SDL Gauge Library

Reference Manual

Rastergraf

Rastergraf, Inc.
1804-P SE First St.
Redmond, OR 97756
(541) 923-5530
FAX (541) 923-6575

web: http://www.rastergraf.com

email: support@rastergraf.com

Release 3.6.2

November 16, 2006

Notices

Trademarks mentioned in this manual are the property of their respective owners.

This manual is based in part on *Xlib - C Language X Interface, Version 11, Release 6* which is copyrighted material. This documentation is used under the terms of its copyright which grants free use as noted below.

SDL is distributed by Rastergraf, Inc under license from:

Curtiss-Wright Controls Embedded Computing Video and Graphics (formerly Peritek)

Preface

This manual is a reference document for programmers using the SDL Gauge Library and is used in conjunction with the SDL C Function Reference manual. This manual provides describes each function in the Gauge Library.

SDL is licensed in object and source format for use with embedded systems and in other industrial graphics hardware products. Although designed primarily for use with graphics boards from Rastergraf, the source code format allows SDL to be easily ported to the many different products in the marketplace.

The functions described in this manual represent the complete list of SDL library functions. Not all functions may be implemented in all graphics board drivers.

SDL requires a system with an ANSI C compiler and operating system and processor with a linear address space.

SDL is the product of over 20 years of experience developing graphics products for industry and we hope you will find it useful.

Please visit our web page at http://www.rastergraf.com for the latest information on our current graphics products.

Contents

SDL Overview

System Overview

The *Standard Drawing Library, SDL,* is a scaleable C graphics library designed for use with real-time and non-real-time operating systems. *SDL* is small, compact, ROMable, and offers device independent graphics functions for board level and embedded systems applications.

SDL is easy to use and provides a complete set of graphics primitives. These graphics primitives can be extended by adding **utility functions** for specialized graphics tasks**.**

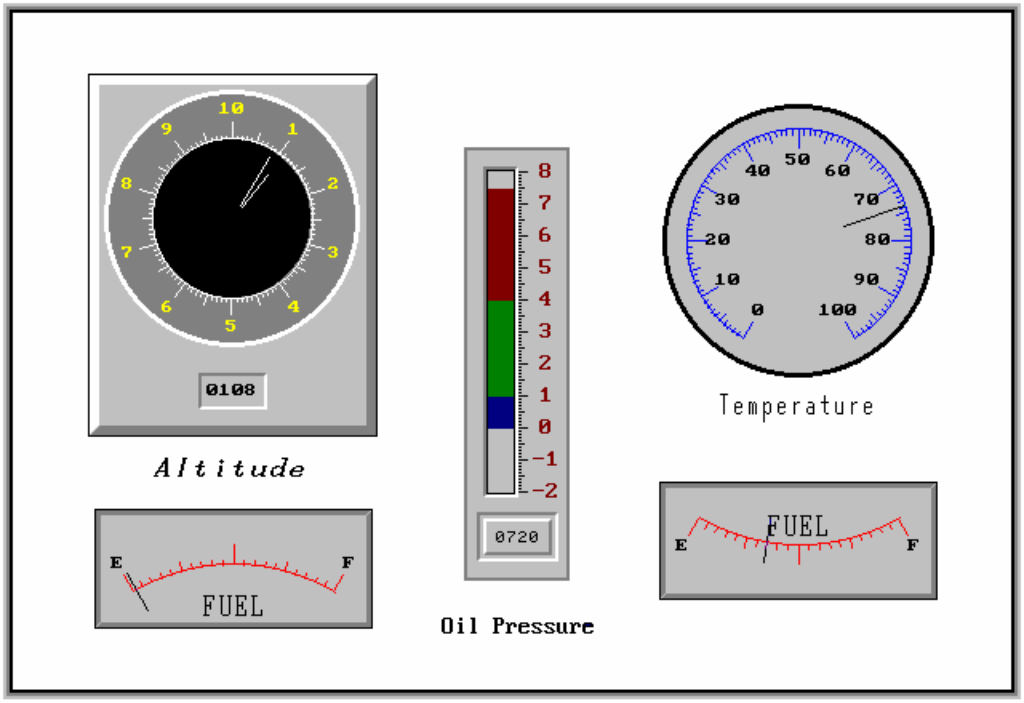
SDL is written in ANSI C and is supplied in library format, which means that its target code size can be controlled by limiting the number of functions used in a given application. *SDL* has been designed to run on any CPU and operating system that uses linear addressing and that is supported by an ANSI C compiler and linker.

SDL includes a *generic graphics driver* module that provides the hardware specific routines needed to interface to the graphics hardware in a user's system. A **graphics driver specification** is provided to customers licensing SDL in source format, allowing the customer to port *SDL* to the specific graphics hardware in the customer's system. *SDL* can be readily ported to a new system by changing this one module. *SDL*, therefore, does not depend on any specific graphics hardware, and versions of *SDL* can be used with VGA chips, EL panels, LCD displays, and most other graphics devices.

SDL Gauge Library

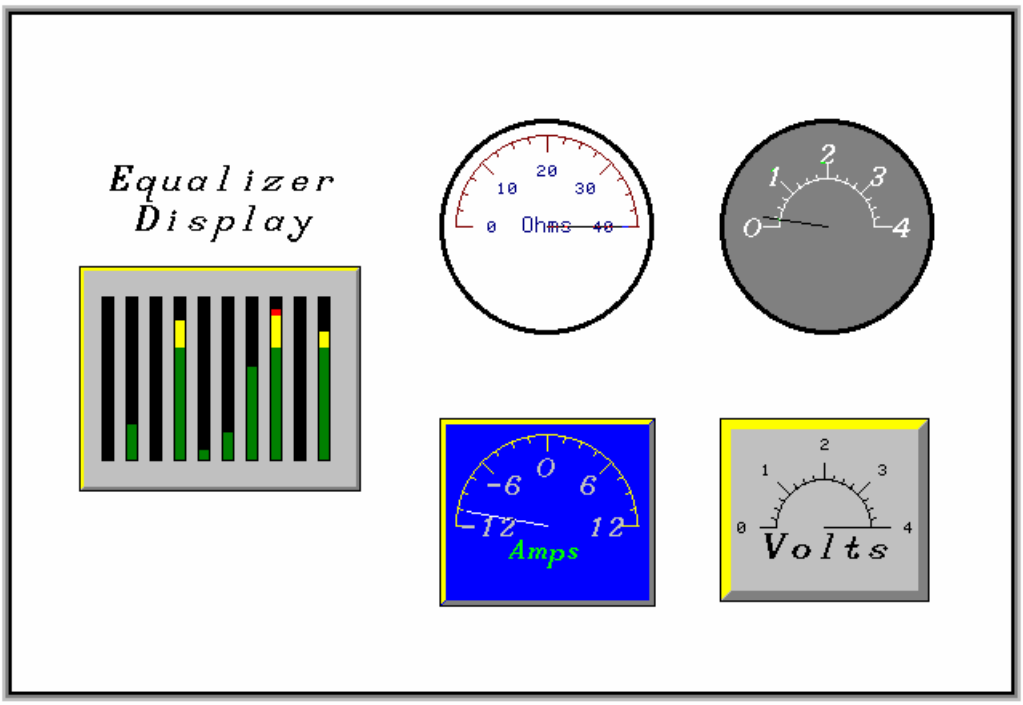
The Gauge Library consists of four different types of gauges. They are Bar, Text, Clock, and Arc. Each type of gauge has many configurable parameters that can be set to produce almost any desired look. A small example of some of the possibilities can be seen in the screen captures.

Examples



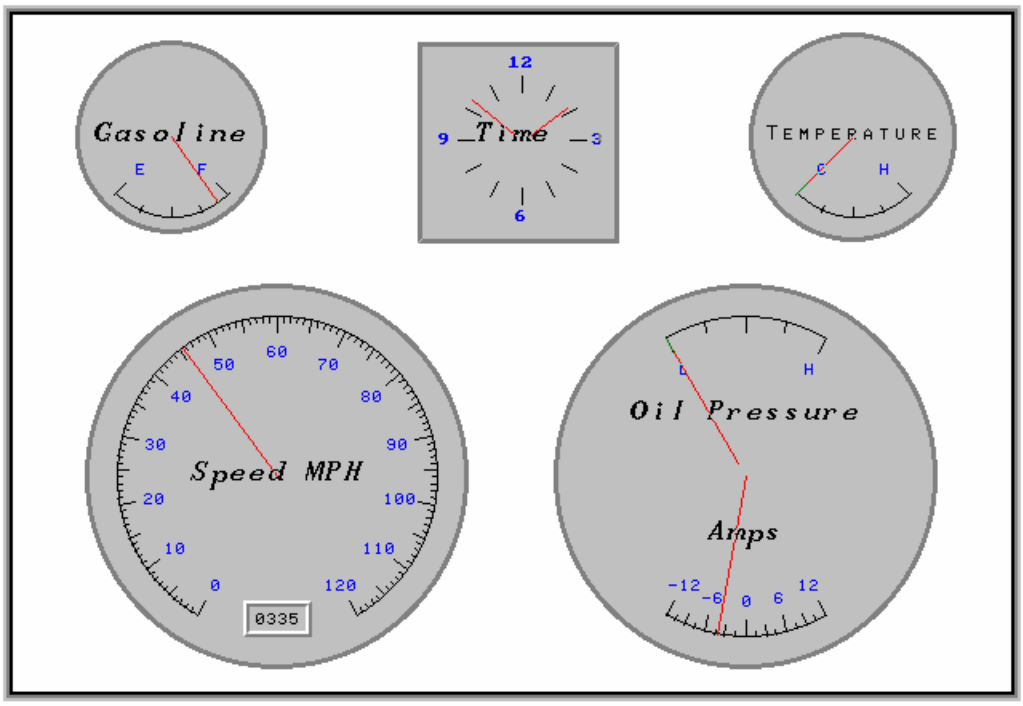
This example screen shows seven gauges: 1 Bar, 2 Text, 1 Clock, and 2 Arc gauges.

Note: At the top left there are two gauges on top of each other, the Text gauge has a large rectangle background and the Clock gauge is on top of that background. The center gauges are similar, the Text gauge is on the background of the Bar gauge.



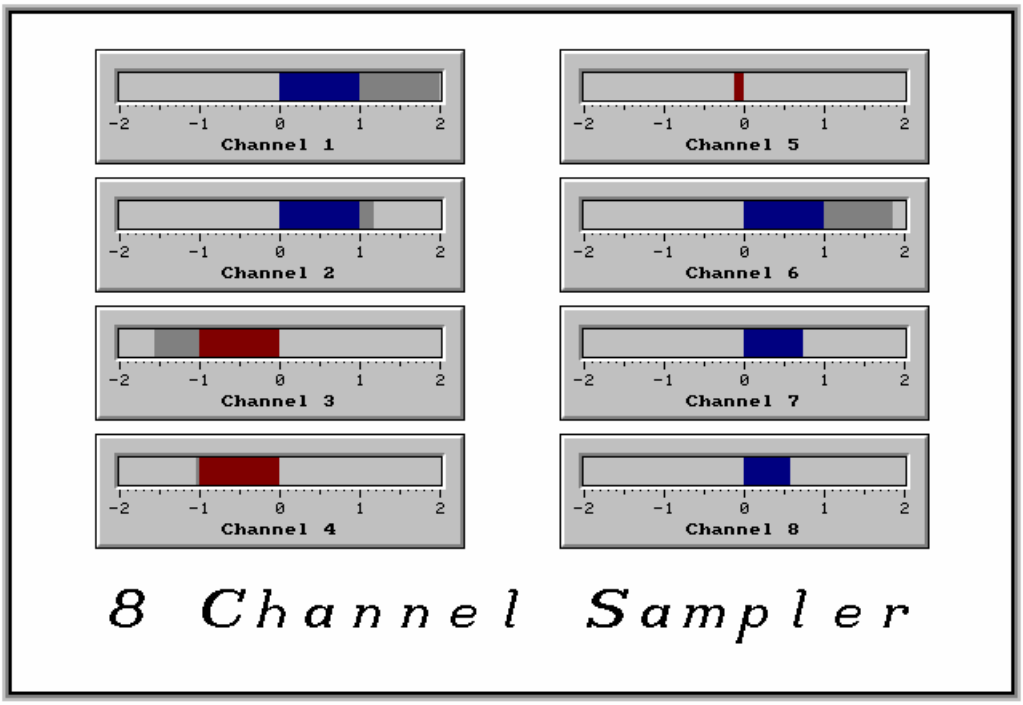
This example screen shows 14 gauges: 10 Bar gauges, and 4 Arc gauges.

Note: On the left there are 10 Bar gauges. Only one of them has a background.



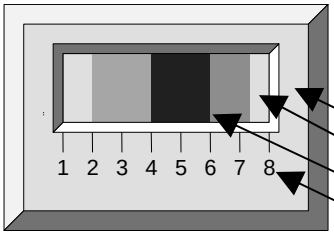
This example screen shows seven gauges: 1 Text, 1 Clock, and 5 Arc gauges.

Note: This particular screen was designed after a 1967 Ford Mustang dash board. The two gauges on the bottom right, only one has a background, and the other is positioned so that it has the same center point.



This example screen shows eight Bar gauges.

Bar Gauge



The *Bar Gauge* displays data by moving a horizontal or vertical bar next to a scale, similar to a thermometer, but the bar gauge uses different colors at different levels.

The Bar gauge consists of four parts:
(1) the background,
(2) the inside,
(3) the bar, and
(4) the gradation (grad) marks.

A *Bar Gauge* is created from the information in the structures shown below. Structure ***sBarParams*** is the main structure and contains structures, pointers to structures, and parameters that define the *Bar Gauge*. Only the first seven members (pointers) are specified by the user. The remaining members (shaded) are filled out by ***initBarGauge().*** An overview of the *Bar Gauge* structures is shown below.

Structure **sBarParams**

Type	Member Name
void	*background
int	pixelPerMinorMark
sGradMarks	*grad
sGradMarkText	*font
sBarIn	*in
sGaugeRange	*values
int	type
sPoint	short xy.x short xy.y
sPoint	short dimensions.x short dimensions.y
sRectangle	short update.x short update.y short update.width short update.height
int	oldValue

sRectBackground
type
borderLeft
borderTop
borderRight
borderBottom
bodyColor
outlineColor
mitreWidth
mitreTopColor
mitreBottomColor

sGradMarks
minorPerMedium
mediumPerMajor
majorPerGauge
lengthMajor
lengthMedium
lengthMinor
color
direction

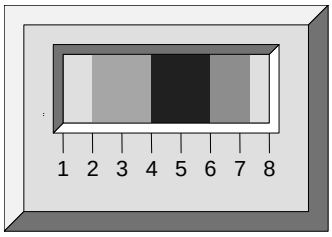
sGradMarkText
number
color
text
xOffset
yOffset

sGaugeRange
min
max

sBarIn
thickness
color1
color2
color3
color4
cutoff1
cutoff2
cutoff3
outlineColor
backgroundColor
miterTopColor
miterBottomColor
miterWidth

Bar Gauge Construction

Completed Bar Gauge

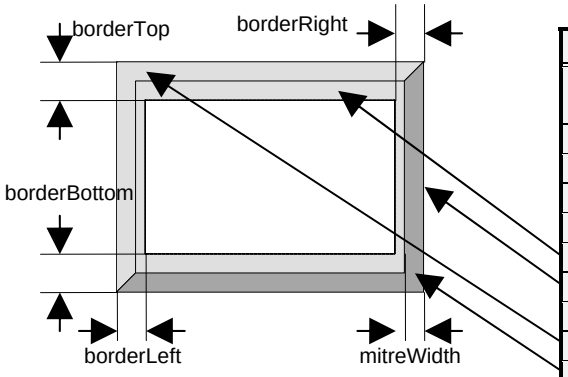


The *Bar Gauge* is defined by the parameters contained in the structure ***sBarParams***, and by the six structures, shown below.

