## 工作流图形方式设计芯片制造领域EAP程序的软件系统

### 芯片制造领域EAP系统简要说明

EAP（Equipment Automation Programming）实现了对生产线上机台的实时监控，是工厂自动化不可缺少的控制系统。EAP系统与FAB中的机台紧密相关，系统的设计与开发必须与生产线的机台实际生产流程相一致，才能达到自动化控制机台生产的目的。

EAP通过SECS/GEM国际标准协议与机台进行数据传输。SECS/GEM是半导体设备（半导体行业称设备为机台）必须遵循的一种国际通信协议。EAP就是通过此通讯协议与设备通信、传输数据、发送指令控制设备按照预先定义的流程进行生产加工，达到对设备远程控制和状态监控，实现设备运行的自动化。

### 技术背景

传统EAP编程主要根据剧本规范(以下称Scenario)设计文档使用通用编程语言(C# Java)等实现相关接口规范与函数功能,但当系统规模急剧扩大时,也产生了非常多的问题如:

1.使用传统编程语言由于不同人对不同编程语言的使用方式和习惯不尽相同,所以写出的代码可能会风格迥异甚至对统一份scenario都会出现完全不同的多种代码实现,随着项目规模的扩大这些代码可能会变得相当难以维护。

2.代码实现体现不出scenario上所描述的流程之间的关系,使得很多时候非常让人费解也很难从代码反向看出scenario中描述的流程,这样当其中任何一个步骤需要改变时,很可能会牵一发而动全身

3.多个SECS标准函数库的实现版本,SECS协议本是可以作为协议标准形成一个高度封装的库函数,但在这种由不同开发人工程师”自治”的代码结构中每个工程师都在实现自己的SECS协议类库而且互不兼容,这极大地破坏了整个编程高内聚低耦合的原则,对整个EAP的产品结构造成了极大的破坏，甚至出现问题时都很难修改,大大降低了系统的健壮性。

4.极高的实施成本,由于scenario流程图和scenario具体代码之间的双向转换存在巨大的鸿沟和差异所以EAP工程师在现场实施时要费尽心思做这层极不友好的转换,使得实施成本和时间急剧增加。

5.对于运行态的scenario只能依靠日志判断运行状态缺乏可视化和直观的图形界面展示，使得发现运行中的问题比较困难。

所有以上这些问题已经出现在了现有的EAP系统中,并已经产生了非常负面的影响。在这种背景环境下，研发图形化工作流引擎设计器(以下简称Workflow)为应对和解决这些问题的想法应运而生。

### 归类

芯片制造领域 -> EAP(Equipment Automation Programming)设备自动化编程 ->

工作流程序设计器。

### 现有技术

4.1手动编程

这种方式完全依靠实施人员纯手工编码。

存在问题和缺陷:

1. 开发人员费时费力
2. 代码规范和标准类库各异代码混乱不堪难以阅读和维护
3. 执行效率低下难以调试
4. 代码与剧本规范设计文档的结构相差甚远
5. 无法直观的以图形化的方式查看程序运行的状态
6. 客户无法理解程序造成难以沟通

4.2基于流程图通过’反射’方式’解释’执行各子模块(动态链接库)

这种方式先绘制流程图,然后通过特定的’解释程序’通过’反射’的方式将各个已经实现好的子模块(动态链接库)拼接起来,形成完整的程序调用和运行。

比起’手动编程’的方式这种方式已经进步了很多,但是仍然有许多问题和不足。

存在问题和缺陷:

1. 庞大的基础类库
2. 对复杂功能支持不足
3. 无法进行反向从代码到流程图的生成
4. 调试困难
5. 执行效率仍然低下
6. 无法支持多种编程语言
7. 自定义修改困难
8. 不易进行版本控制

### 发明内容

5.1技术方案

针对EAP系统 Workflow与上述方案不同而采用”三段式结构”的手段和措施来简化和解决上述方案出现的问题,该结构如下图所示:

