FAW HS7

IVI System Design

Programming Guide

Released on the: 2017.06.05

Version Number: 0.500

Neusoft Automotives ChangChun

**Version History**

| No. | Version | Section | Brief Description | Date | Rseponsible |
| --- | --- | --- | --- | --- | --- |
| 1 | 0.500 | All | Initial Create | 2017-05-31 | ZhangShuang |
| 2 | 0.501 | 7 | 添加了开机启动相关的描述 | 2017-07-07 | ZhangShuang |
| 3 | 0.502 | 13.11 | 添加了app boot log的使用方法 | 2017-09-20 | ZhangShuang |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Content

[1 概述 4](#_Toc493663842)

[2 HS7 IVI 系统简介 4](#_Toc493663843)

[3 硬件组成 4](#_Toc493663844)

[4 软件架构 4](#_Toc493663845)

[5 App的几种形式 5](#_Toc493663846)

[6 Hello World 5](#_Toc493663847)

[7 App的生命周期及显示管理 7](#_Toc493663848)

[8 App打包方法 8](#_Toc493663849)

[9 电源管理 8](#_Toc493663850)

[10 声音管理 8](#_Toc493663851)

[11 状态管理与PCL 8](#_Toc493663852)

[12 进程间通讯 8](#_Toc493663853)

[13 调试工具 8](#_Toc493663854)

[13.1 串口 9](#_Toc493663855)

[13.2 网络配置 9](#_Toc493663856)

[13.3 telnet 9](#_Toc493663857)

[13.4 log 9](#_Toc493663858)

[13.5 journalctl命令 10](#_Toc493663859)

[13.6 appClient命令 10](#_Toc493663860)

[13.7 systemctl命令 10](#_Toc493663861)

[13.8 LayerManagerControl命令 10](#_Toc493663862)

[13.9 pm命令 10](#_Toc493663863)

[13.10 rpm包的使用 10](#_Toc493663864)

[13.11 app boot log工具 11](#_Toc493663865)

[14 RPC 11](#_Toc493663866)

[15 输入法 11](#_Toc493663867)

[16 弹出消息框 11](#_Toc493663868)

[17 发送Titlebar通知 11](#_Toc493663869)

[18 使用蓝牙 11](#_Toc493663870)

[19 Sample程序 11](#_Toc493663871)

# 概述

本文档描述HS7 IVI系统的硬件组成，软件架构，重要特性，编程方法，调试工具等。

# HS7 IVI 系统简介

HS7 IVI系统是基于东软C4-Alfus平台的一款车载娱乐系统。它提供了基于linux系统，组合众多功能组件实现的一个单主机、多屏幕的编程平台。

# 硬件组成

HS7 IVI系统由主机、LCD仪表屏、IVI屏、RSE1屏、RSE2屏、主扬声器、RSE1耳机孔、RSE2耳机孔、IVI屏Home键、IVI屏Power键、倒车摄像头等组成。

需要注意的是，HS7车机的硬件上运行着两套操作系统。分别是HS7 Cluster和HS7 IVI。Cluster系统负责仪表数据的采集和展示。IVI系统负责搭建车载娱乐平台。所以仪表屏这样的设备在IVI系统中是访问不到的。

这样的设计主要是为了Cluster系统的稳定性和性能。这样的设计下，即便娱乐系统出现了严重的故障也不至于使得仪表失效。而且很多组件是Cluster系统不需要的，剔除之后可以使Cluster系统拥有更快的启动速度和更高的运行效率。Cluster系统在这里不做更多介绍了。

# 软件架构

下图展示了本系统的软件架构。

Linux Kernel

Audio Hub

Media Hub

Weston

PCL

DLT

Package Manager

Power Manager

NodeState Manager

AppFramework

BT

**Home**

**Video**

Carplay Service

Your App

Your Service

Apps

Modules

Linux Kernel

本系统基于linux内核，并在linux内核基础上设置了一系列的组件来实现系统提供给App的管理逻辑和必要的服务。注意，上图中并没有完整列出所有的App和组件，它只是软件架构的示意图。

