doi: 10.7690/bgzdh.2024.11.006

基于 ARINC661 的 DF 文件验证平台

唐　乐，孙永荣，许舒晨
（南京航空航天大学自动化学院，南京 210016）

摘要：针对 DF 文件的高效验证需求，对陆空兼容的 DF 文件验证平台进行设计。通过对静态预检测、窗体部件的激励代理、激励实时响应和动态渲染置顶技术进行研究，并对平台进行测试验证。实际运行结果表明：该平台能提高 DF 文件的验证效率，为各类装备系统显示模块的开发与验证提供便利。

关键词：DF 文件；静态预检测；激励代理；实时响应；动态渲染置顶
中图分类号：TP273　　**文献标志码：**A

DF File Verification Platform Based on ARINC661

Tang Le, Sun Yongrong, Xu Shuchen
(*College of Automation Engineering*, *Nanjing University of Aeronautics and Astronautics*, *Nanjing* 210016, *China*)

Abstract: According to the requirement of efficient verification of DF file, a verification platform for DF file with air-ground compatibility is designed. Through the research of static pre-detection, incentive agent of window components, incentive real-time response and dynamic rendering top technology, the platform is tested and verified. The actual operation results show that the platform can improve the efficiency of DF file verification, and provide convenience for the development and verification of display modules of various equipment systems.

Keywords: DF file; static pre-detection; incentive agent; real-time response; dynamic top rendering

0　引言

ARINC661 协议服务于航空电子系统综合化模块化的发展需求[1]，在航空领域得到了成功应用并彰显了巨大优越性。其他装备系统也可借鉴航空电子的系统架构，也采用 ARINC661 协议下的标准化接口[2]，减少开发和维护成本，提升系统的通用性和可移植性。

DF 文件的引入实现了座舱显示系统（cockpit display system，CDS）与用户应用（user application，UA）之间的显控分离[3]，是整个显示系统的重要基础。设计完成的 DF 文件内容固定且通用，正确与否会直接影响整个系统的可靠性[4]。若显示系统运行中出现错误，需要 DF、CDS、UA 共同调试排查，由于三者之间相对独立，排查需要耗费大量的人力和时间，错误检测繁琐与定位困难[5]。为保证定义阶段 DF 文件的可靠性，提高 DF 文件的验证效率，有针对性地开发基于 ARINC661 的 DF 文件验证平台尤为重要。

笔者基于 DF 文件设计平台基础[6]，结合各类装备系统的 DF 文件验证需求，设计了兼容的 DF 文件设计与验证平台的软件结构，针对验证阶段的激励设置、激励解析、事件处理和渲染显示等需求，建立了可视化的 DF 文件生成与动态验证软件系统，为 DF 文件的开发与验证提供了便捷的软件平台。

1　设计与验证软件结构

DF 文件设计与验证平台的主要功能是完成对 DF 文件全生命周期的有效设计与便捷验证，支持调用 ARINC661 窗体部件库[7]，以可视化方式设置窗体部件属性来设计画面内容[8]，并自动生成 DF 文件。设计完成后，可基于 DF 文件便捷的设置主动激励与事件激励，并根据激励内容完成动态仿真验证。

