UM0885
User manual

Multilayer C library used to drive STP1612PW05 on the systems with SPI

1 Introduction

This user manual describes usage of the library dedicated to perform LED dimming functionality. It describes all the library functions and their binding to the standard peripheral library of a chosen microcontroller.

This C library can be directly used with STM32™ microcontroller without any change.

This C library contains a demonstration extension that is tailored for the STMicroelectronics™ evaluation kit STEVAL-ILL028V1.

Contents

List of tables

2 Document and library rules

2.1 Predefined values and structures

2.1.1 Value: ERROR and SUCCESS

```
typedef enum
{
ERROR = 0,
SUCCESS = !ERROR
} ErrorStatus;
```

These values are used to report the result of the functions their feedback is necessary for the following operations.

2.1.2 Struct: buttonsAndADC

```
struct buttonsAndADC
{
  unsigned char buttonLeft;
  unsigned char buttonRight;
  unsigned char buttonCenter;
  unsigned char (*buttonKnob)(void);
};
```

This global structure (defined in board_control.c, declared in board_control.h) provides the status of the buttons and the knob placed on the STEVAL-ILL028V1. Variables linked to the buttons are set to “1”, if the button was pressed. And it remains “1” until the user set it to “0”. Calling the ButtonKnob() returns the actual value of the knob (variable resistor connected to the ADC).

In order to be able to use this structure, the following command must be executed. (BC prefix stands for “Board Control”). The functions are declared in board_control.h:

```
BC_GPIO_Configuration();
BC_EXTI_Configuration();
BC_NVIC_Configuration();
BC_ADC_Configuration();
```

2.1.3 Struct: environment

```
struct environment
{
  unsigned char columns;          //size of display
  unsigned char rows;             //size of display
  unsigned char colors;           //1 - one color, 3 RGB leds.
  unsigned char randomNumber;     //some games need random number
  unsigned char gameFinished;     //when game ends
  unsigned char setupGame;        //set flag when started first.
                                  //The flag is cleared automaticaly
  unsigned char refreshRequest;   //game changed data to display
  unsigned char exitRequest;      //when we want the main menu again
};
```

This structure (declared in led_demonstration.h) is used to pass a big amount of parameters to the demonstration functions. The members of this structure define the logical arrangement of the LED display and help to drive demonstration algorithms.

3 Description of firmware library

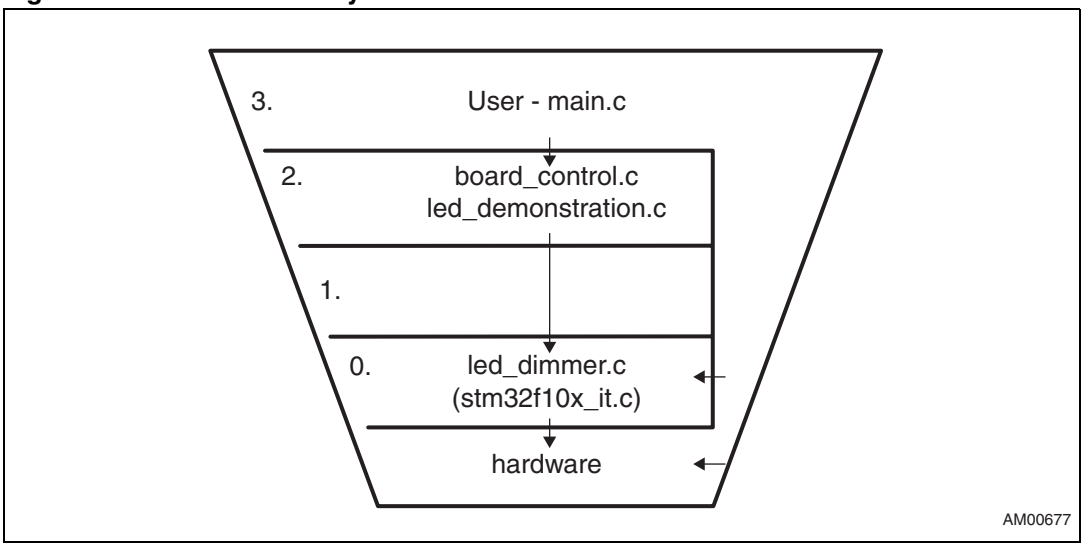
3.1 Summary of firmware library files

Table 1. Firmware library files

L[1]	Filename	Description
0	led_dimmer.c (.h)	Basic dimmer functionality, configuration and commands, HW related.
0	stm32f10x_it.c	Interrupt routines dedicated to service dimming functionality. HW related.
1	—	—
2	board_control.c (h)	Extension that enables simple usage of buttons and ADC on evalkit. HW.
2	led_demonstration.c (.h)	Extension that shows functionality, demonstrations and modes of the evalkit.
3	main.c	Example application using led_demonstration.c, board_control.c.

