光伏监控网络系统PhotonBit需求分析与系统设计

2016-03-08 吕海峰

1. 概述

PhotonBit系统应用在光伏电站中, 获得每一块光伏组件的实时电压, 电流, 温度等状态量, 以后台专家系统判断组件的健康状态, 并在判断出组件故障时提供报警的信息服务系统;

1. 系统概述



PhotonBit系统分为四层对象

1. Interface Layer (IF) 呈现界面层.

* 负责将数据以可视化图表形式呈现给用户, 分为面向持有者与面向运维人员两种维度, 前者注重采集数据的汇集呈现与收益分析, 后者注重系统长期运行视角下的总体系统健康状态, 与偏离平均值的病态组件发现;
* 在系统出现问题时报警, 并在问题修复后汇报给持有者;
* 提供Web与App两种界面;
* Entity: Web Server, Web Browser, App(ios, Android)

1. CloudExpert Layer (CE) 云专家层

* 存储所辖范围内所有受托管的光伏电站的数据, 包括地理信息, 光照度信息, 发电量信息, 组件信息等大数据库(Big Database)
* 专家诊断系统
* Entity: Database, Server

1. Converge Layer (CV) 数据汇集层

* 负责所辖范围内组件实时状态量采集, 通过PLC通信与SS层通信, 通过WLAN, Fiber, 3G/4G与CE层通信
* 在上层通信断开时, 有本地存储, 保证数据不丢
* Entity: RasperPi, Converger, SQLLite

1. Sensor Layer (SS) 传感层

* 负责组件状态量测量, 通过PLC通信与CV层通信
* Entity: Sensor

1. Spec
   1. 设备
      1. SS(组件传感器 SenSor)

位置: 光伏组件的接线盒内/外

功能: 采集光伏组件实时电压, 电流值, 并通过PLC报告给CV

组成: 基于6810的模块

* + 1. CV(数据汇集器ConVerger)

位置: 逆变器

功能: 采集组件传感器, 存储, 上传云服务器

组成: 树莓派核心板(CoreBoard)+通信单元(CommModule)

开发: Python, C

功能划分: 类似集中器与本地通信单元关系, 核心板向上负责数据存储, 上传云端, 向下负责指定组件发起采集任务

问题: 路由关系存储在哪里?

如果用模块

* + 1. CS(云服务器CloudServer)

位置: 云服务器

功能: 收集

* 1. 问题

1. SS与光伏组件的对应关系如何进入电脑

光伏组件有条形码, 安装时需要一一对应, 录入

1. 需求分析
   1. 数据采集

* 电压
* 电流
* 温度(SS附近)
  1. 异常主动报警
* 电压异常主动报警(演示时用, 该功能不实用, 实际系统中不用)
  + 由GW下发阈值电压值
  + 模块在电压下降至阈值以下时启动报警
  + 在停止条件没有满足的情况下, 模块以五秒钟正负随机2毫秒的间隔重复报警;
  + 报警停止的条件: 1. 电压恢复至阈值以上; 2. GW启动一次当前状态信息采集;

1. 硬件架构定义
   1. 组件传感器
      1. 组件结构分析



* + - 1. 一体式型式结构



上面那种型式结构似乎是唯一可行的形式结构,

将锰铜丝做成电感也不可行, 因为无法同时保证电感隔离光伏组件, 又能保证电流采样两端的电压合适, 除非将电源与信号线分开



* + 1. 组件传感器需求分析
* PLC通信
  + MultiSim仿真确定最大效率耦合方案
* 电能电压测量, BL6253
* 温度测量, 热敏电阻+BL6523
* 电源:
  + 前端不使用大电容避免吸信号, 或者串一个1~2mH电感隔离
  + 电源前端加42V稳压保护(第一版加, 后面根据实际情况可以考虑降成本去掉)
  + 2451|2841, 前者适用电压36V, 太低, 后者绝对最高电压45V;
* 开发
  + 串口被占用, 如何调试?
  + 板子上带一个STM32, 调试完成后成熟后去掉;
* 测试点
  + 烧程序
  + 通信测试
  + 计量测试
  + 温度测试
* 可校准
* 可断开(Optional)
  + 需要一个10A磁保持继电器? 能否断点-上电后保持状态?
  + 继电器开路后, 板子
  + 需要两个IO
* BL6810资源:
  + TXD, RXD, P32(TXON), P33(继电器+), P34(继电器-);
    1. 组件传感器计量电路分析
       1. 设定BL6523固定 PGA增益x16

取样电阻阻值2mOhm, 这样最大功耗20^2\*1e-3 = 800mW, 正常电流7A时功耗=100mW, 可以满足要求;

