

Stopwatch Hand-coded

Generated by Doxygen 1.8.11

Contents

1	Module Index	1
1.1	Modules	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
4.1	Utility	7
4.1.1	Detailed Description	7
4.1.2	Function Documentation	7
4.1.2.1	activateAlarm()	7
4.1.2.2	activateSwatch()	8
4.1.2.3	activateTimer()	8
4.1.2.4	disableAlarm()	8
4.1.2.5	disableSwatch()	9
4.1.2.6	disableTimer()	9
4.1.2.7	strencode1digit(char *str, int digit)	9
4.1.2.8	strencode2digit(char *str, int digit)	9
4.1.2.9	updateScreen(uint8_t om, uint8_t m)	10
4.1.2.10	updateTime(uint8_t *oh, uint8_t *om, uint8_t *os, uint8_t *ot, uint8_t oldmode)	10
4.2	Interrupt Handler	11
4.2.1	Detailed Description	11

4.3	Tasks	12
4.3.1	Detailed Description	12
4.3.2	Function Documentation	12
4.3.2.1	main(void)	12
4.3.2.2	TASK(TaskLCD)	12
4.3.2.3	TASK(TaskSwatch)	12
4.3.2.4	TASK(TaskAlarm)	13
4.3.2.5	TASK(TaskTimer)	13
4.3.2.6	TASK(TaskFSM)	13
4.4	Widget	14
4.4.1	Detailed Description	15
4.4.2	Function Documentation	15
4.4.2.1	contains(Widget *w, TPoint *point)	15
4.4.2.2	DrawInit(Widget ws[])	15
4.4.2.3	DrawOff(Widget *w)	16
4.4.2.4	DrawOn(Widget *w)	16
4.4.2.5	OnTouch(const Widget ws[], TPoint *press)	16
4.4.2.6	WPrint(Widget *w, char *s)	17
4.5	Widget Definitions	18
4.5.1	Detailed Description	18
4.5.2	Variable Documentation	18
4.5.2.1	alarm_b	18
4.5.2.2	alarm_exp_i	18
4.5.2.3	backg	19
4.5.2.4	hrs_back	19
4.5.2.5	min_back	19
4.5.2.6	minus_b	19
4.5.2.7	MyWatchScr	19
4.5.2.8	plus_b	20
4.5.2.9	reset_b	20

4.5.2.10	resume_b	20
4.5.2.11	sec_back	20
4.5.2.12	set_b	20
4.5.2.13	start_b	20
4.5.2.14	stop_b	21
4.5.2.15	swatch_b	21
4.5.2.16	timer_b	21
4.5.2.17	timer_exp_i	21
4.5.2.18	tts_back	21
4.5.2.19	txt	21
4.5.2.20	watch_b	21
4.6	Events	22
4.6.1	Detailed Description	22
4.6.2	Macro Definition Documentation	22
4.6.2.1	ClearEvt	22
4.6.2.2	IsEvent	22
4.6.2.3	SetEvt	23
4.7	FSM Definition	24
4.7.1	Detailed Description	24
4.8	Types	25
4.8.1	Detailed Description	25
5	Data Structure Documentation	27
5.1	Icon Struct Reference	27
5.2	Image Struct Reference	27
5.3	Text Struct Reference	27
5.4	time_ Struct Reference	28
5.4.1	Detailed Description	28
5.5	Widget Struct Reference	28

6 File Documentation	29
6.1 code.c File Reference	29
6.1.1 Detailed Description	30
6.1.2 Variable Documentation	31
6.1.2.1 timer_exp	31
6.2 Cplus.h File Reference	31
6.2.1 Detailed Description	32
6.2.2 Macro Definition Documentation	32
6.2.2.1 CLASS	32
6.2.2.2 SUBCLASS	32
6.3 Event.c File Reference	32
6.3.1 Detailed Description	33
6.4 Event.h File Reference	33
6.4.1 Detailed Description	33
6.5 mypictures.c File Reference	34
6.5.1 Detailed Description	34
6.6 mypictures.h File Reference	34
6.6.1 Detailed Description	35
6.7 SWatchFSM.c File Reference	35
6.7.1 Detailed Description	36
6.7.2 Function Documentation	36
6.7.2.1 SWatchFSMdispatch(SWatchFSM *me, Signal sig)	36
6.7.2.2 SWatchFSMinit(SWatchFSM *me)	37
6.7.2.3 tran_(SWatchFSM *me, State dest)	37
6.7.3 Variable Documentation	37
6.7.3.1 timer_exp	37
6.8 SWatchFSM.h File Reference	37
6.8.1 Detailed Description	38
6.8.2 Function Documentation	38
6.8.2.1 SWatchFSMdispatch(SWatchFSM *me, Signal sig)	38
6.8.2.2 SWatchFSMinit(SWatchFSM *me)	39
6.9 types.h File Reference	39
6.9.1 Detailed Description	40
6.10 Widget.c File Reference	40
6.10.1 Detailed Description	41
6.11 Widget.h File Reference	41
6.11.1 Detailed Description	43
Index	45

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Utility	7
Interrupt Handler	11
Tasks	12
Widget	14
Widget Definitions	18
Events	22
FSM Definition	24
Types	25

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Icon	27
Image	27
Text	27
time_	
	Data structure containing timing information	28
Widget	28

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

code.c	Contains the body of all tasks and the global variables defined	29
Cplus.h	Macros for using class-like semantics in C	31
Event.c	Contains the event mask definition	32
Event.h	Contains the macros used to handle the event masks	33
mypictures.c	This file contains the application pictures in RGB565 format	34
mypictures.h	Pictures header file	34
SWatchFSM.c	Contains the nested switch implementation of the FSM	35
SWatchFSM.h	Contains the definition of the FSM and the definitions of its signals and states	37
types.h	Type definitions	39
Widget.c	Contains the functions to manage the widgets on the screen	40
Widget.h	Contains the type definitions and the macros used for the screen widgets	41

