VCB-Studio 教程 04 BDRip 的制作流程

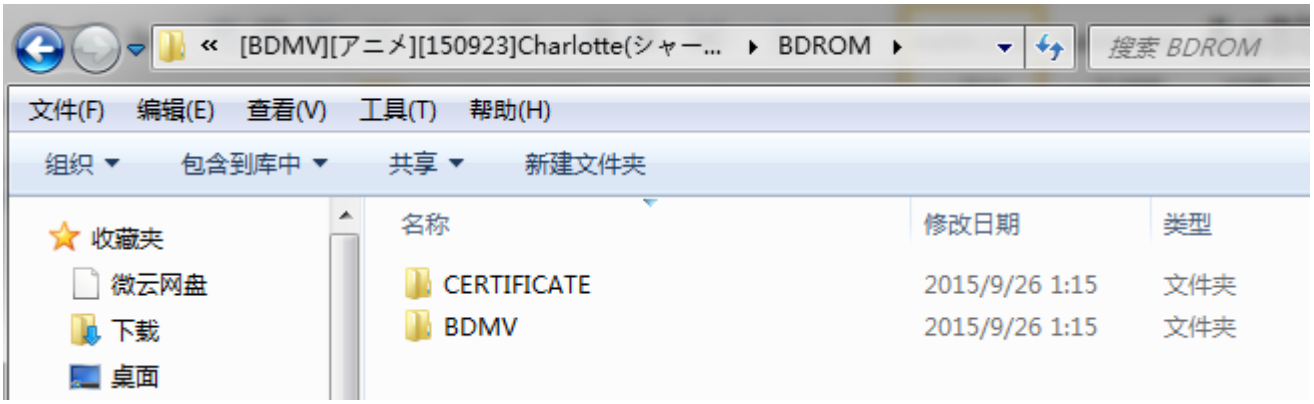
0. 前言

本教程的目的是作为 VCB-Studio 群内教程，按照 VCB-Studio 的标准流程，来讲述 BDRip 的制作。这只是 vcb-s 的习惯流程，针对 vcb-s 团队分工习惯而编写。而没有任何设定“业界标准”或标榜“最佳步骤”的意思。实际操作中，非 vcb-s 组员的读者没有必要完全参照，因为可能有很多步骤和规范，有更好、更简便的做法。

1. 认识 BDMV 目录结构

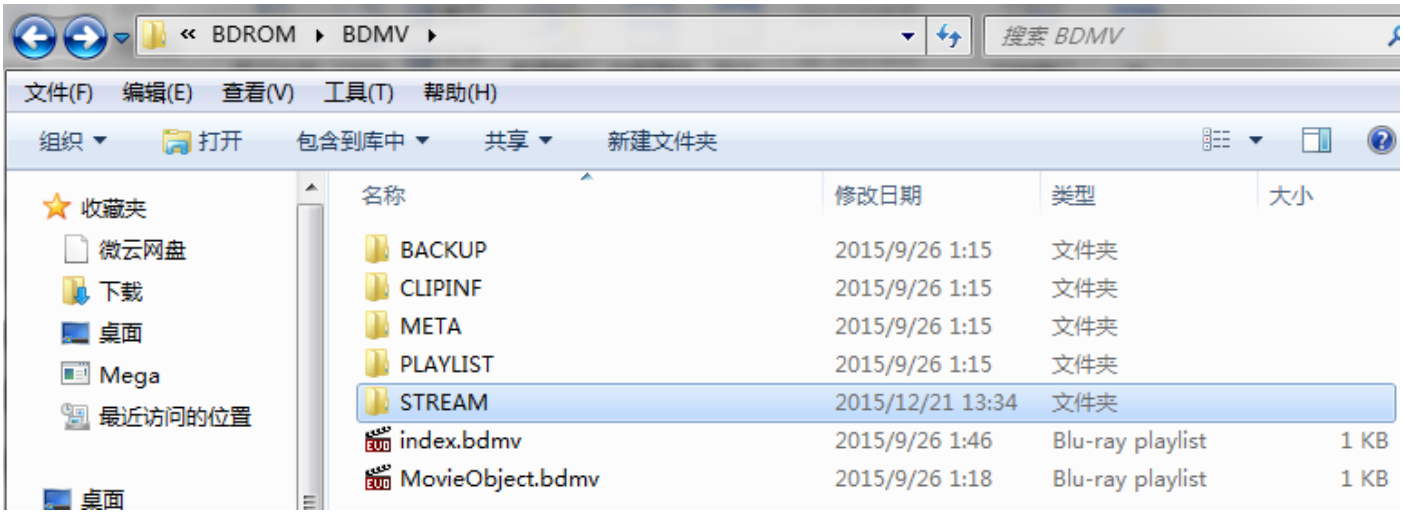
BDMV 因为抓取步骤和放流人士的区别，上层目录可能有各种不同的样子。然而，一定有一个目录包括两个子目录：BDMV 和 CERTIFICATE。这两个目录的上级，我们称为根目录。比如说这是 U2 上《四月是你的謊言 Finale Event》的蓝光，其根目录名称为 BD_VIDEO:

另一个例子是 Charlotte 的 Vol.1:

