# Model Transformation between MATLAB Simulink and Function Blocks

2010\_Yang\_Vyatkin\_INDIN.pdf

* 文章提出了一个方法： 合并 双向转换 FB 和 Simulink 来实现 automation systems的建模；
* Simulink适合control and embedded systems的modelling和simulation，FB适合设计distributed control systems；
* 对应Simulink建立的FB库：

ABS, CompareToConstant, CompareToZero, Derivative, Fcn, Integrator, LogicalOperator, OR, PRODUCT, SIGNUM, SRFlipFlop, Sum\_REAL, SWITCH\_REAL, TransferFcn

* 要想做好Simulink和FB互相转换工作，必须先理解两边模型的语义。

# Transformation of Simulink models to IEC 61499 Function Blocks for verification of distributed control systems

YV CEP 2012 proof.pdf

* 相较于会议论文，这篇期刊论文加入了一些形式化的证明，这很重要，值得我借鉴；
* 不管是IEC 61131-3还是IEC 61499，它们都是用于设计distributed control and automation systems；
* 在文章第一部分有提到，closed-loop plant-controller modelling进行system simulation是业界公认方法；
* 文章第三部分给出了Simulink和Function Block的形式化表达以及转换方法（要转换先得明白两边的语义）；
* 该转换方法是block to block的，所以1对1不需要证明，第四部分证明了多个block连接时转换的正确性。

# Automatic Model Generation of IEC 61499 Function Block Using Net Condition/Event Systems

Cheng\_Indin\_2008.pdf

* 当时的背景情况：存在多种实现FB的方法，作者想搞一个统一。所以文章提出了一个prototype model generator which translates function blocks into Net Condition/Event System(NCES)；
* 参考文献15。

# IEC 61499 Function Block Implementation of Intelligent Mechatronic Component

IEC 61499 Function Block Implementation of IMC-8 vv.pdf

* 没什么太多价值，不过提供了一个开发插件的理由，一个Simulink仿真的理由。

# Modelling of IEC 61499 Function Blocks a Clue to Their Verification

Wroclaw.pdf

# The IEC 61499 Function Block Standard: Overview of the Second Edition

61499\_ED2\_OVW.pdf

* 一篇关于IEC61499与FB的简要概述，可作为参考文献。

# The IEC 61499 Standard and Its Semantics

05352482.pdf

* 一篇介绍IEC61499的文章，IEC61499 bridge the gap between PLC programming languages and distributed systems；
* IEC61499用于做更抽象的System-level设计，不用考虑具体的hardware architecture；
* 在“Elements of IEC 61499 Architecture”一节中，有一个很好的例子：a pneumatic cylinder，帮助我重新理解了SIFB（service interface FB）。FB需要靠SIFB与外界交互。
* ECC中的algorithm应该可以支持多种不同的语言，在我做的插件中，由于条件限制，只能接受文本形式的algorithm，而在文章例子中CYLINDER\_CTL的ECC有一部分是用ladder logic diagram（梯形逻辑图）描述的；
* Basic FB的ECC执行是原子的；
* “Conclusions”里面提出了两个future work。

# Combination of UML Modeling and the IEC 61499 Function Block Concept for the Development of Distributed Automation Systems

04178238.pdf

* 与我的插件目的相同，另一个方法。

# Automatically Generated Layered Ontological Models for Semantic Analysis of Componnet-Based Control Systems

06389772.pdf

* 文章是想自动检查FB的语法和语义；
* “The IEC 61499 function block standard uses XML representation format for all design artefacts.”，MagicDraw也支持XML格式存储，提供了有一个立足点；
* 暂缓阅读，可作为12月17号报告文章。

# IEC 61499 Ontology Model for Semantic Analysis and Code Generation

INDIN11\_DDV.pdf

* 没看出太多名堂，太抽象，转为看Automatically Generated … 一文，希望期刊论文能有更多的看点或者更为具体的例子。

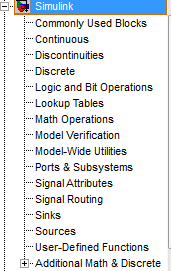
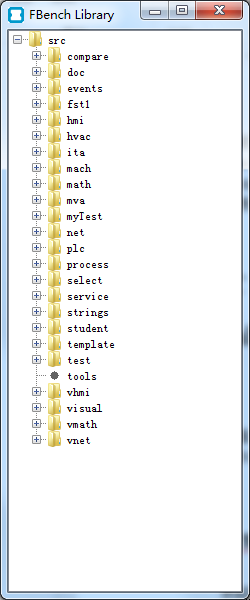
# FBench、FBDK与Simulink Library

