static *void* MX\_SPI2\_Init(*void*)

{

    /\* USER CODE BEGIN SPI2\_Init 0 \*/

    /\* USER CODE END SPI2\_Init 0 \*/

    /\* USER CODE BEGIN SPI2\_Init 1 \*/

    /\* USER CODE END SPI2\_Init 1 \*/

    /\* SPI2 parameter configuration\*/

    hspi2.Instance = SPI2;

    hspi2.Init.Mode = SPI\_MODE\_MASTER;

    hspi2.Init.Direction = SPI\_DIRECTION\_1LINE;

    hspi2.Init.DataSize = SPI\_DATASIZE\_4BIT;

    hspi2.Init.CLKPolarity = SPI\_POLARITY\_LOW;

    hspi2.Init.CLKPhase = SPI\_PHASE\_1EDGE;

    hspi2.Init.NSS = SPI\_NSS\_SOFT;

    hspi2.Init.BaudRatePrescaler = SPI\_BAUDRATEPRESCALER\_2;

    hspi2.Init.FirstBit = SPI\_FIRSTBIT\_MSB;

    hspi2.Init.TIMode = SPI\_TIMODE\_DISABLE;

    hspi2.Init.CRCCalculation = SPI\_CRCCALCULATION\_DISABLE;

    hspi2.Init.CRCPolynomial = 7;

    hspi2.Init.CRCLength = SPI\_CRC\_LENGTH\_DATASIZE;

    hspi2.Init.NSSPMode = SPI\_NSS\_PULSE\_ENABLE;

    if (HAL\_SPI\_Init(&hspi2) != HAL\_OK)

    {

        Error\_Handler();

    }

    /\* USER CODE BEGIN SPI2\_Init 2 \*/

    /\* USER CODE END SPI2\_Init 2 \*/

}

*void* HAL\_SPI\_MspInit(SPI\_HandleTypeDef\* *hspi*)

{

  GPIO\_InitTypeDef GPIO\_InitStruct = {0};

  if(hspi->Instance==SPI2)

  {

  /\* USER CODE BEGIN SPI2\_MspInit 0 \*/

  /\* USER CODE END SPI2\_MspInit 0 \*/

    /\* Peripheral clock enable \*/

    \_\_HAL\_RCC\_SPI2\_CLK\_ENABLE();

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

    /\*\*SPI2 GPIO Configuration

    PB15     ------> SPI2\_MOSI

    PB13     ------> SPI2\_SCK

    \*/

    GPIO\_InitStruct.Pin = GPIO\_PIN\_15|GPIO\_PIN\_13;

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

    GPIO\_InitStruct.Pull = GPIO\_NOPULL;

    GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

    GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI2;

    HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

  /\* USER CODE BEGIN SPI2\_MspInit 1 \*/

  /\* USER CODE END SPI2\_MspInit 1 \*/

  }

}

*void* MX\_UART5\_Init(*void*)

{

    /\* USER CODE BEGIN UART5\_Init 0 \*/

    /\* USER CODE END UART5\_Init 0 \*/

    /\* USER CODE BEGIN UART5\_Init 1 \*/

    /\* USER CODE END UART5\_Init 1 \*/

    huart5.Instance = UART5;

    huart5.Init.BaudRate = 115200;

    huart5.Init.WordLength = UART\_WORDLENGTH\_8B;

    huart5.Init.StopBits = UART\_STOPBITS\_1;

    huart5.Init.Parity = UART\_PARITY\_NONE;

    huart5.Init.Mode = UART\_MODE\_TX\_RX;

    huart5.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

    huart5.Init.OverSampling = UART\_OVERSAMPLING\_16;

    huart5.Init.OneBitSampling = UART\_ONE\_BIT\_SAMPLE\_DISABLE;

    huart5.AdvancedInit.AdvFeatureInit = UART\_ADVFEATURE\_NO\_INIT;

    if (HAL\_UART\_Init(&huart5) != HAL\_OK)

    {

        Error\_Handler();

    }

    /\* USER CODE BEGIN UART5\_Init 2 \*/

    /\* USER CODE END UART5\_Init 2 \*/

}

static *void* MX\_I2C1\_Init(*void*)

{

  /\* USER CODE BEGIN I2C1\_Init 0 \*/

  /\* USER CODE END I2C1\_Init 0 \*/

  /\* USER CODE BEGIN I2C1\_Init 1 \*/

  /\* USER CODE END I2C1\_Init 1 \*/

  hi2c1.Instance = I2C1;

  hi2c1.Init.ClockSpeed = 100000;

  hi2c1.Init.DutyCycle = I2C\_DUTYCYCLE\_2;

  hi2c1.Init.OwnAddress1 = 0;

  hi2c1.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

  hi2c1.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

  hi2c1.Init.OwnAddress2 = 0;

  hi2c1.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

  hi2c1.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

  if (HAL\_I2C\_Init(&hi2c1) != HAL\_OK)