The following parameters determine the width of the update area: pixelPerMinorMark, minorPerMedium, mediumPerMajor, majorPerGauge. All of these parameters are in the ***sGradMarks*** Structure except for pixelPerMinorMark which is in ***sBarParams.***

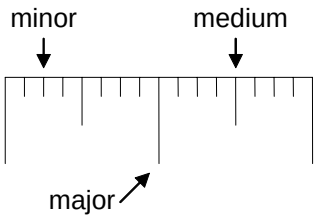
The *Bar Gauge* may be horizontal or vertical. The type member of ***BarParams*** determines this; the constants VFILLGAUGE and HFILLGAUGE are defined types of bar gauges

The grad marks are defined by the two structures ***sGradMarks*** and ***sGradMarkText***, which are also used by some of the other Rastergraf gauges. The number, length and color of the grad marks are controlled by ***sGradMarks***. The color, position, and the text itself are defined in ***sGradMarkText.*** All lengths are in pixels.



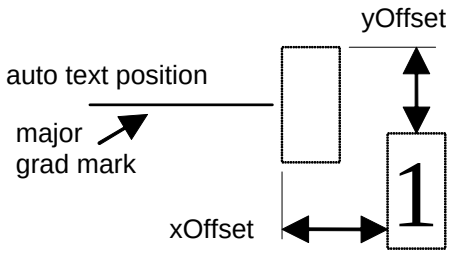
Structure **sRectBackground**

Type	Member Name	Comment
int	type	0=NONE No background 1=RECTANGLE.
int	borderLeft	See diagram at left
int	borderTop	See diagram at left
int	borderRight	See diagram at left
int	borderBottom	See diagram at left
unsigned long	bodyColor	Gauge body color
unsigned long	outlineColor	Color for line around gauge
int	mitreWidth	Top, bottom & sides
unsigned long	mitreTopColor	Top and left side
unsigned long	mitreBottomColor	Bottom & right side



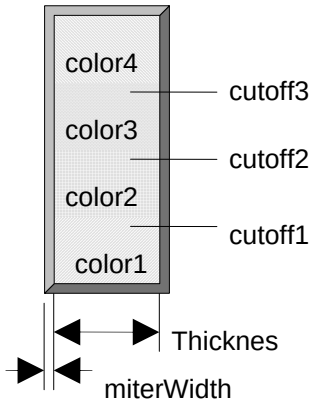
Structure **sGradMarks**

Type	Member Name	Comment
int	minorPerMedium	Number of minior marks between medium marks
int	mediumPerMajor	Number of medium marks between major marks
int	majorPerGauge	Number of major marks for the gauge
int	lengthMajor	Length of major marks
int	lengthMedium	Length of mediummarks
int	lengthMinor	Length of minor marks
unsigned long	color	Color of the marks
int	direction	Up, Down, Left, or Right. The example to the left shows DOWN.



Structure **sGradMarkText**

Type	Member Name	Comment
int	number	Font number or name
unsigned long	color	Color of the text
char	*text	Text to use, use '|' between strings, i.e. "1|2|3", or "Cold|Hot".
int	xOffset	Distance to move text.
int	yOffset	Distance to move text.

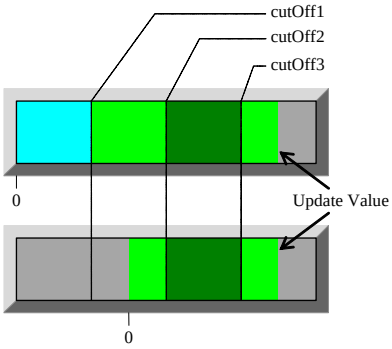


Structure **sBarIn**

Type	Member Name	Comment
int	thickness	the width of the bar
unsigned long	color1	first color
unsigned long	color2	second color
unsigned long	color3	third color
unsigned long	color4	fourth color
int	cutoff1	value where color changes
int	cutoff2	value where color changes
int	cutoff3	value where color changes
unsigned long	outlineColor	Outline color
unsigned long	backgroundColor	background color
unsigned long	miterTopColor	top and left miter color
unsigned long	miterBottomColor	bottom and right miter color
int	miterWidth	width of miter area

sGaugeRange

Type	Member Name	Comment
int	min	minimum allowed update value
int	max	maximum allowed update value



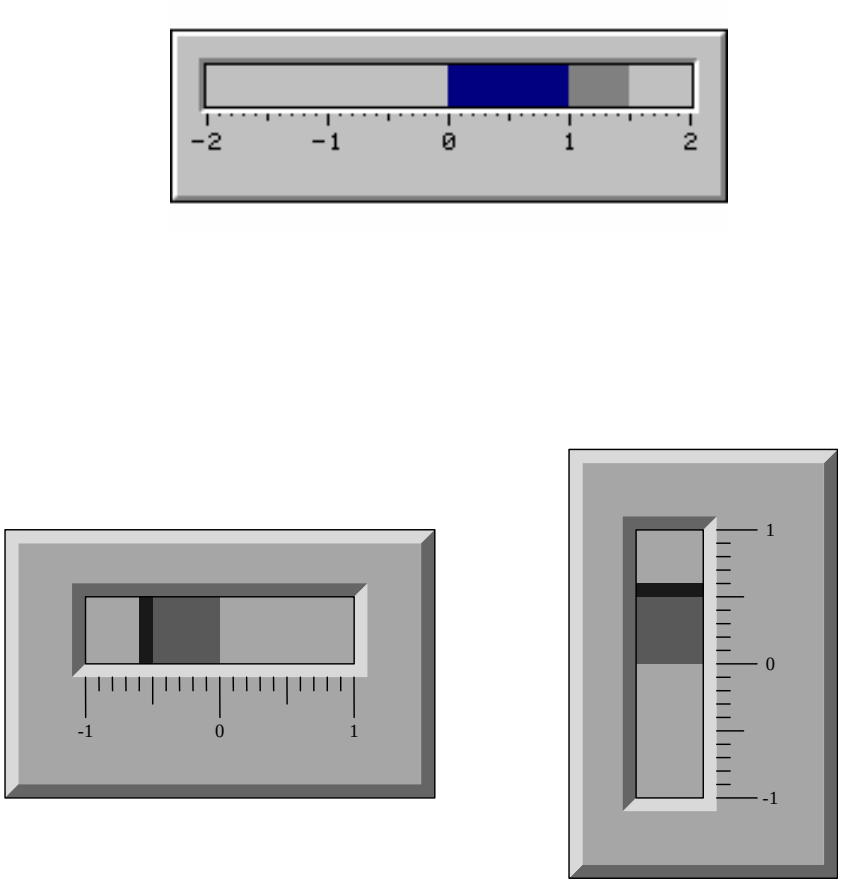
Color and Cutoff Examples

Bar Gauge Functions

There are seven functions that control the *Bar Gauge*.

Functions	Description
int initBarGauge(sBarParams *Bar);	Initialize the bar gauge and internal structures.
int drawBarGauge(sBarParams *Bar, int x, int y);	Draw the gauge on the screen.
int updateBarGauge(sBarParams *Bar, int value,int bool);	Change the displayed value.
int updateBarGaugeRel(sBarParams *Bar, int value);	Change the displayed value.
int getBarGaugeWidth (sBarParams *Bar, int *W);	Get the overall width of the gauge.
int getBarGaugeHeight(sBarParams *Bar, int *H);	Get the overall height of the gauge.
int copyBarGauge(sBarParams *src, sBarParams *dst);	Copy the Bar gauge structures.

These functions are discussed in detail on the following pages.



Bar Gauge Examples

initBarGauge

NAME

initBarGauge() - Initialize the bar gauge and internal structures.

SYNOPSIS

```
int initBarGauge
        (
        sBarParams *Bar        /* main bar gauge structure */
        )
```

DESCRIPTION

This function initializes the bar gauge and internal structures. This function must be the first function called for each instance of a Bar Gauge.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

bargauge.h, gauge.h

SEE ALSO

Bar Gauge Construction, drawBarGauge(), updateBarGauge(), updateBarGaugeRel(), getBarGaugeWidth(), getBarGaugeHeight(), copyBarGauge()

drawBarGauge

NAME

drawBarGauge() - Draw the gauge on the screen.

SYNOPSIS

```
int drawBarGauge
        (
        sBarParams *Bar,        /* main bar gauge structure */
        int x,                  /* x,y placement of upper left corner of gauge */
        int y
        )
```

DESCRIPTION

This function draws the entire gauge on the screen. This function must be called after initBarGauge(). The parameters x and y specify the placement of the upper left corner of the gauge. Functions getBarGaugeWidth() and getBarGaugeHeight() may be called to determine the size of the gauge. This function only has to be called once per instance of the Bar gauge, unless for some reason all or part of the gauge is erased.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

bargauge.h, gauge.h

SEE ALSO

Bar Gauge Construction, initBarGauge(), updateBarGauge(), updateBarGaugeRel(), getBarGaugeWidth(), getBarGaugeHeight(), copyBarGauge()

updateBarGauge

NAME

updateBarGauge() - Change value displayed.

SYNOPSIS

```
int updateBarGauge
        (
        sBarParams *Bar,        /* main bar gauge structure */
        int value,              /* new value to display */
        int bool                /* non-zero draw entire bar */
        )
```

DESCRIPTION

This function changes the displayed value. This function must be called after initBarGauge() and drawBarGauge().

The value parameter must be between the min and max values in the ***sGaugeRange*** structure. In the example below the numbers are in tens, the min value is -20, the max value is 80. the cutoff values are -10,10,40. The value used to update the gauge was 70. The gauge shows values 0 - 10 in color2, 10 - 40 in color3, and values 40 - 70 in color4. (See ***sBarIn*** in the **Bar Gauge Construction** section.)

When the bool parameter is zero, only the part of the bar that has changed is redrawn, for speed. If the entire bar needs to be drawn, then the parameter bool should be non-zero.

If the surrounding parts of the gauge need to be redrawn, call drawBarGauge().

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

bargauge.h, gauge.h

SEE ALSO

Bar Gauge Construction, initBarGauge(), drawBarGauge(), updateBarGaugeRel(), getBarGaugeWidth(), getBarGaugeHeight(), copyBarGauge()

updateBarGaugeRel

NAME

updateBarGaugeRel() - Change value displayed, relative.

SYNOPSIS

```
int updateBarGaugeRel
        (
        sBarParams *Bar,        /* main bar gauge structure */
        int value               /* relative change to the displayed value */
        )
```

DESCRIPTION

This function makes a relative change to the displayed value. This function must be called after initBarGauge() and drawBarGauge(). If the surrounding parts of the gauge need to be redrawn, call drawBarGauge(). Positive values increase the displayed value, negative values decrease the displayed value.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

bargauge.h, gauge.h

SEE ALSO

Bar Gauge Construction, initBarGauge(), drawBarGauge(), updateBarGauge(), getBarGaugeWidth(), getBarGaugeHeight(), copyBarGauge()

getBarGaugeWidth

NAME

getBarGaugeWidth() - Get the overall width of the gauge.

SYNOPSIS

```
int getBarGaugeWidth
        (
        sBarParams *Bar,        /* main bar gauge structure */
        int *W                  /* address of where to put the width */
        )
```

DESCRIPTION

This function gets the overall width of the gauge in pixels. This function must be called after initBarGauge(). It is common to call getBarGaugeWidth() before drawBarGauge() to center the gauge left to right.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

bargauge.h, gauge.h

SEE ALSO

Bar Gauge Construction, initBarGauge(), drawBarGauge(), updateBarGauge(), updateBarGaugeRel(), getBarGaugeHeight(), copyBarGauge()

getBarGaugeHeight

NAME

getBarGaugeHeight() - Get the overall height of the gauge.

SYNOPSIS

```
int getBarGaugeHeight
        (
        sBarParams *Bar,        /* main bar gauge structure */
        int *H                  /* address of where to put the height */
        )
```

DESCRIPTION

This function gets the overall height of the gauge in pixels. This function must be called after initBarGauge(). It is common to call getBarGaugeHeight() before drawBarGauge() to center the gauge top to bottom.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

bargauge.h, gauge.h

SEE ALSO

Bar Gauge Construction, initBarGauge(), drawBarGauge(), updateBarGauge(), updateBarGaugeRel(), getBarGaugeWidth(), copyBarGauge()

copyBarGauge

NAME

copyBarGauge() - Copy the Bar gauge structure.

SYNOPSIS

```
int copyBarGauge
        (
        sBarParams *src,        /* source bar gauge structure */
        sBarParams *dst,        /* destination bar gauge structure */
        )
```

DESCRIPTION

This function copies the Bar gauge structure. This is the only function that may be called before or after initBarGauge(). If it is called before initBarGauge(), then the internal structures of dst have not yet been initialized. Therefore, initBarGauge() must be called for both src and dst. if copyBarGauge() is called after initBarGauge() for src, then all internal structures have been filled out and initBarGauge() does not need to be called for dst.

Note: after copyBarGauge() is called, both src and dst point to the same structures. If the destination gauge is going to have different values for any member of these structures, then that structure pointer should be reassigned to a unique structure.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

bargauge.h, gauge.h

SEE ALSO

Bar Gauge Construction, initBarGauge(), drawBarGauge(), updateBarGauge(), updateBarGaugeRel(), getBarGaugeWidth(), getBarGaugeHeight()

Bar Gauge Structures

See the example for usage.

sBarParams Structure

The first three members of the *sBarParams* structure are set by the user. The remaining members are updated by ***initBarGauge()***.

Type	Members	Description
typedef struct		
{		
void	*background;	See sRectBackground and sRoundBackground
int	pixelPerMinorMark;	The number of pixels between minor gradation marks
sGradMarks	*grad;	See sGradMarks
sGradMarkText	*font;	See sGradMarkText
sBarIn	*in;	See sBarIn
sGaugeRange	*values;	See sGaugeRange
int	type;	One of the defined constants VFILLGAUGE or HFILLGAUGE. Determines whether the gauge is vertical or horizontal
sPoint	xy;	Set by ***initBarGauge() - Used and set internally***
sPoint	dimensions;	Set by ***initBarGauge()- Used and set internally***
sRectangle	update;	Set by ***initBarGauge()- Used and set internally***
int	oldValue;	Set by ***initBarGauge()- Used and set internally***
} sBarParams;		

sRectBackground Structure

The *sRectBackground* structure contains information about the rectangular bar gauge exterior.

