, uint16_t w, uint16_t h)
```

- erstellt ein Panel auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *x, h* = breite und höhe vom Panel

```
void SGUI_PanelSetStyle(SPANEL_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- *style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED*

```
void SGUI_PanelSetFrameSize(SPANEL_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn *STYLE = STYLE_FLAT*)

```
void SGUI_PanelSetColor(SPANEL_t* ptr, uint16_t c1, uint16_t c2)
```

- stellt die Farbe von einem Panel ein
- *c1* = Rahmenfarbe (wenn *STYLE = STYLE_FLAT*)
- *c2* = Hintergrundfarbe

5.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einem Panel

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_PanelCreate(10, 20, 100, 150);          // x=10,y=20,w=100,h=150
}
```

5.3.1 Beispiel-1:

Farbe vom Panel einstellen

```
void foo(void) {
  SPANEL_t *panel;

  SGUI_WindowCreateMain(1);                    // NR=1
  panel =SGUI_PanelCreate(10, 20, 100, 150);   // x=10,y=20,w=100,h=150
  SGUI_PanelSetColor(panel, 0, RGB_COL_RED);   // col=Rot
}
```

6 Objekt: Button

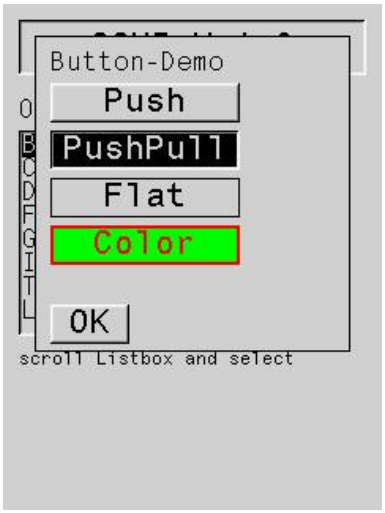

Abbildung 4 Objekt Button

6.1 Beschreibung:

Buttons können vom User per Touch betätigt werden.
Buttons sind Rechtecke die zwei Zustände annehmen können (*aktiv oder inaktiv*)
Es gibt zwei Arten von Buttons:

- PUSH Taster
- PUSHPULL Schalter

Ein Taster ist unbetätigt *"inaktiv"* und wechselt auf *"aktiv"* wenn er per Touch gedrückt wird.

Ein Schalter wechselt seinen aktuellen Zustand jedes Mal, wenn er per Touch betätigt wird.

Der Zustand von einem Button kann entweder *„gepollt“* werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn der Button per Touch betätigt wird.

6.2 Funktionen:

```
SBUTTON_t* SGUI_ButtonCreate(uint16_t x, uint16_t y, uint16_t w, uint16_t h)
```

- erstellt einen Button auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *x, h* = breite und höhe vom Button

```
void SGUI_ButtonSetMode(SBUTTON_t* ptr, SBUTTON_MODE_t mode)
```

- stellt den Mode von einem Button ein
- *mode = SBUTTON_PUSH, SBUTTON_PUSHPULL*

```
void SGUI_ButtonSetStyle(SBUTTON_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- *style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED (STYLE_RAISED oder STYLE_LOWERED sind identisch)*

```
void SGUI_ButtonSetFrameSize(SBUTTON_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn *STYLE = STYLE_FLAT*)

```
void SGUI_ButtonSetColor(SBUTTON_t* ptr, uint16_t c1, uint16_t c2)
```

- stellt die Farben vom Button ein
- c1 = Textfarbe
- c2 = Hintergrundfarbe

```
void SGUI_ButtonSetText(SBUTTON_t* ptr, char* txt)
```

- stellt den Text auf dem Button ein

```
void SGUI_ButtonSetFont(SBUTTON_t* ptr, UB_Font* font)
```

- stellt die Schriftart vom Button ein

```
void SGUI_ButtonSetAktiv(SBUTTON_t* ptr, bool aktiv)
```

- stellt den aktuellen Status vom Button ein

```
void SGUI_ButtonSetHandler(SBUTTON_t* ptr, void *fkt_ptr);
```

- stellt eine „CallBack-Funktion“für den Button ein

```
bool SGUI_ButtonIsAktiv(SBUTTON_t* ptr)
```

- abfrage vom aktuellen Status vom Button

6.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einem Button

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_ButtonCreate(10,20,100,150);            // x=10, y=20, w=100, h=150
}
```

6.3.1 Beispiel-1:

Text von einem Button einstellen

```
void foo(void) {
  SBUTTON_t *btn;

  SGUI_WindowCreateMain(1);                    // NR=1
  btn=SGUI_ButtonCreate(10, 20, 100, 150);     // x=10, y=20, w=100, h=150
  SGUI_ButtonSetText(btn, "Hallo");            // Text="Hallo"
}
```

6.3.2 Beispiel-2:

Mode von einem Button auf PushPull einstellen

```
void foo(void) {
  SBUTTON_t *btn;

  SGUI_WindowCreateMain(1);                    // NR=1
  btn=SGUI_ButtonCreate(10, 20, 100, 150);     // x=10, y=20, w=100, h=150
  SGUI_ButtonSetMode(btn, SBUTTON_PUSHPULL);   // Mode=PushPull
}
```

6.3.3 Beispiel-3:

Status von einem Button abfragen

```
void foo(void) {
  SBUTTON_t *btn;

  SGUI_WindowCreateMain(1);                    // NR=1
  btn=SGUI_ButtonCreate(10, 20, 100, 150);     // x=10, y=20, w=100, h=150


  if(SGUI_ButtonIsAktiv(btn)==true) {
                                               // Button ist betätigt
  }
}
```

6.3.4 Beispiel-4:

CallBack-Funktion einrichten

```
SBUTTON_t *btn;

void btn_fkt(bool aktiv) {
                                        // CallBack-Funktion für User wird
                                        // bei Touch Betätigung aufgerufen
}

void foo(void) {
  SGUI_WindowCreateMain(1);             // NR=1
  btn=SGUI_ButtonCreate(10,20,100,150); // x=10, y=20, w=100, h=150
  SGUI_ButtonSetHandler(btn, btn_fkt);
}
```

7 Objekt: LED

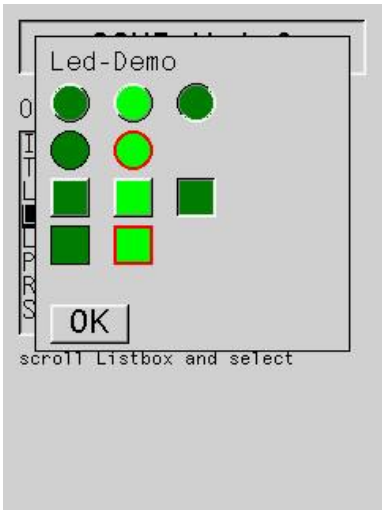

Abbildung 5 Objekt LED-Demo

7.1 Beschreibung:

LEDs dienen als Anzeigeelement.

LEDs können zwei Zustände annehmen (aktiv oder inaktiv)

Die zwei Zustände werden durch zwei verschiedene Farben dargestellt.

Eine LED kann rund oder Quadratisch sein.

7.2 Funktionen:

```
SLED_t* SGUI_LedCreate(uint16_t x, uint16_t y, uint16_t s)
```

- erstellt eine LED auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *s* = Breite und Höhe der LED

```
void SGUI_LedSetTyp(SLED_t* ptr, SLED_TYP_t typ)
```

- stellt den Typ der LED ein
- *typ = SLED_ROUND, SLED_RECT*

```
void SGUI_LedSetStyle(SLED_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- *style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED*

```
void SGUI_LedOn(SLED_t* ptr)
```

- stellt den aktuellen Status der LED auf *"aktiv"*

```
void SGUI_LedOff(SLED_t* ptr)
```

- stellt den aktuellen Status der LED auf *"inaktiv"*

```
void SGUI_LedToggle(SLED_t* ptr)
```

- toggelt den aktuellen Status der LED

```
void SGUI_LedSetAktiv(SLED_t* ptr, bool aktiv)
```

- stellt den aktuellen Status der LED ein

```
void SGUI_LedSetFrameSize(SLED_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn *STYLE = STYLE_FLAT*)

```
void SGUI_LedSetColor(SLED_t* ptr, uint16_t c1, uint16_t c2, uint16_t c3)
```

- stellt die Farben von einer LED ein
- c1 = Farbe vom Rahmen (wenn STYLE = STYLE_FLAT)
- c2 = Farbe für LED EIN
- c3 = Farbe für LED AUS

```
bool SGUI_LedIsAktiv(SLED_t* ptr)
```

- abfrage vom aktuellen Status der LED

7.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einer LED

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_LedCreate(10, 20, 25);                  // x=10,y=20,size=25
}
```

