/\*\*

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

\* @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"

#include <string.h>

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

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

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

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

UART\_HandleTypeDef uart\_handle;

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

#ifdef \_\_GNUC\_\_

/\* With GCC/RAISONANCE, small printf (option LD Linker->Libraries->Small printf

set to 'Yes') calls \_\_io\_putchar() \*/

#define PUTCHAR\_PROTOTYPE int \_\_io\_putchar(int ch)

#else

#define PUTCHAR\_PROTOTYPE int fputc(int ch, FILE \*f)

#endif /\* \_\_GNUC\_\_ \*/

static void SystemClock\_Config(void);

static void Error\_Handler(void);

static void MPU\_Config(void);

static void CPU\_CACHE\_Enable(void);

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

void LEDInit();

void TimerITInit();

void uartInit();

GPIO\_InitTypeDef led;

GPIO\_InitTypeDef tx;

GPIO\_InitTypeDef rx;

UART\_HandleTypeDef uart\_handle;

TIM\_HandleTypeDef TimHandle;

TIM\_OC\_InitTypeDef sConfig;

/\*\*

\* @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();

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();

SystemClock\_Config(); /\* Configure the System clock to have a frequency of 216 MHz \*/

/\* Add your application code here \*/

BSP\_LED\_Init(LED\_GREEN);

uartInit();

LEDInit();

TimerITInit();

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" Welcome to the communication project!\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("Waiting for user input!\n");

/\* Infinite loop \*/

while (1)

{

/\* char input[10] = {};

HAL\_UART\_Receive(&uart\_handle, &input, 10, 1000);

if(strcmp(input, "on") == 0){

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

printf("\ninput: %s!\n", input);

}else if(strcmp(input, "off") == 0){

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

printf("\ninput: %s!\n", input);

} \*/

// HAL\_UART\_Receive\_IT(&uart\_handle, &input, 10);

char input[10] = {};

HAL\_UART\_Receive(&uart\_handle, &input, 10, 1000);

int error = 0;

if(strcmp(input, "on") == 0){

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

printf("\ninput: %s!\n", input);

}else if(strcmp(input, "off") == 0){

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

printf("\ninput: %s!\n", input);

} else if(strcmp(input, "on") != 0 && strcmp(input, "off") != 0 && strlen(input)>0 ) {

error = 1;

printf("\nWrong input!\n");

}

if (error == 1) {

for(int i = 0; i < 3; i++) {

BSP\_LED\_Toggle(LED\_GREEN);

HAL\_Delay(200);

}

error = 0;

} else {

BSP\_LED\_Off(LED\_GREEN);

}

} // end of while

} //end of the main

void LEDInit() {

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

led.Pin = GPIO\_PIN\_6;

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

led.Pull = GPIO\_PULLDOWN;

led.Speed = GPIO\_SPEED\_HIGH;

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

}

void TimerITInit() {

\_\_HAL\_RCC\_TIM1\_CLK\_ENABLE();

TimHandle.Instance = TIM1;

TimHandle.Init.Period = 1000;

TimHandle.Init.Prescaler = 50000;

TimHandle.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

TimHandle.Init.CounterMode = TIM\_COUNTERMODE\_UP;

HAL\_TIM\_Base\_Init(&TimHandle);

HAL\_TIM\_Base\_Start\_IT(&TimHandle);

HAL\_TIM\_PWM\_Init(&TimHandle);

sConfig.OCMode = TIM\_OCMODE\_PWM1;

sConfig.Pulse = 0;

HAL\_TIM\_PWM\_ConfigChannel(&TimHandle, &sConfig, TIM\_CHANNEL\_1);

HAL\_TIM\_PWM\_Start(&TimHandle, TIM\_CHANNEL\_1);

}

void uartInit(){

/\* Configure USART Tx as alternate function \*/

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

tx.Pin = GPIO\_PIN\_9;

tx.Mode = GPIO\_MODE\_AF\_PP;

tx.Pull = GPIO\_PULLUP;

tx.Alternate = GPIO\_AF7\_USART1;

tx.Speed = GPIO\_SPEED\_FAST;

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

/\* Configure USART Rx as alternate function \*/

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

rx.Pin = GPIO\_PIN\_7;

rx.Mode = GPIO\_MODE\_AF\_PP;

rx.Pull = GPIO\_PULLUP;

rx.Alternate = GPIO\_AF7\_USART1;

rx.Speed = GPIO\_SPEED\_FAST;

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

\_\_HAL\_RCC\_USART1\_CLK\_ENABLE();

uart\_handle.Instance = USART1;

uart\_handle.Init.BaudRate = 115200;

uart\_handle.Init.WordLength = UART\_WORDLENGTH\_8B;

uart\_handle.Init.StopBits = UART\_STOPBITS\_1;

uart\_handle.Init.Parity = UART\_PARITY\_NONE;

uart\_handle.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

uart\_handle.Init.Mode = UART\_MODE\_TX\_RX;

HAL\_UART\_Init(&uart\_handle);

}

/\*\*

\* @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

\*/

PUTCHAR\_PROTOTYPE {

/\* Place your implementation of fputc here \*/

/\* e.g. write a character to the EVAL\_COM1 and Loop until the end of transmission \*/

HAL\_UART\_Transmit(&uart\_handle, (uint8\_t \*) &ch, 1, 0xFFFF);

return ch;

}

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\*\*\*\*/