Chapter 4

Module Documentation

4.1 Utility

Functions

- static void [strencode1digit](#) (char *str, int digit)
Converts a one digit integer into a string.
- static void [strencode2digit](#) (char *str, int digit)
Converts a two digits integer into a string.
- void [activateSwatch](#) ()
Activates the Stopwatch task.
- void [activateAlarm](#) ()
Activates the Alarm task.
- void [activateTimer](#) ()
Activates the Timer task.
- void [disableAlarm](#) ()
Terminates the Alarm task.
- void [disableTimer](#) ()
Terminates the Timer task.
- void [disableSwatch](#) ()
Terminates the Stopwatch task.
- static void [updateTime](#) (uint8_t *oh, uint8_t *om, uint8_t *os, uint8_t *ot, uint8_t oldmode)
Updates the time on the screen.
- void [updateScreen](#) (uint8_t om, uint8_t m)
Updates the screen widgets.

4.1.1 Detailed Description

4.1.2 Function Documentation

4.1.2.1 void activateAlarm ()

Activates the Alarm task.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

4.1.2.2 void activateSwatch ()

Activates the Stopwatch task.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

4.1.2.3 void activateTimer ()

Activates the Timer task.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

4.1.2.4 void disableAlarm ()

Terminates the Alarm task.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

4.1.2.5 void disableSwatch ()

Terminates the Stopwatch task.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

4.1.2.6 void disableTimer ()

Terminates the Timer task.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

4.1.2.7 static void strencode1digit (char * *str*, int *digit*) [static]

Converts a one digit integer into a string.

Parameters

<i>str</i>	pointer to the returning string.
<i>digit</i>	integer digit to be converted.

Return values

<i>None</i>	
-------------	--

4.1.2.8 static void strencode2digit (char * *str*, int *digit*) [static]

Converts a two digits integer into a string.

Parameters

<i>str</i>	pointer to the returning string.
<i>digit</i>	integer digits to be converted.

Return values

<i>None</i>	
-------------	--

4.1.2.9 void updateScreen (uint8_t *om*, uint8_t *m*)

Updates the screen widgets.

Parameters

<i>om</i>	Old application mode.
<i>m</i>	New application mode.

Return values

<i>None</i>	
-------------	--

4.1.2.10 static void updateTime (uint8_t * *oh*, uint8_t * *om*, uint8_t * *os*, uint8_t * *ot*, uint8_t *oldmode*) [static]

Updates the time on the screen.

Parameters

<i>oh</i>	Old hours.
<i>om</i>	Old minutes.
<i>os</i>	Old seconds.
<i>ot</i>	Old tenths.
<i>oldmode</i>	Old application mode.

Return values

<i>None</i>	
-------------	--

4.2 Interrupt Handler

Functions

- [ISR2](#) (systick_handler)
System Tick interrupt handler.

4.2.1 Detailed Description

4.3 Tasks

Functions

- **TASK** (TaskLCD)
LDC task body.
- **TASK** (TaskWatch)
Implements the watch mode.
- **TASK** (TaskSwatch)
Implements the Stopwatch mode.
- **TASK** (TaskAlarm)
Implements the Alarm mode.
- **TASK** (TaskTimer)
Implements the Timer mode.
- **TASK** (TaskFSM)
Implements the State Machine of the application.
- `int main` (void)
Main task of the application.

4.3.1 Detailed Description

4.3.2 Function Documentation

4.3.2.1 `int main (void)`

Main task of the application.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	This function should never return.
-------------	------------------------------------

4.3.2.2 `TASK (TaskLCD)`

LDC task body.

This task is periodically activated in order to get the touch events.

4.3.2.3 `TASK (TaskSwatch)`

Implements the Stopwatch mode.

This task is activated by the FSM when the Stopwatch is started.

4.3.2.4 TASK (TaskAlarm)

Implements the Alarm mode.

This task is activated by the FSM when the alarm time is set.

4.3.2.5 TASK (TaskTimer)

Implements the Timer mode.

This task is activated by the FSM when the timer is started.

4.3.2.6 TASK (TaskFSM)

Implements the State Machine of the application.

This task checks whether an event has occurred and dispatches the right signal to the FSM.

4.4 Widget

Modules

- [Widget Definitions](#)

Data Structures

- struct [Image](#)
- struct [Icon](#)
- struct [Text](#)
- struct [Widget](#)

Macros

- `#define NUMWIDGETS 25`
- `#define BAKCG 0`
- `#define BWATCH 1`
- `#define BSWATCH 2`
- `#define BALARM 3`
- `#define BTIMER 4`
- `#define BPLUS 5`
- `#define BMINUS 6`
- `#define BSTART 7`
- `#define BSET 8`
- `#define BRESUME 9`
- `#define BSTOP 10`
- `#define BRESET 11`
- `#define ALARMEXP 12`
- `#define TIMEREXP 13`
- `#define HRSSTR 14`
- `#define MINSTR 15`
- `#define SECSTR 16`
- `#define TTSSTR 17`
- `#define SEP1STR 18`
- `#define SEP2STR 19`
- `#define TTSSEP 20`
- `#define HRSBKG 21`
- `#define MINBKG 22`
- `#define SECBKG 23`
- `#define TTSBKG 24`
- `#define NOEVENT 0x00`
- `#define WATCHBPRESS 0x01`
- `#define SWATCHBPRESS 0x02`
- `#define ALARMBPRESS 0x04`
- `#define TIMERBPRESS 0x08`
- `#define PLUSBPRESS 0x10`
- `#define MINUSBPRESS 0x20`
- `#define STARTBPRESS 0x40`
- `#define STOPBPRESS 0x80`
- `#define WATCHMODE 0`
- `#define SWATCHMODE 1`
- `#define ALARMMODE 2`
- `#define TIMERMODE 3`
- `#define txtinfo(w) ((Text *)((w)->ws))`
- `#define iconinfo(w) ((Icon *)((w)->ws))`
- `#define imginfo(w) ((Image *)((w)->ws))`