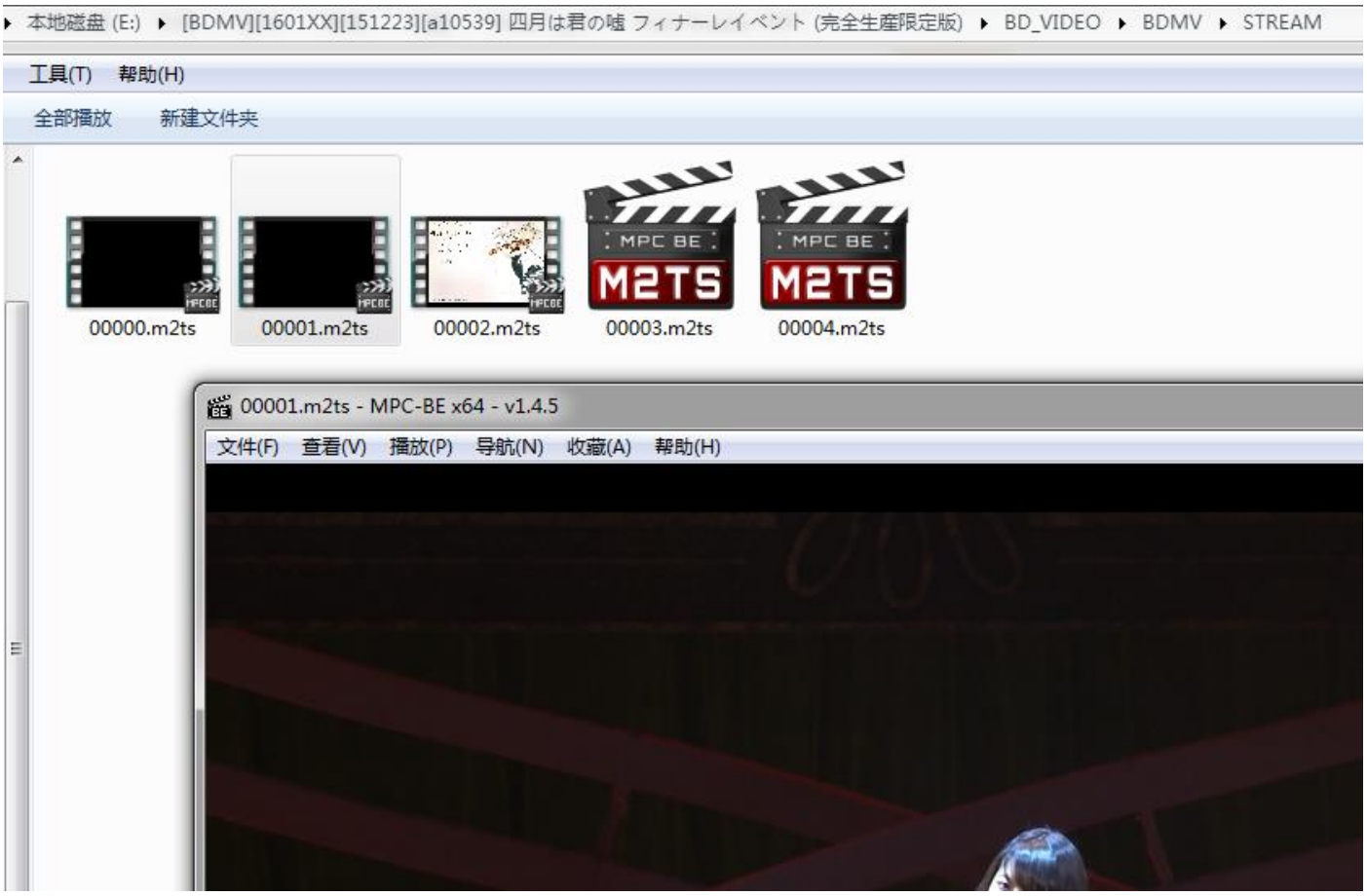


这里蓝光的根目录叫做 BDROM，可见不同蓝光，甚至不同卷之间，根目录的名称不固定，但是根目录下一定有 BDMV 和 CERTIFICATE 两个子目录。

BDMV 目录里的内容大致如下，其中重要的文件夹有两个，STREAM 和 PLAYLIST：

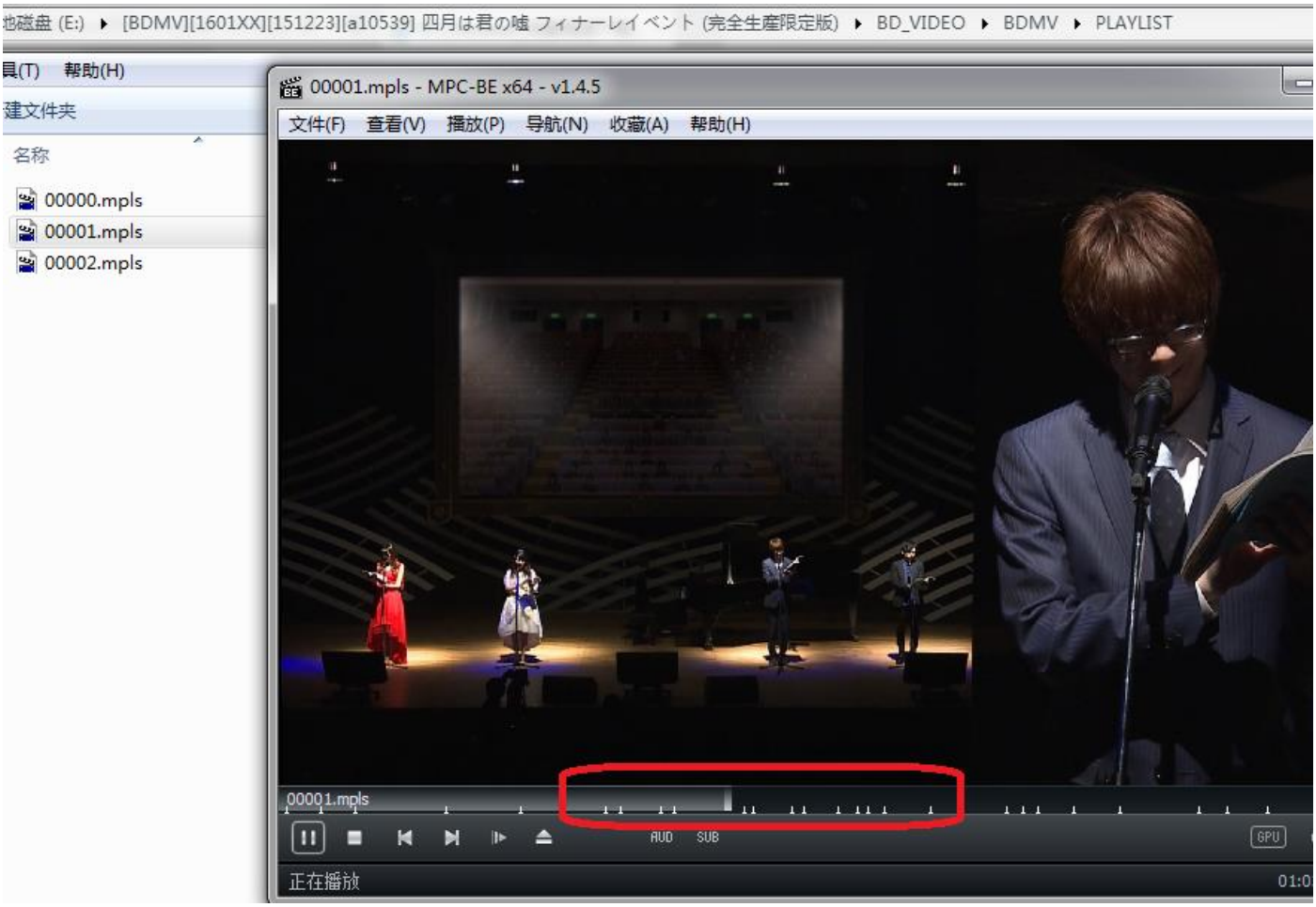


STREAM 文件夹就是蓝光视频所在的地方：



一般来说，m2ts 后缀的就是蓝光视频文件。有些不能播放的，多是菜单按钮等信息。

PLAYLIST 文件夹下的文件后缀为 mpls，是蓝光的播放列表信息，多半可以被播放器直接播放。它当中还储存着蓝光的章节信息（以后讲述提取章节信息，就是从这里获取的），可以在播放器中被显示：

