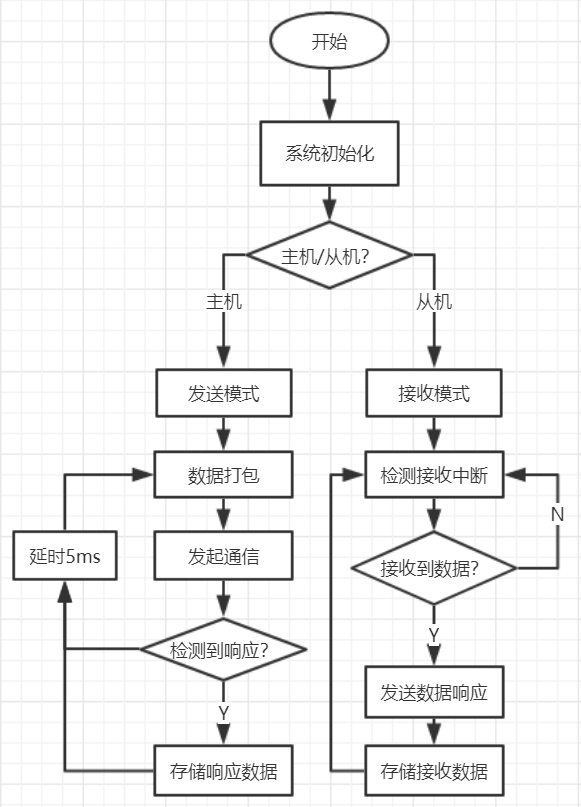
1. 单总线串口通信方案程序框图

在机器人各个电路模块的相互通信方案选择中，此攀爬仿生机器人选择了单总线串口通信方案。串行通信方案是指仅使用一条数据线，将数据一位一位的一次传输，每一位数据占据一个固定的时间长度。相较于常用如IIC、SPI等硬件通信方案，异步串口通信有着连线简单、调试方便等特点，可以借助TLL转USB电路模块利用串口助手软件进行简单调试。但标准串口通信为4线硬件连接（VCC、TX、RX、GND），考虑进一步简化硬件连线，所以采用SN74lvc1G128芯片作为中继器，对TX、RX线进行分时启用，实现单总线串口一定频率上半双工通信的功能。控制程序设计如图。

基本的控制思路是，在单片机系统初始化结束后对不同电路模块的单片机进行主从标签分配，主控板为主机，关节驱动及传感器为从机。并为各个从机分配固定ID号。配置结束后，各个模块开始进行相互通信。

主机在配置结束后默认设置为发送模式（方法见单总线串口通信电路设计），并对需要发送的数据进行打包处理，包括字符拆解、添加数据包标签、数据头尾、校验位等操作。数据打包结束后，由主机首先发起通信，将数据发送到总线上，然后转换为接收模式等待总线上的从机数据响应。在检测到数据响应后开始接收从机的回传数据包后进行有效性校验、字符合成等操作，并存储到单片机固定存储区域，以便于主程序随时调用，然后通过短暂延时控制总线通信频率，至此完成一个完整通信周期。

从机在配置结束后则默认设置为接收模式（方法见单总线串口通信电路设计），打开中断接收检测，并实时检测中断信号。在检测到中断信号后，从机进行中断响应程序，开始从总线上接收数据包并进行数据标签及数据包有效性校验。确定数据包有效性后，从机关断中断检测，将有效数据内容存储到单片机固定存储区域，然后切换到发送模式，进行数据包响应，将数据状态及从机相关数据信息反馈到总线上，切换接收模式，完成单次通信。

1. 通信电路底层驱动配置

通信电路底层驱动以STM32F103系列单片机为例，使用到单片机的1个USART模块和两个通用GPIO，采用官方标准库函数对功能进行配置。

在驱动配置前，通过头文件编写对硬件进行功能映射，Usart.h文件编写如下：

#ifndef \_\_USART\_H\_\_

#define \_\_USART\_H\_\_ //驱动库去重复处理

#include "Global.h" //固件库头文件

#define UART\_TX\_PIN GPIO\_Pin\_2 //定义TX管脚对应端口号

#define UART\_RX\_PIN GPIO\_Pin\_3 //定义RX管脚对应端口号

#define UART\_GPIO GPIOA //定义UART对应模块号

#define UARTx USART2 //定义所使用的UART编号

void UsartIOInit(void); //串口端口初始化函数申明

void UsartConfig(void); //串口参数初始化函数申明

void UsartInit(void); //串口初始化函数申明

void SendData(u8 ch); //串口发送函数申明

void USART2\_IRQHandler(void); //串口接收中断服务函数申明

#endif

然后，在Usart.c文件中对USART模块进行如下配置：

GPIO\_InitTypeDef GPIO\_InitStructure;