思路整理：需要做的是从FB的系统层设计得到一个初始的详细设计，FB的系统层设计有MagicDraw的FunctionBlock插件完成；初始的详细设计由Simulink表达。

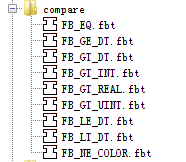
目前有两种思路可尝试：1.FB的概念设计->Simulink的初始详细设计；2.FB的概念设计->FB的中间层->Simulink的初始详细设计。

当前先整理FB的库与Simulink的Library之间的关联，带\*的为分布式大型控制系统相关。

FB的Library与Simulink的Library概览：



* compare包

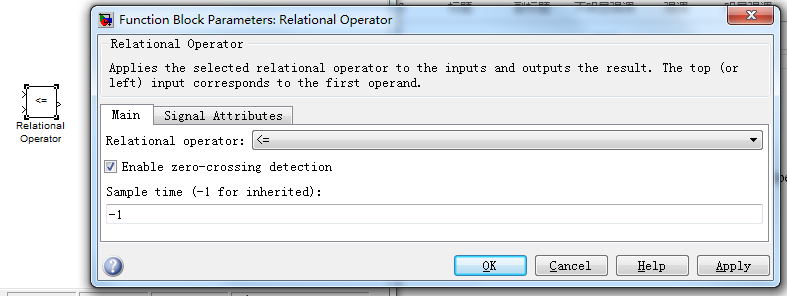


由于不同数据类型大小比较方法不同，所以设计了许多用于比较大小的function blocks。

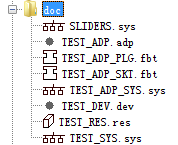
似乎概念设计不用涉及这么细节，只用知道是要比较数据就OK。

Simulink没有color和date这两种数据类型吧？

Simulink->Logic and Bit Operations->Relational Operator实现了比较功能：

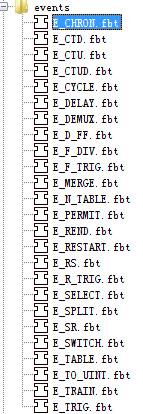


* doc包



功能暂不明了。

* events包

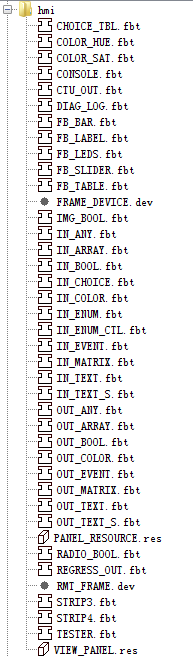


看到E\_D\_FF，我突然发现，某些时候要将Function Block应用于传统的电路设计（控制），与其说是Event，不如说是信号（Signal），Data是参数。