Type	Members	Description
typedef struct		
{		
int	type;	1=RECTANGLE
int	borderLeft;	Size of left border
int	borderTop;	Size of top border
int	borderRight;	Size of right border
int	borderBottom;	Size of bottom border
unsigned long	bodyColor;	Gauge body color
unsigned long	outlineColor;	Color for line around gauge
int	mitreWidth;	Top, bottom & sides
unsigned long	mitreTopColor;	Top and left side
unsigned long	mitreBottomColor;	Bottom & right side
}sRectBackground;		

sGradMarks Structure

The *sGradMarks* structure contains information about the grad marks.

Type	Members	Description
typedef struct		
{		
int	minorPerMedium;	Number of minor marks between medium marks
int	mediumPerMajor;	Number of medium marks between major marks
int	majorPerGauge;	Number of major marks for the gauge
int	lengthMajor;	Length of major marks
int	lengthMedium;	Length of mediummarks
int	lengthMinor;	Length of minor marks
unsigned long	color;	Color of the marks
int	direction;	IN or OUT
}sGradMarks;		

sGradMarkText Structure

The *sGradMarkText* structure contains information about the grad mark text.

Type	Members	Description
typedef struct		
{		
int	number;	Font number or name
unsigned long	color;	Color of the text
char	*text;	Insert text. Use '|' between strings, i.e. "Cold|Hot"
int	xOffset;	Distance to move text
int	yOffset;	Distance to move text
}sGradMarkText;		

sBarIn Structure

The *sBarIn* structure contains the colors, cutoff values, and information about the interior of the bar gauge.

Type	Members	Description
typedef struct		
{		
int	thickness;	The width of the bar
unsigned long	color1;	First color
unsigned long	color2;	Second color
unsigned long	color3;	Third color
unsigned long	color4;	Fourth color
int	cutoff1;	Value where color changes between color 1 and color2
int	cutoff2;	Value where color changes between color 2 and color3
int	cutoff3;	Value where color changes between color 3 and color4
unsigned long	outlineColor;	Outline color
unsigned long	backgroundColor;	Background color
unsigned long	miterTopColor;	Top and left miter color
unsigned long	miterBottomColor;	Bottom and right miter color
int	miterWidth;	Width of miter area
} sBarIn;		

sGaugeRange Structure

The *sGaugeRange* structure contains the minimum and maximum allowed update values.

Type	Members	Description
typedef struct		
{		
int	min;	minimum allowed update value
int	max;	maximum allowed update value
} sGaugeRange		

Bar Gauge Example

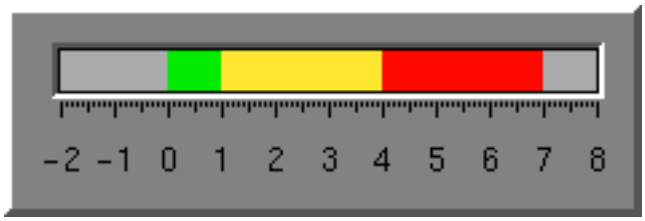
This Example show how to use the Bar gauge. The Bar gauge uses the following SDL functions.

```
drawText              getTextWidth      setForeground
filledCircle          line              setOrigin
filledRectangle       rectangle         setTransparency
getFontStruct         setFont           setPixelProcessing
```

This example has two functions: main() and setupBar(). The initialization, execution loop and cleanup are in main(). The structures are filled out in setupBar().

The function main() does the following:

- Initialize graphics.
- Call setupBar().
- Loop
  - Update gauge
  - Delay
- Free memory allocated for structures.
- Close Graphics.



The function setupBar() does the following:

- Memory allocation for structures.
- The background structure is filled out.
- The interior structure is filled out.
- The gradation mark structure is filled out.
- The gradation mark text structure is filled out.
- The gaugeRange structure is filled out.
- The gauge is specified to be a horizontal filled bar gauge.
- The gauge is initialized.
- The gauge is queried for width and height.
- The gauge is drawn centered on the screen.

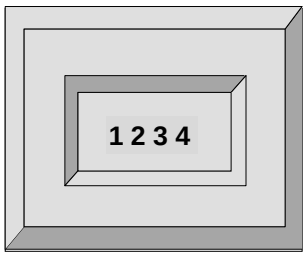
```
/*****************************************************************************/
/*                        STANDARD DRAWING LIBRARY                           */
/*                                                                           */
/*                       Rastergraf, Inc.                                    */
/*       Used under license from CURTISS-WRIGHT CONTROLS, INC.               */
/*       COPYRIGHT (C) 2001 CURTISS-WRIGHT CONTROLS, INC.                    */
/*                                                                           */
/*  This software is licensed software subject to the terms of the           */
/*  Source Code License Agreement. Refer to the file LICENSE for details.    */
/*****************************************************************************/
/* FILE NAME   : exbar.c                                                     */
/* DESCRIPTION : example to use an Bar gauge                                 */
/* AUTHOR      : ldw                                                         */
/* DATE CREATED: 8/28/95                                                     */
/*****************************************************************************/
/* example on how to setup and use the Bar gauge                             */
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <sdl.h>           /* SDL prototypes and typedefs */
#include <colors.h>        /* color names                 */
#include <fonts.h>         /* font names                  */
#include <extern.h>        /* global SDL variables        */
#include <sdltimer.h>      /* define for TICK_DELAY       */
#include <bargauge.h>      /* Bar gauge                   */
```

```
/* local prototype */
void setupBar(sBarParams *bar,int width,int height);

void main(int argc, char**argv)
{
  sBarParams  bar;
  int a,i,change;

  initGraphics(argc,argv);

  /* setup the Bar gauge */
  setupBar(&bar,_maxX,_maxY);

  i=0;
  change = 2;
  for( a=0 ; a<=600 ; a++)
  {
    /* update the gauge */
    updateBarGauge(&bar,i,0 );

    i+=change;
    if(i>=80 || i<=-20)
      change = -change;

    /* short delay */
    TICK_DELAY(2);
  }

  /* clean up */
  free(bar.background);
  free(bar.grad);
  free(bar.font);
  free(bar.in);
  free(bar.values);

  closeGraphics();
}

/*****************************************************************************/

void setupBar(sBarParams *bar,int width,int height)
{
  int w,h;

  sRectBackground*  back    = (sRectBackground*)
      malloc(sizeof(sRectBackground));

  bar->grad      =(sGradMarks      *) malloc(sizeof(sGradMarks));
  bar->font      =(sGradMarkText   *) malloc(sizeof(sGradMarkText));
  bar->in        =(sBarIn          *) malloc(sizeof(sBarIn));
  bar->values    =(sGaugeRange     *) malloc(sizeof(sGaugeRange));

  /* setup background */
  back->type                = RECT;
  back->borderLeft          =   15;
  back->borderTop           =   15;
  back->borderRight         =   15;
  back->borderBottom        =   15;
  back->outlineColor        = Black;
  back->bodyColor           = Gray9;
  back->mitreWidth          =   3;
  back->mitreTopColor       = White;
  back->mitreBottomColor    = DarkGray;
```

```
  bar->background           = back;

  /* setup gauge interior */
  bar->in->thickness              =   15;
  bar->in->color1           = Green0;
  bar->in->color2           = Green0;
  bar->in->color3           = Yellow0;
  bar->in->color4           = Red7;
  bar->in->cutOff1          = -10;
  bar->in->cutOff2          =  10;
  bar->in->cutOff3          =  40;
  bar->in->outlineColor     = Black;
  bar->in->backgroundColor  = LightGray;
  bar->in->mitreTopColor    = DarkGray;
  bar->in->mitreBottomColor = White;
  bar->in->mitreWidth       =    2;

  /* setup the grad marks */
  bar->pixelPerMinorMark    =    2;
  bar->grad->minorPerMedium =    5;
  bar->grad->mediumPerMajor =    2;
  bar->grad->majorPerGauge  =   11;
  bar->grad->lengthMajor    =    5;
  bar->grad->lengthMedium   =    3;
  bar->grad->lengthMinor    =    1;
  bar->grad->color               = Black;
  bar->grad->direction      = RIGHT;

  /* setup the text */
  bar->font->text           = "-2|-1|0|1|2|3|4|5|6|7|8";
  bar->font->number              = HELVR12;
  bar->font->color               = Black;
  bar->font->xOffset             =    0;
  bar->font->yOffset        =   10;

  /* setup min and max values */
  bar->values->max          =  80;
  bar->values->min          = -20;

  /* setup direction of gauge, for a vertical gauge use VFILLGAUGE */
  bar->type = HFILLGAUGE;

  /* initialize the Bar gauge */
  initBarGauge( bar );

  /* get the size of the gauge */
  getBarGaugeWidth (bar,&w);
  getBarGaugeHeight(bar,&h);

  /* draw gauge centered on the screen */
  drawBarGauge( bar, (width-w)/2,(height-h)/2 );
}
```

Text Gauge

The *Text Gauge* displays numerical information in binary, octal, decimal, or hexadecimal format. It consists of three parts: (1) an outer rectangle, (2) an inner rectangle, and (3) text in the center of the gauge.



Text Gauge

A *Text Gauge* is created from the information in the structures shown below. Structure ***sTextParams*** is the main structure and contains structures, pointers to structures, and parameters that define the *Text Gauge*. Only the first three members (pointers) are specified by the user. The remaining members (shaded) are filled out by ***initTextGauge().***

An overview of the *Text Gauge* structures is shown below.

Structure **sTextParams**

Type	Member Name
void	*outerRect
sTextInnerRect	*innerRect
sTextFont	*font
int	type
sPoint	short xy.x short xy.y
sPoint	short dimensions.x short dimensions.y
sRectangle	short update.x short update.y short update.width short update.height
int	oldValue

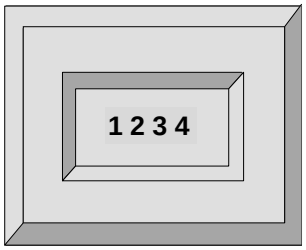
sTextOuterRect
type
borderLeft
borderTop
borderRight
borderBottom
bodyColor
outlineColor
mitreWidth
mitreTopColor
mitreBottomColor

sTextInnerRect
width
height
backgroundColor
outlineColor
mitreWidth
mitreTopColor
mitreBottomColor

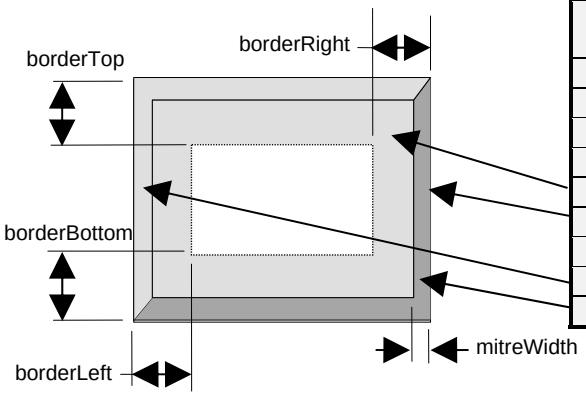
sTextFont
fontName
fgColor
numberOfChars
base
space
xOffset
yOffset

Text Gauge Construction

The *Text Gauge* is defined by the parameters contained in the structure ***sTextParams***, and by the three structures, shown below. These three structures are located by pointers contained in ***sTextParams,*** and the three structures define the three parts of the *Text Gauge*.



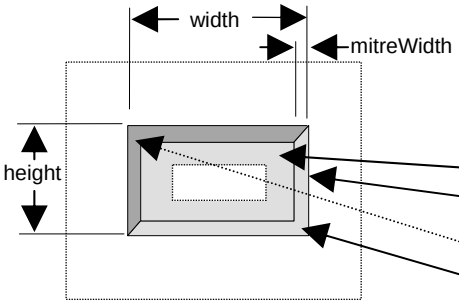
Completed Text Gauge



Outer Rectangle

Structure **sTextOuterRect**

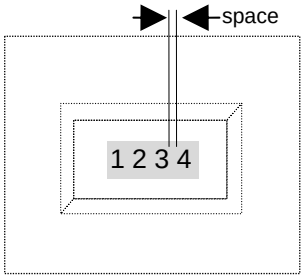
Type	Member Name	Comment
int	type	0=NONE No background 1=RECTANGLE.
int	borderLeft	See diagram at left
int	borderTop	See diagram at left
int	borderRight	See diagram at left
int	borderBottom	See diagram at left
unsigned long	bodyColor	Gauge body color
unsigned long	outlineColor	Color for line around gauge
int	mitreWidth	Top, bottom & sides
unsigned long	mitreTopColor	Top and left side color
unsigned long	mitreBottomColor	Bottom & right side color



Inner Rectangle

Structure **sTextInnerRect**

Type	Member Name	Comment
int	width	Inside rectangle width
int	height	Inside rectangle height
unsigned long	backgroundColor	Inside body color
unsigned long	outlineColor	Line around rectangle
int	mitreWidth	Top, bottom and sides
unsigned long	mitreTopColor	Top and left side color
unsigned long	mitreBottomColor	Bottom and right side color



Font for Gauge

Structure **sTextFont**

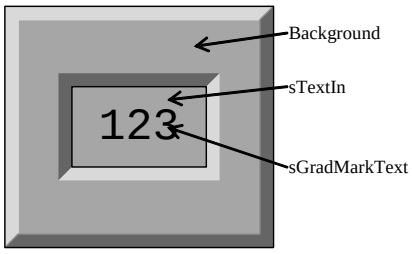
Type	Member Name	Comment
int	fontName	Font name or index value
unsigned long	fgColor	Color for font
int	numberOfChars	used for leading zeros
int	base	Radix: binary, octal, decimal, hexadecimal
int	space	Space between characters
int	xOffset	Center of text string offset in x from center of inner rect.
int	yOffset	Center of text string offset in y from center of inner rect.

Text Gauge Functions

There are seven functions that control the *Text Gauge*.

Functions	Description
int initTextGauge(sTextParams *Text);	Initialize the text gauge and internal structures.
int drawTextGauge(sTextParams *Text, int x, int y);	Draw the gauge on the screen.
int updateTextGauge(sTextParams *Text, int value);	Change the displayed value.
int updateTextGaugeRel(sTextParams *Text, int value);	Change the displayed value.
int getTextGaugeWidth (sTextParams *Text, int *W);	Get the overall width of the gauge.
int getTextGaugeHeight(sTextParams *Text, int *H);	Get the overall height of the gauge.
int copyTextGauge(sTextParams *src, sTextParams *dst);	Copy the Text gauge structures.

These functions are discussed in detail on the following pages.



Parts of the Text Gauge.

initTextGauge

NAME

initTextGauge() - Initialize the text gauge and internal structures.

SYNOPSIS

```
int initTextGauge
        (
        sTextParams *Text       /* main text gauge structure */
        )
```

DESCRIPTION

This function initializes the text gauge and internal structures. This function must be the first function called for each instance of a Text Gauge.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

textgauge.h, gauge.h

SEE ALSO

Text Gauge Construction, drawTextGauge(), updateTextGauge(), updateTextGaugeRel(), getTextGaugeWidth(), getTextGaugeHeight(), copyTextGauge()

drawTextGauge

NAME

drawTextGauge() - Draw the gauge on the screen.

