STM32L433 串口通讯例程详解

一、串口 DMA 接收

1、 stm32l4xx_hal_uart.h 第 1635 行后插入 函数如声明，如下所示：

```
1633 void HAL_UART_IRQHandler(UART_HandleTypeDef *huart);
1634 void HAL_UART_TxHalfCpltCallback(UART_HandleTypeDef *huart);
1635 void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart);
1636 void HAL_UART_RxIdleCallback(UART_HandleTypeDef *huart);
1637 void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef *huart);
1638 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);
1639 void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart);
1640 void HAL_UART_AbortCpltCallback(UART_HandleTypeDef *huart);
1641 void HAL_UART_AbortTransmitCpltCallback(UART_HandleTypeDef *huart);
1642 void HAL_UART_AbortReceiveCpltCallback(UART_HandleTypeDef *huart);
```

void HAL_UART_RxIdleCallback(UART_HandleTypeDef *huart);

2、 stm32l4xx_hal_uart.c 第 1537 行后插入 如下代码：

stm32l4xx_hal_uart.c

```
1531         /* Restore huart->gState to ready */
1532         huart->gState = HAL_UART_STATE_READY;
1533
1534         return HAL_ERROR;
1535       }
1536     }
1537     __HAL_UNLOCK(huart);
1538
1539     /* Enable the UART IDLEIE Interrupt */
1540     SET_BIT(huart->Instance->CR1, USART_CR1_IDLEIE);
1541
1542     /* Enable the UART Parity Error Interrupt */
1543     SET_BIT(huart->Instance->CR1, USART_CR1_PEIE);
1544
1545     /* Enable the UART Error Interrupt: (Frame error, noise error, overrun error) */
1546     SET_BIT(huart->Instance->CR3, USART_CR3_EIE);
1547
1548     /* Enable the DMA transfer for the receiver request by setting the DMAR bit
1549     in the UART CR3 register */
1550     SET_BIT(huart->Instance->CR3, USART_CR3_DMAR);
1551
1552     return HAL_OK;
1553   }
1554   else
1555   {
1556     return HAL_BUSY;
1557   }
1558 }
```

如上所示，插入的代码是第 1540 行，代码功能是打开空闲中断。

修改的函数名称是 HAL_UART_Receive_DMA 。

3、 stm32l4xx_hal_uart.c 的 HAL_UART_IRQHandler 函数末尾，增加如下代码：

```
2528
2529   /* UART Idle interrupt occurred   -----------------------------------------*/
2530   if (((isrflags & UART_FLAG_IDLE) != RESET) && ((crlits & USART_CR1_IDLEIE) != RESET))
2531   {
2532     __HAL_UART_CLEAR_IDLEFLAG(huart);
2533     __HAL_UART_DISABLE_IT(huart, UART_IT_IDLE);
2534     HAL_UART_RxIdleCallback(huart);
2535     return;
2536   }
2537 }
2538
2539 /**
2540   * @brief  Rx Idle callbacks.
2541   * @param  huart: Pointer to a UART_HandleTypeDef structure that contains
2542   *                the configuration information for the specified UART module.
2543   * @retval None
2544   */
2545 __weak void HAL_UART_RxIdleCallback(UART_HandleTypeDef *huart)
2546 {
2547   /* Prevent unused argument(s) compilation warning */
2548   UNUSED(huart);
2549   /* NOTE: This function should not be modified, when the callback is needed,
2550            the HAL_UART_RxIdleCallback can be implemented in the user file
2551    */
2552 }
```

HAL_UART_IRQHandler 函数末尾添加的代码是串口空闲的判断和回调函数的调用，另外增加了哑函数 HAL_UART_RxIdleCallback 的弱定义。__weak 可以防止缺少函数时的编译错误。

4、 bsp_uart.c 或其它 c 文件（如 main.c）中，增加串口接收空闲回调函数。详见例程。这是重点内容 1，在串口接收线上空闲时，触发该回调函数，将 DMA 接收缓冲里的内容转移到一个较大的环形缓冲里。

```c
/**
 * @brief  Rx Idle callbacks.
 * @param  huart: Pointer to a UART_HandleTypeDef structure that contains
 *                the configuration information for the specified UART module.
 * @retval None
 */
void HAL_UART_RxIdleCallback(UART_HandleTypeDef* huart)
{
  uint16_t count, i, j;

  if (huart->Instance == USART1) {
    if (huart->hdmarx->XferCpltCallback != NULL) {
      if (huart->hdmarx->Instance->CNDTR != 0) {
        count = huart->RxXferSize - huart->hdmarx->Instance->CNDTR;
        HAL_UART_DMAStop(huart);
      } else {
        count = huart->RxXferSize;
      }

      UART1_Data_In_Count += count;
      for (j = UART1_Data_In_PSTR, i = 0; (i < count) && (j < UART1_RXbuf_SIZE); ) {
        UART1_RXbuf[j++] = UART1_DMA_RXbuf[i++];
      }
      UART1_Data_In_PSTR += i;
      for (j = 0; i < count; ) {
        UART1_RXbuf[j++] = UART1_DMA_RXbuf[i++];
        UART1_Data_In_PSTR = j;
      }

      memset((uint8_t*)UART1_DMA_RXbuf, 0, UART1_DMA_RXbuf_SIZE);

      HAL_UART_Receive_DMA(huart, (uint8_t*)&UART1_DMA_RXbuf, UART1_DMA_RXbuf_SIZE);
    }
  }
}