又或许是Pulse？

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Function Block | EventInput | DataInput | EventOutput | DataOutput | 作用 | Simulink |
| E\_CHRON | 计时器 | | | | |  |
| E\_CTD | CD | N/A | CDO | -- CV  Q = true when CV= 0 | 读入指定整数递减至0 |  |
| LD | PV (uint) | LDO | CV = PV  Q = false |
| E\_CTU | CU | PV (uint) | CUO | CV++  Q = CV >= PV | 从0开始递增，并判断是否不小于指定整数 |  |
| R | N/A | RO | CV = 0 |
| E\_CTUD | E\_CTU加强版，CV能递增也能递减 | | | | |  |
| E\_CYCLE | 每隔DT时间产生一个信号 | | | | |  |
| E\_DELAY | DT时间后产生一个信号 | | | | |  |
| E\_DEMUX | 多路输出信号选择器 | | | | |  |
| E\_D\_FF | 电平发生变化输出信号 | | | | |  |
| E\_F\_DIV | 不断递增，大于N时，输出信号 | | | | |  |
| E\_F\_TRIG | 上升沿有效，输出信号 | | | | |  |
| E\_MERGE | 两个输入合二为一，任意一个有效则输出信号 | | | | |  |
| E\_N\_TABLE | 轮转选择，每隔一定时间间隔，从1到N中选出一个事件 | | | | |  |
| E\_PERMIT | EI | PERMIT | EO  当PERMIT = true时才会有EO输出 | N/A | 开关 |  |
| E\_REND | El1 | N/A | EO  当El1、El2相继发生才会有EO输出 | N/A | ？ |  |
| El2 |  |
| R |  |
| E\_RESTART | COLD、WARM重启 STOP | | | | |  |
| E\_RS | S->R->S->R … | | | | |  |
| E\_R\_TRIG | 下降沿有效，输出信号 | | | | |  |
| E\_SELECT | 不太明白，当El1&G输出EO，当El0&!G输出EO | | | | |  |
| E\_SPLIT | 信号发散器，一个变两个 | | | | |  |
| E\_SR | 奇怪，跟RS一样 | | | | |  |
| E\_SWITCH | 二路选择器 | | | | |  |
| E\_TABLE | 每隔DT循环选择 1~N~1 | | | | |  |
| E\_TO\_UINT | 根据信号选择数字输出 | | | | |  |
| E\_TRAIN | 每隔DT产生一个信号，一共产生N个信号 | | | | |  |
| E\_TRIG | 电平变化输出对应信号（上升沿、下降沿） | | | | |  |

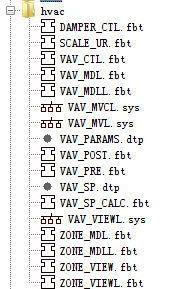
* hmi包 人机接口



看了几个发现该包大部分都是用来表达某种意思，并不产生实际的效果，暂时不细看了。原来这个是人机接口的包。

|  |  |  |
| --- | --- | --- |
| Function Block | 作用 | Simulink |
| CHOICE\_TBL | 生产了一个下拉菜单 |  |
| COLOR\_HUE | 取得一个颜色的三原色中红色的数值 |  |
| COLOR\_SAT | 取得一个颜色的三原色中绿色的数值 |  |
| CONSOLE | 输入什么输出什么，用来模拟控制台 |  |
| … |  |  |

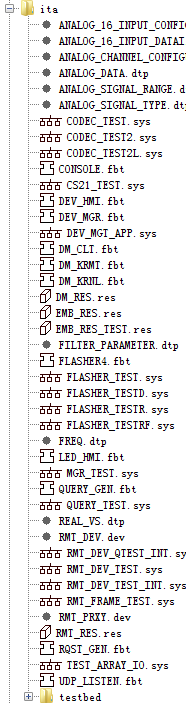
* hvac供热通风与空气调节



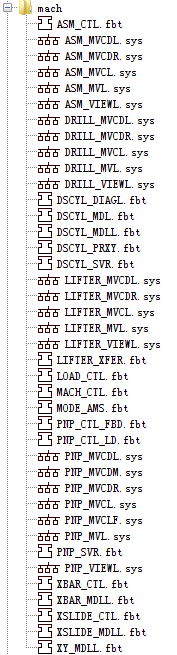
|  |  |  |
| --- | --- | --- |
| Function Block | 作用 | Simulink |
| DAMPER\_CTL | 阻尼控制器 |  |
| SCALE\_UR | 归一化 |  |
| VAV\_MDL | 空气控制阀 |  |
| VAV\_POST | Variable Air Valve Model post-calculations |  |
| VAV\_PRE | Variable Air Valve Model pre-calculation |  |
| VAV\_SP\_CALC | Calculate Variable Air Volume Controller Set Points |  |
| … |  |  |

* ita

包的具体领域未知，但与我想做的事情没有关联。



* mach



都是具体的机械控制器或者系统。

|  |  |  |
| --- | --- | --- |
| Function Block | 作用 | Simulink |
| ASM\_CTL | Controller for Simple Assembly Station with Dual-Cylinder Transport |  |
| ASM\_MVCDL | Assembly System Model+View+Control+Diagnostics+HMI |  |
| … |  |  |

