



ETWS-1810 雷达信处

软件设计讨论

部 门： 研发部

编 制： 罗敏

版 本： 1.0

使用 L<sup>A</sup>T<sub>E</sub>X 撰写于 2018 年 12 月 18 日

## 摘 要

本文主要是关于雷达系统自研信号处理分系统软件设计讨论，目的是确定信号处理分系统软件和算法处理部分的详细工作。因对于信号处理部分均是从零起步，一个好的实现方案和计划更显的尤为重要，本文从信号处理分系统输出目标出发，逐步详细阐述各个设计目标的计算和实现过程。

**关键字：** 雷达系统 信号处理 算法 实现

# 目录

- 1 设计目标 ..... 1
  - 1.1 输出指标 ..... 1
  - 1.2 硬件架构 ..... 1
- 2 设计原理 ..... 3
  - 2.1 数据模型 ..... 3
    - 2.1.1 三维数据模型 ..... 3
    - 2.1.2 数据模型理解 ..... 4
  - 2.2 详细设计 ..... 4
- 3 设计实现 ..... 5
  - 3.1 服务器端 ..... 5
    - 3.1.1 功能设计 ..... 5
    - 3.1.2 接口设计 ..... 5
  - 3.2 设备端 ..... 6
    - 3.2.1 功能设计 ..... 6
    - 3.2.2 接口设计 ..... 6
  - 3.3 客户端 ..... 7
    - 3.3.1 功能设计 ..... 7
    - 3.3.2 接口设计 ..... 7
- 4 详细设计 ..... 8
- 5 总结 ..... 10
- 参考文献 ..... 10

# 第一章 设计目标

## 1.1 输出指标

本节主要是对雷达信号处理分系统的设计指标进行确认，确定设计输出的指标之后，才能够更好的针对设计指标进行软件设计和规划。在图 1.1 中根据《X 波段双偏振天气雷达改

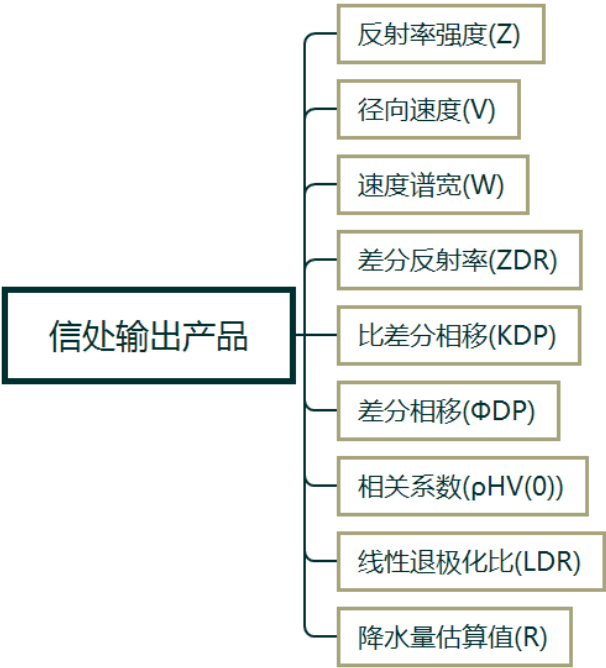


图 1.1 信处输出原始产品

造升级项目立项报告》列了一下信处分系统应该要输出的原始产品。

## 1.2 硬件架构

目前硬件架构已经基本确认，具体如图 1.2 所示主要采用 AD 板 + 接口板 +X86 板的方案。AD 板与接口板之间采用 SIRO 接口连接，接口板与 X86 板之间采用 PCIe3.0 接口连接。如图所示，AD 板主要负责下变频、抽取、滤波等工作，接口板则负责接收 AD 板采集的数据，然后对数据进行 DBF、脉冲压缩等处理，同时需要控制伺服、波控等组件,X86 板则接收接口板处理后的数据做后半段的信号数据处理，最终输出上一节中提到的输出指标。