Enumerations

- enum **WidgetType** { **BACKGROUND**, **ICON**, **TEXT**, **IMAGE** }

Functions

- unsigned char **contains** (**Widget** *w, **TPoint** *point)
Checks if the touched point is inside a widget.
- unsigned char **OnTouch** (const **Widget** ws[], **TPoint** *press)
Handles the touch event.
- void **DrawInit** (**Widget** ws[])
Draws the initial GUI of the application.
- unsigned char **DrawOn** (**Widget** *w)
Draws the 'on' image of a widget.
- unsigned char **DrawOff** (**Widget** *w)
Draws the 'off' image of a widget.
- unsigned char **WPrint** (**Widget** *w, char *s)
Prints a string on the screen.

4.4.1 Detailed Description

4.4.2 Function Documentation

4.4.2.1 unsigned char contains (**Widget** * w, **TPoint** * point)

Checks if the touched point is inside a widget.

Parameters

<i>w</i>	Pointer to the widget.
<i>point</i>	Pointer to the coordinates data structure.

Return values

<i>1</i>	The point is inside the widget.
<i>0</i>	The point is outside the widget.

4.4.2.2 void DrawInit (**Widget** ws[])

Draws the initial GUI of the application.

Parameters

<i>ws</i>	Pointer to the application widgets array.
-----------	---

Return values

<i>None.</i>	
--------------	--

4.4.2.3 unsigned char DrawOff (Widget * w)

Draws the 'off' image of a widget.

Parameters

<i>w</i>	Pointer to the widget structure.
----------	----------------------------------

Return values

<i>1</i>	The image was successfully drawn on the screen.
<i>0</i>	Unable to draw the image.

4.4.2.4 unsigned char DrawOn (Widget * w)

Draws the 'on' image of a widget.

Parameters

<i>w</i>	Pointer to the widget structure.
----------	----------------------------------

Return values

<i>1</i>	The image was successfully drawn on the screen.
<i>0</i>	Unable to draw the image.

4.4.2.5 unsigned char OnTouch (const Widget *ws*[], TPoint * *press*)

Handles the touch event.

Parameters

<i>ws</i>	Pointer to the application widgets array.
<i>press</i>	Pointer to the coordinates data structure.

Return values

<i>1</i>	The touched point is inside one application widget
<i>0</i>	No widget in the application contains the touched point.

This function scans the entire widget array defined for the application and for each of them checks whether the coordinates of the touched point are inside the widget.

4.4.2.6 unsigned char WPrint (Widget * *w*, char * *s*)

Prints a string on the screen.

Parameters

<i>w</i>	Pointer to the widget data structure.
<i>s</i>	Pointer to the string which have to be printed.

4.5 Widget Definitions

Variables

- [Icon](#) `watch_b`
- [Icon](#) `swatch_b`
- [Icon](#) `alarm_b`
- [Icon](#) `timer_b`
- [Icon](#) `plus_b`
- [Icon](#) `minus_b`
- [Icon](#) `start_b`
- [Icon](#) `stop_b`
- [Icon](#) `set_b`
- [Icon](#) `reset_b`
- [Icon](#) `resume_b`
- [Icon](#) `alarm_exp_i`
- [Icon](#) `timer_exp_i`
- [Image](#) `hrs_back`
- [Image](#) `min_back`
- [Image](#) `sec_back`
- [Image](#) `tts_back`
- [Text](#) `txt`
- [Image](#) `backg`
- [Widget](#) `MyWatchScr` [NUMWIDGETS]

This array contains alle the widgets defined for the application.

4.5.1 Detailed Description

4.5.2 Variable Documentation

4.5.2.1 [Icon](#) `alarm_b`

Initial value:

```
= {
    b_alarm_on, b_alarm_off, ALARMBPRESS
}
```

4.5.2.2 [Icon](#) `alarm_exp_i`

Initial value:

```
= {
    alarm_exp_on, alarm_exp_off, NOEVENT
}
```


4.5.2.3 Image backg

Initial value:

```
= {
    bkg
}
```

4.5.2.4 Image hrs_back

Initial value:

```
= {
    hrs_bkg
}
```

4.5.2.5 Image min_back

Initial value:

```
= {
    min_bkg
}
```

4.5.2.6 Icon minus_b

Initial value:

```
= {
    b_minus, hide_minus, MINUSBPRESS
}
```

4.5.2.7 Widget MyWatchScr[NUMWIDGETS]

Initial value:

```
= {
    {0, 0, 320, 240, BACKGROUND, (void *)&backg},
    {0, 0, 80, 45, ICON, (void *)&watch_b},
    {80, 0, 80, 45, ICON, (void *)&swatch_b},
    {160, 0, 80, 45, ICON, (void *)&alarm_b},
    {240, 0, 80, 45, ICON, (void *)&timer_b},
    {142, 125, 36, 35, ICON, (void *)&plus_b},
    {142, 196, 36, 35, ICON, (void *)&minus_b},
    {30, 160, 100, 40, ICON, (void *)&start_b},
    {30, 160, 100, 40, ICON, (void *)&set_b},
    {30, 160, 100, 40, ICON, (void *)&resume_b},
    {190, 160, 100, 40, ICON, (void *)&stop_b},
    {190, 160, 100, 40, ICON, (void *)&reset_b},
    {61, 114, 38, 38, ICON, (void *)&alarm_exp_i},
    {221, 114, 38, 38, ICON, (void *)&timer_exp_i},
    {29, 70, 40, 40, TEXT, (void *)&ttxt},
    {99, 70, 40, 40, TEXT, (void *)&ttxt},
    {168, 70, 40, 40, TEXT, (void *)&ttxt},
    {243, 70, 40, 40, TEXT, (void *)&ttxt},
    {80, 66, 40, 40, TEXT, (void *)&ttxt},
    {149, 66, 40, 40, TEXT, (void *)&ttxt},
    {225, 68, 20, 40, TEXT, (void *)&ttxt},
    {29, 70, 62, 42, IMAGE, (void *)&hrs_back},
    {99, 70, 62, 42, IMAGE, (void *)&min_back},
    {168, 70, 62, 42, IMAGE, (void *)&sec_back},
    {230, 70, 62, 42, IMAGE, (void *)&tts_back}
}
```