* math 数学运算包

这条路黯淡无光，看上去是死路啊。Function Block Library也全部是具体产品，要从Library入手的话，最终会得到的也是一对一的mapping，或者另外一种形式的mapping。

关注Function Block的优点：

易分布；

重用；

重构；

事件驱动。

选择MagicDraw的原因：

适合做概念设计；

比现有的FB设计工具具有更强的图形交互界面；

现有的FB设计工具，不管是FBDK还是FBench都有一定程度的不完善，对于图形化的操作支持的不好，设计起来很费力。

而且这两款工具也不为大众所熟悉。

现在换一个思路，用户提出需求，专家进行分析，给出初始的状态变化设计图（MagicDraw）。

串联之前的工作，弯管机做第一个例子，V&V的Distribution Station做第二个例子。

具体工作在MagicDraw中已经完成，不过可能还需要调整和思考，有的步骤可大可小。

数学表达及规则定义：

一般的DFA（有限状态自动机）定义

五元组：

分别表示：

一个非空有限的状态集合Q，

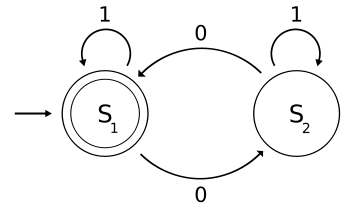
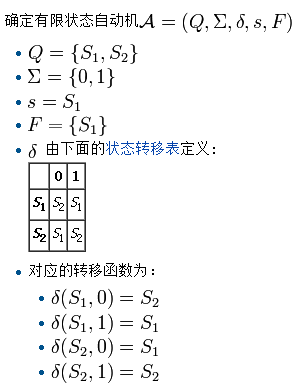
一个输入字母表（非空有限的字符集合），

一个转移函数，

一个开始状态s，

一个接受状态的集合。

简单的例子：

我所使用的过程化控制状态机形式化定义：

七元组

分别表示：

一个非空有限的状态集合Q，

一个输入字母表（非空有限的事件集合，其实也就是我们说的Constraints），

一个转移函数，

一个开始状态s，

一个结束状态的集合，

一个多对一的映射函数，

一个设备集合D。

与DFA经典五元组不同之处在于我的定义多了一个设备集合和与其有关的映射函数。这是因为在过程化控制当中，任何状态变化或者事件发生都与设备相关。在这里定义设备集合，是用来辅助之后Function Block的自动生成。

转换后的Basic Function Block定义为FB=（Interface， ECC， ALG， IV），

Interface = （EI， EO， DI， DO， IA， OA）：

EI（Event Inputs）是输入事件集合；

EO（Event Outputs）是输出事件集合；

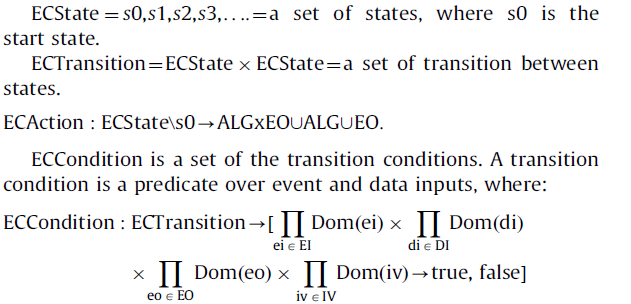
DI（Data Inputs）是输入数据集合；

DO（Data Outputs）是输出数据集合；

IA（Input Associations）是EI和DI的关联关系，；

OA（Output Associations）是EO和DO的关联关系，。

ECC = （ECState， ECAction， ECTransition， ECCondition）：



Composite Function Block只包含Interface和FBNetwork，FBNetwork由Composite Function Block和Basic Function Block组成。

Interface = (EI, EO, DI, DO, IA, OA)，定义与Basic Function Block一样。

转换Rules：

* 建立FB，初始化ECC的Rules

1. ；

文字描述：除了起始和结束状态以外的所有状态都能找到一个设备相对应。

1. , ；

文字描述：每一个事件都能找到一个设备相对应。

状态存在于Function Block内部的ECC中，事件发生于Function Block之间。而任何Function Block最终需要被部署到实际的Device中。

因此Device是最终的容器，之所以采用Device而不是Function Block是考虑到设计状态转换图的专家并不需要了解Function Block，但他肯定清楚某一状态，某一变化是发生于哪个Device中。

