/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file Templates/Src/main.c

\* @author MCD Application Team

\* @version V1.0.3

\* @date 22-April-2016

\* @brief STM32F7xx HAL API Template project

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>

\*

\* Redistribution and use in source and binary forms, with or without modification,

\* are permitted provided that the following conditions are met:

\* 1. Redistributions of source code must retain the above copyright notice,

\* this list of conditions and the following disclaimer.

\* 2. Redistributions in binary form must reproduce the above copyright notice,

\* this list of conditions and the following disclaimer in the documentation

\* and/or other materials provided with the distribution.

\* 3. Neither the name of STMicroelectronics nor the names of its contributors

\* may be used to endorse or promote products derived from this software

\* without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

\* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

\* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE

\* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

\* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

\* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

\* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,

\* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

\* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

/\*\* @addtogroup STM32F7xx\_HAL\_Examples

\* @{

\*/

/\*\* @addtogroup Templates

\* @{

\*/

/\* Private typedef -----------------------------------------------------------\*/

/\* Private define ------------------------------------------------------------\*/

/\* Private macro -------------------------------------------------------------\*/

/\* Private variables ---------------------------------------------------------\*/

/\* Private function prototypes -----------------------------------------------\*/

static void SystemClock\_Config(void);

static void Error\_Handler(void);

static void MPU\_Config(void);

static void CPU\_CACHE\_Enable(void);

/\* Private functions ---------------------------------------------------------\*/

/\*\*

\* @brief Main program

\* @param None

\* @retval None

\*/

int main(void)

{

/\* This project template calls firstly two functions in order to configure MPU feature

and to enable the CPU Cache, respectively MPU\_Config() and CPU\_CACHE\_Enable().

These functions are provided as template implementation that User may integrate

in his application, to enhance the performance in case of use of AXI interface

with several masters. \*/

/\* Configure the MPU attributes as Write Through \*/

MPU\_Config();

/\* Enable the CPU Cache \*/

CPU\_CACHE\_Enable();

/\* STM32F7xx HAL library initialization:

- Configure the Flash ART accelerator on ITCM interface

- Configure the Systick to generate an interrupt each 1 msec

- Set NVIC Group Priority to 4

- Low Level Initialization

\*/

HAL\_Init();

/\* Configure the System clock to have a frequency of 216 MHz \*/

SystemClock\_Config();

/\* Add your application code here \*/

// BSP\_LED\_Init(LED\_GREEN);

// BSP\_LED\_On(LED\_GREEN);

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE(); // we need to enable the GPIOA port's clock first

GPIO\_InitTypeDef tda; // create a config structure

tda.Pin = GPIO\_PIN\_0; // this is about PIN 0

tda.Mode = GPIO\_MODE\_OUTPUT\_PP; // Configure as output with push-up-down enabled

tda.Pull = GPIO\_PULLDOWN; // the push-up-down should work as pulldown

tda.Speed = GPIO\_SPEED\_HIGH; // we need a high-speed output

HAL\_GPIO\_Init(GPIOA, &tda); // initialize the pin on GPIOA port with HAL

\_\_HAL\_RCC\_GPIOF\_CLK\_ENABLE(); // GPIOF port's first use so we need to declare

GPIO\_InitTypeDef tda1;

tda1.Pin = GPIO\_PIN\_10;

tda1.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda1.Pull = GPIO\_PULLDOWN;

tda1.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOF, &tda1);

GPIO\_InitTypeDef tda2;

tda2.Pin = GPIO\_PIN\_9;

tda2.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda2.Pull = GPIO\_PULLDOWN;

tda2.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOF, &tda2);

GPIO\_InitTypeDef tda3;

tda3.Pin = GPIO\_PIN\_8;

tda3.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda3.Pull = GPIO\_PULLDOWN;

tda3.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOF, &tda3);

GPIO\_InitTypeDef tda4;

tda4.Pin = GPIO\_PIN\_7;

tda4.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda4.Pull = GPIO\_PULLDOWN;

tda4.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOF, &tda4);

GPIO\_InitTypeDef tda5;

tda5.Pin = GPIO\_PIN\_6;

tda5.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda5.Pull = GPIO\_PULLDOWN;

tda5.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOF, &tda5);

BSP\_PB\_Init(BUTTON\_KEY, BUTTON\_MODE\_GPIO);