针对 DF 文件的设计与验证需求，笔者研究的 DF 文件设计与验证平台的结构如图 1 所示，包含以下模块：

1) 文件设计模块：支持窗体部件库和符号库，支持窗体部件的个性化设计，提供树形结构预览与静态预览；

2) 文件处理模块：能实现 XML 格式 DF 文件的生成和加载，并完成到 BIN 格式 DF 文件的转换；

3) 仿真验证模块：支持激励库文件，激励处理单元能够生成符合 ARINC661 标准的激励数据 XML 文件，并按时间先后顺序自动组包以 BIN 格

收稿日期：2024-06-28；**修回日期：**2024-07-20
第一作者：唐　乐（1998—），女，江苏人，硕士。

式存储后发送，动态渲染单元加载 BIN 格式的 DF 文件，进行窗体部件的实例化，实时接收并解析激励数据，实时动态更新画面渲染状态。

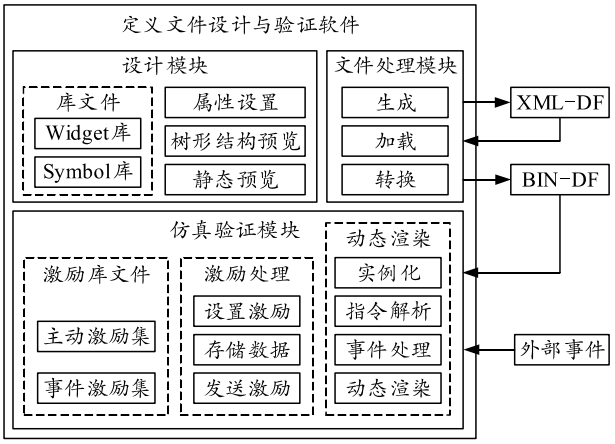

图 1 DF 文件设计与验证平台结构

笔者针对文件的验证需求，重点研究了软件结构中的激励库文件、激励处理、动态渲染等模块。动态渲染单元实例化 DF 文件，需要对 BIN 格式 DF 文件进行加载，笔者研究了加载过程中的静态预检测，保障 DF 文件结构内容的正确性，节约格式错误情况下的验证时间；动态仿真过程中会产生庞大的激励数据流[9]，不停地设置窗体部件激励指令，频繁改动 DF 文件内容，造成 DF 文件版本不稳定，笔者基于窗体部件库的基础研究了窗体部件激励代理，以可视化方式设置激励数据，并维护 DF 文件版本稳定；激励数据以二进制形式传输，用户难以通过查看二进制数据的方式检测激励正确性，笔者研究了激励数据下的实时响应技术，方便用户对应查看显示画面中窗体部件的图形变化；前序窗体部件率先绘制，后序窗体部件滞后绘制[10]，画面动态更新过程中，用户重点关注的窗体部件极易被部分甚至完全遮挡，笔者研究了关键部件动态置顶的渲染技术，使关键部件保持在整个画面的最上层绘制，用户可以实时观测交互激励给画面带来的变化。

2 验证模块关键技术

2.1 静态预检测技术

动态渲染单元首先需要对 BIN 格式 DF 文件进行加载，才能进行后续的实例化显示和动态仿真验证。若在动态仿真过程中出现问题，难以定位是 DF 文件结构格式错误还是参数设置问题；因此，需要在文件加载过程中对静态的 DF 文件进行错误预检测，定位加载过程中的错误情况，便于用户查找问题，并节约后续验证时间，提高 DF 文件的验证效率。

ARINC661 协议规范了 BIN 格式 DF 文件中数据参数的格式结构，每个数据块都按照关键字、块大小和具体数据内容的形式进行存储，动态渲染单元基于该结构加载 DF 文件。加载器首先检查文件头关键字 A661_DF_MAGIC_NUMBER，获取 DF 文件的头部信息；若自定义符号或图片存在，将展开对符号数据段或图片数据段的加载；然后加载 DF 文件的主体内容，即窗体部件定义块；窗体部件定义块读取完毕后，检查 DF 文件结尾关键字 A661_DF_FOOTER，完成整个 DF 文件的加载。

窗体部件定义块以层起始关键字 A661_BEGIN_LAYER_BLOCK 为开始标志，读取创建结构关键字 A661_CMD_CREATE 后，根据窗体部件类型调用对应窗体部件的初始化函数；然后遍历窗体部件的创建结构表，设置对应的窗体部件属性参数，完成该窗体部件的节点的初始化；最后根据窗体部件的父 ID 号找到图层中的父节点，加入父节点的子序列，完成窗体部件结构树的构建。

