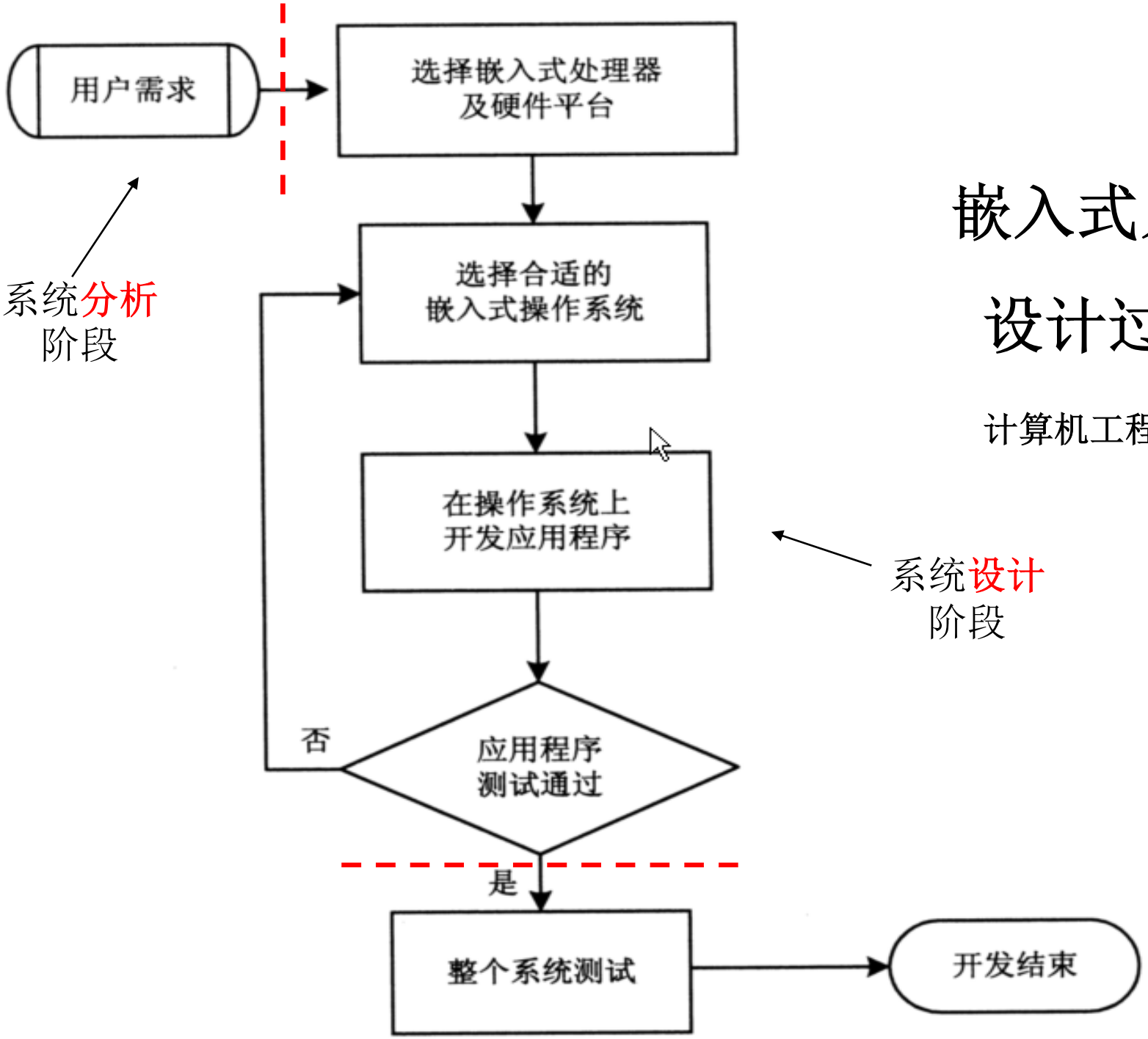
实时嵌入式软件设计

李曦

llxx@ustc.edu.cn

计算机系计算机应用研究室



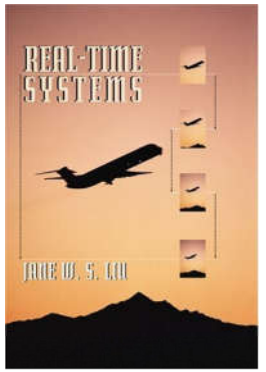
嵌入式系统设计过程

计算机工程view

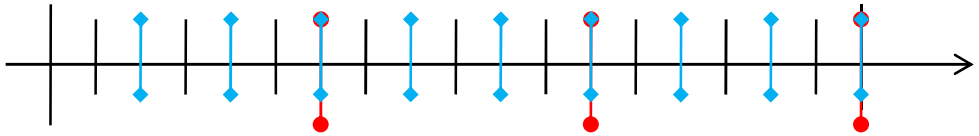
Helicopter Flight Control(多速率、并发)

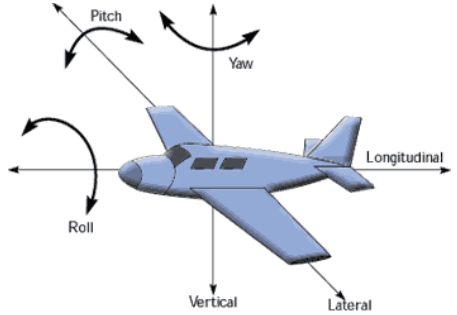
Do the following in each **1/180**-second cycle:

- Validate sensor data and select data source; on failure reconfigure the system.
- Do the following 30Hz avionics tasks, each once every 6 cycles:
  - Keyboard input and mode selection
  - Data normalization and coordinate transformation
  - Tracking reference update
- Do the following 30Hz computations, each once every 6 cycles
  - Control laws of the outer pitch-control loop
  - Control laws of the outer roll-control loop
  - Control laws of the outer yaw- and collective-control loop
- Do the following 90Hz computations, each once every 2 cycles, using outputs produced by the 30Hz computations
  - Control laws of the inner pitch-control loop
  - Control laws of the inner roll- and collective-control loop
- Compute the control laws of the inner yaw-control loop, using outputs from the 90Hz computations
- Output commands to control surfaces
- Carry out built-in-test



任务图?



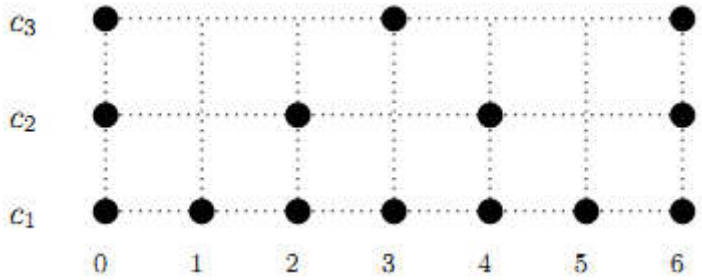


Multi-rate Control Systems

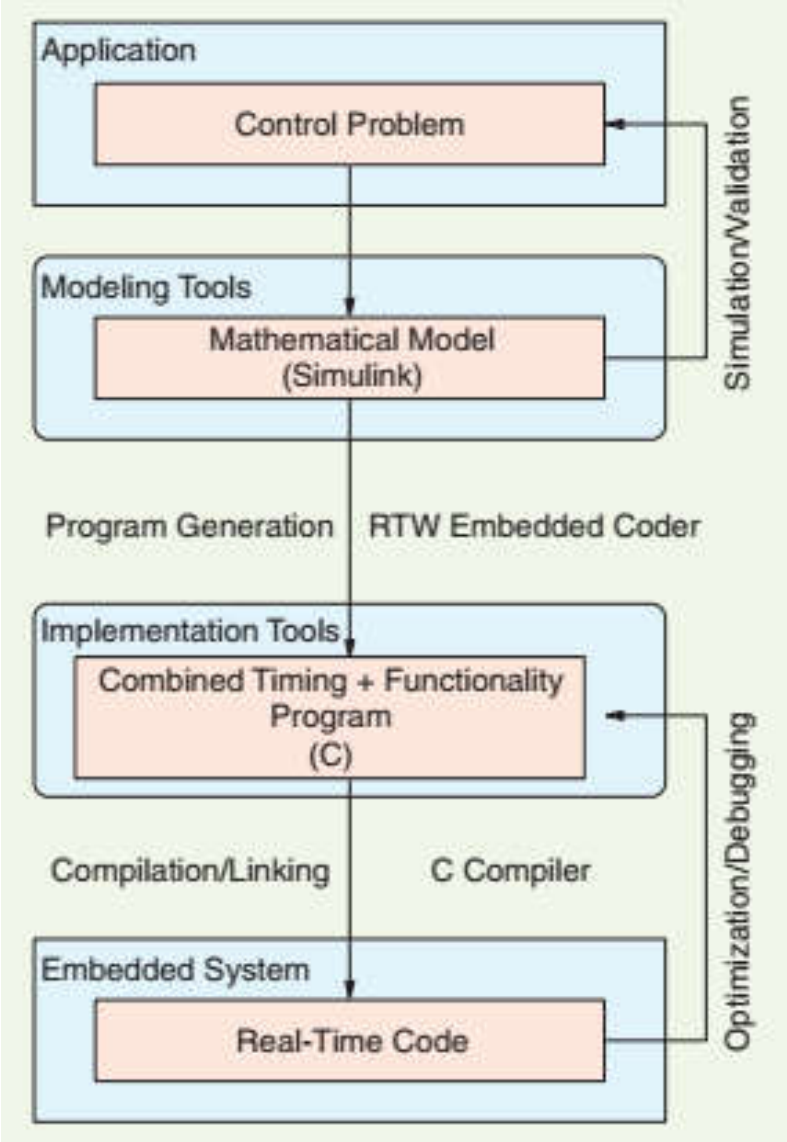
- More complicated control systems have multiple sensors and actuators
  - must support control loops of different rates.
    - Having only *harmonic rates* simplifies the system.
  - EX. Helicopter flight controller
    - The sampling rates of A, B, and C are 180Hz, 90Hz, 30Hz resp.