1. The column “L” groups the library files into the groups on the same level between the user and HW.

Figure 1. Firmware library files



All the source files use libraries dedicated to the peripherals of the microcontroller. They are listed in *Table 2*.

Table 2. Used external libraries and functions

L[1]	Filename	Description
0	stm32f10x_gpio.h	Library for configuration of the general purpose i/o of the STM32.
0	(stm32f10x_dma.h)	(Optional - library for configuration of DMA of STM32).
0	(stm32f10x_nvic.h)	(Optional - library for configuration of the interrupt controller of STM32).
0	stm32f10x_lib.h	Library with the fundamental definitions, addresses of the STM32
1	stdlib.h	Provides the function “malloc” for dynamic memory allocation.
2	stm32f10x_exti.h	Library for configuration of external interrupts that are connected to the buttons.

1. The column “L” groups the library files into the groups on the same level between the user and HW.

3.2 LEVEL 0 - led_dimmer.c (.h)

Constants and functions the file offers

```
REDchannel
GREENchannel
BLUEchannel
```

SDO SPI commands

```
command_DATA_LATCH
command_GLOBAL_LATCH
command_READ_CONFIGURATION
command_ENABLE_ERROR_DETECTION
command_READ_ERROR_STATUS_CODE
command_WRITE_CONFIGURATION
command_RESET_TO_16_BIT_REG_LEN
command_NO_COMMAND
```

setup_driver

```
RegLen256bit
RegLen16bit
ThermalOverTemp
ThermalSave
PWMresolution16
PWMresolution12
PWMmode64MSB
PWMmode16MSB
PWMmode4MSB
PWMmode12bit
ManualSynchro
AutoSynchro
ThermalShutEnable
ThermalShutDisable
PWCLKalertDisable
PWCLKalertEnable
GAIN_2_0
GAIN_1_6
GAIN_1_2
```

GAIN_1_0
GAIN_0_5
GAIN_0_2
GAIN_0_1
Do16to12BitConvertion
DoNot16to12BitConvertion

Preprocessor directive
default_GAIN = GAIN_2_0

Global variable
actual_configuration

Functions
RCC_Configuration
NVIC_Configuration
SPI_DMA_Initsetup
setup_driver
resetup_driver
error_detection
use_new_LEDmap
fillDriverPattern

External functions and types that are called from this file
SPI_I2S_SendData(), SPI_I2C_ReceiveData(), SPI_I2S_GetFlagStatus() and group of functions declared in libraries dedicated to GPIO, RCC, (DMA, NVIC optionally).

3.2.1 Detailed functions description: RCC_Configuration, NVIC_Configuration, SPI_DMA_Init

Table 3. RCC_Configuration, NVIC_Configuration, SPI_DMA_Init functions

Filename	Description
Function name	RCC_Configuration, NVIC_Configuration, SPI_DMA_Init functions.
Function prototype	void RCC_Configuration(), void NVIC_Configuration(), void SPI_DMA_Init().
Behavior description	This three functions configure: system timer, interrupt system, DMA.
Preconditions	—
Called functions	STM32 library functions.

Example 1

```
RCC_Configuration();
NVIC_Configuration();
SPI_DMA_Init();
```

3.2.2 Function: setup_driver

Table 4. setup_driver function

Filename	Description	
Function name	setup_driver.	
Function prototype	int setup_driver(u16 NumberOfDrivers, u16 RegLen, u16 ThermalBehav, u16 PWMcounter, u16 PWMmode, u16 Synchronisation, u16 ThermShut, u16 PWCLKalert, u16 BitConvertion16to12).	
Behavior description	Configure led drivers over SPI and internal variables for requested parameters of dimming.	
Input parameter1	NumberOfDrivers:	Indicates, how many LED drivers are connected to our system.
Input parameter2	RegLen:	RegLen256bit, RegLen16bit(1).
Input parameter3	ThermalBehav:	ThermalOverTemp, ThermalSafe(1).
Input parameter4	PWMcounter:	PWMresolution16, PWMresolution12(1).
Input parameter5	PWMmode:	PWMmode64MSB, PWMmode16MSB, PWMmode4MSB, PWMmode12bit(1).
Input parameter6	Synchronisation:	ManualSynchro, AutoSynchro(1).
Input parameter7	ThermShut:	ThermalShutEnable, ThermalShutDisable(1).
Input parameter8	PWCLKalert:	PWCLKalertDisable, PWCLKalertEnable(1).
Input parameter9	BitConvertion16to12:	Do16to12BitConvertion, DoNot16to12BitConvertion (before sending the data to driver conversion into 12-bit representation possible).
Output parameter	Return SUCCESS if requested parameters are achievable.	
Preconditions	RCC_Configuration(), NVIC_Configuration(), SPI_DMA_Init().	
Called functions	STM32 library functions for SPI.	

