信号处理流程及各通道信号基本特点

1.切换source 流程

Android（简称AN）响应key 后， Android 通过JNI 调用到Supernova（简SN）的MSrv _Control_DVB::SetInputSourceCmd()函数，在这个函数里，先是去 finalize 前一个player，然 后init 新的source 对应的player，紧接着配置av out，当新的player 检测到信号稳定之后， 会去根据当前的 Timing 去 set scaler window，从而把 video 画面显示在 panel 上。

切source 涉及的信号处理流程可以参考下图(里面内容比较多，大家可以先了解下)：

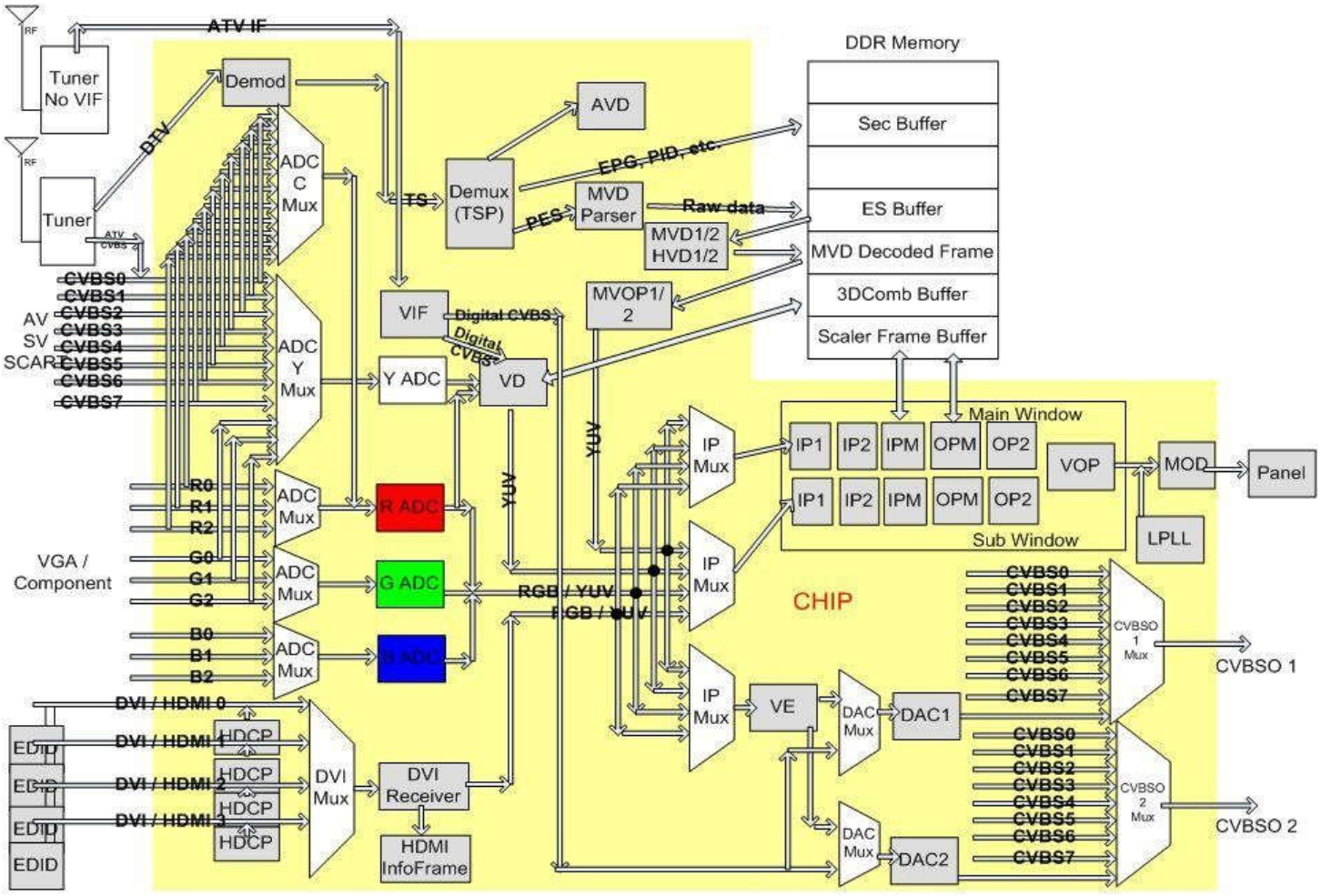


Figure 1

Source 切换Flow 如下：

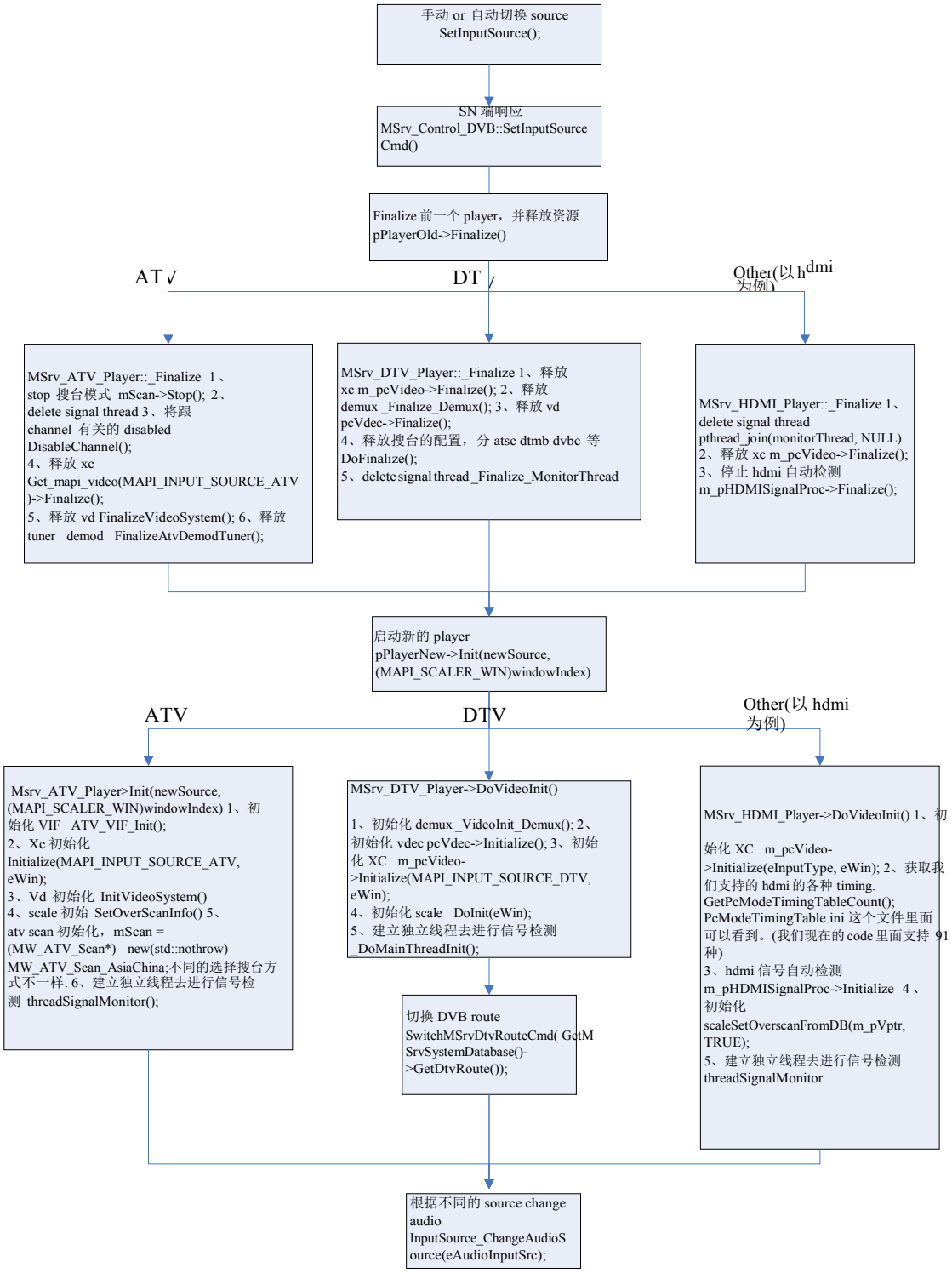


Figure 2

不同 **player** 需要初始化的部分

ATV:

