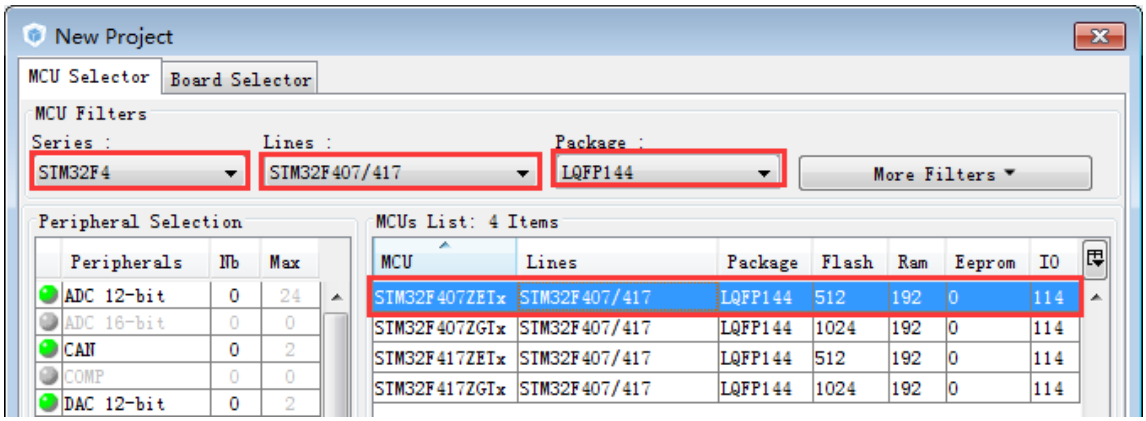
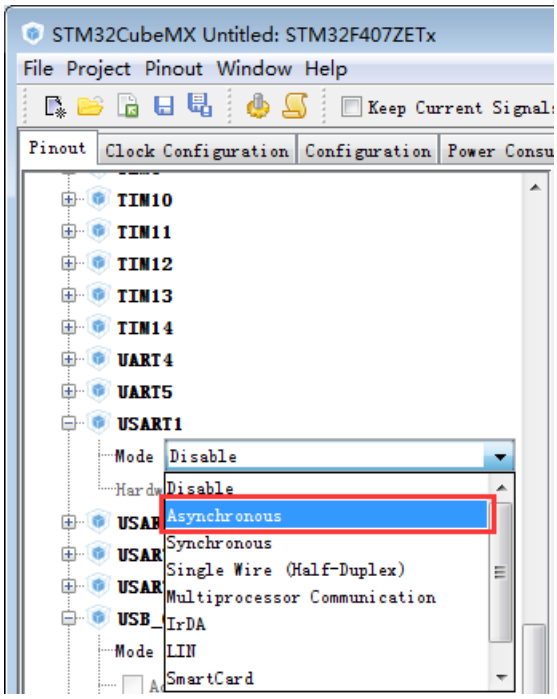
STM32Cube 学习之二：USART

假设已经安装好 STM32CubeMX 和 STM32CubeF4 支持包。

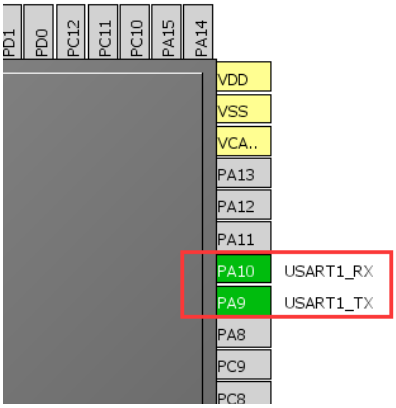
Step1.打开 STM32CubeMX，点击“New Project”，选择芯片型号，STM32F407ZETx。