7.3.1 Beispiel-1:

Eine LED auf *"aktiv"* schalten

```
void foo(void) {
  SLED_t *led;

  SGUI_WindowCreateMain(1);                  // NR=1
  led=SGUI_LedCreate(10,20,25);              // x=10,y=20,size=25

  SGUI_LedOn(led);
}
```

7.3.2 Beispiel-2:

Farben einer LED einstellen

```
void foo(void) {
  SLED_t *led;

  SGUI_WindowCreateMain(1);                    //NR=1
  led=SGUI_LedCreate(10, 20, 25);              // x=10, y=20, size=25
  SGUI_PanelSetColor(led, RGB_COL_BLACK, RGB_COL_GREEN, RGB_COL_RED);
                                               //col=black+green+red
}
```

8 Objekt: LABEL

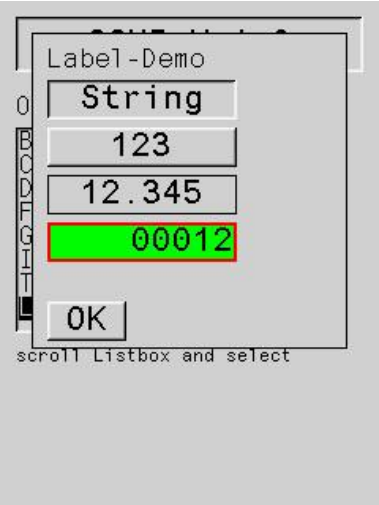

Abbildung 6 Objekt Label

8.1 Beschreibung:

LABEL dienen zur Beschriftung.
Es können Texte, Integer-Zahlen oder Float-Zahlen dargestellt werden.

Es gibt zwei Typen:

- PLAINTEXT : reiner Text
- PANELTEXT : Text mit umgebendem Panel

Beim Plaintext wird nur der Text dargestellt.
Beim Paneltext wird der Text auf einem Panel gezeichnet.
Die Ausrichtung vom Text zum Panel kann eingestellt werden (*Center, Left, Right*)

8.2 Funktionen:

```
SLABEL_t* SGUI_LabelCreate(uint16_t x, uint16_t y, uint16_t w, uint16_t h)
```

- erstellt ein LABEL auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *w, h* = breite und höhe vom Label (bzw. umgebenden Panel)

- void SGUI_LabelSetStyle(SLABEL_t* ptr, STYLE_TYP_t style)

- stellt den Style vom Rahmen ein (falls *Typ=SLABEL_PANELTEXT*)
- *style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED*

```
void SGUI_LabelSetPanelVisible(SLABEL_t* ptr, bool visible)
```

- Darstellung vom umgebenden Panel ein- oder ausschalten
- visible = *true, false*

```
void SGUI_LabelSetFrameSize(SLABEL_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn *STYLE = STYLE_FLAT*)

- void SGUI_LabelSetColor(SLABEL_t* ptr, uint16_t c1, uint16_t c2, uint16_t c3)

- stellt die Farben vom Label ein
- c1 = Textfarbe
- c2 = Rahmenfarbe
- c3 = Hintergrundfarbe

```
void SGUI_LabelSetText(SLABEL_t* ptr, char* txt)
```

- stellt einen String auf dem Label dar

```
void SGUI_LabelSetInt(SLABEL_t* ptr, int32_t value)
```

- stellt eine Integer-Zahl auf dem Label dar

```
void SGUI_LabelSetFloat(SLABEL_t* ptr, float value)
```

- stellt eine Float-Zahl auf dem Label dar

```
void SGUI_LabelSetNumFormat(SLABEL_t* ptr, uint8_t digits, bool padding)
```

- stellt das Format der Zahlen Anzeige ein
- digits = Anzahl der Ziffern der Zahl
- padding = *true*, wenn führende Nullen angezeigt werden sollen

```
void SGUI_LabelSetFont(SLABEL_t* ptr, UB_Font* font)
```

- stellt die Schriftart vom Label ein

```
void SGUI_LabelSetAlignment(SLABEL_t* ptr, SLABEL_ALIGN_t align)
```

- stellt die Ausrichtung vom Text ein
- *align = SLABEL_ALIGN_CENTER, SLABEL_ALIGN_LEFT, SLABEL_ALIGN_RIGHT*

8.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einem Label

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_LabelCreate(10, 20, 100, 30);           // x=10, y=20, w=100, h=30
}
```

8.3.1 Beispiel-1:

einem Text auf Label anzeigen

```
void foo(void) {
  SLABEL_t *label;

  SGUI_WindowCreateMain(1);                    // NR=1
  label=SGUI_LabelCreate(10,20,100,30);        // x=10, y=20, w=100, h=30
  SGUI_LabelSetDec(label, "Test");             // Anzeige="Test"
}
```

8.3.2 Beispiel-2:

Integer-Zahl auf einem Label anzeigen

```
void foo(void) {
  SLABEL_t *label;

  SGUI_WindowCreateMain(1);                    // NR=1
  label=SGUI_LabelCreate(10,20,100,30);        // x=10, y=20, w=100, h=30
  SGUI_LabelSetInt(label, -123);               // Anzeige="-123"
}
```

9 Objekt: Checkbox

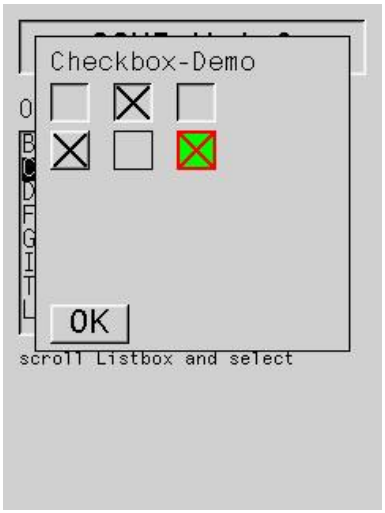

Abbildung 7 Objekt Checkbox

9.1 Beschreibung:

Checkboxen können vom User per Touch betätigt werden.

Checkboxen sind Quadrate die zwei Zustände annehmen können (aktiv oder inaktiv)

Eine Checkbox wechselt ihren aktuellen Zustand jedes Mal, wenn sie per Touch betätigt wird.

Der Zustand von einer Checkbox kann entweder *gepollt*[1] werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn die Checkbox per Touch betätigt wird.

9.2 Funktionen:

```
SCHECKBOX_t* SGUI_CheckboxCreate(uint16_t x, uint16_t y, uint16_t s)
```

- erstellt eine Checkbox auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *s* = Breite und Höhe der Checkbox

```
void SGUI_CheckboxSetStyle(SCHECKBOX_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED

```
void SGUI_CheckboxSetFrameSize(SCHECKBOX_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn STYLE = STYLE_FLAT)

[1] Wiederholt Abgefragt

```
void SGUI_CheckboxSetColor(SCHECKBOX_t* ptr, uint16_t c1, uint16_t c2)
```

- stellt die Farben der Checkbox ein
- c1 = Farbe vom *"checked"* Symbol
- c2 = Hintergrundfarbe

```
void SGUI_CheckboxSetAktiv(SCHECKBOX_t* ptr, bool aktiv)
```

- stellt den aktuellen Status der Checkbox ein

```
void SGUI_CheckboxSetHandler(SCHECKBOX_t* ptr, void *fkt_ptr)
```

- stellt eine „CallBack-Funktion“für die Checkbox ein

```
bool SGUI_CheckboxIsAktiv(SCHECKBOX_t* ptr)
```

- abfrage vom aktuellen Status der Checkbox

9.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einer Checkbox

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_CheckboxCreate(50,70,25);               // x=50,y=70,size=25
}
```

9.3.1 Beispiel-1:

Status von einer Checkbox abfragen

```
void foo(void) {
  SCHECKBOX_t *box;

  SGUI_WindowCreateMain(1);                    // NR=1
  box=SGUI_CheckboxCreate(50, 70, 25);         // x=50,y=70,size=25


  if(SGUI_CheckboxIsAktiv(box)==true) {
                                               // Checkbox ist aktiv
  }
}
```

9.3.2 Beispiel-2:

CallBack-Funktion einrichten