配置剧本

设计流程

生成代码

首先使用流程图绘制器绘制出一张符合EAP剧本规范设计文档的流程图,其中的所有元素都支持插件(控件)式的定制和开发,不同元素之间根据剧本规范设计文档绘制出连线。

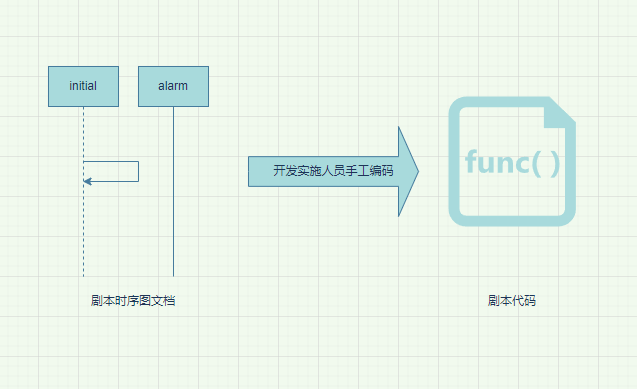
然后将这张图转换为一种特定的数据结构(ui json)描述,并将这种结构传递给分析器,分析器对结构进行分析并做数据的校验,若校验成功则将这个结构转换为相应的代码(该代码形式上式根据绘制的流程图进行生成)。

将生成的代码最后嵌入到需要使用该剧本脚本的宿主程序中,并做相应的数据配置,完成整个EAP程序脚本的开发。

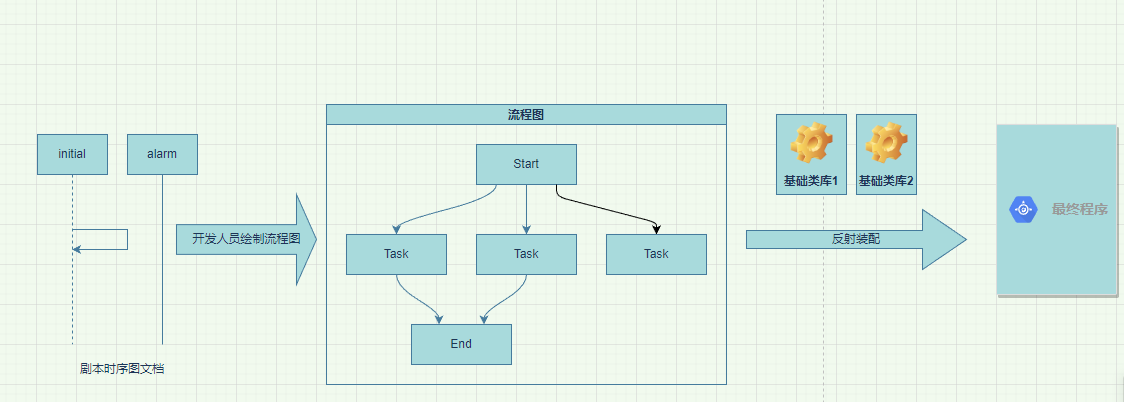
最后若将整个流程反过来,我们可以很自然的构建出一个从开发人员书写的符合一定规范的剧本脚本反向生成整个流程图的程序,从而完成从代码到流程图的转换而闭合上图中的循环结构。

