/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* This example uses HAL library calls to control

the GPIOC peripheral. You’ll be redoing this code

with hardware register access. \*/

/\*

//PART 1

//STEP 1

//\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE(); // Enable the GPIOC clock in the RCC

//\_\_IO uint32\_t APB1ENR; // Enable the GPIOC clock in the RRC < RCC APB1 peripheral clock enable register,

//Address offset: 0x1C

//(1) Enable the peripheral clock of GPIOA, GPIOB and GPIOC

//RCC->AHBENR |= RCC\_AHBENR\_GPIOAEN | RCC\_AHBENR\_GPIOBEN | RCC\_AHBENR\_GPIOCEN;

RCC->AHBENR |= RCC\_AHBENR\_GPIOCEN;

\*/

//PART 2

//STEP 1

RCC->AHBENR |= RCC\_AHBENR\_GPIOAEN;

RCC->AHBENR |= RCC\_AHBENR\_GPIOCEN;

/\*

//PART 1

//STEP 2

// Set up a configuration struct to pass to the initialization function

//GPIO\_InitTypeDef initStr = {GPIO\_PIN\_8 | GPIO\_PIN\_9, GPIO\_MODE\_OUTPUT\_PP, GPIO\_SPEED\_FREQ\_LOW, GPIO\_NOPULL};

//HAL\_GPIO\_Init(GPIOC, &initStr); // Initialize pins PC8 & PC9

//Set the pins to general-purpose output mode in the MODER register.

//GPIOC->MODER |= (1<<16); //01: General purpose output mode. Pin PC8(Orange)(bits 17:16) as Output (01). This means that we need to write a ‘1’ in the 16th position.

//GPIOC->MODER |= (1<<18); //Pin PC9(Green) (bits 19:18) as Output (01). This means that we need to write a '1' in the 18th position.

GPIOC->MODER |= (1<<12); //01: General purpose output mode. Pin PC6(Red)(bits 13:12) as Output (01). This means that we need to write a ‘1’ in the 12th position.

GPIOC->MODER |= (1<<14); //Pin PC7(Blue) (bits 15:14) as Output (01). This means that we need to write a '1' in the 14th position.

//Set the pins to push-pull output type in the OTYPER register.

//GPIOC->OTYPER &= ~(1<<8); // bit 8=0 --> Output push pull. Pin PC8(Orange).

//GPIOC->OTYPER &= ~(1<<9); // bit 9=0 --> Output push pull. Pin PC9(Green).

GPIOC->OTYPER &= ~(1<<6); // bit 6=0 --> Output push pull. Pin PC6(Red).

GPIOC->OTYPER &= ~(1<<7); // bit 7=0 --> Output push pull. Pin PC7(Blue).

//Set the pins to low speed in the OSPEEDER register.

//GPIOC->OSPEEDR |= ((1<<16) | (1<<17)); //Pin PC8(Orange) (bits 17:16) as Low Speed (0:0).

//GPIOC->OSPEEDR |= ((1<<18) | (1<<19)); //Pin PC9(Green) (bits 19:18) as Low Speed (0:0).

//Pin PC7(Blue) (bits 15:14) as Low Speed (0:0).

//Set to no pull-up/down resistors in the PUPDR register.

//GPIOC->PUPDR &= ~((1<<16) | (1<<17)); // Pin PC8(Orange) (bits 17:16) are 0:0 --> no pull up or pulldown

//GPIOC->PUPDR &= ~((1<<18) | (1<<19)); // Pin PC9(Green) (bits 19:18) are 0:0 --> no pull up or pulldown

GPIOC->PUPDR &= ~((1<<12) | (1<<13)); // Pin PC6(Red) (bits 13:12) are 0:0 --> no pull up or pulldown

GPIOC->PUPDR &= ~((1<<14) | (1<<15)); // Pin PC7(Blue) (bits 15:14) are 0:0 --> no pull up or pulldown

\*/

//PART 2

//STEP 2

// Set up a configuration struct to pass to the initialization function

//GPIO\_InitTypeDef initStr = {GPIO\_PIN\_8 | GPIO\_PIN\_9, GPIO\_MODE\_OUTPUT\_PP, GPIO\_SPEED\_FREQ\_LOW, GPIO\_NOPULL};

//HAL\_GPIO\_Init(GPIOC, &initStr); // Initialize pins PC8 & PC9

//Set the pins to general-purpose output mode in the MODER register.