1. ，定义一个Function Block(d)与之对应，并在FB(d)的ECState加入初始状态s0；

文字描述：为每一个设备建立一个对应的Function Block，并置好初始状态。

1. , Function Block(d)的ECState加入状态q；

文字描述：把状态机中的所有状态复制到Function Blocks中。

完成状态初始化的工作。

* 构建ECTransition，丰富ECC以及Interface的Rules
  1. 路径不存在环

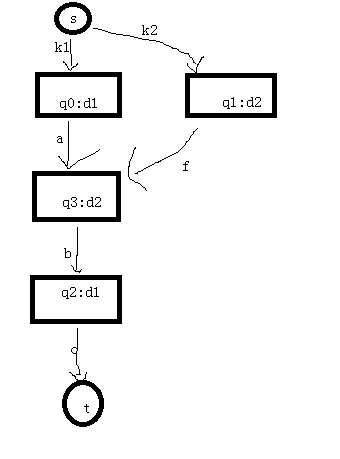
找寻路径上满足并且距离q最近的q’，如果不存在这样的q’，则令q’=s。在Function Block(d)的ECC中加入ECTransition：，其ECCondition为，并在EI集合中加入。

* 1. 路径存在环

探索到自己就结束。

，Function Block(d)的ECC中加入ECTransition：，其ECCondition为，并在EI集合中加入。

这一步是转换过程中最复杂的一步，语言描述或许不是很清楚，画图来说吧。

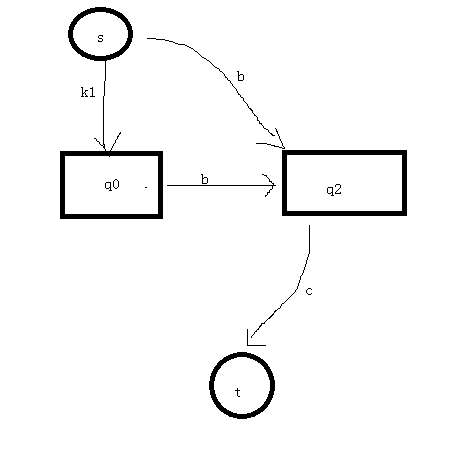


以q2:d1为例，有两条路到q2，分别是p1=s->q0->q3->q2和p2=s->q1->q3->q2。

p1中距离q2最近，且设备相同的是q0，所以加入q0->q2，条件为b；

p2中没有与q2设备相同的，所以取s，加入s->q2，条件为b；

所以最终d1的ECC为：



1. ，如果中，则Function Block(d)的ECState加入，ECTransition加入，其ECCondition为，并在EI集合中加入，EO集合中加入。

这一步是专门为作协调监控的FB准备的，最常见的便是传感器。

* 连接FB网络的Rules，构建最终的Composite Function Block = CFB

1. 对所有，
   1. 如果为1，则CFB的EI加入的状态的EI；
   2. 如果不为1，则CFB的EI加入的状态的EO。
2. 对所有，
   1. 如果CFB的EO加入的状态的EO；
   2. 如果不为1，则CFB的EO加入的状态的EO。

与外部接口已经建立完成。

1. 对所有，，
   1. 如果且，FB(d1)的EO(q)和FB(d2)的EO()连接到SPFB的EI，SPFB的EO连接到FB(d3)的EI(q)，SPFB起的作用为确保两个事件都已发生；
   2. 如果，FB(d1)的EO()连接FB(d2)的EI()。

以上是全部的Rules，由于时间仓促，所以现在有的表达还不完美，而且可能存在错误，接下来会完善Rules，并证明Rules的正确性，主要是FBNetwork连接的正确性。