1. See datasheet STP1612PW05.

Example 2

```
/* We have 3 STP1612PW05 drivers connected, need 12-bit PWM depth*/
      RCC_Configuration();
      NVIC_Configuration();
      SPI_DMA_Init();
setup_driver( 3, RegLen16bit, ThermalSave, PWMresolution12,
                 PWMmode12bit, ManualSynchro, ThermalShutEnable,
                 PWCLKalertEnable, Do16to12BitConvertion);
      use_new_LEDmap(LEDmapClear);
```

3.2.3 Function: resetup_driver

Table 5. resetup_driver function

Filename	Description
Function name	resetup_drive.
Function prototype	int resetup_driver(u16 data).
Behavior description	Has the same functionality as setup_driver function, but the data are passed directly as one 16-bit number.
Input parameter1	data: global variable actual_configuration set by setup_driver can be used.
Output parameter	It always returns 0.
Preconditions	—
Called functions	sendData().

Example 3

```
resetup_driver(actual_configuration | ManualSynchro);
resetup_driver(actual_configuration & (~PWMresolution16));
error_detection(LEDmapClear,48);
resetup_driver(actual_configuration);
```

3.2.4 Function: error_detection

Table 6. error_detection

Filename	Description
Function name	error_detection.
Function prototype	unsigned char error_detection_DM(unsigned char *LEDmapIN, int LEDcount).
Behavior description	Function detects defect LEDs. LEDmapIN will have zeroes on the defect position.
Input parameter1	*LEDmapIN: array of unsigned shorts, the length of the array is LEDcount.
Input parameter2	Not used (function uses number of drive parameter from setup_driver function).
Output parameter	It returns 1 if any error occurred.
Preconditions	setup_driver().
Called functions	sendData().

Example 4

```
unsigned short LEDmapClear[48];
error_detection(LEDmapClear, 48);
```

3.2.5 Function: use_new_LEDmap

Table 7. use_new_LEDmap function

Filename	Description
Function name	use_new_LEDmap.
Function prototype	void use_new_LEDmap(unsigned short LEDmap[]).
Behavior description	It exchanges data shown by LED drivers. For glitch, see setup_driver function. Function convert data to 12-bit representation if set in setup_driver function.
Input parameter1	LEDmap[]: array of unsigned shorts describing the light intensity of LEDs.
Preconditions	setup_driver().
Called functions	sendData().

Example 5

```
/* generation of the pictogram with bidirectional blue-green arrow*/
const unsigned short USB_connected[48] =
  //B1    //G1    //R1      //B2    //G2    //R2      //B3    //G3    //R3
{0x0000, 0x0000, 0x0000,   0x0000, 0xFFFF, 0x0000,   0x0000, 0x0000, 0x0000,
 0x0000, 0xFFFF, 0x0000,   0xFFFF, 0x0000, 0x0000,   0x0000, 0xFFFF, 0x0000,
 0x0000, 0x0000, 0x0000,   0xFFFF, 0x0000, 0x0000,   0x0000, 0x0000, 0x0000,
 0x0000, 0xFFFF, 0x0000,   0xFFFF, 0x0000, 0x0000,   0x0000, 0xFFFF, 0x0000,
 0x0000, 0x0000, 0x0000,   0x0000, 0xFFFF, 0x0000,   0x0000, 0x0000, 0x0000,
 0x0000, 0x0000, 0x0000};
RCC_Configuration();
      NVIC_Configuration();
      SPI_DMA_Init();
setup_driver( 3, RegLen16bit, ThermalSave, PWMresolution12,
                 PWMmode12bit, ManualSynchro, ThermalShutEnable,
                 PWCLKalertEnable, Do16to12BitConvertion);
      use_new_LEDmap((unsigned short*)USB_connected);
```

3.2.6 Function: fillDriverPattern

Table 8. fillDriverPattern function

Filename	Description
Function name	fillDriverPattern.
Function prototype	void fillDriverPattern(int LEDcount, unsigned char pattern).
Behavior description	It shows on the LEDs one out of the five pictograms.
Input parameter1	LEDcount: not used, information used from number of drivers in setup_driver.
Input parameter2	Pattern: 1 - AllZeros, set all the LEDs to dark 2 - AllOnes, set all LEDs to maximum light 3 - USB_check, pictogram of one directional arrow, blue-green 4 - USB_not_connected, pictogram of bidirectional arrow, red-blue-green 5 - USB_connected, pictogram of bidirectional arrow, green-blue-green.
Preconditions	setup_driver().
Called functions	use_new_LEDmap().

