CAN Bootloader 远程升级协议

版本V01

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 日期 | 作者 | 内容 |
| V00 | 2023-04-01 | Liu | 初稿 |
| V01 | 2023-04-21 | Liu | 增加CAN报文协议 |

1. Bootloader协议简介

本协议定义了下位机固件和上位机之间的通讯协议，通过上位机可以实现远程烧录下位机固件的用途，避免了拆机烧录所带来的不便。

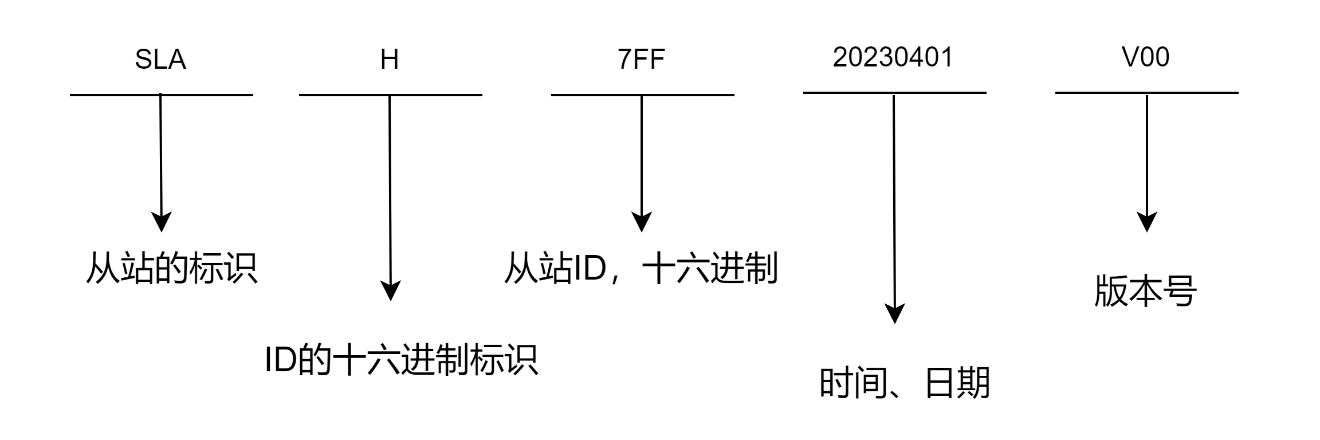
2. CAN通讯参数定义

Bootloader的使用CAN通讯实现，CAN通讯的参数定义如下：

|  |  |  |
| --- | --- | --- |
| 参数名称 | 定义 | 备注 |
| USB/CAN设备 | ZLG、PEAKCAN… | 选用主流的USB/CAN工具，上位机实现驱动加载 |
| CAN波特率 | 500Kbps |  |
| 帧类型 | 标准帧 | CANID范围：0x00-0x7FF |
| 采样点 | ≥80% |  |
| 上位机烧录ID | 0x701 | 上位机bootloader时用该ID发送数据 |
| 下位机烧录ID | 0x702 | 上位机bootloader时，下位机用该ID回复数据 |
| 广播帧ID | 0x700 | 该ID可被所有的CAN节点接收，实现数据的广播,如配置ID、关闭节点的自动发送等； |

3. 所支持的固件格式以及预处理信息

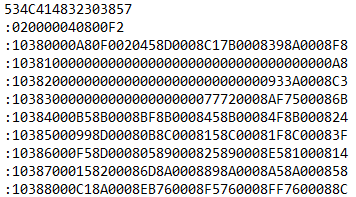
上位机支持.hex文件的加载、分析以及数据的发送。Hex文档命名时按照如下规则：



如，SLAH7F020230402V01，表示需要给从站ID为0x7F0的设备烧录，该固件为2023年4月1日所创建的V00版本。

因为所挂CAN节点众多，为防止hex文件张冠李戴，需要对Hex文档进行预处理，在hex文档中手动添加预处理信息，在远程烧录时可以对信息进行校验。实现方式如下：

在hex文档的头部加入8个ASIIC码，以作区分，内容为:SLAH**208**W，其中208为可变字符，表示从站的ID(十六进制)，其余字符固定。添加后的效果如下：



