# 对具体指令的测试

### 1，初始信息配置指令

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（1Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 终端号（1Byte） | 前端相邻终端号小，大（2Byte） |
| 55AA | 0D | xx | xx | F0 | xx | xx xx |
| 后端相邻终端号小，大（2Byte） | 保留位 | Sum（1Byte） |  |  |  |  |
| xx xx | XX | xx |  |  |  |  |

|  |  |
| --- | --- |
| 前前端，前端，本端，后端 ，后后端 |  |
| 00 00 01 03 05 | 55AA0DFF01F001000003050005 |
| 01 03 05 07 09 | 55AA0DFF01F005010307090015 |
| 03 05 07 09 0B | 55AA0DFF01F0070305090B001F |
|  | 55AA0CFF03F0010003010002 |
| 00 01 03 05 07 | 55AA0DFF01F00300010507000C |
| 55AA0DFF03F00100000305000C07 | 55AA0DFF01F00300010507000C |
| 发给01的初始信息配置指令  02 03 05 01 00  读取指令 ：55AA08FF03F103FD | 55AA0CFF01F0030205010006 |
| 发给03端的初始信息配置指令  05 03 08 01 00  读取指令 ：55AA08FF05F10501 | 55AA0CFF03F005030801000E |
| 发给05端的初始信息配置指令  01 00 03 01 00  读取指令 ：55AA08FF01F101F9 | 55AA0CFF05F0010003010004 |
| 发给03端的初始信息配置指令  03 01 08 00 00  读取指令 ：55AA08FF03F103FD | 55AA0CFF03F0030108000009 |
| 发给03端的初始信息配置指令  03 01 05 00 00  读取指令 ：55AA08FF03F103FD | 55AA0CFF03F0030105000006 |
| 发给03端的初始信息配置指令  03 01 05 01 00  读取指令 ：55AA08FF03F103FD | 55AA0CFF03F0030105010007 |
| 发给03端的初始信息配置指令  03 00 01 05 07 | 55AA0CFF03F003000105070D |

### **2，读取单点配置信息指令**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（1Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 终端号（1Byte） | Sum（1Byte） |
| 55AA | 08 | xx | xx | F1 | xx | xx |

|  |  |  |
| --- | --- | --- |
|  | 三端地址：03 08 09 | 三端地址：01 03 05 |
| 发给接触点的读取单点配置信息指令 | 55AA08FF03F103FD | 55AA08FF01F101F9 |
| 发给第二终端的读取单点配置信息指令 | 55AA08FF08F10807 | 55AA08FF03F103FD |
| 发给第三终端的读取单点配置信息指令 | 55AA08FF09F10909 | 55AA08FF05F10501 |
| 发给第四终端的读取单点配置信息指令 | 55AA08FF07F10705 | |
| 发给第五终端的读取单点配置信息指令 | 55AA08FF09F10909 | |
|  |  | |
| 由5端发给3端的 | 55AA08FF03F103FD | |
| 由7端发给5端 | 55AA08FF05F10501 | |
| 由3端发给1端的 | 55AA08FF01F101F9 | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（1Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 起点地址（1Byte） | 年（2Byte） |
| 55AA | 0A | xx | FF | 52 | xx | xx |
| 月  （1Byte） | 日  （1Byte） | 时  （1Byte） | 分 （1Byte） | 秒 （1Byte） | Sum（1Byte） |
| xx | xx | xx | xx | xx | xx |

