



《单片机实践》课程报告

UART模块分析

小组成员 王若愚 自动化1901 201906060317

小组成员 林宇航 自动化1901 201906060308

小组成员 王若愚 自动化1901 201906060317

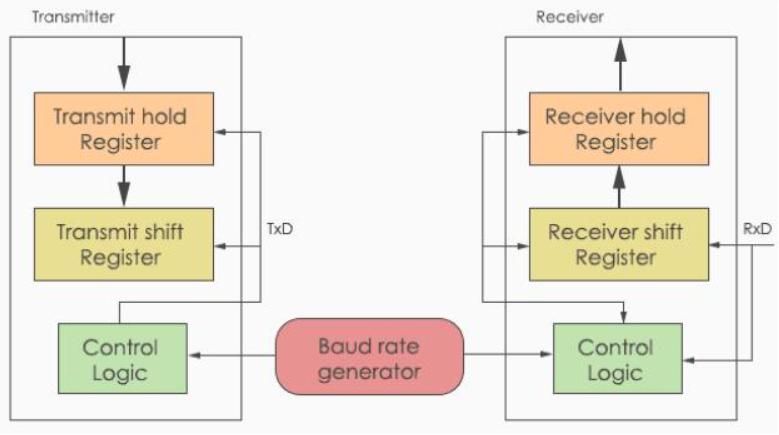
指导老师 郭方洪

**模块分析：UART模块**

1. **UART简介**

“UART”代表通用异步收发器。它是微控制器内部的硬件外围设备。UART的功能是将传入和传出的数据转换为串行二进制流。使用串行到并行转换将从外围设备接收的8位串行数据转换为并行形式，使用串行到并行转换将从CPU接收的并行数据转换为并行形式。该数据以调制形式存在，并以定义的波特率传输。

1. **UART工作原理**
2. **UART框图**

UART由以下核心组件组成。它们是发送器和接收器。发送器由发送保持寄存器，发送移位寄存器和控制逻辑组成。类似地，接收器由接收保持寄存器，接收器移位寄存器和控制逻辑组成。通常，发送器和接收器都配有波特率发生器。****

波特率发生器生成发送器和接收器必须发送/接收数据的速度。发送保持寄存器包含要发送的数据字节。发送移位寄存器和接收移位寄存器将这些位向左或向右移位，直到发送/接收一个字节的数据为止。

除了这些，还提供了读或写控制逻辑以告知何时进行读/写。波特率发生器产生的速度范围从110 bps（每秒比特）到230400。大多数情况下，微控制器提供更高的波特率，例如115200和57600，以实现更快的数据传输。GPS和GSM等设备在4800和9600中使用较低的波特率。

1. **UART通信协议**

UART作为异步串口通信协议的一种，工作原理是将传输数据的每一个字符一位一位地传输。其中每一位(bit)的意义如下：

起始位：先发出一个逻辑“0”的信号，表示传输字符开始

数据位：紧接着起始位之后。数据位的个数可以是4、5、6、7、8等，构成一个字符。通常采用ASCII码。从最低位开始传送，靠时钟定位。

奇偶校验位：数据位加上这一位后，使得“1”的位数应为偶数(偶校验)或奇数(奇校验)，以次来校验数据传送的正确性。

停止位：它是一个字符数据的结束标志。可以是1位、1.5位、2位的高电平。由于数据是在传输线上定时的，并且每一个设备有其自己的时钟，很可能在通信中两台设备间出现了小小的不同步。因此停止位不仅仅是表示传输的结束，并且提供计算机校正时钟同步的机会。适用于停止位的位数越多，不同时钟同步的容忍程度越大，但是数据传输率也就越慢。

空闲位：处于逻辑“1”状态，表示当前线路上没有数据传输。

1. **UART工作时序**

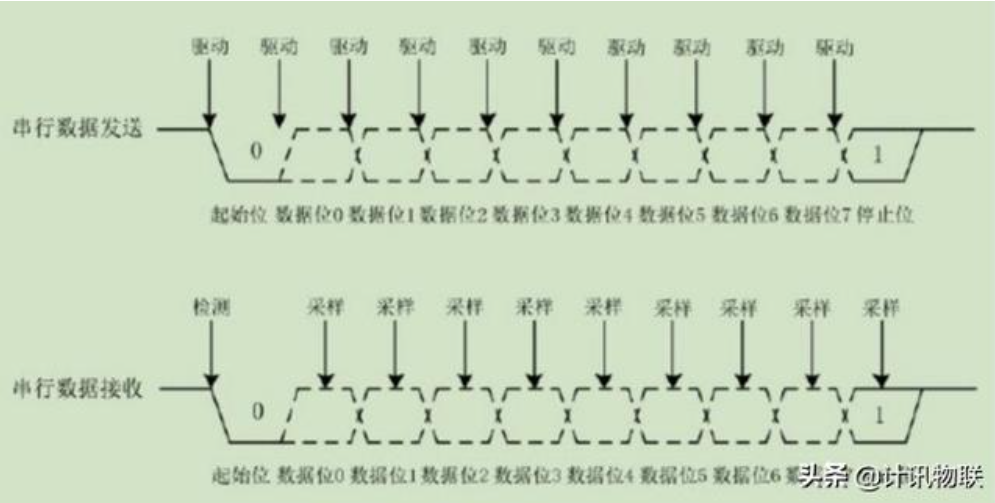
发送数据过程：空闲状态，线路处于高电平;当收到发送指令后，拉低线路的一个数据位的时间T，接着数据按低位到高位依次发送，数据发送完毕后，接着发送奇偶校验位和停止位，一帧数据发送完成。