This array contains alle the widgets defined for the application.

4.5.2.8 Icon plus_b

Initial value:

```
= {  
    b_plus, hide_plus, PLUSBPRESS  
}
```

4.5.2.9 Icon reset_b

Initial value:

```
= {  
    b_reset, hide_stop, STOPBPRESS  
}
```

4.5.2.10 Icon resume_b

Initial value:

```
= {  
    b_resume, hide_start, STARTBPRESS  
}
```

4.5.2.11 Image sec_back

Initial value:

```
= {  
    sec_bkg  
}
```

4.5.2.12 Icon set_b

Initial value:

```
= {  
    b_set, hide_start, STARTBPRESS  
}
```

4.5.2.13 Icon start_b

Initial value:

```
= {  
    b_start, hide_start, STARTBPRESS  
}
```

4.5.2.14 Icon stop_b

Initial value:

```
= {  
    b_stop, hide_stop, STOPBPRESS  
}
```

4.5.2.15 Icon swatch_b

Initial value:

```
= {  
    b_swatch_on, b_swatch_off, SWATCHBPRESS  
}
```

4.5.2.16 Icon timer_b

Initial value:

```
= {  
    b_timer_on, b_timer_off, TIMERBPRESS  
}
```

4.5.2.17 Icon timer_exp_i

Initial value:

```
= {  
    timer_exp_on, timer_exp_off, NOEVENT  
}
```

4.5.2.18 Image tts_back

Initial value:

```
= {  
    tts_bkg  
}
```

4.5.2.19 Text txt

Initial value:

```
= {  
    &Font32x48, White  
}
```

4.5.2.20 Icon watch_b

Initial value:

```
= {  
    b_watch_on, b_watch_off, WATCHBPRESS  
}
```

4.6 Events

Event mask declaration.

Macros

- `#define SetEvt(Event) (evts |= Event)`
Sets an event in the event mask.
- `#define ClearEvt(Event) (evts &= !Event)`
Resets an event in the event mask.
- `#define ClearEvents() (evts = 0)`
Resets the event mask.
- `#define IsEvent(Event) ((unsigned char)(evts & Event))`
Checks if an event has been set.

Typedefs

- `typedef unsigned char Event`
- `typedef unsigned char Events`

4.6.1 Detailed Description

Event mask declaration.

4.6.2 Macro Definition Documentation

4.6.2.1 `#define ClearEvt(Event) (evts &= !Event)`

Resets an event in the event mask.

Parameters

<i>Event</i>	The event to be reset.
--------------	------------------------

4.6.2.2 `#define IsEvent(Event) ((unsigned char)(evts & Event))`

Checks if an event has been set.

Parameters

<i>Event</i>	The event to be checked in the event mask.
--------------	--

4.6.2.3 `#define SetEvt(Event) (evts |= Event)`

Sets an event in the event mask.

Parameters

<i>Event</i>	The event to be set.
--------------	----------------------

4.7 FSM Definition

Enumerations

- enum [Signal](#) {
 watch_b, **swatch_b**, **alarm_b**, **timer_b**,
 plus_b, **minus_b**, **start_b**, **stop_b**,
 ENTRY, **EXIT**, **INIT**, **TICK**,
 ABSENT }
 FSM signals.
- enum [State](#) {
 watch_showtime, **watch_sethours**, **watch_setminutes**, **swatch_stop**,
 swatch_running, **swatch_pause**, **alarm_sethours**, **alarm_setminutes**,
 alarm_running, **timer_sethours**, **timer_setminutes**, **timer_setseconds**,
 timer_running }
 FSM states.

Variables

- [State](#) **state_**
- [State](#) **swatchHistory_**
- [State](#) **alarmHistory_**
- [State](#) **timerHistory_**

4.7.1 Detailed Description

4.8 Types

Data Structures

- struct [time_](#)

Data structure containing timing information.

Typedefs

- typedef char **char_t**
- typedef signed char **int8_t**
- typedef signed short **int16_t**
- typedef unsigned char **uint8_t**
- typedef unsigned short **uint16_t**
- typedef float **float32_t**
- typedef double **float64_t**
- typedef long double **float128_t**
- typedef struct [time_time](#)

Data structure containing timing information.

4.8.1 Detailed Description

Chapter 5

Data Structure Documentation

5.1 Icon Struct Reference

Data Fields

- unsigned char * **iconp**
- unsigned char * **iconr**
- Event **onpress**

The documentation for this struct was generated from the following file:

- [Widget.h](#)

5.2 Image Struct Reference

Data Fields

- unsigned char * **image**

The documentation for this struct was generated from the following file:

- [Widget.h](#)

5.3 Text Struct Reference

Data Fields

- sFONT * **font**
- unsigned short int **color**

The documentation for this struct was generated from the following file:

- [Widget.h](#)

5.4 time_ Struct Reference

Data structure containing timing information.

```
#include <types.h>
```

Data Fields

- uint8_t **hours**
- uint8_t **minutes**
- uint8_t **seconds**
- uint8_t **tenths**

5.4.1 Detailed Description

Data structure containing timing information.