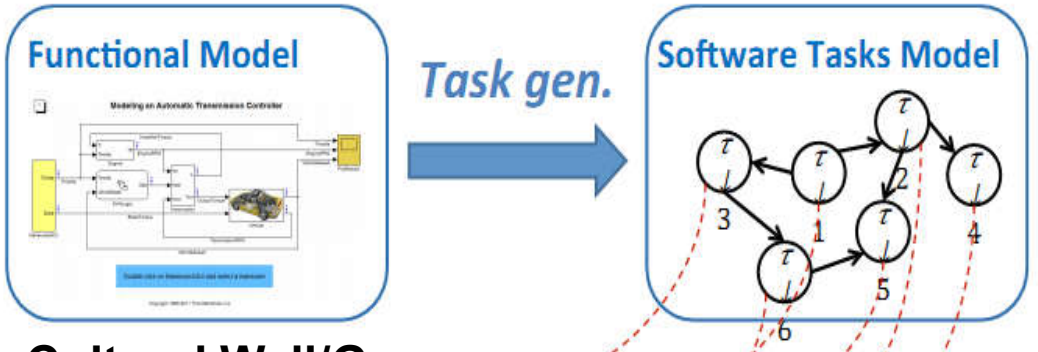
Do the following in each 1 / 180 second cycle:

- Do **A**
- Do **B** once every 2 cycles
  - ……
- Do **C** once every 6 cycles
  - ……
- Output commands
- Wait until the beginning of the next cycle



From CE to CE： A two-steps design

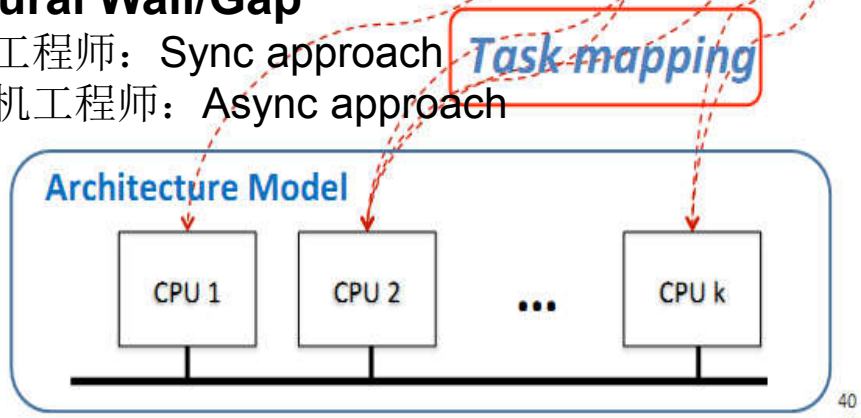


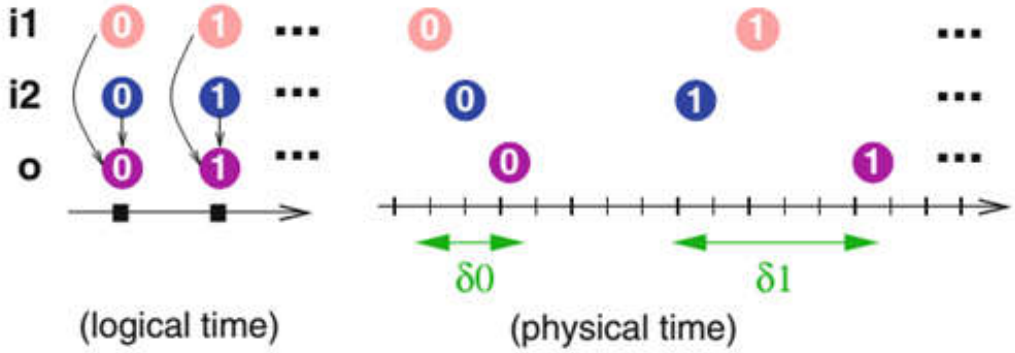


Cultural Wall/Gap

控制工程师：Sync approach

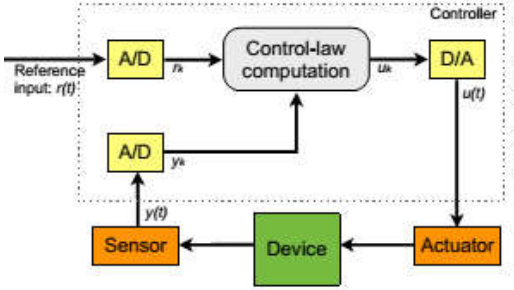
计算机工程师：Async approach

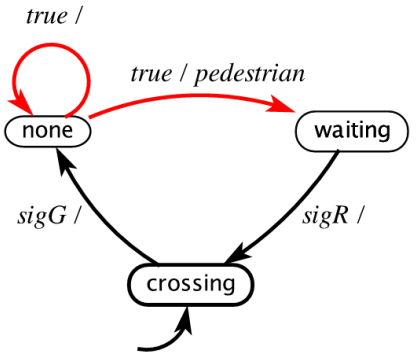


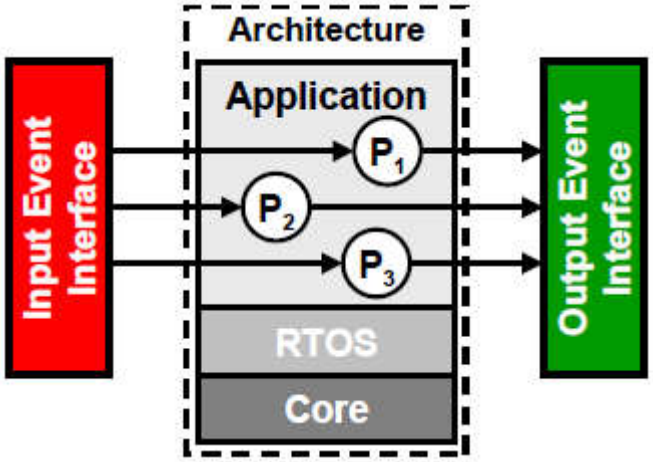


Gap between two domain

- 反应式系统
  - Stimulus-Response：FSM
    - 同步组合
  - logical concurrency
    - sync：temporally deterministic
    - flow-analysis：response-time，e2e delay
- 实时系统
  - multitask models：concurrent tasks
    - Event-driven
    - priority，deadline
    - schedulability analysis
  - physical concurrency
    - async：nondeterminism
    - arrival order/computation time

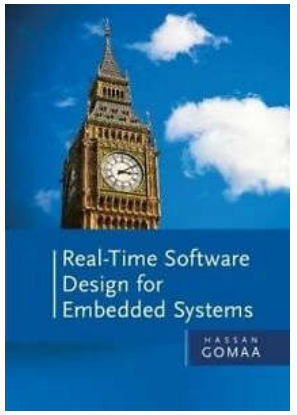


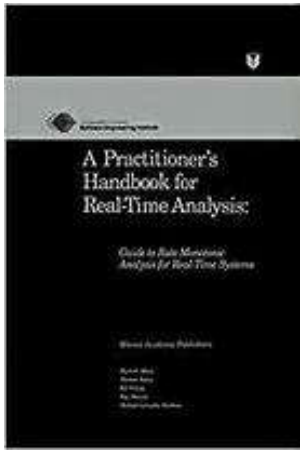




内容提要

- 控制系统设计过程
- 软件设计：*软件体系结构（组件及其关系）*，任务抽象，实现
  - Event-action模型
    - 识别事件和动作，任务=功能or时间聚合
  - DARTS方法：Design Approach for Real-Time Systems
    - 基于系统的DFG进行任务划分
    - 实时软件分析设计方法DARTS：Hassan Gomaa提出
- 实时编程范式(programming paradigms)/编程模型
  - 编程范式：way of thinking about programming
  - 同步方法/异步方法：同步/异步语言
- 参考资料
  - Real-Time Software Design for Embedded Systems，2016/18译
    - 采用SysML、UML和MARTE，从用例到完整软件体系结构，COMET/RTE设计方法。
    - SEI@CMU推荐教材
  - A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems，1993



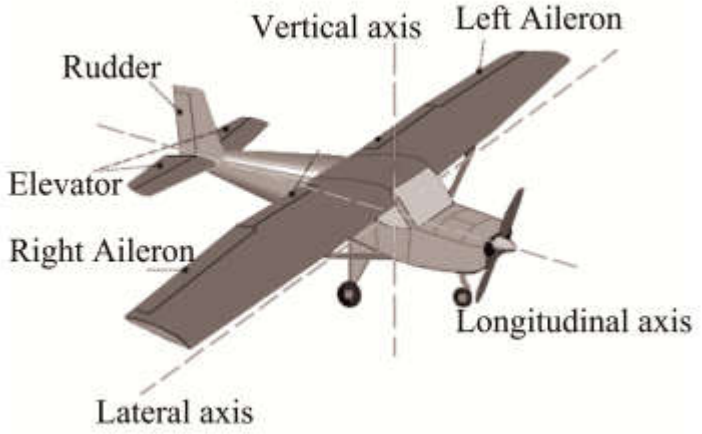


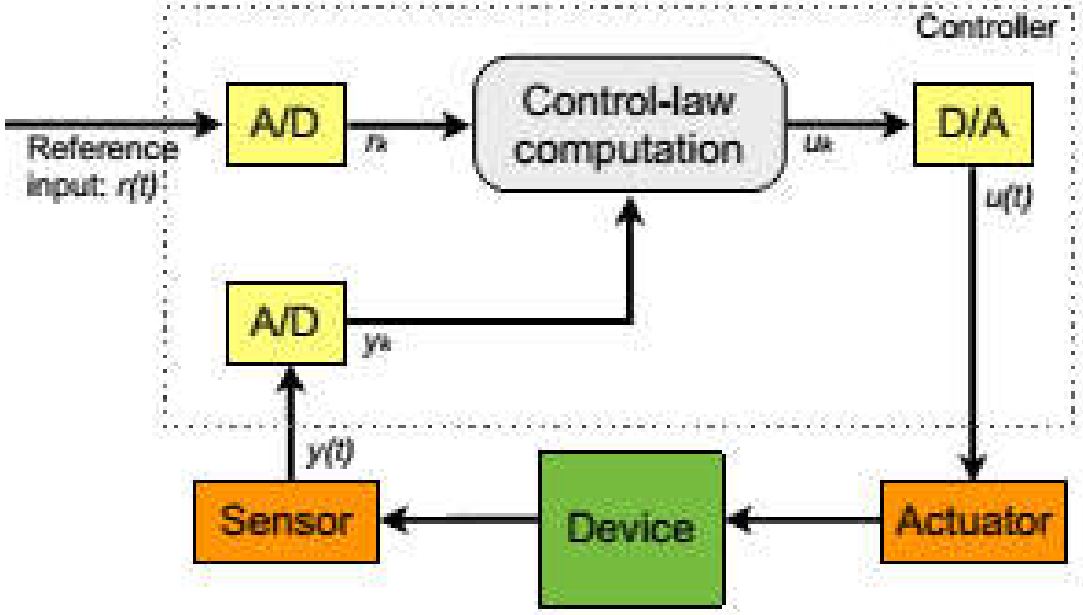
control systems设计流程

- The capture/convert design process：two steps
- the control engineer：将数学模型转换成控制模型
  - modeling of the plant behavior and disturbances
  - deriving and optimizing control laws
  - validating functionality and performance of the model through analysis and simulation
- the software engineer
  - decomposing the necessary computational activities into periodic tasks
  - assigning tasks to CPUs and setting task priorities to meet the desired hard real-time constraints under the given scheduling mechanism and hardware performance
  - achieving the desired degree of fault tolerance through replication and error correction
  - code integration, testing, and optimization

Controller: Control System

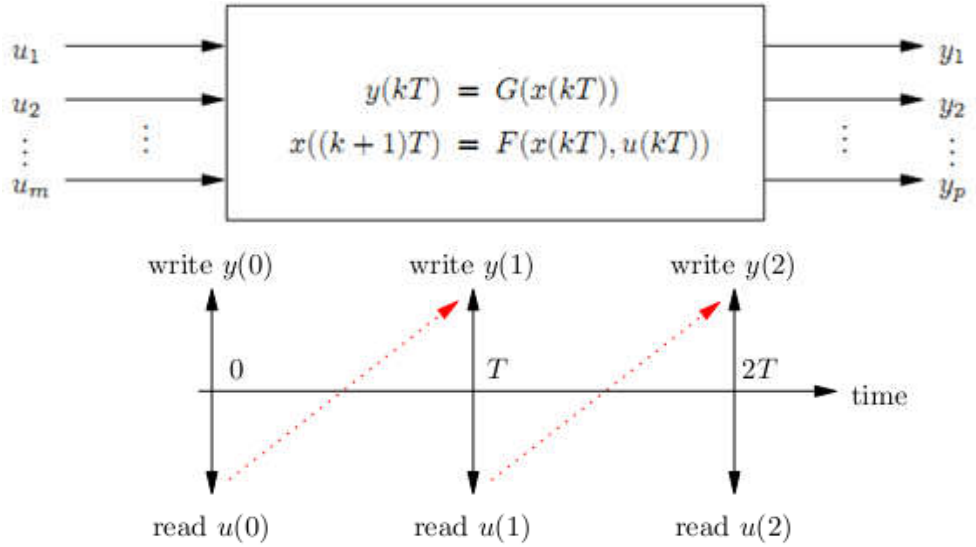
- Control a device using actuator, based on sampled sensor data
  - Control loop compares measured value *y(t)* and reference *r(t)*
    - Depends on correct control law computation, reference input, accuracy of measurements
  - Time between measurements of *y(t), r(t)* is the sampling period, *T*
    - Small *T* better approximates analogue control but large *T* needs less processor time;
    - if *T* is too large, oscillation will result as the system fails to keep up with changes in the input !