SYNOPSIS

```
int drawTextGauge
        (
        sTextParams *Text,      /* main text gauge structure */
        int x,                  /* x,y placement of upper left corner of gauge */
        int y
        )
```

DESCRIPTION

This function draws the entire gauge on the screen. This function must be called after initTextGauge(). The parameters x and y specify the placement of the upper left corner of the gauge. Functions getTextWidth() and getTextHeight() may be called to determine the size of the gauge. This function only has to be called once per instance of the Text gauge, unless for some reason all or part of the gauge is erased.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

textgauge.h, gauge.h

SEE ALSO

Text Gauge Construction, initTextGauge(), updateTextGauge(), updateTextGaugeRel(), getTextGaugeWidth(), getTextGaugeHeight(), copyTextGauge()

updateTextGauge

NAME

updateTextGauge() - Change value displayed.

SYNOPSIS

```
int updateTextGauge
        (
        sTextParams *Text,      /* main text gauge structure */
        int value               /* new value to display */
        )
```

DESCRIPTION

This function changes the displayed value. This function must be called after initTextGauge() and drawTextGauge(). Only the digits that have changed will be redrawn. This minimizes the time needed to make updates to a gauge. If the surrounding parts of the gauge need to be redrawn, call drawTextGauge().

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

textgauge.h, gauge.h

SEE ALSO

Text Gauge Construction, initTextGauge(), drawTextGauge(), updateTextGaugeRel(), getTextGaugeWidth(), getTextGaugeHeight(), copyTextGauge()

updateTextGaugeRel

NAME

updateTextGaugeRel() - Change value displayed, relative.

SYNOPSIS

```
int updateTextGaugeRel
        (
        sTextParams *Text,      /* main text gauge structure */
        int value               /* relative change to the displayed value */
        )
```

DESCRIPTION

This function makes a relative change to the displayed value. This function must be called after initTextGauge() and drawTextGauge(). Only the digits that have changed will be redrawn. This minimizes the time needed to make updates to a gauge. If the surrounding parts of the gauge need to be redrawn, call drawTextGauge(). Positive values increase the displayed value, negative values decrease the displayed value.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

textgauge.h, gauge.h

SEE ALSO

Text Gauge Construction, initTextGauge(), drawTextGauge(), updateTextGauge(), getTextGaugeWidth(), getTextGaugeHeight(), copyTextGauge()

getTextGaugeWidth

NAME

getTextGaugeWidth() - Get the overall width of the gauge.

SYNOPSIS

```
int getTextGaugeWidth
        (
        sTextParams *Text,      /* main text gauge structure */
        int *W                  /* address of where to put the width */
        )
```

DESCRIPTION

This function gets the overall width of the gauge in pixels. This function must be called after initTextGauge(). It is common to call getTextGaugeWidth() before drawTextGauge() to center the gauge left to right.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

textgauge.h, gauge.h

SEE ALSO

Text Gauge Construction, initTextGauge(), drawTextGauge(), updateTextGauge(), updateTextGaugeRel(), getTextGaugeHeight(), copyTextGauge()

getTextGaugeHeight

NAME

getTextGaugeHeight() - Get the overall height of the gauge.

SYNOPSIS

```
int getTextGaugeHeight
        (
        sTextParams *Text,      /* main text gauge structure */
        int *H                  /* address of where to put the height */
        )
```

DESCRIPTION

This function gets the overall height of the gauge in pixels. This function must be called after initTextGauge(). It is common to call getTextGaugeHeight() before drawTextGauge() to center the gauge top to bottom.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

textgauge.h, gauge.h

SEE ALSO

Text Gauge Construction, initTextGauge(), drawTextGauge(), updateTextGauge(), updateTextGaugeRel(), getTextGaugeWidth(), copyTextGauge()

copyTextGauge

NAME

copyTextGauge() - Copy the Text gauge structure.

SYNOPSIS

```
int copygetTextGauge
        (
        sTextParams *src,       /* source text gauge structure */
        sTextParams *dst        /* destination text gauge structure */
        )
```

DESCRIPTION

This function copies the Text gauge structure. This is the only function that may be called before or after initTextGauge(). If it is called before initTextGauge(), then the internal structures of dst have not yet been initialized. Therefore initTextGauge() must be called for both src, and dst. if copyTextGauge() is called after initTextGauge() for src, then all internal structures have been filled out and initTextGauge() does not need to be called for dst.

Note: after copyTextGauge() is called, both src and dst point to the same sTextOuterRect, sTextInnerRect, and sTextFont structures. If the destination gauge is going to have different values for any member of these structures, then that structure pointer should be reassigned to a unique structure.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

textgauge.h, gauge.h

SEE ALSO

Text Gauge Construction, initTextGauge(), drawTextGauge(), updateTextGauge(), updateTextGaugeRel(), getTextGaugeWidth(), getTextGaugeHeight()

Text Gauge Structures

See the example for usage.

sTextParams Structure

The first three members of the *sTextParams* structure are set by the user. The remaining members are updated by ***initTextGauge()***.

Type	Members	Description
typedef struct		
{		
void	*outerRect;	See sTextOuterRect below
sTextInnerRect	*innerRect;	See sTextInnerRect below
sTextFont	*font;	See sTextFont below
int	type;	Set by ***initTextGauge() - Used and set internally***
sPoint	xy;	Set by ***initTextGauge()- Used and set internally***
sPoint	dimensions;	Set by ***initTextGauge()- Used and set internally***
sRectangle	update;	Set by ***initTextGauge()- Used and set internally***
int	oldValue;	Set by ***initTextGauge()- Used and set internally***
} sTextParams;		

sTextOuterRect Structure

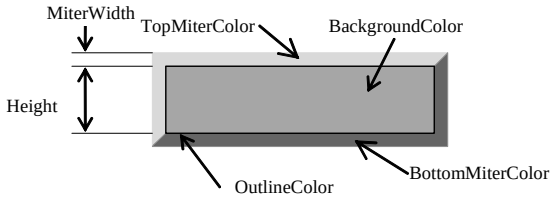
The *sTextOuterRect* structure contains information about the text gauge exterior.

Type	Members	Description
typedef struct		
{		
int	type;	1=RECTANGLE, 0 = none
int	borderLeft;	See sTextOuterRect on page 28
int	borderTop;	See sTextOuterRect on page 28
int	borderRight;	See sTextOuterRect on page 28
int	borderBottom;	See sTextOuterRect on page 28
unsigned long	bodyColor;	Gauge body color
unsigned long	outlineColor;	Color for line around gauge
int	mitreWidth;	Top, bottom & sides
unsigned long	mitreTopColor;	Top and left side
unsigned long	mitreBottomColor;	Bottom & right side
} sTextOuterRect;		

sTextInnerRect Structure

The *sTextInnerRect* structure contains information about the text gauge interior.

Type	Members	Description
typedef struct		
{		
int	width;	Inside rectangle width
int	height;	Inside rectangle height
unsigned long	outlineColor;	Line around rectangle
unsigned long	backgroundColor;	Inside body color
unsigned long	mitreTopColor;	Top and left side
unsigned long	mitreBottomColor;	Bottom and right side
int	mitreWidth;	Top, bottom and sides
} sTextInnerRect;		



sTextIn Structure Components

sTextFont Structure

The *sTextFont* structure contains information about the font to use

Type	Members	Description
typedef struct		
{		
int	fontName;	Font name or index value
unsigned long	fgColor;	Color of font
int	numberOfChars;	Used for leading zeros
int	base;	Radix: binary, octal, decimal, hexadecimal
int	space;	Space between characters
int	xOffset;	Center of text string offset in x from center of inner rect
int	yOffset;	Center of text string offset in y from center of inner rect
}sTextFont;		

Text Gauge Example

This Example show how to use the Text gauge. The Text gauge uses the following SDL functions:

```
drawText
filledCircle
filledRectangle
getFontStruct
getTextWidth
line
rectangle
setFont
setForeground
setOrigin
setTransparency
```

This is an example of how to configure and use the Text Gauge. The gauge in this example is a ten digit decimal display. This example has two functions main(), and setupText(). The initialization, execution loop and cleanup are in main(). The structures are filled out in setupText(). If this were an actual application, the for loop in main() would be the program's normal execution loop, and the call to updateTextGauge would pass valid data. This example simply counts to 10000.

The function main() does the following:

- Initialize graphics.
- Call setupText().
- Loop
  - Update gauge
  - Delay
- Free memory allocated for structures.
- Close Graphics.

The function setupText() does the following:

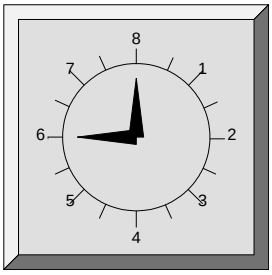
- Allocate memory for structures.
- The background structure is filled out.
- The interior structure is filled out.
- The font structure is filled out.
- The gauge is initialized.
- The gauge is queried for width and height.
- The gauge is drawn on the screen.

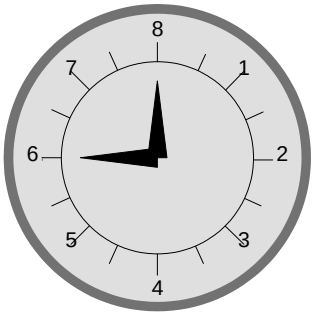
```
/*************************************************************************/
/*                        STANDARD DRAWING LIBRARY                       */
/*                                                                       */
/*                       Rastergraf, Inc.                                */
/*       Used under license from CURTISS-WRIGHT CONTROLS, INC.           */
/*       COPYRIGHT (C) 2001 CURTISS-WRIGHT CONTROLS, INC.                */
/*                                                                       */
/*  This software is licensed software subject to the terms of the       */
/*  Source Code License Agreement. Refer to the file LICENSE for details. */
/*************************************************************************/
/* FILE NAME   : extext.c                                                */
/* DESCRIPTION : example to use an Text gauge                            */
/* AUTHOR      : ldw                                                     */
/* DATE CREATED: 8/28/95                                                 */
/*************************************************************************/
/* example on how to setup and use the Text gauge                        */
/*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <sdl.h>              /* SDL prototypes and typedefs */
#include <colors.h>           /* color names                 */
#include <fonts.h>            /* font names                  */
#include <extern.h>           /* global SDL variables        */
#include <sdltimer.h>         /* define TICK_DELAY           */
#include <textgauge.h>        /* Text gauge                  */

/* local prototype */
void setupText(sTextParams *text,int width,int height);

void main(int argc, char**argv)
{
  sTextParams  text;
  int a;

  initGraphics(argc,argv);
```

```
  /* setup the Text gauge */
  setupText(&text,_maxX,_maxY);

  for( a=0 ; a<= 10000 ; a+=100)
  {
    /* update the gauge */
    updateTextGauge(&text,a );

    /* short delay */
    TICK_DELAY(2);
  }

  /* clean up */
  free(text.outerRect);
  free(text.font);
  free(text.innerRect);

  closeGraphics();
}

/*****************************************************************************/

void setupText(sTextParams *text,int width,int height)
{
  int w,h,font;
  sTextOuterRect*  back   = (sTextOuterRect*)
      malloc(sizeof(sTextOuterRect));

  text->innerRect  = (sTextInnerRect*) malloc(sizeof(sTextInnerRect));
  text->font       = (sTextFont     *) malloc(sizeof(sTextFont));

  /* setup background */
  back->type              = RECT;
  back->borderLeft        =    8;
  back->borderTop         =    8;
  back->borderRight       =    8;
  back->borderBottom      =    8;
  back->outlineColor      = Black;
  back->bodyColor         = LightGray;
  back->mitreWidth        =    2;
  back->mitreTopColor     = White;
  back->mitreBottomColor  = DarkGray;

  text->outerRect         = back;

  /* setup gauge interior */
  text->innerRect->width            =  150;
  text->innerRect->height           =   40;
  text->innerRect->outlineColor     = LightGray;
  text->innerRect->backgroundColor  = LightRed;
  text->innerRect->mitreTopColor    = DarkGray;
  text->innerRect->mitreBottomColor = White;
  text->innerRect->mitreWidth       =    2;

  /* setup the font to use */
  font = HELVR18;

  text->font->fontName       = font;
  text->font->fgColor        = White;
  text->font->xOffset        =    0;
  text->font->yOffset        =    0;
  text->font->base           = DECIMAL;
  text->font->numberOfChars  = 10;
  text->font->space          = 4;
```

```
  /* initial Text gauge */

  initTextGauge ( text );

  /* get the size of the gauge */
  getTextGaugeWidth (text,&w);
  getTextGaugeHeight(text,&h);

  /* draw the Text gauge centered on the screen */
  drawTextGauge ( text, (width-w)/2,(height- h)/2-30 );

  setFont(font);
}
```

Clock Gauge





The *Clock Gauge* displays data by pointing two needles at values on a dial similar, to a clock. It consists of three parts: (1) a background, rectangle or round (2) gradation (marks) marks, and (3) two needles.

A *Clock Gauge* is created from the information in the structures shown below. Structure ***sClockParams*** is the main structure and contains structures, pointers to structures, and parameters that define the *Clock Gauge*. Only the first seven members (pointers) are specified by the user. The remaining members (shaded) are filled out by ***initClockGauge().*** An overview of the *Clock Gauge* structures is shown below.

Structure **sClockParams**

Type	Member Name
void	*background
sClockGradMarks	*clockGrad
sGradMarks	*grad
sGradMarkText	*font
sNeedle	*hand1
sNeedle	*hand2
int	pproc
int	type
sPoint	short xy.x short xy.y
sPoint	short dimensions.x short dimensions.y
sPoint	short center.x short center.y
int	oldValue
int	reserved

sRectBackground
type
borderLeft
borderTop
borderRight
borderBottom
bodyColor
outlineColor
mitreWidth
mitreTopColor
mitreBottomColor

OR

sRoundBackground
type
radius
color
outlineColor
outlineWidth

sClockGradMarks
startAngle
endAngle
clockColor
radius

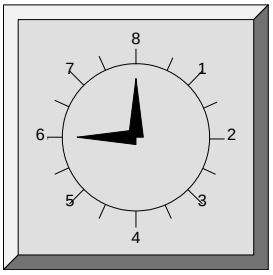
sGradMarks
minorPerMedium
mediumPerMajor
majorPerGauge
lengthMajor
lengthMedium
lengthMinor
color
direction

sGradMarkText
number
color
text
xOffset
yOffset

sNeedle
type
color
radiusInterior
radiusExterior
width

Clock Gauge Construction

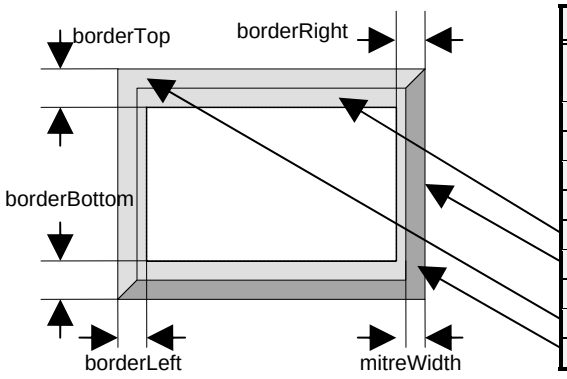
Completed Clock Gauge



The *Clock Gauge* is defined by the parameters contained in the structure ***sClockParams***, and by the six structures, shown below.

