第6章 串行通信接口及中断系统

6.1 异步串行通信的通用基础知识

6.2 基于构件的串行通信编程方法

6.3 中断机制及中断编程步骤

6.4 RISC-V汇编工程框架解析

实验三：基于串行通信构件的汇编程序设计

本章阐述串行通信构件化编程。主要内容有异步串行通信及中断两个模块。异步串行通信模块介绍异步串行通信的通用基础知识，使读者理解串行通信的基本概念及编程模型；并阐述基于构件的串行通信编程方法，这是一般应用级编程的基本模式。中断模块介绍RISC-V中断机制及CH573中断编程步骤。最后介绍串口通信及中断实验，读者通过实验熟悉MCU的异步串行通信UART的工作原理，掌握UART的通信编程方法、串口组帧编程方法以及PC机的C#串口通信编程方法。

6.1 异步串行通信的通用基础知识

串行通信接口，简称“串口”、UART或SCI。在USB未普及之前，串口是PC机必备的通信接口之一。微型计算机中的串口通信，在硬件上，一般只需要三根线，分别称为发送线（TxD）、接收线（RxD）和地线（GND）；通信方式上，属于单字节通信。实现串口功能的模块有的称为通用异步收发器（Universal Asynchronous Receiver-Transmitters，UART），有的称为串行通信接口（Serial Communication Interface，SCI）。

6.1.1 串行通信的基本概念

“位”（bit）是单个二进制数字的简称，是可以拥有两种状态的最小二进制值，分别用“0”和“1”表示。在计算机中，通常用8位二进制表示一个信息单位，称为一个“字节”（byte）。串行通信的特点是数据以字节为单位，按位的顺序（例如最高位优先）从一条传输线上发送出去。这里涉及4个问题：第一，每个字节之间是如何区分开的？第二，发送一位的持续时间是多少？第三，怎样知道传输是正确的？第四，可以传输多远？

1. 异步串行通信的格式

异步串行通信是**标准不归零传号/空号**数据格式，采用**不归零码**，即用负电平表示一种二进制值，正电平表示另一种二进制值，电压均无需回到零，这是早期使用**RS232电平**对串行通信的描述。**异步**的含义是每个字节重新开始。这里举例以**逻辑方式**描述串行通信数据格式：**1位起始位+8位数据位+1位停止位。逻辑方式对应于TTL电平，逻辑1对应高电平，逻辑0对应于低电平。**

可以具体给出发送数据十六进制**56**，即二进制01010110，格式为：

为什么称为异步通信？

这种格式的空闲状态为“1”，发送器通过发送一个“0”表示一个字节传输的开始，随后是数据位，随后为停止位，表示一个字节传送结束。若继续发送下一字节，则重新发送开始位，开始一个新的字节传送，这就是“**异步**”之含义。若不发送新的字节，则维持“1”的状态，使发送数据线处于空闲。从开始位到停止位结束的时间间隔称为一字节帧（Byte Frame）。所以，也称这种格式为**字节帧格式**。每发送一个字节，都要发送“开始位”与“停止位”，这是影响异步串行通信传送速度的因素之一。

2. 串行通信的波特率

串口通信的速度用波特率来表示，它定义为每秒传输的二进制位数，1波特=1位/s，单位bps（位/s）。只有通信双方的波特率一样时才可以进行正常通讯。通常使用的波特率有9600、19200、38400、57600及115200等。