系统中的大部分组件对于App而言是透明的。比如上图列出的组件中，橙色边框的组件都被AppFramework所使用，用于给App提供封装过的系统通知、系统服务。所以对于App而言它们都是不允许或者不必要使用的。

一些组件提供了重要的功能。比如BT模块提供了蓝牙访问的接口；PCL模块提供了持久化存储的接口；Media Hub模块提供了多媒体、多媒体协议解析的接口；等等。这些模块是需要App使用的。由于这些重要组件的SDK往往都是

区分一个模块是否可以被App所使用的方法也很简单：能够允许App使用的接口，我们都会提供API文档的。

**对于App而言，最重要的系统组件是AppFramework。**因为它管理着所有App的生命周期、显示，协调着三块屏幕的电源状态、声音输出等等重要功能。是软硬件资源调度的核心。它提供给App的API叫AppSDK。是一组C++头文件和对应的动态库。详见《FAW HS7 IVI AppSDK User Guide》

# App的几种形式

目前系统支持3种形式的APP：

1. 普通App
2. 无界面Service
3. 基于内置浏览器的HTML5应用

系统中所有App都是基于AppFramework运行的。所以只要是基于AppFramework的可执行程序都可以通过打包成为有效的App。但是由于AppFramework提供的SDK只有C++版本，所以使用C++之外的语言开发App是比较麻烦的。

打包后的App叫做package。一个package可以包含若干个独立的可执行程序（App/Service），每一个都叫做package的一个item。关于package的详细介绍可以参考：[APP打包方法](#_App打包方法)。

**普通App**是指那些有图形界面，可以在Home界面找到的App。由于系统的显示模块是基于wayland协议的，所以只要是基于wayland的UI库原则上都是可以用于开发普通App的。但是我们更推荐使用KANZI库来制作App，因为本系统的界面都是用KANZI库

**无界面Service**同样需要基于App-Framework，但是不可以有图形界面。轻量级的Service可以以较小的代价实现监听消息、维护状态的功能。这里之所以把Service称为“无界面Service”，是因为还存在着有界面的Service。比如系统的Titlebar和输入法，都是有界面的服务。但是因为有界面的服务往往是系统组件，所以在此就不介绍更多了。

**HTML5应用**目前都基于东软HTML5团队开发的HTML5应用框架。HTML5应用框架本质上是一个能够解析特定软件包执行特定脚本的可执行程序，它是浏览器应用的一部分。更多信息请咨询HTML5团队，本文档不再介绍。

# Hello World

一个最简单的HS7 App至少需要包含4部分代码：

1. 初始化AppSDK。
2. 继承AppPLC类，并创建一个它的实例。
3. 根据GetSurfaceID函数返回的surface ID创建一个surface，并用它绘制UI。
4. 返回home的按钮

以下代码展示了使用AppFramework提供的surface creator库和PNG库将一个写有“hello world”的图片绘制出来的方法。这里没有使用推荐的KANZI库是为了减少不必要的细节，突出AppSDK的使用方法。