Step2.在 Pinout 界面下配置 USART1 的模式为 Asynchronous，即异步串口模式。



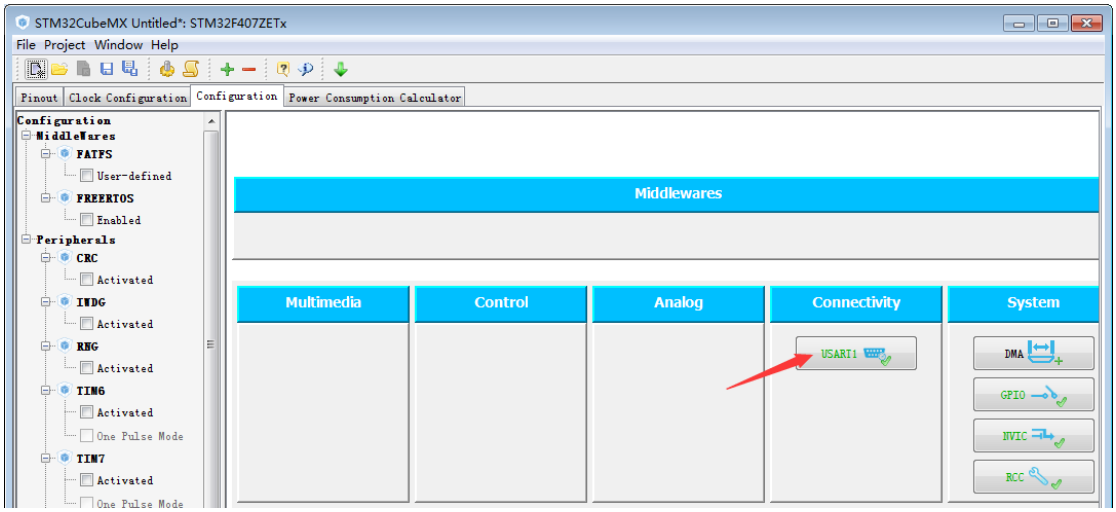
在右边的引脚分配图中可以看到，PA10 和 PA9 分别被配置为 USART1_RX 和 USART1_TX。