The Clock gauge can have two different types of backgrounds, either rectangular or round. Each type of background has its own structure. The background member of the ***sClockParams*** can point to either an ***sRectBackground*** or an ***sRoundBackground*** structure.

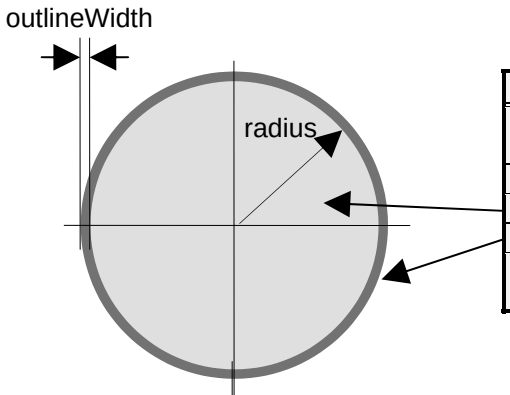
The grad marks are defined by three structures. The structure ***sClockGradMarks*** defines the end angles, color and radius of the grad marks. The grad marks are defined by the two structures ***sGradMarks*** and ***sGradMarkText,*** which are also used by some of the other Rastergraf gauges. The number, length and color of the grad marks are controlled by ***sGradMarks***. The color, position, and the text itself are defined in ***sGradMarkText.***



Outer Rectangle

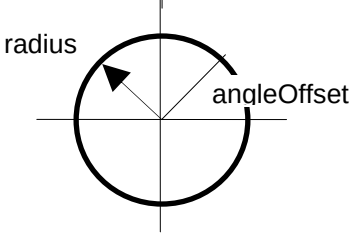
Structure **sRectBackground**

Type	Member Name	Comment
int	type	0=NONE No background 1=RECTANGLE.
int	borderLeft	See diagram at left
int	borderTop	See diagram at left
int	borderRight	See diagram at left
int	borderBottom	See diagram at left
unsigned long	bodyColor	Gauge body color
unsigned long	outlineColor	Color for line around gauge
int	mitreWidth	Top, bottom & sides
unsigned long	mitreTopColor	Top and left side color
unsigned long	mitreBottomColor	Bottom & right side color



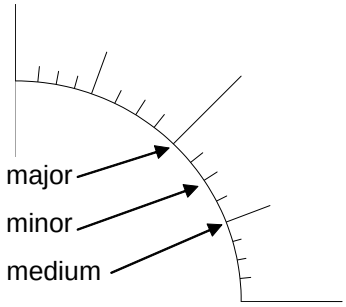
Structure **sRoundBackground**

Type	Member Name	Comment
int	type	0=NONE No background 2=ROUND
int	radius	Radius of filled circle
unsigned long	color	Gauge body color
unsigned long	outlineColor	Color of outline
int	outlineWidth	Width of outline, inside of the radius



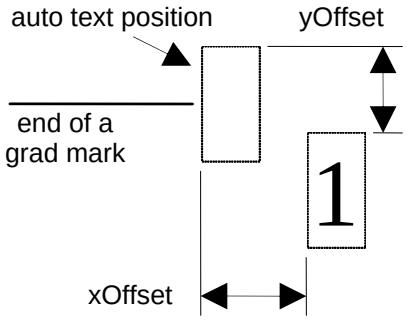
Structure **sClockGradMarks**

Type	Member	Comment
int	angleOffset	position of 0, angle measured in degrees, counter clockwise from the 3 O'clock position
unsigned long	circleColor	Color of circle
int	radius	Radius of circle



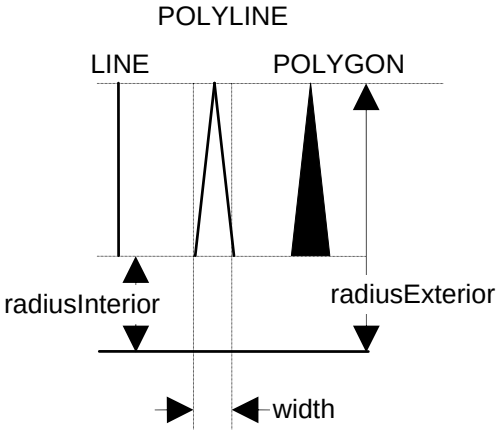
Structure **sGradMarks**

Type	Member Name	Comment
int	minorPerMedium	Number of minor marks between medium marks
int	mediumPerMajor	Number of medium marks between major marks
int	majorPerGauge	Number of major marks for the gauge
int	lengthMajor	Length of major marks
int	lengthMedium	Length of medium marks
int	lengthMinor	Length of minor marks
unsigned long	color	Color of the marks
int	direction	IN or OUT, the example to the left shows OUT.



Structure **sGradMarkText**

Type	Member Name	Comment
int	number	Font number or name
unsigned long	color	Color of the text
char	*text	Text to use, use '|' between strings, i.e. "1|2|3", or "Cold|Hot".
int	xOffset	Distance to move text.
int	yOffset	Distance to move text.



Structure **sNeedle**

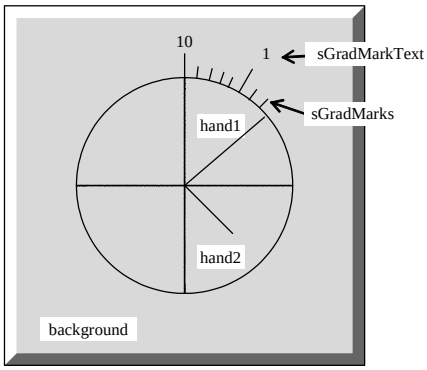
Type	Member Name	Comment
int	type	Type of Needle, LINE, POLYLINE, or POLYGON
unsigned long	color	Color of needle
int	radiusInterior	Distance from center of gauge to start of the needle
int	radiusExterior	Overall length of the needle
int	width	Width of POLYLINE, and POLYGON needles.

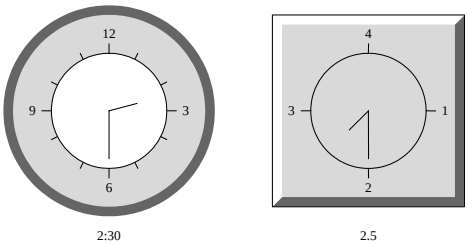
Clock Gauge Functions

There are seven functions that control the *Clock Gauge*.

Functions	Description
int initClockGauge(sClockParams *Clock);	Initialize the clock gauge and internal structures.
int drawClockGauge(sClockParams *Clock, int x, int y);	Draw the gauge on the screen.
int updateClockGauge(sClockParams *Clock, int value);	Change the displayed value.
int updateClockGaugeRel(sClockParams *Clock, int value);	Change the displayed value.
int getClockGaugeWidth (sClockParams *Clock, int *W);	Get the overall width of the gauge.
int getClockGaugeHeight(sClockParams *Clock, int *H);	Get the overall height of the gauge.
int copyClockGauge(sClockParams *src, sClockParams *dst);	Copy the Clock gauge structures.

These functions are discussed in detail on the following pages.





Both of these represent 2.5 or 900º.

initClockGauge

NAME

initClockGauge() - Initialize the clock gauge and internal structures.

SYNOPSIS

```
int initClockGauge
        (
        sClockParams *Clock        /* main clock gauge structure */
        )
```

DESCRIPTION

This function initializes the clock gauge and internal structures. This function must be the first function called for each instance of a Clock Gauge.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

clockgauge.h, gauge.h

SEE ALSO

Clock Gauge Construction, drawClockGauge(), updateClockGauge(), updateClockGaugeRel(), getClockGaugeWidth(), getClockGaugeHeight(), copyClockGauge()

drawClockGauge

NAME

drawClockGauge() - Draw the gauge on the screen.

SYNOPSIS

```
int drawClockGauge
        (
        sClockParams *Clock,   /* main clock gauge structure */
        int x,                 /* x,y placement of upper left corner of gauge */
        int y
        )
```

DESCRIPTION

This function draws the entire gauge on the screen. This function must be called after initClockGauge(). The parameters x and y specify the placement of the upper left corner of the gauge. Functions getClockGaugeWidth() and getClockGaugeHeight() may be called to determine the size of the gauge. This function only has to be called once per instance of the Clock gauge, unless for some reason all or part of the gauge is erased.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

clockgauge.h, gauge.h

SEE ALSO

Clock Gauge Construction, initClockGauge(), updateClockGauge(), updateClockGaugeRel(), getClockGaugeWidth(), getClockGaugeHeight(), copyClockGauge()

updateClockGauge

NAME

updateClockGauge() - Change value displayed.

SYNOPSIS

```
int updateClockGauge
        (
        sClockParams *Clock,   /* main clock gauge structure */
        int value              /* new value to display */
        )
```

DESCRIPTION

This function changes the displayed value. This function must be called after initClockGauge() and drawClockGauge(). The value parameter is the angular rotation of hand1 in degrees. Hand2 moves one major grad for every 360° of hand1. For example, the time 2:30 would be 2*360 + 180 = 900 etc. The maximum value depends on the number of major grad marks, for example a clock has 12 major marks, so the maximum value is 12*360 = 4320, but an altimeter has 10 major marks for 10*360 = 3600. If the surrounding parts of the gauge need to be redrawn, call drawClockGauge().

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

clockgauge.h, gauge.h

SEE ALSO

Clock Gauge Construction, initClockGauge(), drawClockGauge(), updateClockGaugeRel(), getClockGaugeWidth(), getClockGaugeHeight(), copyClockGauge()

updateClockGaugeRel

NAME

updateClockGaugeRel() - Change value displayed, relative.

SYNOPSIS

```
int updateClockGaugeRel
        (
        sClockParams *Clock,   /* main clock gauge structure */
        int value              /* relative change to the displayed value */
        )
```

DESCRIPTION

This function makes a relative change to the displayed value. This function must be called after initClockGauge() and drawClockGauge(). If the surrounding parts of the gauge need to be redrawn, call drawClockGauge(). Positive values increase the displayed value, negative values decrease the displayed value. The value parameter is the angular rotation of hand1 in degrees. Hand2 moves one major grad for every 360° of hand1.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

clockgauge.h, gauge.h

SEE ALSO

Clock Gauge Construction, initClockGauge(), drawClockGauge(), updateClockGauge(), getClockGaugeWidth(), getClockGaugeHeight(), copyClockGauge()

getClockGaugeWidth

NAME

getClockGaugeWidth() - Get the overall width of the gauge.

SYNOPSIS

```
int getClockGaugeWidth
        (
        sClockParams *Clock,   /* main clock gauge structure */
        int *W                 /* address of where to put the width */
        )
```

DESCRIPTION

This function gets the overall width of the gauge in pixels. This function must be called after initClockGauge(). It is common to call getClockGaugeWidth() before drawClockGauge() to center the gauge left to right.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

clockgauge.h, gauge.h

SEE ALSO

Clock Gauge Construction, initClockGauge(), drawClockGauge(), updateClockGauge(), updateClockGaugeRel(), getClockGaugeHeight(), copyClockGauge()

getClockGaugeHeight

NAME

getClockGaugeHeight() - Get the overall height of the gauge.

SYNOPSIS

```
int getClockGaugeHeight
        (
        sClockParams *Clock,   /* main clock gauge structure */
        int *H                 /* address of where to put the height */
        )
```

DESCRIPTION

This function gets the overall height of the gauge in pixels. This function must be called after initClockGauge(). It is common to call getClockGaugeHeight() before drawClockGauge() to center the gauge top to bottom.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

clockgauge.h, gauge.h

SEE ALSO

Clock Gauge Construction, initClockGauge(), drawClockGauge(), updateClockGauge(), updateClockGaugeRel(), getClockGaugeWidth(), copyClockGauge()

copyClockGauge

NAME

copyClockGauge() - Copy the Clock gauge structure.

SYNOPSIS

```
int copyClockGauge
        (
        sClockParams *src,      /* source clock gauge structure */
        sClockParams *dst       /* destination clock gauge structure */
        )
```

DESCRIPTION

This function copies the Clock gauge structure. This is the only function that may be called before or after initClockGauge(). If it is called before initClockGauge(), then the internal structures of dst have not yet been initialized. Therefore initClockGauge() must be called for both src, and dst. if copyClockGauge() is called after initClockGauge() for src, then all internal structures have been filled out and initClockGauge() does not need to be called for dst.

Note: after copyClockGauge() is called, both src and dst point to the same structures. If the destination gauge is going to have different values for any member of these structures, then that structure pointer should be reassigned to a unique structure.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

clockgauge.h, gauge.h

SEE ALSO

Clock Gauge Construction, initClockGauge(), drawClockGauge(), updateClockGauge(), updateClockGaugeRel(), getClockGaugeWidth(), getClockGaugeHeight()

Clock Gauge Structures

See the example for usage.

sClockParams Structure

The first three members of the *sClockParams* structure are set by the user. The remaining members are updated by ***initClockGauge()***.

Type	Members	Description
typedef struct		
{		
void	*background;	See sRectBackground and sRoundBackground
sClockGradMarks	*clockGrad;	See sClockGradMarks
sGradMarks	*grad;	See sGradMarks
sGradMarkText	*font;	See sGradMarkText
sNeedle	*hand1;	See sNeedle
sNeedle	*hand2;	See sNeedle
int	pproc;	Pixel processing on the hands, (XOR is most common
int	type;	Set by ***initClockGauge() - Used and set internally***
sPoint	xy;	Set by ***initClockGauge( )- Used and set internally***
sPoint	dimensions;	Set by ***initClockGauge( )- Used and set internally***
sPoint	center;	Set by ***initClockGauge() - Used and set internally***
int	oldValue;	Set by ***initClockGauge() - Used and set internally***
int	reserved;	Set by ***initClockGauge() - Used and set internally***
} sClockParams;		

sRectBackground Structure

The *sRectBackground* structure contains information about the rectangular clock gauge exterior.

Type	Members	Description
typedef struct		
{		
int	type;	1=RECTANGLE, 0 = none
int	borderLeft;	Size of left border
int	borderTop;	Size of top border
int	borderRight;	Size of right border
int	borderBottom;	Size of bottom border
unsigned long	bodyColor;	Gauge body color
unsigned long	outlineColor;	Color for line around gauge
int	mitreWidth;	Top, bottom & sides
unsigned long	mitreTopColor;	Top and left side color
unsigned long	mitreBottomColor;	Bottom & right side color
}sRectBackground;		

sRoundBackground Structure

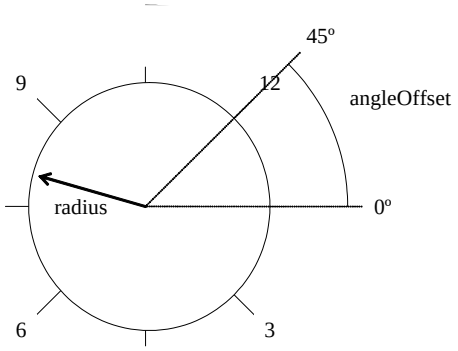
The *sRoundBackground* structure contains information about the round clock gauge exterior.