\* 20A时对应的AD端口电压40mV, 对应的AD转换完是600139左右;

\* x16设置时, ADC每个step电压为63.6nV左右;

\* ADC最大输出1048576, 也即2^20, 21位; x16时, 最大电压输入66mV左右;

如果用25℃, 1A, 20A两个典型值做校准, 即2mV, 40mV, 对应的取值为3713.8, 600139.1(wave\_ia\_x16.txt), 得到的拟合值为k = 15695.4/mV, b = -27677

所以, 以adc的值反推电压的算法即

(value-b)/k

以” e:\Works\PLC\_G1\_Project\Documents\2016-03-11 BL6523test\wave\_ia\_x16.txt”数据为源, 测试1A, 20A两点校准电压反推的误差, 如下: (分析程序MatlabWork\test\bl6523test\calib\_analysis.m)

反推电压与参考电压对比:



绝对误差



相对误差(以参考电压为分母)



可见对于大于0.01mv, 相对误差小于0.01

若以有更多数据点的电阻分压的数据为数据源(e:\Works\PLC\_G1\_Project\Documents\2016-03-11 BL6523test\resistance\_divided\_wave\_ia\_x16.txt), 仍以相差20倍的两点法校正, 则结果如下:







单独看0~0.045mv处的相对误差



与上一版是很像的, 而实际上这两种情况, ADC输入端口获得的电压绝对不同, 这说明:

\* 这种误差可能是电源引入的, 也即控制电源输出0.001, 0.002, 本身就存在着图中的误差;

\* 一级表的精度为1%, 0.5级表的精度为0.5%, 没道理我们就一定要做到千分之一精度;

\* Fluke的电流表的精度都为1%

\* 因此这种精度是够用的

1. 通信协议设计
   1. Powermesh v1.0的优势与劣势

powermesh v1.0已经成功应用了三年多, 并没有遇到重大的Bug和不足. 这里总结一下powermesh的优势与不足

优势:

\* 物理层和链路层做的比较solid, 对于scan, diag的支持后来证明非常有用;

\* srf的快速应答机制证明非常有效;

\* 分层的协议设计, 保证了弹性和扩充性;

不足:

\* 包长定义受了256字节限制, 后来增加了一位, 最大支持512字节;

\* 没有对MAC层的支持;

\* 源地址目的地址有点长, 对于短包效率有点低;

v1.1版设计

\* 保持v1.0的设计, 支持长包, 最长512字节, (再长硬件不支持,没有意义);

\* 支持地址编码

* 1. 软件定义

PLC通信设备固件共有两个版本:

photonmesh\_ss: 用于ss设备, 基于8051

photonmesh\_cv: 用于cv设备, 基于STM32F030C8

采用跨平台C语言设计

架构如下:



说明:

1. se是ss的debug版本, 与ss的区别在于 基于stm32f030, 6810工作在device模式, 有debug输出, 可断点调试

2.

1. 应用层协议定义
   1. ACP应用层协议帧格式

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 定义 | | | | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| APFH(1Byte) | | | | MUST BE 0 | APP Protocol  ACP Protocol = 0x18 | | | | | | |
| ACPR(1Byte) | | | | NA | IDTP  00-Domain/01-IDVD  /02-MUVD/03-GRUP | | FOLW | TRAN\_ID | | | |
| DOMAIN ID(2Byte) | | | | DOMAIN\_H | | | | | | | |
| DOMAIN\_L | | | | | | | |
| *IDTP=0* | *IDTP=1* | *IDTP=2* | *IDTP=3* |  | | | | | | | |
| 0B | VID  (2B) | START VID  (2B) | GID  (2B) | Addr Section:  \* IDTP = 0, All nodes in the same domain got the apdu, using for broadcast;  \* IDTP = 1, only the node match domain+indiVidual ID got the apdu;  \* IDTP = 2, nodes in domain that match [start, end] VID got the apdu;  \* IDTP = 3, nodes match the group id got the apdu, used for special purpose; | | | | | | | |
| END VID  (2B) |
| SELF VID  (2B) |
| BODY | | | |  | | | | | | | |
| ACPC | | | | mod(sum(APFH+ACPR+…ACPC),256)=0 | | | | | | | |

TRAN\_ID的意义：一个TRAN\_ID像一条gmail的会话, 由一个FOLW为0的帧发起, 后续的属于同一会话的数据交换都使用同一个TRAN\_ID, 但FOLW=1;

* + 1. BODY定义

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
| CMD | Normal:0 | Req  (0: No Res  1: Need Res) | Res | 0 | Command ID | | | |
| Except:1 | 0 | 0 | 0 | Except ID | | | |
| 参数… |  |  |  |  |  |  |  |  |