加载过程中可能会出现以下错误，导致文件加载失败：

1) 关键字不匹配。ARINC661 协议规范了 DF 文件在 BIN 格式下关键字的数据格式，如数据块的起始关键字 A661_BEGIN_BLOCK 为 0xB0，若关键字没有按照协议要求在期望位置以期望数据出现，则关键字无法匹配。

2) 偏移数据剩余。每个数据段单元都包含描述数据块大小的字段，初始化函数的指针读取数据时根据数据块大小进行偏移，读取完所有数据长度后偏移量应该归零，若剩余字节不为零，则该数据段格式出错。

3) 内存申请失败。创建节点初始化时需要申请动态内存，若内存申请失败，则未能成功实例化该节点。

4) 结构树构建失败。实例化单个窗体部件块后需要根据其父 ID 将该窗体部件插入到父窗体部件或所在层的子序列中，若未能找到所在位置，则无法正确插入，难以构建后续的结构树。

为方便用户查看加载情况，软件将生成直观的 DF 文件加载日志，保存各节点的解析情况。若出现上述错误，加载器将获取错误类型和窗体部件 ID，输出至加载日志中，并终止该 DF 文件的加载流程。

2.2 激励代理研究

动态仿真过程中会产生庞大的激励数据流，激

励设置过程中不停地设置窗体部件激励指令，若每生成一次主动激励，都直接使用其中的各个窗体部件属性参数值作为编辑对象，频繁改动 DF 文件内容，会导致已实例化的数据被修改或者数据成员被重新定义，造成 DF 文件之间版本不匹配，使得模块的更新维护困难。

基于以上问题，笔者基于窗体部件库的基础研究了窗体部件激励代理，提取窗体部件库中的各窗体部件的部分特征，修改数据过程中，不修改窗体部件自身属性参数，只设置其激励代理。如图 2 所示，窗体部件中的圆圈指向一个激励代理，激励设置阶段仅显示窗体部件的运行可变属性，供用户根据需求设置激励指令，仅设置窗体部件激励代理的参数，不会改动 DF 文件的原始数据，保证 DF 文件的稳定性和安全性。

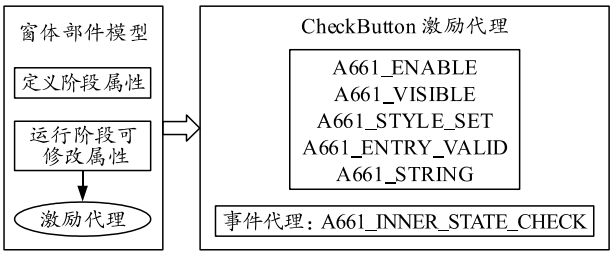

图 2　窗体部件激励代理

为了实现对不同窗体部件的激励设置，需要为各个窗体部件设置激励代理，软件中表现为可供用户操作的激励属性框。图 2 以单选按钮 CheckButton 为例，提取其运行阶段可变参数：可见性 Visible、能动性 Enable、字符串 LabelString、风格样式 StyleSet、有效条目参数 EntryValid，作为代理 CheckButton 的主动激励代理集；另外，可交互式窗体部件有事件激励产生，因此还要将其事件属性 CheckButtonState 放入事件激励代理中，构成运行阶段激励集合。图中 A661_ENABLE 等属性表示其并非窗体部件实际属性，与窗体部件本身加以区分。

2.3　激励实时响应技术

仿真动态运行阶段，激励设置单元发送组包后的激励数据包，渲染显示单元接收到这段激励指令后，激励响应由图层管理，图层在运行性阶段接收激励指令之后，根据激励指令的内容设置自身的可见性或者激活态后，修改对应窗体部件的属性参数。

ARINC661 协议中运行阶段的激励数据符合标准激励指令块结构，由激励起点 A661_BEGIN_BLOCK、激励数据总大小 BlockSise、具体激励内容和激励终点 A661_END_BLOCK 组成。为提高解析效率并减少错误发生率，节约激励响应时间，筛选阶段先将整包数据按照激励数据块进行分割，判断该激励数据块的起点是否符合协议要求，获取该激励数据块的总体大小后，判断激励终点是否符合协议要求，将不符合协议要求的激励数据块丢弃并报告错误信息，直接进入下一个激励数据块，将符合要求的激励数据块加入待响应队列，直到整包数据全部筛选完毕。