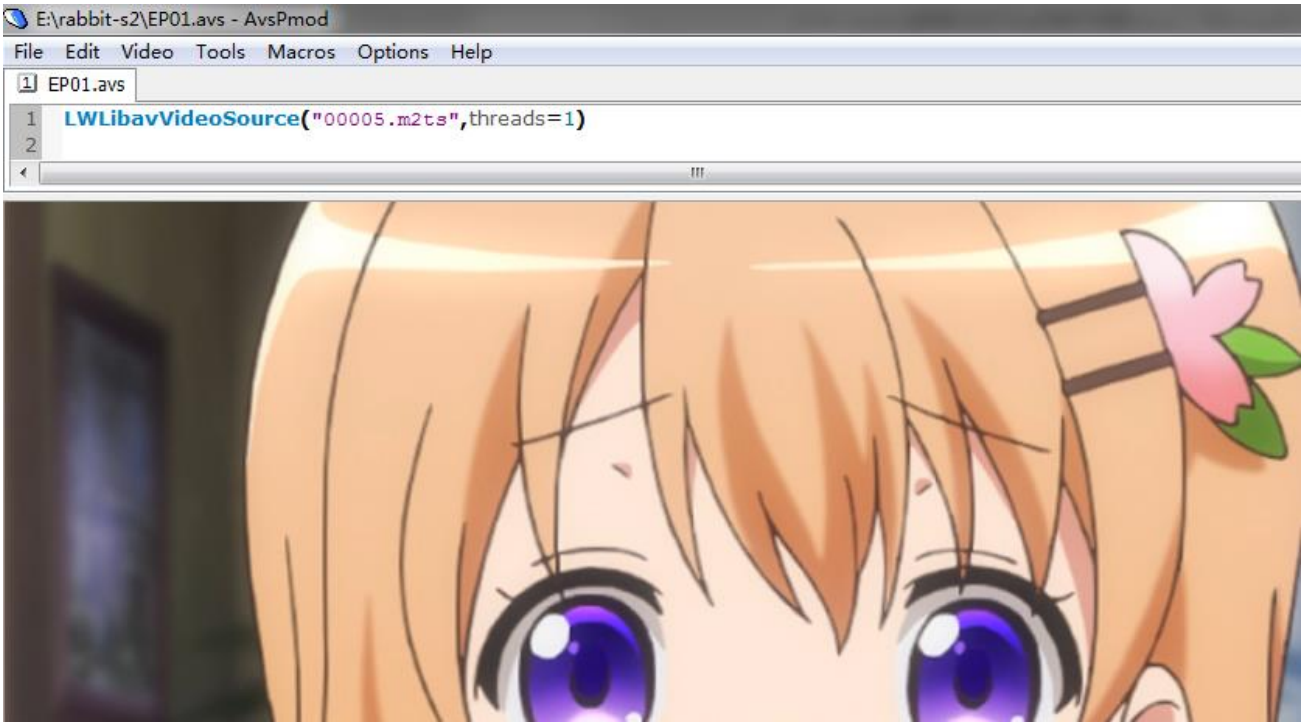


BDMV 中对 BDRip 的有效信息大概就这么多。

2. 用 avs/vs 载入视频

压制视频，我们一般是先用 avs/vs 载入视频，并做一定的处理。这份教程中我们只介绍最基本的载入视频的方法。
avs 的写法如下：
LWLibavVideoSource("00000.m2ts",threads=1)
就是用 Libav(LAV)分离并解码 00000.m2ts 的视频部分。文件名自己改；threads=1 是保证单线程解码，避免解码过程中出现未知问题（多线程解码在一些视频上容易出现问题）

用 avspmod 调试，效果如下：



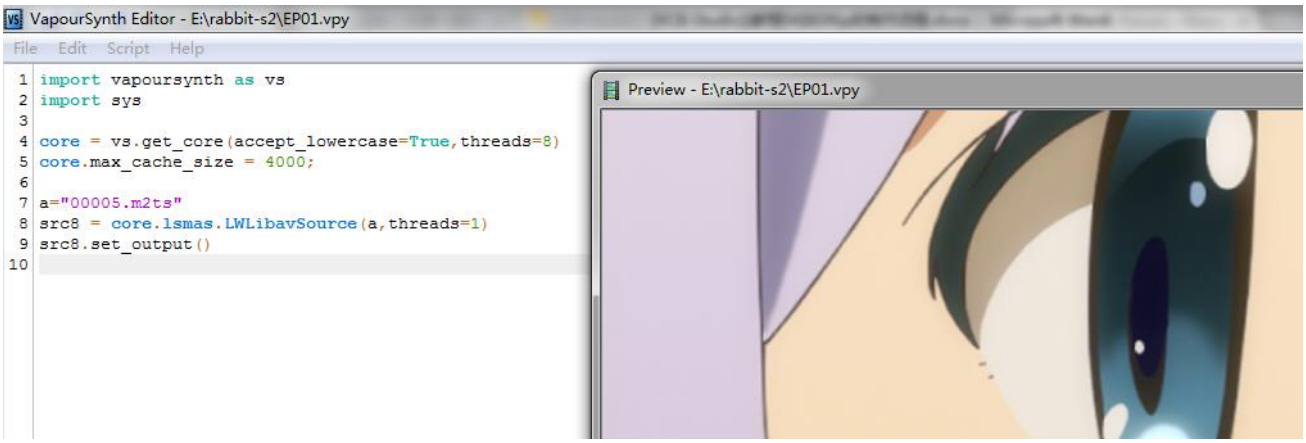
VS 的写法如下：

```
import vapoursynth as vs
import sys
core = vs.get_core(accept_lowercase=True,threads=8)
core.max_cache_size = 4000;

src8 = core.lsmas.LWLibavSource("00000.m2ts",threads=1)
src8.set_output()
```

最上面 4 行是在 Python 环境中定义 VS 的运行，threads 指定 vs 最多可以多线程优化的线程数，max_cache_size 则是可以分配的内存数。后两行是载入视频，并输出(set_output)

