下位机发送指令主体结构：Head + Datalength + cmdType + cmd + CRC\_8 ，

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HEAD (4 Bytes) | DataLength (cmdType + cmd) 2 Bytes | cmdType ( 1 Byte) | Cmd ( … ) | CRC\_8 |
| eb 80 08 be |  |  |  | <http://www.ip33.com/crc.html> |

上位机回复指令结构：同上 （ Head 不同）

Head => eb 90 09 be

指令详解：

1. 下位机请求连接上位机：

例如： send: eb 80 08 be 00 01 fc A4 (cmdtype = fc, cmd = null) 上位机只判断了 cmdtype，下位机随意

response: EB 90 09 BE 00 01 CF C1 （固定的回复指令）

1. 驱动 Jlink 检测

例如 ： send：eb 80 08 be 00 01 f0 00 01 B1 (cmdtype = f0, cmd = 00 01) 00: Jtag number- usb port 0, 1 ; 01: set /(00 : reset)

Response: sucessed => same as send data ;

failed => eb 90 09 be 00 01 f0 ff ff xx jlink not used/ jlink not ready

3: 获取配置信息：（由于该部分涉及下位机指令， 同时也涉及下位机测试流程， 故 分别表示 ae:下位机指令 b0:测试流程）

Cmdtype => aa

Cmd – 1 byte

// 单次检测指令

"SignalCMD": 0D

// 连续指令

"ContinualCMD": 0C

// 停止连续

"StopContCMD": 0B

// 内部版本

"InnerVersion": 89 ,

// 复位指令 "" / null 代表没有该项

"ResetDev": "FF",

//

“UseAE”:AE

“UseEA”:EA

Cmdtype ==> b0

Cmd – 1 byte

“ProcessEnableEA” : “EE”

Response: 2 byte

EE 00/01

例如 ： send: eb 80 08 be 00 02 ae xx xx

Response: eb 90 09 be 00 02 ae xx xxxxxxxxxx xx

Send: eb 90 09 be 00 02 b0 xx xx

Response: eb 90 09 be 00 02 b0 xx 00(01) xx

4: // 节点空板电流状态

cmdType Nodebarecurrent: 43 cmd – (node number 0 - 15)(value float )

Response: 01: pass

00: error

5: jlink 烧写指令

cmdType ==》 50

cmd => 1 byte (node number: 0 - 15)

send: eb 80 08 be xx xx 50 xx xx

Response: eb 90 09 be 00 02 50 xx 00/01/bb xx

00： error

01: pass

Bb: busy

6: 运行电流

cmdType NodeRuncurrent: 44 其余同 空板电流

7：版本号：(message 类)

cmdType inversion 89

8：发送 原始数据

cmdType adcdata 5E

cmd – (node number 0 - 15)(value adcdata + devdata )

response:

5e (number)(00: error 01: pass bb:busy)

上报消息类 数据格式：

cmdType:

{

// 节点连接状态

Nodeconnected: 91: cmd – 2 byte: (node number 0 - 15)(connect status 00/01)

Response: same as send

//节点短路状态

nodeShortOut: 92 同上

response: same as send

// 节点无响应

NodeTimeOut: 93 cmd (node number)

Response same as send

}

///////////////////////////////////////////////////////////////////////////////////

下位机发送指令主体结构：Head + Datalength + cmdType + cmd + CRC\_8 ，

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HEAD (4 Bytes) | DataLength (cmdType + cmd) 2 Bytes | cmdType ( 1 Byte) | Cmd ( … ) | CRC\_8 （1 byte） |
| eb 80 08 be |  |  |  | <http://www.ip33.com/crc.html> |

上位机回复指令结构：同上 （ Head 不同）

Head => eb 90 09 be

以下基本按照工作流程将 指令梳理出来（指令头 和 数据长度 不再赘述， 默认加上， 校验位不写出，均以 XX 代表）

注： \*\* 值 指该位省去

/ 符号表示可选的几种值

Same as request : CmdType + msgType + ? Node number + (00: data error / 01: msg received / BB: busy)

|  |  |  |  |
| --- | --- | --- | --- |
| 释义 | CmdType | CMD | Response( cmdtype + cmd ) |
| 下位机请求连接上位机  （执行完成后才能执行以下指令,否则会返回错误） | FC | \*\* | cmdtype  CF：connected  FF: connect error (未握手，错误，用于以下指令)  除了该回复数据，其他指令Request 和 response 的 cmdtype均为一致。之后将不再赘述  Cmd: \*\* |
| 驱动JTAG 检测 | F0 | 2byte  JTAG usb port number: 00/01  JTAG IO Set/Reset: 00/01 | 2 byte:  cmdType: F0  cmd: 00/FF (cmd has execute / something error) |
| 获取node配置 | A0（测试指令/ byte[]） | 单次检测指令： 0D | A0 + 0D + xxxxxxx |
| 内部版本号指令：89 | A0 + 89 + xxxxxxx |
| 复位指令： FF | A0 + FF + xxxxxxxx |
| AE 指令： AE | A0 + AE + xxxxxxx |
| B0 (测试流程 / bool) | AE 指令是否使用：E5 | A0 + E5 + 00/01 (not /use) |
|  |  |  |
| 开始测试状态 上报 | CA | msgType – DA | Same as request |
| 压板到位状态 上报 | CA | msgType - D0  DevStatus 00/01 | Same as request |
| 节点连接状态 上报 | CA | msgType – 91  node number 0 – 15  connect status 00/01 | Same as request |
| 节点短路状态 上报 | CA | msgType – 92  其他同上 | Same as request |
| 节点空板电流 | 43（Node Bare current） | Node number: 0 -15  ADC Value: float \* 100 | 43 + Node number +  00: error  01: pass |
| Jlink 烧写程序指令 | 50 | Node number: 0 -15 | 50 + Node number +  00：error  01： pass  BB: dev busy |
| 节点运行电流 | 44 / 45 / 46 | 同空板电流 | 44 / 45 / 46 + Node number +  00：error  01： pass |
| 节点版本号 上报 | CA | msgType – 89  Node number: 0 -15  InnerVersion: XXXXXX | Same as request |
| 节点采集数据 上报 | 5E | Node number: 0 -15  ADC Value: XXXXXX | 5E + Node number +  00: data error (crc)  01: pass  BB: busy |
| 5C（OU/OV） | Node number: 0 -15 | 5E + Node number +  01 : pass  BB : busy |