栏目编辑 | 杨启森
yangqs@idnovo.com.cn

# CAN 总线波特率自适应程序设计

文 | 株洲时代电子技术有限公司 刘洁 曾海林

**摘 要**：本文提出了一种基于 CAN 网络控制平台的节点控制单元通讯波特率自适应方案，设计了主从机的通信协议和程序流程图，开发了基于该方案的软件程序，并通过在实验室搭建相应的硬件平台进行了测试，测试结果充分验证了该方案的切实可行性和实用性，为实现公司后期 CAN 总线波特率的在线修改提供了有力保障。

**关键词**：网络控制平台 ；CAN 总线；波特率；自适应；程序设计

## 一、引言

CAN 总线由于采用差分信号进行数据传输，有着极强的抗干扰能力，同时，由于它采用非破坏性的总线仲裁机制，使得即便同一时刻不同节点同时往 CAN 总线发送数据，也不会出现数据混乱和通信出错的情况，由于 CAN 总线的各种优点，使得其在欧美国家乃至全世界被广泛应用，尤其是在工业控制、工业仪表和汽车电子等领域，CAN 总线控制系统广泛应用于各类机械自动化设备中。

随着设备内各总成节点不断增加，与此同时接入到网络控制系统的的第三方设备与传感器也逐渐增加，出现了各种各样通信波特率，CAN 总线在不同波特率下允许的最大通讯距离是：10km （5Kbps）、6.7km （10Kbps）、3.3km （20Kbps）、1.3km （50Kbps）、620m （100Kbps）、530m（125Kbps）、270m（250Kbps）、130m（500Kbps）和 40m（1Mbps）。由于 CAN 总线通信时必须保持同种波特率，故接入的第三方控制系统中主控制器与从节点之间只能用同一波特率进行通信；由于某些第三方设备的波特率是无法修改的（如发动机 ECU 和变速箱 VTDC 等），故此时接入第三设备后，所有同一网络的从节点都需通过修改底层程序重新烧写程序才能实现波特率的修改，大大增加了工作量，如果能够开发一种在线修改波特率的程序，势必大大减少工作量，同时使得波特率的修改变得极为方便快捷。为此，本文提出了一种 CAN 总线波特率自适应程序设计的方案，设计了基于此方案的软件程序，通过在实验室调试，充分验证了该软件方案的可行性。

## 二、程序流程设计

目前的网络控制系统根据现场应用按照网络控制单元所处的网络位置，分为主控制单元与从控制单元，其中主控制单元相互之间以主 CAN 网络连接，以固定波特率通讯；从控制单元与从控制单元之间以设备 CAN 网络连接，通讯波特率采用自适应方式。现场应用示意图如图 1 所示。

图 1 中显示控制单元、主控制单元和从控制单元均为自



投稿邮箱 cadcam@idnovo.com.cn
IMCHINA@idnovo.com.cn

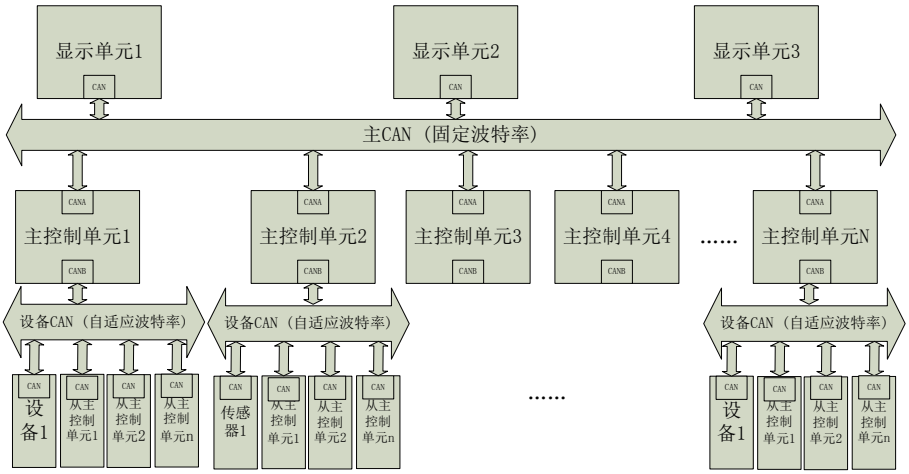



图 1 总线应用方案图

主研发的网络控制单元，设备与传感器均为标准 CAN 设备与传感器，整个系统构成一个网络控制平台。一个主控制单元或几个主控制单元配合设备 CAN 组成整系统中的一个分系统，例如，控制一组作业部件时，由一个或几个主控制单元根据相应算法控制多个从控制单元发生相应动作。此时控制系统的界限更加明显，如作业系统可以分为捣固、稳定和测量，加上走行和发动机控制，独立组成局域网，这样主 CAN 网络数据量也会相应减少。