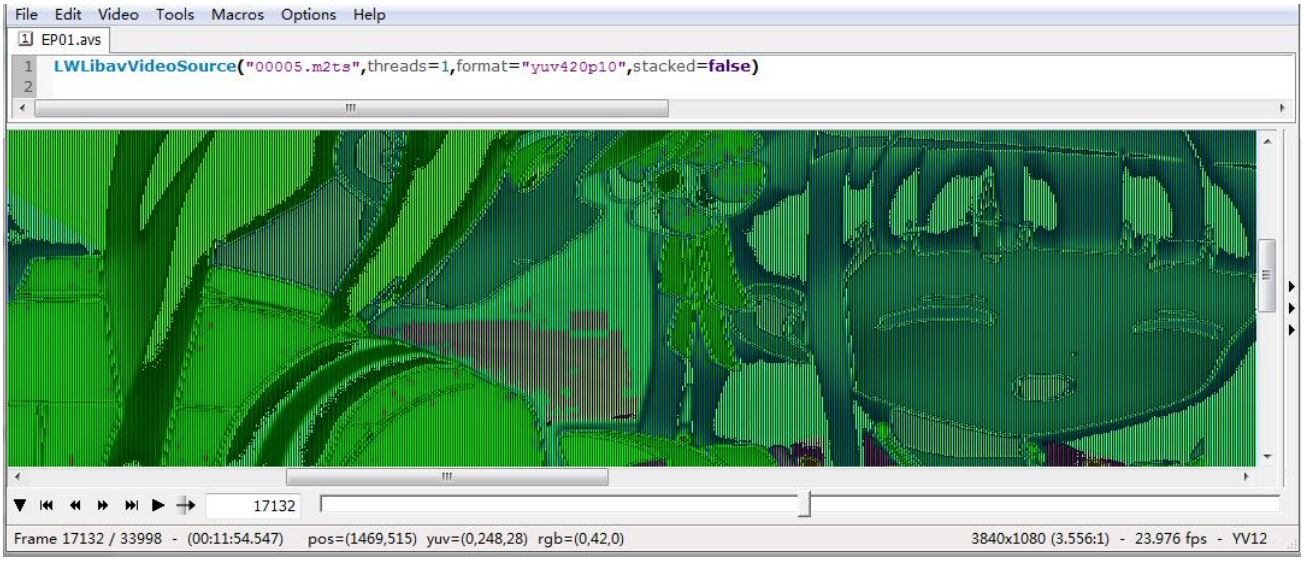
用 vseditor 预览，可以看到画面：



3. avs/vs 的纪律性检查

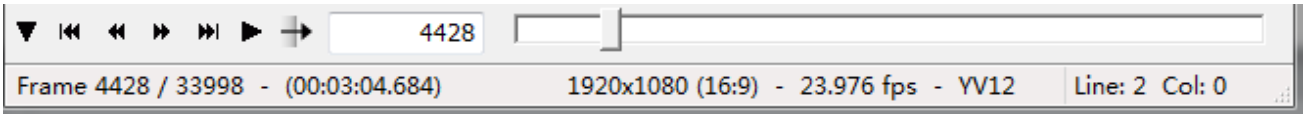
在日常压制任务中，调试 avs/vs 的时候，对 avs/vs 进行纪律性检查是非常重要的，不然很有可能跑了几十小时后，最终发现做错了，得全盘重做。无数血淋淋的经验表明：纪律性检查不会让你压片效率提升多少，但是做不到这个绝对会有一天让你效率变成负数。

1. 首先检查的因素是画面。简单说，要压第一集，你是不是载入正确的 m2ts，别压成了第二集。这个通过 avspmod/vseditor 能很容易的检查到。接着检查画面正常与否。对于 8bit 压制任务，avs/vs 都应该显示正常的画面，你可以检查是否有不正常的输出效果，花屏，等等。
如果是 10bit 压制任务，vcb-s 一般会使用 Interleaved 10bit 输出（以后会细讲），画面看起来是这样的：



这时候可以选择 Video - Bitdepth - Interleaved yuv420p10 or yuv444p10, avspmod 就可以显示正常。而 vs 中画面始终正常，不论是 8bit 还是 10bit。

2. 随后需要检查的是 avspmod/vseditor 的状态栏。这两个状态栏可以给你非常丰富的信息。首先是 avspmod 的：



从左到右，先是先是总帧数。一般一集 24 分钟，24p 帧率的番，总帧数在 35000 帧左右。如果差的太多（10%以上）一般就有问题。
然后是分辨率。8bit 输出的前提下，分辨率该是多少就是多少；Interleaved 10bit 输出，分辨率一般是横向*2，纵向不变。比如 Interleaved 10bit 输出，如果不调整，分辨率会显示 3840x1080；调整 bitdepth 之后则显示 1920x1080.

3. 接着需要检查的是帧率。avspmod 会显示帧率是 23.976，还是 29.970，或者是 59.940。一般不同帧率的东西不能用一套制作方案；比如说，用于 24p 正片的参数，一般不能直接用于 30fps 的特典。
现在动漫蓝光基本上以 24p 为主，然而，你一旦发现 30fps 特典，应该第一时间有警觉，这不能按照 24p

的做法去制作。

4. 最后要检查的是格式。YV12=YUV420, YV24=YUV444, RGB 则表示输出是 RGB。一般我们是不可能把 RGB 作为压制输出的，所以如果你拿到的 avs 输出是 RGB，那么就是调试开关忘记关闭了。

vs 的检查基本类似：

```
Frames: 33998 | Time: 0:23:38.000 | Size: 1920x1080 | FPS: 24000/1001 = 23.976 | Format: YUV420P8
```

跟 avs 的区别在于这几点：

1. vs 的 size 直接就是原生的分辨率，不管输出精度是 8/10/16;
2. Format 中附带了精度。所以检查 8bit/10bit，在 format 中检查。一般 8bit 压制输出是 YUV420P8，10bit 是 YUV420P10
3. 千万不要忘记检查帧率是 23.976/29.970 还是 59.940

一般拿到 avs/vs 脚本，或者自己改写，只有当你调试完毕，确认没问题，才可以开始下一步骤的操作。

4. x264/x265 的使用

avs 和 vpy 就位之后，就可以送压 x264/x265。一般来说，我们使用 64bit 版本。x264/x265 有 8bit/10bit 的区分，一般来说，我们使用的 x264，8bit/10bit 是不同的 exe（不同源码编译成不同的执行文件）；而 x265 官方版，8/10/12bit 是一个 exe，通过参数来控制输出；也有分开编译的第三方。（比如 Yukki Mod）

本系列教程中，我们一般认为 x264 有 8bit/10bit 之分；10bit x264 往往会以 x264_10bit 这种名称标注。

将预处理脚本转换为 YUV 数据，送给编码器，需要一个中间“转换器”。对于 avs，转换器为 avs4x264.exe、avs4x265.exe(这两个可以在 megui\tools\x264 和 x265 中找到)，对于 vs，转换器为 vspipe.exe，这个可以从 VapourSynth\Core64 里面找。

因为其高度相似性，avs4x264 和 avs4x265 可以被合二为一，比如最新的 tMod 版本就支持启动 x262/x264/x265(https://www.nmm-hd.org/newbbs/viewtopic.php?f=8&t=403&hilit=avs4x26x)。本教程中，我们还是区别对待。

所以压片的流程，如果跟播放的分离-解码-修改-渲染相比，那就是：
“转换器”执行 avs/vs 脚本，对（本身是压缩格式的）片源做分离、解码和修改，输出 YUV 数据；
x264/x265 把 YUV 数据编码成 AVC/HEVC 的视频压缩格式。

压片一般会写一个批处理。以下教程假设我们做 8bit 压制，就是预处理输出，和压制工具都是 8bit。
用 x264 压制 avs，使用 avs4x264.exe:
avs4x264 --x264-binary "x264_64.exe" --preset slow --crf 18 -o "EP01.264" "EP01.avs"
其中文件名（输入、输出、exe）可以自己改。--preset slow --crf 18 属于 x264 参数设置，后续教程会介绍。

如果是用 x265 压制的话，大同小异：
avs4x265 --x265-binary "x265_64.exe" --preset slow --crf 18 -o "EP01.265" "EP01.avs"
同理，--preset slow --crf 18 属于 x265 的参数设置。

使用 tMod 版本，直接把--x264-binary 和--x265-binary 合并为--x26x-binary

如果是 vs 压制，则使用 vspipe.exe:
vspipe.exe --y4m "EP01.vpy" - | "x264_64" --demuxer y4m --preset slow --crf 18 --output "00010.264" -
vspipe.exe --y4m "EP01.vpy" - | "x265_64" --y4m --preset slow --crf 18 --output "00010.265" -