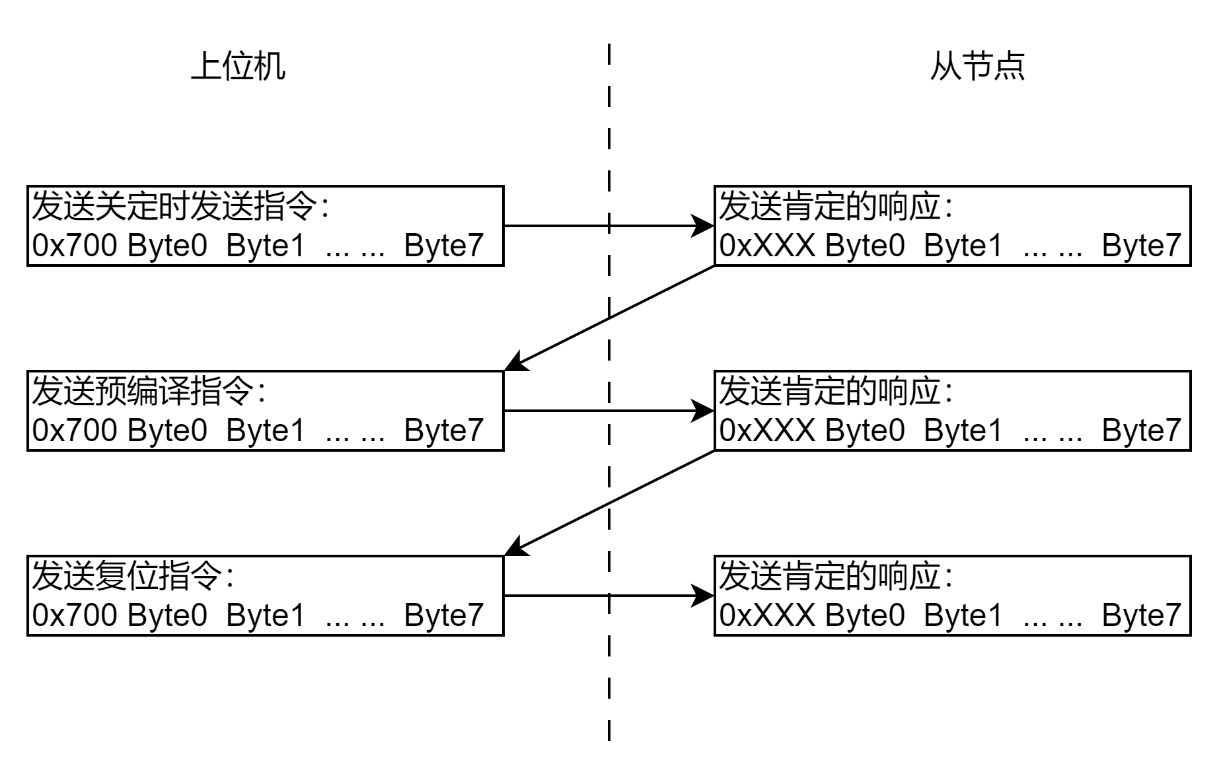
4. 远程烧录过程

远程烧录过程包含三个部分，1)预编程阶段；2)主编程阶段；3)后编程阶段。

4.1 预编程阶段

该阶段为后面的主编程提供条件，整个系统在正常工作时其他CAN节点可能存在定时发送、触发发送、诊断等会产生与远程烧录无关的数据，需要将其他节点“禁言”。

从上位机角度，预编程流程如下：



定时发送关闭指令如下表所示：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ①定时发送关闭指令 | | | | | | | | | |
|  | ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 上位机发送 | 0x700 | 0x42 | 0x44 | 0x43 | 0x01 | xx | xx | xx | xx |
| 广播ID | B | D | C | 0x01: 关闭发送  0x00: 打开发送 |  |  |  |  |
| 下位机回复 | 0xXXX | 0x42 | 0x44 | 0x52 | 0x01 | xx | xx | xx | xx |
| 从机ID | B | D | R | 0x01：关闭成功  0x00：关闭失败 |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ②发送预编译指令 | | | | | | | | | |
|  | ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 上位机发送 | 0x700 | 0x53 | 0x4C | 0x41 | 0x48 | 0X32 | 0X30 | 0X38 | 0X57 |
| 广播ID | S | L | A | H | 从节点ID, 例：0x208 | | | W |
| 下位机回复 | 0x208 | 0x53 | 0x4C | 0x41 | 0x52 | xx | xx | xx | xx |
| 从机ID | S | L | A | R |  |  |  |  |

注意，hex中填入分机ID时用十六进制，0x02, 0x00,0x08代表0x208。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ③发送从节点复位指令 | | | | | | | | | |
|  | ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 上位机发送 | 0x700 | 0X52 | 0X45 | 0X53 | 0X45 | 0X54 | xx | xx | xx |
| 广播ID | R | E | S | E | T |  |  |  |
| 下位机回复 | 0x208 | 0X52 | 0X45 | 0X53 | 0X45 | 0X54 | xx | xx | xx |
| 从机ID | R | E | S | E | T |  |  |  |

上位机发送复位指令后，下位机没回复。

4.2 主编程阶段

该阶段为输出传输的关键阶段，包含下位机接收数据-写RAM-写Flash的整个过程。 上位机以2k字节为一个Block传输数据，如果数据不足2K则需要用0xFF或者0x00填充至2K(用Flash默认数据)，为加快烧写速度下位机先通过RAM接收2K数据，Check-Sum校验通过后，再执行Flash的擦除-写入工作。

