

Pinout & Configuration **Clock Configuration** **Software Packs**

Categories **A-Z**

System Core

- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog >

Timers >

Connectivity >

CAN1

CAN2

FMP12C1

I2C1

I2C2

I2C3

QUADSPI

SDIO

SPI1

SPI2

SPI3

UART4

UART5

USART1

USART2

USART3

USART6

USB_OTG_FS

USB_OTG_HS

USART2 Mode and Configuration

Mode **Asynchronous**

Hardware Flow Control (RS232) **Disable**

Configuration

Reset Configuration

Parameter Settings

Configure the below parameters :

Basic Parameters

Baud Rate 115200 Bits/s

Word Length 8 Bits (including Parity)

Parity None

Stop Bits 1

Advanced Parameters

Data Direction Receive and Transmit

Over Sampling 16 Samples

Pinout & Configuration **Clock Configuration** **Software Packs**

Categories **A-Z**

System Core

- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog >

Timers >

Connectivity >

Multimedia >

Computing >

Middleware >

NVIC Mode and Configuration

Configuration

NVIC

Code generation

Priority Group **...** ☐ Sort by Preemption Priority and Sub Priority ☐ Sort by interrupt

Search **...** Show **available interrupts** ☒ Force DMA

NVIC Interrupt Table	Enabled	Preemption Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0
Memory management fault	<input checked="" type="checkbox"/>	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0
Debug monitor	<input checked="" type="checkbox"/>	0
Pendable request for system service	<input checked="" type="checkbox"/>	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0
Flash global interrupt	<input type="checkbox"/>	0
RCC global interrupt	<input type="checkbox"/>	0
USART2 global interrupt	<input type="checkbox"/>	0
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	1
FPU global interrupt	<input type="checkbox"/>	0

PA5 Configuration :

GPIO output level **High**

GPIO mode **Output Push Pull**

GPIO Pull-up/Pull-down **No pull-up and no pull-down**

Maximum output speed **Low**

User Label

PC13 Configuration :

GPIO mode **External Interrupt Mode with Falling edge trigger detection**

GPIO Pull-up/Pull-down **No pull-up and no pull-down**

User Label

次からは USER が入力する部分のみを示した

(1)

```
/* Private includes -----
-----*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */
```

(2)

```
/* Private user code -----
-----*/
/* USER CODE BEGIN 0 */
__IO uint8_t Pushed; //shimojo
```

//ARMは環境依存の変数型を無くすために普通のCとは違う名前で型が宣言だそうです

```
//uint8_t Pushed; //これでも動作した
```

```
int ic; //icounter
//ic=0;
/* USER CODE END 0 */
```

(3)

```
/* USER CODE BEGIN 1 */

    setbuf(stdout, NULL); //bufferを提供

    ic=0;
/* USER CODE END 1 */
```

(4)

```
/* USER CODE BEGIN 2 */
printf("Hello World!! %r%n");
//printf("Hello World again!! %d%r%n",ic);
//HAL_Delay(1000);
/* USER CODE END 2 */
```

(5)

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
```

```

/* USER CODE BEGIN 3 */
    Pushed = 0; //shimojo
    while (Pushed == 0)
        ; //ここでループして待っている

    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5); //blueBotton_Pushで実
行

    printf("Hello World in user code 3 ic= %d¥r¥n",ic);

} // end of while(1)ですね
/* USER CODE END 3 */

(6)
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) //blueBottonの割込み
ルーチン
{
    if (GPIO_Pin == GPIO_PIN_13)
    {
        Pushed = 1;
        ic=ic+1;
/*
        HAL_NVIC_DisableIRQ(EXTI15_10_IRQn); //shimojo
        EXTI15_10からの割込みをdisableにする
*/
    }
    printf("Hello World in interrupt_Handler ¥r¥n");
}
//
int _write(int file, char *ptr, int len)
{
    HAL_UART_Transmit(&huart2,(uint8_t *)ptr,len,10);
    return len;
}
/* USER CODE END 4 */

```

以上です

以下は全リスト