```
SCHECKBOX_t *box;

void box_fkt(bool aktiv) {
                                        // CallBack-Funktion für User wird
                                        // bei Touch Betätigung aufgerufen
}

void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  box=SGUI_CheckboxCreate(50, 70, 25);         // x=50,y=70,size=25
  SGUI_CheckboxSetHandler(box, box_fkt);
}
```

10 Objekt: Radiobutton

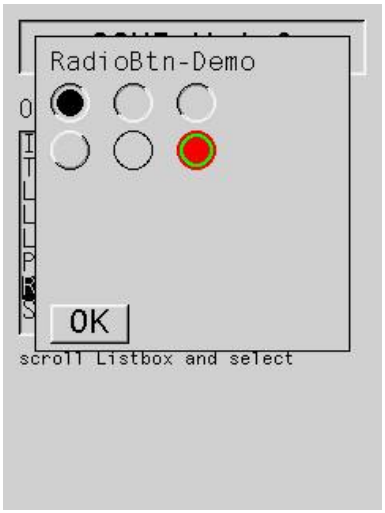

Abbildung 8 Objekt Radiobutton

10.1 Beschreibung:

Radiobuttons können vom User per Touch betätigt werden.

Radiobuttons sind rund und können zwei Zustände annehmen (aktiv oder inaktiv)

Jeder Radiobutton hat eine Gruppen-Nummer. Von allen Radiobuttons der gleichen Gruppe kann immer nur einer aktiv sein.

Beim aktivieren von einem Radiobutton, werden die anderen (der gleichen Gruppe) automatisch deaktiviert.

Der Zustand von einem Radiobutton kann entweder *„gepollt“* werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn der Button per Touch betätigt wird.

10.2 Funktionen:

```
SRBTN_t* SGUI_RadioButtonCreate(uint16_t x, uint16_t y, uint16_t s)
```

- erstellt einen Radiobutton auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *s* = Size vom Radiobutton

```
void SGUI_RadioButtonSetGroup(SRBTN_t* ptr, uint16_t group_nr)
```

- stellt die Gruppen-Nummer vom Radiobutton ein

```
void SGUI_RadioButtonSetStyle(SRBTN_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- *style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED*

```
void SGUI_RadioButtonSetFrameSize(SRBTN_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn *STYLE = STYLE_FLAT*)

```
void SGUI_RadioButtonSetColor(SRBTN_t* ptr, uint16_t c1, uint16_t c2)
```

- stellt die Farben vom Radiobutton ein
- *c1* = Farbe vom *"checked"* symbol
- *c2* = Hintergrundfarbe

```
void SGUI_RadioButtonSetAktiv(SRBTN_t* ptr)
```

- aktiviert den Radiobutton und deaktiviert alle anderen der gleichen Gruppe

```
void SGUI_RadioButtonSetHandler(SRBTN_t* ptr, void *fkt_ptr)
```

- stellt eine „CallBack-Funktion“für den Radiobutton ein

```
bool SGUI_RadioButtonIsAktiv(SRBTN_t* ptr)
```

- abfrage vom aktuellen Status vom Radiobutton

10.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einem Radiobutton

```
void foo(void) {
  SGUI_WindowCreateMain(1);                          // NR=1
  SGUI_RadioButtonCreate(50,70,25);                  // x=50,y=70,size=25
}
```

10.3.1 Beispiel-1:

Status von einem Radiobutton abfragen

```
void foo(void) {
  SRBTN_t *rbtn;

  SGUI_WindowCreateMain(1);                          // NR=1
  rbtn=SGUI_RadioButtonCreate(50, 70, 25);           // x=50,y=70,size=25


  if(SGUI_RadioButtonIsAktiv(rbtn)==true) {
                                                     // Radiobutton ist aktiv
  }
}
```

10.3.2 Beispiel-2:

CallBack-Funktion einrichten

```
SRBTN_t *rbtn;

void rbtn_fkt(bool aktiv) {
                                        // CallBack-Funktion für User wird
                                        // bei Touch Betätigung aufgerufen
}
void foo(void) {
  SGUI_WindowCreateMain(1);                          // NR=1
  rbtn=SGUI_RadioButtonCreate(50, 70, 25);           // x=50, y=70, size=25
  SGUI_RadioButtonSetHandler(rbtn, rbtn_fkt);
}
```

11 Objekt: Gauge

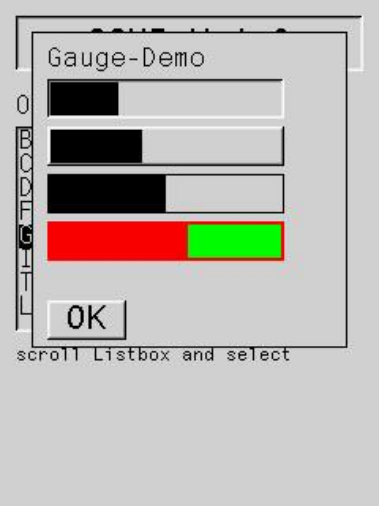

Abbildung 9 Objekt Gauge

11.1 Beschreibung:

Eine Gauge[2] (hier Fortschrittsbalken) stellt einen Zahlenwert als Balken-Grafik dar.
Der Balken ist Rechteckig und kann in vier Ausrichtungen auf einem Window platziert werden.

1. Horizontal (von links nach rechts auffüllend)
2. Horizontal (von rechts nach links auffüllend)
3. Vertikal (von unten nach oben auffüllend)
4. Vertikal (von oben nach unten auffüllend)

Der Balken besitzt einen Minimalwert, einen Maximalwert und einen Istwert als *Integer-Values*. Die Länge vom Balken wird entsprechend dieser 3 Parameter dargestellt.

Der *Maximalwert* muss größer als der *Minimalwert* sein und der *Istwert* muss zwischen Minimum und Maximum liegen.

[2] Anzeige Instrument

11.2 Funktionen:

```
SGAUGE_t* SGUI_GaugeCreate(uint16_t x, uint16_t y, uint16_t w, uint16_t h)
```

- erstellt eine Gauge auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *x, h* = Breite und Höhe der Gauge

```
void SGUI_GaugeSetTyp(SGAUGE_t* ptr, SGAUGE_TYP_t typ)
```

- stellt den Typ der Gauge ein (die Ausrichtung vom Balken)
- *typ = SGAUGE_RL, SGAUGE_LR, SGAUGE_TB, SGAUGE_BT*

```
void SGUI_GaugeSetStyle(SGAUGE_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- *style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED*

```
void SGUI_GaugeSetFrameSize(SGAUGE_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn *STYLE = STYLE_FLAT*)

```
void SGUI_GaugeSetColor(SGAUGE_t* ptr, uint16_t c1, uint16_t c2)
```

- stellt die Farben der Gauge ein
- c1 = Balkenfarbe
- c2 = Hintergrundfarbe

```
void SGUI_GaugeSetMinMax(SGAUGE_t* ptr, int32_t min, int32_t max)
```

- stellt die Grenzwerte ein
- min = Minimalwert
- max = Maximalwert

```
void SGUI_GaugeSetValue(SGAUGE_t* ptr, int32_t value)
```

- stellt den Istwert ein
- *value = Istwert (muss zwischen min und max liegen)*

```
int32_t SGUI_GaugeGetValue(SGAUGE_t* ptr)
```

- auslesen vom aktuellen Istwert

11.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einer Gauge

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_GaugeCreate(10, 20, 100, 30);           // x=10, y=20, w=100, h=30
}
```

11.3.1 Beispiel-1:

Istwert von einem Gauge einstellen

```
void foo(void) {
  SGAUGE_t *gauge;

  SGUI_WindowCreateMain(1);                    // NR=1
  gauge=SGUI_GaugeCreate(10, 20, 100, 30);     // x=10,y=20,w=100,h=30
  SGUI_GaugeSetValue(gauge,30);                // Istwert=30
}
```

12 Objekt: Slider

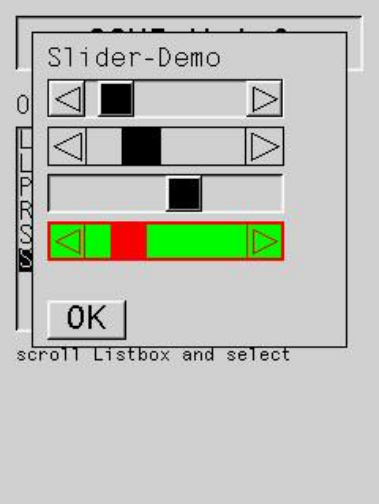

Abbildung 10 Objekt Slider

12.1 Beschreibung:

Slider können vom User per Touch betätigt werden.

Der Slider ist Rechteckig und kann in zwei Ausrichtungen auf einem Window platziert werden:

1. Horizontal
2. Vertikal

Der Slider besitzt einen *Minimalwert*, einen *Maximalwert* und einen *Istwert*, der vom User per Touch verändert werden kann.

Der *Maximalwert* muss größer als der *Minimalwert* sein und der *Istwert* liegt zwischen Minimum und Maximum.

Der aktuell eingestellte Wert kann entweder *„gepollt“* werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn der Slider per Touch betätigt wird.