### **3,实时时钟配置指令**

|  |  |
| --- | --- |
| 发给第一端的实时时钟配置指令  2020年10月10日16点48分5秒 | 55AA0FFF01520107E40A0A103005 |
| 发给第二端的实时时钟配置指令  2020年10月10日16点48分5秒 | 55AA0FFF03520307E40A0A103005 |
| 发给第一二端的实时时钟配置指令  2020年10月10日16点48分5秒 | 55AA0FFF03520107E40A0A103005 |
| 发给第三端的实时时钟配置指令  2020年10月10日16点48分5秒 | 55AA0FFF05520507E40A0A103005 |
| 发给1,2,3端的实时时钟配置  2020年10月10日16点48分5秒 | 55AA0FFF05520107E40A0A103005 |
| 发给1,2,3,4端的实时时钟配置  2020年10月10日16点48分5秒 | 55AA0FFF07520107E40A0A103005 |
| 发给1,2,3,4端的实时时钟配置  2020年10月10日0点0分0秒 | 55AA0FFF07520107E40A0A000000 |
| 发给1,2,3,4,5,6端的实时时钟配置  2020年10月10日0点0分0秒 | 55AA0FFF0B520107E40A0A000000 |
| 发给1,2,3,4,5,6,7端的实时时钟配置  2020年10月10日0点0分0秒 | 55AA0FFF0D5201140A0A00000095 |
| 发给1,2,3,4,5,6,7端的实时时钟配置  2020年10月10日0点0分0秒 | 55AA0FFF0D5201140A0A000000 |
| 发给5端的实时时钟配置  2020年10月10日0点0分0秒 | 55AA0EFF055205140A0A00000090 |
| 发给一端的实时时钟配置指令  16年11月23日15:17：00 | 55AA0EFF015201100B170F1100B2 |
| 发给5端的实时时钟配置指令  16年11月23日00:00：00 | 55AA0EFF055205100B170000009A |
| 发给5端和7端的实时时钟配置指令  16年11月23日00:00：30 | 55AA0EFF075205100B1700001EBA |
| 发给3端的实时时钟配置  16年11月23日00:00：45 | 55AA0EFF035203100B1700002DC3 |
| 发给7端的实时时钟配置  16年11月23日00:00：30 | 55AA0EFF075207100B1700001EBC |
| 发给1端的实时时钟配置指令  16年11月23日00：00:15 | 55AA0EFF015201100B1700000FA1 |

### **5，获取某段铁轨信息指令**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（1Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 起点地址（1Byte） | 内容（1Byte） |
| 55AA | 09 | xx | xx | 55 | xx | 00 |
| Sum（1Byte） |  |  |  |  |  |  |
| xx |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  | 三端地址：03 08 09 | 三端地址：01 03 05 |
| 发给接触点的 | 55AA09FF0355030062 | 55AA09FF015501005E |
| 发给第二终端的 | 55AA09FF085508006C | 55AA09FF0355030062 |
| 发给接触点和第二终端的 | 55AA09FF0855030067 | 55AA09FF0355010060 |
| 发给第二终端和第三终端的 | 55AA09FF095508006D | 55AA09FF0555030064 |
| 发给接触点到第三终端的 | 55AA09FF0955030068 | 55AA09FF0555010062 |
| 发给第三终端的 | 55AA09FF095509006E | 55AA09FF0555050066 |
| 发送给第四终端 |  | 55AA09FF075507006a |
| 发给接触点到第四终端的 |  | 55AA09FF0755010064 |
| 发给接触点到第六终端的 |  | 55AA09FF0B55010068 |
| 由五端到1端的 |  | 55AA09FF0155050062 |
| 由5端开始，多播3端和一端 |  | 55AA09FF0155030060 |
| 由5端开始，多播3端和一端 |  |  |
| 由五端开始，发给7端 |  | 55AA09FF0755050068 |
| 由3端开始，多播3端 和五端 |  | 55AA09FF0555030064 |

### **6,获取单点铁轨信息指令**

|  |  |  |
| --- | --- | --- |
|  | 三端地址 01 03 05 |  |
| 发给接触点的获取单点铁轨信息指令 | 55AA09FF01F50000FD |  |
| 发给第二终端的获取单点铁轨信息指令 | 55AA09FF03F50000FF |  |
| 发给第三终端的获取单点铁轨信息指令 | 55AA09FF05F5000001 |  |
| 发给第四终端的获取单点铁轨信息指令 | 55AA09FF07F5000003 |  |

### **7,超声信号发射指令**

|  |  |  |
| --- | --- | --- |
|  | 三端地址 01 03 05 |  |
|  | 55AA070301F3FD |  |

### **10，门限设置指令**

指令内容：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（1Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 内容（1Byte） | 内容（1Byte） | Sum（1Byte） |
| 55AA | 09 | xx | xx | F2 | xx | xx | xx |

|  |  |  |
| --- | --- | --- |
|  | 三端地址 01 03 05 | 三端地址 01 03 05 |
| 发给第一端的门限设置指令 | 55AA09FF01F20000FA | 55AA09FF01F26464C2 |
| 发给第二端的门限设置指令 | 55AA09FF03F20000FC | 55AA09FF03F26464C4 |
| 发给第三端的门限设置指令 | 55AA09FF05F20000FE | 55AA09FF05F26464C6 |
| 发给第四端的门限设置指令 | 55AA09FF07F2000000 | 55AA09FF07F26464C8 |
| 发给第五端的门限设置指令 | 55AA09FF09F2000002 | 55AA09FF09F26464CA |