```

/* USER CODE BEGIN Header */
/* push_printfのプログラム。GPIO_Interrupt と printf_TeraTermの合成。
 * それにいろいろコメントなどを追加したもの。2022.05.03 shimojo
 */
/**
*****

* @file      : main.c
* @brief     : Main program body
*****

* @attention
*
* Copyright (c) 2022 STMicroelectronics.
* All rights reserved.
*
* This software is licensed under terms that can be found in the LICENSE file
* in the root directory of this software component.
* If no LICENSE file comes with this software, it is provided AS-IS.
*
*****

*/
/* USER CODE END Header */
/* Includes -----*/
#include "main.h"

/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */

/* Private typedef -----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define -----*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -----*/
/* USER CODE BEGIN PM */

```

```

/* USER CODE END PM */

/* Private variables -----*/
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----*/
/* USER CODE BEGIN 0 */
__IO uint8_t Pushed; //shimojo
//ARMは環境依存の変数型を無くすために普通のCとは違う名前が宣言だそうです
//uint8_t Pushed; //これでも動作した
int ic; //icounter
//ic=0;
/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */
        setbuf(stdout, NULL); //bufferを提供
        ic=0;
    /* 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 */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
/* USER CODE BEGIN 2 */
printf("Hello World!! %r\n");
//printf("Hello World again!! %d%r\n",ic);
//HAL_Delay(1000);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
        Pushed = 0; //shimojo
        while (Pushed == 0)
            ; //ここでループして待っている
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5); //blueBotton_Pushで実行
        printf("Hello World in user code 3 ic= %d%r\n",ic);
    }// end of while(1)ですね
    /* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)

```

```

{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    __HAL_RCC_PWR_CLK_ENABLE();
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);

    /** Initializes the RCC Oscillators according to the specified parameters
    * in the RCC_OscInitTypeDef structure.
    */
    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();
    }

    /** Initializes the CPU, AHB and APB buses clocks
    */
    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();
    }
}

```



```

/**
 * @brief USART2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART2_UART_Init(void)
{

    /* USER CODE BEGIN USART2_Init 0 */

    /* USER CODE END USART2_Init 0 */

    /* USER CODE BEGIN USART2_Init 1 */

    /* USER CODE END USART2_Init 1 */
    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();
    }
    /* USER CODE BEGIN USART2_Init 2 */

    /* USER CODE END USART2_Init 2 */

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};

```

```

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOH_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);

/*Configure GPIO pin : PC13 */
GPIO_InitStruct.Pin = GPIO_PIN_13;
GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pin : PA5 */
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);

/* EXTI interrupt init*/
HAL_NVIC_SetPriority(EXTI15_10_IRQn, 1, 0);
HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

}

/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) //blueBottonの割込みルーチン
{
    if (GPIO_Pin == GPIO_PIN_13)
    {
        Pushed = 1;
        ic=ic+1;
/*
        HAL_NVIC_DisableIRQ(EXTI15_10_IRQn);//shimojo
        EXIT15_10からの割込みをdisableにする
*/
    }
    //    printf("Hello World in interrupt_Handler ¥r¥n");
}
}

```

```
//
int _write(int file, char *ptr, int len)
{
    HAL_UART_Transmit(&huart2,(uint8_t *)ptr,len,10);
    return len;
}

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return state */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

#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 CODE BEGIN 6 */
    /* 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) */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

注意点 https://qiita.com/kotetsu_yama/items/c0ee9651d1ea4829f2c5

例の EXTI13 は、IRQType の定義ファイル内で EXTI[15:10]となっています。

DisableIRQSampleF446ZE/Drivers/CMSIS/Device/ST/STM32F4xx/Include/stm32f446xx.h

```
EXTI15_10_IRQn = 40, /*!< External Line[15:10] Interrupts */
```

これは EXTI15 から EXTI10 の 6 ラインを一つの割り込みにまとめて入力しているという意味で、この割り込みを無効にすると、EXTI13 だけではなく、EXTI10 や EXTI15 も無効になります。STM32 では回避する手段が無いので、外部回路を工夫して無効にしたい割り込みを EXTI0 から EXTI4 などの単独で無効にできるピンに割り当てる必要があります。