The documentation for this struct was generated from the following file:

- [types.h](#)

5.5 Widget Struct Reference

Data Fields

- unsigned short int **xl**
- unsigned short int **yt**
- unsigned short int **xw**
- unsigned short int **yh**
- WidgetType **wt**
- void * **ws**

The documentation for this struct was generated from the following file:

- [Widget.h](#)

Chapter 6

File Documentation

6.1 code.c File Reference

Contains the body of all tasks and the global variables defined.

```
#include "ee.h"
#include "ee_irq.h"
#include <stdio.h>
#include "stm32f4xx_conf.h"
#include "stm32f4_discovery.h"
#include "stm32f4_discovery_lcd.h"
#include "stm32f4xx.h"
#include "STMPE811QTR.h"
#include "mypictures.h"
#include "Widget.h"
#include "Touch.h"
#include "Event.h"
#include "lcd_add.h"
#include "fonts.h"
#include "types.h"
#include "SWatchFSM.h"
```

Functions

- static void [strencode1digit](#) (char *str, int digit)
Converts a one digit integer into a string.
- static void [strencode2digit](#) (char *str, int digit)
Converts a two digits integer into a string.
- void [activateSwatch](#) ()
Activates the Stopwatch task.
- void [activateAlarm](#) ()
Activates the Alarm task.
- void [activateTimer](#) ()
Activates the Timer task.
- void [disableAlarm](#) ()
Terminates the Alarm task.

- void `disableTimer` ()
Terminates the Timer task.
- void `disableSwatch` ()
Terminates the Stopwatch task.
- static void `updateTime` (uint8_t *oh, uint8_t *om, uint8_t *os, uint8_t *ot, uint8_t oldmode)
Updates the time on the screen.
- void `updateScreen` (uint8_t om, uint8_t m)
Updates the screen widgets.
- `ISR2` (systick_handler)
System Tick interrupt handler.
- `TASK` (TaskLCD)
LDC task body.
- `TASK` (TaskWatch)
Implements the watch mode.
- `TASK` (TaskSwatch)
Implements the Stopwatch mode.
- `TASK` (TaskAlarm)
Implements the Alarm mode.
- `TASK` (TaskTimer)
Implements the Timer mode.
- `TASK` (TaskFSM)
Implements the State Machine of the application.
- int `main` (void)
Main task of the application.

Variables

- uint8_t `mode` = 0
Application mode.
- uint8_t `alarm_status` = 0
Alarm status. 0: Alarm not set yet. 1: Alarm set. 2: Alarm expired.
- uint8_t `timer_exp` = 0
- uint8_t `swatchrun` = 0
- uint8_t `watchset` = 0
- uint8_t `alarm_cycle` = 200
- time `display_time`
- time `watch_time`
- time `swatch_time`
- time `alarm_time`
- time `timer_time`
- static SWatchFSM `watch`

6.1.1 Detailed Description

Contains the body of all tasks and the global variables defined.

Author

Paolo Sassi

Date

21 January 2016

Attention

ERIKA Enterprise - a tiny RTOS for small microcontrollers

Copyright (C) 2002-2013 Evidence Srl

This file is part of ERIKA Enterprise.

ERIKA Enterprise is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation, (with a special exception described below).

Linking this code statically or dynamically with other modules is making a combined work based on this code. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this code with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this code, you may extend this exception to your version of the code, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

ERIKA Enterprise is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License version 2 for more details.

You should have received a copy of the GNU General Public License version 2 along with ERIKA Enterprise; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

6.1.2 Variable Documentation

6.1.2.1 `uint8_t timer_exp = 0`

1 if the timer is expired, 0 otherwise.

6.2 Cplus.h File Reference

Macros for using class-like semantics in C.

Macros

- `#define CLASS(name_)`
Macros for declaring classes.
- `#define METHODS`};
- `#define END_CLASS`
- `#define SUBCLASS(class_, superclass_)`
Macros for declaring subclasses.

6.2.1 Detailed Description

Macros for using class-like semantics in C.

Author

Paolo Sassi

Date

22 January 2016

6.2.2 Macro Definition Documentation

6.2.2.1 `#define CLASS(name_)`

Value:

```
typedef struct name_ name_;\n    struct name_ {
```

Macros for declaring classes.

6.2.2.2 `#define SUBCLASS(class_, superclass_)`

Value:

```
CLASS(class_) \n    superclass_ super_;
```

Macros for declaring subclasses.

6.3 Event.c File Reference

Contains the event mask definition.

```
#include "Event.h"
```

Variables

- Events **evts**

6.3.1 Detailed Description

Contains the event mask definition.

Author

Paolo Sassi

Date

22 January 2016

6.4 Event.h File Reference

Contains the macros used to handle the event masks.

Macros

- `#define SetEvt(Event) (evts |= Event)`
Sets an event in the event mask.
- `#define ClearEvt(Event) (evts &= !Event)`
Resets an event in the event mask.
- `#define ClearEvents() (evts = 0)`
Resets the event mask.
- `#define IsEvent(Event) ((unsigned char)(evts & Event))`
Checks if an event has been set.

Typedefs

- `typedef unsigned char Event`
- `typedef unsigned char Events`

Variables

- Events **evts**

6.4.1 Detailed Description

Contains the macros used to handle the event masks.

Author

Paolo Sassi

Date

22 January 2016

6.5 mypictures.c File Reference

This file contains the application pictures in RGB565 format.

