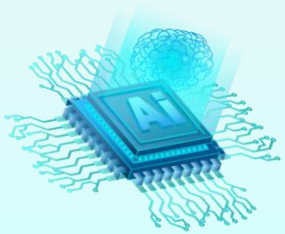
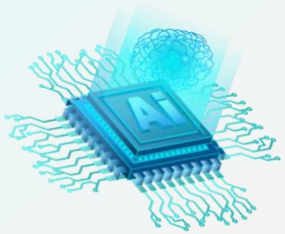
第8章 输入输出系统



本章导读：

微型计算机要与外界打交道，必然涉及输入输出系统。本章给出微型计算机中一些基本输入输出系统，包括中断系统与定时器、串行通信接口、模数转换接口、PWM等，主要目的是阐述如何通过软件编程接受输入信号和干预外部器件，即理解软件干预硬件的基本方法。



8.1 中断系统与定时器

8.1.1 关于中断的通用基础知识

中断提供了一种程序运行的机制，用来打断当前正在运行的程序，并且保存当前CPU状态（CPU内部寄存器），转而去运行一个中断服务例程，然后恢复CPU到运行中断之前的状态，同时使得中断前的程序得以继续运行。

1．中断的基本概念

1）中断与异常的基本含义

中断与异常属于同一概念的两种不同产生条件。异常（exception）是CPU强行从正常的程序运行切换到诸如被除数为0这样一类非正常情况的任务上去，而把来自CPU外围设备的强行任务切换请求称为中断（interrupt）。软件上，中断与异常均表现为将程序计数器（PC）指针强制转到中断服务例程（Interrupt Service Routine，ISR）入口地址运行。

2）中断源、中断服务例程、中断向量号与中断向量表

可以引起CPU中断的外部器件被称为中断源。一个CPU能识别多个中断源，芯片制造时，给CPU能够识别的各个中断源编个号，就叫中断向量号。中断向量表按中断向量号从小到大的顺序填写ISR的首地址。

3）中断优先级、可屏蔽中断和不可屏蔽中断（了解）。

2．中断处理的基本过程

1）中断请求

当某一中断源需要CPU为其服务时，它将会向CPU发出中断请求信号（一种电信号）。

2）中断采样

一般情况下，CPU在每条指令结束的时候，会检查中断请求或者系统是否满足异常条件，为此，一些CPU专门在指令周期中使用了中断周期。

3）中断响应过程（重点）

中断响应的过程是由系统自动完成的，对于用户来说是透明的操作。在中断的响应过程中，首先CPU会查找中断源所对应的模块中断是否被允许，若被允许，则响应该中断请求。中断响应的过程要求CPU保存当前环境的“上下文（context）”于堆栈中。通过中断向量号找到中断向量表中对应的中断服务例程ISR，转而去运行该中断服务例程ISR。

上下文：中断处理术语中，简单的理解“上下文”即指CPU内部寄存器，其含义是在中断发生后，由于CPU在中断服务例程中也会使用CPU内部寄存器，所以需要在调用ISR之前，将CPU内部寄存器保存至指定的RAM地址（栈）中，在中断结束后再将该RAM地址中的数据恢复到CPU内部寄存器中，从而使中断前后程序的“运行现场”没有任何变化。

8.1.2 CH573中断向量表及中断向量号宏定义

1．CH573中断向量表

打开startup_CH573.S文件

```
.section    .vector,"ax",@progbits
.align  1
_vector_base:
.option norvc;
.word   _eusrstack
.word   0
j   NMI_Handler            /* NMI Handler */
```

为什么要弱定义?

```
.weak   NMI_Handler
```

2．CH573中断向量号宏定义

打开CH573SFR.h文件

```
typedef enum IRQn
{
    Reset_IRQn          = 1,
    NMI_IRQn            = 2,    /*!<  Non Maskable Interrupt   */
    EXC_IRQn            = 3,    /*!<  Exceptions Interrupt     */
    SysTick_IRQn        = 12,   /*!<  System timer Interrupt  */
    SWI_IRQn            = 14,   /*!<  software Interrupt */
    TMR0_IRQn           = 16,
    GPIO_A_IRQn         = 17,
        ……
    WDOG_BAT_IRQn       = 35
} IRQn_Type;
```