Type	Members	Description
typedef struct		
{		
int	type;	2=ROUND, 0 = none
int	radius;	Radius of filled circle
unsigned long	color;	Gauge body color
unsigned long	outlineColor;	Color of outline
int	outlineWidth;	Width of outline, inside of the radius
}sRoundBackground;		

sClockGradMarks Structure

The *sClockGradMarks* structure contains information about the clock gauge clock.

Type	Members	Description
typedef struct		
{		
int	startAngle;	Starting angle in degrees, CCW from 3 O'clock position
int	endangle;	Ending angle in degrees, CCW from 3 O'clock position
unsigned long	clockColor;	Color of clock
int	radius;	Radius of clock
}sClockGradMarks;		



sClockGradMarks

sGradMarks Structure

The *sGradMarks* structure contains information about the grad marks.

Type	Members	Description
typedef struct		
{		
int	minorPerMedium;	Number of minor marks between medium marks
int	mediumPerMajor;	Number of medium marks between major marks
int	majorPerGauge;	Number of major marks for the gauge
int	lengthMajor;	Length of major marks
int	lengthMedium;	Length of medium marks
int	lengthMinor;	Length of minor marks
unsigned long	color;	Color of the marks
int	direction;	IN or OUT
}sGradMarks;		

sGradMarkText Structure

The *sGradMarkText* structure contains information about the grad mark text.

Type	Members	Description
typedef struct		
{		
int	number;	Font number or name
unsigned long	color;	Color of the text
char	*text;	Insert text . Use '|' between strings, i.e. "Cold|Hot".
int	xOffset;	Distance to move text.
int	yOffset;	Distance to move text.
}sGradMarkText;		

sNeedle Structure

The *sNeedle* structure contains information about the needle.

Type	Members	Description
typedef struct		
{		
int	type;	Type of Needle, LINE, POLYLINE, or POLYGON
unsigned long	color;	Color of needle
int	radiusInterior;	Distance from center of gauge to start of the needle
int	radiusExterior;	Overall length of the needle
int	width;	Width of POLYLINE, and POLYGON needles
}sNeedle;		

Clock Gauge Example

This Example show how to use the Clock gauge. The Clock gauge uses the following SDL functions:

`arc`	`getFontStruct`	`setFont`
`drawText`	`getTextWidth`	`setForeground`
`filledCircle`	`line`	`setOrigin`
`filledPolygon`	`polyline`	`setTransparency`
`filledRectangle`	`rectangle`	`setPixelProcessing`

This example has four functions main(), myClock(), setupClock(), and getCurentTime(). The initialization, execution loop and cleanup are in main() and myClock(). The structures are filled out in setupClock(). The function getCurentTime() gets the current time in the format hours, minutes, and seconds.

The function main() does the following:

- Initialize graphics.
- Call myClock().
- Close Graphics.

The function myClock does the following:

- Call setupClock().
- Loop
  - Get the time
  - Update gauge
  - Delay
- Free memory allocated for structures.

The function setupClock() does the following:

- Allocate memory for structures.
- The background structure is filled out.
- The two needle structures are filled out.
- The general gradation mark structure is filled out.
- The Clock specific gradation mark structure is filled out.
- The gradation mark text structure is filled out.
- The gauge is initialized.
- The gauge is queried for width and height.
- The gauge is drawn on the screen.

The function getCurentTime() puts the current time into the format hours, minutes, and seconds. Note VxWorks uses a different function to get the current time.

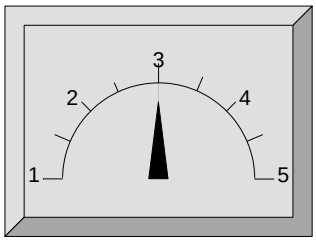
```
/*****************************************************************************/
/*                        STANDARD DRAWING LIBRARY                           */
/*                                                                           */
/*                       Rastergraf, Inc.                                    */
/*       Used under license from CURTISS-WRIGHT CONTROLS, INC.               */
/*       COPYRIGHT (C) 2001 CURTISS-WRIGHT CONTROLS, INC.                    */
/*                                                                           */
/*  This software is licensed software subject to the terms of the           */
/*  Source Code License Agreement. Refer to the file LICENSE for details. */
/*****************************************************************************/
/* FILE NAME   : exclock.c                                                   */
/* DESCRIPTION : example to use a Clock gauge                                */
/* AUTHOR      : ldw                                                         */
/* DATE CREATED: 8/28/95                                                     */
/*****************************************************************************/
/* example on how to set up the Clock gauge to show hours and minutes        */
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
```

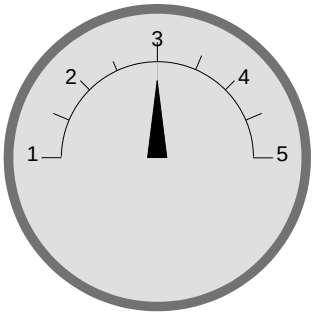
```
#include <string.h>
#include <time.h>
#include <sdl.h>           /* SDL prototypes and typedefs */
#include <colors.h>        /* color names                 */
#include <fonts.h>         /* font names                  */
#include <extern.h>        /* global SDL variables        */
#include <sdltimer.h>      /* define SLEEP                */
#include <clockgauge.h>    /* Clock gauge                 */

#ifdef VXWORKS
time_t rtc_read(struct tm *tp);
#endif

/* local prototypes */
static void myClock(int x,int y,int width,int height);
static void setupClock(sClockParams *clock,int width,int height);
static void getCurentTime(int *h,int *m, int *s);

void main(int argc, char**argv)
{
  /* start graphics */
  initGraphics(argc,argv);

  myClock(0,0,_maxX,_maxY);

  /* stop graphics */
  closeGraphics();
}
/*************************************************************************/

static void myClock(int x,int y,int width,int height)
{
  sClockParams  clock;
  int a,h,m,s;

  /* setup the Clock gauge */
  setupClock(&clock,width,height);

  for( a=0 ; a<=29 ; a++)
  {
    /* get the current time in hours,minutes and seconds */
    getCurentTime(&h,&m,&s);

    /* update gauges */
    updateClockGauge(&clock,(h*360)+(m*6)+(s/10) );
    /* 1 second delay */
    SLEEP(1);
  }

  /* clean up */
  free(clock.hand1);
  free(clock.hand2);
  free(clock.grad);
  free(clock.clockGrad);
  free(clock.font);
  free(clock.background);
}
/*************************************************************************/

static void getCurentTime(int *h,int *m, int *s)
{
  time_t timer;
  struct tm *now;

  /* get current time */
```

```
#ifdef VXWORKS
  timer = rtc_read(NULL);
#else
  timer = time(NULL);
#endif
  now = localtime(&timer);
  *h = now->tm_hour;
  *m = now->tm_min;
  *s = now->tm_sec;
}
/*************************************************************************/

static void setupClock(sClockParams *clock,int width,int height)
{
  int w,h;
  sRoundBackground* Clock_ext  = (sRoundBackground*)
      malloc(sizeof(sRoundBackground));

  /* allocate memory for structures */
  clock->hand1     = (sNeedle        *) malloc(sizeof(sNeedle));
  clock->hand2     = (sNeedle        *) malloc(sizeof(sNeedle));
  clock->grad      = (sGradMarks     *) malloc(sizeof(sGradMarks));
  clock->clockGrad = (sClockGradMarks*) malloc(sizeof(sClockGradMarks));
  clock->font      = (sGradMarkText  *) malloc(sizeof(sGradMarkText));

  /* setup a round background */
  Clock_ext->type               = ROUND;
  Clock_ext->radius             = 85;
  Clock_ext->outlineColor       = Blue4;
  Clock_ext->color              = White;
  Clock_ext->outlineWidth       = 3;

  clock->background             = Clock_ext;

  /* setup "minute hand"  */
  clock->hand1->type            = POLYGON;
  clock->hand1->width           = 10;
  clock->hand1->color                 = LightRed;
  clock->hand1->radiusExterior  =  80;
  clock->hand1->radiusInterior  =  0;

  /* setup "hour hand"  */
  clock->hand2->type            = POLYGON;
  clock->hand2->width           = 10;
  clock->hand2->color                 = LightRed;
  clock->hand2->radiusExterior  =  55;
  clock->hand2->radiusInterior  =  0;

  /* setup marks */
  clock->grad->minorPerMedium   =   5;
  clock->grad->mediumPerMajor   =   2;
  clock->grad->majorPerGauge    =  12;
  clock->grad->lengthMajor      =  10;
  clock->grad->lengthMedium     =   0;
  clock->grad->lengthMinor      =   0;
  clock->grad->color            = Black;
  clock->grad->direction        = OUT;

  /* setup Clock specific grad mark info */
  clock->clockGrad->angleOffset =  90;
  clock->clockGrad->radius      =  50;
  clock->clockGrad->circleColor = White;

  /* setup the text */
  clock->font->text             = "12| | |9| | |6| | |3"  ;
```

```
  clock->font->number           = HELVR12;
  clock->font->color            = Black;
  clock->font->xOffset          =   0;

  clock->pproc = XOR;

  /* initialize the gauge */
  initClockGauge(clock );

  /* get the size of the gauge */
  getClockGaugeWidth(clock, &w);
  getClockGaugeHeight(clock, &h);

  /* draw gauge centered on the screen */
  drawClockGauge(clock, (width-w)/2,130);
}
```

Arc Gauge





The *Arc Gauge* displays data by pointing a needle at a value on an arc. It consists of three parts; (1) a background, rectangle or round (2) gradation (grad) marks, and (3) a needle.

A *Arc Gauge* is created from the information in the structures shown below. Structure ***sArcParams*** is the main structure and contains structures, pointers to structures, and parameters that define the *Arc Gauge*. Only the first five members (pointers) are specified by the user. The remaining members (shaded) are filled out by ***initArcGauge().*** An overview of the *Arc Gauge* structures is shown below.

Structure **sArcParams**

Type	Member Name
void	*background
sArcGradMarks	*arcGrad
sGradMarks	*grad
sGradMarkText	*font
sNeedle	*needle
int	type
sPoint	short xy.x short xy.y
sPoint	short dimensions.x short dimensions.y
sPoint	short center.x short center.y
int	oldValue
int	reserved

sRectBackground
type
borderLeft
borderTop
borderRight
borderBottom
bodyColor
outlineColor
mitreWidth
mitreTopColor
mitreBottomColor

OR

sRoundBackground
type
radius
color
outlineColor
outlineWidth

sArcGradMarks
startAngle
endAngle
arcColor
radius

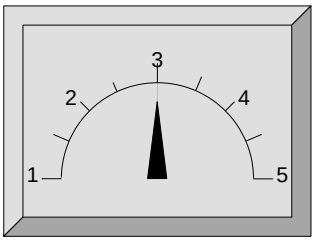
sGradMarks
minorPerMedium
mediumPerMajor
majorPerGauge
lengthMajor
lengthMedium
lengthMinor
color
direction

sGradMarkText
number
color
text
xOffset
yOffset

sNeedle
type
color
radiusInterior
radiusExterior
width

Arc Gauge Construction

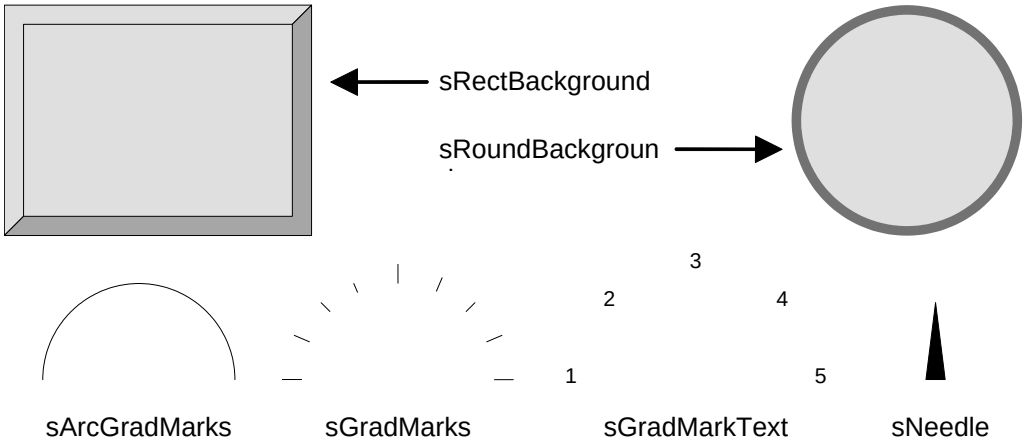
Completed Arc Gauge



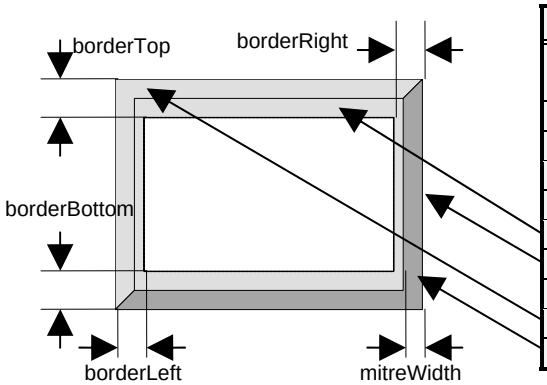
The *Arc Gauge* is defined by the parameters contained in the structure ***sArcParams***, and by the six structures, shown below.

The Arc gauge can have two different types of backgrounds, either rectangular or round. Each type of background has its own structure. The background member of the ***sArcParams*** can point to either an ***sRectBackground*** or an ***sRoundBackground*** structure.

The figure below shows all of the different parts that make up the arc gauge.

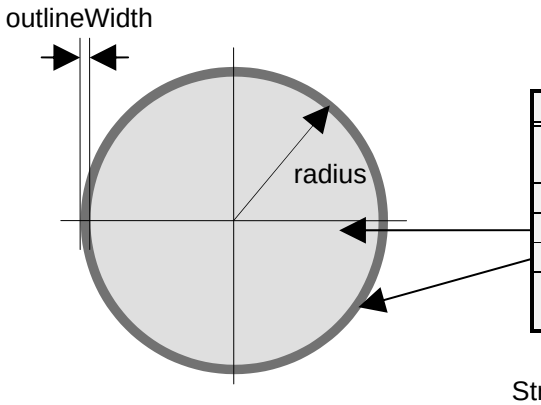


The grad marks are defined by three structures. The structure ***sArcGradMarks*** defines the end angles, color and radius of the grad marks arc. The grad marks are defined by the two structures ***sGradMarks*** and ***sGradMarkText,*** which are also used by some of the other Rastergraf gauges. The number, length and color of the grad marks are controlled by ***sGradMarks***. The color, position, and the text itself are defined in ***sGradMarkText.*** All lengths and positions are in pixels.



