最近公司有个项目，要用 MTKMTK10A1032 版本出软件。今天刚刚拿到了 MTK10A1032 版本的软件，大致了解了一下，感觉比之前的版本改动要大很多，可以说是脱胎换骨了。改动后的版本 C 源文件明显比原来减小，而附加了很多 xml、res 文件，各模块资源定义和文件结构也做了较大改动。对于程序员来说，由于资源和操作分离了，加些资源什么的比之前要容易一些，程序也简洁了很多。当然老 MTK 程序员需要用一段时间来适应新版本，一段时间内可能不是很顺手。本文从创建一个 APP 为例逐步探讨该平台的开发流程。由于水平有限，不保证所有内容均准确无误，只是抛砖引玉给想尝试新版本的同仁们（那些想第一个吃螃蟹的人，嘿嘿，偶也是）。若有疏忽遗漏错误之处，欢迎指正，欢迎交流。

PS：10A 开发环境为 VC9+RVDS3.1，bin 档编译只能单机（目前暂不知道怎么用分布式，开启分布式会报错），模拟器可以使用分布式。编译 bin 档前请卸载分布式编译（偷懒想退出 IncrediBuild 而不卸载的，不要告诉我没提醒你，你会死的很难看，嘎嘎）。有谁知道 10A 下分布式怎么配的，留言告诉我，谢谢！至于环境的配置我就不罗嗦了，VC9 很好装，RVDS 也不是很困难，网上找找资料就可以搞定。

1、文件结构

目录一：plutommi\MMI\FirstApp
目录二：plutommi\MMI\FirstApp\FirstAppInc（该结构未改变）
文件列表：
FirstAppDefs.h：用于存放本程序所需要的类型，结构，常量的定义
FirstAppProt.h：用于存放本程序中的所有函数声明，但此文件只被本文件的源程序所加载
FirstAppGprot.h：也是用于存放函数声明，但是此文件是用于别的程序加载，即此文件中的函数声明的都是对外的接口
FirstAppResDef.h：用于存放本资源 ID 的定义接口

目录三：plutommi\MMI\FirstApp\FirstAppSrc（该结构未改变）
FirstAppSrc.c 程序的主源文件

目录四：plutommi\MMI\FirstApp\FirstAppRes（新的目录）

FirstApp.res：资源文件定义，包含字串、图片、菜单、屏幕等的定义，该文件实际为一个标准 xml 文件

ref_list_FirstApp.txt：该模块多国语言字串定义（目前发现该文件并未生效，不知道是否是设置问题）

2、将文件加入项目

修改 make\plutommi\mmi_app\下的三个文件：
1）mmi_app.lis:此文件用来申明 MMI 所要编译的所有源文件，添加如下一行：
plutommi\MMI\FirstApp\FirstAppSrc\FirstAppSrc.c
2）mmi_app.inc：此文件用来指明 MMI 所有头文件所在目录，同样添加：
plutommi\MMI\FirstApp\FirstAppInc
3）mmi_app.pth：此文件用来指明 MMI 所有源文件所在目录，添加：
plutommi\MMI\FirstApp\FirstAppSrc

4）在 plutommi\Customer\ResGenerator\MakeFile 文件中添加如下一行：
-I "../../MMI/FirstApp/FirstAppInc"\

3、应用程序 ID 定义

1）在基础 ID 统一定义文件 plutommi\MMI\Inc\mmi_res_range_def.h（原来为 MMIDataType.h）中找到如下定义块：

```
RESOURCE_BASE_ENUM_BEGIN()
    /*****************************************************************
     * Declare resource ID range below

     *****************************************************************/

    ......

  /*****************************************************************
     * Declare resource ID range above
     *****************************************************************/

  RESOURCE_BASE_ENUM_END()
```

在其中添加：

```
#ifdef __MMI_FIRSTAPP__
RESOURCE_BASE_RANGE(FIRSTAPP, 50),
#endif
```

2）找到如下定义块：

```
/* Beginning of resource table */
RESOURCE_BASE_TABLE_BEGIN()
......
/* End of resource table */
RESOURCE_BASE_TABLE_END()
```