### **11，擦除flash指令**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（1Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 起点地址（1Byte） | 内容  （2Byte） |
| 55AA | 0a | xx | xx | 56 | xx |  |
| 内容（2Byte） | Sum（1Byte） |  |  |  |  |  |
| xx | xx |  |  |  |  |  |

该指令可以用上位机或者手持盒发出。

起始地址之后的内容是2字节，第一字节是擦除的起始扇区号，第二字节是擦除的终点扇区

号。Flash一共有4096个子扇区，可以一次性擦除16个子扇区。

两个参数的取值范围是0~4095，如果第一个参数是0，第二个参数是4095就会触发全片擦除。

目前内容的位宽只能最大擦除256个扇区，感觉已经够用了。

|  |  |  |
| --- | --- | --- |
|  | 擦除第298到第4008扇区 | 擦除第0扇区 |
| 发给接触点的 | 55AA0CFF015601012A0FA844 | 55AA0CFF0156010000000062 |
| 发给第二终端的 |  | 55AA0AFF035603000064 |
| 发给接触点和第二终端的 |  | 55AA0AFF035601000062 |
| 发给第二终端和第三终端的 |  |  |
| 发给接触点到第三终端的 |  |  |
| 发给第三终端的 |  |  |
| 发送给第四终端 |  |  |
| 发给接触点到第四终端的 |  |  |
| 发给接触点到第六终端的 |  |  |
| 发给第0端的 |  | 55AA0CFF0056000000000060 |

### 14，擦除备份寄存器指令

注意：

内容的范围只能是0到17中的偶数。有9个备份寄存器。每个备份寄存器的地址分别是：

0,2,4,6,8,10,12,14,16.

该指令会擦除地址范围内的所有备份寄存器。擦除方式是向备份寄存器写0XFF。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（1Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 起点地址（1Byte） | 内容  （1Byte） |
| 55AA | 0a | xx | xx | 57 | xx |  |
| 内容（1Byte） | Sum（1Byte） |  |  |  |  |  |
| xx | xx |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | 擦除第0到16备份寄存器 |
| 发给接触点的 | 55AA0CFF015601012A0FA844 | 55AA0AFF015701001071 |
| 发给第二终端的 |  | 55AA0AFF035703001075 |
| 发给接触点和第二终端的 |  |  |
| 发给第二终端和第三终端的 |  |  |
| 发给接触点到第三终端的 |  |  |
| 发给第三终端的 |  | 55AA0AFF055705001079 |
| 发送给第四终端 |  | 55AA0AFF07570700107D |

### **12，获取flash里的铁轨信息指令**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（1Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 内容（6Byte） | 内容（6Byte） |
| 55AA | 13 | xx | xx | F4 | xx | xx |
| Sum（1Byte） |  |  |  |  |  |  |
| xx |  |  |  |  |  |  |