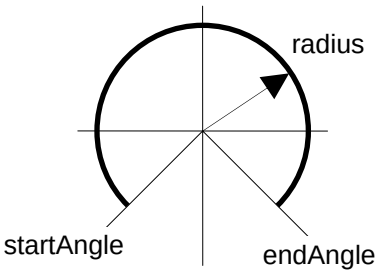
Structure **sRectBackground**

Type	Member Name	Comment
int	type	0=NONE No background 1=RECTANGLE.
int	borderLeft	See diagram at left
int	borderTop	See diagram at left
int	borderRight	See diagram at left
int	borderBottom	See diagram at left
unsigned long	bodyColor	Gauge body color
unsigned long	outlineColor	Color for line around gauge
int	mitreWidth	Top, bottom & sides
unsigned long	mitreTopColor	Top and left side
unsigned long	mitreBottomColor	Bottom & right side



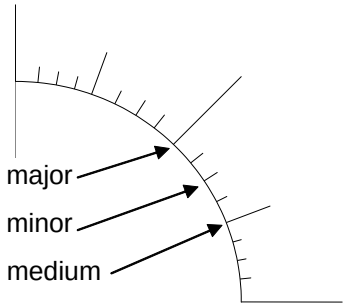
Structure **sRoundBackground**

Type	Member Name	Comment
int	type	0=NONE No background 2=ROUND
int	radius	Radius of filled circle
unsigned long	color	Gauge body color
unsigned long	outlineColor	Color of outline
int	outlineWidth	Width of outline, inside of the radius



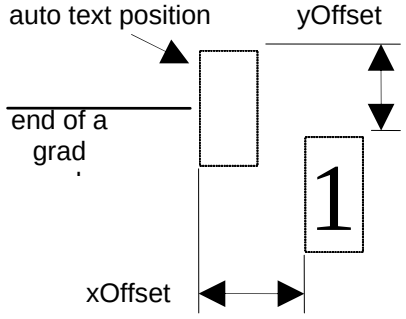
Structure **sArcGradMarks**

Type	Member	Comment
int	startAngle	Starting angle, measured in degrees, counter clock wise from the 3 O'clock position
int	endAngle	Ending angle, measured in degrees, counter clock wise from the 3 O'clock position
unsigned long	arcColor	Color of arc
int	radius	Radius of arc



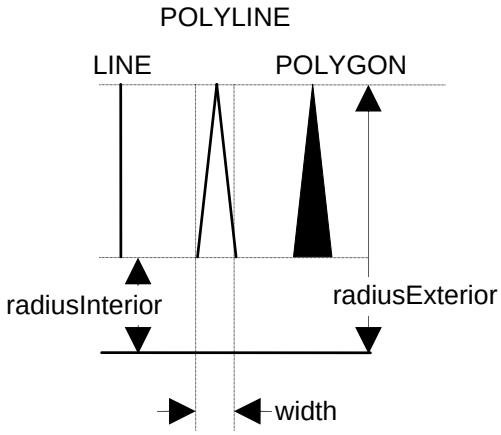
Structure **sGradMarks**

Type	Member Name	Comment
int	minorPerMedium	Number of minior marks between medium marks
int	mediumPerMajor	Number of medium marks between major marks
int	majorPerGauge	Number of major marks for the gauge
int	lengthMajor	Length of major marks
int	lengthMedium	Length of mediummarks
int	lengthMinor	Length of minor marks
unsigned long	color	Color of the marks
int	direction	IN or OUT. The example to the left shows OUT.



Structure **sGradMarkText**

Type	Member Name	Comment
int	number	Font number or name
unsigned long	color	Color of the text
char	*text	Text to use; use '|' between strings, i.e. "1|2|3", or "Cold|Hot".
int	xOffset	Distance to move text.
int	yOffset	Distance to move text.



Structure **sNeedle**

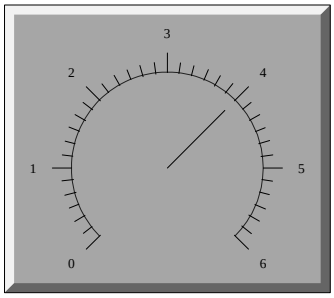
Type	Member Name	Comment
int	type	Type of Needle, LINE, POLYLINE, or POLYGON
unsigned long	color	Color of needle
int	radiusInterior	Distance from center of gauge to start of the needle
int	radiusExterior	Overall length of the needle
int	width	Width of POLYLINE, and POLYGON needles.

Arc Gauge Functions

There are seven functions that control the *Arc Gauge*.

Functions	Description
int initArcGauge(sArcParams *Arc);	Initialize the arc gauge and internal structures.
int drawArcGauge(sArcParams *Arc, int x, int y);	Draw the gauge on the screen.
int updateArcGauge(sArcParams *Arc, int value);	Change the displayed value.
int updateArcGaugeRel(sArcParams *Arc, int value);	Change the displayed value.
int getArcGaugeWidth (sArcParams *Arc, int *W);	Get the overall width of the gauge.
int getArcGaugeHeight(sArcParams *Arc, int *H);	Get the overall height of the gauge.
int copyArcGauge(sArcParams *src, sArcParams *dst);	Copy the Arc gauge structures.

These functions are discussed in detail on the following pages.



Arc

initArcGauge

NAME

initArcGauge() - Initialize the arc gauge and internal structures.

SYNOPSIS

```
int initArcGauge
        (
        sArcParams *Arc        /* main arc gauge structure */
        )
```

DESCRIPTION

This function initializes the arc gauge and internal structures. This function must be the first function called for each instance of an Arc Gauge.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

arcgauge.h, gauge.h

SEE ALSO

Arc Gauge Construction, drawArcGauge(), updateArcGauge(), updateArcGaugeRel(), getArcGaugeWidth(), getArcGaugeHeight(), copyArcGauge()

drawArcGauge

NAME

drawArcGauge() - Draw the gauge on the screen.

SYNOPSIS

```
int drawArcGauge
        (
        sArcParams *Arc,        /* main arc gauge structure */
        int x,                  /* x,y placement of upper left corner of gauge */
        int y
        )
```

DESCRIPTION

This function draws the entire gauge on the screen. This function must be called after initArcGauge(). The parameters x and y specify the placement of the upper left corner of the gauge. Functions getArcGaugeWidth() and getArcGaugeHeight() may be called to determine the size of the gauge. This function only has to be called once per instance of the Arc gauge, unless for some reason all or part of the gauge were erased.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

arcgauge.h, gauge.h

SEE ALSO

Arc Gauge Construction, initArcGauge(), updateArcGauge(), updateArcGaugeRel(), getArcGaugeWidth(), getArcGaugeHeight(), copyArcGauge()

updateArcGauge

NAME

updateArcGauge() - Change value displayed.

SYNOPSIS

```
int updateArcGauge
        (
        sArcParams *Arc,        /* main arc gauge structure */
        int value               /* new value to display */
        )
```

DESCRIPTION

This function changes the displayed value. This function must be called after initArcGauge() and drawArcGauge(). The value parameter is the number of degrees from the starting angle of the gauge. For example if the starting angle was 45, and the ending angle was 135, valid values would be 0-90. This function only redraws the needle, drawArcGauge(). should be called if the surrounding parts of the gauge need to be redrawn. For example, if a window or a message box has obscured part of the gauge.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

arcgauge.h, gauge.h

SEE ALSO

Arc Gauge Construction, initArcGauge(), drawArcGauge(), updateArcGaugeRel(), getArcGaugeWidth(), getArcGaugeHeight(), copyArcGauge()

updateArcGaugeRel

NAME

updateArcGaugeRel() - Change value displayed, relative.

SYNOPSIS

```
int updateArcGaugeRel
        (
        sArcParams *Arc,        /* main arc gauge structure */
        int value               /* relative change to the displayed value */
        )
```

DESCRIPTION

This function makes a relative change to the displayed value. This function must be called after initArcGauge() and drawArcGauge(). The value parameter is the angle of change, positive values increase the displayed value, negative values decrease the displayed value. The new value is not incremented or decremented beyond the minimum and maximum valid values. This function only redraws the needle, drawArcGauge(). should be called if the surrounding parts of the gauge need to be redrawn. For example, if a window or a message box has obscured part of the gauge.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

arcgauge.h, gauge.h

SEE ALSO

Arc Gauge Construction, initArcGauge(), drawArcGauge(), updateArcGauge(), getArcGaugeWidth(), getArcGaugeHeight(), copyArcGauge()

getArcGaugeWidth

NAME

getArcGaugeWidth() - Get the overall width of the gauge.

SYNOPSIS

```
int getArcGaugeWidth
        (
        sArcParams *Arc,        /* main arc gauge structure */
        int *W                  /* address of where to put the width */
        )
```

DESCRIPTION

This function gets the overall width of the gauge in pixels. This function must be called after initArcGauge(). It is common to call getArcGaugeWidth() before drawArcGauge() to center the gauge left to right.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

arcgauge.h, gauge.h

SEE ALSO

Arc Gauge Construction, initArcGauge(), drawArcGauge(), updateArcGauge(), updateArcGaugeRel(), getArcGaugeHeight(), copyArcGauge()

getArcGaugeHeight

NAME

getArcGaugeHeight() - Get the overall height of the gauge.

SYNOPSIS

```
int getArcGaugeHeight
        (
        sArcParams *Arc,        /* main arc gauge structure */
        int *H                  /* address of where to put the height */
        )
```

DESCRIPTION

This function gets the overall height of the gauge in pixels. This function must be called after initArcGauge(). It is common to call getArcGaugeHeight() before drawArcGauge() to center the gauge top to bottom.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

arcgauge.h, gauge.h

SEE ALSO

Arc Gauge Construction, initArcGauge(), drawArcGauge(), updateArcGauge(), updateArcGaugeRel(), getArcGaugeWidth(), copyArcGauge()

copyArcGauge

NAME

copyArcGauge() - Copy the Arc gauge structure.

SYNOPSIS

```
int copyArcGuage
        (
        sArcParams *src,        /* source arc gauge structure */
        sArcParams *dst         /* destination arc gauge structure */
        )
```

DESCRIPTION

This function copies the Arc gauge structure. This is the only function that may be called before or after initArcGauge(). If it is called before initArcGauge(), then the internal structures of dst have not yet been initialized. Therefore, initArcGauge() must be called for both src and dst. If copyArcGauge() is called after initArcGauge() for src, then all internal structures have been filled out and initArcGauge() does not need to be called for dst.

Note: after copyArcGauge() is called, both src and dst point to the same structures. If the destination gauge is going to have different values for any member of these structures, then that structure pointer should be reassigned to a unique structure.

RETURNS

0 if successful.
non-zero otherwise.

INCLUDE FILES

arcgauge.h, gauge.h

SEE ALSO

Arc Gauge Construction, initArcGauge(), drawArcGauge(), updateArcGauge(), updateArcGaugeRel(), getArcGaugeWidth(), getArcGaugeHeight()

Arc Gauge Structures

See the example for usage.

sArcParams Structure

The first five members of the *sArcParams* structure are set by the user. The remaining members are updated by ***initArcGauge()***.

Type	Members	Description
typedef struct		
{		
void	*background;	See sRectBackground and sRoundBackground
sArcGradMarks	*arcGrad;	See sArcGradMarks
sGradMarks	*grad;	See sGradMarks
sGradMarkText	*font;	See sGradMarkText
sNeedle	*needle;	See sNeedle
int	type;	Set by ***initArcGauge() - Used and set internally***
sPoint	xy;	Set by ***initArcGauge() - Used and set internally***
sPoint	dimensions;	Set by ***initArcGauge() - Used and set internally***
sPoint	center;	Set by ***initArcGauge() - Used and set internally***
int	oldValue;	Set by ***initArcGauge() - Used and set internally***
int	reserved;	Set by ***initArcGauge() - Used and set internally***
} sArcParams;		

sRectBackground Structure

The *sRectBackground* structure contains information about the rectangular arc gauge exterior.

Type	Members	Description
typedef struct		
{		
int	type;	1=RECTANGLE, 0 = none
int	borderLeft;	Size of left border
int	borderTop;	Size of top border
int	borderRight;	Size of right border
int	borderBottom;	Size of bottom border
unsigned long	bodyColor;	Gauge body color
unsigned long	outlineColor;	Color for line around gauge
int	mitreWidth;	Top, bottom & sides
unsigned long	mitreTopColor;	Top and left side color
unsigned long	mitreBottomColor;	Bottom and right side color
}sRectBackground;		

sRoundBackground Structure

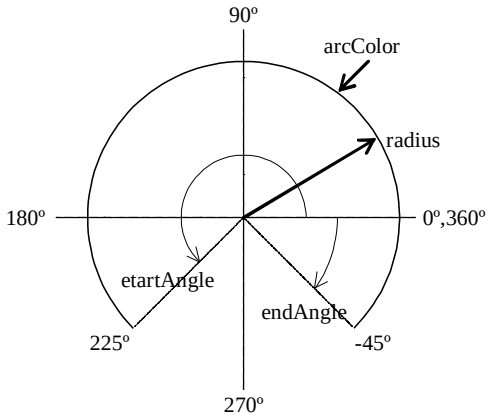
The *sRoundBackground* structure contains information about the round arc gauge exterior.

Type	Members	Description
typedef struct		
{		
int	type;	2=ROUND, 0 = none
int	radius;	Radius of filled circle
unsigned long	color;	Gauge body color
unsigned long	outlineColor;	Color of outline
int	outlineWidth;	Width of outline, inside of the radius
}sRoundBackground;		

sArcGradMarks Structure

The *sArcGradMarks* structure contains information about the arc gauge dial.

Type	Members	Description
typedef struct		
{		
int	startAngle;	Starting angle in degrees, CCW from 3 O'clock position
int	endangle;	Ending angle in degrees, CCW from 3 O'clock position
unsigned long	arcColor;	Color of dial
int	radius;	Radius of dial
}sArcGradMarks;		



sArcGradMarks

sGradMarks Structure

sGradMarks structure contains information about the grad marks.

Type	Members	Description
typedef struct		
{		
int	minorPerMedium;	Number of minor marks between medium marks
int	mediumPerMajor;	Number of medium marks between major marks
int	majorPerGauge;	Number of major marks for the gauge
int	lengthMajor;	Length of major marks
int	lengthMedium;	Length of medium marks
int	lengthMinor;	Length of minor marks
unsigned long	color;	Color of the marks
int	direction;	IN or OUT
}sGradMarks;		

sGradMarkText Structure

The *sGradMarkText* structure contains information about the grad mark text.

Type	Members	Description
typedef struct		
{		
int	number;	Font number or name
unsigned long	color;	Color of the text
char	*text;	Insert text Use '|' between strings, i.e. "Cold|Hot"
int	xOffset;	Distance to move text
int	yOffset;	Distance to move text
}sGradMarkText;		

sNeedle Structure

The *sNeedle* structure contains information about the needle.