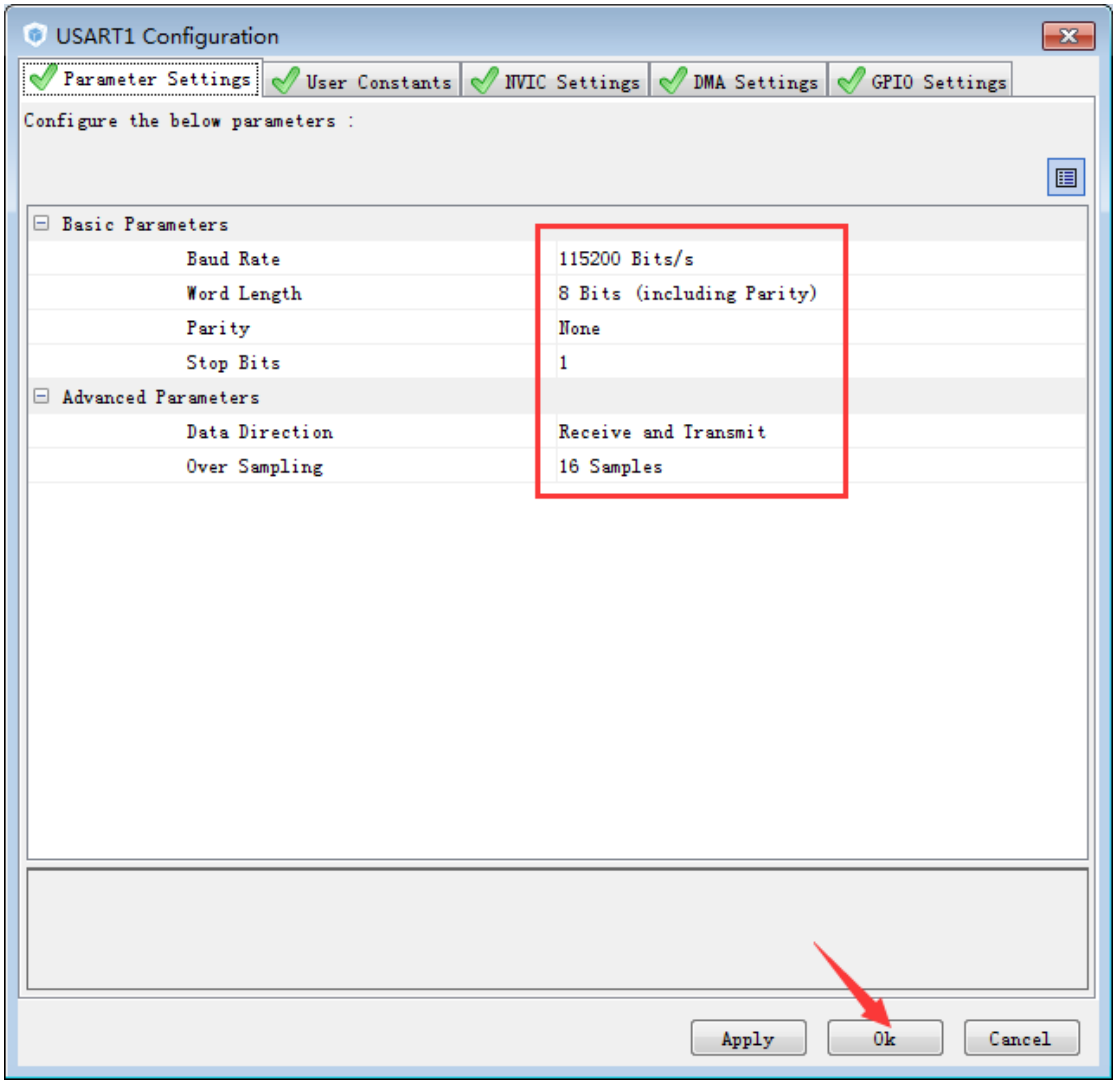
时钟树的配置不作任何修改，使用内部 16M 时钟源，内核时钟 16M。

Step3.配置串口参数

在 configuration 界面中点击 USART1 按钮，可以进入参数配置界面。

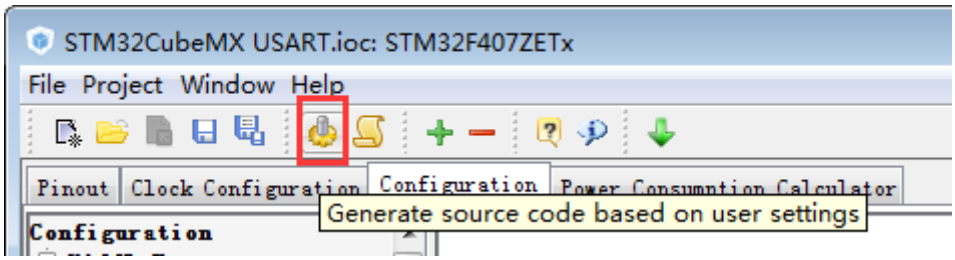