8.1.3 定时器通用基础知识

1．完全软件方式

是利用计算机执行指令的时间实现定时，但这种方式占用CPU，不适用于多任务环境，一般仅用于时间极短的延时且重复次数较少的情况。

```
//延时若干指令周期（注意加volatile表示不优化，以保证延时一致性）
for (volatile  uint32_t i = 0; i < 80000; i++)  __ASM("NOP");
```

2．可编程定时器

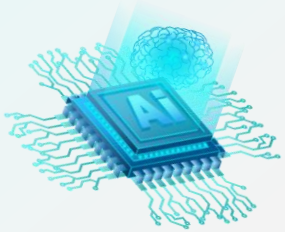
是根据需要的定时时间，用指令对定时器进行初始常数设定，并用指令启动定时器开始计数，当计数到指定值时，便自动产生一个定时输出，通常为中断信号告知CPU，在定时中断处理程序中，对时间进行基本运算。在这种方式中，定时器开始工作以后，CPU不必去管它，可以运行其他程序，计时工作并不占用CPU的工作时间。在实时操作系统中，利用定时器产生中断信号，建立多任务程序运行环境，可大大提高CPU的利用率。

8.1.4 内核中系统定时器SysTick

运行程序。

要点：（重点掌握）

（1）中断入口问题
（2）中断服务例程中的静态变量；
（3）秒加1程序及其健壮性问题。

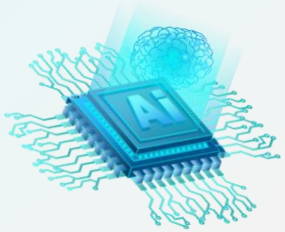


8.2 串行通信接口

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。作为设备间简便的通信方式，在相当长的时间内，串口还不会消失，在市场上也很容易的购买到各种电平到USB的串口转接器，以便与没有串口但具有多个USB口的笔记本电脑或PC机连接。MCU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，是嵌入式开发中重要的打桩调试手段。实现串口功能的模块在一部分MCU中被称为通用异步收发器（Universal Asynchronous Receiver-Transmitters，UART），在另一些MCU中被称为串行通信接口（Serial Communication Interface，SCI）。

8.2.1 串行通信的基本概念与编程模型

串行通信的基本概念导引：“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，通常一个信息单位用8位二进制表示，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。这里至少涉及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6A，即二进制01101010，格式为：