12.2 Funktionen:

```
SSLIDER_t* SGUI_SliderCreate(uint16_t x, uint16_t y, uint16_t w, uint16_t h)
```

- erstellt einen Slider auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *x, h* = breite und höhe vom Slider

```
void SGUI_SliderSetTyp(SSLIDER_t* ptr, SSLIDER_TYP_t typ)
```

- stellt den Typ vom Slider ein (die Ausrichtung)
- *typ = SSLIDER_H, SSLIDER_V*

```
void SGUI_SliderSetStyle(SSLIDER_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- *style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED*
- (STYLE_RAISED oder STYLE_LOWERED sind identisch)

```
void SGUI_SliderSetArrowVisible(SSLIDER_t* ptr, bool visible)
```

- darstellung der Pfeil-Buttons ein/ausschalten
- *visible = true, false*

```
void SGUI_SliderSetFrameSize(SSLIDER_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn *STYLE = STYLE_FLAT*)

```
void SGUI_SliderSetColor(SSLIDER_t* ptr, uint16_t c1, uint16_t c2)
```

- stellt die Farben vom Slider ein
- *c1* = Farbe vom Slider
- *c2* = Hintergrundfarbe

```
void SGUI_SliderSetMinMax(SSLIDER_t* ptr, int32_t min, int32_t max)
```

- stellt die Grenzwerte ein
- min = Minimalwert
- max = Maximalwert

```
void SGUI_SliderSetStep(SSLIDER_t* ptr, uint16_t step)
```

- stellt die Schrittweite ein (für Touch Betätigung)
- step = Schrittweite [1...n]

```
void SGUI_SliderSetValue(SSLIDER_t* ptr, int32_t value)
```

- stellt den Istwert ein
- *value = Istwert (muss zwischen min und max liegen)*

```
int32_t SGUI_SliderGetValue(SSLIDER_t* ptr)
```

- auslesen vom aktuellen Istwert

```
void SGUI_SliderSetHandler(SSLIDER_t* ptr, void *fkt_ptr)
```

- stellt eine *„CallBack-Funktion“* für den Slider ein

12.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einem Slider

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_SliderCreate(10, 20, 100, 30);          // x=10, y=20, w=100, h=30
}
```

12.3.1 Beispiel-1:

Istwert von einem Slider abfragen

```
void foo(void) {
  SSLIDER_t *slider;
  int32_t istwert;

  SGUI_WindowCreateMain(1);                    // NR=1
  slider=SGUI_SliderCreate(10, 20, 100, 30);   // x=10, y=20, w=100, h=30

  istwert=SGUI_SliderGetValue(slider);         // Istwert auslesen
}
```

12.3.2 Beispiel-2:

CallBack-Funktion einrichten

```
SSLIDER_t *slider;

void slider_fkt(int32_t value) {
                                        // CallBack-Funktion für User wird
                                        // bei Touch Betätigung aufgerufen
}

void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  slider=SGUI_SliderCreate(10, 20, 100, 30);   // x=10,y=20,w=100,h=30
  SGUI_SliderSetHandler(slider, slider_fkt);
}
```

13 Objekt : Select-Button

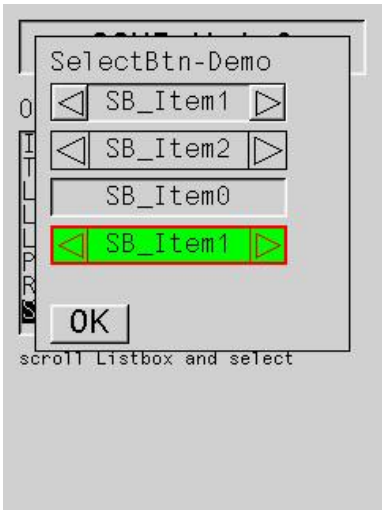

Abbildung 11 Objekt Select-Button

13.1 Beschreibung:

Select-Button zeigen aus einer Liste von Strings einen einzelnen an.

Der Select-Button kann vom User per Touch betätigt werden und so kann über zwei Pfeil-Buttons durch die String-Liste geschaltet werden.

Die zwei Pfeil-Buttons können entweder rechts/links oder oben/unten vom Select-Button angeordnet werden.

Für das manipulieren der Strings in der String Liste gibt es extra Befehle.

Das aktuell eingestellt Item kann entweder *„gepollt“* werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn der Button per Touch betätigt wird.

13.2 Funktionen:

```
SSELECTBTN_t* SGUI_SelectButtonCreate(uint16_t x, uint16_t y, uint16_t w,
uint16_t h)
```

- erstellt einen Select-Button auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *x, h* = breite und höhe vom Select-Button

```
void SGUI_SelectButtonSetTyp(SSELECTBTN_t* ptr, SSELECTBTN_TYP_t typ)
```

- stellt den Typ vom Select-Button ein (die Ausrichtung der Pfeil-Buttons)
- typ = SSELECTBTN_H, SSELECTBTN_V

```
void SGUI_SelectButtonSetStyle(SSELECTBTN_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED
- (STYLE_RAISED oder STYLE_LOWERED sind identisch)

```
void SGUI_SelectButtonSetArrowVisible(SSELECTBTN_t* ptr, bool visible)
```

- darstellung der Pfeil-Buttons ein/ausschalten
- visible = *true, false*

```
void SGUI_SelectButtonSetFrameSize(SSELECTBTN_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn STYLE = STYLE_FLAT)

```
void SGUI_SelectButtonSetColor(SSELECTBTN_t* ptr, uint16_t c1, uint16_t c2,
uint16_t c3)
```

- stellt die Farben vom Selet-Button ein
- *c1* = Textfarbe
- *c2* = Rahmenfarbe
- *c3* = Hintergrundfarbe

```
void SGUI_SelectButtonSetFont(SSELECTBTN_t* ptr, UB_Font* font)
```

- stellt die Schriftart vom Select-Button ein

```
void SGUI_SelectButtonAddItem(SSELECTBTN_t* ptr, char *txt)
```

- fügt ein String-Item ans Ende der String-Liste hinzu
- txt = String

```
void SGUI_SelectButtonDeleteItem(SSELECTBTN_t* ptr, uint16_t pos)
```

- löscht ein einzelnes String-Item aus der String-Liste
- pos = Item-Position die gelöscht wird

```
void SGUI_SelectButtonInsertItem(SSELECTBTN_t* ptr, uint16_t pos, char
*txt)
```

- fügt ein String-Item in eine String-Liste hinzu
- pos = Item-Position an der hinzugefügt wird
- txt = String

```
void SGUI_SelectButtonSetItem(SSELECTBTN_t* ptr, uint16_t pos, char *txt)
```

- ändert ein String-Item in der String-Liste
- pos = Item-Position die geändert wird
- txt = neuer String

```
char* SGUI_SelectButtonGetItem(SSELECTBTN_t* ptr, uint16_t pos)
```

- auslesen vom einem einzelnen String-Item der String-Liste
- pos = Item-Position die ausgelesen werden soll

```
uint16_t SGUI_SelectButtonGetItemCnt(SSELECTBTN_t* ptr)
```

- auslesen der Anzahl aller Elemente in der String-Liste

```
void SGUI_SelectButtonSetAktivItemNr(SSELECTBTN_t* ptr, uint16_t pos)
```

- auswählen welcher String aus der String-Liste angezeigt werden soll
- pos = Item-Position die angezeigt werden soll

```
int16_t SGUI_SelectButtonGetAktivItemNr(SSELECTBTN_t* ptr)
```

- auslesen welche Item-Position gerade angezeigt wird
- wenn kein Item angezeigt wird: *-1*

```
void SGUI_SelectButtonSetHandler(SSELECTBTN_t* ptr, void *fkt_ptr)
```

- stellt eine *„CallBack-Funktion“* für den Select-Button ein

13.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einem Select-Button

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_SelectButtonCreate(10, 20, 150, 30);    // x=10, y=20, w=150, h=30
}
```

13.3.1 Beispiel-1:

hinzufügen von zwei String-Elementen

```
void foo(void) {
  SSELECTBTN_t *sbtn;

  SGUI_WindowCreateMain(1);                    // NR=1
  sbtn=SGUI_SelectButtonCreate(10,20,150,30);  // x=10, y=20, w=150, h=30
  SGUI_SelectButtonAddItem(sbtn, "Hallo");     // Item-0 = "Hallo"
  SGUI_SelectButtonAddItem(sbtn, "Welt");      // Item-1 = "Welt"
}
```

Beispiel-2:

CallBack-Funktion einrichten

```
SSELECTBTN_t *sbtn;

void sbtn_fkt(uint16_t aktiv) {
                                        // CallBack-Funktion für User wird
                                        // bei Touch Betätigung aufgerufen
}

void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  sbtn=SGUI_SelectButtonCreate(10,20,150,30);  // x=10, y=20, w=150, h=30
  SGUI_SelectButtonAddItem(sbtn, "Hallo");     // Item-0 = "Hallo"
  SGUI_SelectButtonAddItem(sbtn, "Welt");      // Item-1 = "Welt"
  SGUI_SelectButtonSetHandler(sbtn, sbtn_fkt);
}
```

14 Objekt: Listbox

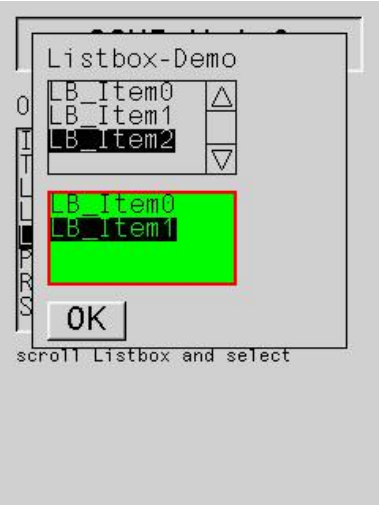

Abbildung 12 Objekt Listbox

14.1 Beschreibung:

Listboxen zeigen aus einer Liste von Strings ein paar Strings an (je nachdem wie hoch die Listbox ist)

Die Listbox kann vom User per Touch betätigt werden und so kann über zwei Pfeil-Buttons durch die String-Liste gescrollt werden.