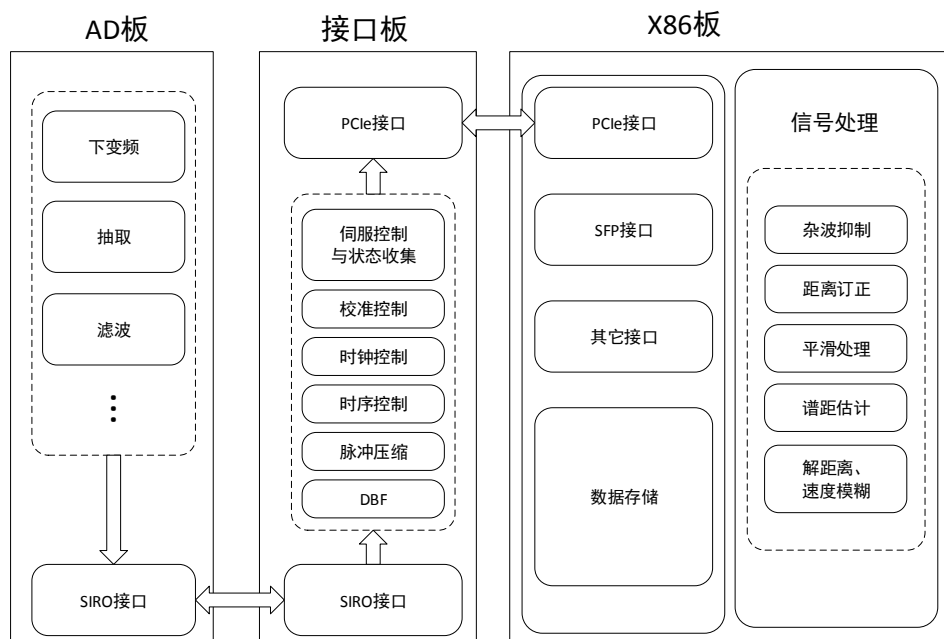


图 1.2 硬件设计

因本文主要讨论软件的设计实现，主要会聚焦于 X86 板的信号处理部分，目前接口板与 X86 板的数据通道已经是对接过，可以从接口板读取相关数据，后续的工作便是两个主要的工作：

- 设计接口板与 X86 板接收数据的协议
- 处理接收到的数据

## 第二章 设计原理

### 2.1 数据模型

#### 2.1.1 三维数据模型

需要对雷达回波进行数据处理，首先要将回波数据抽象成一个数据处理模型，在这里采用了三维数据块作为一个数据处理模型<sup>[4]</sup>，这一处理方法引用自雷达信号处理基础第三章，基本处理模型如图 2.1 所示：