// for the button, button is in D00 -> pc7

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

GPIO\_InitTypeDef tda6;

tda6.Pin = GPIO\_PIN\_7;

tda6.Mode = GPIO\_MODE\_INPUT;

tda6.Pull = GPIO\_PULLUP;

tda6.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOC, &tda6);

GPIO\_InitTypeDef tda7;

tda7.Pin = GPIO\_PIN\_6;

tda7.Mode = GPIO\_MODE\_INPUT;

tda7.Pull = GPIO\_PULLUP;

tda7.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOC, &tda7);

\_\_HAL\_RCC\_GPIOG\_CLK\_ENABLE();

GPIO\_InitTypeDef tda8;

tda8.Pin = GPIO\_PIN\_6;

tda8.Mode = GPIO\_MODE\_INPUT;

tda8.Pull = GPIO\_PULLUP;

tda8.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOG, &tda8);

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

GPIO\_InitTypeDef tda9;

tda9.Pin = GPIO\_PIN\_4;

tda9.Mode = GPIO\_MODE\_INPUT;

tda9.Pull = GPIO\_PULLUP;

tda9.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOG, &tda9);

//matrix rows

GPIO\_InitTypeDef tda10;

tda10.Pin = GPIO\_PIN\_8;

tda10.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda10.Pull = GPIO\_PULLDOWN;

tda10.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOB, &tda10);

GPIO\_InitTypeDef tda11;

tda11.Pin = GPIO\_PIN\_9;

tda11.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda11.Pull = GPIO\_PULLDOWN;

tda11.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOB, &tda11);

GPIO\_InitTypeDef tda12;

tda12.Pin = GPIO\_PIN\_14;

tda12.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda12.Pull = GPIO\_PULLDOWN;

tda12.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOB, &tda12);

\_\_HAL\_RCC\_GPIOI\_CLK\_ENABLE();

GPIO\_InitTypeDef tda13;

tda13.Pin = GPIO\_PIN\_1;

tda13.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda13.Pull = GPIO\_PULLDOWN;

tda13.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOI, &tda13);

GPIO\_InitTypeDef tda14;

tda14.Pin = GPIO\_PIN\_15;

tda14.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda14.Pull = GPIO\_PULLDOWN;

tda14.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOB, &tda14);

//matrix columns

GPIO\_InitTypeDef tda15;

tda15.Pin = GPIO\_PIN\_8; // D10 PA8

tda15.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda15.Pull = GPIO\_PULLDOWN;

tda15.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOA, &tda15);

GPIO\_InitTypeDef tda16;

tda16.Pin = GPIO\_PIN\_15; // D9 PA15

tda16.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda16.Pull = GPIO\_PULLDOWN;

tda16.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOA, &tda16);

GPIO\_InitTypeDef tda17;

tda17.Pin = GPIO\_PIN\_2; // D8 PI2

tda17.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda17.Pull = GPIO\_PULLDOWN;

tda17.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOI, &tda17);

GPIO\_InitTypeDef tda18;

tda18.Pin = GPIO\_PIN\_3; // D7 PI3

tda18.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda18.Pull = GPIO\_PULLDOWN;

tda18.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOI, &tda18);

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

GPIO\_InitTypeDef tda19;

tda19.Pin = GPIO\_PIN\_6; // D6 PH6

tda19.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda19.Pull = GPIO\_PULLDOWN;

tda19.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOH, &tda19);

GPIO\_InitTypeDef tda20;

tda20.Pin = GPIO\_PIN\_0; // D5 PI0

tda20.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda20.Pull = GPIO\_PULLDOWN;

tda20.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOI, &tda20);

GPIO\_InitTypeDef tda21;

tda21.Pin = GPIO\_PIN\_7; // D4 PG7

tda21.Mode = GPIO\_MODE\_OUTPUT\_PP;

tda21.Pull = GPIO\_PULLDOWN;

tda21.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(GPIOG, &tda21);

/\*

GPIO\_PinState led\_matrix\_state[LED\_MATRIX\_ROWS][LED\_MATRIX\_COLS];

const led\_matrix\_pin\_t rows[] = {

{GPIOA, GPIO\_PIN\_8}, //row[0]

{GPIOA, GPIO\_PIN\_15}, //row[1]

{GPIOI, GPIO\_PIN\_2}, //row[2]

{GPIOI, GPIO\_PIN\_3}, //row[3]

{GPIOH, GPIO\_PIN\_6}, //row[4]

{GPIOI, GPIO\_PIN\_0}, //row[5]