已知波特率，计算出发送N个字节的时间，注意发送每个字节至少是10位（至少一个开始位，一个停止位）

3. 奇偶校验 （了解）

在异步串行通信中，可以增加一个位（奇偶校验位），供错误检测使用。

4. 串行通信传输方式术语（了解）

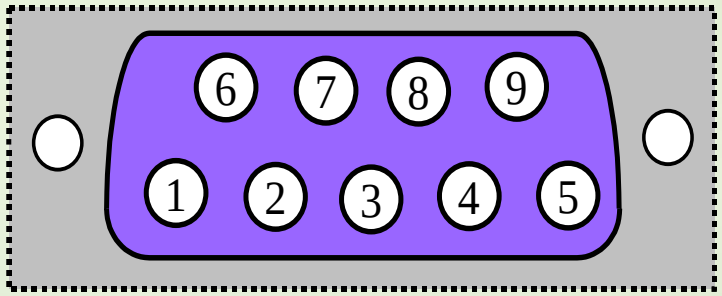
全双工：数据传送方向为双向，可以同时接收与发送数据
半双工：数据传送方向为双向，不可同时接收与发送数据
单工：数据传送方向为单向

6.1.2 RS232和RS485总线标准

1. RS232

RS232采用负逻辑，-15V～-3V为逻辑“1”，+3V～+15V为逻辑“0”。RS232最大的传输距离是30m，通信速率一般低于20kbps。

早期的标准串行通信接口是25芯，后来改为9芯，目前部分PC机带有9芯RS232串口，其引出脚排列如右图所示。



引脚号	功　能	引脚号	功　能
1	接收线信号检测	6	数据通信设备准备就绪（DSR）
2	接收数据线（RxD）	7	请求发送（RTS）
3	发送数据线（TxD）	8	允许发送（CTS）
4	数据终端准备就绪（DTR）	9	振铃指示
5	信号地（SG，与 GND 一致）		
注：在 RS232 通信中，常常仅使用 3 根线：接收线、发送线和地线			

2．RS485

RS485弥补了RS232通信距离短、速率低等缺点，采用差分信号负逻辑，-2V～-6V表示“1”，+2V～+6V表示“0”。所谓差分，就是两线电平相减，得出一个电平信号，可以较好地抑制电磁干扰，延长通信距离，通信距离在1000米左右。