该阶段上位机所用的ID为0x701，下位机响应回复用的ID为0x702。该阶段包含四个过程：

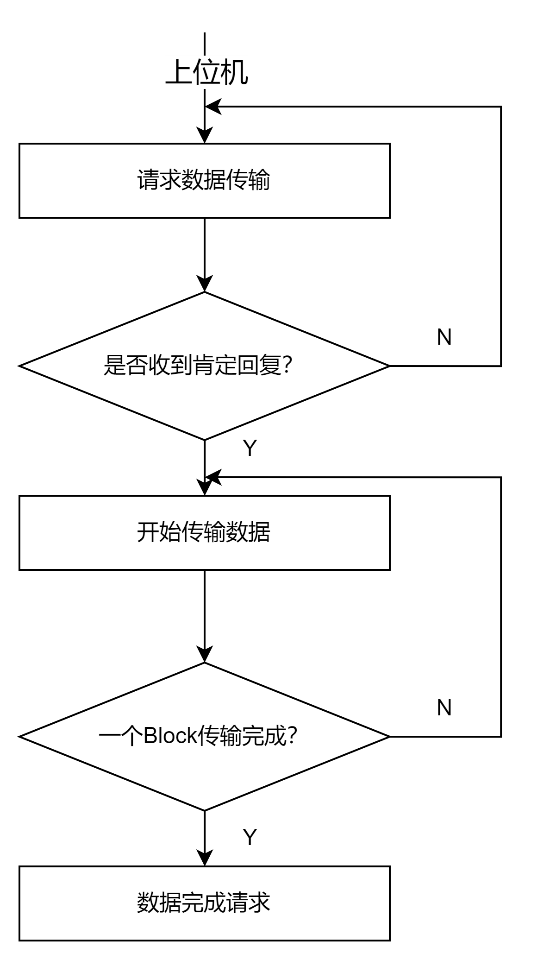
1)数据传输请求(以Block为单位)，

2)数据传输；

3)Block传输完成请求；

4)复位跳转；

当存在多个Block时，数据传输过程重复以上1)-3)步骤。单个Block数据传输流程如下：



4.2.1数据传输请求

请求传输过程，上位机CAN报文中REQ为固定数据，第3、4字节为当前所要传输的Block序号，从0开始，下位机回复的CAN报文中，RDY为固定数据，第5、6字节为下位机RAM所能存储的数据容量，上位机中一个Block的大小由该两个字节决定。本例中为2k字节，所以下位机固定发送0x800。 上位机发送数据时，一个Block内的帧时间间隔由Byte7决定，本例为3ms。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ①数据传输请求 | | | | | | | | | |
|  | ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 上位机发送 | 0x701 | 0X52 | 0X45 | 51 | 0XXX | 0XXX | xx | xx | xx |
| 上位机ID | R | E | Q | Block的序号，从0开始 | |  |  |  |
| 下位机回复 | 0x702 | 0X52 | 0X44 | 0X59 | 0XXX | 0XXX | 0x08 | 0x00 | 0x03 |
| 下位机回复ID | R | D | Y | Block的序号，从0开始 | | 下位机一次能接收的数据 | | 上位机帧间隔3ms |

下位机回复后，上位机开始数据传输。

4.2.2数据传输

上位机传输数据时，一帧CAN报文的八个字节中，前两个字节为帧计数器，计数器范围从0-342，每帧包含6个有效数据。

下位机对上位机的每个数据帧进行回复，前两个字节为帧计数器，后两个字节为**有效数据**的Check-Sum校验和。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ②主编程数据格式 | | | | | | | | | |
|  | ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 上位机发送 | 0x701 | 0x00 | 0x00 | D1 | D2 | D3 | D4 | D5 | D6 |
| … |  |  |  |  |  |  |  |
| 0x01 | 0x55 | D2041 | D2042 | D2043 | D2044 | D2045 | D2046 |