1. #include <PNG.h>
2. #include <SurfaceCreator.h>
3. #include <AppSDK.h>
5. **class** MyAppPLC
6. : **public** AppPLC
7. {
8. **public**:
9. **virtual** **void** OnCreate()
10. {
11. ;
12. }
13. **virtual** **void** OnPreShow(**long** nArgs)
14. {
15. ;
16. }
17. **virtual** **void** OnShow()
18. {
19. ;
20. }
21. **virtual** **void** OnResume()
22. {
23. ;
24. }
25. **virtual** **void** OnPreHide()
26. {
27. ;
28. }
29. **virtual** **void** OnHide()
30. {
31. ;
32. }
33. **virtual** **void** OnStop()
34. {
35. ;
36. }
37. };
39. PNG png;
41. **void** Redraw(unsigned **int** pixels[720][1280])
42. {
43. **if**(png.IsOK())
44. {
45. **for**(**int** i = 0; i < 720; ++i)
46. **for**(**int** j = 0; j < 1280; ++j)
47. pixels[i][j] = png.GetPixel(j, i).GetRawPixel(ARGB);
48. }
49. }
51. **void** OnTouch(unsigned **int** surfaceID, **enum** E\_TOUCH touch\_type, **float** x, **float** y, unsigned **int** finger\_id, **int** seat\_id)
52. {
53. **if**(touch\_type==E\_TOUCH\_UP)
54. StartHome();
55. }
57. **int** main()
58. {
59. AppSDKInit();
60. MyAppPLC app;
61. app.Initialize(GetAMPID());
62. png.Open("/opt/system/apps/com.hs7.hello-world/res/drawable/helloWorld.png");
64. CreateSurface(GetSurfaceID(), Redraw, OnTouch);
66. **return** 0;
67. }

# App的生命周期及显示管理

App的生命周期相关请参考《FAW HS7 IVI PLC-API User Guide》

App的显示管理相关请参考《FAW HS7 IVI AppFramework Design HMIC》

如果希望设置App开机启动，需要在app的alfus-manifest.xml中相应item节点下添加子节点：<isAutoStart>true</ isAutoStart >。