12字节内容代表时间，该指令的含义是获取从最近的X年X月X日X时X分X秒到

X年X月X日X时X分X秒的FLASH里存储的信息。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 最近2小时到5小时 | | 最近一小时 |
| 发给第一端的指令 | 55AA09FF01F4020503 | | 55AA09FF01F40001FD |
| 发给第二端的指令 | 55AA09FF03F4020505 | | 55AA09FF03F40001FF |
| 发给第三端的指令 |  | | 55AA09FF05F4000101 |
| 发给第四端的指令 |  | | 55AA09FF07F4000103 |
| 发给第五端的指令 |  | |  |
|  |  | |  |
| 下面的指令就是发给第一终端的，  修改宏定义让程序每隔75秒写一次，每次写2016个字节  在开始之前要擦除备份区，并且配置实时时钟为：16年11月23日15:17：00  大概过了30秒按下CPU复位，这样CPU向flash存储数据的时间和地址就是确定的 | | | |
|  | | 首先要在CPU上电之后，存第一包数据之前发一个指令，以便检验上位机要数据而FLASH中没有数据的情况下CPU会怎样反应  T2：15：15：00 T1：15：14：00  55AA13FF01F4100B170F0E00100B170F0F00A5 | |
| 首先是第0扇区从地址0写到2016  写入时间 2016\_11\_23\_15:17:40 | | 发一个T2是15:18:00，  T1是15:17：42  55AA13FF01F4100B170F112A100B170F1200D5  T1是15:17：40  T1是15:17：38  T1是15:16：25  T1是15:16：20  T1是13:00：00  发一个T2是15:17:43，  T1是15:17：42  55AA13FF01F4100B170F112A100B170F112BFF  T1是15:17：40  T1是15:17：38  T1是15:16：25  T1是15:16：20  T1是13:00：00  发一个T2是15:17:40，  T1是15:17：42  55AA13FF01F4100B170F112A100B170F1128FC  T1是15:17：40  T1是15:17：38  T1是15:16：25  T1是15:16：20  T1是13:00：00  发一个T2是15:17:38，  T1是15:17：42  55AA13FF01F4100B170F112A100B170F1126FA  T1是15:17：40  T1是15:17：38  T1是15:16：25  T1是15:16：20  T1是13:00：00  发一个T2是13:28:00，  T1是15:17：42  55AA13FF01F4100B170F112A100B170D1C00DD  T1是15:17：40  T1是15:17：38  T1是15:16：25  T1是15:16：20  T1是13:00：00 | |
| 然后是第0扇区从地址2016写到4032  写入时间 2016\_11\_23\_15:18:55 | | 发一个T2是15:20:00，  T1是15:18：57  55AA13FF01F4100B170F1239100B170F1400E7  发一个T2是15:18:57，  T1是15:18：57  55AA13FF01F4100B170F1239100B170F12391E  发一个T2是15:18:55，  T1是15:18：57  55AA13FF01F4100B170F1239100B170F12371C  发一个T2是15:18:50  T1是15:18：57  55AA13FF01F4100B170F1239100B170F123217  发一个T2是15:17:45，  T1是15:18：57  55AA13FF01F4100B170F1239100B170F112D11  发一个T2是15:17:40，  T1是15:18：57  55AA13FF01F4100B170F1239100B170F11280C  发一个T2是15:17:35，  T1是15:18：57  55AA13FF01F4100B170F1239100B170F112307  发一个T2是13:28:00，  T1是15:18：57  55AA13FF01F4100B170F1239100B170D1C00ED | |
| 然后是第1扇区从地址4096写到6112  写入时间 2016\_11\_23\_15:20:10 | | 发一个T2是15:20:10，  T1是15:20:10,  55AA13FF01F4100B170F140A100B170F140AC4  发一个T2是15:20:00，  从上一个表格中找T2是15:18:55  ......... | |
| 然后是第1扇区从地址6112写到8128  写入时间 2016\_11\_23\_15:21:25 | |  | |
| 然后是第2扇区从地址8192写到10208  写入时间 2016\_11\_23\_15:22:40 | |  | |
| 然后是第2扇区从地址10208写到12224  写入时间 2016\_11\_23\_15:23:55 | |  | |
| 然后是第3扇区从地址12288写到14304  写入时间 2016\_11\_23\_15:25:10 | |  | |
| 然后是第3扇区从地址14304写到16320  写入时间 2016\_11\_23\_15:26:25 | |  | |
| 然后第0扇区+2016  写入时间 2016\_11\_23\_15:27:40 | | 发一个T2是15:28:00，  T1是15:20:10,  55AA13FF01F4100B170F140A100B170F1C00C2  发一个T2是15:27:40，  T1是15:20:10,  55AA13FF01F4100B170F140A100B170F1B28E9  发一个T2是15:27:38，  T1是15:20:10,  55AA13FF01F4100B170F140A100B170F1B26E7  发一个T2是15:26:28，  T1是15:20:10,  55AA13FF01F4100B170F140A100B170F1A1CDC  发一个T2是15:26:25，  T1是15:20:10,  55AA13FF01F4100B170F140A100B170F1A19D9  发一个T2是15:26:23，  T1是15:20:10,  55AA13FF01F4100B170F140A100B170F1A17D7 | |
| 然后第0扇区+4032  写入时间 2016\_11\_23\_15:28:55 | |  | |
| 然后第1扇区+2016  写入时间 2016\_11\_23\_15:30:10 | |  | |
| 然后第1扇区+4032  写入时间 2016\_11\_23\_15:31:25 | | 发一个T2是15:31:30，  T1是15:20:10  55AA13FF01F4100B170F140A100B170F1F1EE3  发一个T2是15:31:25，  T1是15:20:10  55AA13FF01F4100B170F140A100B170F1F19DE  发一个T2是15:31:20，  T1是15:20:10  55AA13FF01F4100B170F140A100B170F1F14D9  发一个T2是15:30:11，  T1是15:20:10  55AA13FF01F4100B170F140A100B170F1E0BCF  发一个T2是15:30:10，  T1是15:20:10  55AA13FF01F4100B170F140A100B170F1E0ACE  发一个T2是15:30:8，  T1是15:20:10  55AA13FF01F4100B170F140A100B170F1E08CC | |
| 然后第1扇区+2016  写入时间 2016\_11\_23\_15:32:40 | |  | |
| 然后第2扇区+4032  写入时间 2016\_11\_23\_15:33:55 | |  | |
| 然后第3扇区+2016  写入时间 2016\_11\_23\_15:35:10 | |  | |
| 然后第3扇区+4032  写入时间 2016\_11\_23\_15:36:25 | |  | |
|  | |  | |