| 0X01 | 0X56 | D2047 | D2048 | xx | xx | xx | xx |
| 下位机回复 | 0x702 | 0x00 | 0x00 | xx | xx | xx | xx | 第1帧Check-Sum | |
| … |  | xx | xx | xx | xx | 第N帧Check-Sum | |
| 0x01 | 0x55 | xx | xx | xx | xx | 第341帧Check-Sum | |
| 0X01 | 0X56 | xx | xx | xx | xx | 第342帧Check-Sum | |

4.2.3数据传输完成请求

一个Block传输完成，上位机会发送Block完成请求，同时下位机会将2K的数据写入Flash。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ③数据传输完成请求 | | | | | | | | | |
|  | ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 上位机发送 | 0x701 | 0X44 | 0X4F | 0X45 | 0XXX | 0XXX | xx | xx | xx |
| 上位机ID | D | O | E | Block的序号，从0开始 | |  |  |  |
| 下位机回复 | 0x702 | 0X44 | 0X4F | 0X45 | 0XXX | 0XXX | Byte5 | Byte6 | Byte7 |
| 下位机回复ID | D | O | E | Block的序号，从0开始 | | xx | xx | xx |

注意，上位机发送完成请求后，下位机开始写Flash，写Flash完成后才会回复数据，由于擦除-写入Flash需要耗费大约100ms左右的时间，所以上位机需要等待较长的时间。

4.2.4 数据传输完成

当所有的Block都传输完成后，上位机通知下位机，下位机收到指令后，将backup区的数据会复制到APP区。

上位机发送的命令如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 上位机发送复位指令 | | | | | | | | | |
|  | ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 上位机发送 | 0x701 | 0x43 | 0x4F | 0x4D | 0x50 | 0X4C | 0X45 | 0X54 | 0X4C |
| 广播ID | C | O | M | P | L | E | T | E |
| 下位机回复 | 0x702 | 0x43 | 0x4F | 0x4D | 0x50 | 0X4C | 0X45 | 0X54 | 0X4C |
| 从机ID | C | O | M | P | L | E | T | E |