下面以更形象的图形方式对比本方案和其他两种方案的特点。

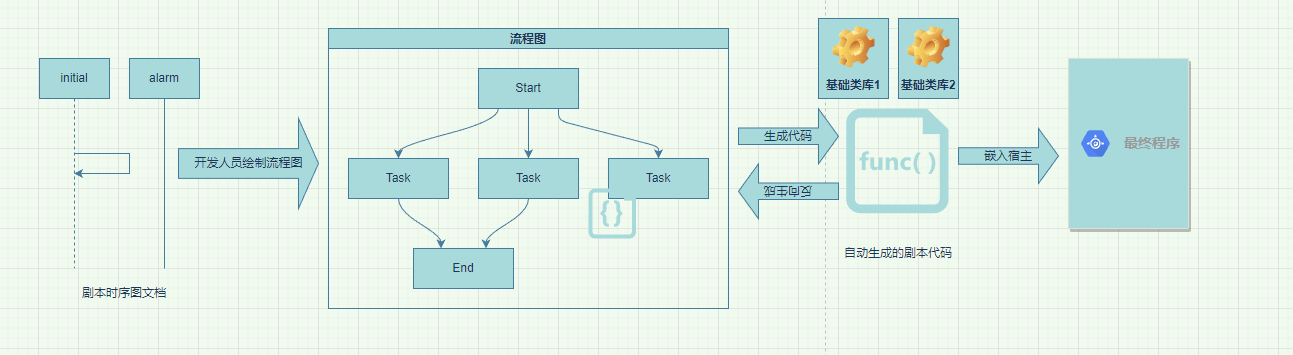
1. 手动编程方案:



1. ’反射’、’解释’和’拼装’方案:



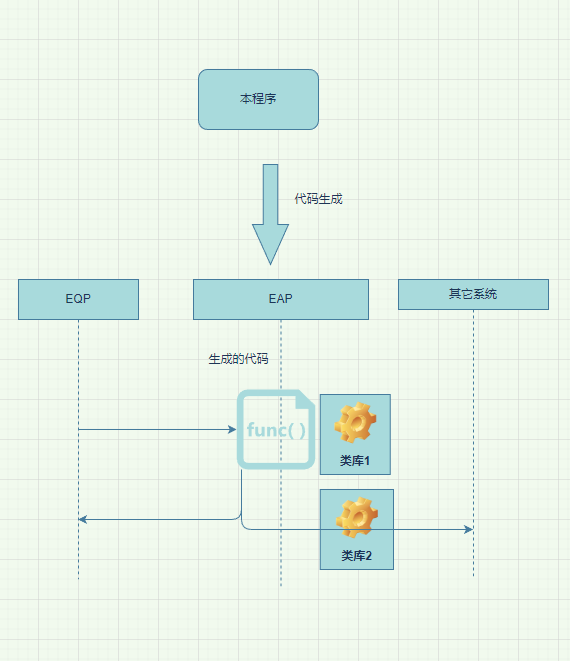
1. 本方案:



5.2本方案具体到每个阶段的技术方案示意图

本程序流程图示意

本程序所处的逻辑结构附图:



5.3说明

从本质上将该方案不同于上述两种方案,既不是纯手写也非代码模块的拼装。

而是讲流程图通过技术手段转换为所要的代码，即输入是流程图输出是EAP剧本代码，本程序则为生成EAP剧本代码的代码。

5.4提高和有益效果

1. 避免开发和实施人员从0开始一点点耗时耗力的手写代码提高效率
2. 自动生成的代码符合一定标准和规范具可读性,并且调用的基础类库不会重复开发
3. 由于类库相对标准所以避免了一些开发人员书写一些效率底下的代码程序质量得 以提高并且相对易于调试。
4. 流程图结构与剧本规范设计文档的结构比较近似,易于理解和参考。
5. 若程序运行时可以将当前运行步骤实时反应在流程图中。
6. 图形的方式表现程序的运行,客户更容易理解和沟通
7. 由于最终的产出物是代码,如果开发想要进行定制则直接修改代码即可,灵活度大 大提高。
8. 基础类库可配置从而大大提高了程序的扩展性
9. 由于是自定义的代码生成方式可以轻松实现不同目标的宿主机器程序和环境如(如 c# java c/c++ 其他 等等)。
10. 由于生成的程序可以是文本结构,所以若结合现代的运维技术做版本控制将会变得 非常容易。

### 实施案例

待补充...