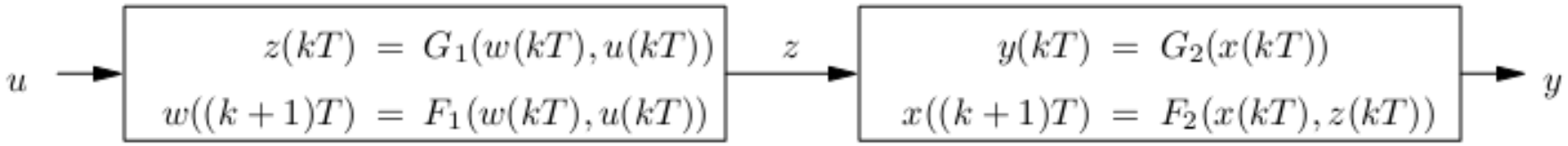


Control Law computation

- sampling period, *T*
  - reads its inputs *u* ,
  - computes its outputs *y* ,
  - up dates its state *x*
- control equations
  - *output equation*
    - *y(kT)，z(kT)*
  - *state equation*
    - *x((k+1)T)，w((k+1)T)*
  - 时序假设（两种同步语义）：
    - one-step delay同步：Moore
    - zero time同步：Mealy
- *control systems composition*




```
set timer to interrupt periodically with period T;
at each timer interrupt do
    do analog-to-digital conversion to get y;
    compute control output u;
    output u and do digital-to-analog conversion;
od
```

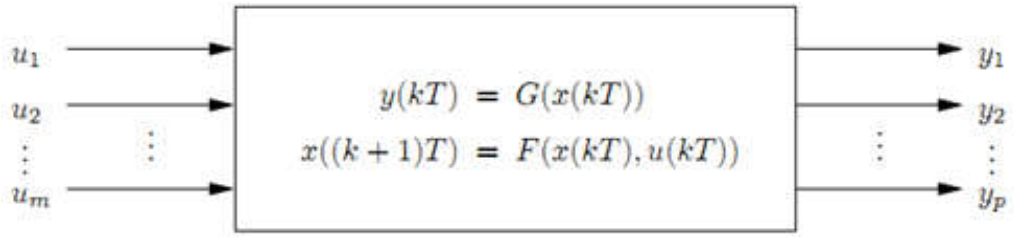


RT Control Systems Implementation

- The controller algorithm is executed once in every sampling period *h.*
  - The sampling period is a compromise between Nyquist-Shannon Law *(f_s>2B),* the computation time delay τ with it's possible jitter and the limits of the hardware.
  - *一般，周期抖动<1%*
- When in a system we have *0*<τ<*h* we're facing a delay, as for when $\tau \geq h$ we have a loss.

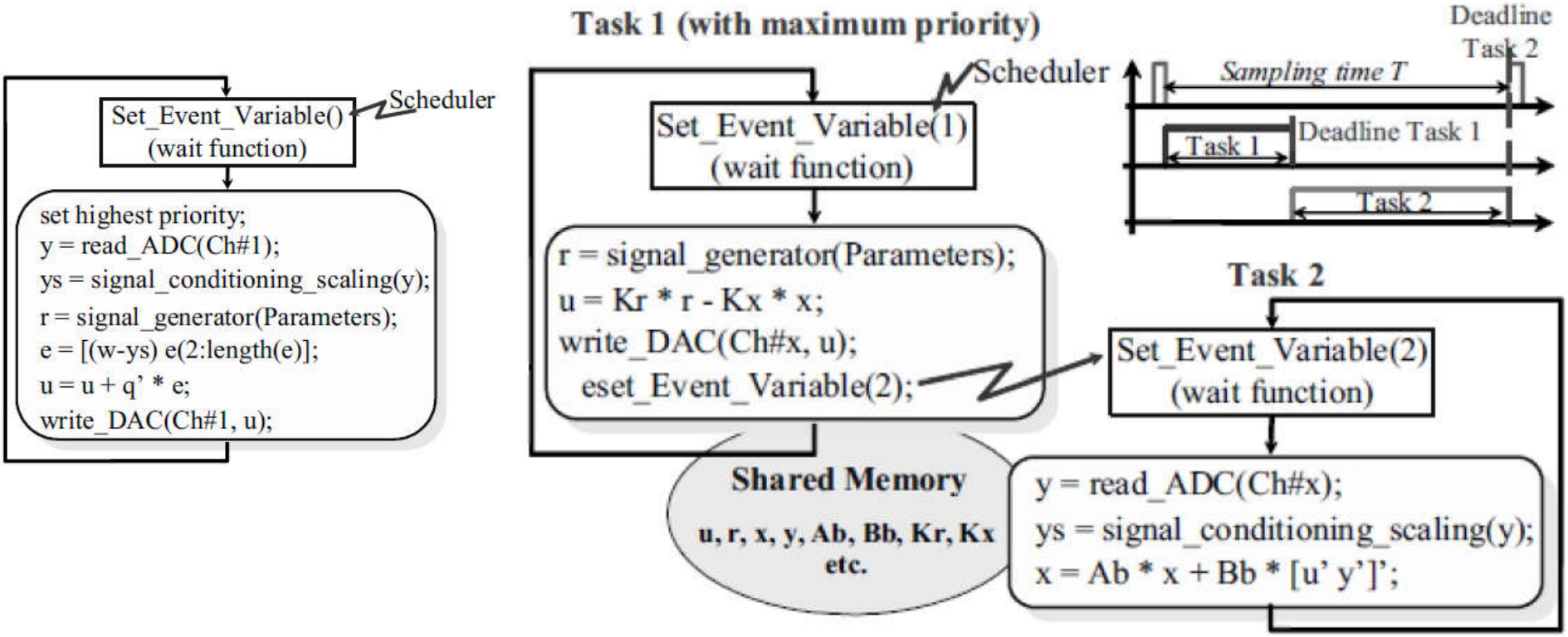
单任务与多任务

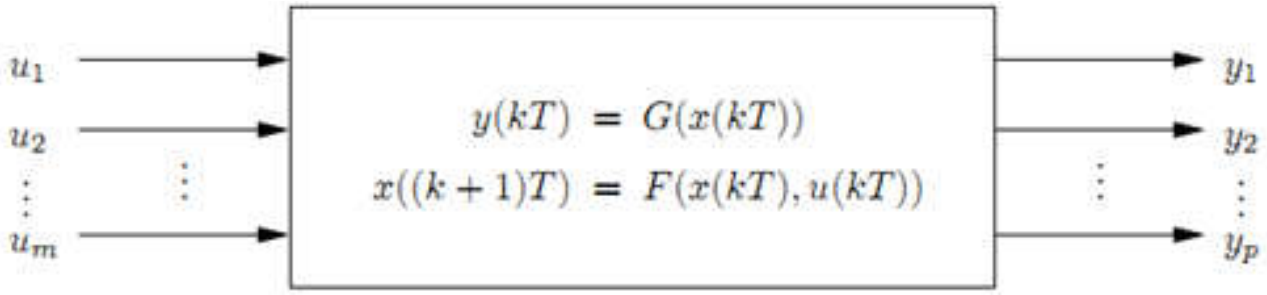
- monolithic approach
  - implementation with a single periodic task.
  - simple，but tends to lead to a larger feedback-delay.
    - In order to diminish this delay a predictive controller（？） may be implemented.
- multiple task approach
  - dividing the process in more than one real-time task.
  - The main task calculates the new control signal.
  - Followed by Task 2 where the new state variable are calculated.



set timer to interrupt periodically with period T;
at each timer interrupt **do**
    do analog-to-digital conversion to get y;
    compute control output u;
    output u and do digital-to-analog conversion;
od

Single task or Two

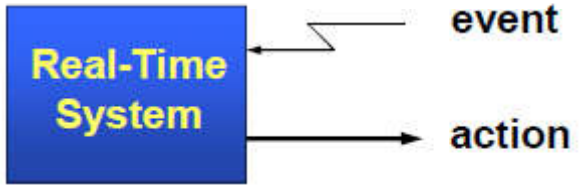




优先级、通信、同步？

Event-action model（paradigm）

- 发生与预定状态相关的事件，则执行预定动作
  - FSM based：“状态”由状态变量的特定值标识
- 事件：在特定时刻瞬间发生，标志系统的行为
  - 外部事件：外部动作产生的事件
  - 内部事件：系统内产生或激活的事件
  - 开始事件和终止事件：标识动作开始和结束的事件
  - primitive events，composite events
  - Mode：事件的集合
- 动作：系统对事件的反应。
  - 简单动作：原子操作
  - 复杂动作：一个动作序列
    - 三种：顺序、选择、并行
  - 可按所需使用的资源划分动作
- 条件（condition）：定义允许的动作
  - 条件通过允许或禁止某些操作来影响系统的行为。
  - 动作执行或发生外部事件可以引起条件变化，条件变化又可引起事件发生。



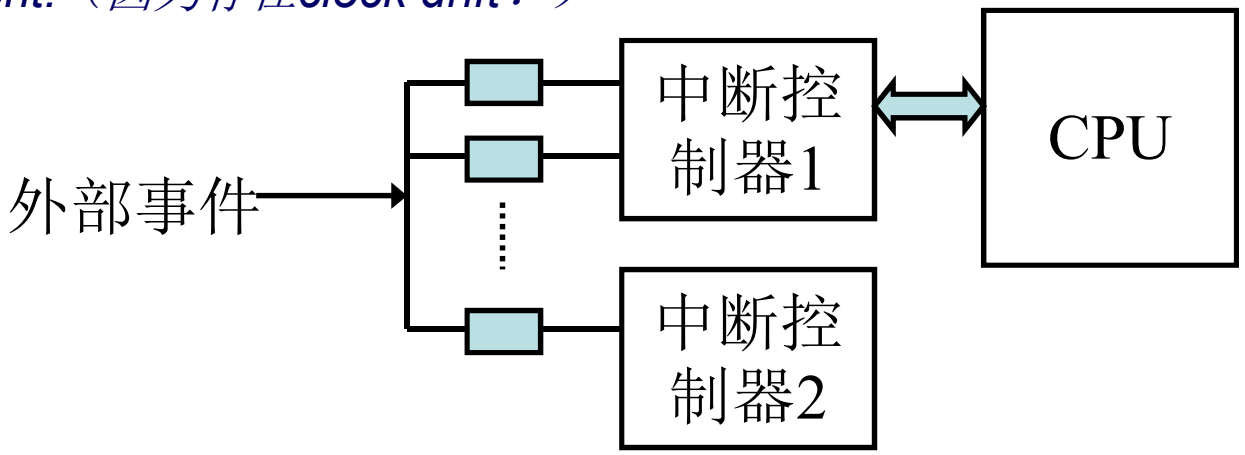
外部事件的到达模式

- 周期性事件
- 非周期性事件的时间特性：

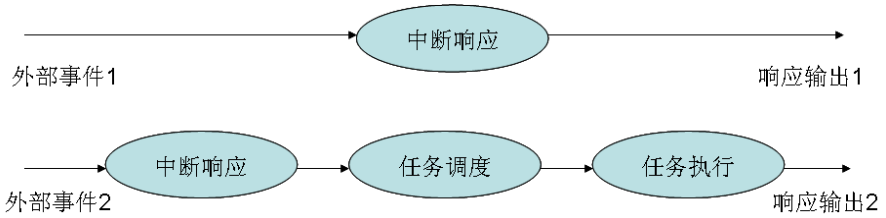
不规则的	事件之间存在某个已知的，但长短可变的时间间隔序列；
突发的	事件的出现序列中，任意两次之间的间隔可能相当近，但是突发情况下事件的数目也不会超过某个已知范围。
有界的	已知最小两次到达间隔（称为界限）的事件序列。
平均速度有界的	事件队列中单个事件的到达事件是不可预测的，但是它们在某个平均值上下波动。
无界的	到达间隔可以用统计原理进行预测的事件序列。它们是一个从某个概率密度函数中提取出来的可重复或者不可重复过程。