### **详细测试**

|  |  |
| --- | --- |
| 下面的指令就是发给第一终端的，  修改宏定义让程序每隔75\*4秒写一次，每次写252\*4个字节  在开始之前要擦除备份区，并且配置实时时钟为：16年11月23日15:17：00  大概过了10秒按下CPU复位，这样CPU向flash存储数据的时间和地址就是确定的 | |
| 15:17:40 从0写到1008 | T2是15:17:45,  T1是15:17:50  55AA13FF01F4100B170F1132100B170F112D09  T1是15:17:48  55AA13FF01F4100B170F1130100B170F112D07  T1是15:17:37  55AA13FF01F4100B170F1125100B170F112DFC  T1是15:12:40  55AA13FF01F4100B170F0C00100B170F112DD2  T1是15:16:25  55AA13FF01F4100B170F1019100B170F112DEF  T1是15:16:20  55AA13FF01F4100B170F1014100B170F112DEA  T2是15:17:40  T1是15:10:40  T2是15:17:30  T1是15:15:20  55AA13FF01F4100B170F0F14100B170F111EDA  T2是15:12:40  T2是15:10:00  T1是15:8:20  55AA13FF01F4100B170F0800100B170F0A009A |
| 15:22:39, 从1008写到2016 |  |
| 15:27:40，从2016写到3024 |  |
| ,15:32:40，从3024写到4032 | T2是15:32:40  T1是15:27:40  55AA13FF01F4100B170F1B28100B170F202813  T1是15:22:40  55AA13FF01F4100B170F1628100B170F20280E  T1是15:17:40  55AA13FF01F4100B170F1100100B170F2028E1  T1是15:12:40  55AA13FF01F4100B170F0C28100B170F202804 |
| 15:37:40，从4096写到5104 |  |
| 15:42:40，从5104写到6112 | T2是15:42:40  T1是15:22:40  55AA13FF01F4100B170F1628100B170F2A2818  T1是15:16:40  55AA13FF01F4100B170F1028100B170F2A2812  T1是15:11:38  55AA13FF01F4100B170F0B26100B170F2A280B  T1是15:00:00  55AA13FF01F4100B170F0000100B170F2A28DA |
| 15:47:40，从6112写到7120 | T2是15:47:40  T1是15:27:40  55AA13FF01F4100B170F1B28100B170F2F2822  T1是15:22:40  55AA13FF01F4100B170F1628100B170F2F281D  T2是15:47:48  T1是15:17:59  55AA13FF01F4100B170F113B100B170F2F3033 |
| 15:57：40，从8192写到了9200 | T2是15:57:45  T1是15:57:40  55AA13FF01F4100B170F3928100B170F392D4F  T1是15:57:39  55AA13FF01F4100B170F3927100B170F392D4E  T1是15:27:28 |
| 16:12:40，从11216写到12224 | T2是16:12:40  T1是16:12:40  55AA13FF01F4100B17100C28100B17100C28F2 |
| 16:32:39，从15312写到16320 | T2是16:32:39  T1是16:10:39  55AA13FF01F4100B17100A27100B1710202702  T1是15:30:20  55AA13FF01F4100B170F1E14100B1710202702  T1是15:32:40  55AA13FF01F4100B170F2028100B1710202718  T1是15:25:38  55AA13FF01F4100B170F1926100B171020270F |
| 16:37:40，从0写到1008 | T2是16:37:40  T1是16:32:40  55AA13FF01F4100B17102028100B171025281F  T1是16:32:35  55AA13FF01F4100B17102023100B171025281A  T1是16:32:43  55AA13FF01F4100B1710202B100B1710252822  T1是15:27:40  55AA13FF01F4100B170F1B28100B1710252819  T1是16:27:40  55AA13FF01F4100B17101B28100B171025281A |
|  |  |
| 21:45:09 ,从2016写到3024 | T2是 |
|  |  |
| 16年11月25日 |  |
| 15:43:44，从3024写到4032 |  |
| 15：48：55 ,从4096写到5104 |  |
| 16:0:09，从5104写到6112 | T2是16:0:28  T1是16:0:05  55AA13FF01F4100B19100005100B1910001CAF |
| 16:05:10，从6112写到7120 | T2是16:05:15  T1是16:05:13  55AA13FF01F4100B1910050D100B1910050FB4  T1是16:00:05  55AA13FF01F4100B19100005100B1910050FA7  T1是16年，11月23日，16:12:40  55AA13FF01F4100B17100C28100B1910050FD4 |
|  |  |
| 12月1日 4:00 到12月2号12:32(发给3端的！) | 55AA13FF03F4100C01040000100C020C200073 |
|  |  |