1、初始化VIF
2、VD 初始化
3、初始化Audio
4、Scaler 初始
5、建立独立线程去进行信号检测

DTV:

1、初始化Demod
2、初始化 Demux
3、初始化 Vdec
4、初始化 Audio
5、初始化 scaler
6、Switch DTV route
7、建立独立线程去进行信号检测

HDMI/YPbPr/VGA:

1、初始化Audio
2、初始化 Scaler
3、建立独立线程去进行信号检测

AV out

Av out 包含 Scart out VE output。在通过 Scart 端子将视频输出到其他设备上时，切换
source 时需要对这个做处理：

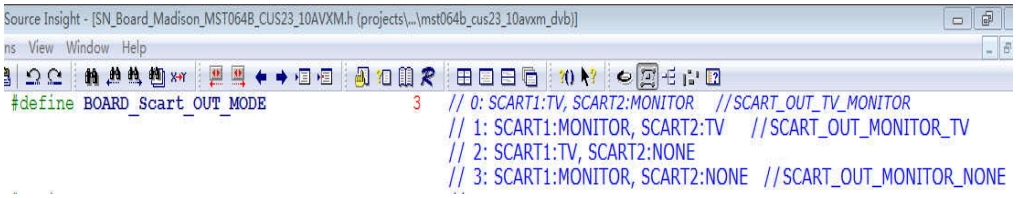


Figure.3

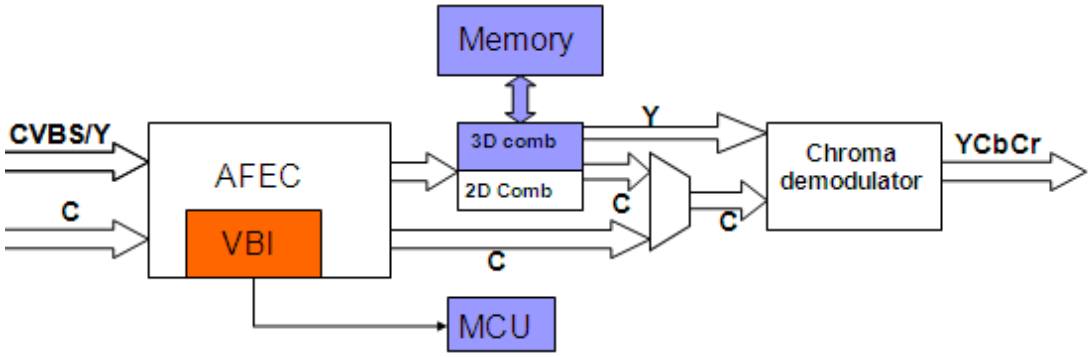
TV: 只输出ATV & DTV 信号，若前一个source 是ATV，切到DTV 时,因为DTV 用的tuner & demod 不一样，需要将ATV finalize。而切到其他source,因为是TV 模式，就不需要 finalizeATV 的tuner & demod。这样av out 的还是atv 的信号。DTV 同样处理。

Monitor: 根据当前source 输出对应的信号，所以从ATV/DTV 切到其他source 时，都要重新 配置VIF 或demod. 若是VE output，则根据芯片的支持情况来决定是否支持所有的source av out，这部分 code 比较固定，改动不多。

2 信号检测流程

ATV/ AV/SV source 是在threadSignalMonitor()中检测signal 变化。 这三个source 都是通过VD 来断定信号是否锁定，信号框图如下。

Figure.4



VD 根据Hsync 来判断信号是否锁定，对应如下寄存器：

AFEC_CD[6] = VD Hsync lock

在Supernova，判断Signal stable 状态的api 接口为

```
MAPI_BOOL mapi_vd::IsSignalStable(void)
{
    return ((m_u16LatchStatus & MAPI_VD_HSYNC_LOCKED) != 0)?
    MAPI_TRUE : MAPI_FALSE;
}
```

而m_u16LatchStatus 的状态是在checkVideoStanderd()中实时更新的。

下面分别详细介绍下ATV/AV/SV 的信号检测机制以及差异。

对于ATV source，ATV 的信号要流经Tuner、VIF 和VD 三部分，即首先要给Tuner 设置 正确的频点，Tuner 把 RF 信号变频成中频信号，然输出中频信号给 VIF 去做解调，解调出来 的CVBS 信号经过ADC 产生digital CVBS 信号，送给VD，在VD 里面经过AFEC 进行制式识别，

（其实是通过color burst 识别的，同时AEFC 会产生H/V 同步信号），信号经过AFEC 后再经 过comb filter 进行亮色分离得到Y/C 信号，然后C 信号经过chroma demodulator(色度信号解 调器) decoder 解出 YUV 给 scaler 处理。所以这三部分都要设置正确，后端 VD 才能正确识别 出Video standard 和Lock 信号。

在 Supernova 里，函数 CheckVideoStandard()会被循环调用，它的功能就是实时判断 VD 信号的状态，包括 Lock & Video Standard 的变化，以便能及时通过 m_bIsVideoFormatChanged 告知UpdateVDandScaler()去重新set window，使画面能正确显示。

在上层我们通过函数 IsSignalStable（）来判定信号的锁定情况。在 ATV 搜台时，我们会 同时判断 VIF 和VD 是否锁住，然后才去判断信号的彩色制式和伴音制式。

