PVMESH FPU与CV之间的通信控制协议设计

2016-09-05 吕海峰

1. Preface

在PMMesh网络体系中, FPU是一个工控计算机, 通过485/以太网总线等与CV连接, 因此需要一套通信协议规范用于FPU与CV之间的信息交互





1. Requirement

FPU与CV间的通信控制协议需要有以下特性:

* CV越简单越好--- CV升级困难, 而FPU升级简单, 将CV程序设计的简单, 以后升级操作只升级FPU即可;
* 支持”一个FPU-多CV”的应用场景;
* 支持CV主动向FPU发起Indication, 以支持SS端来的报警事件;
* 支持目前powermesh提供的全部功能, 包括diag, ebc, psr等等;

需要支持的powermesh功能

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | Interfaces | Functions |
| APP | | |  | app\_send(), app\_rcv() |
| MGNT | | |  | ping(), psr\_diag()… |
| PSR | DST | PTP |  | psr\_setup(), psr\_send(), dst\_flooding\_search()… |
| DLL | | |  | diag(), ebc\_broadcast(), ebc\_identify(), ebc\_confirm() |
| PHY | | | phase,ch, rate, ss, snr, data | phy\_send(), phy\_rcv() |

1. Consideration
   1. Plan
      1. Plan A

类似376.2, 所有功能都是应用层的, CV隐藏所有通信细节, 只约定几个功能.

缺点: CV编程难度大, 出错概率高, 调试困难, 固件更新困难;

结论:最不可取；

* + 1. Plan B

CV只做到PHY层, 所有收发对象均由485总线传递.

优点: 灵活性最大

缺点: 通信过于密集, 实时性不够好;

* + 1. Plan C

CV定义一堆powermesh的函数, 定义数据类型, 定义返回格式.

优点: 相当于把CV当库函数用, 库函数相对静态, 需要升级的概率低. 同时又能最大程度的利用现有的powermesh功能.

缺点: 编程难度有点大

另外: CV如何主动提示SS主动传来的数据接收?

* 1. Scenario

以下几种通信控制情景需要支持:

* 可靠应答-单次召唤单次响应: 如读写6810寄存器, read\_reg(), write\_reg();
* 可靠应答-单次召唤多次响应: 如获得总线上GV的个数与各自的UID;
* 不可靠应答-单次召唤单次回复: 确切的时间内给予(不给予?)不确定的回复, 如diag(), vc\_call();
* 不可靠应答-单次召唤多次回复: 确切的时间内给予多个不确定的回复, 如mc\_call();
* 多次召唤多次回复不可靠型transaction型: 多次反复多次通信才能完成的, 如explorer();
* SS端主动, FPU端被动的异步通信:从SS发来的警告信息, 如失压报警
  1. Interface

以下powermesh接口需要支持:

|  |  |
| --- | --- |
| Hardware |  |
| u8 read\_reg\_entity(u8 addr); |  |
| write\_reg(phase,addr,value) |  |
| void get\_uid\_entity(u8 xdata \* pt); |  |
| PHY Layer |  |
| SEND\_ID\_TYPE phy\_send(PHY\_SEND\_HANDLE pps) |  |
| SEND\_STATUS inquire\_send\_status(SEND\_ID\_TYPE sid) |  |
| DLL Layer |  |
| SEND\_ID\_TYPE dll\_send(DLL\_SEND\_HANDLE pdss) |  |
| STATUS dll\_diag(DLL\_SEND\_HANDLE pdss, ARRAY\_HANDLE buffer) |  |
| void set\_ebc\_response\_enable(BOOL enable) |  |
| u8 ebc\_broadcast(EBC\_BROADCAST\_HANDLE pt\_ebc) |  |
|  |  |
| APP Layer |  |
| SEND\_ID\_TYPE app\_send(APP\_SEND\_HANDLE pas); |  |
| BASE\_LEN\_TYPE app\_rcv(APP\_RCV\_HANDLE pa); |  |
|  |  |

* 1. 考虑
     1. 技术路线选择

1. 考虑到3.2 Scenario中提出的这些不同的通信控制场景, 如果采用Plan C + 485总线结构, 极难定义一个统一的通信控制模型, 能处理上述所有的通信类型, 而且, 总线型485从目前的搜索结果来看, 没有见到工程里有做成对等网的先例, 真的做成对等网将带来许多许多麻烦和难以解决的问题.

2. 485是一种半双工通信线路, 为了多个CV同时工作, 也必须将网络类型做成以FPU为中心的星形485(否则就只能一个CV一个CV的操作, 效率要低得多);

因此, 将问题简化:

1) 否定485总线结构! 如果需要总线结构, 则使用带冲突避免的以太网方案; FPU的一个UART口对应一个CV, 以避免冲突问题, 同时解决多个CV同时通信的问题;

2) 采用Plan B, 将所有的不可靠应答统一放在FPU内解决(Python调试powermesh还是相对容易的), CV将符合自己UID的PLC帧整包发给FPU, 只留DLL层接口;

几个考虑:

1) diag之类需要返回帧ss, snr的怎么办?