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。

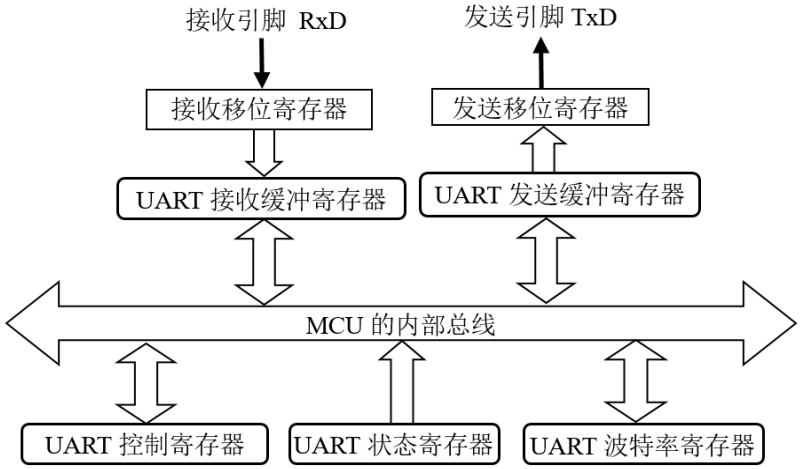
3. 串行通信传输方式术语（了解）

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

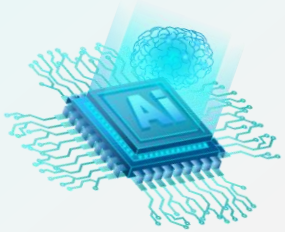
单工：数据传送方向为单向

4. 串行通信编程模型（了解）



8.2.2 串行通信的硬件信号变换（了解）

1. RS232
2. RS485
3. TTL-USB串口



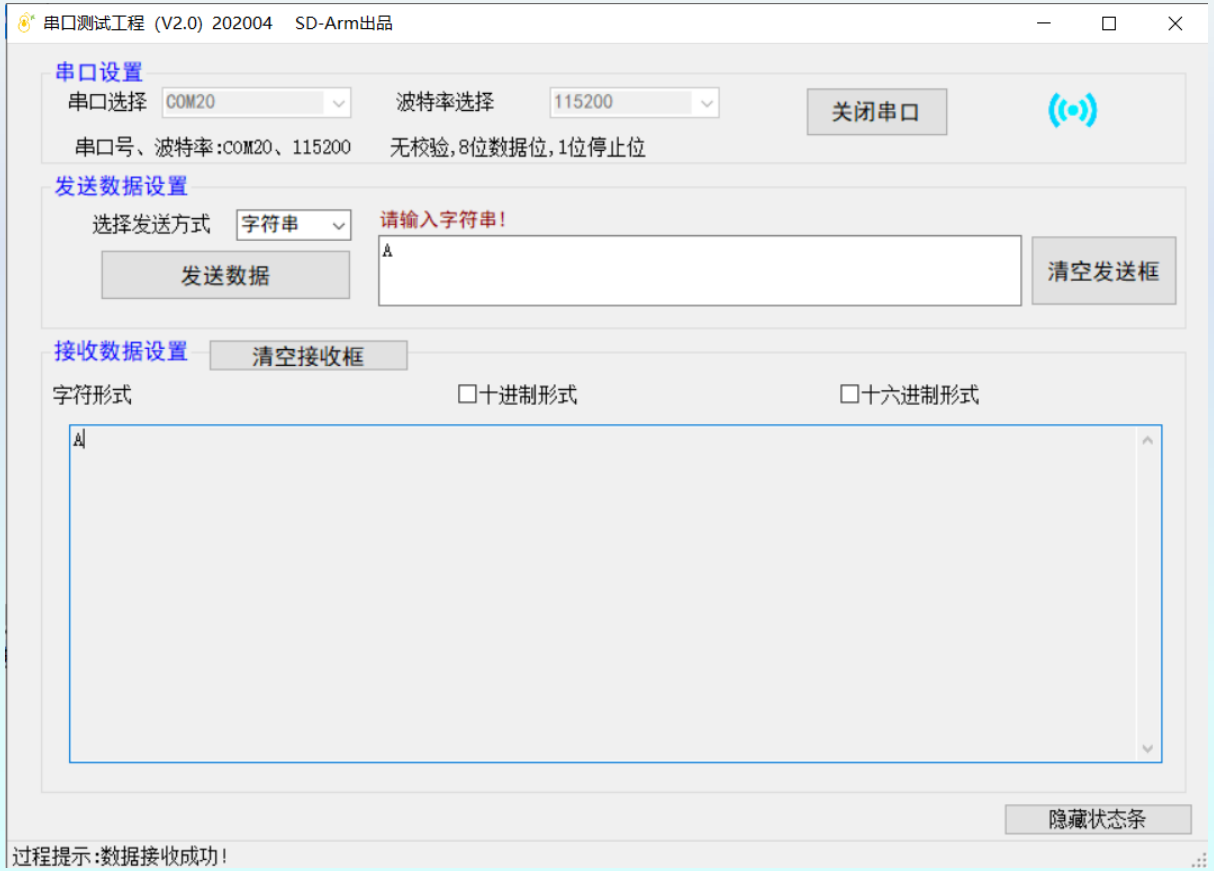
8.2.3 UART构件API

```
//=====================================================================
//函数名称：uart_init
//功能概要：初始化uart模块
//参数说明：uartNo—串口号，如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:如 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);
```

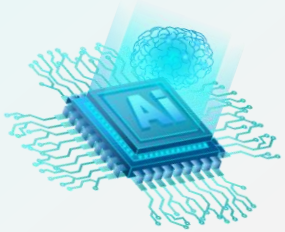
8.2.4 中断编程步骤—以串口接收中断为例

（参见实例）



8.3 A/D转换

在过程控制和仪器仪表中，多数情况下是由嵌入式计算机进行实时控制及实时数据处理的。计算机所加工的信息是数字量，而被测控对象往往是一些连续变化的模拟量（如温度、压力、速度或流量等）。模数（A/D，Analog/Digital）转换模块是嵌入式计算机与外界连接的纽带，是大部分嵌入式应用中必不可少的重要组成部分，该部分的性能直接影响到嵌入式设备的总体性能。