//GPIOC->MODER |= (1<<16); //01: General purpose output mode. Pin PC8(Orange)(bits 17:16) as Output (01). This means that we need to write a ‘1’ in the 16th position.

//GPIOC->MODER |= (1<<18); //Pin PC9(Green) (bits 19:18) as Output (01). This means that we need to write a '1' in the 18th position.

GPIOC->MODER |= (1<<12); //01: General purpose output mode. Pin PC6(Red)(bits 13:12) as Output (01). This means that we need to write a ‘1’ in the 12th position.

GPIOC->MODER |= (1<<14); //Pin PC7(Blue) (bits 15:14) as Output (01). This means that we need to write a '1' in the 14th position.

//Set the pins to push-pull output type in the OTYPER register.

//GPIOC->OTYPER &= ~(1<<8); // bit 8=0 --> Output push pull. Pin PC8(Orange).

//GPIOC->OTYPER &= ~(1<<9); // bit 9=0 --> Output push pull. Pin PC9(Green).

GPIOC->OTYPER &= ~(1<<6); // bit 6=0 --> Output push pull. Pin PC6(Red).

GPIOC->OTYPER &= ~(1<<7); // bit 7=0 --> Output push pull. Pin PC7(Blue).

//Set the pins to low speed in the OSPEEDER register.

//GPIOC->OSPEEDR |= ((1<<16) | (1<<17)); //Pin PC8(Orange) (bits 17:16) as Low Speed (0:0).

//GPIOC->OSPEEDR |= ((1<<18) | (1<<19)); //Pin PC9(Green) (bits 19:18) as Low Speed (0:0).

//Pin PC7(Blue) (bits 15:14) as Low Speed (0:0).

//Set to no pull-up/down resistors in the PUPDR register.

//GPIOC->PUPDR &= ~((1<<16) | (1<<17)); // Pin PC8(Orange) (bits 17:16) are 0:0 --> no pull up or pulldown

//GPIOC->PUPDR &= ~((1<<18) | (1<<19)); // Pin PC9(Green) (bits 19:18) are 0:0 --> no pull up or pulldown

GPIOC->PUPDR &= ~((1<<12) | (1<<13)); // Pin PC6(Red) (bits 13:12) are 0:0 --> no pull up or pulldown

GPIOC->PUPDR &= ~((1<<14) | (1<<15)); // Pin PC7(Blue) (bits 15:14) are 0:0 --> no pull up or pulldow

//Enable the pull-down resistor in the PUPDR register.

GPIOA->PUPDR = 0x00100; // Button B1 (bits 1:0) are (1:0) --> enable pull-down.

GPIOC->ODR = 0;

/\*

GPIO\_InitTypeDef initStr = {GPIO\_PIN\_8 | GPIO\_PIN\_9, GPIO\_MODE\_OUTPUT\_PP, GPIO\_SPEED\_FREQ\_LOW, GPIO\_NOPULL};

HAL\_GPIO\_Init(GPIOC, &initStr); // Initialize pins PC8 & PC9

HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_8, GPIO\_PIN\_SET); // Start PC8 high

\*/

//PART 1

//STEP 3

//HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_8, GPIO\_PIN\_SET); // Start PC8 high

//GPIOC->BSRR |= (1<<8); // Set the Pin PC8. Turn LED ON.

//GPIOC->BSRR |= (1<<25); // Reset the Pin PC9. Turn LED OFF.

//GPIOC->ODR |= 1<<8; // Set the Pin PC8(Orange). Turn LED ON.

//GPIOC->ODR &= ~(1<<9); // Reset the Pin PC9(Green). Turn LED OFF.

GPIOC->ODR |= 1<<6; // Set the Pin PC6(Red). Turn LED ON.

GPIOC->ODR &= ~(1<<7); // Reset the Pin PC7(Blue). Turn LED OFF.

//GPIOC->ODR |= 1<<6; // Set the Pin PC6(Red). Turn LED ON.