{GPIOG, GPIO\_PIN\_7} //row[6]

};

const led\_matrix\_pin\_t cols[] = {

{GPIOB, GPIO\_PIN\_14}, //col[0]

{GPIOB, GPIO\_PIN\_8}, //col[1]

{GPIOB, GPIO\_PIN\_9}, //col[2]

{GPIOB, GPIO\_PIN\_15}, //col[3]

{GPIOI, GPIO\_PIN\_1} //col[4]

};

\*/

/\*

char a[5][7] = {

{GPIOA, GPIO\_PIN\_8, GPIOA, GPIO\_PIN\_15, GPIOI, GPIO\_PIN\_2, GPIOI, GPIO\_PIN\_3, GPIOH, GPIO\_PIN\_6, {GPIOI, GPIO\_PIN\_0, GPIOG, GPIO\_PIN\_7} , /\* initializers for row indexed by 0 \*/

// {4, 5, 6, 7} , /\* initializers for row indexed by 1 \*/

// {8, 9, 10, 11} /\* initializers for row indexed by 2 \*/

// };

/\* Infinite loop \*/

while (1)

{

/\* for ( int i = 0; i < 5; i++ ) {

for (int j = 0; j < 7; j++ ) {

printf("a[%c][%c] = %d\n", i,j, a[i][j] );

}

}

\*/

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_14, GPIO\_PIN\_SET); // 1.sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 2. SOR

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 3. sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 4. sor

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_1, GPIO\_PIN\_SET); // 5. sor

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, GPIO\_PIN\_RESET); // 1. oszlop

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 2. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_2, GPIO\_PIN\_SET); // 3. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_3, GPIO\_PIN\_SET); // 4. oszlop

HAL\_GPIO\_WritePin(GPIOH, GPIO\_PIN\_6, GPIO\_PIN\_SET); // 5. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 6. oszlop

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7, GPIO\_PIN\_SET); // 7. oszlop

//Flashes

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1. oszlop

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_RESET); // 2. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_2, GPIO\_PIN\_SET); // 3. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_3, GPIO\_PIN\_SET); // 4. oszlop

HAL\_GPIO\_WritePin(GPIOH, GPIO\_PIN\_6, GPIO\_PIN\_SET); // 5. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 6. oszlop

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7, GPIO\_PIN\_SET); // 7. oszlop

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1. oszlop

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 2. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_2, GPIO\_PIN\_RESET); // 3. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_3, GPIO\_PIN\_SET); // 4. oszlop

HAL\_GPIO\_WritePin(GPIOH, GPIO\_PIN\_6, GPIO\_PIN\_SET); // 5. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 6. oszlop

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7, GPIO\_PIN\_SET); // 7. oszlop

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1. oszlop

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 2. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_2, GPIO\_PIN\_SET); // 3. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_3, GPIO\_PIN\_RESET); // 4. oszlop

HAL\_GPIO\_WritePin(GPIOH, GPIO\_PIN\_6, GPIO\_PIN\_SET); // 5. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 6. oszlop

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7, GPIO\_PIN\_SET); // 7. oszlop

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1. oszlop

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 2. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_2, GPIO\_PIN\_SET); // 3. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_3, GPIO\_PIN\_SET); // 4. oszlop

HAL\_GPIO\_WritePin(GPIOH, GPIO\_PIN\_6, GPIO\_PIN\_RESET); // 5. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 6. oszlop

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7, GPIO\_PIN\_SET); // 7. oszlop

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1. oszlop

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 2. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_2, GPIO\_PIN\_SET); // 3. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_3, GPIO\_PIN\_SET); // 4. oszlop

HAL\_GPIO\_WritePin(GPIOH, GPIO\_PIN\_6, GPIO\_PIN\_SET); // 5. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_0, GPIO\_PIN\_RESET); // 6. oszlop

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7, GPIO\_PIN\_SET); // 7. oszlop

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1. oszlop

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 2. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_2, GPIO\_PIN\_SET); // 3. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_3, GPIO\_PIN\_SET); // 4. oszlop

HAL\_GPIO\_WritePin(GPIOH, GPIO\_PIN\_6, GPIO\_PIN\_SET); // 5. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 6. oszlop

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7, GPIO\_PIN\_RESET); // 7. oszlop

HAL\_Delay(200);

// rows

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, GPIO\_PIN\_RESET); // 1. oszlop