8.3.1 ADC的通用基础知识

1．模拟量、数字量及模/数转换器的基本含义

模拟量（Analogue Quantity）是指变量在一定范围连续变化的物理量，从数学角度，连续变化可以理解为可取任意值。例如，温度这个物理量，可以有28.1℃，也可以有28.15℃，还可以有28.152℃，……，也就是说，原则上可以有无限多位小数点，这就是模拟量连续之含义。当然，实际达到多少位小数点则取决于问题需要与测量设备性能。

数字量（Digital Quantity）是分立量，不可连续变化，只能取一些分立值。现实生活中，有许多数字量的例子，如1部手机、2部手机，……，你不能说你买0.12部手机，那它接不了电话！在计算机中，所有信息均使用二进制表示。例如，用一位只能表达0、1两个值，8位可以表达0、1、2、……、254、255，共256个值，不能表示其他值，这就是数字量。

模/数转换器（Analog-to-Digital Converter，ADC）是将电信号转换为计算机可以处理的数字量的电子器件，这个电信号可能是由温度、压力等实际物理量经过传感器和相应的变换电路转化而来的。

2．与A/D转换编程直接相关的技术指标

1）转换精度　　（重点）

转换精度（Conversion accuracy）是指数字量变化一个最小量时对应模拟信号的变化量，也称为分辨率（Resolution），通常用模数转换器ADC的二进制位数来表征，通常有8位、10位、12位、16位、24位等，转换后的数字量简称A/D值。通常位数越大，精度越高。设ADC的位数为N，因为N位二进制数可表示的范围是0～(2^N-1)，因此最小能检测到的模拟量变化值就是$1/2^N$。例如，某一ADC的位数12位，若参考电压为5V（即满量程电压），则可检测到的模拟量变化最小值为$5/2^{12}$=0.00122（V）=1.22（mV），就是这个ADC的理论精度（分辨率）了。这也是12位二进制数的最低有效位（Least Significant Bit，LSB）所能代表的值，即在这个例子中，1LSB=5*（1/4096）=1.22（mV）。实际上由于量化误差的存在，实际精度达不到。与二进制最低有效位相对应的是最高有效位（Most Significant Bit，MSB），12位二进制数的最高有效位MSB代表2048，而最低有效位代表1/4096。不同位数的二进制中，MSB和LSB代表的值不同。

2）单端输入与差分输入

3）软件滤波问题

4）物理量回归问题

3．与A/D转换编程关联度较弱的技术指标

1）量化误差　（重点）

在把模拟量转换为数字量过程中，要对模拟量进行采样和量化，使之转换成一定字长的数字量，量化误差（Quadratuer Error）就是指模拟量量化过程而产生的误差。举个例子来说，一个12位A/D转换器，输入模拟量为恒定的电压信号1.68V，经过A/D转换器转换，所得的数字量理论值应该是2028，但编程获得的实际值却是2026～2031之间的随机值，它们与2028之间的差值就是量化误差。

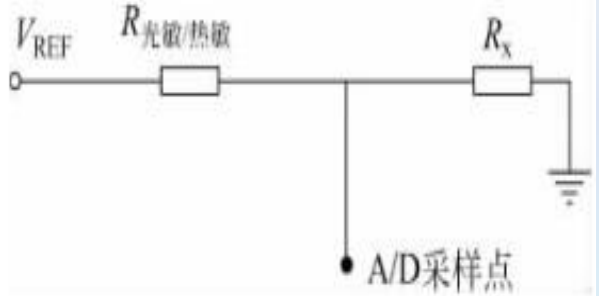
2）转换速度

3）A/D参考电压

4．最简单的A/D转换采样电路举例

(a)

(b)



(c)

$$V_{A/D} = \frac{R_x}{R_{光敏} + R_x} \times V_{REF}$$

8.3.2 基于构件的ADC编程方法

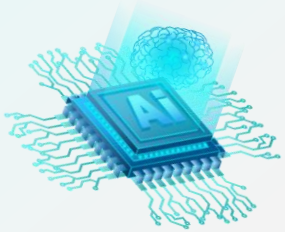
```
//=====================================================================
//函数名称：adc_init
//功能概要：初始化一个AD通道号
//参数说明：Channel：通道号（使用本文件头部宏定义）
//          NC：本函数未使用，为增强函数可移植性
//=====================================================================
void adc_init(uint16_t Channel,uint16_t NC);

//=====================================================================
//函数名称：adc_read
//功能概要：将模拟量转换成数字量，并返回
//参数说明：Channel：通道号（使用本文件头部宏定义）
//=====================================================================
uint16_t adc_read(uint8_t Channel);
```