通过筛选，符合结构要求的激励数据块进入响应流程。激励数据类型分为 UA 请求激励 A661_CMD_UA_REQUEST 和参数设置激励 A661_CMD_SET_PARAMETER 2 种。以参数设置激励指令为例，响应流程如图 3 所示。

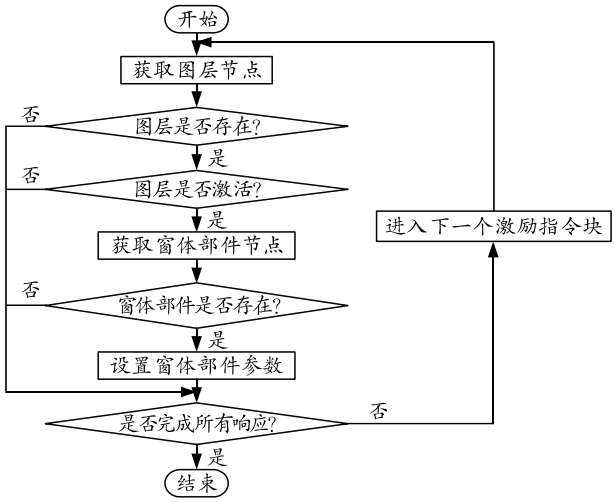

图 3　激励实时响应流程

对于队列中的激励指令块，判断激励指令类型后，获取图层节点信息，并判断图层是否存在，若图层不存在或未激活，则结束该激励指令块的响应进程，进入下一个激励指令块；若图层存在，则获取图层内的窗体部件信息，判断窗体部件存在后，响应激励指令内容，即修改对应的窗体部件参数，直到响应完所有的有效激励指令。

2.4　动态置顶渲染技术

窗体部件响应激励指令，属性参数发生修改，需要对应更新画面内容。原有的渲染架构遍历窗体部件树形结构，前序窗体部件率先绘制，后序窗体部件滞后绘制，画面动态更新过程中，用户重点关注的窗体部件极易被部分甚至完全遮挡，无法给用户实时展示激励响应后的画面变化情况。

针对上述情况，笔者将响应激励的窗体部件定义为关键部件，研究了关键部件的动态置顶渲染技术，优化现有渲染架构，分离窗体部件信息采集和

渲染过程，利用链表灵活多变的结构在渲染过程中将关键部件置顶显示。

为简化每个渲染周期内遍历窗体部件结构树的过程，节约信息采集时间成本，同时便于关键部件的置顶显示，笔者采用链表结构实例化窗体部件树，将可见图层内的可见窗体部件存入待渲染链表，过滤不可见图层以及可见图层内的不可见窗体部件。

实现动态置顶渲染分为 2 步：1) 置顶关键部件所在图层，获取关键部件的图层号后，先遍历渲染其他可见图层内的窗体部件链表，最后渲染关键部件所在图层；2) 在图层内将关键部件置顶，图层内的动态置顶渲染流程如图 4 所示。首先获取图层内激励指令指向的窗体部件号，遍历该图层的待显示窗体部件链表，定位关键部件并移动至链表末尾。若关键部件为容器型窗体部件，还需遍历容器内的子窗体部件，将可见的子窗体部件移动到链表末尾，若子窗体部件中仍存在容器型窗体部件，则递归遍历直到所有窗体部件遍历完成。

