#include "main.h"

static void SystemClock\_Config(void);

static void Error\_Handler(void);

static void MPU\_Config(void);

static void CPU\_CACHE\_Enable(void);

int main(void)

{

MPU\_Config();

/\* Enable the CPU Cache \*/

CPU\_CACHE\_Enable();

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);

/\* Infinite loop \*/

while (1)

{

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();

}

}

static void Error\_Handler(void)

{

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

while(1)

{

}

}

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);

}

static void CPU\_CACHE\_Enable(void)

{

SCB\_EnableICache();

SCB\_EnableDCache();

}

#ifdef USE\_FULL\_ASSERT

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