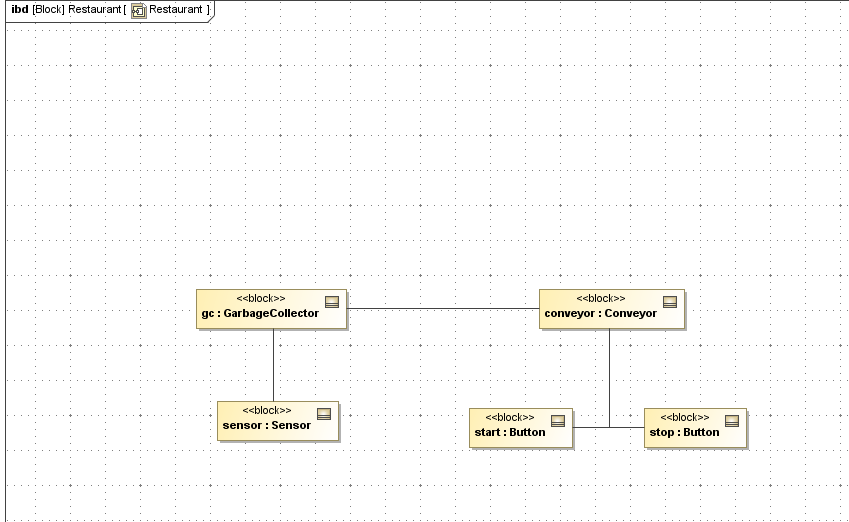
以一个简化的食堂传输带例子说明转换规则：

外部有两个按钮：一个开始，一个停止；

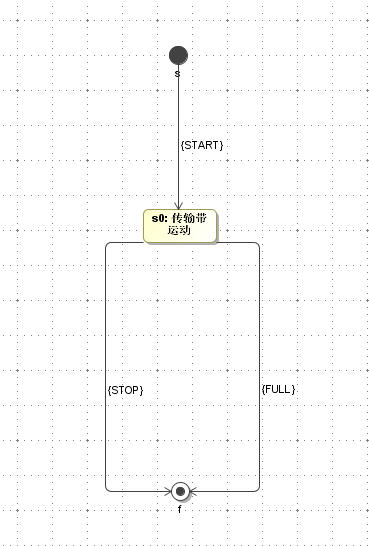
传输带不断传动；

垃圾箱上有一个传感器，能够检测垃圾箱是否满；

简化的机械构成：

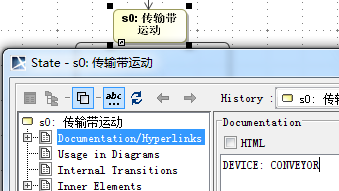
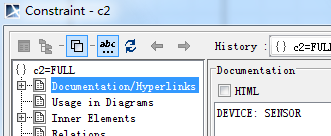