该过程可能耗时较长，上位机需要留够等待时间。

4.2.5 复位跳转

一个Block传输完成后，重复以上步骤传输剩余的Block。当所有Block传输完成后，上位机发送跳转指令，下位机复位完成跳转。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 上位机发送复位指令 | | | | | | | | | |
|  | ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 上位机发送 | 0x701 | 0x47 | 0x4F | 0x54 | 0x4F | 0X41 | 0X50 | 0X50 |  |
| 广播ID | G | O | T | O | A | P | P |  |
| 下位机回复 | 0x702 | 0x47 | 0x4F | 0x54 | 0x4F | 0X41 | 0X50 | 0X50 |  |
| 从机ID | G | O | T | O | A | P | P |  |

4.3 后编程阶段

后编程阶段是预编程阶段的逆过程，把预编程阶段所禁掉的自动发送重新打开，恢复其他节点的数据发送。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 定时发送打开指令 | | | | | | | | | |
|  | ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 上位机发送 | 0x700 | 0x42 | 0x44 | 0x43 | 0x00 | xx | xx | xx | xx |
| 广播ID | B | D | C | 0x01: 关闭发送  0x00: 打开发送 |  |  |  |  |
| 下位机回复 | 0xXXX | 0x42 | 0x44 | 0x52 | 0x01 | xx | xx | xx | xx |
| 从机ID | B | D | R | 0x01：关闭成功  0x00：关闭失败 |  |  |  |  |

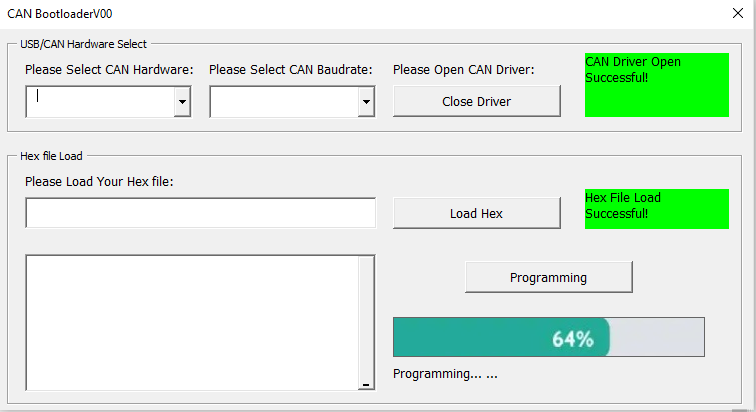
至此，CAN Bootloader远程烧录数据完成。

5. 上位机GUI建议

本章节内容均为建议，实际以编程方便为准。

GUI包含两个Group区域，USB/CAN硬件工具选择区、Hex文件选择区，每个区域的内容都有提示信息。

界面参考如下：



Hardware Selection、CAN Baudrate区域为下拉框，Open/Close Driver按钮互锁，CAN driver打开成功与否在右侧可以提示，成功用绿色背景，失败用红色背景。

Hex file Load区域可以加载hex文件和烧录，并显示实时的CAN报文和进度。 “Load Hex“按钮被按下后选择本地的hex文件，文件加载成功后左侧的编辑框内显示地址，右侧的信息提示区显示加载情况，加载成功用绿色背景，失败用红色背景。

“Programming“按钮被按下后开始烧录过程，烧录时”Programming“按钮、”Load Hex“按钮、”Close Driver“按钮Disable，并在左侧文本框内实时显示带时间戳的CAN报文，进度条实时刷新烧录进度，下方的提示区显示烧录所用时间。

烧录完成后，所有按钮恢复Enable状态。

1. CAN报文通讯协议

本部分定义了CAN报文的协议、每个byte、每个bit的含义。基板的CAN报文采用定时发送的方式将信息传送给主站，暂定发送周期为100ms。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 字节  含义 |  |  |  |  |  |  |  | CheckSum |
| 含义  描述 |  |  |  |  |  |  |  | 前七个字节之和，保留低八位 |