ATV 信号检测Flow：

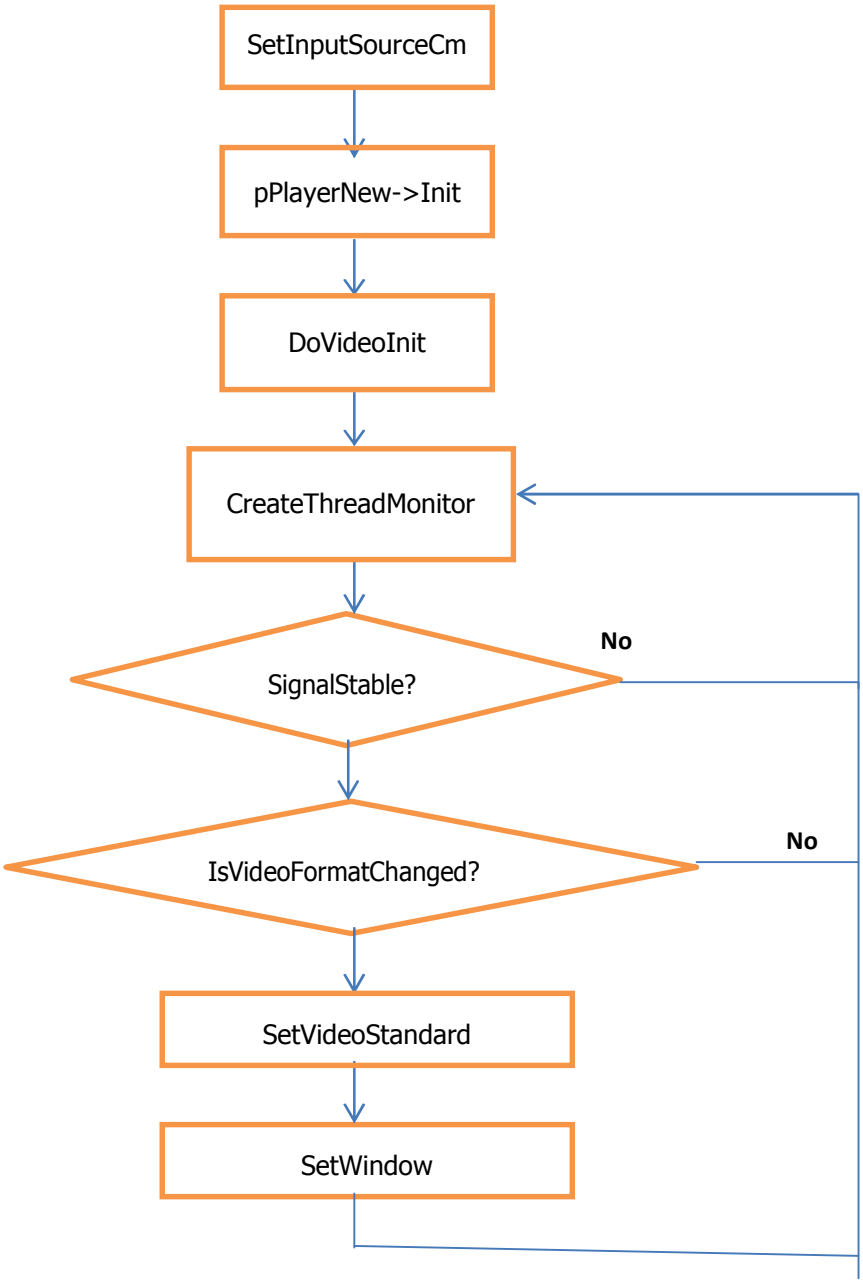


Figure.5

对于AV source，相比ATV 少了Tuner，且送进来的直接是CVBS 信号，所以信号flow 和
ATV 的后段一样。

对于SV source，因SV 直接输入Y/C 信号，经过ADC 产生digital Y/C 信号，不用再经过 comb filter 处理，然后直接通过色度解调器得到YUV 信号送给Scaler ,其他的和ATV&AV 一 样。

下面以AV 为例介绍下它的信号检测流程：

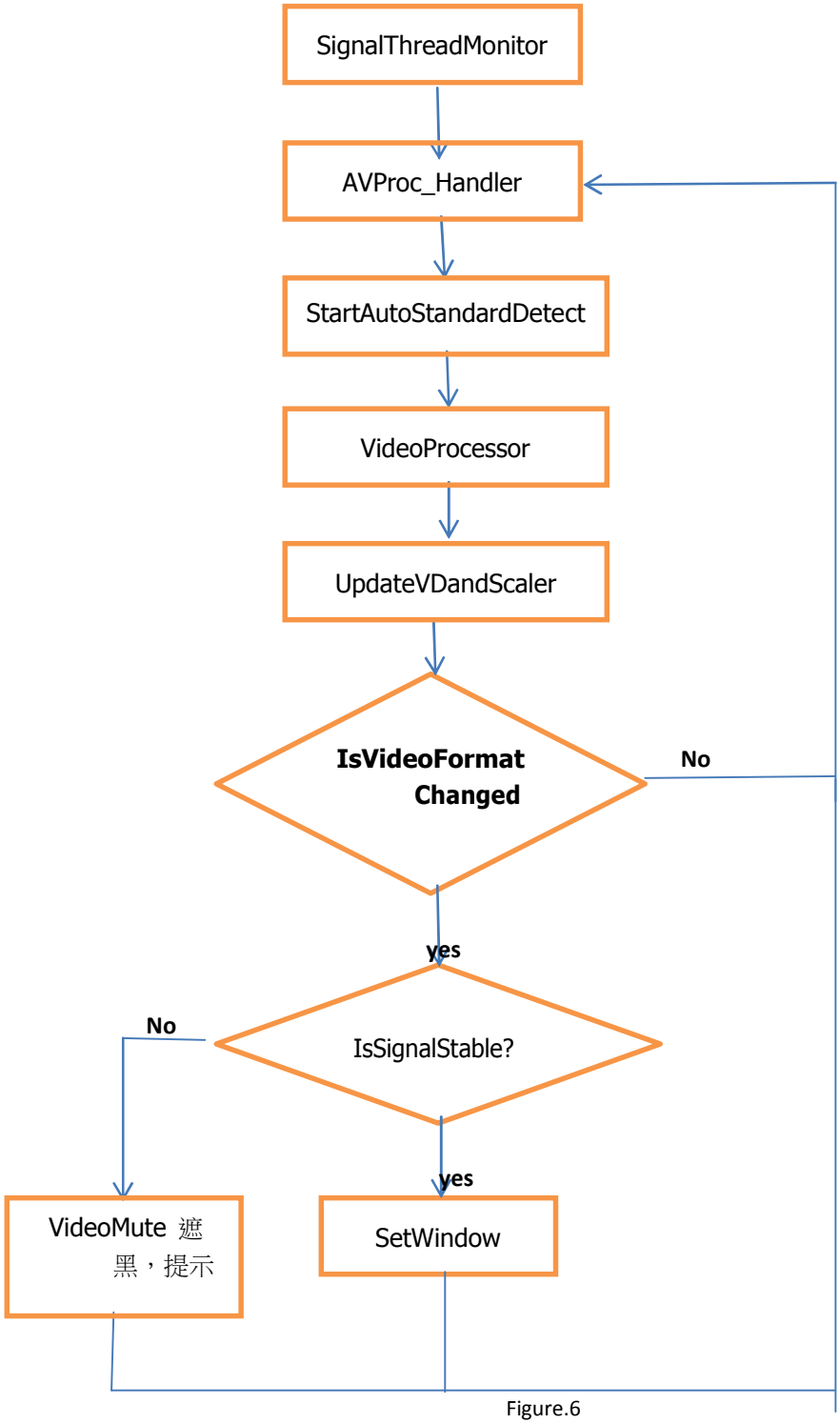


Figure.6

PC/YPbPr/HDMI source 在monitorThreadFunc()中检测signal 变化。VGA 与YPbPr 已经是 亮度、色差分离信号，VGA 是RGB 信号，YPbPr 是由Y/Pb/Pr 组成，所以不需要再进行VD

解码处理，直接经ADC 转换后送给Scaler。信号在Scaler IP1 模块检测，VGA/YPbPr 在monitorThreadFunc()中进行signal 实时检测处理.

对于VGA 信号，RGB 信号不包含同步信号，信号稳定与否是在Scaler IP1 通过VSync 来 识别的，在信号中会包含 timing sync 状态的信息，IP1 根据 Vsync 判断信号是否稳定，一帧 画面是否开始。同时IP1 也可以识别到timing 信息。