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_15, GPIO\_PIN\_RESET); // 2. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_2, GPIO\_PIN\_RESET); // 3. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_3, GPIO\_PIN\_RESET); // 4. oszlop

HAL\_GPIO\_WritePin(GPIOH, GPIO\_PIN\_6, GPIO\_PIN\_RESET); // 5. oszlop

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_0, GPIO\_PIN\_RESET); // 6. oszlop

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7, GPIO\_PIN\_RESET); // 7. oszlop

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_14, GPIO\_PIN\_SET); // 1.sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 2. SOR

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 3. sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 4. sor

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_1, GPIO\_PIN\_SET); // 5. sor

HAL\_Delay(400);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_14, GPIO\_PIN\_RESET); // 1.sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 2. SOR

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 3. sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 4. sor

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_1, GPIO\_PIN\_SET); // 5. sor

HAL\_Delay(400);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_14, GPIO\_PIN\_SET); // 1.sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_8, GPIO\_PIN\_RESET); // 2. SOR

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 3. sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 4. sor

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_1, GPIO\_PIN\_SET); // 5. sor

HAL\_Delay(400);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_14, GPIO\_PIN\_SET); // 1.sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 2. SOR

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_9, GPIO\_PIN\_RESET); // 3. sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 4. sor

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_1, GPIO\_PIN\_SET); // 5. sor

HAL\_Delay(400);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_14, GPIO\_PIN\_SET); // 1.sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 2. SOR

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 3. sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_15, GPIO\_PIN\_RESET); // 4. sor

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_1, GPIO\_PIN\_SET); // 5. sor

HAL\_Delay(400);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_14, GPIO\_PIN\_SET); // 1.sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 2. SOR

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 3. sor

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_15, GPIO\_PIN\_SET); // 4. sor

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_1, GPIO\_PIN\_RESET); // 5. sor

HAL\_Delay(400);

if(HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_4) == 0) {

for(int i = 0; i < 64; i++){

int remain = 0;

int x = 0;

x = i / 32;

remain = i % 32;

if( x ==1 ){

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1.

} else{

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

}

x = remain / 16;

remain = remain % 16;

if( x ==1 ){

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 2.

} else{

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

}

x = remain / 8;

remain = remain % 8;

if( x ==1 ){

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_SET); // 3.

} else{

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

}

x = remain / 4;

remain = remain % 4;

if( x ==1 ){

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 4.

} else{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

}

x = remain / 2;

remain = remain % 2;

if( x ==1 ){

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_SET); //5.

} else{

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

}

x = remain / 1;

remain = remain % 1;

if( x ==1 ){

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_SET); //6.

} else{

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

}

HAL\_Delay(500);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

}

}

if(HAL\_GPIO\_ReadPin(GPIOC, GPIO\_PIN\_7) == 0) {

//when the not integrated button pressed all leds fleshes from outside to the middle

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_SET); //6.

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 2.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_SET); //5.

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 4.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_SET); // 3.

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_Delay(300);

}

if(HAL\_GPIO\_ReadPin(GPIOC, GPIO\_PIN\_6) == 0) {

//when the not integrated button2 pressed all leds fleshes

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_SET); //6.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 2.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_SET); //5.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_SET); // 3.

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 4.

} else {

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

}

if(HAL\_GPIO\_ReadPin(GPIOG, GPIO\_PIN\_6) == 0) {

//when the not integrated button3 pressed all leds fleshes in a row

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1.

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 2.

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_SET); // 3.

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 4.

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_SET); //5.

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_SET); //6.

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_Delay(100);

// backwards

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_SET);

HAL\_Delay(100);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_Delay(100);

}

/\*

if(HAL\_GPIO\_ReadPin(GPIOC, GPIO\_PIN\_7) == 0) { //when the not integrated button pressed all leds fleshes

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_SET); //6.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 2.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_SET); //5.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_SET); // 3.

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 4.

} else {

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

}

\*/

/\*

if(BSP\_PB\_GetState(BUTTON\_KEY)) { //when button pressed all leds fleshes

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_SET); //6.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 2.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_SET); //5.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_SET); // 3.

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 4.

} else {

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

}

\*/

/\*

if(BSP\_PB\_GetState(BUTTON\_KEY)) { //when button pressed leds fleshes from the outside to the middle

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_SET); // 1.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_SET); //6.

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_8, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_SET); // 2.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_SET); //5.

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, GPIO\_PIN\_RESET);

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_SET); // 4.