简单的状态机：



根据Rule1、2：

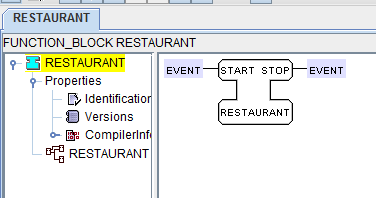
每一个状态和转移条件（约束）都需要标明Device。

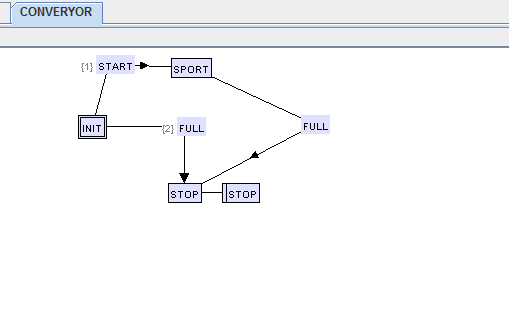
=CONVEYOR，=BUTTON，=BUTTON，=SENSOR。

根据Rule3：

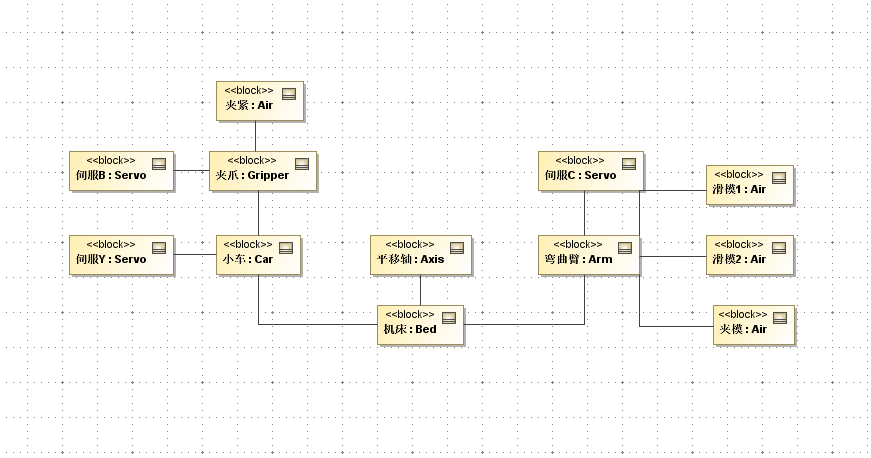
食堂模型Composite Function Block外部框架：



CONVEYOR ECC图：

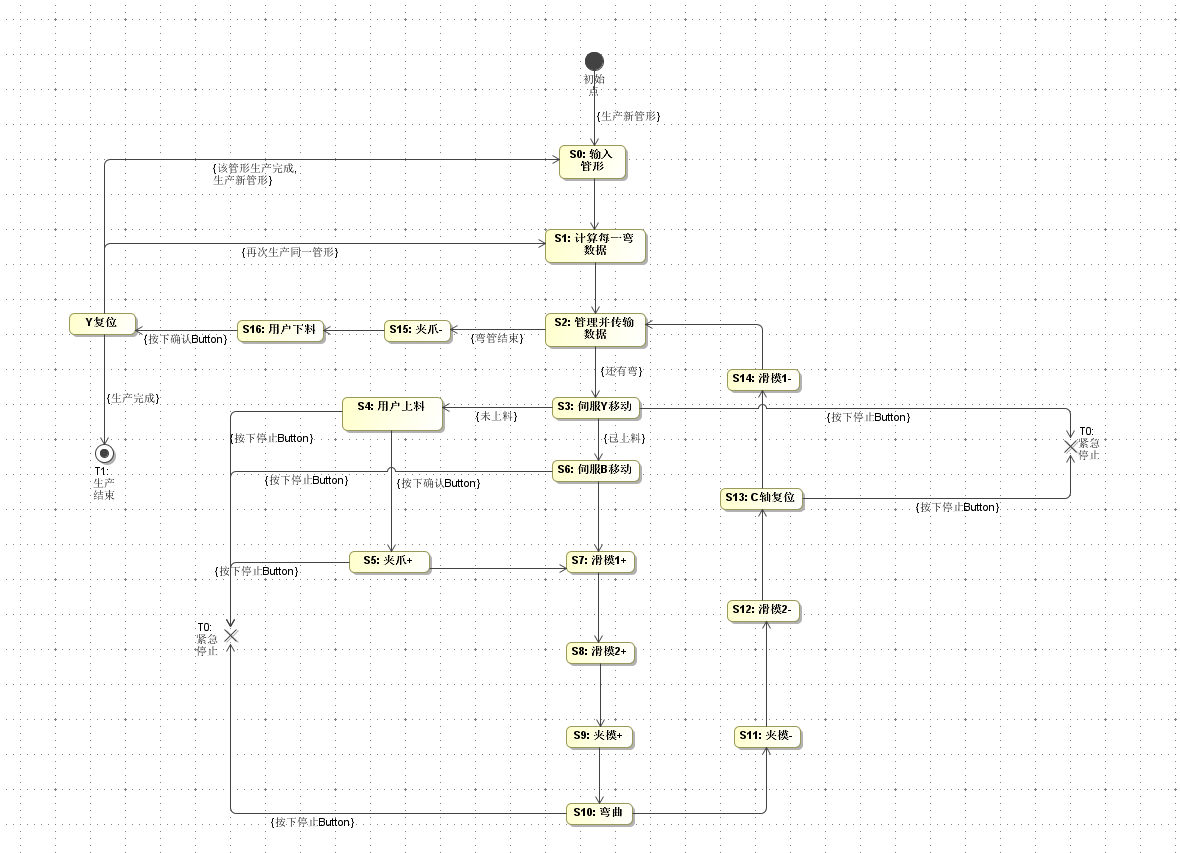


弯管机的机械部分主要设备组成：



弯管机的初步状态图如下：

需要改进点：将文字描述转换为数学符号表示，尽量用英文描述。



自动生成对应的Function Block图（暂时先用FBench描述）：