在控制单元设计时选用具有两个独立 CAN 控制单元的 DSP 芯片，在以往的设计基础上丰富硬件功能，满足通讯需求。从控制单元只使用一个 CAN，原则上设备与传感器只能挂在设备 CAN 网络。

在保证原有程序正常运行的基础上，设计从控制单元具备自适应波特率调整功能，主控制单元有下发更改波特率的功能。当局域网络添加新的控制单元或者其他设备时由操作者在人机交互界面点击新增某局域网络设备，启动自适应匹配波特率程序。

### 1、显示器程序设计

图 2 是显示器启动波特率更改程序的基础流程图，省略了标志位设置以及程序溢出流程，基本的功能由命令下发与读取主控制单元收到命令返回值组成，命令的下发增加了主控制单元收到后的校验过程，如果主控制单元收到命令，会返回一个收到命令的状态，确保要修改子网络波特率的主控制单元已经收到修改命令。当主控制单元成功修改了其子网络波特率后会返回相应的响应码到显示器，此时，显示控制单元会记录下本主控制单元下子网络的最新的 CAN 通讯波特率。显示器波特率修改子程序流程图如图 2 所示。

### 2、主机程序设计

某个需要更改其子网络 CAN 通信波特率的主控制单元，首先会在 CANA 收到显示控制单元的命令，然后保持现有波特率通过 CANB 下发更改波特率的命令给从控制单元。图 3 为主控制单元波特率更改程序流程图，省略了标志位设置以及程序溢出流程。在此程序设计时没有使用从控制单元收到命令后的反馈，是因为在从控制单元设计时增加了通讯异常判断，所以只要在 200ms 的时间里发送改变波特率的命令即可。还需要特别指出的是在 CANB 波特率改变之前必须要屏蔽别的数据发送命令，使子网络仅仅只有专用校验码广播存在，提高网络可靠性。

主机外设寄存器初始化包括 CAN 寄存器的初始化，确保主机上电后 CAN 通讯波特率为默认可知值，主机收到上位机请求改变波特率命令后，会立即将修改命令下发到子网络的所有从机，然后等待从机修改波特率操作，此过程需要一定时间，只有等所有从机修改完成后，主机才开始以修改后的波特率下发特定校验码到从机，通过从机是否

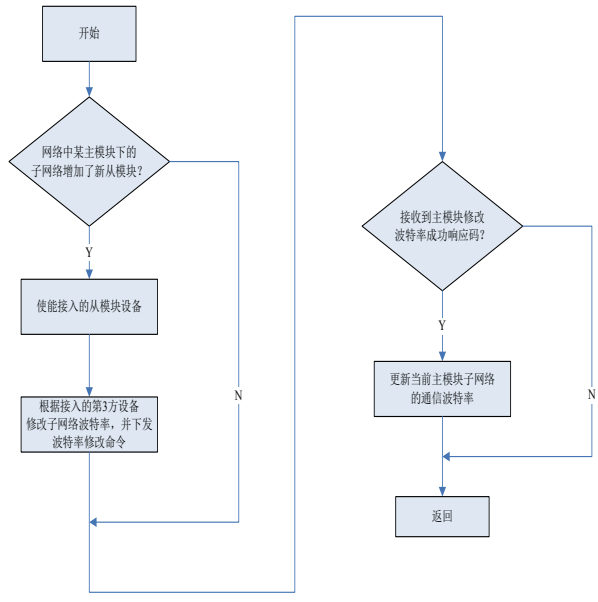



图 2 显示器启动波特率更改子程序流程图

收到特定检验码来判定从机修改波特率是否成功，当从机修改成功后，主机会收到相应的从机回应码，主机程序再进入正常的作业程序，进行正常的作业数据收发。主机程序流程图如图 3 所示。