```

5、 bsp_uart.c 或其它 c 文件（如 main.c）中，增加串口接收完成回调函数。详见例程。这是重点内容 2，在串口 DMA 接收缓冲填满时，触发该回调函数，将 DMA 接收缓冲里的内容转移到一个较大的环形缓冲里。

```c
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
#define UART1_DMA_RXbuf_SIZE  16
#define UART1_RXbuf_SIZE      128
__IO uint8_t UART1_DMA_RXbuf[UART1_DMA_RXbuf_SIZE];   /* UART1 DMA Receive buffer */
__IO uint8_t UART1_RXbuf[UART1_RXbuf_SIZE];           /* UART1 Receive buffer */
__IO uint16_t UART1_Data_In_PSTR = 0;
__IO uint32_t UART1_Data_In_Count = 0;


uint8_t UART1_TXbuf[512];
uint8_t UART1_Printf_DMA_TXbuf[1024];

```

其中 UART1_DMA_RXbuf 是 DMA 接收缓冲，接收数据时，不占用 CPU 时间资源。其中 UART1_RXbuf 是环形接收缓冲。

```c
/**
  * @brief  Rx Transfer completed callback.
  * @param  huart UART handle.
  * @retval None
  */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  /* Prevent unused argument(s) compilation warning */
  uint16_t count, i, j;

  if (huart->hdmarx->Instance->CNDTR == 0) {
    count = huart->RxXferSize;
  } else {
    count = 0;
  }
  UART1_Data_In_Count += count;
  for (j = UART1_Data_In_PSTR, i = 0; (i < count) && (j < UART1_RXbuf_SIZE); ) {
    UART1_RXbuf[j++] = UART1_DMA_RXbuf[i++];
  }
  UART1_Data_In_PSTR += i;
  for (j = 0; i < count; ) {
    UART1_RXbuf[j++] = UART1_DMA_RXbuf[i++];
    UART1_Data_In_PSTR = j;
  }
  memset((uint8_t*)UART1_DMA_RXbuf, 0, UART1_DMA_RXbuf_SIZE);

  HAL_UART_Receive_DMA(huart, (uint8_t*)&UART1_DMA_RXbuf, UART1_DMA_RXbuf_SIZE);

  /* NOTE : This function should not be modified, when the callback is needed,
            the HAL_UART_RxCpltCallback can be implemented in the user file.
   */
}
```

测试串口接收驱动时，如果 PC 串口工具发送 17 个字符，则会调用以下函数：

（1） 调用 HAL_UART_RxHalfCpltCallback 函数，DMA 接收到一半产生此中断。此中断我们可以不用处理。

（2） 调用 HAL_UART_RxCpltCallback 函数，DMA 接收设定的 16 个字符，产生此中断。该函数我们已经重写，将字符取走，重新启用 DMA 接收，再收到 1 个字符，产生空闲中断，然后调用 HAL_UART_RxIdleCallback 函数，取走最后一个字符。如果串口 DMA 接收启动后，PC 串口工具发送 16 个字符，调用 HAL_UART_RxCpltCallback 函数后，没有继续收到字符，但空闲发生，仍然会调用串口空闲中断。

二、串口发送函数的实现（__printf__）

1、 增加一些必要的头文件，如下图所示，包含的是 C 语言基础头文件：

```c
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "dma.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

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

```

2、 增加__printf__函数的实体，DMA 发送方式，发送数据时，不占用 CPU 时间资源。

```c
/**
 * @brief  __printf__
 * @param  UART_HandleTypeDef* huart
 *         format
 *         ...       param list
 *
 * format:
 *
 * __printf__ (&huart1, "\r\n %s\r\n",s);
 * __printf__ (&huart1, "\r\n %d %d %d\r\n",i,j,k);
 * __printf__ (&huart1, "\r\n 0x%x 0x%x 0x%x\r\n", i, j, k);
 *
 *
 */
uint16_t __printf__(UART_HandleTypeDef* huart, const char* format, ...)
{
  uint16_t real_len = 0;

  va_list arg;
  va_start(arg, format);
  real_len = vsprintf((char*)UART1_Printf_DMA_TXbuf, format, arg);
  va_end(arg);

  if (real_len != 0) {
    HAL_UART_Transmit_DMA(huart, UART1_Printf_DMA_TXbuf, real_len);
  }

  return real_len;
}
```