数据接收过程：空闲状态，线路处于高电平;当检测到线路的下降沿(高电平变为低电平)时说明线路有数据传输，按照约定的波特率从低位到高位接收数据，数据接收完毕后，接着接收并比较奇偶校验位是否正确，如果正确则通知后续设备接收数据或存入缓冲。

由于UART是异步传输，没有传输同步时钟，为了保证数据的正确性，UART采用16倍数据波特率的时钟进行采样。每个数据有16个时钟采样，取中间的采样值，以保证采样不会滑码或误码。一般UART一帧的数据位数为8，这样即使每个数据有一个时钟的误差，接收端也能正确地采样到数据。

UART的接收数据时序为：当检测到数据的下降沿时，表明线路上有数据进行传输，这是计数器CNT开始计数，当计数器为24=16+8时，采样的值为第0位数据;当计数器的值为40时，采样的值为第一位数据，依次类推，进行后面6个数据的采样。如果需要进行奇偶校验，则当计数器的值为152时，采样的值即为奇偶位;当计数器的值为168时，采样的值为“1”表示停止位，数据接收完成。

一个标准的10位异步串行通信协议(1个起始位、1个停止位和8个数据位)收发时序，如下图所示：

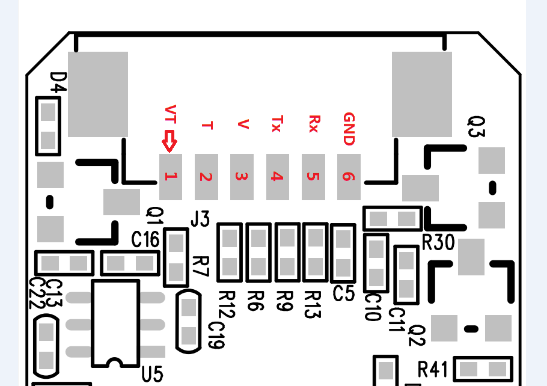


1. **UART在指纹模块中的使用**
2. **模块简介**

FPC1020A是为单片机设计的指纹识别二次开发模块，具有体积小、功耗低、接口简单、可靠性高、指纹模板小( 200字节)、大容量指纹识别( 1000枚指纹识别响应时间小于1秒)等优点，可以非常方便将其嵌入用户系统，组成满足客户需求的指纹识别产品。

1020A模块通讯接口为UART或USB，本模块作为从设备，由主设备发送相关命令对其进行控制，本次使用的是UART。

1. **模块引脚图**



|  |  |  |  |
| --- | --- | --- | --- |
| **引脚号** | **名 称** | **类型** | **功 能 描 述** |
| 1 | VT | In | 手指检测电源正输入端，3.3V |
| 2 | T | Out | 手指检测信号输出端 |
| 3 | V | in | 电源正输入端,3.3V |
| 4 | TX | out | 串行数据输出，TTL逻辑电平 |
| 5 | RX | in | 串行数据输入，TTL逻辑电平 |
| 6 | GND | － | 信号地，内部与电源地连接 |

根据TXRX交叉接线的原则将指纹模块和单片机相连

1. **模块通讯代码**

**1）功能函数**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*功能：返回信息处理

\*\*参数： cmd 不同命令不同处理

\*\*返回：处理结果

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **Check\_Package**(**unsigned** **char** cmd)

{

**unsigned** **char** flag = FALSE;

**if**(!WaitFpData()) **return** flag; //等待接收返回信息

p = 0 ;

**if**(g\_ucUartRxEnd)

g\_ucUartRxEnd = 0;//清数据包接收标志

**else**

**return** flag;

**if**(rBuf[0] != DATA\_START)**return** flag;

**if**(rBuf[1] != cmd)**return** flag;

**if**(rBuf[6] != CmdGenCHK(g\_ucUartRxLen - 3, &rBuf[1]))**return** flag;

**switch**(cmd)

{

**case** CMD\_ENROLL1:

**case** CMD\_ENROLL2:

**case** CMD\_ENROLL3:

**if**(ACK\_SUCCESS == rBuf[4])flag = TRUE;

**else** **if**(ACK\_USER\_EXIST == rBuf[4])

{

// Spk\_HaveUser();