Example 6

```
fillDriverPattern(48, 2);
emberSerialInit();  //USB detection and enumeration
if(usb_plugged)
  fillDriverPattern(48, 4);
else
  fillDriverPattern(48, 3);
waitAbit(2000);
fillDriverPattern(48, 0);
```

3.3 LEVEL 0 - stm32f10x_it.c

Functions the file offers

This interrupts handlers are used by this project:

```
EXTI15_10_IRQHandler
USB_LP_CAN_RX0_IRQHandler
(SysTickHandler                    - optionally)
(DMAChannel3_IRQHandler - optionally)
(SPI1_IRQHandler                - optionally)
```

Detailed function description

All the functions are called by microcontroller when interrupt occurs, so user cannot call them directly. These interrupts handlers can be found in STM32f10x_it.c file.

3.4 LEVEL 1

Constants and functions the file offers

There are no functions. The LEVEL 1 is kept for legacy reason.

3.5 LEVEL 2 - board_control.c (h)

Constants and functions the file offers

```
void BC_ADC_Configuration(void);
void BC_GPIO_Configuration(void);
void BC_EXTI_Configuration(void);
void BC_NVIC_Configuration(void);
struct buttonsAndADC;
```

External functions and types that are called from this file

ADC, GPIO, EXTI, (NVIC - optional) functions.

3.5.1 Detailed functions description: BC_ADC_Configuration, BC_GPIO_Configuration, BC_EXTI_Configuration, BC_NVIC_Configuration

Table 9. setup function

Filename	Description
Function name	BC_ADC_Configuration, BC_GPIO_Configuration, BC_EXTI_Configuration, BC_NVIC_Configuration.
Function prototype	void BC_ADC_Configuration(void), void BC_GPIO_Configuration(void), void BC_EXTI_Configuration(void), void BC_NVIC_Configuration(void).
Behavior description	This four functions configure: ADC for knob, GPIO for buttons, external interrupts for buttons, interrupt controller for buttons. All configuration is prepared for STEVAL-ILL028V1 evaluation kit.
Preconditions	—
Called functions	See source code of this library.

Example 7

```
/* BC_ADC_Configuration must be called before using structure:
unsigned char buttonsAndADC.(*buttonKnob)(void);*/
```

3.6 LEVEL 2 - led_demonstration.c (.h)

Constants and functions the file offers

```
modeSelect
tetris_color
```

`solid_color_demo`

`wave_color_demo`

`error_demo`

`writeCharacter`

External functions and types that are called from this file

Structures used: environment, buttonsAndADC.

No HW related functions are called from this file.

3.6.1 Detailed function description: modeSelect

Table 10. modeSelect function

Filename	Description
Function name	modeSelect.
Function prototype	void modeSelect(unsigned char *LEDmap, struct buttonsAndADC *btnADC, struct environment *env, signed int *userData).
Behavior description	Provides a user interface. Using buttons, user can choose one letter out of more in order to choose preferred application.
Input parameter1	*LEDmap: input and output brightness display map
Input parameter2	*btnADC: global variable buttonsADC
Input parameter3	*env: struct environment (described at the beginning of this document)
Input parameter4	*userData: array of signed integers, length of 11.
Output parameter	userData[0] return index which is relevant to the chosen letter. userData[0] returns 100 if no letter has not been chosen yet.
Preconditions	struct environment, *env: must be filled with correct data. BC_ADC_Configuration must, BC_GPIO_Configuration, BC_EXTI_Configuration, BC_NVIC_Configuration should be called before using of this function.
Called functions	buttonsAndADC->buttonKnob(), writeCharacter().