Variables

- const unsigned char **bkg** [153654]
- const unsigned char **b_watch_on** [7254]
- const unsigned char **b_watch_off** [7254]
- const unsigned char **b_swatch_on** [7254]
- const unsigned char **b_swatch_off** [7254]
- const unsigned char **b_alarm_on** [7254]
- const unsigned char **b_alarm_off** [7254]
- const unsigned char **b_timer_on** [7254]
- const unsigned char **b_timer_off** [7254]
- const unsigned char **b_plus** [2646]
- const unsigned char **b_minus** [2574]
- const unsigned char **b_start** [8054]
- const unsigned char **b_stop** [8054]
- const unsigned char **b_set** [8054]
- const unsigned char **b_reset** [8054]
- const unsigned char **b_resume** [8054]
- const unsigned char **hide_start** [8054]
- const unsigned char **hide_stop** [8054]
- const unsigned char **hide_plus** [2574]
- const unsigned char **hide_minus** [2574]
- const unsigned char **hrs_bkg** [5262]
- const unsigned char **min_bkg** [5262]
- const unsigned char **sec_bkg** [5262]
- const unsigned char **tts_bkg** [5262]
- const unsigned char **alarm_exp_on** [2942]
- const unsigned char **alarm_exp_off** [2942]
- const unsigned char **timer_exp_on** [2942]
- const unsigned char **timer_exp_off** [2942]

6.5.1 Detailed Description

This file contains the application pictures in RGB565 format.

Author

Paolo Sassi

Date

22 January 2016

6.6 mypictures.h File Reference

Pictures header file.

Variables

- const unsigned char **bkg** [153654]
- const unsigned char **b_watch_on** [7254]
- const unsigned char **b_watch_off** [7254]
- const unsigned char **b_swatch_on** [7254]
- const unsigned char **b_swatch_off** [7254]
- const unsigned char **b_alarm_on** [7254]
- const unsigned char **b_alarm_off** [7254]
- const unsigned char **b_timer_on** [7254]
- const unsigned char **b_timer_off** [7254]
- const unsigned char **b_plus** [2504]
- const unsigned char **b_minus** [2504]
- const unsigned char **b_start** [8054]
- const unsigned char **b_stop** [8054]
- const unsigned char **b_set** [8054]
- const unsigned char **b_reset** [8054]
- const unsigned char **b_resume** [8054]
- const unsigned char **hide_start** [8054]
- const unsigned char **hide_stop** [8054]
- const unsigned char **hide_plus** [2574]
- const unsigned char **hide_minus** [2574]
- const unsigned char **hrs_bkg** [5262]
- const unsigned char **min_bkg** [5262]
- const unsigned char **sec_bkg** [5262]
- const unsigned char **tts_bkg** [5262]
- const unsigned char **alarm_exp_on** [2866]
- const unsigned char **alarm_exp_off** [2866]
- const unsigned char **timer_exp_on** [2942]
- const unsigned char **timer_exp_off** [2942]

6.6.1 Detailed Description

Pictures header file.

Author

Paolo Sassi

Date

22 January 2016

6.7 SWatchFSM.c File Reference

Contains the nested switch implementation of the FSM.

```
#include "Cplus.h"
#include "SWatchFSM.h"
#include "types.h"
```

Functions

- void `activateAlarm` ()
Activates the Alarm task.
- void `activateSwatch` ()
Activates the Stopwatch task.
- void `activateTimer` ()
Activates the Timer task.
- void `disableTimer` ()
Terminates the Timer task.
- void `disableAlarm` ()
Terminates the Alarm task.
- void `disableSwatch` ()
Terminates the Stopwatch task.
- void `SWatchFSMinit` (SWatchFSM *me)
FSM initialization function.
- static void `tran_` (SWatchFSM *me, `State` dest)
FSM transition private function.
- void `SWatchFSMdispatch` (SWatchFSM *me, `Signal` sig)
Dispatch function of the FSM, implemented using the nested switch.

Variables

- uint8_t `watchset`
- uint8_t `mode`
Application mode.
- uint8_t `swatchrun`
- uint8_t `alarm_status`
Alarm status. 0: Alarm not set yet. 1: Alarm set. 2: Alarm expired.
- uint8_t `timer_exp`
- uint8_t `alarm_cycle`
- `time` `display_time`
- `time` `watch_time`
- `time` `swatch_time`
- `time` `alarm_time`
- `time` `timer_time`

6.7.1 Detailed Description

Contains the nested switch implementation of the FSM.

Author

Paolo Sassi

Date

22 January 2016

6.7.2 Function Documentation

6.7.2.1 void SWatchFSMdispatch (SWatchFSM * me, Signal sig)

Dispatch function of the FSM, implemented using the nested switch.

Parameters

<i>me</i>	Pointer to the FSM data structure.
<i>sig</i>	Signal to be dispatched.

Return values

<i>None.</i>	
--------------	--

6.7.2.2 void SWatchFSMinit (SWatchFSM * *me*)

FSM initialization function.

Parameters

<i>me</i>	Pointer to the FSM data structure.
-----------	------------------------------------

Return values

<i>None.</i>	
--------------	--

6.7.2.3 static void tran_ (SWatchFSM * *me*, State *dest*) [static]

FSM transition private function.

Parameters

<i>me</i>	Pointer to the FSM data structure.
<i>dest</i>	Destination state of the transition.

Return values

<i>None.</i>	
--------------	--

6.7.3 Variable Documentation**6.7.3.1 uint8_t timer_exp**

1 if the timer is expired, 0 otherwise.