在块中间末尾位置添加：

```
/***************************************************************************** FirstApp
*****************************************************************************/
#ifdef __MMI_FIRSTAPP__
#define FIRSTAPP_BASE ((U16) GET_RESOURCE_BASE(APP_FIRSTAPP))
#define FIRSTAPP_BASE_MAX ((U16) GET_RESOURCE_MAX(APP_FIRSTAPP))
RESOURCE_BASE_TABLE_ITEM_PATH(APP_FIRSTAPP, ".\\MMI\\FirstApp\\FirstAppRes\\")//这里较之前有所变化
#endif
```

4、字串、图片、屏幕资源的添加

先来说字串、图片、屏幕资源 ID 的添加。10A 版本中对资源定义的改动比较大，稍微对比一下新老版本，你会发现很多原来很大的文件变小了，而多出来了很多.res 后缀的文件，这就是新版本的资源定义文件。10A 版本仍部分保持了老版本的定义方式，不过笔者建议采用新版本的方式来定义资源，而且与之前相比新版本的资源定义要方便不少。读者可以随便找一个 res 文件，会看到如下定义格式：

```
/* Needed header files of the compile option in XML files, if you need others need to add here */
#include "mmi_features.h"
#include "custresdef.h"

/* Need this line to tell parser that XML start, must after all #include. */
<?xml version="1.0" encoding="UTF-8"?>

/* APP tag, include your app name defined in MMIDataType.h */
<APP id="APP_FIRSTAPP">/* 这里定义的 id 必须和之前定义的 APP 的 ID 一致 */

    /* When you use any ID of other module, you need to add
       that header file here, so that Resgen can find the ID */
    <!--Include Area-->

    <!-------------------------------------------------------

                    String Resource Area

    ------------------------------------------------------->
    /* String ID of you Application, we will get string from ref_list.txt for all languages */

       <STRING id="STR_ID_APP_FIRSTAPP_HELLO"/>/* 在这里添加自己的字串 ID */

  /* 这里说下，字串的内容定义和之前的版本一样放在 YourProjectPath\plutommi\Customer\CustResource\PLUTO_MMI\ref_list.txt 下，为了更好的兼容，最好将你自己定义的字串重新规整到 YourProjectPath\plutommi\MMI\FirstApp\FirstAppRes\ref_list_FirstApp.txt 下，具体定义方式和 ref_list.txt 类似，参考本博客文章《MTK 编程起步——建立新 APP 和资源定义》 */

    <!-------------------------------------------------------

                        Image Resource Area

  ------------------------------------------------------->
```

```
    /* Image Id and path of you Application , you can use compile option in Path, but need out
    of "" */
        <IMAGE
id="IMG_ID_APP_FIRSTAPP_HELLO">CUST_IMG_PATH"\\\\MainLCD\\\\FirstApp\\\\HELLO.BMP"</IMAGE>

  /* 这里定义自己的图片 ID 和路径，注意图片现在的 CUST_IMG_PATH 路径是在
YourProjectPath\plutommi\Customer\Images\FTE320x480，请将图片文件夹放在这里，然后打包整个
MainLCD 文件夹为 image.zip（改了 mtk_resgenerator.cpp 的可以不用打包） */

    <!-------------------------------------------------------

                        Menu Resource Area

  ------------------------------------------------------->
    /* Only MENUITEM need compile option, MENUITEM_ID does not need */
    /* 这里定义你的菜单 ID，具体定义方法稍后给出 */

    <!-------------------------------------------------------

                        Other Resource

  ------------------------------------------------------->
    <SCREEN id="SCR_ID_APP_FIRSTAPP_HELLO"/>/* 这里定义你的屏幕 ID */

</APP>
```

这个 res 文件类似与 xml 文件，不过你可以在其中使用一些 C 的预处理命令和注释。使用这种方式你不需要自己去定义 res_app_firstapp.c，系统会在这个文件中搜索 ID 并自动生成名为 mmi_rp_app_firstapp_def.h 和 mmi_rp_app_firstapp.c 的文件，并在后者中定义对应的 populate 函数。至于 res 文件中各标签和其属性定义，请参看 MTK 官方文档 10A_MMI_Resource_Training.pdf。

