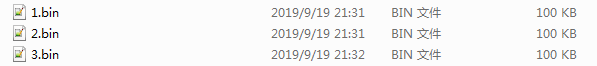
**VMM数据格式说明Ver1.1**

修改时间：20191009

1. **原始数据文件 以下均以RUN86数据为例。**

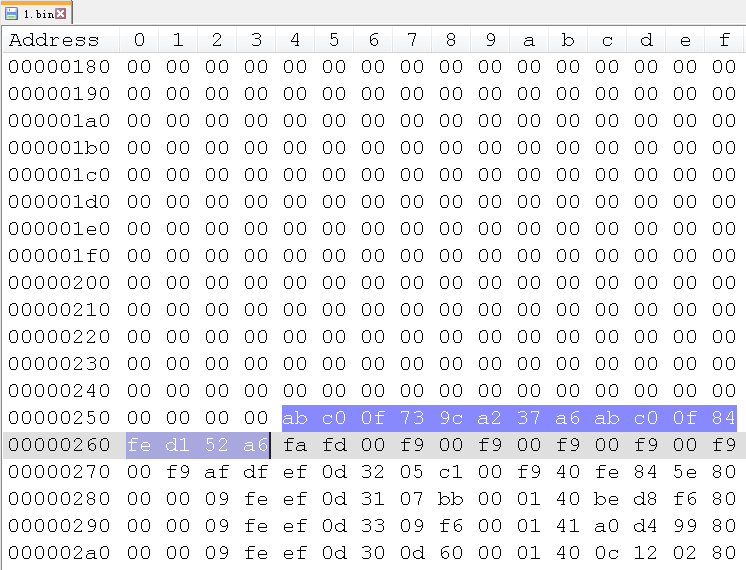


原始数据文件为二进制文件，命名规则为1.bin、2.bin、3.bin以此类推。文件个数没有限制，最少只有一个文件；单个文件大小也没有限制但都是1KB的整数倍。

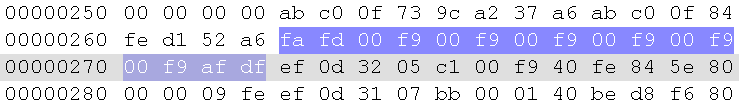
1. **三种数据帧格式**

原始数据文件以16进制显示。共有三种不同的数据帧格式，每种数据帧均为16字节。除此以外还有非法的数据帧需要剔除。

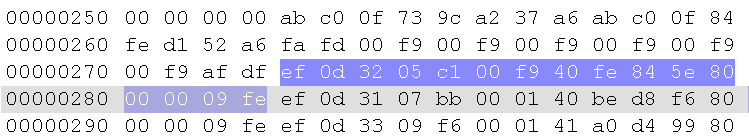
1. TDC数据帧，16字节，包含两组8字节的TDC数据（用途是作为两次事例的时间零点），每组以abc(h)为帧头但无帧尾，如下图高亮部分。目前可以直接判断到TDC数据帧后跳过它即可。以其中一组8字节数据ab\_c0\_0f\_73\_9c\_a2\_37\_a6(h)为例，将第二字节后4位与第三字节、第四字节前四位顺序拼接，组成00f7(h)，转为十进制数247(d)，名称为TDC\_TrigID；第四字节后四位与第五字节顺序拼接，组成39c(h)，转为十进制数924(d)，名称记为TrigBCID；第六字节前四位a(h)和后四位2(h)，分别做一次4bit格雷码转4bit二进制数的转换，然后再转换为十进制数，得到12(d)和3(d)，用后者减去前者，如果结果大于0则不变，如果结果小于0则加16，得到7(d)，将其乘以2.5，加上｛第七个字节37(h)转换成十进制数55(d)除以88｝，减去｛第八个字节a6(h)转换成十进制数166(d)除以88｝，保留两位小数，得到16.24，时间单位为ns，名称记为TDC\_Value。



1. TLU数据帧，16字节，如下图高亮显示，以fafd(h)作为帧头，以afdf(h)作为帧尾。中间共12个字节，TLU的数据实际上只有2个字节，这里让数据传输时重复六次得到12字节，主要为了加强其可靠性，同时也可以作为判断是否是TLU数据帧的一个标志。这里的TLU数据帧只需记下“重复六次”的00f9(h)转换为十进制为249(d)，名称为TLU\_TrigID。