Ein einzelnes String-Element kann per Touch selektiert werden.

Für das manipulieren der Strings in der Stringliste gibt es extra Befehle.

Das aktuell aktivierte Item kann entweder *„gepollt“* werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn die Listbox per Touch betätigt wird.

14.2 Funktionen:

```
SLISTBOX_t* SGUI_ListboxCreate(uint16_t x, uint16_t y, uint16_t w, uint16_t
h)
```

- erstellt eine Listbox auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *x, h* = Breite und Höhe der Listbox

```
void SGUI_ListboxSetStyle(SLISTBOX_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED
- (STYLE_RAISED oder STYLE_LOWERED sind identisch)

```
void SGUI_ListboxSetFrameSize(SLISTBOX_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn STYLE = STYLE_FLAT)

```
void SGUI_ListboxSetSliderSize(SLISTBOX_t* ptr, uint8_t px)
```

- stellt die breite vom Slider ein

```
void SGUI_ListboxSetSliderVisible(SLISTBOX_t* ptr, bool visible)
```

- darstellung vom Slider ein/ausschalten
- visible = *true, false*

```
void SGUI_ListboxSetColor(SLISTBOX_t* ptr, uint16_t c1, uint16_t c2,
uint16_t c3)
```

- stellt die Farben der Listbox ein
- c1 = Textfarbe
- c2 = Rahmenfarbe
- c3 = Hintergrundfarbe

```
void SGUI_ListboxSetFont(SLISTBOX_t* ptr, UB_Font* font)
```

- stellt die Schriftart der Listbox ein

```
void SGUI_ListboxAddItem(SLISTBOX_t* ptr, char *txt)
```

- fügt ein String-Item ans Ende der String-Liste hinzu
- txt = String

```
void SGUI_ListboxDeleteItem(SLISTBOX_t* ptr, uint16_t pos)
```

- löscht ein einzelnes String-Item aus der String-Liste
- pos = Item-Position die gelöscht wird

```
void SGUI_ListboxInsertItem(SLISTBOX_t* ptr, uint16_t pos, char *txt)
```

- fügt ein String-Item in eine String-Liste hinzu
- pos = Item-Position an der hinzugefügt wird
- txt = String

```
void SGUI_ListboxSetItem(SLISTBOX_t* ptr, uint16_t pos, char *txt)
```

- ändert ein String-Item in der String-Liste
- pos = Item-Position die geändert wird
- txt = neuer String

```
char* SGUI_ListboxGetItem(SLISTBOX_t* ptr, uint16_t pos)
```

- auslesen vom einem einzelnen String-Item der String-Liste
- pos = Item-Position die ausgelesen werden soll

```
uint16_t SGUI_ListboxGetItemCnt(SLISTBOX_t* ptr)
```

- auslesen der Anzahl aller Elemente in der String-Liste

```
void SGUI_ListboxSetAktivItemNr(SLISTBOX_t* ptr, int16_t pos)
```

- auswählen welcher String aus der String-Liste aktiviert werden soll
- pos = Item-Position die aktiviert werden soll (-1 = kein Item aktivieren)

```
int16_t SGUI_ListboxGetAktivItemNr(SLISTBOX_t* ptr)
```

- auslesen welche Item-Position gerade aktiviert ist
- wenn kein Item aktiv ist : -1

```
void SGUI_ListboxSetFirstItemNr(SLISTBOX_t* ptr, uint16_t pos)
```

- auswählen welche Item-Position als erstes in der Listbox angezeigt werden soll

```
void SGUI_ListboxSetHandler(SLISTBOX_t* ptr, void *fkt_ptr)
```

- stellt eine *„CallBack-Funktion“* für die Listbox ein

14.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einer Listbox

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_ListboxCreate(10,20,150,200);           // x=10,y=20,w=150,h=200
}
```

14.3.1 Beispiel-1:

hinzufügen von zwei String-Elementen

```
void foo(void) {
  SLISTBOX_t *box;

  SGUI_WindowCreateMain(1);                    // NR=1
  box=SGUI_ListboxCreate(10,20,150,200);       // x=10,y=20,w=150,h=200
  SGUI_ListboxAddItem(box,"Hello");            // Item-0 = "Hello"
  SGUI_ListboxAddItem(box,"World");            // Item-1 = "World"
}
```

14.3.2 Beispiel-2:

CallBack-Funktion einrichten

```
SLISTBOX_t *box;

void box_fkt(uint16_t aktiv) {
                                  // CallBack-Funktion für User wird
                                  // bei Touch Betätigung aufgerufen)
}

void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  box=SGUI_ListboxCreate(10,20,150,200);       // x=10,y=20,w=150,h=200
  SGUI_ListboxAddItem(box,"Hello");            // Item-0 = "Hello"
  SGUI_ListboxAddItem(box,"World");            // Item-1 = "World"
  SGUI_ListboxSetHandler(box, box_fkt);
}
```

15 Objekt: DropDown-Box

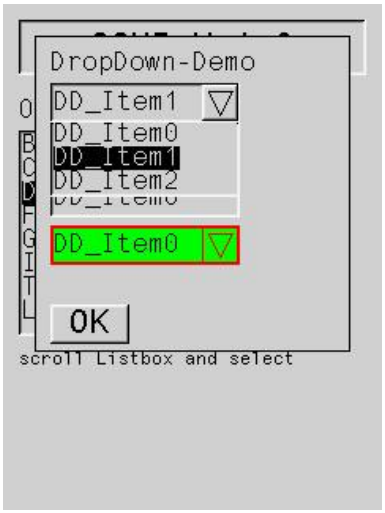

Abbildung 13 Objekt DropDown-Box

15.1 Beschreibung:

Eine DropDown-Box zeigt aus einer Liste von Strings einen einzelnen an.

Die DropDown-Box kann vom User per Touch betätigt werden, diese klappt dann nach unten auf und verhält sich dann wie eine Listbox.

Ein einzelnes String-Element kann per Touch selektiert werden.

Falls nicht alle Items auf dem Screen dargestellt werden können, wird ein Slider eingeblendet, um die Items per Touch scrollen zu können.

Die DropDown-Box kann per Touch wieder zugeklappt werden.

Für das manipulieren der Strings in der Stringliste gibt es extra Befehle.

Das aktuell aktivierte Item kann entweder *„gepollt“* werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn die DropDown-Box per Touch betätigt wird.

15.2 Funktionen:

```
SDROPDOWN_t* SGUI_DropdownCreate(uint16_t x, uint16_t y, uint16_t w,
uint16_t h)
```

- erstellt eine DropDown-Box auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *x, h* = Breite und Höhe der DropDown-Box

```
void SGUI_DropdownSetStyle(SDROPDOWN_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED
- (STYLE_RAISED oder STYLE_LOWERED sind identisch)

```
void SGUI_DropdownSetArrowVisible(SDROPDOWN_t* ptr, bool visible)
```

- darstellung der Pfeil-Buttons ein/ausschalten
- visible = *true, false*

```
void SGUI_DropdownSetFrameSize(SDROPDOWN_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn STYLE = STYLE_FLAT)

```
void SGUI_DropdownSetColor(SDROPDOWN_t* ptr, uint16_t c1, uint16_t c2,
uint16_t c3)
```

- stellt die Farben der DropDown-Box ein
- c1 = Textfarbe
- c2 = Rahmenfarbe
- c3 = Hintergrundfarbe

```
void SGUI_DropdownSetFont(SDROPDOWN_t* ptr, UB_Font* font)
```

- stellt die Schriftart der DropDown-Box ein

```
void SGUI_DropdownAddItem(SDROPDOWN_t* ptr, char *txt)
```

- fügt ein String-Item ans Ende der String-Liste hinzu
- txt = String

```
void SGUI_DropdownDeleteItem(SDROPDOWN_t* ptr, uint16_t pos)
```

- löscht ein einzelnes String-Item aus der String-Liste
- pos = Item-Position die gelöscht wird