根据需要设置波特率和数据宽度等参数，在此使用默认的 115200，8，N，1。

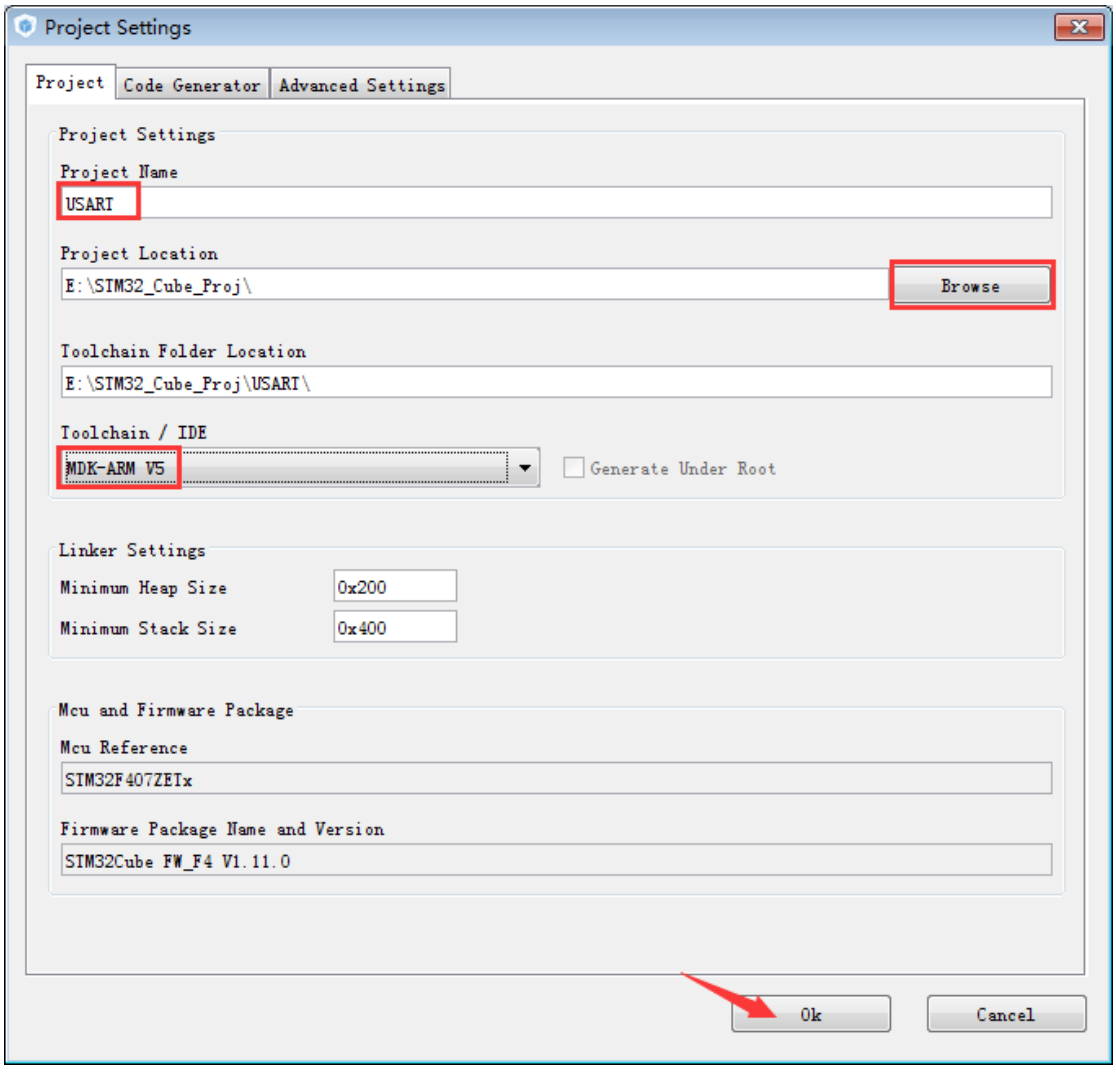


Step4.生成源代码。

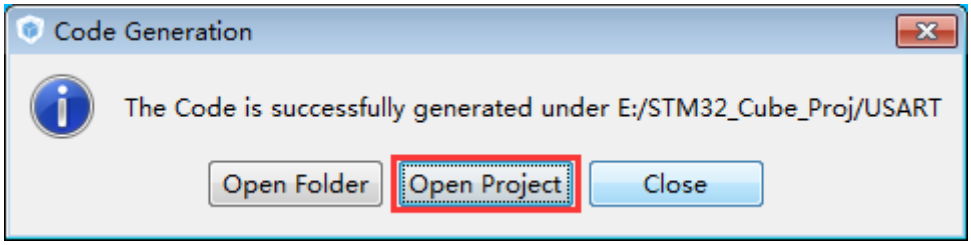
点击生成源代码按钮。



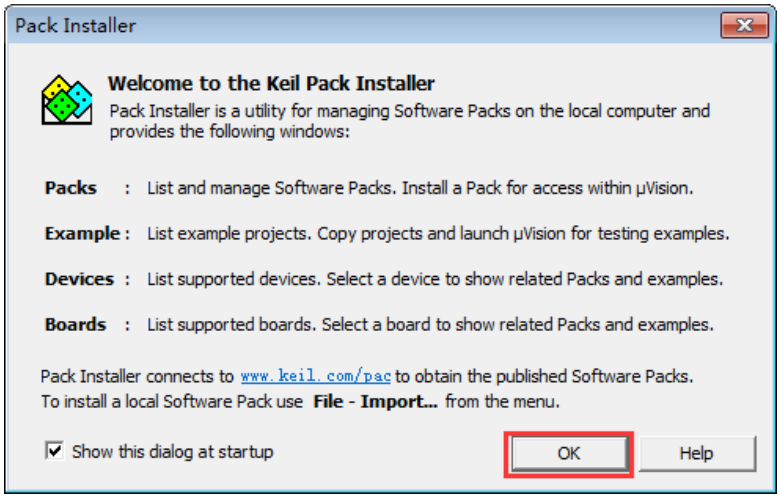
在设置界面中输入工程名，保存路径，工程 IDE 类型，点 OK 即可。