//定义UART管脚结构体

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);

//对UART管脚所挂载的APB2时钟进行使能

GPIO\_InitStructure.GPIO\_Pin = UART\_TX\_PIN;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

//设置TX管脚为复用输出模式，工作频率50MHz

GPIO\_InitStructure.GPIO\_Pin = UART\_RX\_PIN;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(UART\_GPIO, &GPIO\_InitStructure);

//设置RX管脚为浮空输入模式，完成UART管脚初始化

USART\_InitTypeDef USART\_InitStructure;

//定义USART串口结构体

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2, ENABLE);

//对USART2所挂载的APB1时钟进行使能

USART\_DeInit(USART2);

//重载USART初始化

USART\_InitStructure.USART\_BaudRate = 9600;

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

USART\_Init(UARTx, &USART\_InitStructure);

//设置波特率9600，字节长度8位，1个停止位，无校验位

//对USART进行参数初始化

NVIC\_InitTypeDef NVIC\_InitStructure;

//定义串口中断结构体

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_0);

//中断优先级分组设置位Group0

NVIC\_InitStructure.NVIC\_IRQChannel = USART2\_IRQn;

//链接中断响应服务函数

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1;

//设置中断优先级为1

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;

//设置子中断优先级为0

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

//中断通道使能

NVIC\_Init(&NVIC\_InitStructure);

//初始化中断参数配置

USART\_ITConfig(UARTx, USART\_IT\_RXNE, ENABLE);

//打开UARTx串口中断检测

USART\_Cmd(UARTx, ENABLE);

//启动UARTx串口功能

至此，完成对单片机USART功能的参数配置及初始化的过程，通用GPIO端口配置过程类似，设置推免输出模式即可。

1. 接收中断程序及数据校验

硬件单总线串口总线通信，主要是基于单片机检测到串口RX管脚上的中断信号后，暂定主程序以跳转到串口接收中断程序来实现的。而在串口中断接收程序中，则包含了中断状态位操作、数据接收、数据校验等过程，程序示例如下：

if(RESET != USART\_GetITStatus(UARTx,USART\_IT\_RXNE)) //判断RX管脚检测到中断信号

{

USART\_ClearITPendingBit(UARTx,USART\_IT\_RXNE); //清空中断状态位

uint8\_t checkdata; //定义校验数据变量

rece = USART\_ReceiveData(UARTx); //逐位接收总线数据

if(rece == 0xAA || rece == 0xBB) //判断数据头和数据尾

{

Get\_data[0] = rece; //对有效数据进行存储

data\_num = 0x01;

}

else if(rece == 0x0A)

{

Get\_data[5] = rece;

data\_num = 0x00;

}

else if((data\_num !=0x00)&&(data\_num<6))

{

Get\_data[data\_num] = rece;

data\_num++;

}

else Get\_data[6] = 0x0E; //对数据错误进行标记

}

至此串口通信程序配置完成。

1. 问题分析

单总线控制程序设计过程中，通信频率及单次通信数据包长度是程序设计的主要考虑的问题。在此方案中，通信频率的主要决定因素是主机发起通信间隔时长，次要因素又包含了单次通信数据包长度。由串口通信原理知，数据的发送是通过逐位将字节发送到串行总线上，同时每一个字节都占据了固定的时间长度。所以若单次通信数据包较长，则数据包发送接收的时间也会相对变长，导致通信频率降低产生延迟。但通信频率也并非越高越好，在单片机工作过程中，依赖定时中断、外部中断等技术手段将不同的任务进行了一定程度的时间片划分，而高频率的通信则会导致通信进程占据较多的MCU时间片，使得MCU资源紧张，无法稳定完成其他进程任务等问题。因此，在设计单总线控制程序的过程中，尽可能的利用数据包长度资源，对状态位类参数传递进行字节整合等。与此同时，通过调试效果，选择较为合适的通信频率，保证无明显的通信延迟现象，又能稳定维持系统主任务运行。