同步与异步事件：CPU's perspective

- 同步事件：发生时间可预测，驱动的任务为同步任务
  - Synchronous events are those that occur at predictable times in the flow-of-control.
  - 如：内部中断（internal trap interrupt）
- 异步事件：随机发生的事件，驱动的任务为异步任务
  - Asynchronous events occur at unpredictable points in the flow-of-control and are usually caused by external sources.
  - 如：外部中断
  - *A clock that pulses "regularly" at 5 milliseconds is not a synchronous event.（因为存在clock drift！）*

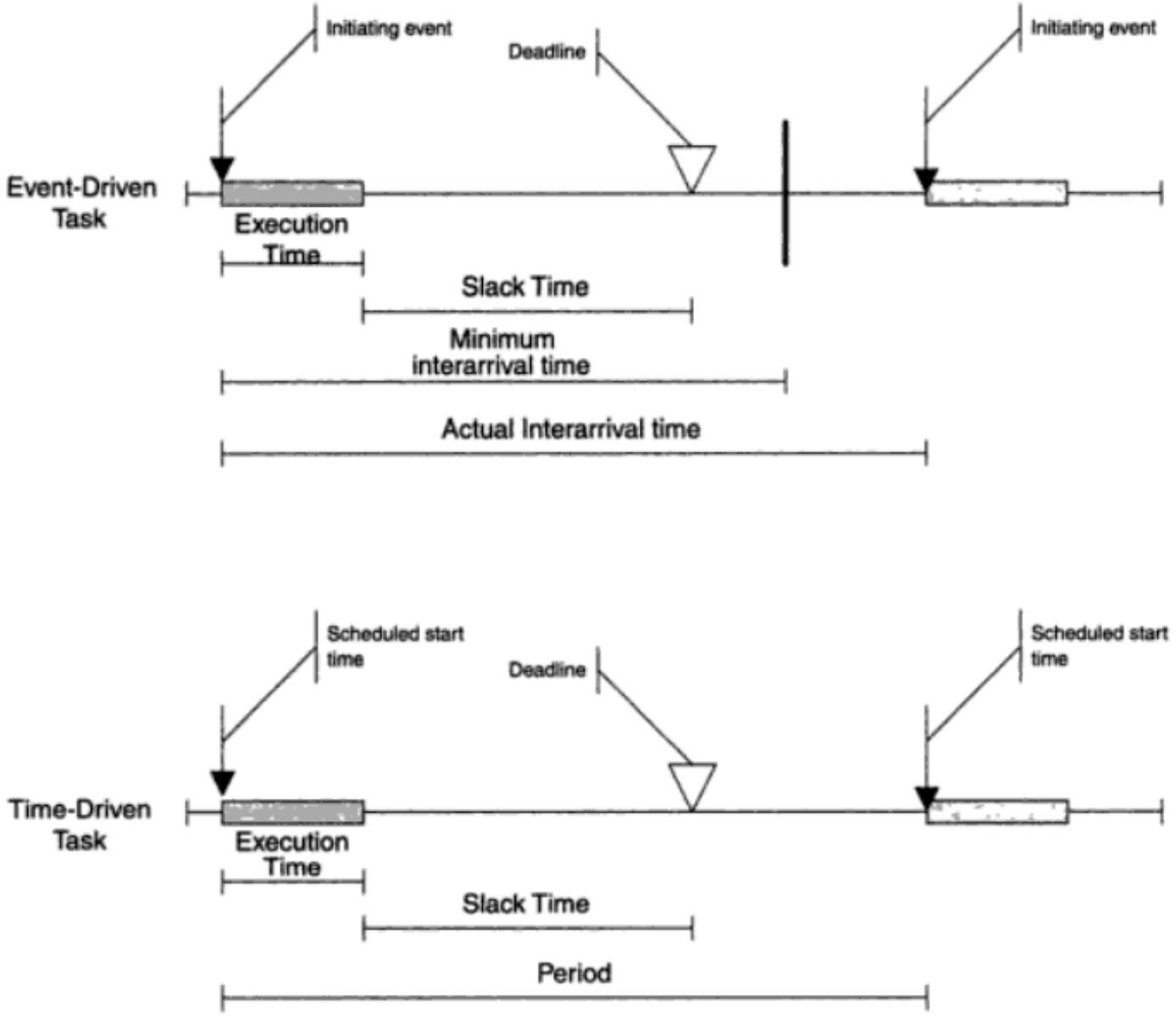


事件响应模式：ET、TT



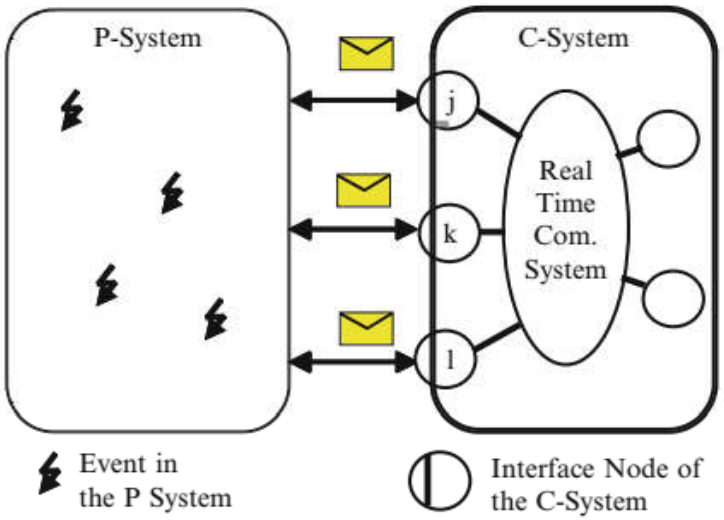
- 事件触发系统ET
  - 中断驱动：依赖于事件的顺序
    - 低负载时（事件少）响应快。
    - 事件风暴（event shower）：大量事件同时到来，系统崩溃
  - 按arrival pattern响应事件，适合采用动态调度
- 时间触发系统Time-triggered
  - 事件是异步的，响应时间是预先确定的(时间间隔不一定固定)
    - 将事件缓存，每个时钟tick中断时再进行分析处理
  - 按response time requirements响应事件，适合采用静态调度
- 例：考虑一个100层电梯的控制器设计
  - 当前电梯在60层，有人在一层按钮，100ms后有人在90层按钮
    - 事件触发系统：1层先相应，将100层按钮缓存。
    - 时间触发系统：设500ms一个tick，触发采样。
      - 同时多个事件，则需进行仲裁。可按“最近用户优先”原则。

Execution Schemes for RT Tasks



ET/TT编程框架

- Event-driven
  - Interrupt-driven
  - ET schemes
- Clock-driven
  - TT schemes




```
< Initialize Memory >
foreach input_event do
    < Compute Outputs >
    < Update Memory >
end
```

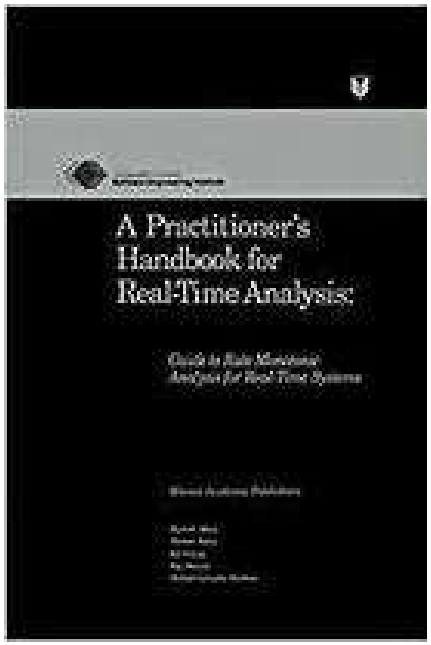
a. "Event driven"

```
< Initialize Memory >
foreach period do
    < Read Inputs >
    < Compute Outputs >
    < Update Memory >
end
```

b. "Sampling"

事件-动作范式：设计分析方法

- 框架、场景（组）、表
  - 基于场景：场景表->实现表->技术表->分析
    - offer a set of implementations for each situation
    - 资源调度策略：fixed priority, preemptive scheduling
    - 评估事件响应的可靠度性（满足dl？）

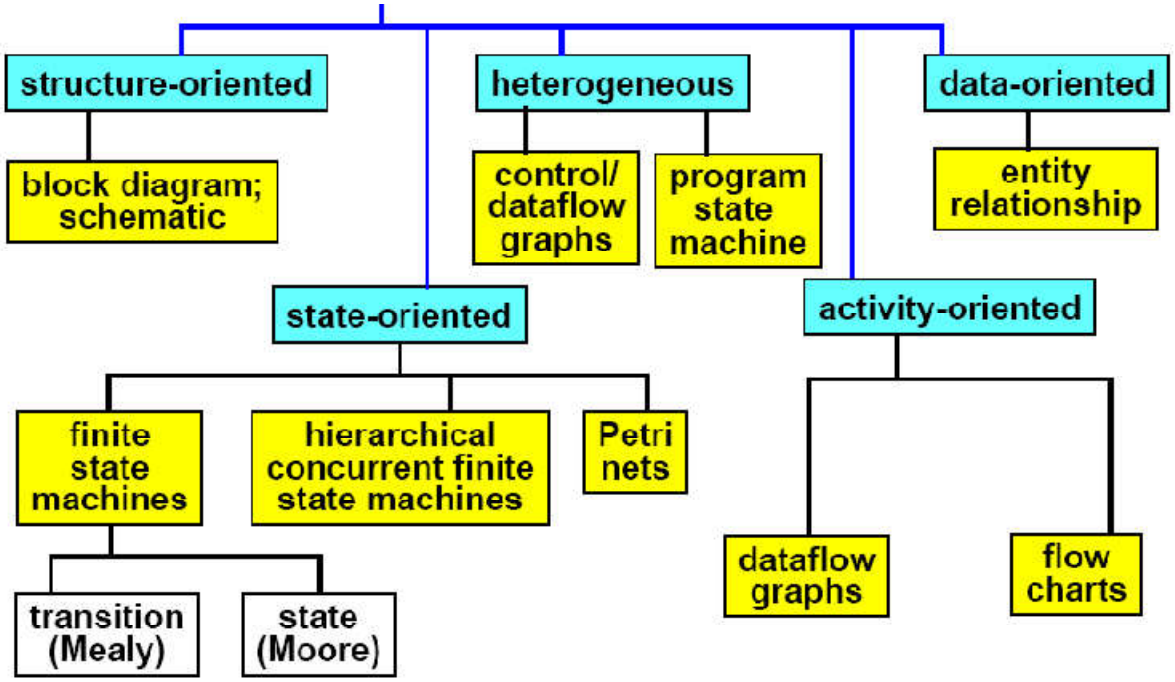


实时系统分析方法

- 需求分析：根据需求，确定系统目标及其约束
  - 对软硬件做合理的分解
    - Miller法则：一个系统应该分解成7±2个子系统
- 系统分析：对已有系统的行为与性能进行分析验证
  - 数学方法（重点：系统性能分析）
    - 为实时系统建模，并估计时间与所需资源大小
    - 如Thomas McCabe提出的数学方法
      - 使分析者能够为实时系统的硬件和软件组件建模
      - 用概率的方法表示控制，采用网络分析、排队论、图论、Markov数学模型等得出系统时序和资源大小
  - 建模和仿真方法（重点：系统结构与行为）
    - 建立各种形式化或非形式化模型
    - 不仅可以分析系统的性能，还能建立一个可执行的原型，从而了解系统的动态行为。