同样，输入输出和编码器名称自己设定。编码器的名称，exe 后缀可以省略。

输入文件名没啥好说的，一般为.avs 或者.vpy;
输出文件名，用 x265 的老老实实把后缀设置为.265，用 x264 则建议设置为 mkv。输出 mkv 的好处在于，可以直接预览半成品，方便检查。如果是准备封装成 mp4 成品，后缀名也可以设置为.mp4。

x264 会根据后缀名，自动选择输出文件是否自带容器。.mp4 则用 MP4 容器，.mkv 则用 mkv，否则就是纯 AVC 数据流。

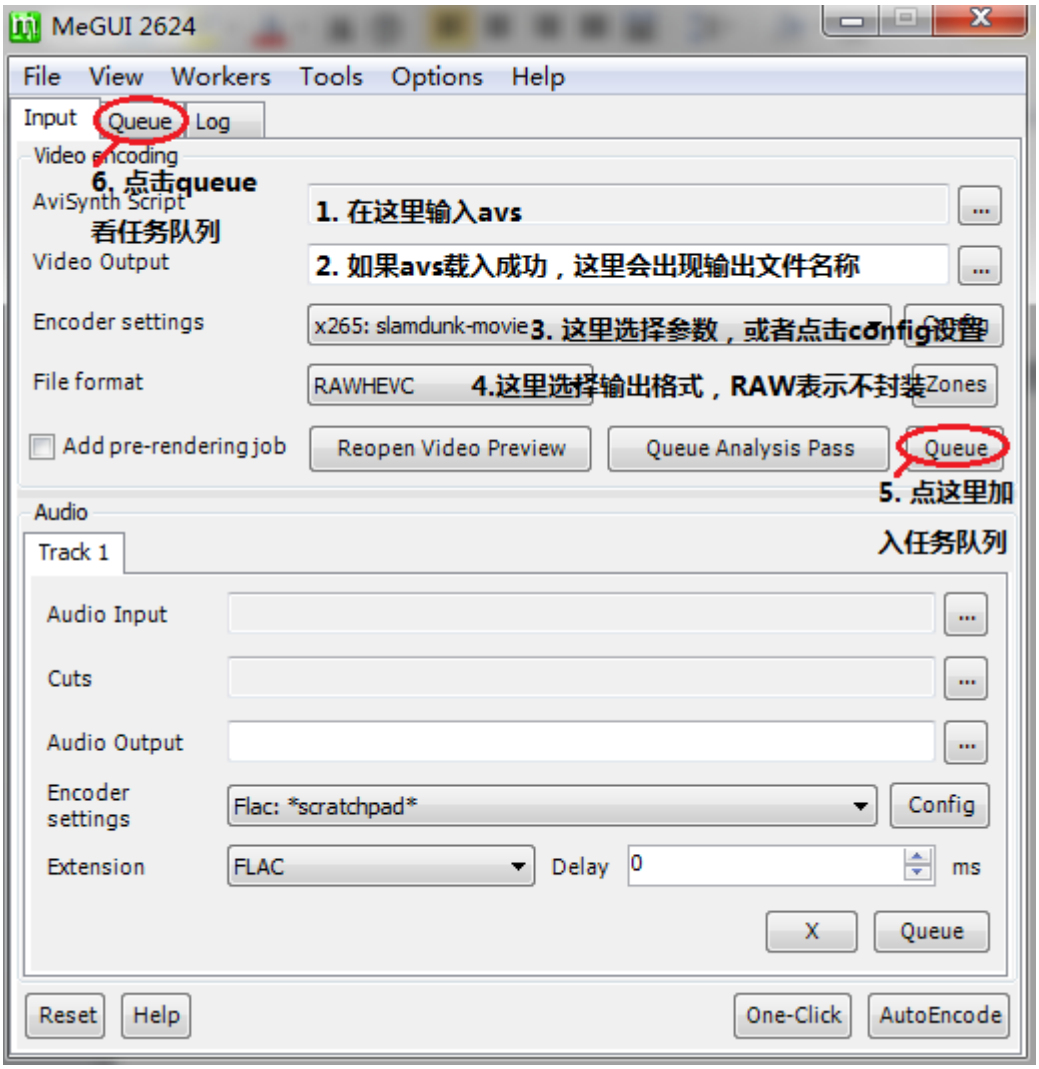
x264 输出的 mkv，并非是最标准的封装，表现为在一些时候无法拖动进度条，花屏等。经过正规封装之后（后文会说）一般问题解决。

如果需要进行 10bit 的压制，x264 的做法是把 exe 换成 10bit 版本的（例如 MeGUI 自带的 x264-10b_64.exe）；x265 的做法是参数中加一个 -D 10 表示输出精度为 10bit。、

以后的教程中我们还会讲原生高精度的输出，如果 avs 的输出是 10bit/16bit 精度，参数中需要加一个 --input-depth 10 或者--input-depth 16，来保证 avs4x264 能正确的把高精度颜色喂给编码器。vs 压制无需照顾这一点，因为 vs 在设计的时候就有原生的高精度支持。

5. 用 MeGUI 压制 avs 任务

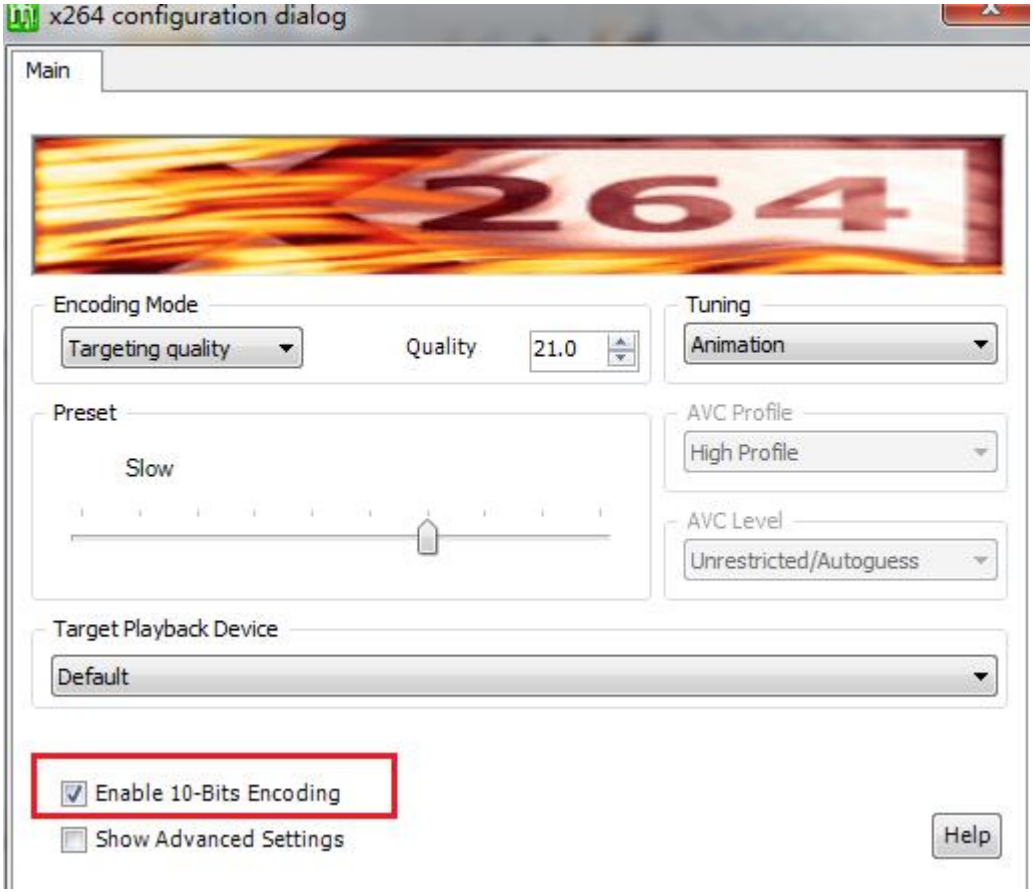
如果是 avs 输入，上述 x264/x265 的使用可以通过 MeGUI 实现：