Delay\_ms(1500);

}

**break**;

**case** CMD\_DELETE: //删除指定编号指纹

**case** CMD\_CLEAR: //清空所有指纹

**case** CMD\_IDENTIFY: //1:1比对

**if**(ACK\_SUCCESS == rBuf[4])flag = TRUE;

**break**;

**case** CMD\_USERNUMB: //取用户总数

**if**(ACK\_SUCCESS == rBuf[4])

{

flag = TRUE;

l\_ucFPID = rBuf[3];

}

**break**;

**case** CMD\_SEARCH: //1:N比对

**if**((1 == rBuf[4])||(2 == rBuf[4])||(3 == rBuf[4]))

{

flag = TRUE;

l\_ucFPID = rBuf[3];

}

**break**;

**default**:

**break**;

}

**return** flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*功能：等待数据包发送完成

\*\*参数：

\*\*返回：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **WaitFpData**(**void**)

{

**unsigned** **char** i;

**for**(i=200; i>0; i--)//等待指纹芯片返回

{

Delay\_ms(40);

**if**(g\_ucUartRxEnd)**break**;

}

**if**(i==0)**return** FALSE;//指纹芯片没有返回

**else** **return** TRUE;

}

**void** **UartSend**(**unsigned** **char** \*Datar,**unsigned** **char** cLength)

{

**do**

{

**UARTCharPut**(UART6\_BASE,\*(Datar++));//默认用UART6的P0P1脚

} **while** (--cLength != 0);

}

**1）指纹录入**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*注册指纹

\*\*输入两次指纹注册一个指纹模板

\*\*参数：UserID 指纹号

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **Enroll\_Step1**(**unsigned** **int** u\_id)

{

**unsigned** **char** buf[BUF\_N];

\*buf = CMD\_ENROLL1;

\*(buf+1) = u\_id>>8;

\*(buf+2) = u\_id&0xff;

\*(buf+3) = 1;

\*(buf+4) = 0x00;

UART\_SendPackage(5, buf);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*注册指纹

\*\*输入三次指纹注册一个指纹模板

\*\*参数：UserID 指纹号

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **Finger\_Enroll**(**unsigned** **int** u\_id)

{

Enroll\_Step1(u\_id);

**if**(FALSE == Check\_Package(CMD\_ENROLL1))**return** FALSE;//一

Delay\_ms(200);

Enroll\_Step2(u\_id);

**if**(FALSE == Check\_Package(CMD\_ENROLL2))**return** FALSE; //二

Delay\_ms(200);

Enroll\_Step3(u\_id);//三

Delay\_ms(200);

**return** Check\_Package(CMD\_ENROLL3);

}

**2）指纹删除**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*功能：删除指定编号指纹

\*\*参数：u\_id

\*\*返回：void

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **FP\_Delete**(**unsigned** **int** u\_id)

{

**unsigned** **char** buf[BUF\_N];

\*buf = CMD\_DELETE;

\*(buf+1) = u\_id>>8;

\*(buf+2) = u\_id&0xff;

\*(buf+3) = 0x00;

\*(buf+4) = 0x00;

UART\_SendPackage(5, buf);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*删除指定指纹

\*\*

\*\*参数：UserID 指纹号

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **Finger\_Delete**(**unsigned** **int** u\_id)

{

FP\_Delete(u\_id);

// if(FALSE == WaitFpData())return FALSE;

**return** Check\_Package(CMD\_DELETE);

}

**3）指纹识别**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*功能：以CharBuffer1 或CharBuffer2 中的特征文件搜索整个或部分指纹库

\*\*参数：

\*\*返回：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **FP\_Search**(**void**)

{

**unsigned** **char** buf[BUF\_N];

\*buf = CMD\_SEARCH; //1:N比对

\*(buf+1) = 0x00;

\*(buf+2) = 0x00;

\*(buf+3) = 0x00;

\*(buf+4) = 0x00;

UART\_SendPackage(5, buf);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*读取用户总数

\*\*

\*\*参数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **Finger\_Search**(**void**)

{

FP\_Search();

**return** Check\_Package(CMD\_SEARCH);

}

1. **模块使用心得**

第一次使用指纹模块还是十分生疏的，但当你真正掌握了那种使用模块的方法，其他不管是何种模块，只要他有一个较好的通讯协议，使用起来也能比较方便。

拿到模块先是利用卖家的测试上位机来确定模块的正常工作，再根据例程中的通讯时序来编写代码。

模块化就是提高其使用的灵活性，降低系统的设计难度，模块化的思想在生活中很常见，不仅仅是一个系统，在日后的工作也时刻体现出这个思想，每个人负责一部分，最后再组合起来，变成一个庞大的系统。

1. **思考与总结**