### **13立即获取未知终端基本信息指令**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（1Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | Sum（1Byte） |  |
| 55AA | 09 | FF | FF | F6 | xx |  |

|  |
| --- |
| 55AA07FFFFF6FA |

**立即响应失败的处理**

从程序的角度来说：处于WaitImeAck状态时在该状态下等待立即响应，如果规定时间内等到了就跳转下一状态，如果没等到就把行动步骤加5（这样状态跳转就指向了备用的行动步骤）。

备用的行动步骤只处理一个问题：相邻终端失联怎么办（相邻终端没有立即响应即认为失联，目前只要一次没有立即响应即认为失联，以后可以设置重发机制，重发机制也设在备用行动步骤里

另一种情况是相邻终端在回应了立即响应后没有回应数据，这种情况下不作任何处理，终端不会返回任何数据，依靠上位机的超时限制发现问题）

对于普通的单播指令：如果相邻终端失联，直接向上发送错误报告，格式如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（2Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 失联终端号（1Byte） | Sum（2Byte） |
| 66CC | xxxx | xx | xx | 88 | xx | xxxx |

对于多播指令：

如果此多播指令不需要返回有效数据（比如多播的是实时时钟配置指令）处理方法同上。

如果此多播指令需要返回有效数据，还要话分两头：

如果失联终端在多波段之前，处理方法同上

如果失联终端在多波段之中，就从失联终端之前返回数据，对于上位机来说，它将收到 残缺的数据。

# 串口中断接收数据的机制

在把串口中断中接收到的完整数据拷贝到缓冲区之前，主程序通过标志位限制了串口接收新的数据。也就是说主程序对串口接收数据有控制权。

要有一个时限，如果时限内串口还没接受完数据，主程序要清标志以便串口接收新的数据。

如果是数据，中断中最少需要接收4个字节才能获得数据长度，才能进一步判断数据是否接收完全。

无论数据还是指令，中断中最少接收2个字节才知道新的指令或者数据来了。

假如串口只收到了1个数据就不再有数据触发接收中断了，中断不会告知主程序有数据要等待接受。没问题。

假如串口只收到了4个数据就不再有数据触发接收中断了，主程序会通过时间限制和数据长度合理作出反应，没问题。

假如串口只收到了2个数据就不再有数据触发接收中断了，怎么办？

假如串口只收到了3个数据就不再有数据触发接收中断了，怎么办？

也就是说串口接收中断计数器等于2,或者3的时间是要有限制的！

解决办法是中断在开始接收数据之后立即给一个计数变量赋值，这个计数变量一直在系统滴答定时器中断里自减，以此作为接收前四个字节的时间限制。

# 关于指令是否等待回应的设置

对于指令的回应有3种，而且他们的帧头都是数据帧头：

1，立即响应

2，数据回应

3，失联报告

配置指令不需要数据回应，但是他却可能接到相邻终端的一种数据回应：终端失联报告

所以配置指令下发之后终端不需要等待数据回应，但是却必须对类型为0X88的数据回应做出反应。这就难办了！

解决办法是设置一个标志：NoneedDataResponse\_Flag

该标志为0时代表需要数据回应。

为1时代表不需要数据回应。

为2时代表不需要数据回应，可能需要等待失联报告。

为3时代表需要等待数据回应或者失联报告。

总结如下：

**如果指令是配置指令：**

**对于发给本端的配置指令：直接立即响应，然后执行配置**

**对于发给临端的配置指令：直接立即响应，然后等待临端的立即响应。**

**NoneedDataResponse\_Flag=1；**

**对于发给远端的配置指令：直接立即响应，然后等待可能出现的失联报告。**

**NoneedDataResponse\_Flag=2；**

**如果指令是读取信息指令：**

**对于发给本端的读取信息指令：直接数据回应。**

**对于发给临端的读取信息指令：直接立即响应，然后等待临端的数据回应。**

**NoneedDataResponse\_Flag=0；**

**对于发给远端的读取信息指令：直接立即响应，然后等待临端的立即响应，**

**然后等待临端的数据回应或失联报告。**

**NoneedDataResponse\_Flag=3；**

关于等待时限的设置也要注意，在没有终端失联的情况下，只要设置一个足够长的等待时限就可以。假如目的终端失联了，它的前一端一定会耗尽自己的等待时限然后才上传失联报告，而此时前一端的前端的等待时限肯定也耗尽了。暂时的解决办法是目的终端前一端的等待时限设置的小一点。