应用方法：实例程序，掌握编程步骤

8.4 脉宽调制PWM

脉宽调制是一种可以通过软件编程方式从芯片引脚输出周期性高低电平持续时间可以调整的信号，用于电机的变频控制、灯光的细分亮暗控制等。

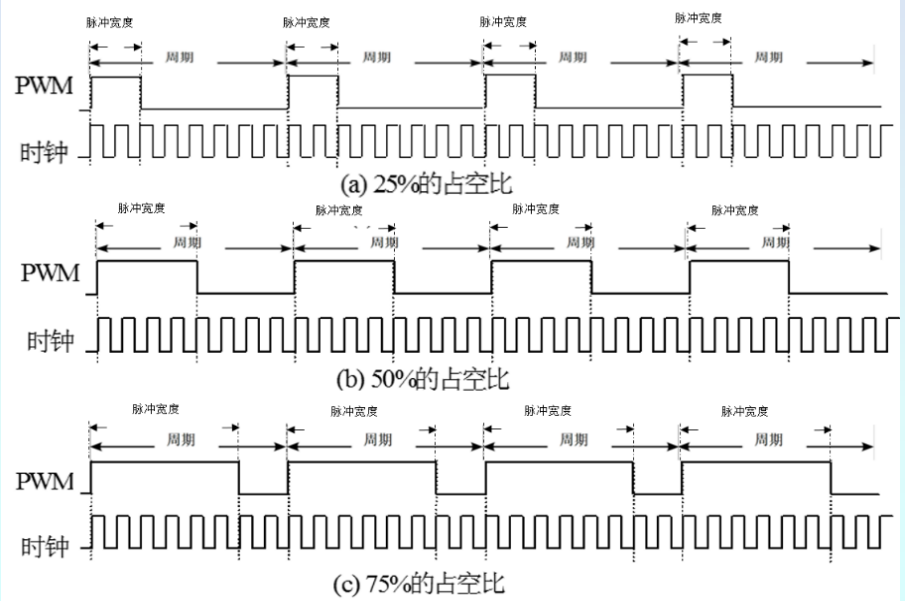


8.4.1 脉宽调制PWM通用基础知识

1．PWM知识要素

脉宽调制（Pulse Width Modulator，PWM）是电机控制的重要方式之一，PWM信号是一个高/低电平重复交替的输出信号，通常也叫脉宽调制波或PWM波。

1）时钟源频率、PWM周期与占空比



2）脉冲宽度与分别率

脉冲宽度是指一个PWM周期内，PWM波处于有效电平的时间（用持续的时钟周期数表征）。

PWM分辨率△T是指脉冲宽度的最小时间增量，等于时钟源周期，△T=T_{CLK}，也可不作为一个独立的技术指标。例如，若PWM是利用频率F_{CLK}=48MHz的时钟源产生的，即时钟源周期T_{CLK}=（1/48）μs=20.8ns，那么脉冲宽度的每一增量值即为△T=20.8ns，就是PWM的分辨率。它就是脉冲宽度的最小增量，脉冲宽度的增加与减少只能是△T的整数倍，实际上脉冲宽度正是用有效电平持续的时钟周期数（整数）来表征。

3）极性

PWM极性决定了PWM波的有效电平。正极性表示PWM有效电平为高电平，负极性表示PWM有效电平为低电平。

4）对齐方式（了解）

2．PWM的应用场合

（1）**利用PWM为其他设备产生类似于时钟的信号**。例如，PWM可用来控制灯以一定频率闪烁。

（2）**利用PWM控制输入到某个设备的平均电流或电压**，在一定程度上可以替代D/A转换。例如，一个直流电机在输入电压时会转动，而转速与平均输入电压的大小成正比。假设每分钟转速（rpm）=输入电压的100倍，如果转速要达到125rpm，则需要1.25V的平均输入电压；如果转速要达到250rpm，则需要2.50V的平均输入电压。在不同占空比的图8-2中，如果逻辑1是5V，逻辑0是0V，则（a）的平均电压是1.25V，（b）的平均电压是2.5V，（c）的平均电压是3.75V。可见，利用PWM可以设置适当的占空比来得到所需的平均电压，如果所设置的PWM周期足够小，电机就可以平稳运转（即不会明显感觉到电机在加速或减速）。