//GPIOC->ODR &= ~(1<<7); // Reset the Pin PC7(Blue). Turn LED OFF.

/\* Initialize all configured peripherals \*/

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

uint32\_t debouncer = 0;

int PastX = 0;

int X = 0;

while (1)

{

PastX = X;

X = GPIOA->IDR & 0x1;

if((X==1) & (PastX==0)) {

GPIOC->ODR ^= (1<<6)|(1<<7);

}

HAL\_Delay(200);

debouncer = (debouncer << 1); // Always shift every loop iteration

if ( GPIOA->IDR & 1) { // If input signal is set/high

//HAL\_GPIO\_TogglePin(GPIOC, GPIO\_PIN\_8 | GPIO\_PIN\_9);

debouncer |= 0x01; // Set lowest bit of bit-vector

}

if (debouncer == 0xFFFFFFFF) {

// This code triggers repeatedly when button is steady high!

GPIOC->ODR = 0;

}

if (debouncer == 0x00000000) {

// This code triggers repeatedly when button is steady low!

GPIOC->ODR = (1<<6) | (1<<7);

//HAL\_GPIO\_TogglePin(GPIOC, GPIO\_PIN\_8 | GPIO\_PIN\_9);

}

if (debouncer == 0x7FFFFFFF) {

// This code triggers only once when transitioning to steady high!

//GPIOC->ODR |= 1<<6; // Set the Pin PC6(Red). Turn LED ON.

//GPIOC->ODR &= ~(1<<7); // Reset the Pin PC7(Blue). Turn LED OFF.

//HAL\_GPIO\_TogglePin(GPIOC, GPIO\_PIN\_8 | GPIO\_PIN\_9);

}

// When button is bouncing the bit-vector value is random since bits are set when the button is high and not when it bounces low.

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

/\*

//PART 1

//STEP 4

//HAL\_Delay(200); //Delay 200ms

// Toggle the output state of both PC8 and PC9

//HAL\_GPIO\_TogglePin(GPIOC, GPIO\_PIN\_8 | GPIO\_PIN\_9);

//GPIOC->BSRR |= (1<<8); // Set the Pin PC8. Turn LED ON.

//GPIOC->BSRR |= (1<<25); // Reset the Pin PC9. Turn LED OFF.

//GPIOC->ODR |= 1<<8; // Set the Pin PC8(Orange). Turn LED ON.

//GPIOC->ODR &= ~(1<<9); // Reset the Pin PC9(Green). Turn LED OFF.

GPIOC->ODR |= 1<<6; // Set the Pin PC6(Red). Turn LED ON.

GPIOC->ODR &= ~(1<<7); // Reset the Pin PC7(Blue). Turn LED OFF.

HAL\_Delay(200); // Delay 200ms

//GPIOC->BSRR |= (1<<24); // Reset the Pin PC8. Turn LED OFF.

//GPIOC->BSRR |= (1<<9); // Set the Pin PC9. Turn LED ON.

//GPIOC->ODR &= ~(1<<8); // Reset the Pin PC8(Orange). Turn LED OFF.

//GPIOC->ODR |= 1<<9; // Set the Pin PC9(Green). Turn LED ON.

GPIOC->ODR &= ~(1<<6); // Reset the Pin PC6(Red). Turn LED OFF.

GPIOC->ODR |= 1<<7; // Set the Pin PC7(Blue). Turn LED ON.

HAL\_Delay(200); // Delay 200ms

\*/

//PART 2

//STEP 4

//Monitor the button pin input state within the endless program loop.

//Use the IDR register.

/\*

//Reading PA0 bit

if( ( GPIOA->IDR & 0x01) == 0 ) {

//PORT A's 0th bit is 0

GPIOC->ODR |= 1<<6; // Set the Pin PC6(Red). Turn LED ON.

GPIOC->ODR &= ~(1<<7); // Reset the Pin PC7(Blue). Turn LED OFF.

}

else {

//PORT A's 0th bit is 1

GPIOC->ODR &= ~(1<<6); // Reset the Pin PC6(Red). Turn LED OFF.

GPIOC->ODR |= 1<<7; // Set the Pin PC7(Blue). Turn LED ON.

}

\*/

}

/\* USER CODE END 3 \*/

}