Stopwatch Hand-coded

Generated by Doxygen 1.8.11

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

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

None	

Return values

None	

4.1.2.2 void activateSwatch ( )

Activates the Stopwatch task.

Parameters

None	

Return values

None	

4.1.2.3 void activateTimer ( )

Activates the Timer task.

Parameters

None	

Return values

None	

4.1.2.4 void disableAlarm ( )

Terminates the Alarm task.

Parameters

None	

Return values

None	

4.1.2.5 void disableSwatch ( )

Terminates the Stopwatch task.

Parameters

None	

Return values

None	

4.1.2.6 void disableTimer ( )

Terminates the Timer task.

Parameters

None	

Return values

None	

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

Converts a one digit integer into a string.

Parameters

str	pointer to the returning string.
digit	integer digit to be converted.

Return values

None	

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

Converts a two digits integer into a string.

Parameters

str	pointer to the returning string.
digit	integer digits to be converted.

Return values

None	

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

Updates the screen widgets.

Parameters

om	Old application mode.
m	New application mode.

Return values

None	

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

oh	Old hours.
om	Old minutes.
os	Old seconds.
ot	Old tenths.
oldmode	Old application mode.

Return values

None	

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

None	

Return values

None	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

w	Pointer to the widget.
point	Pointer to the coordinates data structure.

Return values

1	The point is inside the widget.
0	The point is outside the widget.

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

Draws the initial GUI of the application.

Parameters

ws	Pointer to the application widgets array.

Return values

None.	

4.4.2.3 unsigned char DrawOff (Widget ∗ *w*)

Draws the 'off' image of a widget.

Parameters

w	Pointer to the widget structure.

Return values

1	The image was successfully drawn on the screen.
0	Unable to draw the image.

4.4.2.4 unsigned char DrawOn (Widget ∗ *w*)

Draws the 'on' image of a widget.

Parameters

w	Pointer to the widget structure.

Return values

1	The image was successfully drawn on the screen.
0	Unable to draw the image.

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

Handles the touch event.

Parameters

ws	Pointer to the application widgets array.
press	Pointer to the coordinates data structure.

Return values

1	The touched point is inside one application widget
0	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

w	Pointer to the widget data structure.
s	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 *)&txt},
        {99, 70, 40, 40, TEXT, (void *)&txt},
        {168, 70, 40, 40, TEXT, (void *)&txt},
        {243, 70, 40, 40, TEXT, (void *)&txt},
        {80, 66, 40, 40, TEXT, (void *)&txt},
        {149, 66, 40, 40, TEXT, (void *)&txt},
        {225, 68, 20, 40, TEXT, (void *)&txt},
        {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

Event	The event to be reset.

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

Checks if an event has been set.

Parameters

Event	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

Event	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

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_;\
   struct name_ {
```

Macros for declaring classes.

6.2.2.2 #define SUBCLASS(*class_*, *superclass_*)

Value:

```
CLASS(class_)                             \
      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

me	Pointer to the FSM data structure.
sig	Signal to be dispatched.

Return values

None.	

6.7.2.2 void SWatchFSMinit (SWatchFSM ∗ *me*)

FSM initialization function.

Parameters

me	Pointer to the FSM data structure.

Return values

None.	

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

FSM transition private function.

Parameters

me	Pointer to the FSM data structure.
dest	Destination state of the transition.

Return values

None.	

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

me	Pointer to the FSM data structure.
sig	Signal to be dispatched.

Return values

None.	

6.8.2.2 void SWatchFSMinit (SWatchFSM * *me*)

FSM initialization function.

Parameters

me	Pointer to the FSM data structure.

Return values

None.	

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