# Flash存储单点铁轨全部信息的机制

在FPGA配合CPU的情况下，CPU每隔75秒会发送一次超声信号发射通告指令。就在预发送指令发送之后进行一次FLASH存储。

Flash的前10个字节作为备份区，CPU每次开机从备份区读出掉电之前最后一次数据存储的地址。

同时手持盒的指令目前包括以下几种：

1，访问特定时间段内存储进flash的信息。

2，更改flash备份区的内容。（目前的用意就是把备份区的地址清0，这样以后的数据就会从头开始覆盖掉以前的信息。）

3，擦除特定扇区的内容。

4，把普通终端改为4G点（前面三条都是在这一条实现的基础上实现的，拔掉手持盒之前要用指令把终端从4G点恢复成普通终端）

# 等待指令回应的机制



等待数据回应的所有信息注册在上图结构体中。

其中指令类型和指令方向用于确认数据回应是否正确。

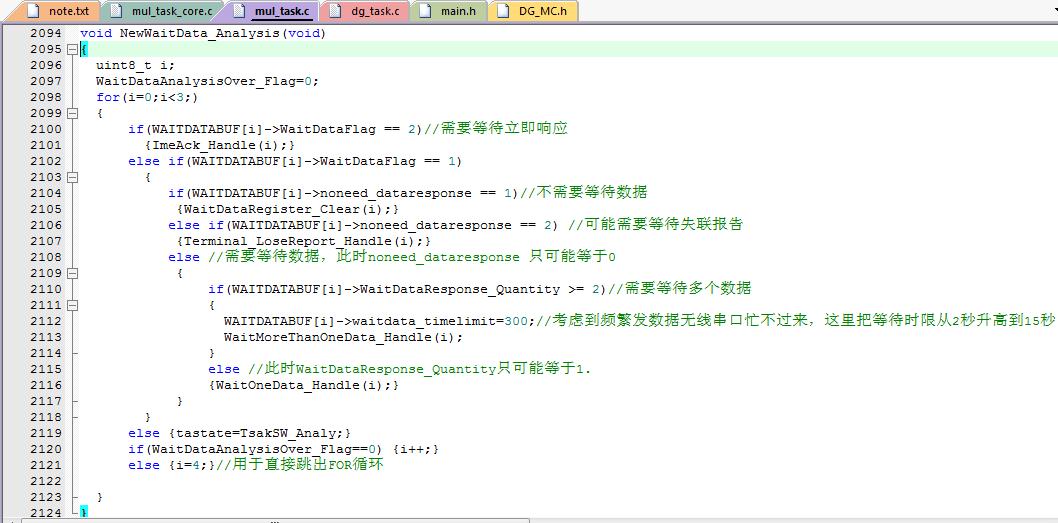
还有些信息是保存了OOM\_TABLE的数组下标，用于恢复备用行动步骤的。

WaitDataFlag

noneed\_dataresponse

WaitDataResponse\_Quantity

这三个信息使得等待回应变得很复杂。



# 手持盒接入

手持盒通过串口2与普通终端相连，收发指令数据的格式与上位机无异。

再细想想会不会有本端基本信息未知的情况呢？（比如发生了事故，终端被偷走了毁坏了，终端编号已经无法得知）

为了保险起见，要有一条特殊指令可以获得任一终端的基本信息。

# Flash存储数据的格式以及读取方式

FLASH 中要有一片区域保存终端基本信息。（5端地址）。就用地址5,6,7,8,9吧。

不如flash的第一扇区就用于存储基本信息吧。

幅值等信息从flash的第二扇区开始写入。

上电自检flash的ID，如果ID不符就认为flash失效了。这个自检有可能是不准确的，所以我在想要不要有一条指令能让CPU自检flash或者触发CPU的一次复位。