实时系统设计方法

- 软件设计阶段：在给定的约束条件下完成系统的目标
  - 重点：实时软件的设计，描述系统行为。
- 设计方法
  - 基于面向对象、面向数据流和面向数据的方法，扩充实时能力
  - 基于FSM、Petri网、消息传递机制、特殊的实时语言等



实时系统分析设计方法

- 分析方法，关注需求分析阶段
- 设计方法，关注系统设计阶段
- 对于实时系统，常常合而为一，分为：
  - 结构化的方法
    - DARTS
  - 面向对象的方法：实现信息隐藏
    - UML
  - 基于组件的方法：强调组件复用
- 关键
  - 任务定义：划分、通信与同步

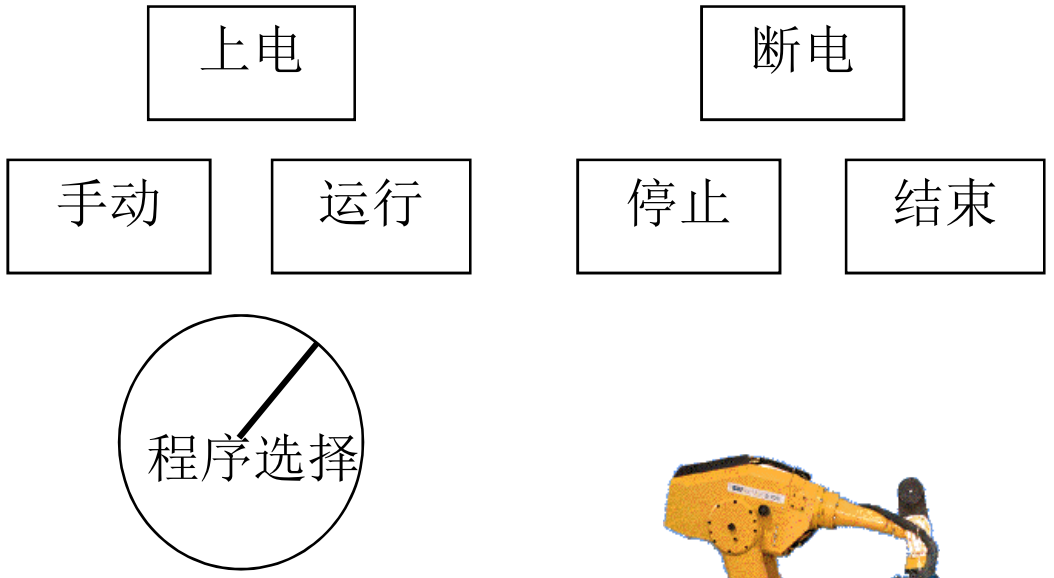
结构化分析设计方法

- JSD(Jackscn System Development)
- SCR(Software Cost Reduction)
- RTSAD(Real-Time Structured Analysis and Design)
  - SASD+RT： FSM+DFG
- DARTS(Design Approach for Real-Time Systems)：实时系统结构设计方法
  - “在数据流图上执行状态转换图的一个转换”
- Gomaa对DARTS进行扩充
  - ADARTS：支持基于Ada的设计
  - CODARTS(Concurrent DARTS)
  - COMET/RTE

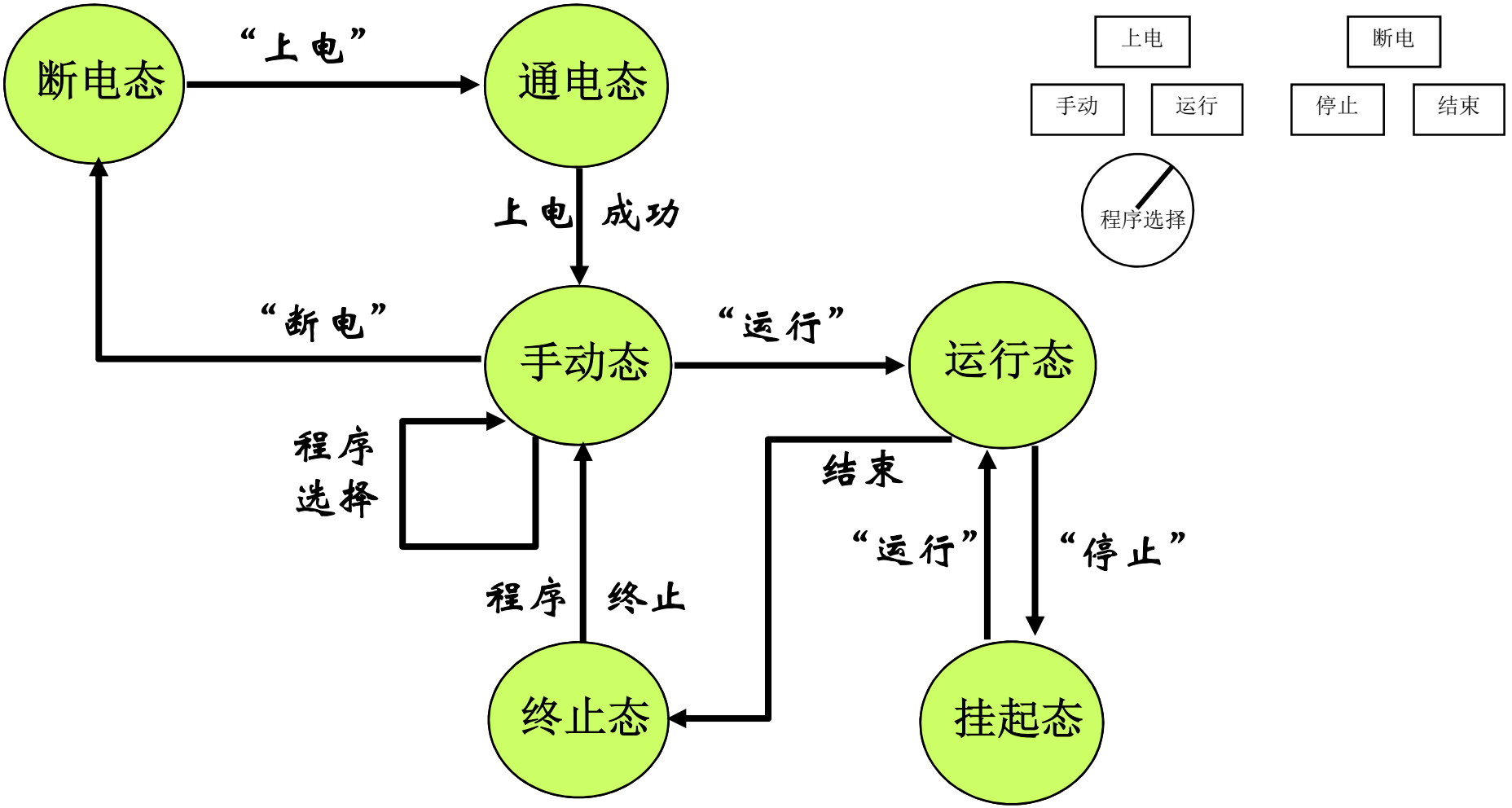
实 例 说 明——机器人控制器系统

- 由内部控制器和外部控制面板组成
  - 控制器控制六个转轴，并与数字I/O传感器交互作用。
  - 转轴和I/O由程序控制
  - 程序由控制面板操作启动执行

控制面板



需求分析与说明

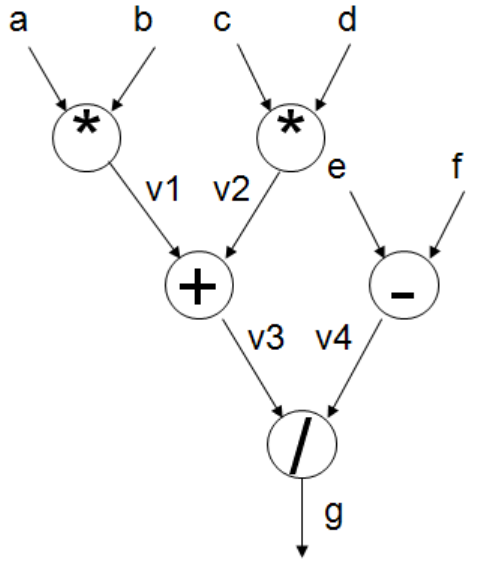


状态变迁图

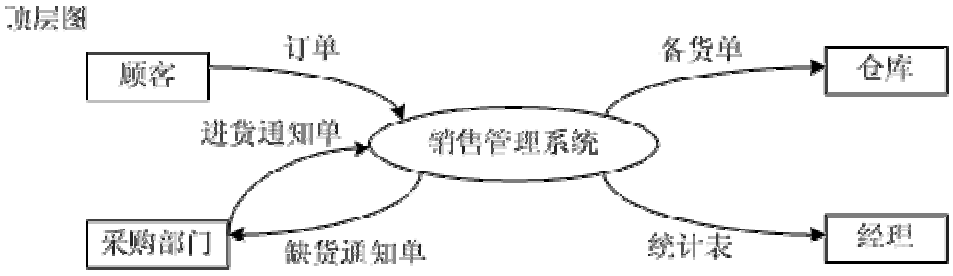
控制执行过程

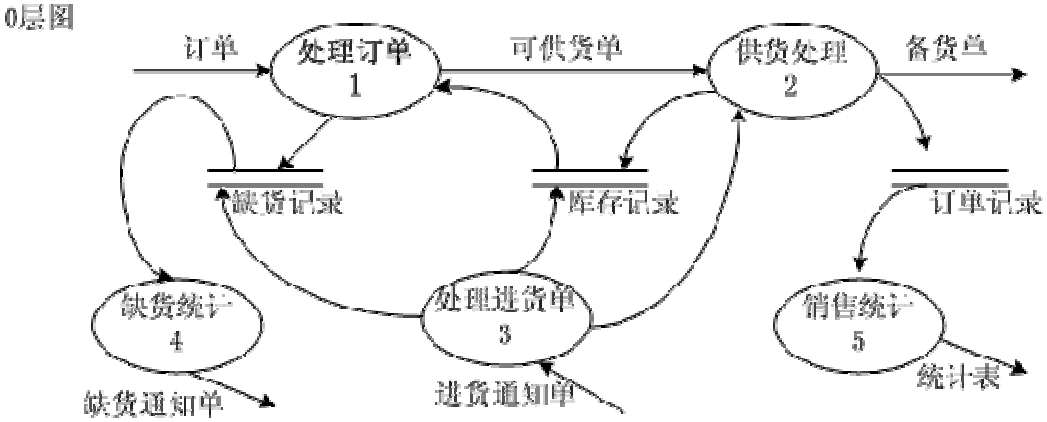
- 按下“上电”按钮，系统进入了上电状态。
- 上电成功后，系统进入了“手动”状态。此时，操作者可以通过程序选择开关选择程序
- 按下“运行”按钮，则选定的程序开始运行，系统转为运行态。
- 程序运行中如果按下“停止”键，程序被挂起。之后，操作者可以按下“运行”键，使程序恢复执行，也可按下“结束”键，结束程序。
- 按下“结束”键后，系统进入终止态。当程序最终终止执行时，系统返回手动状态。