# App打包方法

参考04/07/00/《将程序打包到HS7系统的方法》；这篇文档介绍了目标机中程序的安装位置、打包方式和一种推荐使用的源码组织方式。

参考04/07/AppManager/《HS7\_ID\_Define》；这篇文档介绍了打包过程中用到的ID的来源和意义。

# 电源管理

电源管理相关API文档请参考《FAW HS7 IVI PLC-API User Guide》

电源管理规则请参考《》

# 声音管理

参考《FAW HS7 IVI Audio-API User Guide》

# 状态管理与PCL

状态管理参考《FAW HS7 IVI SM-API User Guide》

PCL的使用参考《》

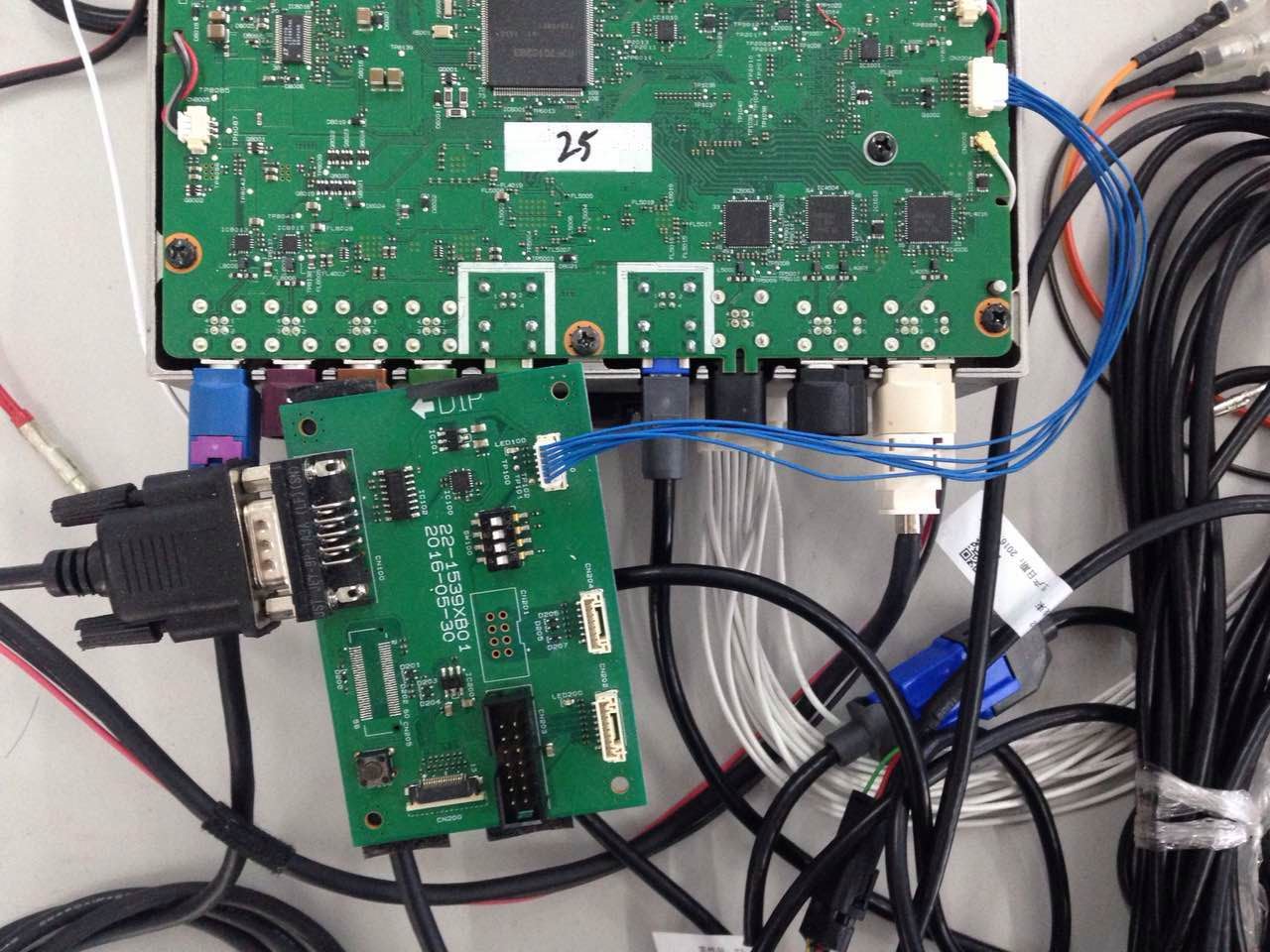
# 进程间通讯

除了linux下常规的进程间通讯方式外，本系统支持使用基于dbus的CommonAPI进行跨进程调用。

Common API的使用参考《》

# 调试工具

## 串口

使用串口需要连接debug板。Debug板和串口线的连接方法如下图所示：

使用前请确认连接正确。

连接好后，需要将串口通讯软件的波特率设置为115200。车机打开的情况下，如果设置正确，在命令行输入回车会提示用户输入用户名和密码。

系统默认的用户名是root，密码是123。

## 网络配置

通过串口可以根据自己的网络设置为车机设置网络。具体的设置方法与linux系统相同，请自行搜索。

配置好后，挂载一个局域网中的samba目录就可以方便的传输文件了。

## ssh

设置好网络以后，就可以使用ssh来连接车机，实现多人共享一个调试机器。用户名密码跟串口登陆相同。

## log

本系统推荐的log工具为AppFramework提供的ama-log库。ama-log库是一个基于DLT的log函数库。ama-log库的使用方法见《FAW HS7 IVI LOG-API User Guide》。

由于当前平台上使用的gdb功能不完善，log是重要的调试工具。又因为使用了systemd启动app，使得获取标准输出比较麻烦。所以推荐使用ama-log库输出log，并使用DLT viewer查看log。

对于分配了log ID的进程，拔IG关机的情况下可以在/media/IVI-LOG/文件夹下找到上次运行期间产生的log。

## journalctl命令

如果一定要查看标准输出，可以使用journalctl工具。具体使用方法请自行百度。

但是journalctl实际上是集合了systemd输出的所有log，所以可能需要对结果进行过滤。

默认情况下，journal不输出任何APP的标准输出。想要看到APP的输出，必须修改/etc/system/system.conf文件。找到其中的“DefaultStandardOutput”项，将其修改为“DefaultStandardOutput=journal”。这样就能看到APP的标准输出了。