Flash存储数据的格式：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头（2Byte） | 长度（2Byte） | 源地址（1Byte） | 目的地址（1Byte） | 类型（1Byte） | 终端号  （1Byte） | 年（1Byte） |
| 66CC | xxxx | xx | xx | F4 | xx | xx |
| 月（1Byte） | 日（1Byte） | 时  （1Byte） | 分 （1Byte） | 秒 （1Byte） | 通断  （2Byte） | 应力（4Byte） |
| xx | xx | xx | xx | xx | xx | xxxx |
| 温度（4Byte） | 本端信号幅值  (4Byte) | 远端信号幅值  (64Byte) | ......... | 年（1Byte） | 月（1Byte） | 日（1Byte） |
| xxxx | xx...xx | xx...xx |  | xx | xx | xx |
| 时  （1Byte） | 分 （1Byte） | 秒  （1Byte） | 通断  （2Byte） | 应力（4Byte） | 温度（4Byte） | 本端信号幅值  (4Byte) |
| xx | xx | xx | xx | xxxx | xxxx | xxxx |
| 远端信号幅值  （）  (64Byte) | Sum  (2byte) |  |  |  |  |  |

CPU存够10组有效数据的时候再统一写入flash，不必带着帧头和校验和，因为他们会随着终端基本信息的改变而改变。

同时在备份区存着最晚的有效数据的地址。

备份区还要有一个flash是否写满的标志，程序上电之后从flash里获得这个是否写满的标志，这个标志发生变化的时候再把它写入flash。

**对于每一个终端，如果flash未写满，最早有效数据一定存在最小地址处，**

**如果flash写满了，最早有效数据一定存在最晚有效数据紧挨着的后面。**

**所以备份区只需存储flash是否写满的标志和最晚有效数据的相关信息即可。**

**要留意：对于刚刚经历全片擦除和未被使用过的flash，或者数据写到边缘临界地址的情况。**

下面是备份区的地址对应的数据所代表的含义：

|  |  |
| --- | --- |
| Flash地址 | 具体含义 |
| 0 | 最近一次写的数据的地址的[31:24]位 |
| 1 | 最近一次写的数据的地址的[23:16]位 |
| 2 | 最近一次写的数据的地址的[15:8]位 |
| 3 | 最近一次写的数据的地址的[7:0]位 |
|  |  |
| 4 | 本端终端号 |
| 5 | 前前端终端号 |
| 6 | 前端终端号 |
| 7 | 后端终端号 |
| 8 | 后后端终端号 |
| 9 | 在终端基本信息被改变并且flash正常工作的情况下该位会被写1。  这一位用来反映备份区是否存有终端基本信息 |
| 10 | reserved |
| 11 | Backup寄存器是否存有掉电前的终端基础信息 |
| 12 | reserved |
| 13 | Flash是否写满的标志，为1的时候是写满，为0或者FF代表未写满 |
| 14 | Flash当前存储的扇区号 |
| 15 |
| 17 | reserved |
| 18 | reserved |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| 25 | reserved |
|  |  |

**注意：**

**由于flash在往相同地址写入数据前必须擦除原有数据（根源在于flash只能写0不能写1，所以用擦除指令把flash全部写1，再用写指令给特定位写0）。**

**所以上表格的备份区内容全部存在CPU的备份寄存器中。**

**备份寄存器在写数据的时候需要开时钟，读数据的时候不需要。到底是一直开着时钟还是写数据之后把时钟关掉呢？**

**在flash写满之后，要从头写的时候，必须先擦除再写。那就一个扇区一个扇区的擦除吧。**

**因此也导致存取数据的时候以扇区为单位。**

# **从获取flash中的铁轨信息指令中的时间参数确定FLASH的读取地址**

1，上传的数据量不必正正好好，只多不少就行。比如上位机问3:00~5:00的数据。我可以回2:48~5:10的数据。

首先看时间参数中比较晚的那一个：下面就把他叫做T2。它对应的FLASH地址叫做addr2

如果T2比flash中最新存储数据的时间还晚，说明有数据在CPU数组里还未写进flash。

这种情况下addr2就是最新存储数据的地址。

如果T2就是FLASH中最新存储数据的时间，addr2就是最新存储数据的地址。

如果T2早于FLASH中最新存储数据的时间，就根据每一个数据相隔75秒，每一个扇区相隔48\*75秒估计出addr2。

考虑到T2并不一定是75秒的整数，而且flash可能存在掉电的情况。这样估算出来的addr2可能是不准的。还需要一个功能函数在发现addr2不准的情况下向前或者向后查询时间，原则是只多不少。最终获得只多不少的addr2.

时间参数中比较早的那一个：下面就把他叫做T1。它对应的FLASH地址叫做addr1

在获得了addr2之后，把T2与T1的时间差换算成扇区的个数差x，然后从addr2向前寻找x个扇区获得add1.

如果addr1不准确，就向前或者向后查询，最终获得只多不少的addr1.

在确定了addr1和addr2之后。。。。。。