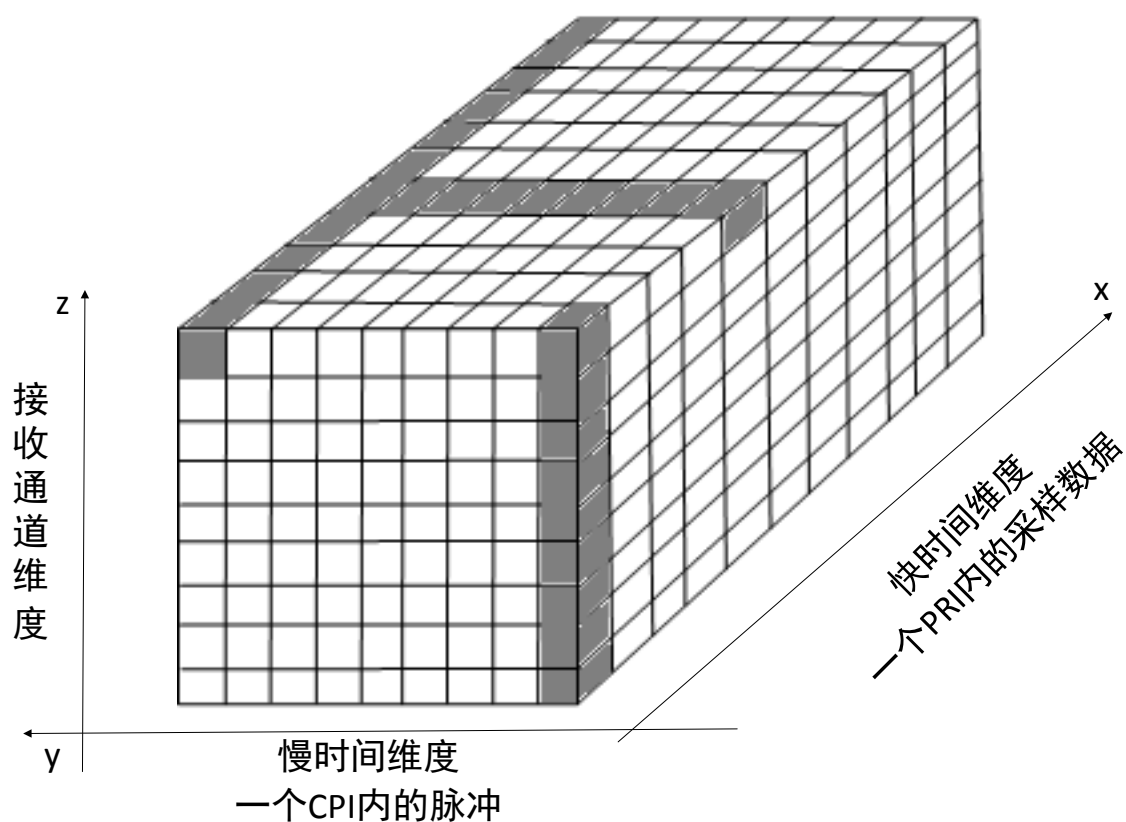


图 2.1 信号处理数据块

可以看到, 在上图中的  $x$  轴方向标注为快时间维度, 本质就是一个 PRI(脉冲重复间隔) 内的回波采样数据. $y$  轴标注为慢时间维度, 本质是一个 CPI(相干处理间隔) 内的多组回波数

据。z 轴标注的是接收通道维度，本质就是对应天线的接收通道数目。

### 2.1.2 数据模型理解

上一节是介绍了基本的数据模型，在本节中还需要深入的理解一下设计的数据模型，在图 2.2 中，针对数据模型的三个维度进行了进一步的标注。一些基本处理手段。

在本文的设计中，数据服务器主要负责建立交互通信通道，客户端可以直接穿透 Internet 网络访问内网的综合控制网关设备，获取控制网关的设备信息和数据。

## 2.2 详细设计

整个软件系统其实基本符合经典的 C/S 架构，即客户端与服务端架构，架构图如图 2.2 所示，设备端运行远程配置软件，负责收集数据与服务器程序交互，客户端通过简单的数据配置或者获取接口便可以方便的访问数据服务器，设备端其实也是一种特殊的客户端，区别其实只是设备端运行于网关之上，客户端则需要运行在 PC 上，客户端同时还需要交互界面程序与使用者进行友好的交互。

在基本的框架符合经典 C/S 架构的情况下，本文将采用大量现有的成熟框架进行服务端和设备客户端的设计，在设备端将会采用 curl 工具开发，服务端则采用开源框架根据需求定制开发，客户端则通过浏览器登录远程服务器界面进行人机交互。

## 第三章 设计实现

### 3.1 服务器端

#### 3.1.1 功能设计

服务器端的基本功能主要有三个：

- 处理设备端消息
- 处理客户端消息
- 数据存储管理

如图 3.1 所示，更加形象的描述了服务端的业务逻辑功能。

在我们的使用场景中设备端和客户端一般都是处于内网，服务端则分配有公网 IP 地址，因此设备端和客户端均会主动向服务端发起连接，设备端主要将自身的数据信息发送给服务端，服务端接收数据之后将数据存储到数据库，等待客户端的查询。在本文中拟采用的 rttys 同时提供了客户端直接获取设备端数据的接口，可以即时操作即时获取相关设备端数据。

#### 3.1.2 接口设计

本节主要对服务端与设备端和客户端的通信接口进行详细设计。

##### (1) 设备端接口

设备端接口设计成符合 HTTP 规范的标准接口，接口如下所示：

```
https://Server-ip:port/devs
```

设备端通过服务器提供的 devs 接口采用标准 post 的方式将自己所收集的数据传输给远端服务器，服务器收到数据之后再将其存储起来。

##### (2) 客户端接口

客户端接口同样也是设计成符合 HTTP 规范的标准接口，接口如下所示：



```
https://Server-ip:port/clients
```

服务器端 clients 接口提供给客户端访问的接口，客户端通过 clients 访问指定的设备，获取设备信息。

(3) 即时通讯接口客户端通过即时通讯接口可以直接的访问设备端的内容，连接设备的 shell 命令行，执行相关的设备端命令。

```
https://Server-ip:port/cmd [1]
```