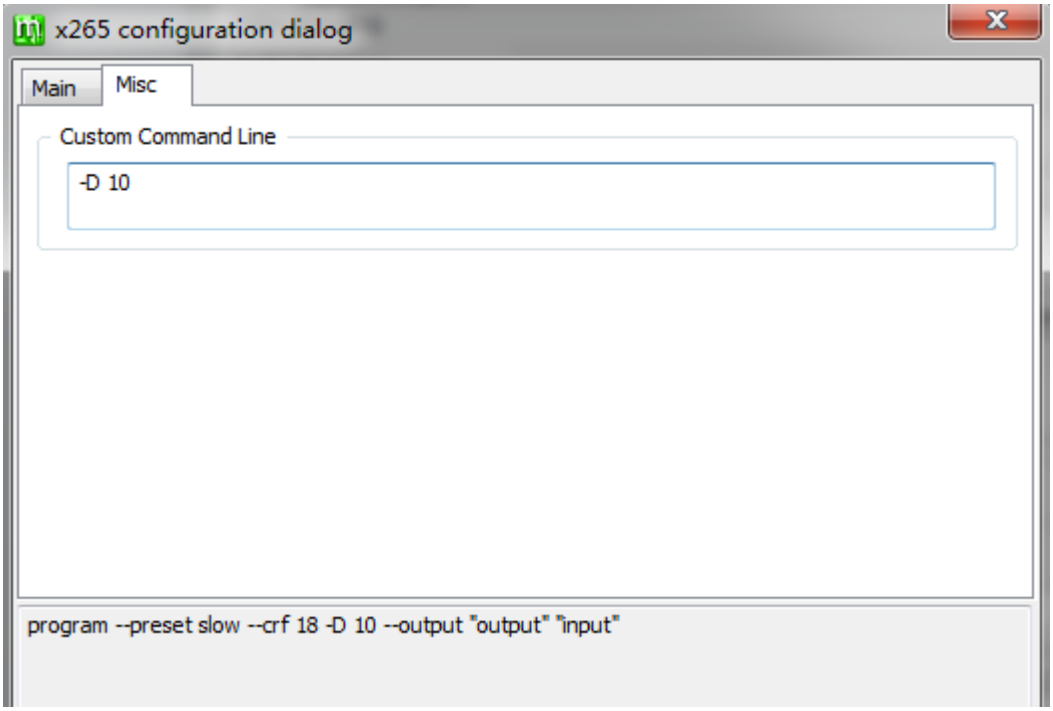
第 1 步中，如果 avs 有问题，MeGUI 会弹窗报错。

第 3 步中，保存好的参数文件，一般是.xml 格式，比如 x265_dp_ slamdunk-movie.xml。对于给定的 xml 文件，正确的载入方式是**在关闭 MeGUI 的前提下**，把 xml 文件放入 MeGUI\allprofiles\x265 或者 MeGUI\allprofiles\x264 中（取决于这是 x264 还是 x265 的参数）

手写参数的前提下，如果你用 x264，要进行 10bit 压制，你需要在设置中勾选 Enable 10-bits Encoding:



如果你用 x265，则需要在设置中手动写入-D 10:

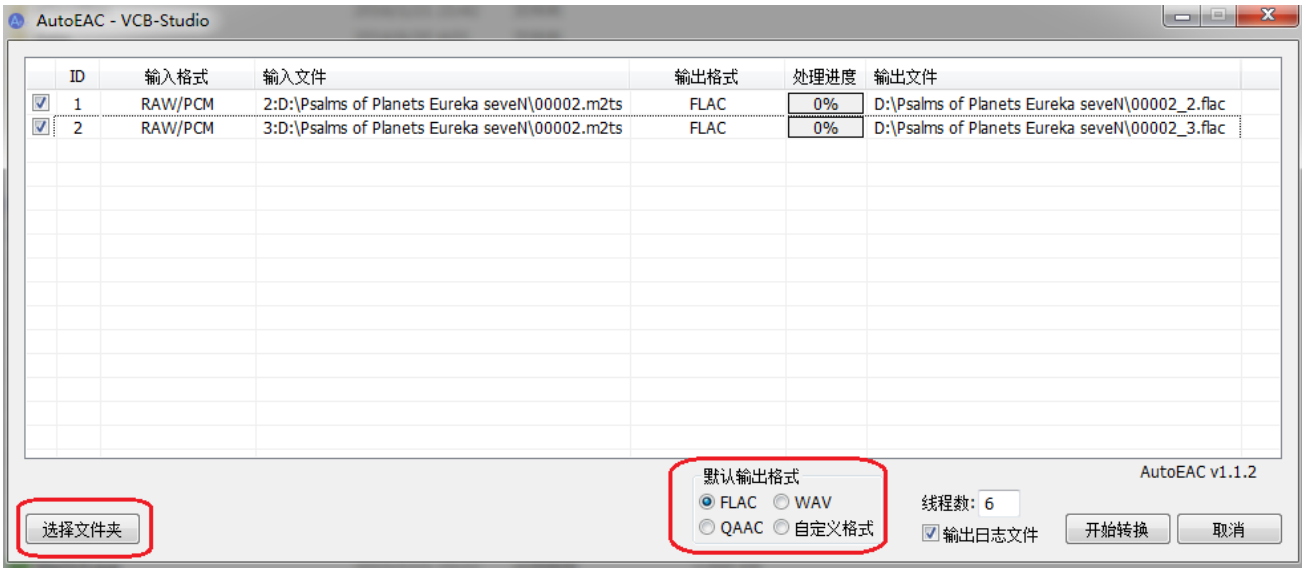


6. 抽取原盘音轨

抽取原盘音轨，最标准的方法是在 MeGUI 中，使用 Tools-HD Stream Extractor 来调用 eac3to 进行抽取。这部分知识我们会在后续教程中详细说，本教程则教大家使用 VCB-Studio AutoEAC 来便捷抽取。

把 AutoEAC 解压到某个地方，就可以使用了。AutoEAC 自带了 eac3to 和 qaac，如果你已经按照上文设置了 MeGUI，可以只解压 exe 和 ini 放到 megui 根目录下，这样 AutoEAC 会自动调用 MeGUI 的 tools。