Normal/Except: 表示此帧用于传递普通信息还是递交错误信息

Req: 此帧数据是否需要回复, 1为需要回复, 0为不需要回复

Res: 此帧数据为上一同TRAN\_ID的询问帧的回复帧;

* 1. ACP地址设计:
     1. 应用层ID的概念

域Domain ID用于区分不同CV, 防止串扰引起的数据错误, 一般是CV UID的CRC;

个(Individual ID)用于区分不同SS, 由CV分配, 用于需要回复的命令, 数据采集等;

组地址用于同一域下的分组划分, 可以是非连续的序列号, 用于执行批量广播处理, 例如通知全部解列等;

(ID信息和校准参数信息都存储在NVR里)

四种寻址类型: 域广播(IDTP=0)/单地址(IDTP=1)/多地址(IDTP=2)/组广播(IDTP=3)

* 域地址广播(DB): 2B域地址, 用于广播, 如冻结当前电压电流等; (冻结采用随机数标识, 防止测试时引入数据重复), 全0为通配域地址;
* 单地址通信(VC): 2B域地址+2B个体地址, 用于一对一通信, 单独控制某个SS或补抄. 理论上DLL层, NW层已经提供了一对一单独通信的可能, 单地址作为多一层安全性校验增加可靠性, 全0为通配单地址
* 多地址(MC): 2B域地址+2B个体起始地址+2B个体结束地址. 用于批量数据采集, 即CV发送一条采集命令, 下属SS按个体地址顺序逐次上报采集数据

多地址采集方式:

SS接收到采集命令, 首先判断自己是否在地址范围内, 如果在, 看该命令是命令还是回复, 如果是命令, 计算延迟时间, 将数据推送到发送队列里. 如果是回复, 如果自己没接收到命令(以FLLW+TRAN\_ID判断), 则计算延迟时间, 将数据推送到发送队列里, 如果已经接收到命令, 根据当前回复的时间, 校正发送延迟时间.

* 组播(GB): 2B域地址+2B组地址. 用于特殊节点群体广播
  1. 回复地址方式(是否回复以Cmd的Req为准)

回复方式按原地址方式回复, 即无地址则无地址回复, 域地址则域地址回复(软件设计应避免一对多问询时使用广播方式, 应使用IDTP=2方式)

ACP CMD定义

|  |  |  |
| --- | --- | --- |
| ACMD代码 | 附带参数 | 回复内容(if Req=1) |
| 0x01 设置域ID, 个体ID, 组ID | 2字节域地址, 2字节个体地址, 2字节组数据  (0xFFFF为不改变当前地址, 需携带校验码0x9527, 否则仅返回当前地址) | 2字节域地址, 2字节个体地址, 2字节组数据 |
| 0x02 读取当前参数 | 1字节参数掩码  (0, 0, 0, 0, 0,温度,电流,电压) | 1字节参数掩码, 后接按掩码从高到低顺序排列, 每个参数2字节, 大端数据格式, 有符号整数, 单位为0.01V, 0.001A |
| 0x03 冻结当前参数 | ‘F’: 冻结参数, 1字节特征码, 1字节参数掩码 | 1字节特征码 |
|  | ‘R’: 读取冻结参数, 1字节特征码, 1字节参数掩码 | 1字节参数掩码, 后接按掩码从高到低顺序排列, 每个参数2字节, Big Endian |
| 0x04 下发连续 | 1字节参数掩码, 2字节特征码, 2字节间隔毫秒数, 2字节点数 |  |
| 0x05 模块欠压报警功能 | ‘V’: 设置欠压阈值, 后接2字节绝对阈值, 2字节相对阈值. 当电压跌落至绝对阈值以下, 且与前一次检查的跌落幅度大于相对阈值, 则向GW发出提示; | 空 |
| 0x0A 模块状态提示 | 空. 此帧由模块主动发出, 提示GW当前状态有异常, GW读取一次当前参数则消除提示, 否则每10秒钟左右重复一次提示; |  |
|  |  |  |
|  |  |  |
| ~~0x06 设置通信参数(测试)~~ | ~~TBD~~ | ~~TBD~~ |
| ~~0x07 读电量参数寄存器系数(测试)~~ | ~~1字节参数掩码~~ |  |
| ~~0x08 读校正系数(测试)~~ | ~~1字节参数掩码~~  ~~(0, 0, 0, 0, 0,温度,电流,电压)~~ | ~~按掩码从高到低顺序排列, 大端~~ |
| ~~0x09 设置校正系数(测试)~~ | ~~1字节参数掩码~~  ~~(0, 0, 0, 0, 0,温度,电流,电压), N字节校正系数~~ | ~~空~~ |
| ~~0x0A 保存校正参数(测试)~~ | ~~空~~ | ~~写入校验通过后, 回复空~~  ~~否则EXCEPT~~ |
| ~~0x0B 校正温度(出厂测试)~~ | ~~1字节序号, 1字节温度值~~ |  |
| 0x0C 校正功能(出厂测试, 需携带校验码) | ‘T’: 校正温度. 1字节序号, 1字节温度值 | 温度寄存器值 |
| ‘I’: 校正电流. 1字节序号, 2字节电流值(有符号整数, 单位mA) | 电流寄存器值 |
| ‘U’: 校正电压. 1字节序号, 2字节电压值(有符号整数, 单位为10mV) | 电压寄存器值 |
| ‘S’: 保存校正系数 | 空 |
| ‘R’: 复位 | 空 |
|  |  |  |
|  |  |  |
| 0x0E NVR相关功能 | 0x45(‘E’): Erase NVR  0x57(‘W’): N字节数据, 包括校验  0x52(‘R’): 1字节长度 | ‘E’: 回复为空 ‘W’: 写入校验完成后, 回复空, 否则EXCEPT  ‘R’: 回复从0开始的N字节数据 |
|  |  |  |
|  |  |  |
| 0x0F 扩展 |  |  |