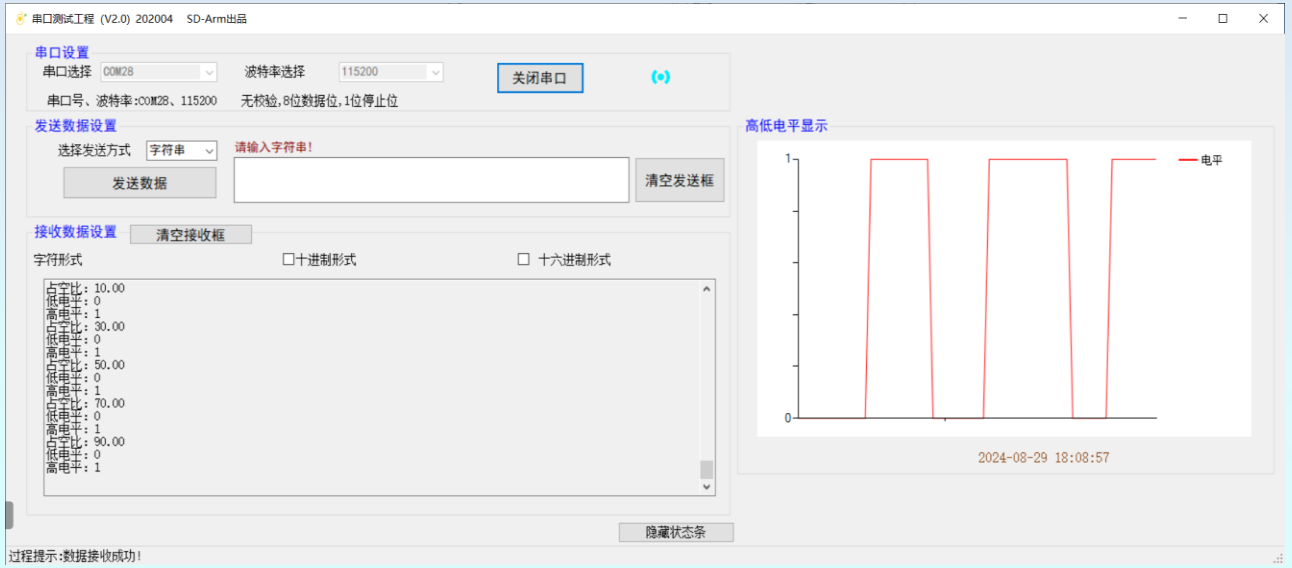
（3）**利用PWM控制命令字编码**。例如，通过发送不同宽度的脉冲，代表不同含义。假如用此来控制无线遥控车，宽度1ms代表左转命令，4ms代表右转命令，8ms代表前进命令。接收端可以使用定时器来测量脉冲宽度，在脉冲开始时启动定时器，脉冲结束时停止定时器，由此来确定所经过的时间，从而判断收到的命令。

8.4.2 基于构件的PWM编程方法

```
//=====================================================================
// 函数名称：pwm_init
// 功能概要：pwm初始化函数
// 参数说明：pwmNo：通道号，使用.h文件中的宏常数
//           clockFre：时钟频率，单位：hz
//                     定时器PWM选2-60M；专用PWM选：235294-60M
//           period：  周期，单位为个数，即计数器跳动次数，
//                     定时器PWM：可选不大于时钟频率的任何正整数；
//                     专用PWM：只能取31、32、63、64、127、128、255、256
//           duty：占空比：0.0-100.0对应0%-100%
//           align：对齐方式，本构件无，取0
//           pol：极性，在头文件宏定义给出，如PWM_PLUS为正极性
//函数返回：无
//使用说明：使用时注意，专用PWM的时钟频率和周期是所有通道共用的，
//           一个通道不能独立设置，但占空比和极性可以独立设置
//=====================================================================
void pwm_init(uint16_t pwmNo,uint32_t clockFre,uint32_t period,
              double  duty,uint8_t align, uint8_t pol);
```

```
//=====================================================================
// 函数名称：pwm_update
// 功能概要：PWM更新
// 参数说明：pwmNo：通道号，使用.h文件中的宏常数
//           duty：占空比：0.0-100.0对应0%-100%
// 函数返回：无
//=====================================================================
void pwm_update(uint16_t pwmNo,double duty);
```

应用方法：实例程序
掌握编程步骤



Thank you