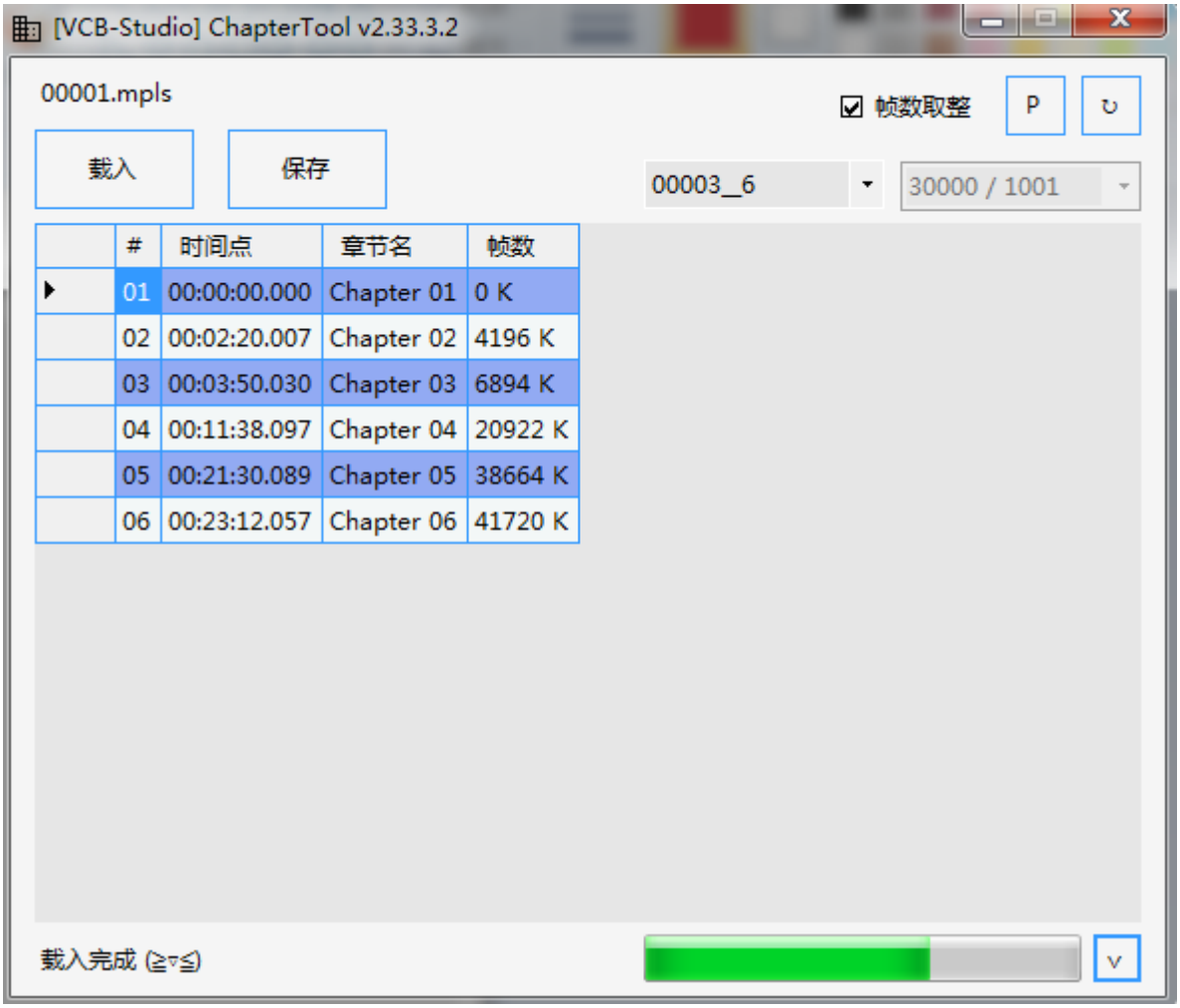
AutoEAC 的使用非常简单，左下角选择 BDMV 目录，再选择输出格式就好。如果不想要某条轨道，可以通过最前面的选择框取消，或者直接双击删除轨道。



AutoEAC 输出的 AAC，使用 qaac --cvbr 128K 编码。

7. 抽取原盘章节

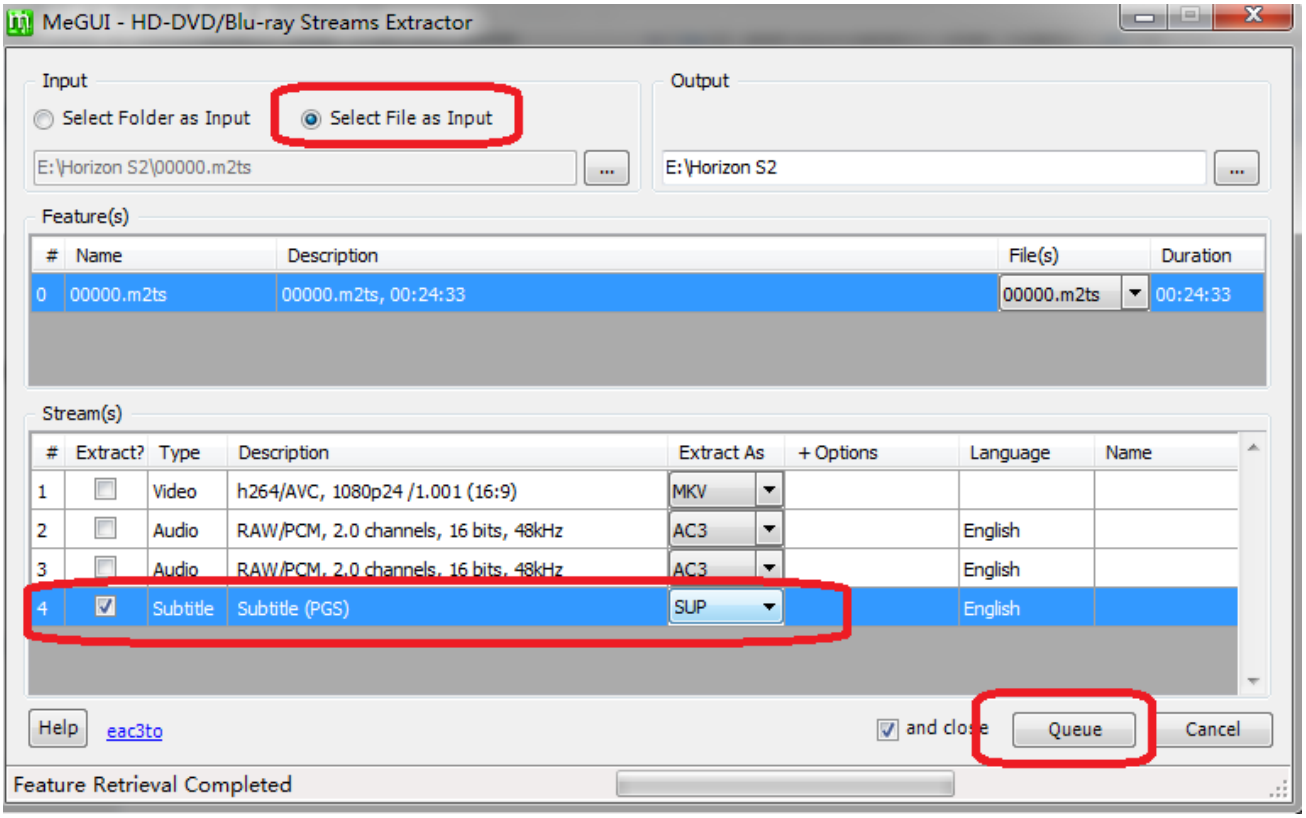
原盘章节的抽取可以用 Chapter Tools 来制作，这里只讲述它最简单的用法：