EXCP定义

|  |  |  |
| --- | --- | --- |
| EXCP代码 | 意义 |  |
| 0x81 硬件错误 | NVR读写错误 |  |
| 0x82 格式错误 | 命令错误 |  |
| 0x83 无法执行命令 | 无有效特征码 |  |
| 0x84 ID匹配错误 | 域地址或个体地址不匹配 |  |
|  |  |  |

注1:

链路层使用带地址的通信方式, 可以保证只有唯一节点接收到消息. 因此可以让应用层使用通配地址, 设置域地址, 个体地址, 组地址;

注2:

应用层的4种寻址通信方式, 与ACMD的处理, 两个过程是各自独立的. 因此寻址通信只能保证符合寻址定义的帧会被送给ACMD处理, 并将ACMD处理返回的帧按寻址通信的返回设计发送到信道. 却不能保证这二者的结合是正确的, 这一点需要设计程序时来保证. 例如:

\* 使用域寻址方式, 却要求读取返回当前参数, 除非链路层保证了只有一个接收节点, 否则势必会引发响应的冲突; 也就是说: 域寻址, 组寻址, 应该使用无响应的命令;

\* 使用MC寻址, 却没有在响应帧的回复数据体首字节指明读取的参数掩码, 因为MC的回复可能是REQ激发的, 也可能是RES激发的, 因此需要RES帧里也带有参数掩码, 这样其他被RES激发的帧就能够知道有哪些状态量需要抄读;

Consider:

1. 设置地址, 擦除NVR, 等危险操作考虑设置授权码校验;

2. 以防万一, CV初始化时采集全网域节点的地址, 校正参数备份, 出现NVR错误时可以远程下发恢复;

3. 为解决主站串扰, 考虑不同域设置不同频率分区(设置通信参数命令功能);

1. 硬件设计
   1. 光伏通信组件PCB设计

大语设计了第二版光伏通信组件的PCB布局, 大致如下:



我看后觉得问题在于大电流在PCB走的距离太长, 容易引起板子发热, 不安全, 而且还没地方拧螺丝. 最终还是改成以下方案:



原来这种方案被否定是因为觉得线是长短长短排列太丑, 现在让位与功能性, 就先不管丑不丑了(这种方案其实就是电表的接线方法)

Vin和Vout最好能让厂家定做一个双排座插件, 这样电流就可以完全不从板子上走了.

* 1. 工装设计考虑

工装板需要有以下功能:

1. 对模块烧录程序, 并检查;

2. 自带衰减器, 检查模块通信能力;

3. 控制外加电流电压源, 校准模块;

4. 检查模块的电压电流测量;

5. 打印UID标签;

因此工装板需要有以下接口:

1. 通信接口三个: 与电脑通信(下载程序, 打印输出), 与电源通信(控制电压电流输出), 与打印机通信(打印标签);

2. 有LED指示灯;

3. 有拨档开关, 万一没有电源, 可以通过拨档控制跳过电源校正环节;

4. 有衰减器;

5. 工装上有电源接口;

功能框图如下:



另外需要关注可能没有外控电源的情况, 因此需要:

1. 双供电: 工装板可以直接给DUT供电; 如果有程控电源存在, 则DUT由程控电源供电; 二者由工装台接线方式决定;

2.