在硬件连接上采用两线制，数据发送和接收都要使用这对差分信号线，发送和接收不能同时进行，故采用半双工的方式工作。若要全双工通信，必须使用四线。

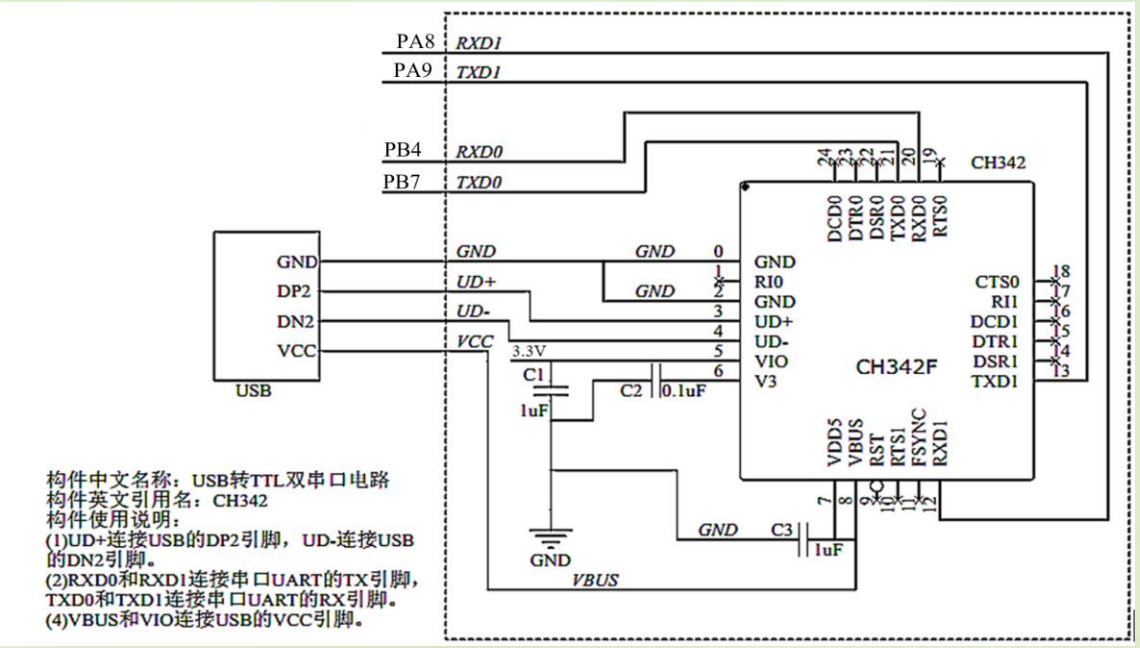
【思考】 为什么差分传输可以较好地抑制电磁干扰？

6.1.3 TTL-USB串口

1. CH342简介

是一款TTL-USB串口转接芯片，能够实现两个串口与USB信号的转换。

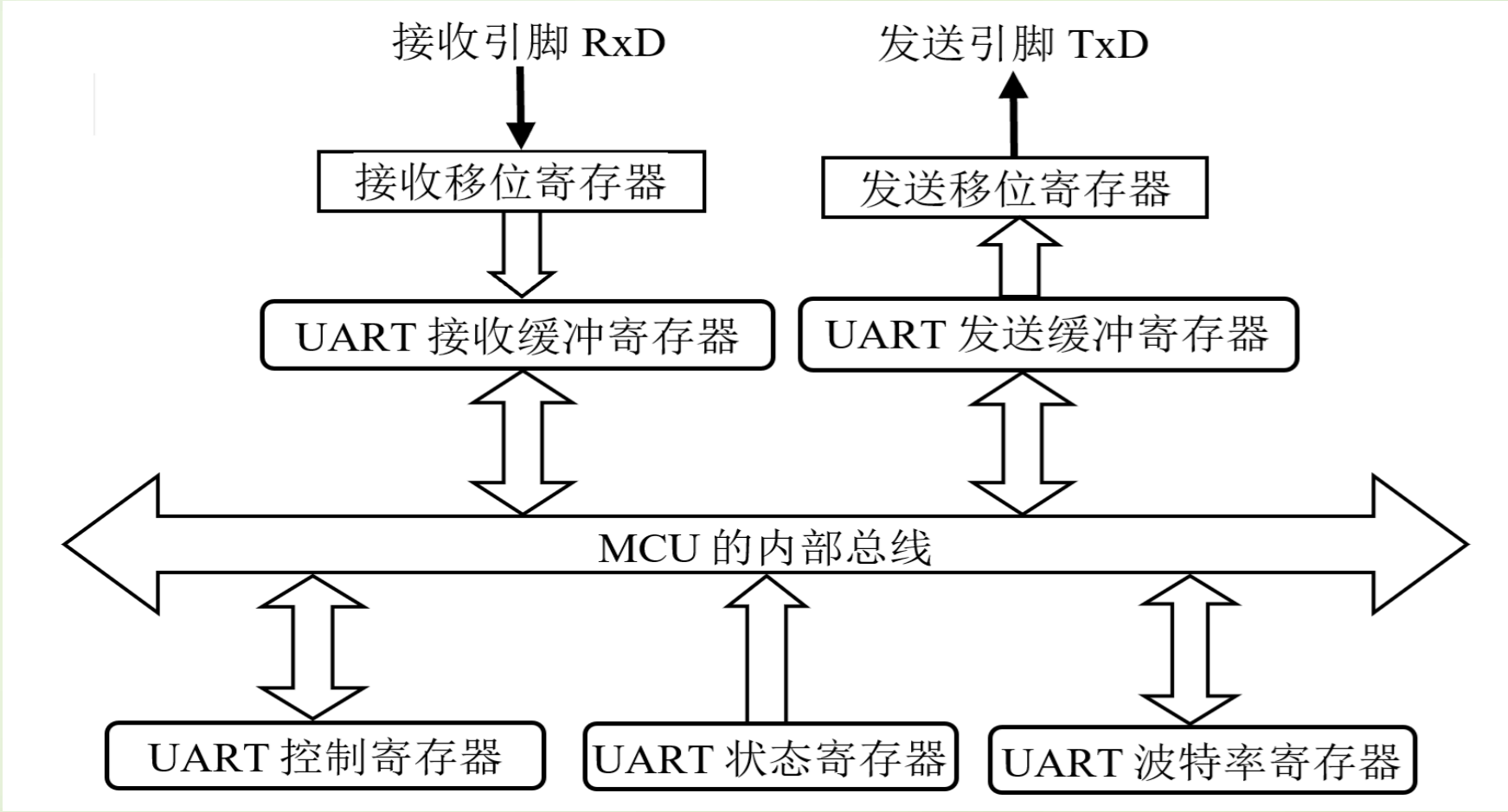
2. CH342与CH573的连接电路



3. CH342串口的使用

可在设备管理器中查看 

6.1.4 串行通信编程模型



6.2 基于构件的串行通信编程方法

6.2.1 CH573芯片UART对外引脚

表6-2 CH573芯片UART模块的引脚分布

串行口	MCU 引脚号	MCU 引脚名	收/发引脚	AHL-CH573 默认使用
UART_0	15	PB4	RX	编程默认使用（UART_User）
	14	PB7	TX	
UART_1	6	PA8	RX	编程默认使用（UART_Debug） （BIOS 用于串口程序下载及 printf）
	7	PA9	TX	
UART_2	17	PB22	RX	用户可自行用于功能设计
	16	PB23	TX	
UART_3	23	PA4	RX	用户可自行用于功能设计
	24	PA5	TX	

6.2.2 UART构件API

1. UART常用接口函数简明列表

表6-3 UART常用接口函数

序号	函数名	简明功能	描述
1	uart_init	初始化	传入串口号及波特率，初始化串口
2	uart_send1	发送一个字节数据	向指定串口发送一个字节数据
3	uart_sendN	发送 N 个字节数据	向指定串口发送 N 字节数据
4	uart_send_string	发送字符串	向指定串口发送字符串
5	uart_re1	接收一个字节数据	从指定串口接收一个字节数据
	…		

2. UART构件的头文件uart.h

```
//=========================================================================
//函数名称：uart_init
//功能概要：初始化 uart 模块
//参数说明：uartNo: 串口号:UART_0、UART_1、UART_2、UART_3
//          baud_rate ：波特率，可取 9600、19200、115200...
//函数返回：无
//=========================================================================
void uart_init (uint8_t uartNo, uint32_t baud_rate);


//=========================================================================
//函数名称：uart_send1