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_SET); // 3.

HAL\_Delay(300);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_10, GPIO\_PIN\_RESET);

HAL\_Delay(300);

}

\*/

//TODO: Flash the led with 200 ms period time

//BSP\_LED\_On(LED\_GREEN);

//HAL\_Delay(200);

//BSP\_LED\_Off(LED\_GREEN); when u use toogle dont need on and of functions

//BSP\_LED\_Toggle(LED\_GREEN);

//HAL\_Delay(200);

}

}

/\*\*

\* @brief System Clock Configuration

\* The system Clock is configured as follow :

\* System Clock source = PLL (HSE)

\* SYSCLK(Hz) = 216000000

\* HCLK(Hz) = 216000000

\* AHB Prescaler = 1

\* APB1 Prescaler = 4

\* APB2 Prescaler = 2

\* HSE Frequency(Hz) = 25000000

\* PLL\_M = 25

\* PLL\_N = 432

\* PLL\_P = 2

\* PLL\_Q = 9

\* VDD(V) = 3.3

\* Main regulator output voltage = Scale1 mode

\* Flash Latency(WS) = 7

\* @param None

\* @retval None

\*/

static void SystemClock\_Config(void)

{

RCC\_ClkInitTypeDef RCC\_ClkInitStruct;

RCC\_OscInitTypeDef RCC\_OscInitStruct;

/\* Enable HSE Oscillator and activate PLL with HSE as source \*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_OFF;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 25;

RCC\_OscInitStruct.PLL.PLLN = 432;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 9;

if(HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\* activate the OverDrive to reach the 216 Mhz Frequency \*/

if(HAL\_PWREx\_EnableOverDrive() != HAL\_OK)

{

Error\_Handler();

}

/\* Select PLL as system clock source and configure the HCLK, PCLK1 and PCLK2

clocks dividers \*/

RCC\_ClkInitStruct.ClockType = (RCC\_CLOCKTYPE\_SYSCLK | RCC\_CLOCKTYPE\_HCLK | RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2);

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

if(HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_7) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @param None

\* @retval None

\*/

static void Error\_Handler(void)

{

/\* User may add here some code to deal with this error \*/

while(1)

{

}

}

/\*\*

\* @brief Configure the MPU attributes as Write Through for SRAM1/2.

\* @note The Base Address is 0x20010000 since this memory interface is the AXI.

\* The Region Size is 256KB, it is related to SRAM1 and SRAM2 memory size.

\* @param None

\* @retval None

\*/

static void MPU\_Config(void)

{

MPU\_Region\_InitTypeDef MPU\_InitStruct;

/\* Disable the MPU \*/

HAL\_MPU\_Disable();

/\* Configure the MPU attributes as WT for SRAM \*/

MPU\_InitStruct.Enable = MPU\_REGION\_ENABLE;

MPU\_InitStruct.BaseAddress = 0x20010000;

MPU\_InitStruct.Size = MPU\_REGION\_SIZE\_256KB;

MPU\_InitStruct.AccessPermission = MPU\_REGION\_FULL\_ACCESS;

MPU\_InitStruct.IsBufferable = MPU\_ACCESS\_NOT\_BUFFERABLE;

MPU\_InitStruct.IsCacheable = MPU\_ACCESS\_CACHEABLE;

MPU\_InitStruct.IsShareable = MPU\_ACCESS\_SHAREABLE;

MPU\_InitStruct.Number = MPU\_REGION\_NUMBER0;

MPU\_InitStruct.TypeExtField = MPU\_TEX\_LEVEL0;

MPU\_InitStruct.SubRegionDisable = 0x00;

MPU\_InitStruct.DisableExec = MPU\_INSTRUCTION\_ACCESS\_ENABLE;

HAL\_MPU\_ConfigRegion(&MPU\_InitStruct);

/\* Enable the MPU \*/

HAL\_MPU\_Enable(MPU\_PRIVILEGED\_DEFAULT);

}

/\*\*

\* @brief CPU L1-Cache enable.

\* @param None

\* @retval None

\*/

static void CPU\_CACHE\_Enable(void)

{

/\* Enable I-Cache \*/

SCB\_EnableICache();

/\* Enable D-Cache \*/

SCB\_EnableDCache();

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t\* file, uint32\_t line)

{

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* Infinite loop \*/

while (1)

{

}

}

#endif

/\*\*

\* @}

\*/

/\*\*

\* @}

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/