5、菜单添加

菜单由于有上下级关系，定义要相对复杂些，这里先介绍几个标签：MENU、MENUITEM 和 MENUITEM_ID。MENU 是用来定义菜单树的标签，MENUITEM 是用来定义单个菜单项的标签，MENUITEM_ID 是用来在 MENU 树中安置菜单项的标签。下面来讨论菜单的定义。

新版本中菜单的定义方式有很多种，现介绍最常用的三种方式，其它方式请读者参看上面给出的 MTK 官方文档。

方法一：MENU 中包含 MENUITEM_ID 方式定义

这种方式采用如下格式：

```
<MENUITEM id="SUBMENU1" str="STR_SUBMENU1"/>
<MENUITEM id="SUBMENU2" str="STR_SUBMENU2"/>

<MENU id="MENU1" type="OPTION" str="STR_MENU1" highlight="HighlightMenu1" hint="HintMenu1">

        <MENUITEM_ID>SUBMENU1</MENUITEM_ID>
        <MENUITEM_ID>SUBMENU2</MENUITEM_ID>
</MENU>
```

采用这种方式要在 MENU 体外申明对应 MENUITEM 的定义，可以放在引用 MENU 之前也可在其后，但 MENU 中 MENUITEM_ID 包含的内容必须是已定义的 MENUITEM 的 ID，如果不存在系统将会将该菜单忽略。另外提一点，为了和先前的版本兼容，MTK 提供了@OID:前缀，用来引用原先在 c 文件中定义的菜单 ID，使用方法是将其放在<MENUITEM_ID></MENUITEM_ID>标签对之间即可，例如：
<MENUITEM_ID>@OID:SUBMENU3</MENUITEM_ID>。不过引用前请使用<INCLUDE file="XXXResDef.h"/>引用你 ID 所在文件，将其至于<!--Include Area-->下。

方法二：直接将 MENUITEM 定义在 MENU 中，格式如下：

```
 <MENU id="MENU1" type="OPTION" str="STR_MENU1" highlight="HighlightMenu1" hint="HintMenu1">

        <MENUITEM id="SUBMENU1" str="STR_SUBMENU1">SUBMENU1</MENUITEM_ID>
        <MENUITEM id="SUBMENU2" str="STR_SUBMENU2">SUBMENU2</MENUITEM_ID>
 </MENU>
```

这种方式不需要在 MENU 体外定义对应 MENUITEM，只需要放在 MENU 标签内同时定义 MENUITEM 信息即可。

方法三：嵌套 MENU，格式如下：

```
<MENU id="MENU1" type="OPTION" str="STR_MENU1" highlight="HighlightMenu1" hint="HintMenu1">

        <MENU id="SUBMENU1" type="OPTION" str="STR_SUBMENU1" highlight="HighlightSubMenu1"
              hint="HintSubMenu1"></MENU>
        <MENU id="SUBMENU2" type="OPTION" str="STR_SUBMENU2" highlight="HighlightSubMenu2"
              hint="HintSubMenu2"></MENU>
</MENU>
```

采用这种方式使得菜单的定义一次完成，也易于理解，但是当属性较多层次较深的时候可能显得较乱。这种方式是直观的 MENU 树表现方式，显示了子菜单与父级菜单的对应关系。有些读者可能对 MENU 和 MENUITEM 有些迷茫，笔者认为，在大部分时候 MENU 和 MENUITEM 可以通用，你可以把他们看成一种东西（MENU）。你可以将方法三第二行替换成方法一或方法二的形式。不过如果有 SUBMENU1 有子菜单的时候，方法三可以直接嵌套在对应 MENU 体之中，而方法一或方法二则需在 MENU 方法体外做如下定义：

<MENU id="SUBMENU1" type="OPTION" str="STR_SUBMENU1" highlight="HighlightSubMenu1" hint="HintSubMenu1">

……

</MENU>

以上三种方法可以根据需要混合使用。最后为了完整实现菜单功能，别忘了添加对应菜单的 highlight 和 hint 函数。