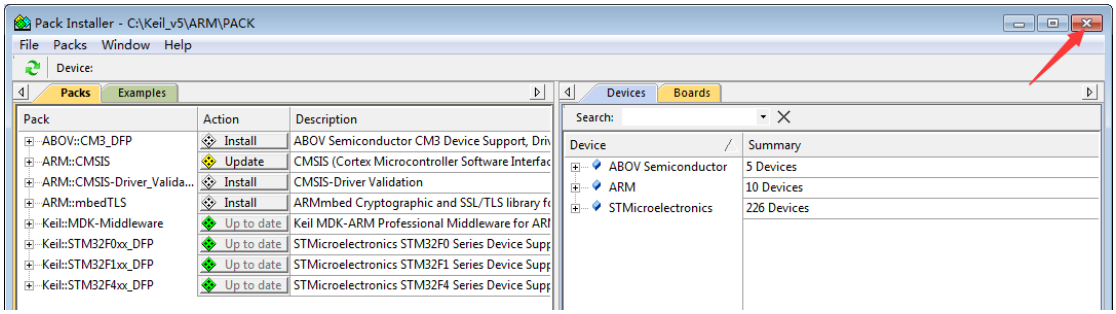
生成代码完成后可直接打开工程。



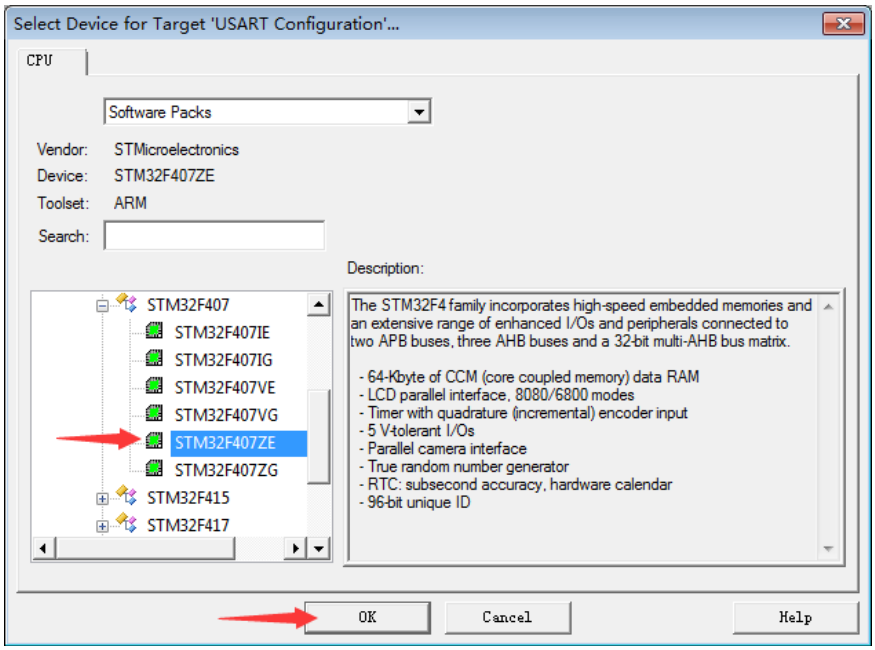
弹出如下对话框时，如果已经安装了 F4 的支持包，则点击 OK 关闭。如果没有安装，则点击界面中的 www.keil.com/...链接，找到芯片的支持包，然后安装。