数据流图

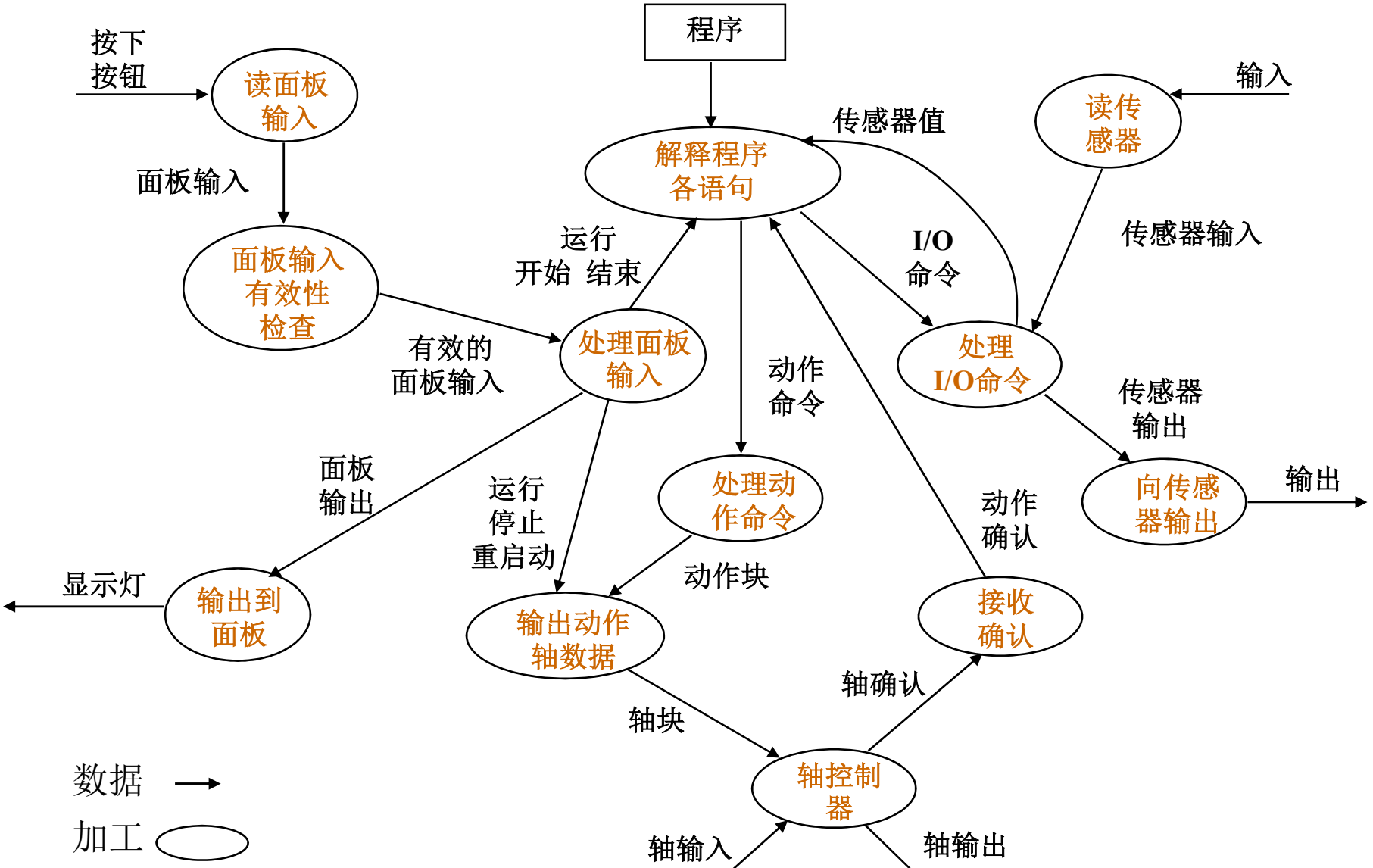


- 每个数据流图都包含：
  - 加工/变换，椭圆
    - 对数据的操作变换
  - 数据流，箭头
    - 变换间的数据流动
  - 数据存储区，方框
    - 数据的存储场所
  - 文件，双线
    - 输入/输出文件
  - 数据字典
    - 定义数据流和数据存储区所包含的数据项
    - 含：数据项，数据结构/数据元素，数据存储，处理
- 数据流图和数据字典共同构成系统的逻辑模型



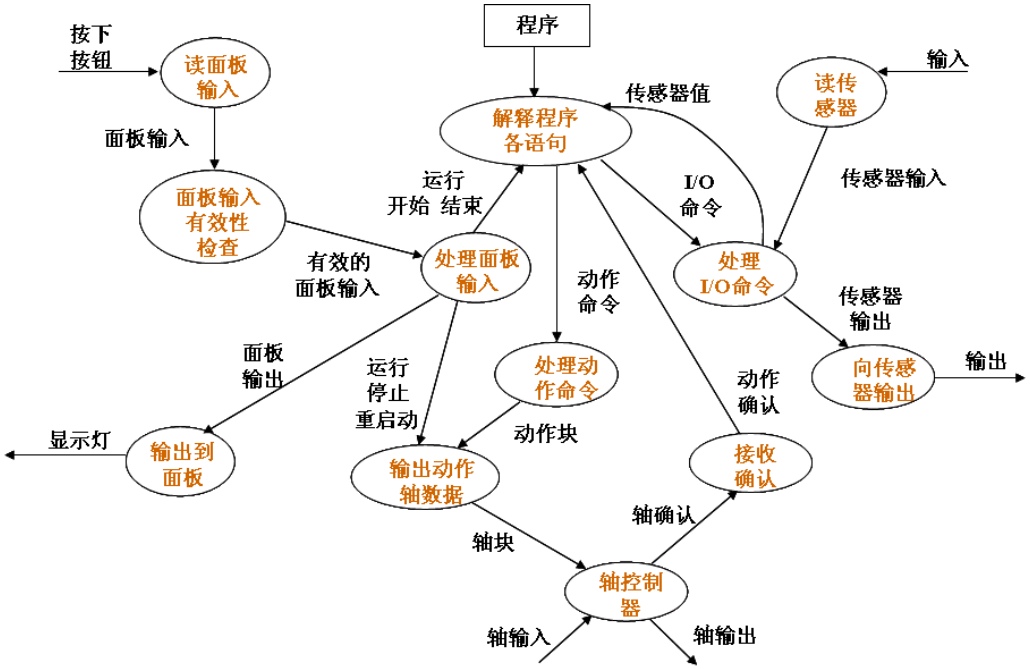


机器人控制器数据流图


程序
按下
按钮
读面板
输入
面板输入
面板输入
有效性
检查
解释程序
各语句
传感器值
读传
感器
输入
传感器输入
运行
开始 结束
I/O
命令
处理面板
输入
有效的
面板输入
处理
I/O命令
传感器
输出
向传感
器输出
输出
动作
命令
面板
输出
运行
停止
重启动
处理动
作命令
动作
确认
显示灯
输出到
面板
动作块
输出动作
轴数据
接收
确认
轴确认
轴块
轴控制
器
轴输入
轴输出
数据
加工

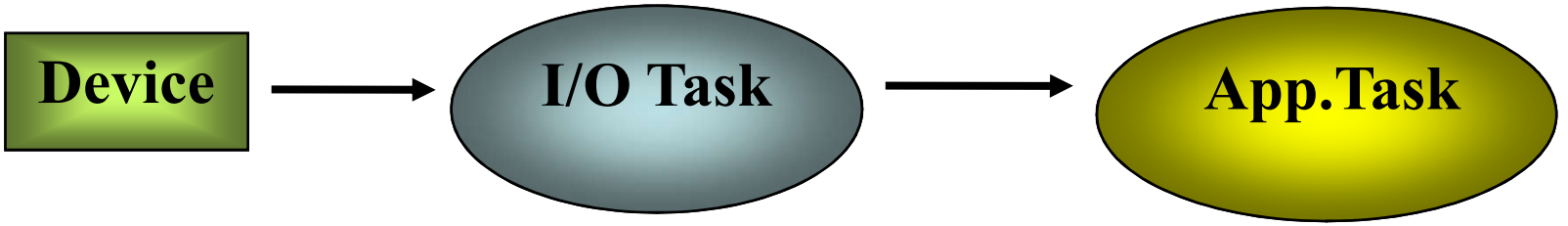
划分任务

- 识别出并行性的功能
  - 变换：分析数据流图中的变换，确定哪些变换可以并行，哪些变换本质上是顺序的。
  - 任务：顺序的执行过程
- 一个任务可对应一个变换，也可对应多个变换。
  - 划分任务的规则
    - I/O 依赖性
    - 功能的时间关键性
    - 计算需求
    - 功能内聚
    - 时间内聚
    - 周期执行



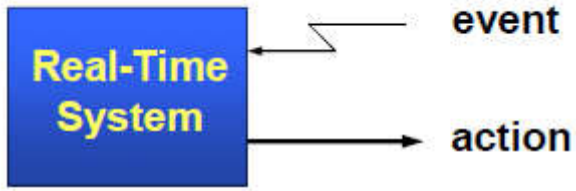
I/O 依赖性

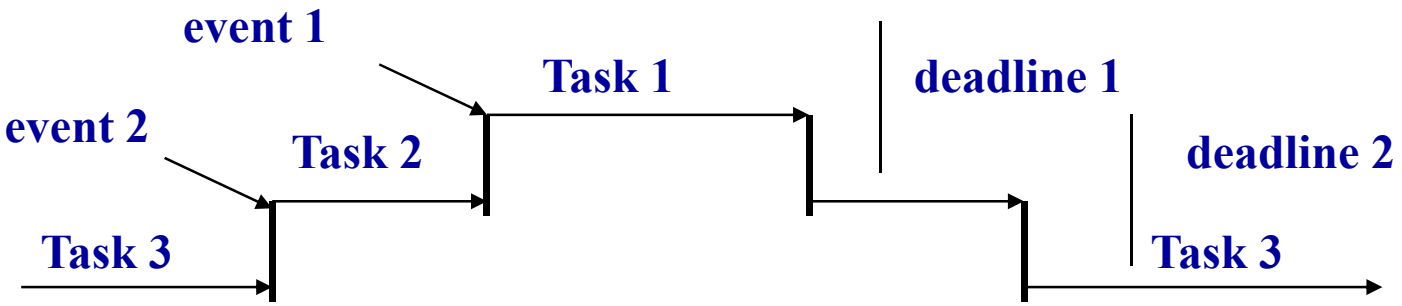
- 如果变换依赖于I/O，性能受限I/O，可独立成任务
  - 在系统中创建与I/O设备数目相当的I/O任务
    - 在任务中分离设备相关性
  - I/O任务只完成与设备相关的功能
  - I/O任务的执行只受限于I/O设备的性能，而不是处理器