### 3、从机程序设计

改变波特率的最终目的就是在子网络可以加挂不同频段的设备，因此在加入新设备时需要更改在局域网中所有从控制单元的波特率。图 4 为从控制单元波特率更改程序流程图，省略了标志位设置以及程序溢出流程。因此在程序设计时，从控制单元设计为三种状态都能够自动匹配波特率，一个是从控制单元出厂后未设定波特率，一个是通讯异常，另外一个是主控制单元启动改变波特率的程序，这三种状态均能进入波特率调整入口，这时波特率寄存器依次更改为 800K、500K 和 250K 等，重置 CAN 控制寄存器，主程序不断地读取网络数据，如果通讯正常或者专用校验码识别成功，说明波特率修改成功，保存波特率参数并返回主程序，主程序将以修改成功后的波特率与其他设备通讯。从机的波特率修改部分程序流程图如图 4 所示。

其中波特率修改函数的流程图如图 5 所示。

其中，MAX 一般定义为 80~150，特定校验码由主机与从机开发人员事先约定好，修改波特率依次从高到低，分别采取 800K、500K 和 250K 等分别进行试探，当波特率与主机不匹配达一定时间（由 MAX 定义），说明所有可能匹配的波特率已经进行试探，仍然收不到数据，说明通信已经出错，此时需修改初始化波特率为默认的 800K，以供重新上电恢复通信；当波特率与主机匹配时，从机才会收到来自主机的特定校验码，此时，从机需将波特率参数存储到 FLASH 区，下次上电则从 FLASH 读取出该波特率参数。

## 三、实验验证

### 1、实验平台搭建

根据测试需要，在实验室搭建了相应的测试平台，测试中选择一台主机和多台从机，如图 6 所示。

### 2、测试结果

现场试验先验证一主多从方式，此时开启主从正常通讯内容，使网络数据接近正常工作状态，再由主控制单元发送命令改变波特率，发现从控制单元收到命令不稳定，考虑实际应用时，一旦启动波特率更改命令，设备 CAN 网络上不