关闭后面的界面。

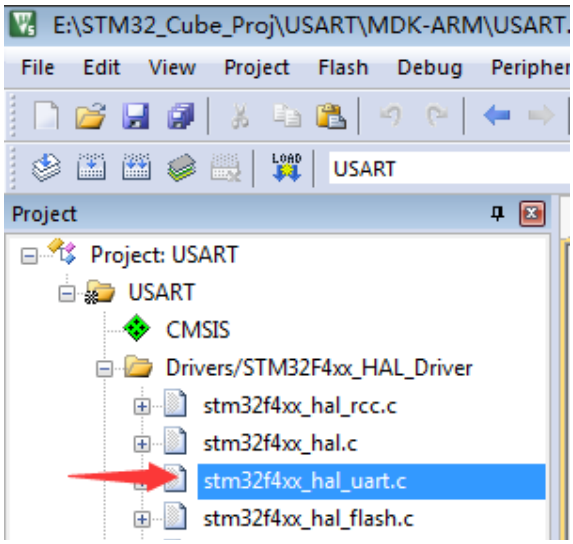


点击“是”，然后选择芯片型号。

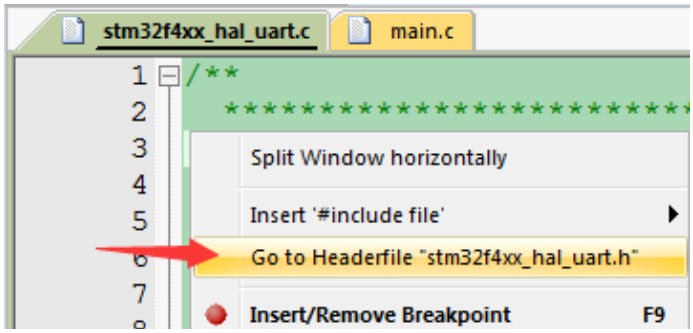


Step5.添加功能代码。

先编译一遍工程，然后打开 stm32f4xx_hal_uart.c 文件。



在 stm32f4xx_hal_uart.c 文件的任意地方点击右键，选择 Go to Headfile...可以打开相应的头文件。



找到串口发送函数。

```
646 /* IO operation functions *****************************************
647 HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_
648 HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t
649 HAL_StatusTypeDef HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uir
650 HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint
651 HAL_StatusTypeDef HAL_UART_Transmit_DMA(UART_HandleTypeDef *huart, ui
652 HAL_StatusTypeDef HAL_UART_Receive_DMA(UART_HandleTypeDef *huart, uir
653 HAL_StatusTypeDef HAL_UART_DMAPause(UART_HandleTypeDef *huart);
654 HAL_StatusTypeDef HAL_UART_DMAResume(UART_HandleTypeDef *huart);
655 HAL_StatusTypeDef HAL_UART_DMAStop(UART_HandleTypeDef *huart);
656 void HAL_UART_IRQHandler(UART_HandleTypeDef *huart);
657 void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart);
658 void HAL_UART_TxHalfCpltCallback(UART_HandleTypeDef *huart);
659 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);
660 void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef *huart);
661 void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart);
```

在 main 函数中添加如下代码。

```c
int main(void)
{

  /* USER CODE BEGIN 1 */
    uint8_t tx_buf[] = "USART Test\r\n";
  /* USER CODE END 1 */

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

  /* Reset of all peripherals, Initializes the
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();

  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

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

  /* USER CODE BEGIN 3 */
      HAL_UART_Transmit(&huart1, "Hello!\r\n", 8, 10);
      HAL_Delay(1000);
      HAL_UART_Transmit(&huart1, tx_buf, 12, 10);
      HAL_Delay(1000);
  }
  /* USER CODE END 3 */

}
```