## 3.2 设备端

### 3.2.1 功能设计

设备端的功能则相对比较简单，主要是通过 curl 工具根据通信协议将网关设备的信息和与网关设备相连接的设备的消息传递给服务器端，同时从服务器端下载设备配置文件。

设备端基本功能和程序处理逻辑如图 3.2 所示，通过定时维护与服务器的数据传递，获取配置文件和将收集的网关、UPS 和摄像头等设备的数据传输给服务器端，同时比较从服务器端下载的配置文件与当前配置文件，如果配置文件有变化，则解析配置文件配置设备对应的项目。

### 3.2.2 接口设计

设备端接口设计成符合 HTTP 规范的标准接口，接口如下所示：

```
https://Server-ip:port/devs
```

设备端通过服务器提供的 devs 接口采用标准 post 的方式将自己所收集的数据传输给远端服务器，服务器收到数据之后再将其存储起来，同时 post 的过程中获取服务器的设备配置文件，并和本地文件相比较，如果没有变化的不进行设备配置，如果配置文件有变化的配置相应的项目。

## 3.3 客户端

### 3.3.1 功能设计

在本文中客户端的设计主要采取使用者直接通过浏览器软件登录远程服务器 Web 进行进行人际交互的方式进行，但是考虑到后期可能会有需要将设备管理嵌入到 PC 端软件或者手机端 APP 的需求，设计时预留客户端接口，通过客户端接口，可以获取对应的设备节点的信息，并且通过该接口也可以设置对应的网关设备。基本功能如图 3.3 所示。

### 3.3.2 接口设计

客户端接口同样也是设计成符合 HTTP 规范的标准接口，接口如下所示：

```
https://Server-ip:port/clients
```

服务器端 clients 接口提供给客户端访问的接口，客户端通过 clients 访问指定的设备，获取设备信息。

## 第四章 详细设计

本章的协议设计是针对上一章所设计的通讯接口的补充，更加详细描述了通讯接口传输的数据内容和格式。

### (1) 设备端通讯协议

设备端与服务器通信的协议格式如下：

```
{
  "DevId" : id-number,
  "Status" : status-id,
  "name" : "Dev name",
  "Discrip": "Dev discription",
  "locate" : "position",
  "time" : "time",
  "date" : "xxxx-xx-xx",
  "data" : [
    {
      "name" : "son dev name",
      "status" : status-id,
      "discrip": "Dev discription",
      "data" : [],
    },
    ...
  ]
}
```

通信协议格式采用标准的 json 格式，协议内部需要包含设备状态信息和挂载的子设备状态信息。

### (2) 客户端接口

客户端接口则是通过 client 获取从服务器获取设备相关的数据和信息，接口协议如下所

示:

```
{
  "target-id" : id-number,
  "Status" : status-id,
  "name" : "Dev name",
  "Discrip": "Dev discription",
  "locate" : "position",
  "time" : "time",
  "date" : "xxxx-xx-xx",
  "data" : [
    {
      "name" : "son dev name",
      "status" : status-id,
      "discrip": "Dev discription",
      "data" : [],
    },
    ...
  ]
}
```

### (3) 即时通讯协议

客户端通过即时通讯接口协议是 rtty 框架提供的一个直接访问设备的方式,这里参考 rtty 框架提供的协议,如下所示:

```
curl -k https://server-ip:5912/cmd -d '{"devid":"test","username":"test",
  "password":"123456","cmd":"ls","params":["/"],"env":{}}'
```

## 第五章 总结

整个网关网络管理还是比较纷繁复杂的，主要是因为设备分散，且网关后面又连接了一定数量的子设备，且网关一般是位于内网，需要穿透公网对网络设备进行网络管理和配置，本文基于功能需求的考虑，结合将要使用的实际环境，参考了 `rtty` 和 `rttys`<sup>[1]</sup> 等开源反向代理程序框架进行方案设计。

## 参 考 文 献

- [1] <https://github.com/zhaojh329/rtty>
- [2] <https://github.com/zhaojh329/rttys>
- [3] <https://tools.ietf.org/html/rfc5343>
- [4] Mark A.Richards . 雷达信号处理基础 (第二版) [M]. 2008.