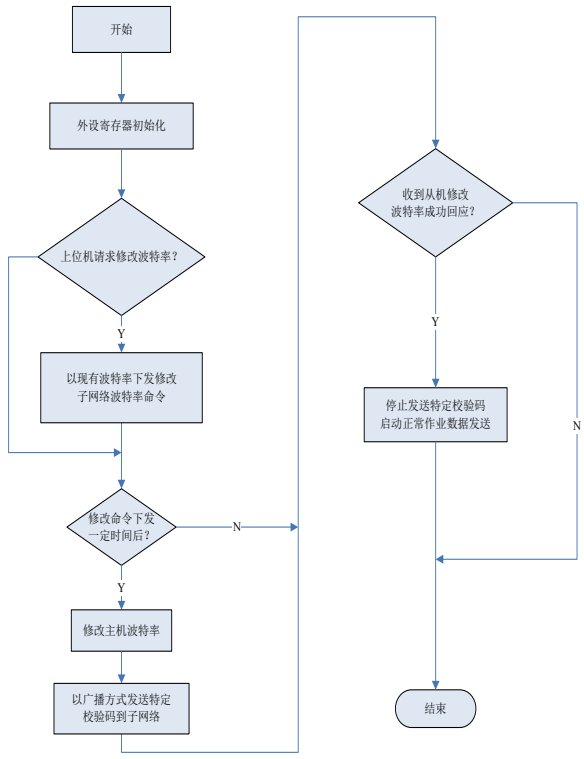



图 3 主机程序流程图

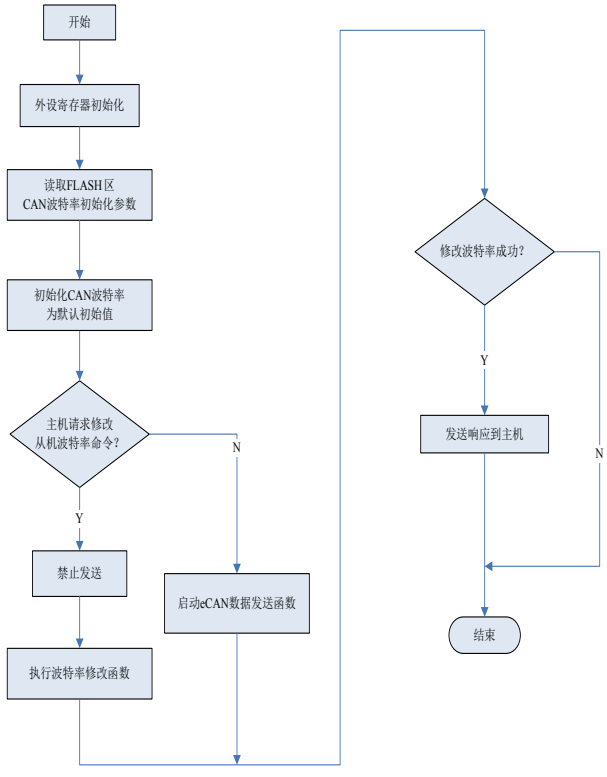



图 4 从机程序流程图


投稿邮箱 cadcam@idnovo.com.cn
IMCHINA@idnovo.com.cn

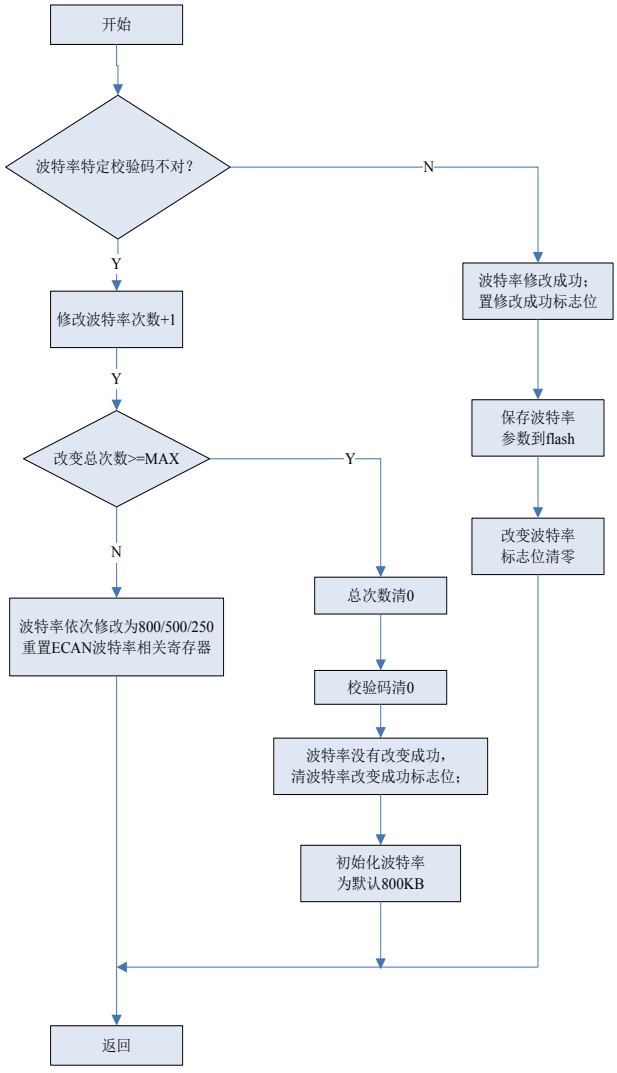



图 5 波特率修改程序流程图

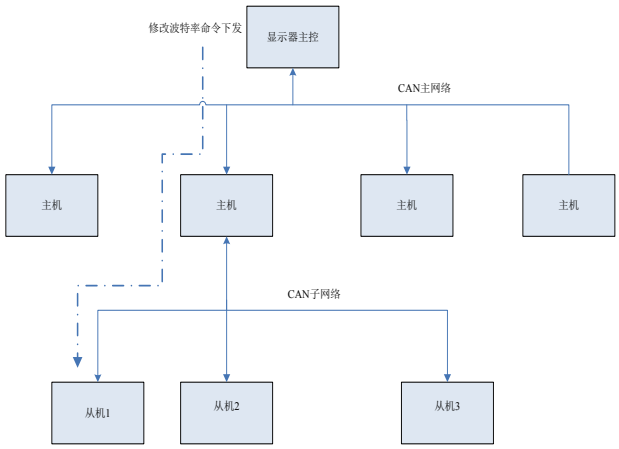



图 6 测试平台网络结构拓扑图

应该存在其他命令，因此对其他命令屏蔽，从控制单元能够正常响应。最终 2 个从控制单元的波特率调整为与主控制单元相同的波特率并通信正常。试验验证程序设计可以实现主控制单元下发命令与从控制单元的配合基本正常，对从控制单元的波特率设定也仅需要更改与波特率相关的寄存器即可完成。从控制单元收到更改命令后一些试验数据如图 7~图 13 所示。