点击载入，选择 mpls（前文说过，这玩意就是章节文件所在处）
载入后，点击保存，就可以去 playlist 文件夹下找对应的 txt

8. 抽取原盘字幕

原盘字幕的抽取，需要用 MeGUI-tools-HD Stream Extractor：



如图所示，选择 Select File As Input，选中 m2ts。MeGUI 会自动分析，分析完毕后状态栏显示 Feature Retrieved Completed。

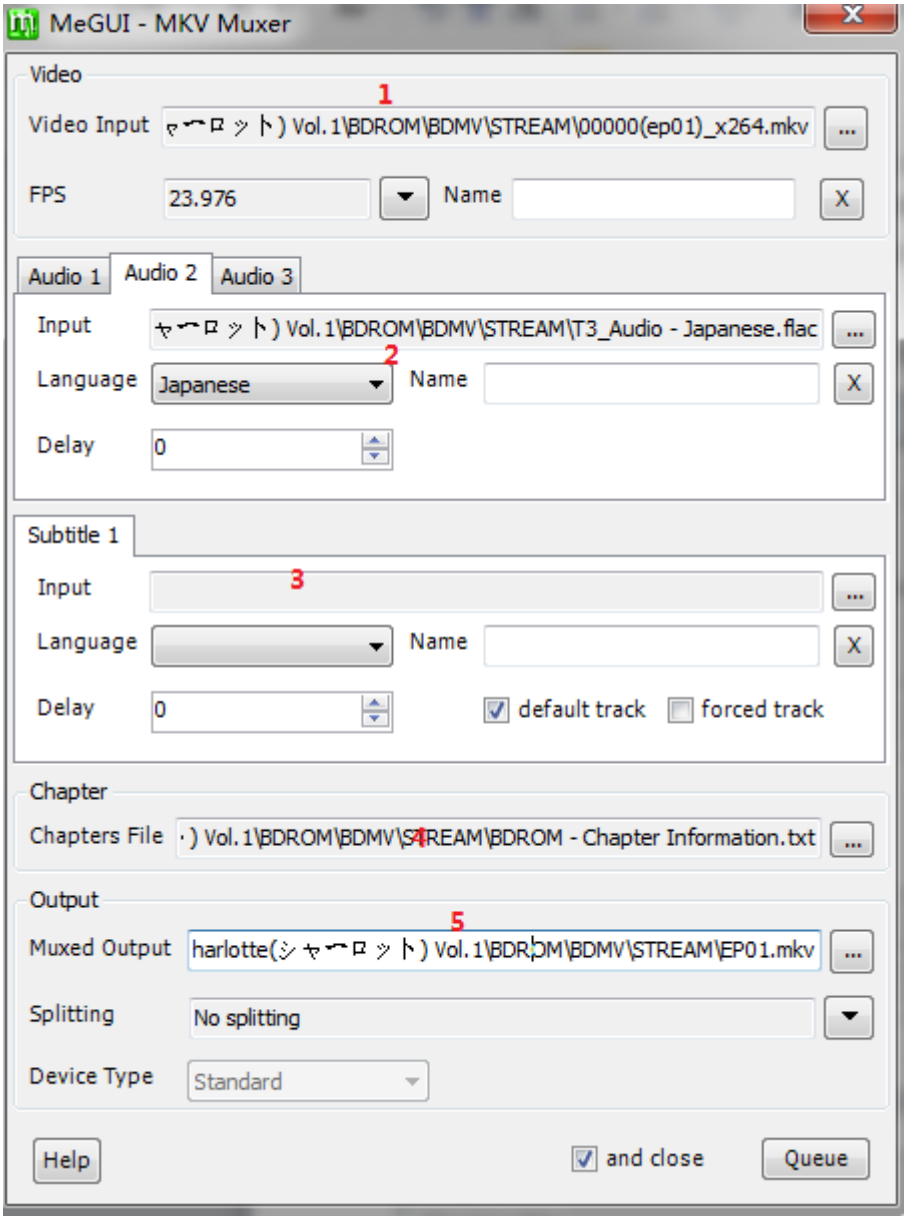
然后就选中 Subtitle，queue，执行。抽取出来的字幕后缀是*.sup，在 m2ts 同目录中。

这个工具其实也可以用来抽音轨，效果和操作逻辑跟 AutoEAC 类似，都是调用 eac3to。但是切记不要用它抽取 aac。（因为 MeGUI 抽 aac 调用的是 NeroAAC，需要另外配置）

9. 成品的封装与检查

准备好压制完毕的视频、音频、章节和字幕之后，就可以着手准备封装成品了。MeGUI 提供了简单的 MP4/MKV 封装工具，在 tools-muxer 中。其他更高级工具，例如 mkvtoolnix，会在以后的教程中讲解。

下图以 MKV muxer 为例，MP4 的大同小异：



1：输入视频文件。注意封装 MKV 的时候，输入可以是 264/mp4/mkv，封装 mp4 则必须是 264/mp4。如果压制出来纯视频是 mkv 格式，则先要抽取纯 264 视频流。检查帧率，一般都是 24p 23.976，如果是 60fps 版就是 59.940。

2：输入音频。如果有多条音轨，输入第一条后，旁边自动出现第二条的位置，如图所示。

3：输入字幕（如果有），类似输入音频，然后设置音频语言信息，一般动漫么都是日语音轨，选 japanese，

如果有英配，中配，则选 English 或者 Chinese，诸如此类。

4：输入章节文件，一般为 txt 格式

5：设置输出名称。vcb-s 的规范一般以卷数-标号为准，比如 v1-00000.mkv，表明是 Vol.1 的 00000.m2ts 压制而来。设置好后就可以点击 Queue 加入队列，在队列中启动项目

封装好的成品，要进行再一次的检查，主要包括：

1. 播放效果是否正常，这时候的成品应该完全符合 BDRip 成品要求；
2. 章节是否正确，能否通过章节来实现跳段；
3. 音轨/字幕是否齐全，能不能正确播放，切换时候语言显示正确与否，是否与画面时间匹配；
4. mediainfo 中，分辨率、比例、多音轨字幕、bitdepth 等信息是否正确。

10. 成品的上传规范

VCB-Studio 压制组的成品一般是传到百度网盘，然后给整理组接手。百度网盘默认只允许最大 4GB 的文件，所以任何更大的文件必须通过 rar 分卷解决：

打包时候，有几点需要注意：

1. 压缩方式选储存。这种是不做任何压缩，直接复制。觉得想通过压缩包省点体积的不妨试试能省多少。（事实上，x264/x265 编码最后一步就是类似 7z 做的事情）
2. 分卷大小，选择 4GB
3. 添加恢复记录，这是为了防止传输过程中，数据损坏导致压缩包不能正常解压。有恢复记录（默认 3%）足够解决绝大多数时候的传输数据损坏。

其他时候对于小文件，我们更多使用 RapidCRC 来添加 md5/CRC32. 使用方法很简单，打开界面，把成品 mkv 拖进去，点击 Put CRC into Filename。软件会提示你是否把你选中的，所有不带 CRC32 后缀的文件打上后缀，点击 Yes，文件名后就自动带上了 CRC32，类似 v1-00005 [7D7C6223].mkv。

这个机制可以帮助整理组快速检查文件损坏。你可以试着把某个文件的 CRC32 改动下，然后再拖到软件中看看。

设置好了之后，就可以上传到度盘准备交作业了。