大致flow 如下：

1、在ThreadMonitor 中进行信号侦测操作，通过Scaler IP1 获取signal 状态和Timing 信息。

2、 通过 MApi_XC_PCMonitor 获取信号状态及对应的 timing 是否支持，如果不支持，提示 Not Supported

3、 通过 GetModeInfo(m_pModeInfo)获取到timing 信息，并进行显示

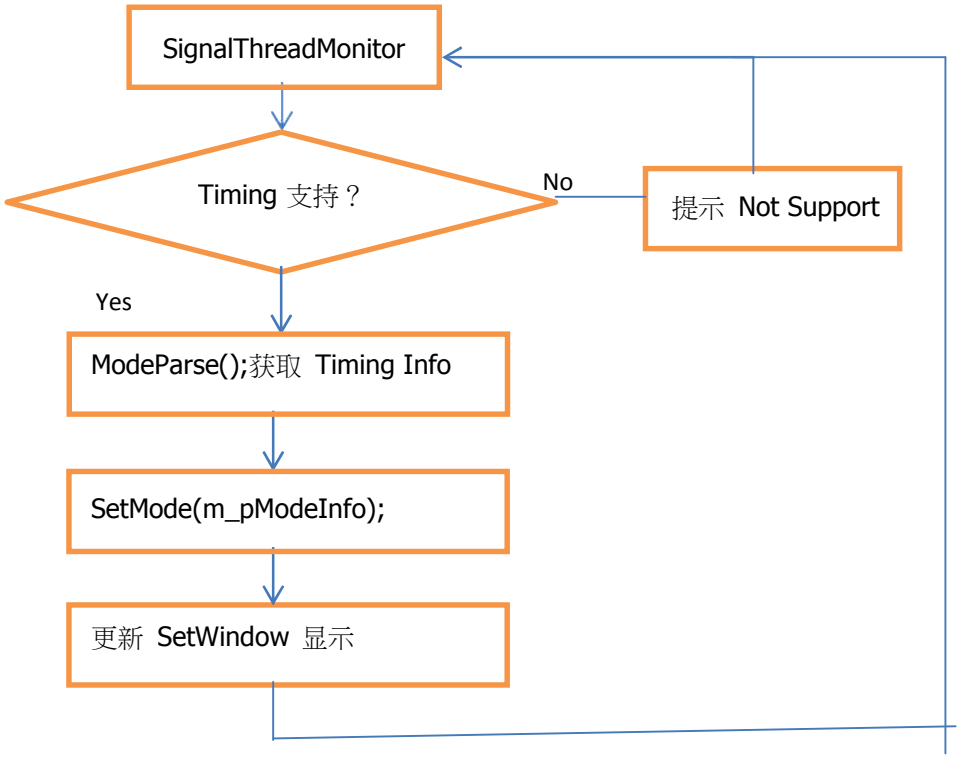


Figure.7

对于YPbPr，Y 上会包含VSync 同步信号，IP1 会根据Y 上的同步信号识别到一帧画面开 始。YPbPr 信号code flow 与VGA 类似 HDMI 已经是数字信号，不用经过VD decoder 处理，Scaler IP1 前有一个HDMI engine 会 将HDMI 信号进行转换，送给Scaler IP1，转换后的信号中包含IP1 可以识别的每一帧画面起 始的同步信息。

同VGA/YPbPr 类似，在MSrv_HDMI_Player::PlayerMain(void)中进行检测判断

```
MSrv_HDMI_Player::PlayerMain(void)
{
  m_pHDMISignalProc->Handler();
      enCurrSyncStatus = m_pHDMISignalProc->CheckSignalStatus();
      if((enCurrSyncStatus ==
      mapi_signalprocess_datatype::E_SIGNALPROC_STABLE_SUPPORT_MODE)
      ||==
      mapi_signalprocess_datatype::E_SIGNALPROC_STABLE_UN_SUPPORT_M
      ODE))
      {
            m_bSignalStable = TRUE;
      }
}
```

DTV 信号经过 Tuner 转换成数字中频 ,然后送数字中频信号给 demod 去解调，送出 TS 流给Demux。在搜台时通过Demod 端的m_pcFrontEnd->GetStatus()函数确定是否有lock 住 信号,在Demod 的driver 中(以ATSC 为例)通过DTV_ATSC_GetLockStatus()获取最终的demod lock status。

DTV 在 MSrv_DTV_Player_DVB::_ScreenSaverMonitor 函数里实时检测信号锁定情况。

在 MSrv_DTV_Player::MonitorThreadFunc()中检测TS 中PSIP 节目信息变化。

大概code flow 如下：

1、 首先在 ScanMain 里面进行整个搜台过程

2、 ScanState_Init 进行 Tuner/Demod 参数设定

3、 然后在 ScanState_TuneToRFChannel 中进行 Demod 前端信号检测，如果信号稳定，通 过Demod 检测到前端有lock 住，说明信号稳定，则进行存台，完后进行显示。

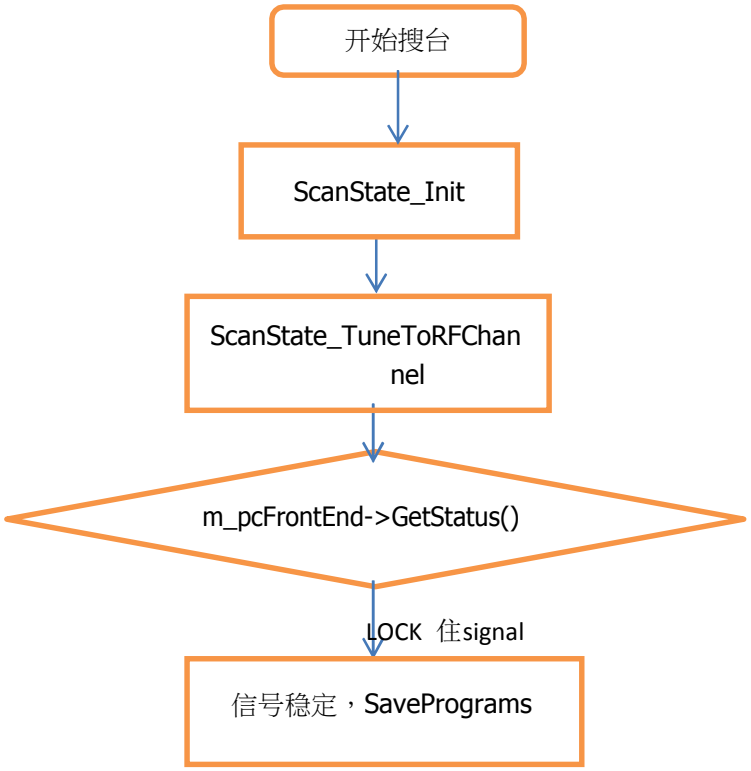
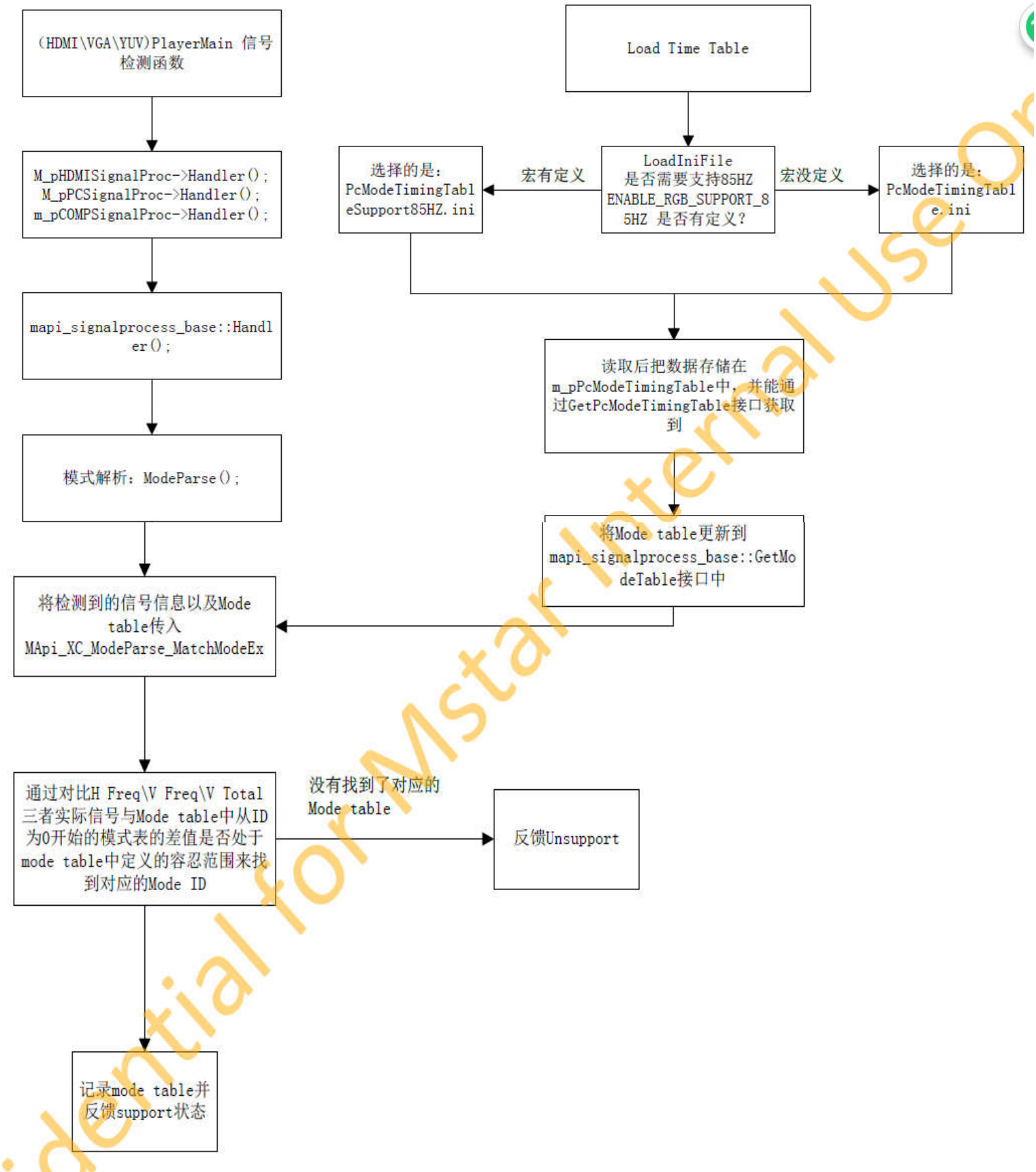