如果只用串口 1 输出调试信息，可以依照上述例程，重新实现一个__printf__函数。
对应的 6-1 视频课、源代码包下载地址：http://www.mcu.so/index.htm
6-1 视频课已上传到 B 站。
重复发消息，群主不会回复，工作忙，只是业余做下视频。
基础问题可在大虾论坛发帖询问：http://www.daxia.com

编制日期 2021/3/15
修订日期 2021/3/19

附录：

- STM32 商业开发环境

1. IAR Embedded Workbench IDE
   推荐版本：EWARM-CD-8402-22891.exe
   参考：通用 Cortex-M3/M4/M7 内核 ARM 芯片软件开发，
   扩展支持 S32K 汽车电子、基于模型设计软件开发，全局变量实时观测。
2. MDK-ARM
   推荐版本：MDK533.EXE
   参考：通用 Cortex-M3/M4/M7 内核 ARM 芯片软件开发，全局变量实时观测。
3. Visual Studio + VisualGDB Ultimate Edition
   推荐版本：
   Microsoft Visual Studio Professional 2019
   VisualGDB 5.5 rc1.msi
   参考：通用 Cortex-M3/M4/M7 内核 ARM 芯片软件开发，全局变量实时观测。

➢ STM32 免费开发环境

1. STM32CubeIDE

   推荐版本：

   st-stm32cubeide_1.5.1_9029_20201210_1234_x86_64.exe

   参考：STM32 芯片软件开发，全局变量实时观测。

   http://wjandcf.xicp.vip:88/soft/st/st-stm32cubeide_1.5.1_9029_20201210_1234_x86_64.exe

2. Atollic_TrueSTUDIO_for_STM32

   推荐版本：

   Atollic_TrueSTUDIO_for_STM32_windows_x86_v9.3.0_20190212-0734.exe

   参考：STM32 芯片软件开发，全局变量实时观测。

   该软件被 ST 收购之后，改名 STM32CubeIDE。

   http://wjandcf.xicp.vip:88/soft/st/Atollic_TrueSTUDIO_for_STM32_windows_x86_v9.3.0_20190212-0734.exe

3. System Workbench for STM32

   推荐版本：

   install_sw4stm32_win_64bits-v2.9.exe

   参考：STM32 芯片软件开发，支持 Cortex-A 架构软件开发。

   http://wjandcf.xicp.vip:88/soft/st/install_sw4stm32_win_64bits-v2.9.exe

4. Visual Studio Code + IoT Link

   推荐版本：

   VSCodeUserSetup-x64-1.49.3.exe

   IoT Link 1.3.0

   参考：支持通用 Cortex-M3/M4/M7 内核 ARM 芯片软件开发。

   http://wjandcf.xicp.vip:88/soft/Microsoft/VSCodeUserSetup-x64-1.49.3.exe

5. Eclipse IDE for C/C++ Developers + System Workbench Plug-in

   推荐版本：eclipse-cpp-2019-12-R-win32-x86_64.zip

   参考：采用 Eclipse IDE for C/C++ Developers + System Workbench Plug-in 安装方式，支持通用 Cortex-M3/M4/M7 内核 ARM 芯片、RISC-V 架构芯片软件开发。

   警告：**完全兼容 System Workbench Plug-in 的最新版本是 Eclipse 2019-12。**

5.1 Eclipse IDE for C/C++ Developers 及其工具插件下载：

http://wjandcf.xicp.vip:88/Eclipse-updates/eclipse-cpp-2019-12-R-win32-x86_64.zip