```
void SGUI_DropdownInsertItem(SDROPDOWN_t* ptr, uint16_t pos, char *txt)
```

- fügt ein String-Item in eine String-Liste hinzu
- pos = Item-Position an der hinzugefügt wird
- txt = String

```
void SGUI_DropdownSetItem(SDROPDOWN_t* ptr, uint16_t pos, char *txt)
```

- ändert ein String-Item in der String-Liste
- pos = Item-Position die geändert wird
- txt = neuer String

```
char* SGUI_DropdownGetItem(SDROPDOWN_t* ptr, uint16_t pos)
```

- auslesen vom einem einzelnen String-Item der String-Liste
- pos = Item-Position die ausgelesen werden soll

```
uint16_t SGUI_DropdownGetItemCnt(SDROPDOWN_t* ptr)
```

- auslesen der Anzahl aller Elemente in der String-Liste

```
void SGUI_DropdownSetAktivItemNr(SDROPDOWN_t* ptr, uint16_t pos)
```

- auswählen welcher String aus der String-Liste angezeigt werden soll
- pos = Item-Position die angezeigt werden soll

```
int16_t SGUI_DropdownGetAktivItemNr(SDROPDOWN_t* ptr)
```

- auslesen welche Item-Position gerade angezeigt wird
- wenn kein Item angezeigt wird : -1

```
void SGUI_DropdownSetExpand(SDROPDOWN_t* ptr, bool expand)
```

- DropDown-Box expandieren ein/aus
- expand = true=expandieren

```
void SGUI_DropdownSetHandler(SDROPDOWN_t* ptr, void *fkt_ptr)
```

- stellt eine „CallBack-Funktion“ für die DropDown-Box ein

15.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einer DropDown-Box

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_DropdownCreate(10,20,150,30);           // x=10, y=20, w=150, h=30
}
```

15.3.1 Beispiel-1:

hinzufügen von zwei String-Elementen

```
void foo(void) {
  SDROPDOWN_t *box;

  SGUI_WindowCreateMain(1);                    // NR=1
  box=SGUI_DropdownCreate(10,20,150,30);       // x=10, y=20, w=150, h=30
  SGUI_DropdownAddItem(box,"Hello");           // Item-0 = "Hello"
  SGUI_DropdownAddItem(box,"World");           // Item-1 = "World"
}
```

15.3.2 Beispiel-2 :

CallBack-Funktion einrichten

```
SDROPDOWN_t *box;

void box_fkt(uint16_t aktiv) {
                                          // CallBack-Funktion für User wird
                                          // bei Touch Betätigung aufgerufen
}

void foo(void) {
  SGUI_WindowCreateMain(1);                     // NR=1
  box=SGUI_DropdownCreate(10,20,150,30);        // x=10, y=20, w=150, h=30
  SGUI_DropdownAddItem(box,"Hello");            // Item-0 = "Hello"
  SGUI_DropdownAddItem(box,"World");            // Item-1 = "World"
  SGUI_DropdownSetHandler(box, box_fkt);
}
```

16 Objekt: IntEdit-Feld

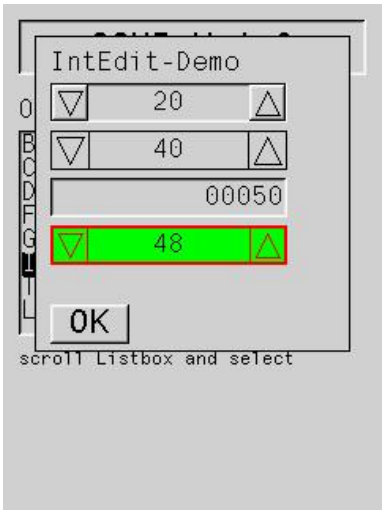

Abbildung 14 Objekt IntEdit-Feld

16.1 Beschreibung:

IntEdit-Felder dienen zum Darstellen von Integer-Zahlen.

Das IntEdit-Feld kann vom User per Touch betätigt werden und so kann über zwei Pfeil-Buttons der Zahlenwert verändert werden.

Das IntEdit-Feld besitzt einen *Minimalwert,* einen *Maximalwert* und einen *Istwert*, der vom User per Touch verändert werden kann.

Der Istwert ändert sich mit einer einstellbaren Schrittweite.

Die zwei Pfeil-Buttons können entweder rechts/links oder oben/unten vom IntEdit-Feld angeordnet werden.

Der *Maximalwert* muss größer als der *Minimalwert* sein und der *Istwert* liegt zwischen Minimum und Maximum.

Der aktuell eingestellte Wert kann entweder *„gepollt“* werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn das IntEdit-Feld per Touch betätigt wird.

16.2 Funktionen :

```
SINTEDIT_t* SGUI_IntEditCreate(uint16_t x, uint16_t y, uint16_t w, uint16_t h)
```

- erstellt ein IntEdit-Feld auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *x, h* = breite und höhe vom IntEdit-Feld

```
void SGUI_IntEditSetTyp(SINTEDIT_t* ptr, SINTEDIT_TYP_t typ)
```

- stellt den Typ vom IntEdit-Feld ein (die Ausrichtung der Pfeil-Buttons)
- typ = SINTEDIT_H, SINTEDIT_H2, SINTEDIT_V

```
void SGUI_IntEditSetStyle(SINTEDIT_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED
- (STYLE_RAISED oder STYLE_LOWERED sind identisch)

```
void SGUI_IntEditSetNumFormat(SINTEDIT_t* ptr, uint8_t digits, bool
padding)
```

- stellt das Format der Anzeige ein
- digtis = Anzahl der Ziffern der Zahl
- padding = *true,* wenn führende Nullen angezeigt werden sollen

```
void SGUI_IntEditSetAlignment(SINTEDIT_t* ptr, SINTEDIT_ALIGN_t align)
```

- stellt die Ausrichtung vom Text ein
- align = SINTEDIT_ALIGN_CENTER, SINTEDIT_ALIGN_LEFT, SINTEDIT_ALIGN_RIGHT

```
void SGUI_IntEditSetArrowVisible(SINTEDIT_t* ptr, bool visible)
```

- darstellung der Pfeil-Buttons ein/ausschalten
- visible = *true, false*

```
void SGUI_IntEditSetFrameSize(SINTEDIT_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn STYLE = STYLE_FLAT)

```
void SGUI_IntEditSetColor(SINTEDIT_t* ptr, uint16_t c1, uint16_t c2,
uint16_t c3)
```

- stellt die Farben vom IntEdit-Feld ein
- c1 = Textfarbe
- c2 = Rahmenfarbe
- c3 = Hintergrundfarbe

```
void SGUI_IntEditSetFont(SINTEDIT_t* ptr, UB_Font* font)
```

- stellt die Schriftart vom IntEdit-Feld ein

```
void SGUI_IntEditSetMinMax(SINTEDIT_t* ptr, int32_t min, int32_t max)
```

- stellt die Grenzwerte ein
- min = Minimalwert
- max = Maximumwert

```
void SGUI_IntEditSetStep(SINTEDIT_t* ptr, uint16_t step)
```

- stellt die Schrittweite ein (für Touch Betätigung)
- step = Schrittweite [1...n]

```
void SGUI_IntEditSetValue(SINTEDIT_t* ptr, int32_t value)
```

- stellt den Istwert ein
- value = Istwert (muss zwischen min und max liegen)

```
int32_t SGUI_IntEditGetValue(SINTEDIT_t* ptr)
```

- auslesen vom aktuellen Istwert

```
void SGUI_IntEditSetHandler(SINTEDIT_t* ptr, void *fkt_ptr)
```

- stellt eine *„CallBack-Funktion“* für das IntEdit-Feld ein

16.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einem IntEdit-Feld

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_IntEditCreate(10,20,100,30);            // x=10, y=20, w=100, h=30
}
```

16.3.1 Beispiel-1:

Istwert von einem IntEdit-Feld abfragen

```
void foo(void) {
  SINTEDIT_t *ptr;
  int32_t istwert;

  SGUI_WindowCreateMain(1);                    // NR=1
  ptr=SGUI_IntEditCreate(10,20,100,30);        // x=10, y=20, w=100, h=30

                                               // istwert auslesen
  istwert=SGUI_IntEditGetValue(num);
}
```

16.3.2 Beispiel-2:

CallBack-Funktion einrichten