开始搜台
ScanState_Init
ScanState_TuneToRFChannel
m_pcFrontEnd->GetStatus()
LOCK 住signal
信号稳定，SavePrograms

Figure 8

HDMI\VGA\YPbPr 模式检测流程如下所示。



3.消息处理机制

在进行上层应用开发时，特别是在进行 TV apk 相关开发的时候，经常需要从 Supernova 中拿到一些状态比如播放码流状态，信号状态，CI/CA 卡的状态，扫台等状态。这些状态通 常是Supernova 底层发送event，在通过binder 到jni，jni 再到我们的 XXManager.java。然后通过实现 listener 接口监听，接着在 handler 实现 listener 中的 event 处理，最终完成所需 逻辑操作。下面就看一下具体的实现流程。

1.event 的流程

a. supernova send event

在Supernova 中，很多请求都不是直接处理的，而是新建一个Event 对象，将其加入消息队列，然后相应的服务中不停地从消息队列中取出消息进行处理。在 Supernova 中用以下 一个泛型类表示一个事件：mapi_event。例如：

path://supernova/mstarsdk/dvbt/include/ Mapi_utility.h mapi_event

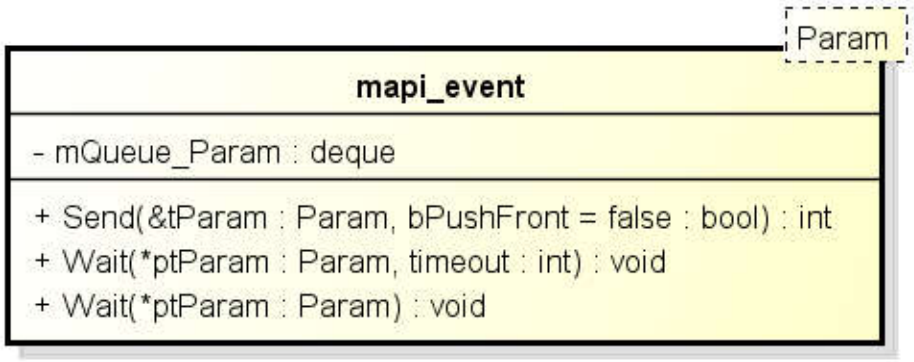


Figure.10

mapi_event 是一个泛型类，可以看到，在其内部有一个queue，每次向服务发送一个消 息时都会将消息对象加入到这个队列。

一般需要事件处理的每个服务都有自己的消息处理循环，如果需要消息循环的话，只需 要在相应的类中声明一个mapi_event 对象，当然可以有多个，当需要发送消息时只需要调 用这个成员对象的Send 方法，在这个服务中的消息循环中或其他需要的位置调用这个成员 对象的Wait 方法取得消息队列中的消息进行处理。

mapi_event 是泛型类，deque 也是泛型的，所以每个服务可以有不同的消息体。 比如说调用DTV 的自动搜台时,event 处理flow 如下:

发送event:

1、首先在_DoEventCreate 中new 一个 mapi_event<MS_DTV_PlayerInfo>m_pcDtvEv； 2、通过 SetCMD 传入对应的 event，最后调用m_pcDtvEv->Send(cDtvPlayerInfo);方法

将此event 放入queue

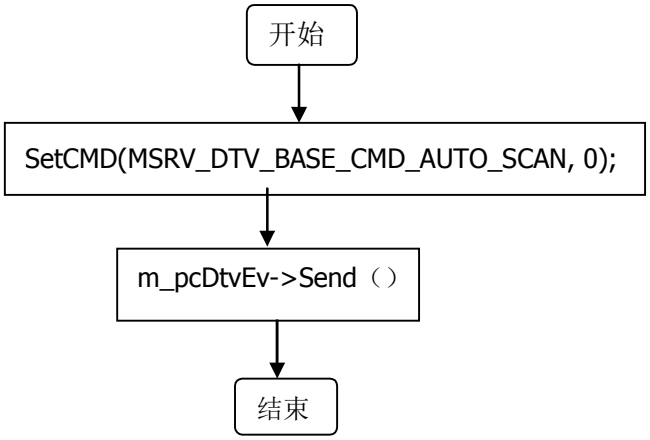


Figure.11

接收event:

在DtvPlayerMain 中有一个while 循环，一直在检测消息队列中是否有消息

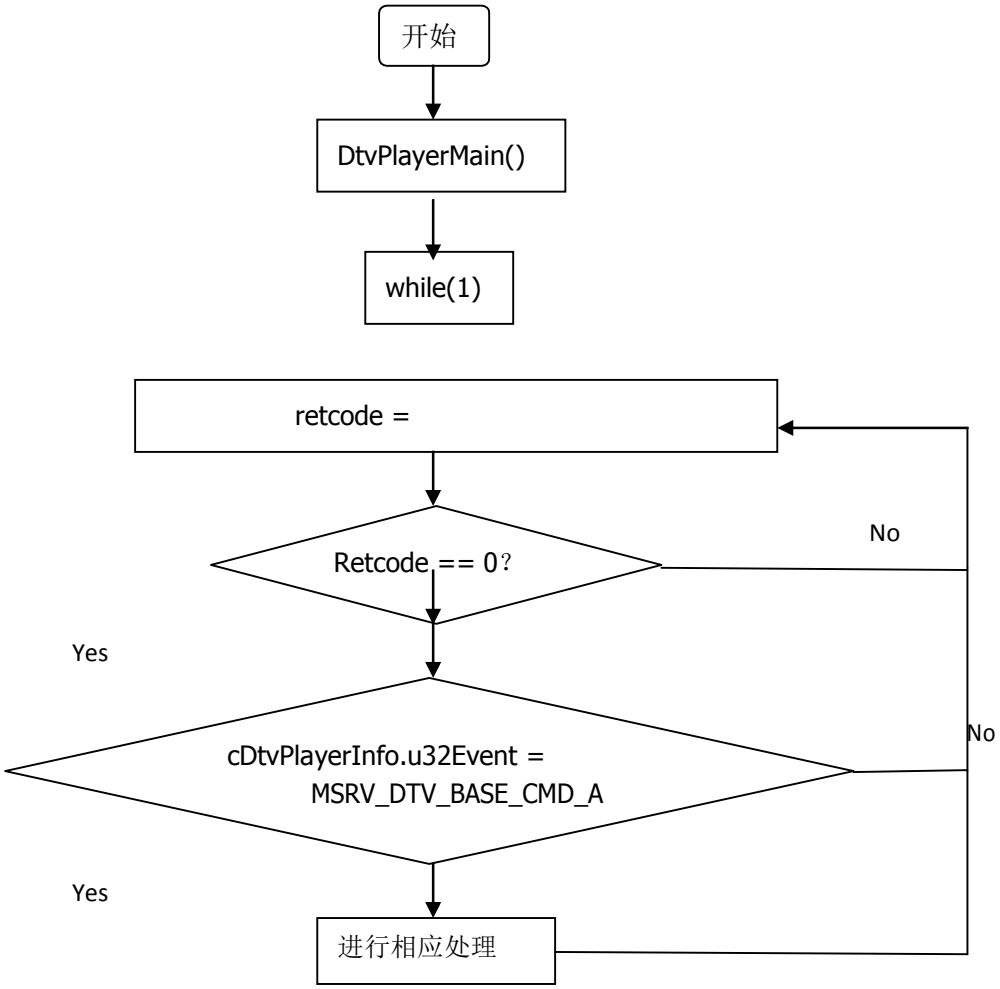


Figure 12

上面讲的是supernova 这个系统里面event 是怎么send，怎么被接受的。现在说下supernova 里面event 是怎么通过binder 与上层交互的。