//参数说明：uartNo: 串口号:UART_0、UART_1、UART_2、UART_3
//          ch：要发送的字节
//函数返回：函数执行状态，1 表示发送成功；0 表示发送失败
//功能概要：串行发送 1 个字节
//=========================================================================
uint_8 uart_send1(uint8_t uartNo, uint8_t ch);
```

6.2.3 UART构件API的发送测试方法

1. MCU方程序的编制

（1）对所使用的串口号进行宏定义。

在工程的05_UserBoard文件夹下的user.inc进行宏定义

.equ UART_User,(0)

（2）初始化串口。在main.s中，其进行初始化，uart_init需要串口号和波特率两个参数，分别由寄存器a0、a1传入：

```
/*  初始化串口 UART_User */
li a0,UART_User                    /*  第 1 参数 a0：串口号  */
li a1,115200                       /*  第 2 参数 a1：波特率  */
call uart_init                     /*  调用函数  */
```

（3）循环发送数据。在main.s的主循环中，发送数字“48～57”

```
        /*  【6.2 节增加部分（开始）】 UART_User 发送增加的语句  */
        li t1,10                                    /*  需要发送的数据个数，作为循环次数  */
        li t2,0                                     /*  使用临时寄存器 t2 作为比较  */
        li t3,48                                    /*  待发送的数据初值  */
main_uart_sent:
        /*  从串口 UART_User 发送数据  */
        li a0,UART_User                             /*  串口号  */
        mv a1,t3                                    /*  待发送的一个字节  */
        call uart_send1                             /*调用函数  */
        addi t3,t3,1
        addi t1,t1,-1                               /*  待发送的数据+1 */
        bne t1,t2,main_uart_sent                    /*  循环次数-1
        /*  【6.2 节增加部分（结尾）】  */          /*  等于 0 转  */
```