```
SINTEDIT_t *ptr;

void intedit_fkt(int32_t value) {
                                        // CallBack-Funktion für User wird
                                        // bei Touch Betätigung aufgerufen
}

void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  ptr=SGUI_IntEditCreate(10,20,100,30);        // x=10, y=20, w=100, h=30
  SGUI_IntEditSetHandler(ptr, intedit_fkt);
}
```

17 Objekt: FloatEdit-Feld

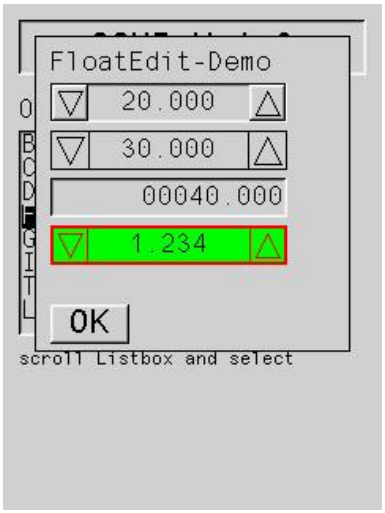

Abbildung 15 Objekt FloatEdith-Feld

17.1 Beschreibung:

FloatEdit-Felder dienen zum Darstellen von Float-Zahlen.

Das FloatEdit-Feld kann vom User per Touch betätigt werden und so kann über zwei Pfeil-Buttons der Zahlenwert verändert werden.

Das FloatEdit-Feld besitzt einen *Minimalwert*, einen *Maximalwert* und einen *Istwert*, der vom User per Touch verändert werden kann.

Der Istwert ändert sich mit einer einstellbaren Schrittweite.

Die zwei Pfeil-Buttons können entweder rechts & links oder oben & unten vom FloatEdit-Feld angeordnet werden.

Der *Maximalwert* muss größer als der *Minimalwert* sein und der *Istwert* liegt zwischen Minimum und Maximum.

Der aktuell eingestellte Wert kann entweder *„gepollt“* werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn das FloatEdit-Feld per Touch betätigt wird.

17.2 Funktionen:

```
SFLOATEDIT_t* SGUI_FloatEditCreate(uint16_t x, uint16_t y, uint16_t w,
uint16_t h)
```

- erstellt ein FloatEdit-Feld auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *x, h* = breite und höhe vom FloatEdit-Feld

```
void SGUI_FloatEditSetTyp(SFLOATEDIT_t* ptr, SFLOATEDIT_TYP_t typ)
```

- stellt den Typ vom FloatEdit-Feld ein (die Ausrichtung der Pfeil-Buttons)
- typ = SFLOATEDIT_H, SFLOATEDIT_H2, SFLOATEDIT_V

```
void SGUI_FloatEditSetStyle(SFLOATEDIT_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED
- (*STYLE_RAISED* oder *STYLE_LOWERED* sind identisch)

```
void SGUI_FloatEditSetNumFormat(SFLOATEDIT_t* ptr, uint8_t digits, bool
padding)
```

- stellt das Format der Anzeige ein
- digtis = Anzahl der Ziffern der Zahl
- padding = *true*, wenn führende Nullen angezeigt werden sollen

```
void SGUI_FloatEditSetAlignment(SFLOATEDIT_t* ptr, SFLOATEDIT_ALIGN_t align)
```

- stellt die Ausrichtung vom Text ein
- align = SFLOATEDIT_ALIGN_CENTER, SFLOATEDIT_ALIGN_LEFT, SFLOATEDIT_ALIGN_RIGHT

```
void SGUI_FloatEditSetArrowVisible(SFLOATEDIT_t* ptr, bool visible)
```

- darstellung der Pfeil-Buttons ein/ausschalten
- visible = *true, false*

```
void SGUI_FloatEditSetFrameSize(SFLOATEDIT_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn STYLE = STYLE_FLAT)

```
void SGUI_FloatEditSetColor(SFLOATEDIT_t* ptr, uint16_t c1, uint16_t c2, uint16_t
c3)
```

- stellt die Farben vom FloatEdit-Feld ein
- c1 = Textfarbe
- c2 = Rahmenfarbe
- c3 = Hintergrundfarbe

```
void SGUI_FloatEditSetFont(SFLOATEDIT_t* ptr, UB_Font* font)
```

- stellt die Schriftart vom FloatEdit-Feld ein

```
void SGUI_FloatEditSetMinMax(SFLOATEDIT_t* ptr, float min, float max)
```

- stellt die Grenzwerte ein
- min = Minimalwert
- max = Maximumwert

```
void SGUI_FloatEditSetStep(SFLOATEDIT_t* ptr, float step)
```

- stellt die Schrittweite ein (für Touch Betätigung)
- step = Schrittweite [1...n]

```
void SGUI_FloatEditSetValue(SFLOATEDIT_t* ptr, float value)
```

- stellt den Istwert ein
- value = Istwert (muss zwischen min und max liegen)

```
float SGUI_FloatEditGetValue(SFLOATEDIT_t* ptr)
```

- auslesen vom aktuellen Istwert

```
void SGUI_FloatEditSetHandler(SFLOATEDIT_t* ptr, void *fkt_ptr)
```

- stellt eine *„CallBack-Funktion“* für das FloatEdit-Feld ein

17.3 Minimal-Beispiel:

Erzeugen von einem Main-Window und anzeigen von einem FloatEdit-Feld

```
void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  SGUI_FloatEditCreate(10,20,100,30);          // x=10, y=20, w=100, h=30
}
```

17.3.1 Beispiel-1:

Istwert von einem FloatEdit-Feld abfragen

```
void foo(void) {
  SFLOATEDIT_t *ptr;
  float istwert;

  SGUI_WindowCreateMain(1);                    // NR=1
  ptr=SGUI_FloatEditCreate(10,20,100,30);      // x=10, y=20, w=100, h=30

  // istwert auslesen
  istwert=SGUI_FloatEditGetValue(num);
}
```

17.3.2 Beispiel-2:

CallBack-Funktion einrichten

```
SFLOATEDIT_t *ptr;

void floatedit_fkt(float value) {
                                       // CallBack-Funktion für User wird
                                       // bei Touch Betätigung aufgerufen
}

void foo(void) {
  SGUI_WindowCreateMain(1);                    // NR=1
  ptr=SGUI_FloatEditCreate(10,20,100,30);      // x=10,y=20,w=100,h=30
  SGUI_FloatEditSetHandler(ptr, floatedit_fkt);
}
```

18 Objekt: PICTURE

Abbildung 16 Objekt Picture

18.1 Beschreibung:

Pictures dienen zum anzeigen von Bildern und können per Touch betätigt werden.

Die Bilder müssen als C-Image im Flash liegen. Zum umwandeln kann mein Converter-Programm benutzt werden.

Alle Bilder müssen in der IDE eingebunden und per *„Include“* bekannt sein.

Es gibt drei Arten von Bildern:

- NOBTN ohne Button Funktion
- PUSH Taster
- PUSHPULL Schalter

Taster und Schalter können zwei Bilder zugewiesen werden eines für *"aktiv"* und eines für *"inaktiv"*

Ein Taster ist unbetätigt *"inaktiv"* und wechselt auf *"aktiv"* wenn er per Touch gedrückt wird.

Ein Schalter wechselt seinen aktuellen Zustand jedes Mal, wenn er per Touch betätigt wird.

Der Zustand von einem Picture kann entweder *„gepollt“* werden oder es kann eine *„CallBack-Funktion“* eingerichtet werden, die automatisch aufgerufen wird, wenn das Picture per Touch betätigt wird.

18.2 Funktionen :

```
SPICTURE_t* SGUI_PictureCreate(uint16_t x, uint16_t y, uint16_t w, uint16_t h)
```

- erstellt ein Picture auf dem aktuellen Window
- *x, y* = Position auf dem Window
- *w, h* = breite und höhe vom Picture

```
void SGUI_PictureSetMode(SPICTURE_t* ptr, SPICTURE_MODE_t mode)
```

- stellt den Mode von einem Picture ein
- mode = SPICTURE_NOBTN, SPICTURE_PUSH, SPICTURE_PUSHPULL

```
void SGUI_PictureSetImage1(SPICTURE_t* ptr, UB_Image *img)
```

- zuweisung eines Bildes aus dem Flash zu einem Picture fuer den status "inaktiv"
- img = Pointer auf das Bild

```
void SGUI_PictureSetImage2(SPICTURE_t* ptr, UB_Image *img)
```

- zuweisung eines Bildes aus dem Flash zu einem Picture fuer den status "aktiv"
- img = Pointer auf das Bild

```
void SGUI_PictureSetAktiv(SPicture_t* ptr, bool aktiv)
```

- stellt den aktuellen Status vom Picture ein

```
void SGUI_PictureSetHandler(SPicture_t* ptr, void *fkt_ptr);
```

- stellt eine „CallBack-Funktion“für das Picture ein

```
bool SGUI_PictureIsAktiv(SPicture_t* ptr)
```

- abfrage vom aktuellen Status vom Picture

18.3 Minimal-Beispiel:

- Erzeugen von einem Main-Window und anzeigen von einem Bild

```
void foo(void) {
  SPICTURE_t *picture;