编译程序，下载运行，单片机即可循环发送“Hello!”和“UART Test”。

函数解析：

HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)

第一个参数 huart：串口的句柄结构体指针，本例中使用&huart1，因为 CubeMX 配置生成的代码中已经定义了 huart1，并且在初始化函数 MX_USART1_UART_Init( )代码中已经将该变量和硬件的 USART1 进行了关联，所以操作的就是 USART1。

```c
/* USART1 init function */
void MX_USART1_UART_Init(void)
{

  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  HAL_UART_Init(&huart1);

}
```

第二个参数 pData：是要发送的数据的指针，可以像本例一样，直接写入字符串。但是要注意的是，该函数并不像 printf()一样有格式转换功能。
第三个参数 Size：本次要发送的字符数量。
第四个参数 Timeout：超时时间，单位是 ms。这是发送一个字符的超时时间，如果发送某个字符超过了所给的参数，则函数会返回 HAL_TIMEOUT。

该函数是有返回值的，返回值反映的是整个发送过程是否有错误。本例中没有考虑返回值。

另外，如何实现标准输入输出库的 printf()函数呢？打开 stm32cubef4.zip 解压后 STM32Cube_FW_F4_V1.11.0\Projects\STM324xG_EVAL\Examples\UART\UART_Printf 中的 MDK 工程。该工程给出了 printf()函数的实现方法。
其实很简单，就是实现一个串口输出一个字符的函数即可。该函数名已经在标准输入输出库头文件 stdio.h 中定义，原型为 int fputc(int ch, FILE *f)。

```
57  /* Private function prototypes -----------------------------------------------*/
58  #ifdef __GNUC__
59    /* With GCC/RAISONANCE, small printf (option LD Linker->Libraries->Small printf
60       set to 'Yes') calls __io_putchar() */
61    #define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
62  #else
63    #define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
64  #endif /* __GNUC__ */
65  static void SystemClock_Config(void);
66  static void Error_Handler(void);
```

```
121 /**
122   * @brief  Retargets the C library printf function to the USART.
123   * @param  None
124   * @retval None
125   */
126 PUTCHAR_PROTOTYPE
127 {
128   /* Place your implementation of fputc here */
129   /* e.g. write a character to the EVAL_COM1 and Loop until the end of transmission */
130   HAL_UART_Transmit(&UartHandle, (uint8_t *)&ch, 1, 0xFFFF);
131
132   return ch;
133 }
```

上两个图片看上去有点别扭，是因为 Cube 的例程考虑了兼容性，在 GCC 等一些编译器中，使用的标准输入输出库，字符输出的函数原型为 int __io_putchar(int ch)。

将 PUTCHAR_PROTOTYPE 展开，即可得到

```
121 /**
122   * @brief  Retargets the C library printf function to the USART.
123   * @param  None
124   * @retval None
125   */
126 int fputc(int ch, FILE *f)
127 {
128   /* Place your implementation of fputc here */
129   /* e.g. write a character to the EVAL_COM1 and Loop until the end of transmission */
130   HAL_UART_Transmit(&UartHandle, (uint8_t *)&ch, 1, 0xFFFF);
131
132   return ch;
133 }
```

这就是我们要实现 printf()需要做的。
当然还必须包含头文件#include "stdio.h"。

这样，就可以像例程中一样，调用 printf()函数了。

```
111
112   /* Output a message on Hyperterminal using printf function */
113   printf("\n\r UART Printf Example: retarget the C library printf function to the UART\n\r");
114
115   /* Infinite loop */
116   while (1)
117   {
118   }
119 }
```

更进一步的说，由上述 printf()的实现方法可知，如果将 fputc()函数的功能换成将一个字符输出到 LCD 上，则可以方便地使用 printf()来输出字符串到 LCD 上了。但要注意的是要在调用 printf()前设置显示的起始位置，并确保输出一个字符串之后显示位置自增相应的偏移量。

S.D.Lu 于 深圳
2016 年 8 月