2. 编译下载测试

样例工程见【03-Software\CH06\UART-Sent】

6.3 中断机制及中断编程步骤

6.3.1 中断基本概念及处理过程

1. 中断相关基本概念

（1）异常和中断。
（2）中断源与中断向量号。
（3）中断服务例程。
（4）中断向量表。
（5）可屏蔽中断与不可屏蔽中断。
（6）中断源的优先级。

2. 中断处理的基本过程

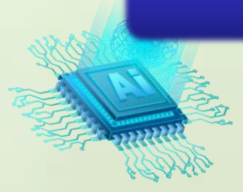
（1）中断请求。
（2）中断检测与响应。
（3）运行ISR。
（4）中断返回。

6.3.2 CH573中断源及RISC-V3A中断结构

1. CH573中断源（表6-4）

2. RISC-V3A中断结构（了解）

3. 使用PFIC寄存器进行模块中断使能与除能的编程（了解）



6.3.2 CH573中断编程步骤—以串口接收中断为例（重点）

1. 功能描述

在第4章GPIO-ASM样例程序基础上增加UART_User串口的收发功能。主循环功能不变，仍为蓝灯闪烁，增加UART_User串口接收中断服务例程UART_User_Handler，其功能为收到一个字节，立即发送该字节，这个功能可以通过开发环境中的串口工具进行测试。

2. 主程序中的编程及有关设置

1）复制模板程序

2）在用户接口文件user.inc中增加用户串口相关宏定义

```
.equ  UART_User,(0)
```

3）初始化串口并使能串口模块接收中断
（对照程序）
4）注册中断服务例程（难点）
（对照程序）

3. 在isr_asm.s文件中编写用户串口接收中断服务例程

（对照程序）

4. 运行结果

6.4 RISC-V汇编工程框架解析

在软件工程的的语境中，开发一个软件被郑重地归为做一个工程，要向做工程一样对待软件开发。一个软件包含工程说明文件、程序文件、头文件、编译生成的文件等，文件众多，合理组织这些文件，规范工程组织，可以提高开发效率、阅读清晰度及可维护性，还可以降低维护难度。工程组织应体现软件工程的基本原则与基本思想，做到“分门别类，各有归处”。

6.4.1 汇编工程框架的基本内容

1. 工程文件夹内的基本内容

表6-5 工程文件夹内的基本内容

文件夹中文名	文件夹英文名		简明功能
文档文件夹	01_Doc		内含 readme.txt 文件，供书写工程备忘使用。
CPU 文件夹	02_CPU		与内核相关的文件，由 CPU 厂家提供。
MCU 文件夹	03_MCU	linker_File	链接文件夹，存放链接文件。
		MCU_drivers	MCU 底层驱动构件文件夹，存放芯片级硬件驱动。
		startup	启动文件夹，存放芯片头文件及芯片初始化文件。
GEC 相关文件夹	04_GEC		为构建通用嵌入式计算机做准备。
用户板文件夹	05_UserBoard		用户板文件夹，存放应用级硬件驱动，即应用构件。
算法构件文件夹	06_AlgorithmComponent		算法构件文件夹，存放与硬件不密切相关的软件构件。
应用程序文件夹	07_AppPrg	include.inc	总头文件，包含各类宏定义。
		isr_asm.s	中断服务例程文件，存放各中断服务例程子函数。
		main.s	主程序文件，存放芯片启动的入口函数 main。

