```c
#include "main.h"

UART_HandleTypeDef huart2;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
int main(void)
{

HAL_Init();

SystemClock_Config();

MX_GPIO_Init();
MX_USART2_UART_Init();
while (1)
{

HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 1);
HAL_Delay (600);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 0);
HAL_Delay (1200);
}

}

void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

__HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);

RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
RCC_OscInitStruct.PLL.PLLM = 16;
RCC_OscInitStruct.PLL.PLLN = 336;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
RCC_OscInitStruct.PLL.PLLQ = 2;
```

```
RCC_OscInitStruct.PLL.PLLR = 2;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
Error_Handler();
}

RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
{
Error_Handler();
}
}

static void MX_USART2_UART_Init(void)
{

huart2.Instance = USART2;
huart2.Init.BaudRate = 115200;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart2) != HAL_OK)
{
Error_Handler();
}

}
static void MX_GPIO_Init(void)
{
GPIO_InitTypeDef GPIO_InitStruct = {0};
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOH_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();
```

```
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);

GPIO_InitStruct.Pin = B1_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

GPIO_InitStruct.Pin = GPIO_PIN_5;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}

void Error_Handler(void)
{
__disable_irq();
while (1)
{
}
}

#ifdef USE_FULL_ASSERT
void assert_failed(uint8_t *file, uint32_t line)
{
}
#endif /* USE_FULL_ASSERT */
```

The code is based on the STM32 microcontroller series and the HAL library, and it configures system clocks, initializes GPIO and UART interfaces, and controls a GPIO pin to flash an LED in the main loop. Here is a detailed explanation of each part of the code:

Including Header Files and Defining Global Variables:

- '#include "main.h"' : Includes the main header file, which generally contains the necessary HAL library and other required definitions.
- 'UART_HandleTypeDef huart2;' : Defines a UART (Universal Asynchronous Receiver/Transmitter) handle for subsequent serial communication.

Function Declarations:

- 'void SystemClock_Config(void);' : Function for system clock configuration.
- 'static void MX_GPIO_Init(void);' : GPIO (General-Purpose Input/Output) initialization function.
- 'static void MX_USART2_UART_Init(void);' : Initialization function for UART2.

Main Function:

- Initializes the HAL library.
- Calls the clock configuration, GPIO, and UART initialization functions.
- Controls the 5th pin of GPIOA (usually connected to an LED) in the main loop to toggle between high and low states periodically, making the LED flash.

System Clock Configuration (SystemClock_Config):

- Sets up the internal high-speed clock (HSI).
- Configures and starts the PLL (Phase-Locked Loop) with HSI as the input source.
- Sets the system clock and bus clocks.

GPIO Initialization (MX_GPIO_Init):

- Enables the clock for the GPIO ports.
- Sets the 5th pin of GPIOA to output mode for controlling the LED.

USART2 UART Initialization (MX_USART2_UART_Init):

- Configures the parameters of USART2, including baud rate, word length, stop bits, etc., for serial communication.

Error Handling (Error_Handler):

- If a system error occurs, this function is called to disable all interrupts and enter an infinite loop.

This code is primarily used for basic hardware operations, demonstrating the fundamental usage of the STM32 microcontroller through simple GPIO control and UART communication.