答: 每个接收的PLC帧, 都放在一个有序列号的缓存里, 超出容量后, 旧的覆盖新的. 缓存的内容除了帧的报文外, 还包括ss, snr, agc等信息; FPU需要的话可以用一个命令将其调出;

有了上述功能, FPU理论上就已经可以完成diag了, 但diag还是单独调用现成的powermesh里的好.

2) ebc一类的怎么处理?

目前倾向于FPU自己完成;

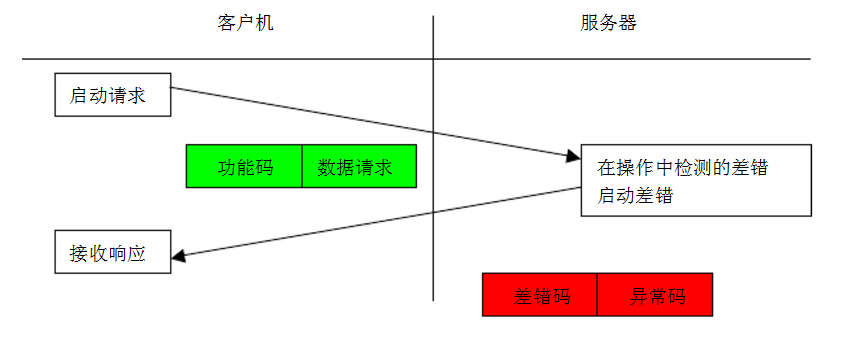
3) SS传来的报警帧如何确保SS到CV, CV到FPU的可靠传递?

SS上行的帧, CV仅双向传递, 不保证双方是能可靠接收到的, 需要FPU给予确认, 否则SS将随机N秒后再次报告, 直到FPU给确认为止;

(这个功能仅有演示意义, 一旦多个点同时出故障, 将可能使整个系统的正常工作受到干扰;)

1. Design
   1. 通信模型:





* 1. 协议帧
     1. 帧结构设计

借鉴MODBUS RTU协议, 采用如下帧结构:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Section** | **Bit7** | **Bit6** | **Bit5** | **Bit4** | **Bit3** | **Bit2** | **Bit1** | **Bit0** |
| **ADDR\_H** | **High CRC byte of target UID (0x0000 is wildcard address)** | | | | | | | |
| **ADDR\_L** | **Low CRC byte of target UID** | | | | | | | |
| **FUNC** | **0: Normal**  **1: Exception** | **0: FPU to CV**  **1: CV to FPU** | **Function Code** | | | | | |
| **BODY** |  | | | | | | | |
| **CRC\_H** | **High CRC byte of CRC from ADDR\_H to last byte of BODY** | | | | | | | |
| **CRC\_L** | **Low CRC byte of CRC from ADDR\_H to last byte of BODY** | | | | | | | |

Comments:

\* 地址域2B, 是CV UID的CRC校验值, 0x0000为通配地址;

\* 功能码为1-63;

\* 操作码与功能码同;

\* 差错码 = 功能码 | 0x80;

\* CV返回标志 =功能码 | 0x40;

\* 为与CV的Debug信息区别, 交互协议的报文统一用十六进制ASCII表示, 同时用<>括起来;

* + 1. FPU-CV交互协议功能码设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caller | Function Code | Data Request | Data Response | Comment |
| FPU | 0x01: Read UID | NULL | 6 Bytes CV UID | 读取CV的UID; |
| 0x02: Notify Addr | 6 Byte FPU UID | 2 Byte FPU UID CRC | 通知CV FPU的UID, CV返回计算的CRC; |
| 0x03: Read PHY Reg | 2B phy reg address | 1B phy reg value | 读CV PHY寄存器 |
| 0x04: Write PHY Reg | 2B phy reg address, 1B phy reg value | 1B phy reg value after writing; | 写CV PHY寄存器 |
| 0x05: set plc indication level (TBD) | 1B indication\_level  0x00:  now indication(no modification)  0x01: phy\_rcv\_indication  0x02:  dll\_rcv\_indication | 1B indication level | 设置PLC indication的级别.  00: 读取当前级别;  01: phy indication(所有接收的帧都上报);  02: dll indication(只有符合powermesh格式且uid符合的帧才上报) |
| 0x10: Phy Send | 1Bphase, 1B len, 1B xmode, 1B prop, 4B delay, NB psdu | send queue id | phy\_send() |
| 0x11: DLL Send (TBD) | 1B phase, 1B lsdu\_len, 6B uid, 1B prop, 1B xmode, 1B rmode, 4B delay, NB lsdu | send queue id | dll\_send() |
| 0x12: dll\_diag | 6B uid, 1B xmode, 1B rmode, 1B scan | 16B diag result | 如Diag成功, 返回依次为下行指定频率的dbuv, ebn0和上行指定频率的dbuv和ebn0 |
| CV | 0x20: PLC indication | 1B phase, 1B phy\_rcv\_valid, NB | 无回复 | dll/phy层接收到的数据帧 |

* + 1. Exception代码

CV正常情况FUNC域返回与FPU下发帧相同的Function Code, 并将Bit6设置为1. 当出现错误时Bit7位设置为1. 同时附带一字节错误代码, 定义如下:

|  |  |
| --- | --- |
| Error Code | Meaning |
| 0x81 | 不支持的命令字 |
| 0x82 | 错误的命令格式 |
| 0x83 | diag失败 |
|  |  |
|  |  |