Expressions | Breakpoints

| Expression | Type | Value |
|---|---|---|
| baud1 | unsigned int | 800 |
| CountData209 | int | 0 |
| CountData207 | int | 1500 |
| CountData208 | int | 22258 |
| Time1msCount | unsigned int | 37 |
| ChangeBaudFlagAdd | unsigned int | 0 |
| LoopforCnt | unsigned int | 40720 |
| Frame0Data | unsigned int[4] | 0x0000C060@Data |
| CanRateCheck | int | 0x0000 (Hex) |
| ChangeBaudFlag | int | 0 |
| ChaCanBaudOK | int | 0 |
| Add new expression | | |

图 7 从控制单元 0x209 波特率初始设定 800Kb 的一些试验数据

Expressions | Breakpoints

| Expression | Type | Value |
|---|---|---|
| baud1 | unsigned int | 500 |
| CountData209 | int | 0 |
| CountData207 | int | 14297 |
| CountData208 | int | 3024 |
| Time1msCount | unsigned int | 12 |
| ChangeBaudFlagAdd | unsigned int | 20 |
| LoopforCnt | unsigned int | 56312 |
| Frame0Data | unsigned int[4] | 0x0000C060@Data |
| CanRateCheck | int | 0x0000 (Hex) |
| ChangeBaudFlag | int | 0 |
| ChaCanBaudOK | int | 0 |
| Add new expression | | |

图 8 从控制单元 0x209 修改为 500Kb 的一些试验数据

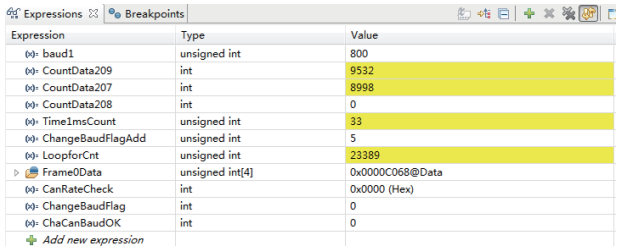
Expressions | Breakpoints

| Expression | Type | Value |
|---|---|---|
| baud1 | unsigned int | 250 |
| CountData209 | int | 0 |
| CountData207 | int | 3650 |
| CountData208 | int | 29183 |
| Time1msCount | unsigned int | 46 |
| ChangeBaudFlagAdd | unsigned int | 10 |
| LoopforCnt | unsigned int | 32907 |
| Frame0Data | unsigned int[4] | 0x0000C060@Data |
| CanRateCheck | int | 0x0000 (Hex) |
| ChangeBaudFlag | int | 0 |
| ChaCanBaudOK | int | 0 |
| Add new expression | | |

图 9 从控制单元 0x209 修改为 250Kb 的一些试验数据

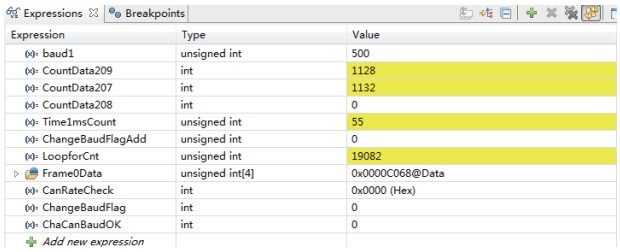
由图 7~9 可以看出，从控制单元 0x209 无论是在修改前的 800K 的波特率还是在修改为 500K 和 250K 都能正常收到了来自其他控制单元 0x207 和 0x208 的数据（变量 CountData207 和 CountData208 代表接收数据帧数），控制单元是收不到自身所发数据，故 CountData209 一直为 0，以上数据充分说明了波特率自适应调节是成功的。

栏目编辑 | 杨启森
yangqs@idnovo.com.cn



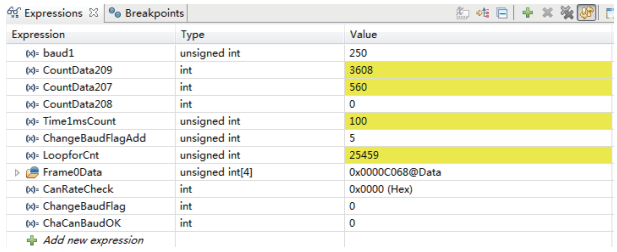
| Expression | Type | Value |
|---|---|---|
| baud1 | unsigned int | 800 |
| CountData209 | int | 9532 |
| CountData207 | int | 8998 |
| CountData208 | int | 0 |
| Time1msCount | unsigned int | 33 |
| ChangeBaudFlagAdd | unsigned int | 5 |
| LoopforCnt | unsigned int | 23389 |
| Frame0Data | unsigned int[4] | 0x0000C068@Data |
| CanRateCheck | int | 0x0000 (Hex) |
| ChangeBaudFlag | int | 0 |
| ChaCanBaudOK | int | 0 |
| Add new expression | | |