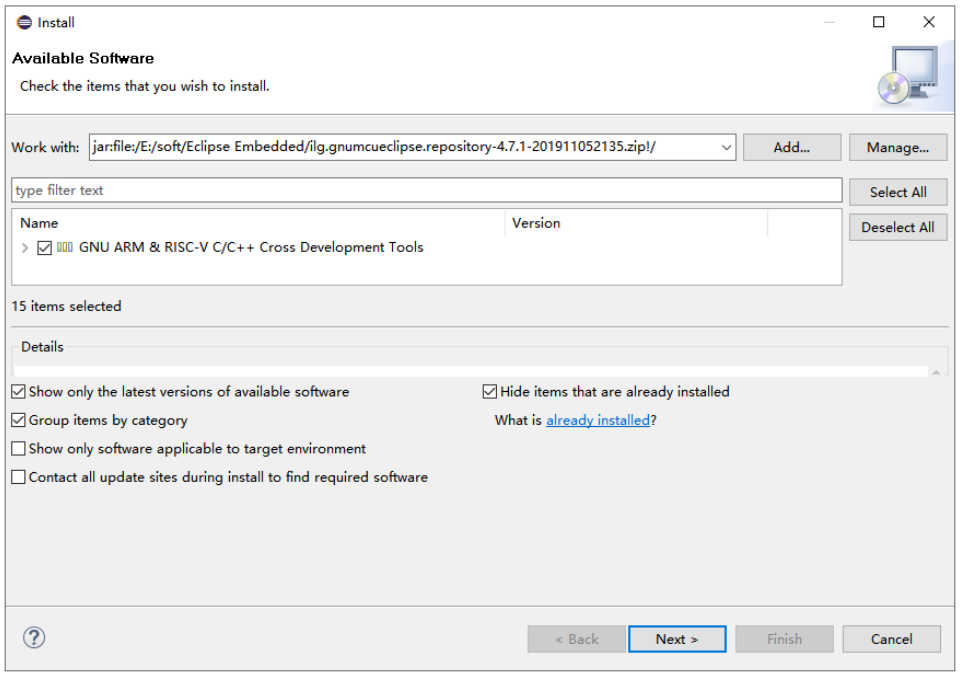
http://wjandcf.xicp.vip:88/Eclipse-updates/ilg.gnumcueclipse.repository-4.7.1-201911052135.zip

http://wjandcf.xicp.vip:88/Eclipse-updates/xPacks.7z

http://wjandcf.xicp.vip:88/Eclipse-updates/Readme.txt

5.2 GNU MCU Eclipse Plug-ins 的安装

在网站下载离线包，不联网，离线安装 ilg.gnumcueclipse.repository-4.7.1-201911052135.zip。



或者下载已整合 GNU MCU Eclipse Plug-ins 插件的版本，解压后即可运行。

http://wjandcf.xicp.vip:88/Eclipse-updates/20200115-1949-gnumcueclipse-4.7.1-2019-12-R-win32.win32.x86_64.zip

5.3 EmbSysRegView Plug-ins 的安装

在线安装网址:

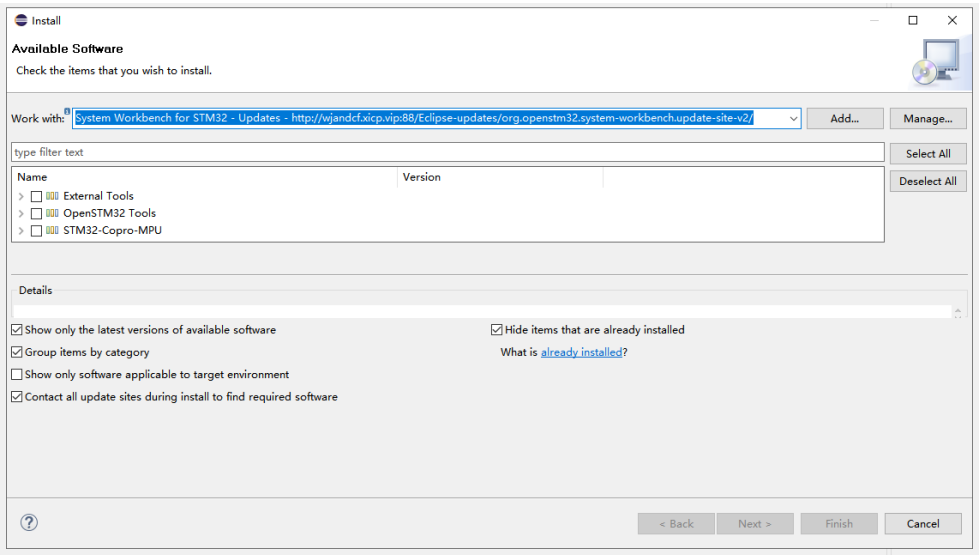
http://wjandcf.xicp.vip:88/Eclipse-updates/ilg.gnumcueclipse.repository-4.7.1-201911052135

5.4 System Workbench Plug-ins 的在线安装

http://ac6-tools.com/Eclipse-updates/org.openstm32.system-workbench.update-site-v2

http://wjandcf.xicp.vip:88/Eclipse-updates/org.openstm32.system-workbench.update-site-v2

警告：国外站点速度有限，推荐使用国内站点。



编制日期 2021/3/19

修改日期 2021/3/20