6.8 SWatchFSM.h File Reference

Contains the definition of the FSM and the definitions of its signals and states.

```
#include "Cplus.h"
#include "stm32f4xx.h"
```

Enumerations

- enum [Signal](#) {
 watch_b, swatch_b, alarm_b, timer_b,
 plus_b, minus_b, start_b, stop_b,
 ENTRY, EXIT, INIT, TICK,
 ABSENT }
 FSM signals.
- enum [State](#) {
 watch_showtime, watch_sethours, watch_setminutes, swatch_stop,
 swatch_running, swatch_pause, alarm_sethours, alarm_setminutes,
 alarm_running, timer_sethours, timer_setminutes, timer_setseconds,
 timer_running }
 FSM states.

Functions

- void [SWatchFSMinit](#) (SWatchFSM *me)
 FSM initialization function.
- void [SWatchFSMdispatch](#) (SWatchFSM *me, [Signal](#) sig)
 Dispatch function of the FSM, implemented using the nested switch.

Variables

- [State](#) **state_**
- [State](#) **swatchHistory_**
- [State](#) **alarmHistory_**
- [State](#) **timerHistory_**

6.8.1 Detailed Description

Contains the definition of the FSM and the definitions of its signals and states.

Author

Paolo Sassi

Date

22 January 2016

6.8.2 Function Documentation

6.8.2.1 void SWatchFSMdispatch (SWatchFSM * me, [Signal](#) sig)

Dispatch function of the FSM, implemented using the nested switch.

Parameters

<i>me</i>	Pointer to the FSM data structure.
<i>sig</i>	Signal to be dispatched.

Return values

<i>None.</i>	
--------------	--

6.8.2.2 void SWatchFSMinit (SWatchFSM * *me*)

FSM initialization function.

Parameters

<i>me</i>	Pointer to the FSM data structure.
-----------	------------------------------------

Return values

<i>None.</i>	
--------------	--

6.9 types.h File Reference

Type definitions.

Data Structures

- struct [time_](#)
Data structure containing timing information.

Typedefs

- typedef char **char_t**
- typedef signed char **int8_t**
- typedef signed short **int16_t**
- typedef unsigned char **uint8_t**
- typedef unsigned short **uint16_t**
- typedef float **float32_t**
- typedef double **float64_t**
- typedef long double **float128_t**
- typedef struct [time_](#) **time**
Data structure containing timing information.

6.9.1 Detailed Description

Type definitions.

Author

Paolo Sassi

Date

22 January 2016

6.10 Widget.c File Reference

Contains the functions to manage the widgets on the screen.

```
#include "Widget.h"
#include "Event.h"
#include "mypictures.h"
#include <stdio.h>
#include "stm32f4_discovery_lcd.h"
```

Functions

- unsigned char [contains](#) ([Widget](#) *w, [TPoint](#) *point)
Checks if the touched point is inside a widget.
- unsigned char [OnTouch](#) (const [Widget](#) ws[], [TPoint](#) *press)
Handles the touch event.
- void [DrawInit](#) ([Widget](#) ws[])
Draws the initial GUI of the application.
- unsigned char [DrawOn](#) ([Widget](#) *w)
Draws the 'on' image of a widget.
- unsigned char [DrawOff](#) ([Widget](#) *w)
Draws the 'off' image of a widget.
- unsigned char [WPrint](#) ([Widget](#) *w, char *s)
Prints a string on the screen.

Variables

- [Icon](#) [watch_b](#)
- [Icon](#) [swatch_b](#)
- [Icon](#) [alarm_b](#)
- [Icon](#) [timer_b](#)
- [Icon](#) [plus_b](#)
- [Icon](#) [minus_b](#)
- [Icon](#) [start_b](#)
- [Icon](#) [stop_b](#)
- [Icon](#) [set_b](#)

- [Icon](#) `reset_b`
- [Icon](#) `resume_b`
- [Icon](#) `alarm_exp_i`
- [Icon](#) `timer_exp_i`
- [Image](#) `hrs_back`
- [Image](#) `min_back`
- [Image](#) `sec_back`
- [Image](#) `tts_back`
- [Text](#) `txt`
- [Image](#) `backg`
- [Widget](#) `MyWatchScr` [NUMWIDGETS]

This array contains alle the widgets defined for the application.

6.10.1 Detailed Description

Contains the functions to manage the widgets on the screen.

Author

Paolo Sassi

Date

22 January 2016

6.11 Widget.h File Reference

Contains the type definitions and the macros used for the screen widgets.

```
#include "Event.h"
#include "Touch.h"
#include "fonts.h"
```

Data Structures

- struct [Image](#)
- struct [Icon](#)
- struct [Text](#)
- struct [Widget](#)

Macros

- `#define NUMWIDGETS 25`
- `#define BAKCG 0`
- `#define BWATCH 1`
- `#define BSWATCH 2`
- `#define BALARM 3`
- `#define BTIMER 4`
- `#define BPLUS 5`
- `#define BMINUS 6`
- `#define BSTART 7`
- `#define BSET 8`
- `#define BRESUME 9`
- `#define BSTOP 10`
- `#define BRESET 11`
- `#define ALARMEXP 12`
- `#define TIMEREXP 13`
- `#define HRSSTR 14`
- `#define MINSTR 15`
- `#define SECSTR 16`
- `#define TTSSTR 17`
- `#define SEP1STR 18`
- `#define SEP2STR 19`
- `#define TTSSEP 20`
- `#define HRSBKG 21`
- `#define MINBKG 22`
- `#define SECBKG 23`
- `#define TTSBKG 24`
- `#define NOEVENT 0x00`
- `#define WATCHBPRESS 0x01`
- `#define SWATCHBPRESS 0x02`
- `#define ALARMBPRESS 0x04`
- `#define TIMERBPRESS 0x08`
- `#define PLUSBPRESS 0x10`
- `#define MINUSBPRESS 0x20`
- `#define STARTBPRESS 0x40`
- `#define STOPBPRESS 0x80`
- `#define WATCHMODE 0`
- `#define SWATCHMODE 1`
- `#define ALARMMODE 2`
- `#define TIMERMODE 3`
- `#define txtinfo(w) ((Text *)((w)->ws))`
- `#define iconinfo(w) ((Icon *)((w)->ws))`
- `#define imginfo(w) ((Image *)((w)->ws))`