  SGUI_WindowCreateMain(1); // NR=1
  picture=SGUI_PictureCreate(10,20,100,50); // x=10, y=20, w=100, h=50
  SGUI_PictureSetImage1(picture,&Bild1);    // Bild1 wird angezeigt
}
```

18.3.1 Beispiel-1:

Picture als Push-Button erstellen

```
void foo(void) {
  SPICTURE_t *picture;

  SGUI_WindowCreateMain(1); // NR=1
  picture=SGUI_PictureCreate(10,20,100,50); // x=10,y=20,w=100,h=50
  SGUI_PictureSetMode(picture,SPICTURE_PUSH);
  SGUI_PictureSetImage1(picture,&Bild1);    // Bild1 für inaktiv
  SGUI_PictureSetImage2(picture,&Bild2);    // Bild2 für aktiv
}
```

18.3.2 Beispiel-2:

Status von einem Picture abfragen

```
void foo(void) {
  SPICTURE_t *picture;

  SGUI_WindowCreateMain(1);                       // NR=1
  picture=SGUI_PictureCreate(10,20,100,50);       // x=10, y=20, w=100, h=50
  SGUI_PictureSetMode(picture,SPICTURE_PUSH);
  SGUI_PictureSetImage1(picture,&Bild1);          // Bild1 für inaktiv
  SGUI_PictureSetImage2(picture,&Bild2);          // Bild2 für aktiv

  if(SGUI_PictureIsAktiv(btn)==true) {
                                                  // picture ist betätigt
  }
}
```

18.3.3 Beispiel-3:

CallBack-Funktion einrichten

```
SPICTURE_t *picture;

void pic_fkt(bool aktiv) {
                                        // CallBack-Funktion für User wird
                                        // bei Touch Betätigung aufgerufen
}

void foo(void) {
  SGUI_WindowCreateMain(1);                       // NR=1
  picture=SGUI_PictureCreate(10,20,100,50);       // x=10, y=20, w=100, h=50
  SGUI_PictureSetMode(picture,SPICTURE_PUSH);
  SGUI_PictureSetImage1(picture,&Bild1);          // Bild1 für inaktiv
  SGUI_PictureSetImage2(picture,&Bild2);          // Bild2 für aktiv
```

19 Objekt : Graph

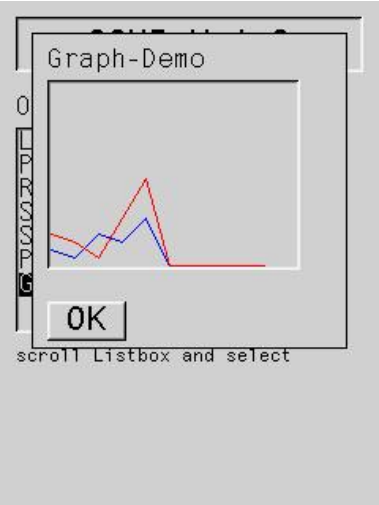

Abbildung 17 Graph-Demo

19.1 Beschreibung :

Mit Graphen koennen Zahlen-Arrays als Kurven bzw. Linien (wie bei einem Oszi) dargestellt werden.

Jeder Graph kann mehrere Kanaele gleichzeitig darstellen.

19.2 Funktionen :

```
SGRAPH_t* SGUI_GraphCreate(uint16_t x, uint16_t y, uint16_t w, uint16_t h)
```

- erstellt ein Graph auf dem aktuellen Window
- x, y = Position auf dem Window
- x, h = breite und höhe vom Graph

```
void SGUI_GraphSetStyle(SGRAPH_t* ptr, STYLE_TYP_t style)
```

- stellt den Style vom Rahmen ein
- style = STYLE_FLAT, STYLE_RAISED, STYLE_LOWERED

```
void SGUI_GraphSetFrameSize(SGRAPH_t* ptr, uint8_t px)
```

- stellt die dicke vom Rahmen ein (wenn STYLE = STYLE_FLAT)

```
void SGUI_GraphSetColor(SGRAPH_t* ptr, uint16_t c1, uint16_t c2)
```

- stellt die Farbe von einem Graph ein
- *c1* = Rahmenfarbe (wenn STYLE = STYLE_FLAT)
- *c2* = Hintergrundfarbe

```
void SGUI_GraphCreateDataArray(SGRAPH_t* ptr, uint8_t ch, uint16_t cnt)
```

- erstellt das zwei Dimensionale DatenArray (reserviert Speicherplatz)
- ch = Anzahl der Kanäle (1...4)
- cnt = Anzahl der Datenwerte pro Kanal (1...400)

```
void SGUI_GraphSetCHValue(SGRAPH_t* ptr, uint8_t ch, uint16_t pos, uint8_t value)
```

- speichert einen Datenwert im Datenarray
- ch = Kanal-Nummer (0...n)
- pos = Datenposition (0...m)
- value = Datenwert

```
uint8_t SGUI_GraphGetCHValue(SGRAPH_t* ptr, uint8_t ch, uint16_t pos)
```

- auslesen von einem Datenwert aus dem Datenarray
- ch = Kanal-Nummer (0...n)
- pos = Datenposition (0...m)

```
void SGUI_GraphSetCHVisible(SGRAPH_t* ptr, uint8_t ch, bool visible)
```

- Kanal sichbar oder unsichtbar schalten
- ch = Kanal-Nummer (0...n)

```
void SGUI_GraphSetCHColor(SGRAPH_t* ptr, uint8_t ch, uint16_t c)
```

- Farbe von einem Kanal einstellen
- ch = Kanal-Nummer (0...n)
- c = Farbe

19.3 Minimal-Beispiel :

Erzeugen von einem Main-Window und anzeigen von einem Graph mit einem Kanal und 50 Datenwerten (ohne Dateninhalt)

```
void foo(void) {
  SGRAPH_t *graph;

  SGUI_WindowCreateMain(1);                    // nr=1
  graph=SGUI_GraphCreate(10,20,100,150);       // x=10, y=20, w=100, h=150
  SGUI_GraphCreateDataArray(graph,1,50);       // ch=1, anzahl=50
}
```

19.3.1 Beispiel-1 :

Ein Graph mit einem Kanal und 50 Datenwerten anlegen und die ersten 5 Werte mit Daten füllen

```
void foo(void) {
  SGRAPH_t *graph;

  SGUI_WindowCreateMain(1);                // nr=1
  graph=SGUI_GraphCreate(10,20,100,150); // x=10, y=20, w=100, h=150
  SGUI_GraphCreateDataArray(graph,1,50); // Kanäle=1, Datenwerte=50

  SGUI_GraphSetCHValue(graph,0,0,20);     // ch0,pos0 =20
  SGUI_GraphSetCHValue(graph,0,1,10);     // ch0,pos1 =10
  SGUI_GraphSetCHValue(graph,0,2,40);     // ch0,pos2 =40
  SGUI_GraphSetCHValue(graph,0,3,20);     // ch0,pos3 =20
  SGUI_GraphSetCHValue(graph,0,4,50);     // ch0,pos4 =50
}
```

19.3.2 Beispiel-2 :

Ein Graph mit zwei Kanälen und je 30 Datenwerten anlegen und die ersten 5 Werte beider Kanäle mit Daten füllen

```
void foo(void) {
  SGRAPH_t *graph;

  SGUI_WindowCreateMain(1);                // nr=1
  graph=SGUI_GraphCreate(10,20,100,150); // x=10, y=20, w=100, h=150
  SGUI_GraphCreateDataArray(graph,2,30); // Kanäle=2, Datenwerte=30

  SGUI_GraphSetCHValue(graph,0,0,20);           // ch0,pos0 =20
  SGUI_GraphSetCHValue(graph,0,1,10);           // ch0,pos1 =10
  SGUI_GraphSetCHValue(graph,0,2,40);           // ch0,pos2 =40
  SGUI_GraphSetCHValue(graph,0,3,20);           // ch0,pos3 =20
  SGUI_GraphSetCHValue(graph,0,4,50);           // ch0,pos4 =50
  SGUI_GraphSetCHColor(graph,0,RGB_COL_RED);    // farbe=rot

  SGUI_GraphSetCHValue(graph,1,0,50);           // ch1,pos0 =50
  SGUI_GraphSetCHValue(graph,1,1,20);           // ch1,pos1 =20
  SGUI_GraphSetCHValue(graph,1,2,40);           // ch1,pos2 =40
  SGUI_GraphSetCHValue(graph,1,3,10);           // ch1,pos3 =10
  SGUI_GraphSetCHValue(graph,1,4,20);           // ch1,pos4 =20
  SGUI_GraphSetCHColor(graph,1,RGB_COL_GREEN); // farbe=grün
```

20 Revisionsgraf

NR	Anlass	Version
1	Erstellung des Dokuments	1.3
2	Nachtrag Objekt Graf	1.3

Mit freundlicher Genehmigung des Programmierers Uwe Becker

Erstellt von Michael Steinsträßer DD4MS 16.06.2015