Example 8

```
/* See the complete code in the chapter: LEVEL 3 - main.c */
```

3.6.2 Function: writeCharacter

Table 11. writeCharacter function

Filename	Description
Function name	writeCharacter.
Function prototype	void writeCharacter(unsigned char *LEDmap, unsigned char *mask, unsigned short brightnessR, unsigned short brightnessG, unsigned short brightnessB, unsigned char Rotate).
Behavior description	It sets bytes in the *LEDmap array according to the requested character to be displayed.
Input parameter1	*LEDmap: an output brightness display map.
Input parameter2	*mask: table with the character shape.
Input parameter3, 4, 5	brightnessR, brightnessG, brightnessB: defines the color of the generated character.
Input parameter4	rotate: 0 - no rotation, 1-rotation: change character orientation up side down.
Preconditions	—
Called functions	—

Example 9

```
/* .. .. ..
writeCharacter(LEDmap, (unsigned char *) number_mask[n100], 0,
0x50, 0, ROTATE);
.. .. ..*/
```

3.6.3 Functions: tetris_color, solid_color_demo, wave_color_demo, error_demo

Table 12. tetris_color, solid_color_demo, wave_color_demo, error_demo functions

Filename	Description
Function name	tetris_color, solid_color_demo, wave_color_demo, error_demo.

Function prototype	void tetris_color(unsigned char *LEDmap, struct buttonsAndADC *btnADC, struct environment *env, signed int *userData);
	void solid_color_demo(unsigned char *LEDmap, struct buttonsAndADC *btnADC, struct environment *env, signed int *userData);
	void wave_color_demo(unsigned char *LEDmap, struct buttonsAndADC *btnADC, struct environment *env, signed int *userData);
	void error_demo(unsigned char *LEDmap, struct buttonsAndADC *btnADC, struct environment *env, signed int *userData).

Behavior description	Shows different types of LED demonstrations. They never ask for termination.
Input parameter1	*LEDmap: input and output brightness display map.
Input parameter2	*btnADC: global variable buttonsADC.
Input parameter3	*env: struct environment (described at the beginning of this document).
Input parameter4	*userData: array of signed integers, length of 11.
Preconditions	struct environment, *env: must be filled with correct data. BC_ADC_Configuration must, BC_GPIO_Configuration, BC_EXTI_Configuration, BC_NVIC_Configuration should be called before using of this function.
Called functions	buttonsAndADC->buttonKnob(), writeCharacter().

Example 10

```
/* See the complete code in the chapter: LEVEL 3 - main.c */
```

3.7 LEVEL 3 - main.c

Functions the file offers

`waitAbit(), main().`

Functions that are called from LEVEL 2, 1, 0

All functions from Level 2, a few from level 1 and exceptionally from level 0 are called from this unit. It is recommended to use as high level as possible while solving any task.

External Functions that are called from this file

No directly hardware related functions. Could be any from level 0 to level 2.

Detailed function description

```
int main(void){
  RCC_Configuration();
  NVIC_Configuration();
  SPI_DMA_Init();
setup_driver( 3, RegLen16bit, ThermalSave, PWMresolution12,
PWMmode12bit, ManualSynchro, ThermalShutEnable, PWCLKalertEnable,
Do16to12BitConvertion);
BC_GPIO_Configuration();
  BC_EXTI_Configuration();
  BC_NVIC_Configuration();
  BC_ADC_Configuration();
env.columns = 3;
  env.rows = 5;
  env.setupGame = 1;
  env.exitRequest = 0;
  #define modes 4;
modeSelector = 100;
  userData[1] = modes;
while(1){
    switch(modeSelector){
case   0: tetris_color(LEDmapClear, &buttonsADC, &env,
&(userData[0]));             //A - tetris
                  break;
    case   1: wave_color_demo(LEDmapClear, &buttonsADC, &env,
&(userData[0]));  //B - WAVE demo
                  break;
    case   2: solid_color_demo(LEDmapClear, &buttonsADC, &env,
&(userData[0]));  //C - solid color demo
                  break;
    case   3: error_demo(LEDmapClear, &buttonsADC, &env,
&(userData[0]));               //D - error detection
                  if(env.gameFinished){
                     env.gameFinished = 0;
                     error_detection(LEDmapClear,48);
                  }
            break;
```

```
    case 100: modeSelect(LEDmapClear, &buttonsADC, &env,
&(userData[0]));
                if(env.gameFinished){
                  env.setupGame = 1;
                  env.gameFinished = 0;
                  modeSelector = userData[0];
                  break;
                  }
    default:modeSelector = 100;break;
    }//case
    if(env.refreshRequest){
       use_new_LEDmap(LEDmapClear);
       env.refreshRequest = 0;
    }
    if(env.exitRequest){
      modeSelector = 100; userData[1] = modes; env.setupGame = 1;
      env.gameFinished = 0; env.exitRequest = 0;
    }
  }//while
}//main
```

4 Revision history

Table 13. Document revision history

Date	Revision	Changes
18-Dec-2009	1	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com