1. VMM数据帧，16字节，如下图高亮显示其中一组，以ef0d(h)作为帧头，以fe(h)作为帧尾，中间共13字节原始数据。
2. 第三个字节，其范围30(h)~37(h)，其十进制数48(d)~55(d)，将其减去48，表示板子编号0~7，名称记为Board\_ID，例如图中为32h则表示Board\_ID=2。
3. 第四个字节后四位和第五个字节，顺序拼接共12bit，例如图中5c1(h)，将高低位翻转，对应83a(h)，其为格雷码，将12bit格雷码转换成12bit二进制数再转换至十进制数，得到4051(d)，记录此值，名称记为BCID。（BCID用来表示时间测量的粗计数）
4. 第六个字节和第七个字节，顺序拼接共16bit，例如图中00f9(h)，转换为十进制数，得到249(d)，记录此值，名称记为VMM\_TrigID。
5. 第八个字节，取其最低三位，例如图中40(h)取其bit2~bit0得到000（B），转换为十进制数，得到0，记录此值，名称记为VMM\_Chip\_ID。
6. 第九个字节，取其最高六位，例如图中fe(h)取其bit7~bit2，高低位翻转，得到111111（B），转换为十进制数为63(d)，记录此值，名称记为Channel\_ID。（本例不太好，以图中下一组数据为例，be(h)取其高六位，高低位翻转后得到十进制数为61(d)）
7. 第九个字节取其最低两位，即bit1~bit0，顺序拼接第十个字节，共10bit，例如图中fe84(h)，取其低10bit，将高低位翻转，得到0010000101（B）转换为十进制数，得到133(d)，记录此值，名称记为PDO，或者Q（实际就是ADC值，表示幅度信息）
8. 第十一个字节，例如图中5e(h)，将其高低位翻转，得到7a(h)，转换为十进制数，得到122(d)，记录此值，名称记为TDO（TDO用来表示时间测量的细计数）
9. 以上未提到的bit位，暂时无用。



1. **数据对照**

以RUN86中VMM数据的1.bin为例，最初分析的数据应当如下，可用作校验：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 分类 | TLU\_TrigID | TDC\_TrigID | TDC\_Value | TrigBCID | TDC\_TrigID | TDC\_Value | TrigBCID |
| 1 | TDC数据帧 |  | 247 | 16.24 | 1278 | 248 | 19.05 | 1278 |
| 2 | TLU数据帧 | 249 |  |  |  |  |  |  |
| 12 | TLU数据帧 | 1 |  |  |  |  |  |  |
| 20 | TDC数据帧 |  | 249 | 26.02 | 772 | 250 | 36.07 | 3484 |
| 21 | TLU数据帧 | 2 |  |  |  |  |  |  |
| 53 | TLU数据帧 | 3 |  |  |  |  |  |  |
| 60 | TDC数据帧 |  | 1 | 33.43 | 732 | 2 | 26.55 | 334 |
| 61 | TLU数据帧 | 4 |  |  |  |  |  |  |
| 74 | TLU数据帧 | 5 |  |  |  |  |  |  |
| 84 | TDC数据帧 |  | 3 | 25.11 | 3332 | 4 | 14.2 | 2782 |



1. **数据处理**
2. 每次数据有之前的部分缓存，在复位完成后，可以看到三种数据帧的Trig\_ID都是从1开始的，将此之前的数据丢弃。部分情况例外，例如不使用TLU时，Trig\_ID没有用处，因此没有进行复位操作。也有忘记复位，导致数据与触发号无法对齐的情况。这些情况的数据只能单独测试VMM本身。
3. 将TLU数据帧独立出来，只判断其是否连续即可，TLU\_TrigID如果不连续，说明丢失触发。
4. 如果TLU触发号一直是连续的，并且VMM有复位操作。那么我们可以认为VMM\_TrigID和TLU\_TrigID等效。如果不连续，例如TLU\_TrigID从1~99，101~200，那么VMM\_TrigID从1~99仍视为与TLU\_TrigID对应，而VMM\_TrigID=100对应TLU\_TrigID=101，VMM\_TrigID=101对应TLU\_TrigID=102以此类推，不过目前暂未发现这种情况。
5. VMM\_TrigID如果不连续，是正常的。在有触发而VMM没有探测到信号的情况下，会发生这种情况，并且VMM\_TrigID仍然正常工作。例如TLU\_TrigID从1~200连续，VMM\_TrigID从1~99，101~200，VMM\_TrigID仍然与TLU\_TrigID一一对应，只是ID=100的触发下没有数据，这是个效率的问题。
6. 时间测量分析第一步，先将TDC的数据帧中的TDC\_TrigID，找到与VMM\_TrigID相同的VMM数据帧，将两者所有信息绑定在一起。（将TDC\_Value和TrigBCID添加到对应的VMM数据帧即可，后续有用）