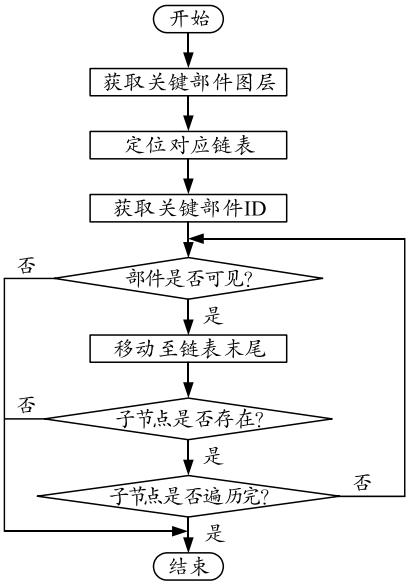

图 4 图层内关键部件置顶流程

将关键部件后移至窗体部件链表尾端，将其保持在整个画面的最上层绘制，用户可以实时观察交互激励给画面带来的变化，不会被其他内容遮挡。

3 功能实例测试

动态渲染模块加载 DF 文件，在加载过程中对静态文件结构和格式进行错误检测与定位，并记录到加载日志方便用户查看。如图 5 所示，符号数据块完成后，ID 为 1 的窗体部件在没能成功加入结构树，导致 DF 文件解析失败。通过错误检测与定位，初步保障 DF 文件结构内容的正确性，节约格式错误情况下的验证时间。

图 5 加载失败

通过对窗体部件激励代理的研究，可以维护 DF 文件稳定性，同时以可视化界面的方式设置并生成激励文件。如图 6 所示，图中左边显示的是 DF 文件的树形结构，右侧显示的是设置图层和窗体部件的激励属性栏，图中针对第 4 图层内名为地图基础容器的窗体部件，正在根据需求通过激励代理添加坐标参数 A661_POS_XY 的激励指令。通过设置所选窗体部件的激励代理，可以实现 DF 文件不同的激励数据配置。

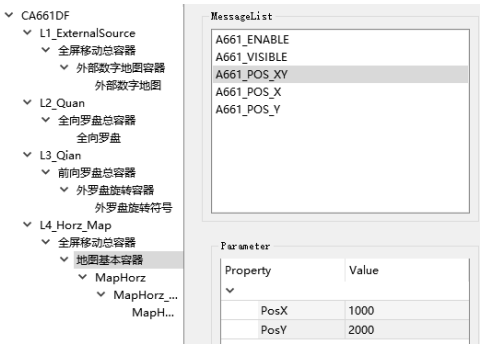

图 6 激励代理使用

添加完设置坐标参数的激励指令，软件自动更新生成激励 XML 文档如图 7 所示。

图 7 XML 激励文档

点击下方 Concvert 转换按钮后，可转换为如图 8 所示的符合 ARINC661 格式的二进制激励数据。

```
b0 01 00 00  00 00 00 1c  ca 03 00 08  da 01 00 00
ca 03 00 08  da 04 00 00  d0 00 00 00  b0 02 00 00
00 00 00 1c  ca 03 00 08  da 01 00 00  ca 03 00 08
da 04 00 00  d0 00 00 00  b0 03 00 00  00 00 00 1c
ca 03 00 08  da 01 00 00  ca 03 00 08  da 04 00 00
d0 00 00 00  b0 04 00 00  00 00 00 34  ca 03 00 08
da 01 00 00  ca 03 00 08  da 04 00 00  ca 02 00 18
00 02 00 00  b3 00 00 00  00 00 03 e8  b3 10 00 00
00 00 07 d0  d0 00 00 00  .. .. .. ..  .. .. .. ..
```

图 8　二进制激励数据

若 DF 文件通过了错误检测的筛选，成功完成加载，用户可以继续对该文件进行动态仿真验证。动态渲染模块接收激励处理模块发出的图层激活激励，对设置可见的图层进行实例化，并渲染图层内可见的窗体部件，动态渲染模块收到图层激活激励，根据时间戳请求激活并设置所有图层可见，当所有图层完成显示后，渲染效果如图 9 所示。根据图层显示的优先级，此时圆弧中心的部件正处于被遮挡状态。

