**CIT操作设计文档**

# 一、Cit文件组成

波形数据由文件头和数据块两部分组成。

## 1、文件头

文件头中包括文件信息、通道信息和补充信息。文件信息块长度固定，通道定义块的长度根据各系统情况变长，补充信息块的长度各系统自定义。

### 1.1文件信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 存储长度  （字节） | 说明 |
| 1 | iDataType | integer | 4 | 文件类型：1轨检、2动力学、3弓网， …. |
| 2 | sDataVersion | String[20] | 1+20 | 文件版本号，用X.X.X表示 第一位大于等于3代表加密后,只加密数据块部分 |
| 3 | sTrackCode | String[4] | 1+4 | 线路代码，同PWMIS |
| 4 | sTrackName | String[20] | 1+20 | 线路名 英文最好 |
| 5 | iDir | integer | 4 | 行别：1上行、2下行、3单线 |
| 6 | sTrain | String[20] | 1+20 | 检测车号，不足补空格 |
| 7 | sDate | String[10] | 1+10 | 检测日期：yyyy-MM-dd |
| 8 | sTime | String[8] | 1+8 | 检测起始时间：HH:mm:ss |
| 9 | iRunDir | integer | 4 | 检测方向，正0，反1 |
| 10 | iKmInc | integer | 4 | 增里程0，减里程1 |
| 11 | fkmFrom | Single/float | 4 | 开始里程 |
| 12 | fkmTo | Single/float | 4 | 结束里程，检测结束后更新 |
| 13 | iSmaleRate | integer | 4 | 采样数，（距离采样>0, 时间采样<0） |
| 14 | iChannelNumber | integer | 4 | 数据块中通道总数 |
| 长度合计 | | | 120 | 通道信息长度＝通道数×单通道定义长 |

### 1.2通道定义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 存储长度  （字节） | 说明 |
| 1 | sID | integer | 4 | 轨检通道从1～1000定义  动力学从1001~2000  弓网从2001~3000 。。。。。。。 |
| 2 | sNameEn | String[20] | 1+20 | 通道名称英文，不足补空格 |
| 3 | sNameCh | String[20] | 1+20 | 通道名称中文，不足补空格 |
| 4 | fScale | Single/float | 4 | 通道比例 |
| 5 | fOffset | Single/float | 4 | 通道基线值 |
| 6 | sUnit | string | 1+10 | 通道单位 |
| 长度合计 | | | 65 | 通道内容待补充，但应包括公里、采样数 |

### 1.3补充信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 存储长度  （字节） | 说明 |
| 1 | iStandby | integer | 4 | 后续预留空间长度，可自行分配 |

## 2、数据块

数据块按照定义的通道信息填写，每个通道数据均使用short数据类型，存储长度为2个字节。读取出来的数值按照i/fScale+fOffset解析。数据记录统一采用公里、采样数，加上数据要素。数据块部分数据用异或(与128进行)方式加密,即每个字节和128进行异或操作,然后存取。

# 二、Cit操作设计

## 1、类设计

类包括实体类、偏移量类、加密类以及操作类。

### 1.1实体类

实体类分为文件信息类、通道定义类以及里程标类，详情如下：

文件信息类【FileInformation】（即文件开头的那120个字节的数据块）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 描述 | 类型 | 备注 |
| 1 | iDataType | 文件类型 | int | 1轨检、2动力学、3弓网 |
| 2 | sDataVersion | 文件版本号 | string |  |
| 3 | sTrackCode | 线路代码 | string |  |
| 4 | sTrackName | 线路名 | string |  |
| 5 | iDir | 行别 | int |  |
| 6 | sTrain | 检测车号 | string |  |
| 7 | sDate | 检测日期 | string |  |
| 8 | sTime | 检测起始时间 | string |  |
| 9 | iRunDir | 检测方向 | int |  |
| 10 | iKmInc | 增里程0，减里程1 | int |  |
| 11 | fkmFrom | 开始里程 | float |  |
| 12 | fkmTo | 结束里程 | float |  |
| 13 | iSmaleRate | 采样数 | int |  |
| 14 | iChannelNumber | 数据块中通道总数 | int |  |

通道定义类【ChannelDefinition】（每个通道都有自己的定义，每个定义65字节）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 描述 | 类型 | 备注 |
| 1 | sID | 通道Id | int |  |
| 2 | sNameEn | 通道名称英文 | string |  |
| 3 | sNameCh | 通道名称中文 | string |  |
| 4 | fScale | 通道比例 | float |  |
| 5 | fOffset | 通道基线值 | float |  |
| 6 | sUnit | 通道单位 | string |  |

里程标类【Milestone】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 描述 | 类型 | 备注 |
| 1 | mKm | 公里 | float |  |
| 2 | mMeter | 米 | float |  |
| 3 | mFilePosition | 位置 | long |  |

### 1.2偏移量类

偏移量类包括文件信息偏移量和通道定义偏移量类和整个文件头的偏移量类。

文件信息偏移量类【DataHeadOffset】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 描述 | 类型 | 备注 |
| 1 | DataType | 文件类型偏移位置 | int | 0 |
| 2 | DataVersion | 文件版本号偏移位置 | int | 4 |
| 3 | TrackCode | 线路代码偏移位置 | int | 25 |
| 4 | TrackName | 线路名偏移位置 | int | 30 |
| 5 | Dir | 行别偏移位置 | int | 51 |
| 6 | Train | 检测车号偏移位置 | int | 55 |
| 7 | Date | 检测日期偏移位置 | int | 76 |
| 8 | Time | 检测起始时间偏移位置 | int | 87 |
| 9 | RunDir | 检测方向偏移位置 | int | 96 |
| 10 | KmInc | 增减里程偏移位置 | int | 100 |
| 11 | kmFrom | 开始里程偏移位置 | int | 104 |
| 12 | kmTo | 结束里程偏移位置 | int | 108 |
| 13 | SmaleRate | 采样数偏移位置 | int | 112 |
| 14 | ChannelNumber | 通道总数偏移位置 | int | 116 |
| 15 | ChannelDef | 通道定义数据偏移位置 | int | 120 |