## appClient命令

appClient提供了一些AppFramework API的命令行封装。可以启动、关闭App，发送Intent，发起App同步等。是重要的调试工具。

appClient的详细用法可以通过appClient help查看。由于该工具是AppFramework的调试版工程的一部分，通常release版里不提供编译好的appClient。如果没有appClient可以在对应版本的SDK的/usr/local/bin文件夹下找到。拷贝进车机即可使用。

## systemctl命令

本系统使用systemd进行App运行状态、模块依赖的管理。所以systemctl是重要的调试工具。Systemctl本身的使用方法可以在网络上自行查询。App的unit name是在alfus-manifest.xml中配置的，节点名是systemd。需要注意的是，App的unit name是需要添加参数的。参数跟启动app后执行ps看到的参数相同。比如1屏的home，它的unit name是com.hs7.home@AMPID0x1001.service。

## LayerManagerControl命令

LayerManagerControl工具用于查看和修改surface layer screen的设置。在了解本系统的HMIC设计的基础上，通过该工具可以帮助调查一些显示效果方面的问题。

使用方法可以通过在命令行不带参数的使用该命令获得。

关于screen layer surface等基础概念请参考《FAW HS7 IVI AppFramework Design HMIC》、《FAW HS7 IVI AppFramework Design ID》，并结合自行搜索进行了解。

## pm命令

pm命令是package manager的命令行接口。命令的使用方法可以通过pm –h获得。

除此之外，目前pm –e选项在帮助信息中是不包含的。pm –e 用于获取指定type的item。比如要获取所有的COMMONAPP，可以这样用：

pm –e COMMONAPP

## rpm包的使用

调试过程中往往需要频繁的将编译好的程序放到车机中。所以通常我们会把app打成rpm包，然后拷贝到车机中，执行rpm命令安装。像是下面这样：

busybox rpm -i app.rpm

从cmake工程生成rpm包的方法请查看《将程序打包到HS7系统的方法》。

## app boot log工具

当把-DIS\_DEBUG\_MODE选项传递给cmake编译ama-app-framework时，所有的APP的启动过程都会在/tmp目录下生成一个ItemID.bootlog文件（比如IVI屏的home就会生成com.hs7.home.scr1.bootlog）。这个文件记载了ama-app-framework从收到启动APP的指令到最终将APP显示出来所经历的各个重要事件，以及这些事件发生的具体时间（从系统启动开始的时间，以秒为单位）。

如果生成文件时，有同名文件存在，ama-app-framework会在文件名后追加.next重新尝试创建文件，直到创建成功为止。

这个文件生成的时机是APP显示出来以后。所以由于各种原因启动失败的APP是无法生成这个文件的。

这个功能有助于分析APP启动的性能问题。

目前，一个典型的输出是这样的（以video应用为例）：

time stamp: 1563.563353488 s app: 0x1005 start command is sent to systemd.

time stamp: 1563.777745148 s app: 0x1005 is started by systemd.

time stamp: 1564.796740848 s app: 0x1005 's surface is created.

time stamp: 1565.876766070 s app: 0x1005 's surface is configured.

time stamp: 1565.935407783 s app: 0x1005 is handling preshow/prehide notify.

time stamp: 1566.025292953 s app: 0x1005 's preshow/prehide notify is handled.

time stamp: 1566.040368720 s app: 0x1005 's surface is show/hide. app is started.

每行log以时间戳开头，第二项是app的AMPID，第三项是事件描述。

第二行…started by systemd代表APP开始启动；最后一行log，…app is started就代表app启动完成。这两条log的时间戳差值就是KANZI进程的启动时间（≈2.26s）。而第一行“…start command is sent to systemd”开始到最后一行则代表从ama-app-manager接到启动video的指令到APP启动完成的时间。

# RPC

# 输入法

# 弹出消息框

# 发送Titlebar通知

# 使用蓝牙

# Sample程序

附件程序使用KANZI库开发了一个展示主要AppFramework API使用方法的App。