我们还是回归到supernova 中的XXXservice.cpp 来看，在这些service 中都有一个PostEvent(.. .. ..)方法，事件就是从这些 Event 中发出的。 例如：

nEvt=EV_DTV_AUTO_TUNING_SCAN_INFO

（path://supernova/projects/tvos/playerimpl/libplayerimplservice.cpp）

Event 发送 flow 如下：

1首先在Supernova 中通过DVBScanResultNotify()将EV_DTV_AUTO_TUNING_SCAN_INFO 及对应的频道信息通过postEvent 发送出来

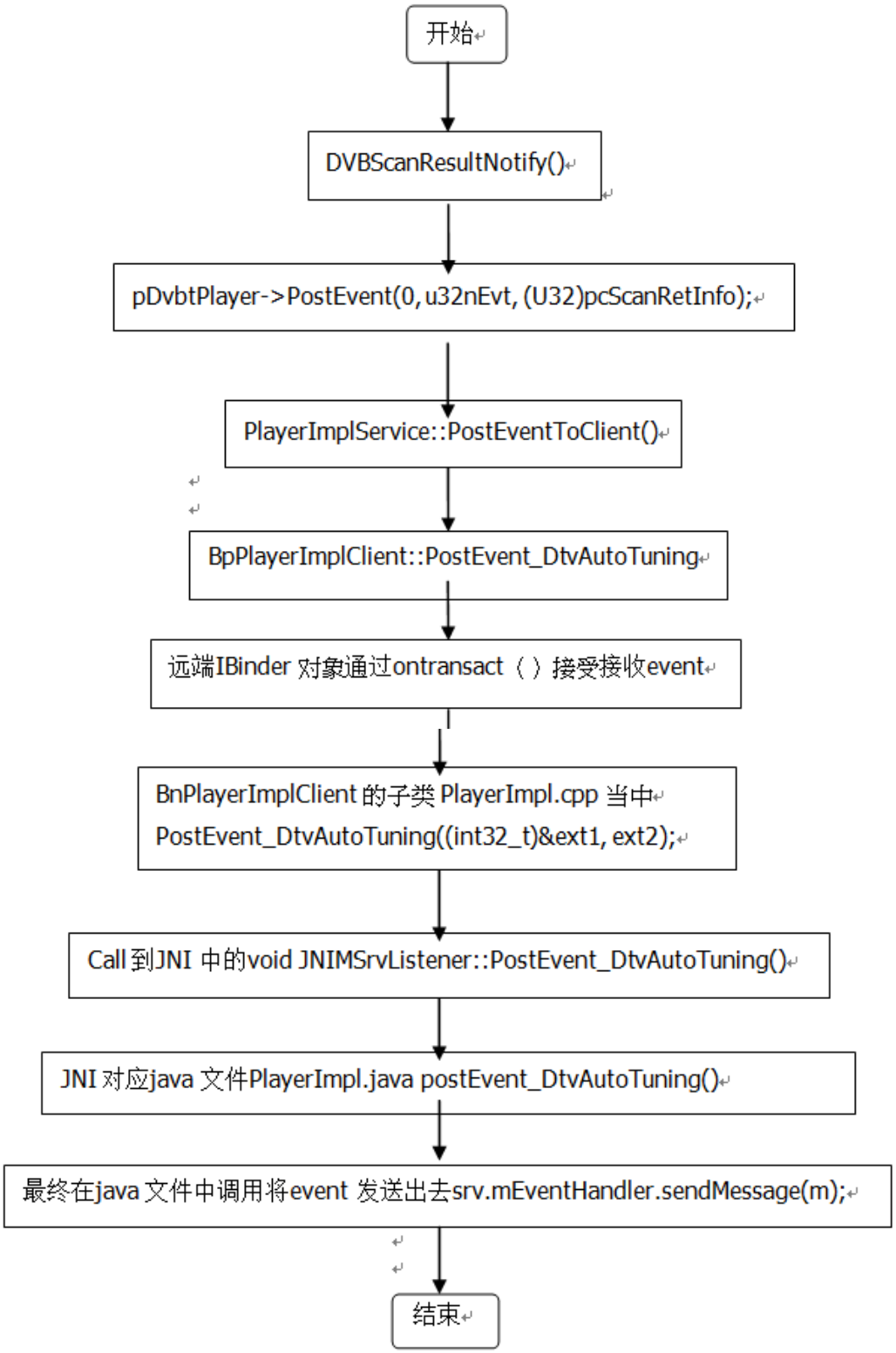
2、在 service 里面接受到这个 event 之后就去调用 Client 里面方法PostEventToClient()

3、IPlayerImplClient.cpp 里面会调用 PostEvent_DtvAutoTuning，远端的 IBinder 对象就 通过 ontransact() 接受此 event。这个远端的 IBinder 对象就是在IPlayerImplClient.cpp 中

4、调用 BnPlayerImplClient 的子类 PlayerImpl.cpp 中的 PostEvent_DtvAutoTuning() 5、在对应的JNI 中call 对应的PostEvent_DtvAutoTuning()

6、call 到 JNI 对应 java 文件 PlayerImpl.java 的 postEvent_DtvAutoTuning()

7、最终在java 文件中调用将event 发送出去srv.mEventHandler.sendMessage(m); 至此，一个event 就从Superonva 发送到java apk 层了



讲到这里event 的整个流程已经完啦。

2. listener 的两种注册方式

目前在我们的code 中有两种注册listener 的方法一种是通过service 里面的register,一种是通 过 TV API 的 set 方法。如下：

path:/device/mstar/common/libraries/tv/java/com/mstar/android/tv/TvChannelManager.java

```
RegisterOnDtvPlayerEventListener：
public boolean registerOnDtvPlayerEventListener(OnDtvPlayerEventListener
listener) { Log.d(TAG, "registerOnDtvPlayerEventListener ");
if (dtvClient == null) {
dtvClient = new
  DtvPlayerEventClientCallBack(); try {
                        TvManager.getInstance().getTvCommon()
                                .addClient("DeskDtvPlayerEventListener", dtvClient);
} catch (RemoteException e) {
                        e.printStackTrace();
}
  }
dtvListeners.add(liste
ner); return true;
}
```

然而这样做的好处就是维持 ArrayList 链表实现可以多个 listener 同时注册。但是这里只有 add 方法，没有 remove，那么就会存在安全隐患，如果注册多个 listener 后就会报 DeadObjectExceptions，binder 挂掉。

再看 set 方法：

```
setOnDtvPlayerEventListener
public void setOnDtvPlayerEventListener(OnDtvPlayerEventListener
        listener) { mOnDtvPlayerEventListener = listener;
}
```