通道定义偏移量类【DataChannelOffset】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 描述 | 类型 | 备注 |
| 1 | ID | 通道Id偏移位置 | int | 0 |
| 2 | NameEn | 通道名称英文偏移位置 | int | 4 |
| 3 | NameCh | 通道名称中文偏移位置 | int | 25 |
| 4 | Scale | 通道比例偏移位置 | int | 46 |
| 5 | Offset | 通道基线值偏移位置 | int | 50 |
| 6 | Unit | 通道单位偏移位置 | int | 54 |

文件头偏移量类【DataOffset】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 描述 | 类型 | 备注 |
| 1 | DataHeadLength | 文件信息长度 | int | 120 |
| 2 | DataChannelLength | 通道定义长度 | int | 65 |
| 3 | ExtraLength | 补充信息长度 | int | 4 |

### 1.3加密类

加密类【Encryption】

包括三个方法，供cit操作类中统一调用

1、对字节进行加密

2、对字节数组进行加密

3、判断是否加密

### 1.4操作类

Cit操作类主要分为读操作和写操作。

每个通道的每个采样数据在转换为可用的数据之前要先做解密（如果加密的话）和转换（i/fScale+fOffset）

一、**读操作**

1、**读取文件信息**

[ 读取整个文件信息（即开头的120个字节）]

根据传入的文件名路径，从头读取整个文件信息的长度，即读取到整个的文件信息的字节流，然后再将读到的字节流转换成文件信息类型。

[ 读取文件信息的具体字段 ]

将读取的位置偏移到该字段的偏移位置，然后读取该字段的长度，然后转换成具体信息。

比如：读取线路名，将读取的位置Position赋值线路名的偏移位置30，然后读取线路名的字节长度20字节，最后转化成具体的值。

2、**读取通道定义信息**

先读取到文件信息中的通道总数，然后根据通道总数乘以一个通道定义所占的字节数(65个字节)，然后去读取整个的通道信息字节块，然后转化成可识别的通道信息。

3、**读取补充信息**

先读取到该cit的文件信息中的通道总数，然后将文件的位置偏移到补充信息的位置（即文件信息的长度+通道的长度\*通道总数）

4、**读取里程标信息**

（1）读取全部的里程标信息，先获取到该cit的数据块的起始位置和结束位置，

（2）然后计算出采样点个数（即（结束位置-开始位置）/一个采样点的大小），一个采样点的大小由通道总数\*2得到。

（3）对采样点个数进行循环，读取每个采样点的Km和m通道，然后把值给里程标类对象。

（4）最后返回List< Milestone>。

5、**读取所有通道数据**

（1）先获取到要读取的数据块的起始位置和结束位置

（2）计算出要读取范围的采样点的个数, （即（结束位置-开始位置）/一个采样点的大小），一个采样点的大小由通道总数\*2得到。

（3）定义List，list中的每个元素是一个double数组，每个数组就是一个通道的数据。有多少个通道，List的长度就是几。

（4）循环读取采样点，每读取一个采样点（包含多个通道的数据），将每个通道的值相应赋值给List集合中的每个数组的对应元素，注意从通道中取出的数据要进行转换，即（值/通道比例+通道基线值）

（5）最后返回List<double[]>。

这个有多种变种形式，比如：

（1）所有数据

（2）开始位置、结束位置

（3）开始位置、采样点个数

（4）开始里程、结束里程

最终都要调用底层这个算法方法。

6、**读取单个通道数据**

（1）先获取到要读取的数据块的起始位置和结束位置

（2）计算出要读取范围的采样点的个数, （即（结束位置-开始位置）/一个采样点的大小），一个采样点的大小由通道总数\*2得到。

（3）定义double数组，数组中的值就是该通道的数据。

（4）循环读取采样点，每读取一个采样点（包含多个通道的数据），找到该通道的位置将字节流转换成值，注意从通道中取出的数据要进行转换，即（值/通道比例+通道基线值）

（5）最后返回double[]。

这个有多种变种形式，比如：

（1）所有数据

（2）开始位置、结束位置

（3）开始位置、采样点个数

（4）开始里程、结束里程

最终都要调用底层这个算法方法。

**二、写操作**

**1、写入文件信息**

（1）首先将文件信息对象的每个值转换成流的形式

（2）然后写入到指定的文件中，位置从0开始

**2、写入通道定义信息**

（1）首先将通道信息列表对象的每个值转换成流的形式

（2）然后写入到指定的位置，如果和写入操作一起执行，直接写入即可，如果单独写入，先计算偏移位置，然后写入。

**3、写入补充信息**

（1）首先将补充信息转换成流的形式

（2）然后写入到通道定义信息的后面。

**4、写入通道数据**

（1）写入的通道数据是以List<double[]>的形式载入

（2）定义一个Byte的集合，即List<Byte[]>，用于存放通道数据的字节

（3）通过List集合中一个double数组的长度，得到采样点个数

（4）对采样点个数进行循环

（5）首先判断是否加密，如果加密执行加密的操作，如果不加密执行不加密的操作

（6）获取每个采样点的每个通道的值，然后进行一下转换，(值-通道基线值)\*通道比例

（7）然后转换成字节数组存入到之前定义的Byte[]的集合中，（就是按照采样点的顺序放到List<Byte>中，等全部获取完毕后）

（8）循环结束后，一并写入文件中。