功能的时间关键性

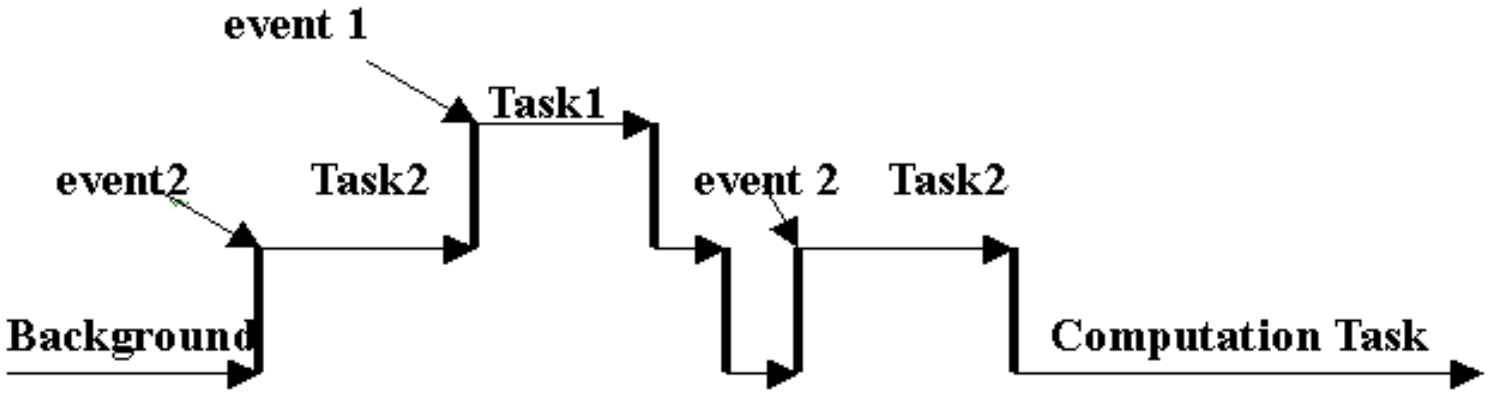
- 将有时间关键性（deadline）的功能分离出来，组成独立运行的任务
- 分别赋予各任务相应的优先级，以满足时间需要





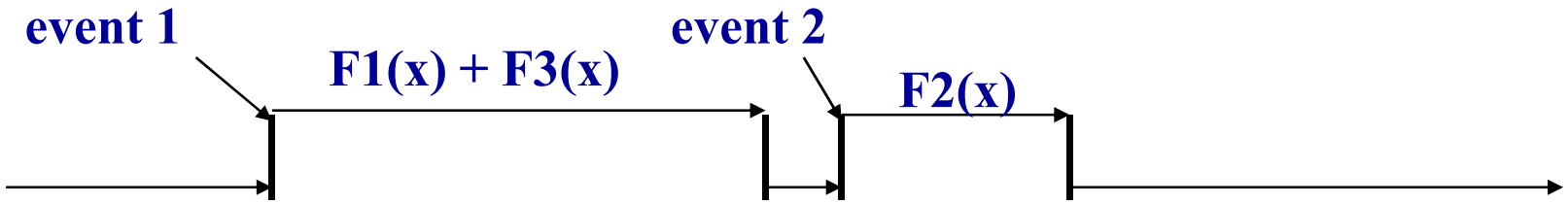
计算需求

- 把计算功能捆绑成任务，以消耗CPU的剩余时间
  - 计算量大的功能占用CPU的时间多，
- 赋予计算量大的任务较低优先级
  - 能被高优先级的任务抢占
  - 保持高优先级的任务是轻量级的
- 多个计算任务可安排成同优先级，按时间片循环轮转



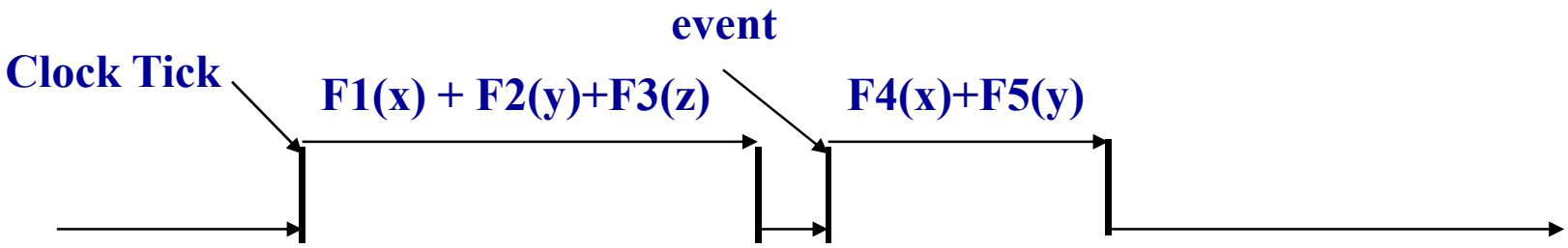
功能内聚

- 将紧密相关的功能变换组成一个任务，减少通信的开销
- 把每个变换都作为同一任务中的一个独立模块，不仅保证了模块级的功能内聚，也保证了任务级的功能内聚



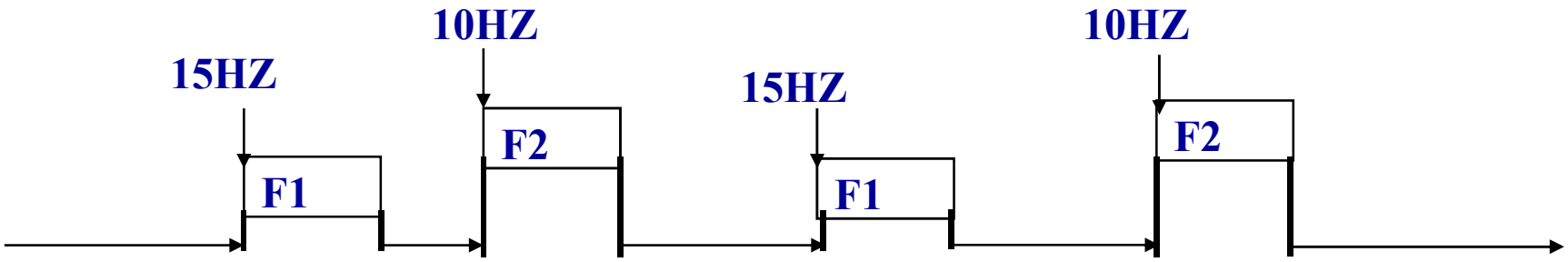
时间内聚

- 将在同一时间内完成的各功能形成一个任务
  - 即使这些功能是不相关的
  - 功能组的各功能是由相同的外部事件驱动的，这样每次任务接收到一个事件，它们都可以同时执行
- 减少任务调度及切换的次数，减少系统的开销



周期执行

- 一个需要周期执行的变换可以作为一个独立的任务，按一定的时间间隔被激活
  - 将在相同周期内执行的各功能组成一个任务
  - 频率高的任务赋予高优先级



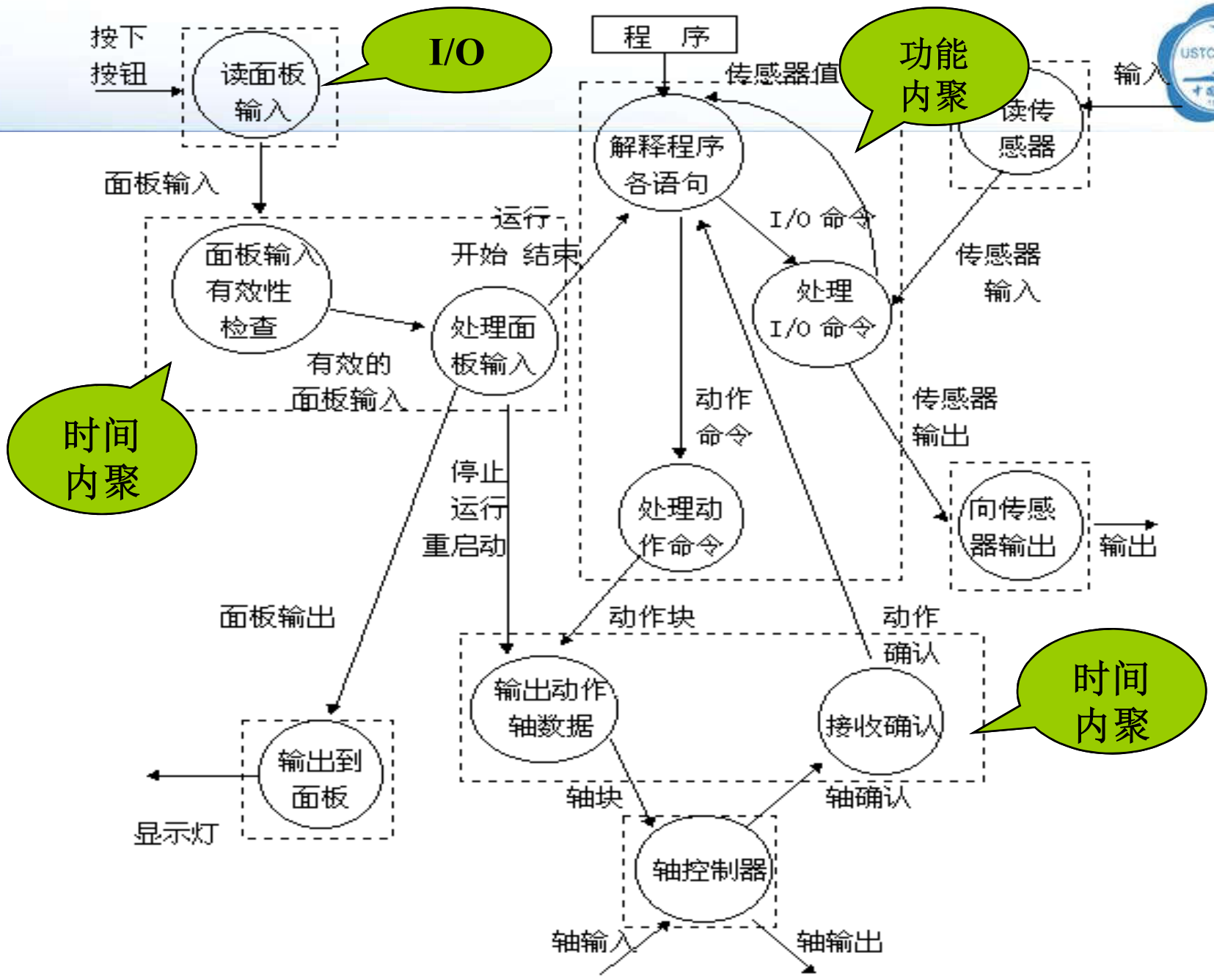
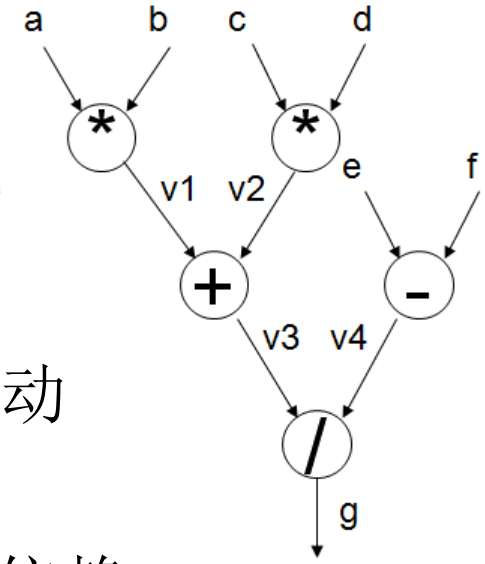