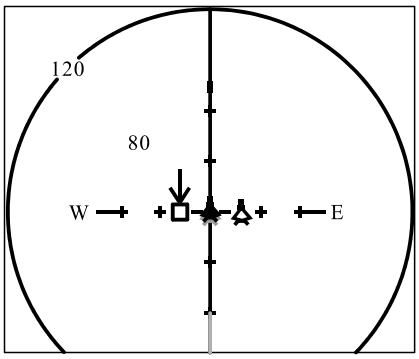

图 9　可见图层渲染

动态渲染模块收到设置参数激励，对窗体部件进行动态置顶。根据激励指令中的图层和窗体部件节点定位到 4 号图层内的地图基本容器，更新其坐标值并将其置于渲染链表的末尾。动态置顶后的渲染效果如图 10 所示，图中原本被部分遮挡的部件置于画面的最上层，可以完整显示。

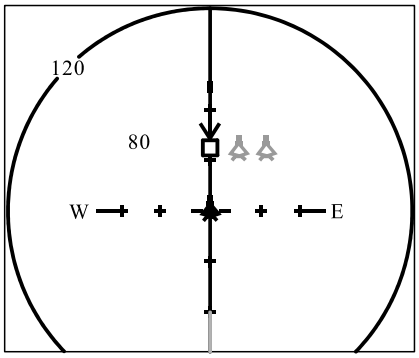

图 10　关键部件动态置顶渲染

4　结束语

笔者基于飞机座舱显示系统 DF 文件设计软件基础，研究并实现了兼容航空与地面装备的 DF 文件设计与验证平台的开发，研究了加载过程中的静态预检测功能，检测与定位可能出现的 DF 文件错误情况，保障文件结构内容的正确性，节约格式错误情况下的验证时间；设计了窗体部件激励代理，以可视化方式设置激励数据，并维护 DF 文件版本稳定；研究了激励数据下的实时响应技术，方便用户对应查看显示画面中窗体部件的图形变化；研究了关键部件动态置顶的渲染技术，关键部件保持在整个画面的最上层绘制，用户可以实时观测交互激励给画面带来的变化。经过测试和验证，可以实现各项功能需求，软件操作方便，能够满足各种装备显示系统的需求。

参考文献：

[1] Airlines Electronic Engineering Committee. ARINC specification 661-7 Cockpit Display System Interfaces To User Systems[S]. America: Aeronautical Radio, INC, 2019.

[2] 李颖洁, 王军, 李柏瑞. 座舱显示控制界面组件化设计方法研究[J]. 电光与控制, 2022, 29(3): 105−109.

[3] 赵科东. 基于 ARINC661 的 CDS 通用内核软件研究[D]. 南京: 南京航空航天大学, 2018.

[4] 崔诗娴. 基于 ARINC661 协议的 DF 文件验证方法[J]. 计算机系统应用, 2018, 27(2): 117−124.

[5] 李笑瑜, 孙永荣, 赵科东. ARINC661 多显示器座舱系统中的 UA 画面处理技术[J]. 电光与控制, 2018, 25(6): 95−97, 114.

[6] 赵文献. ARINC661 定义文件开发软件的设计与实现[D]. 南京: 南京航空航天大学, 2016.

[7] YOON J, BAEK N, LEE H. ARINC661 Graphics Rendering Based on OpenVG and Its Use Cases with Wireless Communications[J]. Wireless Personal Communications, 2017, 94(2): 175−185.

[8] BARLADIAN B K. SHAPIRO L Z, DERYABIN N B, et al. Efficient Rendering for the Cockpit Display System Designed in Compliance with the ARINC 661 Standard[J]. Programming and Computer Software, 2022, 48(3): 147−154.

[9] 冯悦, 孙永荣, 赵科东, 等. ARINC661 座舱显示总线数据过滤技术研究[J]. 电光与控制, 2017, 24(12): 47−50.

[10] 许舒晨, 孙永荣, 赵科东, 等. 基于 ARINC661 的多点触控手势数据提取算法[J]. 兵工自动化, 2020, 39(2): 20−23.