图 10 从控制单元 0x208 初始为 800Kb 收到数据



| Expression | Type | Value |
|---|---|---|
| baud1 | unsigned int | 500 |
| CountData209 | int | 1128 |
| CountData207 | int | 1132 |
| CountData208 | int | 0 |
| Time1msCount | unsigned int | 55 |
| ChangeBaudFlagAdd | unsigned int | 0 |
| LoopforCnt | unsigned int | 19082 |
| Frame0Data | unsigned int[4] | 0x0000C068@Data |
| CanRateCheck | int | 0x0000 (Hex) |
| ChangeBaudFlag | int | 0 |
| ChaCanBaudOK | int | 0 |
| Add new expression | | |

图 11 从控制单元 0x208 转化为 500Kb 收到数据



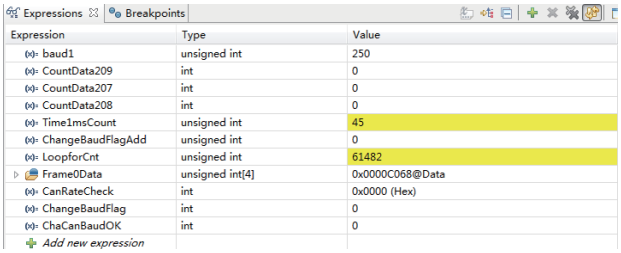
| Expression | Type | Value |
|---|---|---|
| baud1 | unsigned int | 250 |
| CountData209 | int | 3608 |
| CountData207 | int | 560 |
| CountData208 | int | 0 |
| Time1msCount | unsigned int | 100 |
| ChangeBaudFlagAdd | unsigned int | 5 |
| LoopforCnt | unsigned int | 25459 |
| Frame0Data | unsigned int[4] | 0x0000C068@Data |
| CanRateCheck | int | 0x0000 (Hex) |
| ChangeBaudFlag | int | 0 |
| ChaCanBaudOK | int | 0 |
| Add new expression | | |

图 12 从控制单元 0x208 转化为 250KB 收到数据

由图 10~12 可以看出，从控制单元 0x208 无论是在修改前的 800K 的波特率还是在修改为 500K 和 250K 都能正常收到了来自其他控制单元 0x209 和 0x207 的数据（变量 CountData209 和 CountData207 代表接收数据帧数），控制单元是收不到自身所发数据，故 CountData208 一直为 0。

以上数据充分说明了波特率自适应调节是成功的。

当主控制单元发送错误命令时，波特率不会调整，也不会收到数据，当主控制单元调整为系统默认的标准波特率（800K、500K 和 250K）时，从控制单元会再次正常收到数据。



| Expression | Type | Value |
|---|---|---|
| baud1 | unsigned int | 250 |
| CountData209 | int | 0 |
| CountData207 | int | 0 |
| CountData208 | int | 0 |
| Time1msCount | unsigned int | 45 |
| ChangeBaudFlagAdd | unsigned int | 0 |
| LoopforCnt | unsigned int | 61482 |
| Frame0Data | unsigned int[4] | 0x0000C068@Data |
| CanRateCheck | int | 0x0000 (Hex) |
| ChangeBaudFlag | int | 0 |
| ChaCanBaudOK | int | 0 |
| Add new expression | | |

图 13 从控制单元收到错误命令时收到数据为零

当主控制单元重新以正常波特率发送时，从控制单元又会自动调整到正确的波特率进行正常通讯，确保了通信的可靠和可恢复性。

## 四、结论

通过学习和实践，在保持原有主从机程序不变的基础上，通过实验验证波特率调整方案切实可行，并在基于 DSP 的控制单元上进行了一主一从和一主二从试验，在没有改变其他程序的基础上添加了波特率改变的程序，系统排除了多个不稳定因素，已经能够正常运行，并将在一主多从的局域网中广泛试验。

欢迎加粉《智能制造》杂志官方微博“智造”

智 造

成为“智造粉丝”，及时获取更多智能制造领域的有用、有趣信息。



投稿邮箱 cadcam@idnovo.com.cn
IMCHINA@idnovo.com.cn