Type	Members	Description
typedef struct		
{		
int	type;	Type of Needle, LINE, POLYLINE, or POLYGON
unsigned long	color;	Color of needle
int	radiusInterior;	Distance from center of gauge to start of the needle
int	radiusExterior;	Overall length of the needle
int	width;	Width of POLYLINE, and POLYGON needles
}sNeedle;		

Arc Gauge Example

This example shows how to use the Arc gauge. The Arc gauge uses the following SDL functions:

```
arc
drawText
filledCircle
filledPolygon
filledRectangle
getFontStruct
getTextWidth
line
polyline
rectangle
setFont
setForeground
setOrigin
setTransparency
setPixelProcessing
```

This example has two functions: main() and setupArc(). The initialization, execution loop and cleanup are in main(). The structures are filled out in setupArc().

The function main() does the following:

- Initialize graphics.
- Call setupArc().
- Loop
  - Update gauge
  - Delay
- Free memory allocated for structures.
- Close Graphics.

The function setupArc() does the following:

- Allocate memory for structures.
- The background structure is filled out.
- The needle structure is filled out.
- The general gradation mark structure is filled out.
- The Arc specific gradation mark structure is filled out.
- The gradation mark text structure is filled out.
- The gauge is initialized.
- The gauge is queried for width and height.
- The gauge is drawn centered on the screen.

```
/*****************************************************************************/
/*                        STANDARD DRAWING LIBRARY                           */
/*                                                                           */
/*                       Rastergraf, Inc.                                    */
/*       Used under license from CURTISS-WRIGHT CONTROLS, INC.               */
/*       COPYRIGHT (C) 2001 CURTISS-WRIGHT CONTROLS, INC.                    */
/*                                                                           */
/*  This software is licensed software subject to the terms of the           */
/*  Source Code License Agreement. Refer to the file LICENSE for details.    */
/*****************************************************************************/
/* FILE NAME   : exarc.c                                                     */
/* DESCRIPTION : example to use an Arc gauge                                 */
/* AUTHOR      : ldw                                                         */
/* DATE CREATED: 8/28/95                                                     */
/*****************************************************************************/
/* example on how to set up the Arc gauge                                    */
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <sdl.h>           /* SDL prototypes and typedefs */
#include <colors.h>        /* color names                 */
#include <fonts.h>         /* font names                  */
#include <extern.h>        /* global SDL variables        */
#include <sdltimer.h>      /* define for TICK_DELAY       */
#include <arcgauge.h>      /* Arc gauge                   */

/* local prototype */
void setupArc(sArcParams *arc);

void main(int argc, char**argv)
{
  sArcParams  arc;
  int a,i,change;

  initGraphics(argc,argv);

  /* setup the Arc gauge */
  setupArc(&arc);
```

```
  i=0;
  change = 2;
  for( a=0 ; a<=600 ; a++)
  {
    /* update the gauge */
    updateArcGauge(&arc,i );

    i+=change;
    if(i>=300 || i<=0)
      change = -change;

    /* short delay */
    TICK_DELAY(2);
  }

  /* clean up */
  free(arc.needle);
  free(arc.grad);
  free(arc.arcGrad);
  free(arc.font);
  free(arc.background);

  closeGraphics();
}

/*************************************************************************/

void setupArc(sArcParams *arc)
{
  int w,h;
  sRoundBackground* arc_ext  = (sRoundBackground*)
      malloc(sizeof(sRoundBackground));

  arc->needle  =(sNeedle      *) malloc(sizeof(sNeedle));
  arc->grad    =(sGradMarks   *) malloc(sizeof(sGradMarks));
  arc->arcGrad =(sArcGradMarks*) malloc(sizeof(sArcGradMarks));
  arc->font    =(sGradMarkText*) malloc(sizeof(sGradMarkText));

  /* setup a round background */
  arc_ext->type               = ROUND;
  arc_ext->radius             = 85;
  arc_ext->outlineColor       = Yellow;
  arc_ext->color              = Black;
  arc_ext->outlineWidth       = 3;

  arc->background             = arc_ext;

  /* setup needle type,size and color */
  arc->needle->type           = POLYGON;
  arc->needle->width          = 10;
  arc->needle->color          = LightRed;
  arc->needle->radiusInterior = 0;
  arc->needle->radiusExterior = 70;

  /* setup gradation marks */
  arc->grad->minorPerMedium   =   5;
  arc->grad->mediumPerMajor   =   2;
  arc->grad->majorPerGauge    =  11;
  arc->grad->lengthMajor      =  12;
  arc->grad->lengthMedium     =   6;
  arc->grad->lengthMinor      =   0;
  arc->grad->color            = LightCyan;
  arc->grad->direction        = IN;
```

```
  /* setup the arc's range and color */
  arc->arcGrad->startAngle    = 240;
  arc->arcGrad->endAngle      = -60;
  arc->arcGrad->radius        =  70;
  arc->arcGrad->arcColor      = LightCyan;

  /* setup text,font and color for major grad marks */
  arc->font->text             = "0|10|20|30|40|50|60|70|80|90|100" ;
  arc->font->number           = HELVR12;
  arc->font->color            = LightGreen;
  arc->font->xOffset          =   0;

  /* initialize the gauge */
  initArcGauge     ( arc );

  /* get the size of the gauge */
  getArcGaugeWidth ( arc, &w);
  getArcGaugeHeight( arc, &h);

  /* draw gauge centered on the screen */
  drawArcGauge     ( arc, (_maxX-w)/2,(_maxY-h)/2 );
}
```

Backgrounds

A background is a colored area which borders the update region of a gauge. It provides a place to put gradation marks, gradation text, and titles. And it adds some style. The gauge library supports two different types of backgrounds: rectangle with a beveled edge, and round. If none of these will work for you, you can choose to have no background, and then draw your own.

sRectBackground

A rectanguluar background with a beveled edge. This background wraps around the gauge, therefore the size of this background is the size of the gauge plus the size of the border.

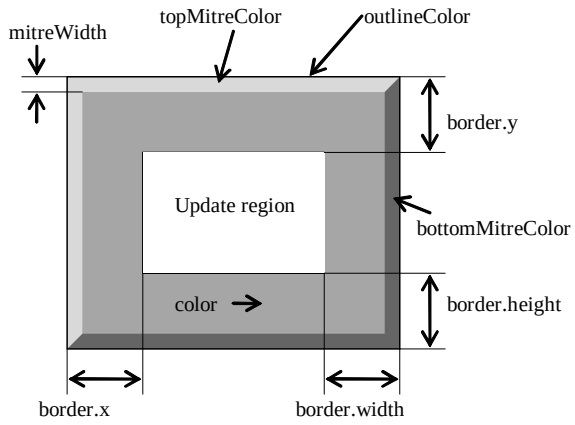
BACKGROUND_TYPE type - This is an enumerated typedef defined as follows:

typedef enum{ RG_ROUND, RG_RECT, RG_RRECT, RG_NONE } BACKGROUND_TYPE;

This is used by the gauge to determine which type of background to draw. For sRectBackground this should be set to RG_RECT.

sRectangle border - This is the number pixels used for a border around the gauge. Values between 10 and 50 are common.

border.x - size of left border.
border.y - size of top border.
border.width - size of right border.
border.height - size of bottom border.



sRectBackground Structure.

unsigned long color - Color to fill background of gauge.

unsigned long outlineColor - This is the color for the one pixel outline around the outside of the rectangle background.

int miterWidth - This is the width of the beveled edge around the outside of the rectangle background. Values between 2 and 6 are common.

unsigned long miterTopColor - Color of the top bevel.

unsigned long miterBottomColor - Color of the bottom bevel.

Type	Name	Min	Max	Comment
BACKGROUND_TYPE	type	RG_RECT		
sRectangle	border	0,0,0,0		
unsigned long	color	0	card dependent	
unsigned long	outlineColor	0	card dependent	
int	mitreWidth	0		
unsigned long	mitreTopColor	0	card dependent	unused if mitreWidth = 0
unsigned long	mitreBottomColor	0	card dependent	unused if mitreWidth = 0

sRoundBackground

A round background. The size of this background is determined entirely by member *radius*.

BACKGROUND_TYPE type - This is an enumerated typedef defined as follows:

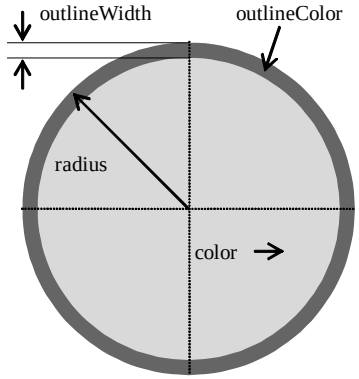
typedef enum{ RG_ROUND, RG_RECT, RG_RRECT, RG_NONE } BACKGROUND_TYPE;

This is used by the gauge to determine which type of background to draw. For sRoundBackground this should be set to RG_ROUND.

int radius - This is the distance from the center to the outside of the Outline.

unsigned long color - Color to fill background of gauge.

unsigned long outlineColor - This is the color for the outline around the outside of the round rectangle background.



sRoundBackground Structure

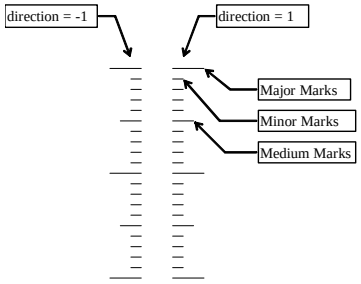
int outlineWidth - This is the width of the outline around the outside of the round rectangle background. Values between 2 and 6 are common.

Type	Name	Min	Max	Coment
BACKGROUND_TYPE	type	RG_ROUND	RG_ROUND	
int	radius	0	384	
unsigned long	color	0	card dependent	
int	outlineWidth	0	192	
unsigned long	outlineColor	0	card dependent	unused if outlineWidth = 0

Common Structures

There are four structures that are common to the different gauges. They are sGradMarks, sGradMarkText, sNeedle, and sGaugeRange. Not all members of the structures are used by all gauges. The ones that apply only to certain gauges are marked as such.

sGradMarks



Grad Marks

sGradMarks	
minorPerMedium	The number of short marks between medium marks.
mediumPerMajor	The number of medium length marks between long marks.
majorPerGauge	The total number of long marks for the entire gauge.
lengthMajor	The length of the long marks.
lengthMedium	The length of the center marks.
lengthMinor	The length of the short marks.
color	Color of marks.
direction	Direction of marks.

int majorPerGauge - The total number of long gradation marks for the entire gauge.

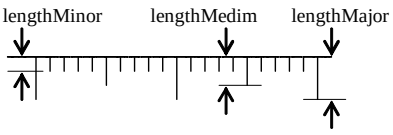
int minorPerMedium - The number of medium length marks between long marks. Values between 1 and 5 are common

int minorPerMedium - The number of short marks between medium marks. Values between 1 and 5 are common.

int lengthMajor - The length of the long gradation marks. Values between 5 and 15 are common.

int lengthMedium - The length of the center gradation marks. Values between 2 and 10 are common.

int lengthMinor- The length of the short gradation marks. Values between 1 and 5 are common.



sGradMarks.

unsigned long color - The color of all gradation marks.

int direction - The direction of grad marks [+1/-1]. The direction is shown below.

Gauge type	Direction =-1	Direction =1
Clock, Arc	in	out
Bar – HFILLGAUGE	up	down
Bar – VFILLGAUGE	left	right

Type	Name	Min	Max	Required	Comment
int	minorPerMedium	1		yes	
int	mediumPerMajor	1		yes	
int	majorPerGauge	1		yes	
int	lengthMajor	0		yes	
int	lengthMedium	0		yes	
int	lengthMinor	0		yes	
unsigned long	color	0	card dependent	yes	
int	direction	-1	+1	yes	

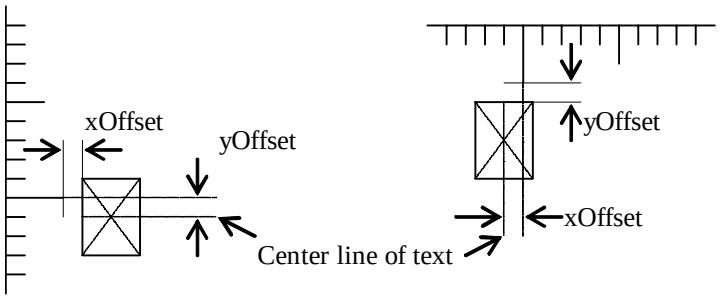
sGradMarkText

int number - Font number.

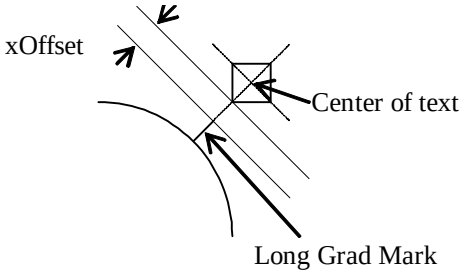
unsigned long color - Color of font.

char *text - Text for gradation marks in the format “1|2|2|3|4|5”. This can be set to NULL or “” for no text. Not used in Text type gauges.

int xOffset, yOffset - Position of text relative the end of a long gradation mark. yOffset is not used for Clock and Arc type gauges.



xOffset and yOffset for Bar gauges.



xOffset for Clock and Arc gauges.

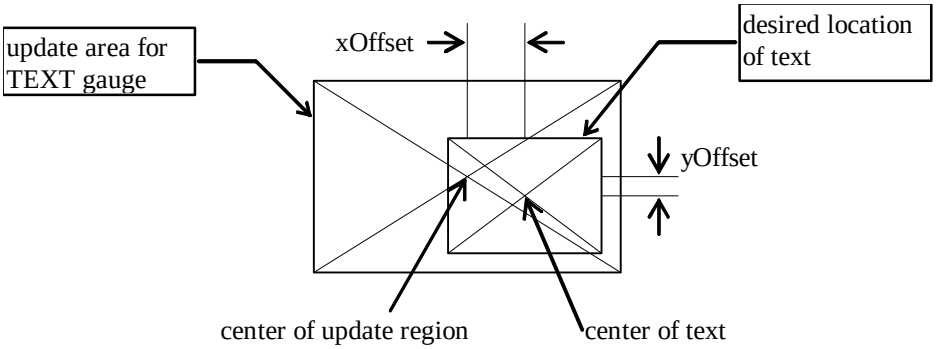


Figure 1: xOffset and yOffset for Text gauges.

sGaugeRange

int max - The max value for update. For Clock and Arc type gauges [0..+-360].

int min - The min value for update. For Clock and Arc type gauges [0..+-360]. Also for Clock and Arc type gauges the delta between Max and Min <=360.

sNeedle

The sNeedle structure holds information about the needle used for Clock and Arc type gauges.

unsigned long color - Color of Needle.

int interiorRadius - Distance from center point to beginning of the needle.
int exteriorRadius - Distance from center point to ending of the needle.

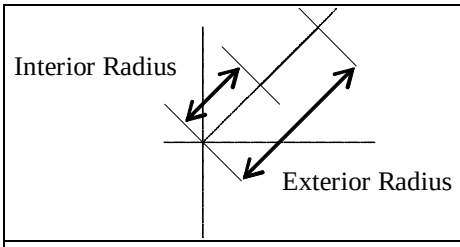


Figure 2: sNeedle Structure.