  {

    Error\_Handler();

  }

  /\*\* Configure Analogue filter

  \*/

  if (HAL\_I2CEx\_ConfigAnalogFilter(&hi2c1, I2C\_ANALOGFILTER\_ENABLE) != HAL\_OK)

  {

    Error\_Handler();

  }

  /\*\* Configure Digital filter

  \*/

  if (HAL\_I2CEx\_ConfigDigitalFilter(&hi2c1, 0) != HAL\_OK)

  {

    Error\_Handler();

  }

  /\* USER CODE BEGIN I2C1\_Init 2 \*/

  /\* USER CODE END I2C1\_Init 2 \*/

}

*void* MX\_USART3\_UART\_Init(*void*)

{

  /\* USER CODE BEGIN USART3\_Init 0 \*/

  /\* USER CODE END USART3\_Init 0 \*/

  /\* USER CODE BEGIN USART3\_Init 1 \*/

  /\* USER CODE END USART3\_Init 1 \*/

  huart3.Instance = USART3;

  huart3.Init.BaudRate = 115200;

  huart3.Init.WordLength = UART\_WORDLENGTH\_8B;

  huart3.Init.StopBits = UART\_STOPBITS\_1;

  huart3.Init.Parity = UART\_PARITY\_NONE;

  huart3.Init.Mode = UART\_MODE\_TX\_RX;

  huart3.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

  huart3.Init.OverSampling = UART\_OVERSAMPLING\_16;

  if (HAL\_UART\_Init(&huart3) != HAL\_OK)

  {

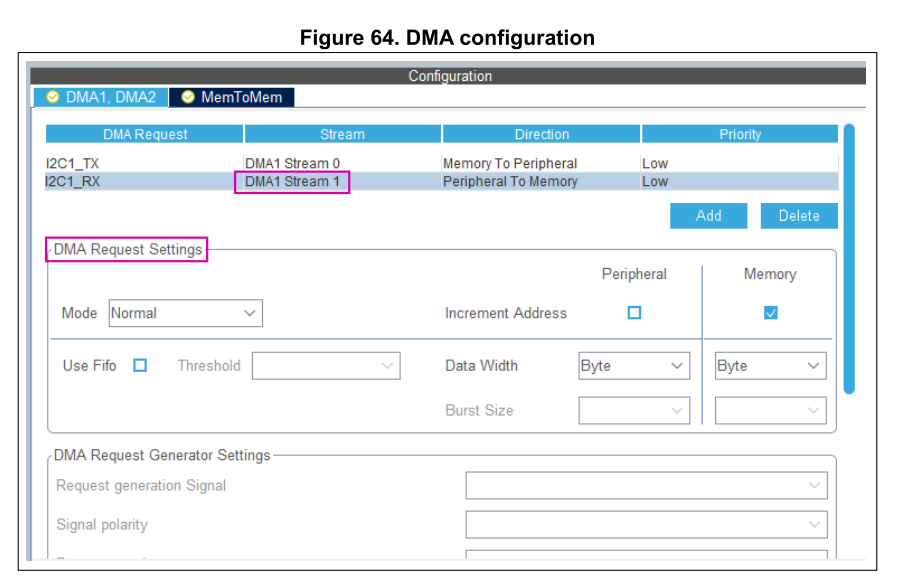
    Error\_Handler();

  }

  /\* USER CODE BEGIN USART3\_Init 2 \*/

  /\* USER CODE END USART3\_Init 2 \*/

}



stm32f4xx\_hal\_conf.h文件：该文件定义了启用的HAL模块并将一些参数（例如外部高速振荡器频率）设置为预定义的默认值或根据用户配置（时钟树）。

MSP = MCU Support package

stm32f4xx\_hal\_msp.c（MSP = MCU支持软件包）：该文件定义了所有初始化功能，以根据用户配置（引脚分配，时钟使能，DMA使用和中断）来配置外围设备实例

main.c

通过调用HAL\_init（）函数将MCU重置为已知状态，该函数将重置所有外设，初始化闪存接口和SysTick。

配置和初始化系统时钟。

配置和初始化外设不使用的GPIO。

为每个配置的外围设备定义和调用外围设备初始化函数，该函数定义一个句柄结构，该句柄结构将传递给相应的外围设备HAL初始化函数，该函数又将调用外围设备HAL MSP初始化函数。请注意，当使用LwIP（分别为USB）中间件时，底层以太网（分别为USB外设）的初始化C代码从main.c移到LwIP（分别为USB）初始化C代码本身。