很简单就是个单纯的 set，这里就有一个确定，每次只能注册一个 listener 对象，新的对 象总会覆盖掉老的对象。但不会出现 binder 挂掉问题，

解决方法：在我们上层应用维持一个 list，添加 add，

remove 方法。

3. TVOS 与 AN 交互的 EVENT

底层Supernova 会用三个Event 来控制上层显示的内容分别是：

1.EV_SIGNAL_LOCK//从无信号到有信号的时候会发

2. EV_SIGNAL_UNLOCK//从有信号到无信号的时候会发

3 EV_SCREEN_SAVER_MODE //signal lock 时 status 应该显示什么

4 EV_TVOS_UTIITY_EVENT //类似ScreenSaverMode,可以快速添加一个客制化的event

1.SN 中的EVENT 枚举

```
/// Screen Saver
Mode typedef
enum
{
/// The screen saver mode is invalid service.
//信号非标或信号很差，无法解除video
```

```
    MSRV_DTV_SS_INVALID_SERVICE,
     /// The screen saver mode is no CI module.
    //没有插CI 卡
     MSRV_DTV_SS_NO_CI_MODULE,
     /// The screen saver mode is CI+ Authentication.
    //CI+校验认证成功EVENT
     MSRV_DTV_SS_CI_PLUS_AUTHENTICATION,
     /// The screen saver mode is scrambled program.
    //加密节目EVENT，加密节目没有被解密，发送此EVENT
     MSRV_DTV_SS_SCRAMBLED_PROGRAM,
     /// The screen saver mode is channel block.
    //手动lock channel
     MSRV_DTV_SS_CH_BLOCK,
     /// The screen saver mode is parental block.
    //父母锁EVENT,码流中的EVENT 比实际设置的父母锁等级高或相同，则发送此EVENT
     MSRV_DTV_SS_PARENTAL_BLOCK,
     /// The screen saver mode is audio only.

    // Audio Only 节目
     MSRV_DTV_SS_AUDIO_
     ONLY,
     /// The screen saver mode is data only.
    //DVB DATA 节目
     MSRV_DTV_SS_DATA_ONLY,
    //正常的Video/Audio 节目
     MSRV_DTV_SS_COMMON_VIDEO,
     #if (ATSC_SYSTEM_ENABLE == 1)
    //没有任何节目信息时EVENT
    MSRV_DTV_SS_NO_CHANNEL,
    #endif
    /// The screen saver mode is Unsupported
    Format.
    MSRV_DTV_SS_UNSUPPORTED_FORMAT,
    MSRV_DTV_SS_INVALID_PMT,
    /// The screen saver mode support
    type. MSRV_DTV_SS_MAX
} EN_MSRV_SS_MODE;
```

ATV 会用到的 ENUM：

```
typedef enum
{
      //没有用到
     MSRV_ATV_SS_NORMAL,
      //ATSC GTV ATV 没有搜到台时postEvent，用于UI 提示
      MSRV_ATV_SS_NO_CHA
      NNEL,
      MSRV_ATV_SS_MAX
}EN_MSRV_ATV_SS_MODE;
```

HDMI/VGA:

```
typedef enum
```

```
{
    ///< Input timing stable, no input sync detected
    //信号没有检测到
    E_SIGNALPROC_NOSYNC = 0,
    ///< Input timing stable, has stable input sync and support this timing
    //对应的信号Timing 支持
    E_SIGNALPROC_STABLE_SUPPORT_MODE,
    ///< Input timing stable, has stable input sync but this timing is not supported
    //有检测到信号，对应的HDMI/VGA Timing 不支持，post 此EVENT
    E_SIGNALPROC_STABLE_UN_SUPPORT_MODE,
    ///< Timing change
    //信号不稳定

    E_SIGNALPROC_UNSTABLE,
    ///< Timing change, has to auto adjust if PCRGB
    input E_SIGNALPROC_AUTO_ADJUST,
} MAPI_SIGNALPROC_SYNC_STATUS;
```

4.常见 PostEvent:

、Audio Only 节目：

在_ScreenSaverMonitor 中会通过下面方式 post 一个 AUDIO_ONLY Event: PostEvent(0, EV_SCREEN_SAVER_MODE, (U32)MSRV_DTV_SS_AUDIO_ONLY);

、手动频道锁，post CH_BLOCK Event:

PostEvent(0, EV_SCREEN_SAVER_MODE, (U32)MSRV_DTV_SS_CH_BLOCK);

、信号非标或信号很差，无法解出 video, post INVALID_SERVICE Event:

PostEvent(0, EV_SCREEN_SAVER_MODE, (U32)MSRV_DTV_SS_INVALID_SERVICE);

、VChip 打开，post PARENT_BLOCK Event:

PostEvent(0, EV_SCREEN_SAVER_MODE, (U32)MSRV_DTV_SS_PARENTAL_BLOCK);

、正常的Video/Audio，post COMMON_VIDEO Event:

PostEvent(0, EV_SCREEN_SAVER_MODE, (U32)MSRV_DTV_SS_COMMON_VIDEO);

、EV_TVOS_UTIITY_EVENT

SN 里面PostEvent 的万能接口，可以使用这个event 附带其他参数，发送一些event. 用法 类似EV_SCREEN_SAVER_MODE.只是ScreenSaveMode 主要是用于屏保状态通知.

EV_TVOS_UTIITY_EVENT 会比较客制化。

如在SN 里面定义EVENT，通知APK，SN 切台，EVENT 值的定义与apk 处理约定好即 可

```
#define EV_TVOS_UTIITY_EVENT_CHANNEL_CHANGE 0x200
    PostEvent(0, EV_TVOS_UTIITY_EVENT, (U32)MSRV_DTV_SS_COMMON_VIDEO);
```

4. Offline Detect 介绍

Offline Detect 是指可以检测除了当前正在播放的信源之外的其他信源是否有信号。

此功能用一个专门的IP 来实现,除了USB 和TV Source 目前没法检测,其他Source 都是通过检 测sync 状态，判断对应Source 有没有信号接入。HDMI 也可以通过GPIO 来检测，缺点是硬 件要多占用GPIO。在Supernova 通过MSrv_SrcDetect::Msrv_DetectInputSource()函数来实现 此功能。

Offline Detect 检测分为以下4 类：

1、AV/SV：通过 offline 检测电路检测 sync 信号
2、YPbPr：通过 offline 检测电路检测 sync 信号
3、VGA:通过offline 检测电路检测Hsync/VSync 信号
4、HDMI:通过HDMI engine，将信号转换后，通过offline 检测sync 信号。