Enumerations

- `enum WidgetType { BACKGROUND, ICON, TEXT, IMAGE }`

Functions

- void [DrawInit](#) ([Widget](#) ws[])
Draws the initial GUI of the application.
- unsigned char [OnTouch](#) (const [Widget](#) ws[], TPoint *press)
Handles the touch event.
- unsigned char [DrawOn](#) ([Widget](#) *w)
Draws the 'on' image of a widget.
- unsigned char [DrawOff](#) ([Widget](#) *w)
Draws the 'off' image of a widget.
- unsigned char [WPrint](#) ([Widget](#) *w, char *s)
Prints a string on the screen.

Variables

- [Widget](#) [MyWatchScr](#) []
This array contains alle the widgets defined for the application.

6.11.1 Detailed Description

Contains the type definitions and the macros used for the screen widgets.

Author

Paolo Sassi

Date

22 January 2016

Index

- activateAlarm
 - Utility, [7](#)
- activateSwatch
 - Utility, [8](#)
- activateTimer
 - Utility, [8](#)
- alarm_b
 - Widget Definitions, [18](#)
- alarm_exp_i
 - Widget Definitions, [18](#)
- backg
 - Widget Definitions, [18](#)
- CLASS
 - Cplus.h, [32](#)
- ClearEvt
 - Events, [22](#)
- code.c, [29](#)
 - timer_exp, [31](#)
- contains
 - Widget, [15](#)
- Cplus.h, [31](#)
 - CLASS, [32](#)
 - SUBCLASS, [32](#)
- disableAlarm
 - Utility, [8](#)
- disableSwatch
 - Utility, [8](#)
- disableTimer
 - Utility, [9](#)
- DrawInit
 - Widget, [15](#)
- DrawOff
 - Widget, [16](#)
- DrawOn
 - Widget, [16](#)
- Event.c, [32](#)
- Event.h, [33](#)
- Events, [22](#)
 - ClearEvt, [22](#)
 - IsEvent, [22](#)
 - SetEvt, [22](#)
- FSM Definition, [24](#)
- hrs_back
 - Widget Definitions, [19](#)
- Icon, [27](#)
- Image, [27](#)
- Interrupt Handler, [11](#)
- IsEvent
 - Events, [22](#)
- main
 - Tasks, [12](#)
- min_back
 - Widget Definitions, [19](#)
- minus_b
 - Widget Definitions, [19](#)
- MyWatchScr
 - Widget Definitions, [19](#)
- mypictures.c, [34](#)
- mypictures.h, [34](#)
- OnTouch
 - Widget, [16](#)
- plus_b
 - Widget Definitions, [19](#)
- reset_b
 - Widget Definitions, [20](#)
- resume_b
 - Widget Definitions, [20](#)
- SUBCLASS
 - Cplus.h, [32](#)
- SWatchFSM.c, [35](#)
 - SWatchFSMdispatch, [36](#)
 - SWatchFSMinit, [37](#)
 - timer_exp, [37](#)
 - tran_, [37](#)
- SWatchFSM.h, [37](#)
 - SWatchFSMdispatch, [38](#)
 - SWatchFSMinit, [39](#)
- SWatchFSMdispatch
 - SWatchFSM.c, [36](#)
 - SWatchFSM.h, [38](#)
- SWatchFSMinit
 - SWatchFSM.c, [37](#)
 - SWatchFSM.h, [39](#)
- sec_back
 - Widget Definitions, [20](#)
- set_b
 - Widget Definitions, [20](#)
- SetEvt
 - Events, [22](#)
- start_b

- Widget Definitions, 20
- stop_b
 - Widget Definitions, 20
- strencode1digit
 - Utility, 9
- strencode2digit
 - Utility, 9
- swatch_b
 - Widget Definitions, 21
- TASK
 - Tasks, 12, 13
- Tasks, 12
 - main, 12
 - TASK, 12, 13
- Text, 27
- time_, 28
- timer_b
 - Widget Definitions, 21
- timer_exp
 - code.c, 31
 - SWatchFSM.c, 37
- timer_exp_i
 - Widget Definitions, 21
- tran_
 - SWatchFSM.c, 37
- tts_back
 - Widget Definitions, 21
- txt
 - Widget Definitions, 21
- Types, 25
- types.h, 39
- updateScreen
 - Utility, 10
- updateTime
 - Utility, 10
- Utility, 7
 - activateAlarm, 7
 - activateSwatch, 8
 - activateTimer, 8
 - disableAlarm, 8
 - disableSwatch, 8
 - disableTimer, 9
 - strencode1digit, 9
 - strencode2digit, 9
 - updateScreen, 10
 - updateTime, 10
- WPrint
 - Widget, 17
- watch_b
 - Widget Definitions, 21
- Widget, 14, 28
 - contains, 15
 - DrawInit, 15
 - DrawOff, 16
 - DrawOn, 16
 - OnTouch, 16
- WPrint, 17
- Widget Definitions, 18
 - alarm_b, 18
 - alarm_exp_i, 18
 - backg, 18
 - hrs_back, 19
 - min_back, 19
 - minus_b, 19
 - MyWatchScr, 19
 - plus_b, 19
 - reset_b, 20
 - resume_b, 20
 - sec_back, 20
 - set_b, 20
 - start_b, 20
 - stop_b, 20
 - swatch_b, 21
 - timer_b, 21
 - timer_exp_i, 21
 - tts_back, 21
 - txt, 21
 - watch_b, 21
- Widget.c, 40
- Widget.h, 41