2. 有关文件夹内容说明

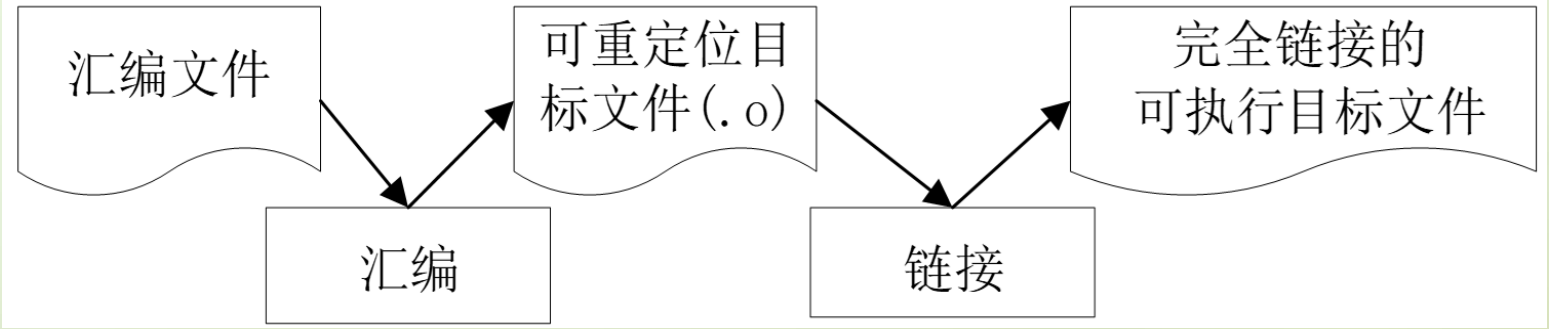
1）CPU文件夹
2）MCU文件夹
3）用户板文件夹
4）应用程序文件夹

3. 编译链接产生的其他相关文件简介

编译链接产生Debug文件夹，内含列表文件（.lst）、映像文件（.map）、十六进制机器码文件（.hex）、可执行链接格式机器码文件（.elf）等。

6.4.2 链接脚本文件的作用 （难点）

脚本是指表演戏剧、拍摄电影等所依据的底本又或者书稿的底本，也可以说是故事的发展大纲，是用来确定故事到底是在什么地点，什么时间，有哪些角色，角色的对白，动作，情绪的变化，等等。而在计算机中，脚本（Script）是一种批处理文件的延伸，是一种纯文本保存的程序，是确定的一系列控制计算机进行运算操作动作的组合，在其中可以实现一定的逻辑分支等。链接脚本文件（.ld），简称链接文件，是用于控制链接的过程，规定了如何把输入的中间文件中的section映射到最终目标文件内，并控制目标文件内各部分的地址分配，它为链接器提供链接脚本。



6.4.3 机器码解析 （重点、难点）

1. 十六进制机器码文件的记录格式

表6-6 .hex文件记录行语义

	字段 1	字段 2	字段 3	字段 4	字段 5	字段 6
名称	记录标记	记录长度	偏移量	记录类型	数据/信息区	校验和
长度	1 字节	1 字节	2 字节	1 字节	N 字节	1 字节
内容	开始标记“:”		数据类型记录有效； 非数据类型，该字段为“0000”。	00-数据记录； 01-文件结束记录； 02-扩展段地址； 03-开始段地址； 04-扩展线性地址； 05-链接开始地址。	取决于记录类型	开始标记之后字段的所有字节之和的补码。 校验和=0xFF-(记录长度+记录偏移+记录类型+数据段)+0x01

2. 实例分析

表6-7　样例工程中的.hex 文件部分行分解

行	记录标记	记录长度	偏移量	记录类型	数据/信息区	校验和
1	:	02	0000	02	10000	EC
2	:	10	C400	00	6F10900E13000000130000001300000	D6
822	:	04	0000	03	1000C4	25
823	:	00	0000	01		FF

参见实际工程，逐行分析

实验三：基于串行通信构件的汇编程序设计

在学院网站上传实验报告及实验程序，注意注释规范。

本章作业：1～5