Offline Detect 流程如下所示：

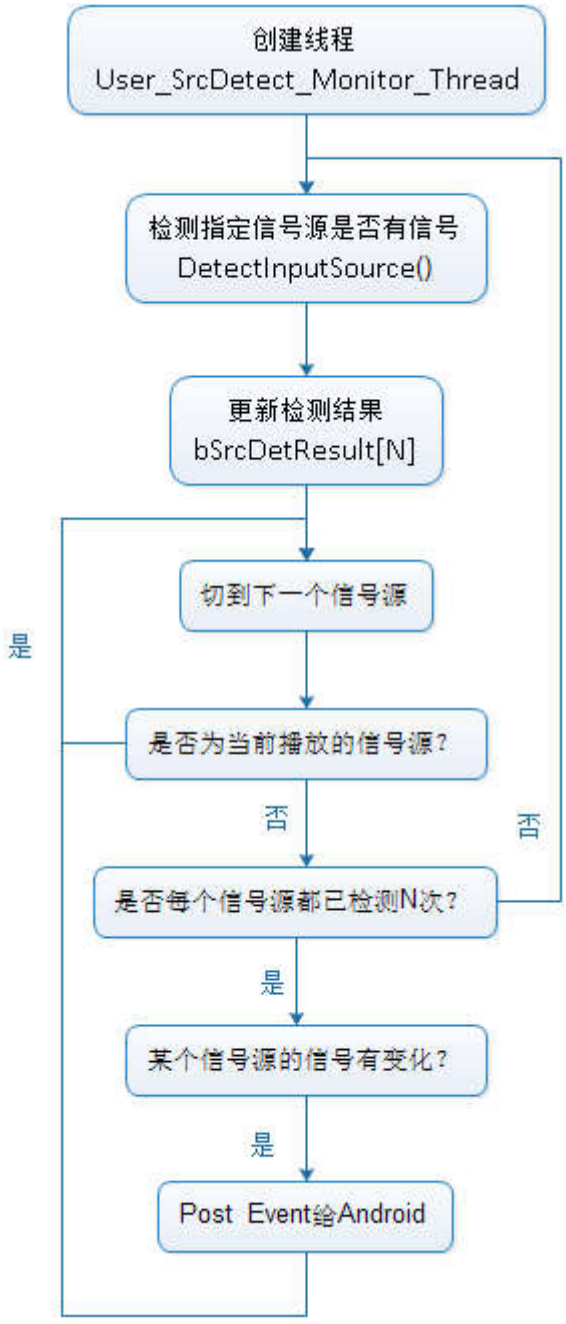


Figure.15

Offline Detect 注意事项：

1.offline 和信号源的信号密切相关，出了问题要多进行公版的对比试验

2.offline 和hw board 的layout 密切相关，所以出了问题也要多进行公版的对比试验

3.offfline 本身是一个经验值性质的东西，不能保证100%的检测正确，这是从原理上得 来的结论。

4.如果gpio pin 脚够用，建议全部用gpio 来实现，可以获得很高的准确性

5 .offline 是一个 loop action,必须和 switch input source 在执行上严格互斥，这是上层软件必须保证的事情，公版互斥flag 为m_bForbidDetection、m_bPauseSwitchInput

6 .出了问题要学会正确打开相关的log，从SN 到utopia 都有log 要打开

7.公版的offline 是经过长期的tuning 得到的结果，所有的相关delay 的值，sog level,和loop times,都不建议修改，一旦修改，就可能造成更大范围的不兼容

5. 常见问题应对措施

1、 Player 有 post event 出来，但是 UI 收不到消息。

针对Pure Supernova 来说比较简单，你看是需要哪个frame 需要处理这个event，然后查看这个 frame 是否有执行到类似 **AddEventRecipient** 这样的函数,比如如果是接收MSrv_ATV_Player.cpp 中发出的event，就需要增加MSrv_Control::GetMSrvAtv()->**AddEventRecipient**(this);

对于TVOS 平台来说，就比较复杂一些，如果这个EVENT 是新加的，就按照增加EVENT 的流程 来增加对这个 EVENT 的接收，这里就不多介绍了。如果不是新加的，那就按照下面的流程来查看。 开机后会在main.cpp 的void * AndroidServiceInitThread(void * pData)函数中来注册service。比如

pTimerManager= TimerManagerService::instantiate();

MSrv_Control::GetMSrvTimer()->RegisterService(pTimerMan

ager);

MSrv_Control::GetMSrvTimer()类就被注册在TimerManagerService 中，这样在MSrvTimer 中发送

的 EVENT 都是在 TimerManagerService 中处理，但是如果在 MSrvTimer 中有发送的在TimerManagerService.cpp 的函数TimerManagerService::PostEvent 中没有被处理的event，android 是 收不到的。同样如果只有在 TimerManagerService.cpp 中的 PostEvent 处理的event，只能通过 MSrv_Control::GetMSrvTimer()来发送event。

比如EV_UPDATE_LASTMINUTE 这个event，如果想要在MSrv_Timer.cpp 之外的文件中来通过 PosetEvent 发送，那就必须注意了，不能直接通过 PosetEvent 来发送，而是通过MSrv_Control::GetMSrvTimer()->PosetEvent 来发送。

2、切换source 的过程中出现死锁或者是coredump。

由于切换source 的时候需要把前一个player 先给finish 掉，之后再把新的 player 进行init。这个 过程中如果走的不对就会出现资源冲突导致 assert 或者是死锁。需要理清 flow，然后找出哪些动作 没有按顺序完成。这类问题最好是利用GDB 进行定位。

6.TV 板卡接口及信号介绍

1.TV 信号

概述：RF 即无线电射频 ，传输的是模拟视频和音频混合编码后的信号，显示设备(TV)的电路将混合编码信号进行一系列分离、解码再输出成像；
缺点：需要进行视频、音频混合编码，信号会互相干扰所以它的画质输出质量是所有接口中最差的；
说明：有线电视和卫星电视(DVB /ATSC/ ISDB 等)接收设备也常用 RF 连接，但这种情况它们传输的是数字信号；

2.AV (Composite Video)

端口标准：RCA 端口黄色为视频，红色一般为右声道，白色(或黑色)为左声道；
概述：“复合”含义是同一信道中传输亮度和色度(Y/C)的模拟视频信号，再通过显示设备对其进行亮/色分离和色度解码才能成像；
优点：实现了音频和视频的分离传输，避免了音/视频混合干扰而导致的图像质量下降；
缺点：由于 AV 接口传输的是亮度/色度混合的视频信号，这种先混合再分离的过程会造成色彩信号的损失，色度信号和亮度信号也会有很大的机会相互干扰，从而影响最终输出的图像质量；
3.CVBS 信号（复合视频广播信号） ：包含了亮度和色度信号，行场同步信号
4.信号的识别：靠识别同步信号来识别是否有信号

3,YPBPR (Component Video)

概述：色差信号即分量信号(Component Video)同时传送三路信号：Y 是亮度信号只包含黑白图象信息；Cr 是 R-Y 信号即红色信号与亮度信号的差；Cb 是 B-Y 信号即蓝色信号与亮度信号的差;
说明：不加 G-Y 色差是要避免传输 G 绿信号，因为 G 信号占据色度信号的 59%不利于数据压缩，用 R-Y 和 B-Y 通过矩阵运算可以得到 G 信号;
优点：YCbCr 与 S 端子相比要多传输 Cb、Cr 两种信号，避免了两路色差混合解码并再次分离的过程，也保持了色度通道的最大带宽，只需要经过反矩阵解码电路就可以还原为 RGB 三原色信号而成像，这就最大限度地缩短了信号源到显示设备成像之间的视频信号通道，避免了因繁琐的传输过程所带来的图像失真，保障了色彩还原的更准确，由于采用了三条线缆来独立传输，并且每条线缆都采用了很好的屏蔽措施，保证了信号间互不产生干扰;

4,VGA(Video Graphics Array)

VGA 也叫 D-Sub，共有 15 针(3 排)，是显卡上应用最为广泛的接口类型，绝大多数显卡都带有此种接口；
R,G,B: 模拟视频信号，用于传输图像信号。RGB 信号的幅度为 0.7V
HS,VS: 行、场信号，用于传输确认每行，每帧图像信号的起始点
I2C 信号 SDA,SCL：用于读取 EDID 信息的通讯信号

5,HDMI（High Definition Multimedia Interface）

HDMI 即高清晰度多媒体接口；
HDMI 和 DVI 一样是传输全数字信号的，不同的是 HDMI 接口不仅能传输高清数字视频信号，还可以同时传输高质量的音频信号，另外接口在数据的保密技术上占有很大优势

6.CI

数字电视通过对数字码流加密，可以保证未授权用户无法接收付费频道节目，这种方法叫做条件接收(Conditional Access)。所以若需要看哪种加密系统的节目必须使用该系统的解码器，这样不同的节目就需要不同的解码器；
CI 通过插各种模块的卡(Smart Card)可达到收看节目目的，CI 解决了对不同加密系统的统一处理方法，使电视机成为一种标准的解码系统；

7.SCART

SCART 接口是欧洲强制要求用于卫星电视接收机、电视机、录像机及其它音视频设备上的互连互通接口；
标准的 SCART 接口为 21 针连接器，外型呈直角梯形。