图 2-7　机器人控制器的任务划分

DARTS设计方法



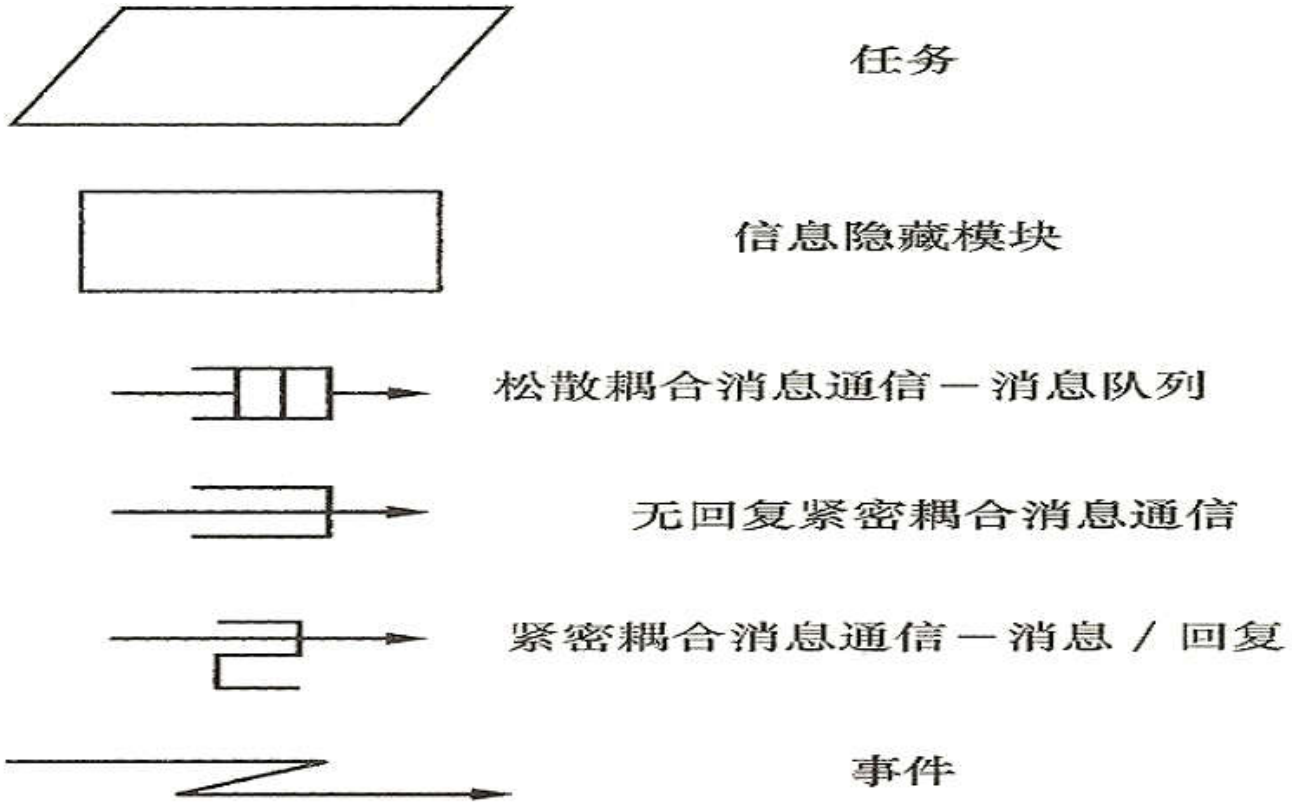
- 扩充DFG
  - DFG强调数据流动，不强调控制信号流动
    - 有利于描述并发，但难于描述同步
  - 扩充：表示任务通信与同步，表示状态依赖
- 定义任务接口
  - 任务间通信模块TCM（Task Communication Module）
    - 消息通信模块：利用同步通信原语和消息队列分别实现
      - 紧耦合通信：发送方发送消息后立即等待回答
      - 松耦合通信：通过消息队列缓冲消息
    - 信息隐藏模块：实现数据结构和访问方法
      - 系统由任务（主动对象）和信息隐藏模块（被动对象）构成
  - 任务同步模块TSM（Task Synchronization Module）
    - 采用互斥等待和信号灯交叉激励的方式实现

DARTS的软件设计周期

- 需求分析与SPEC
  - 分析用户需求，定义系统功能需求和I/O接口
    - 数据流分析：采用DFG
- 系统设计：DARTS方法
  - 完成任务分解(并行进程)，定义任务间接口关系
    - 划分任务：在DFG中确定并发任务
    - 定义任务接口
  - DARTS使用RTSA时间规范将时间预算分配到每个任务上。
- 任务设计
  - 按模块方式设计每个任务，定义模块间接口
- 模块构筑
  - 完成每个模块的详细设计、编码和单元测试
- 任务与系统集成
  - 对每个任务的模块进行集成和测试，然后依次对系统中的任务逐步进行集成和测试。
- 系统测试

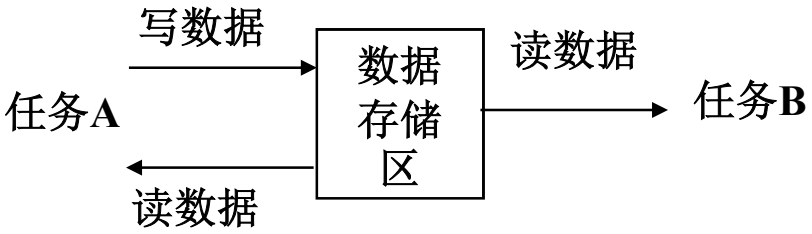
DARTS图例

- 使用**任务结构图**，描述系统分解为并发任务的过程。

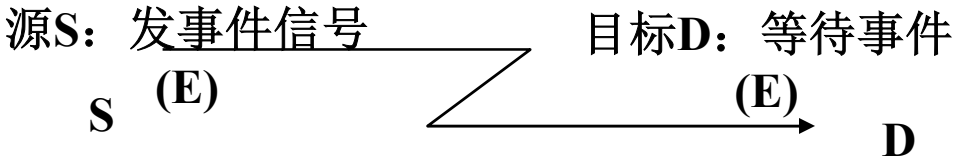


任务接口

- 任务间通信模块
  - 消息通信模块
  - 信息隐藏模块
- 任务同步模块



信息隐藏模块



任务同步模块

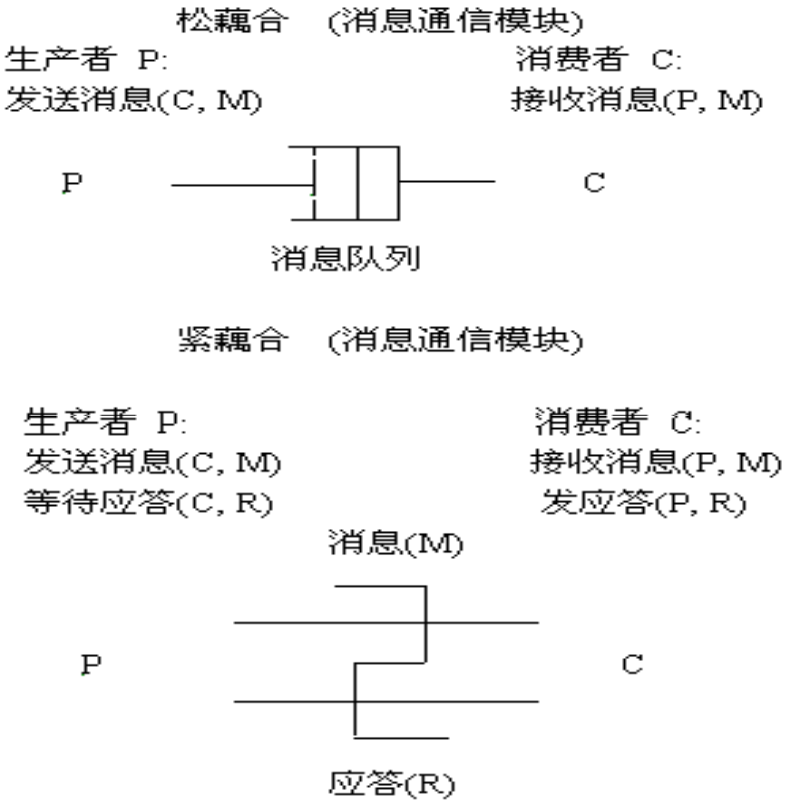


图 2-4 消息通信

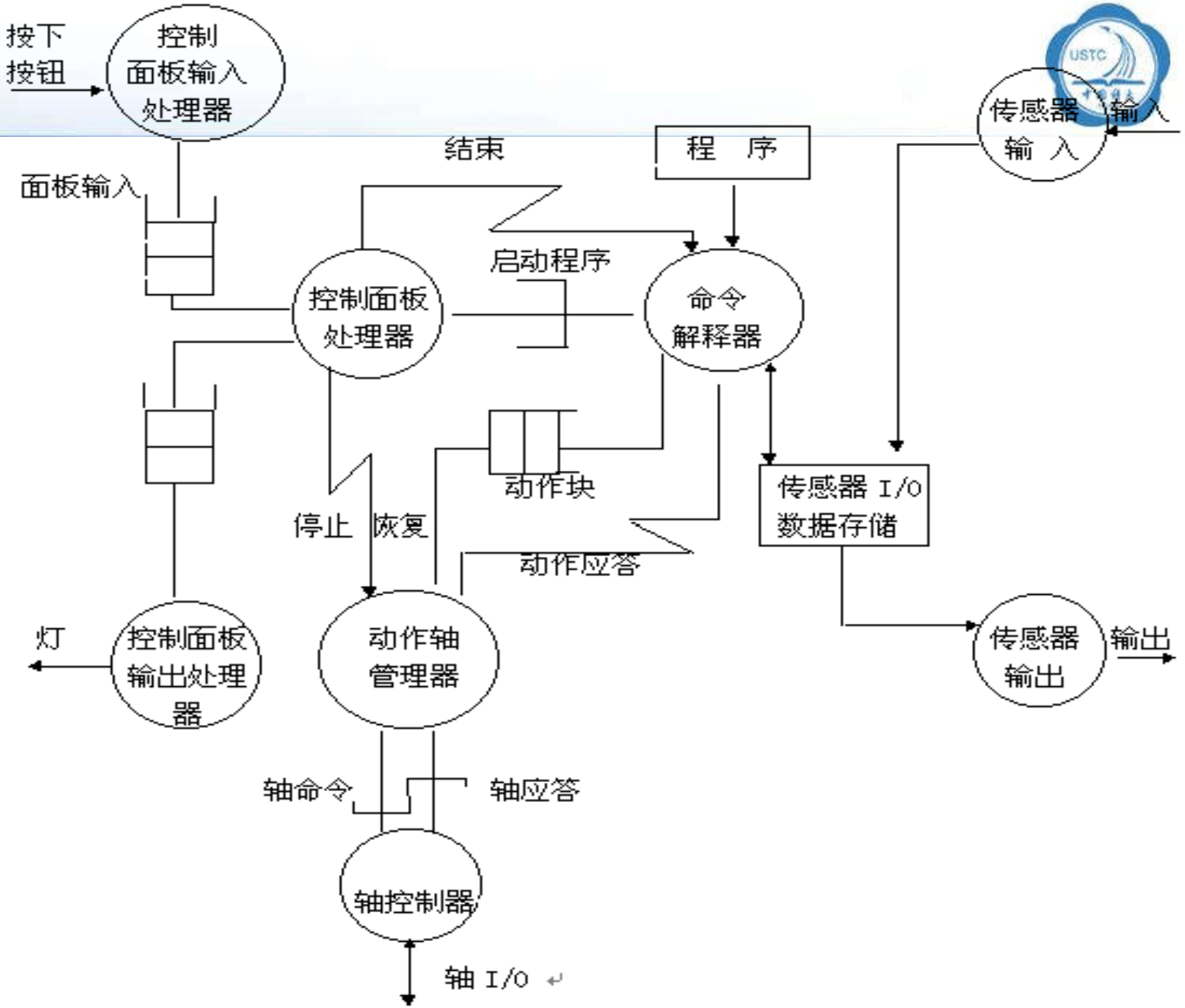


图 2-8　机器人控制器的任务结构图

任务设计

- 详细说明系统中各任务的设计考虑和执行流程，以利于程序员编制程序。
  - 任务体系结构：详细定义任务包含的子模块和模块间的关系
  - 任务执行流程：尽可能详细地描述任务的处理过程
  - 任务内数据结构
  - 任务内模块间接口

任务设计

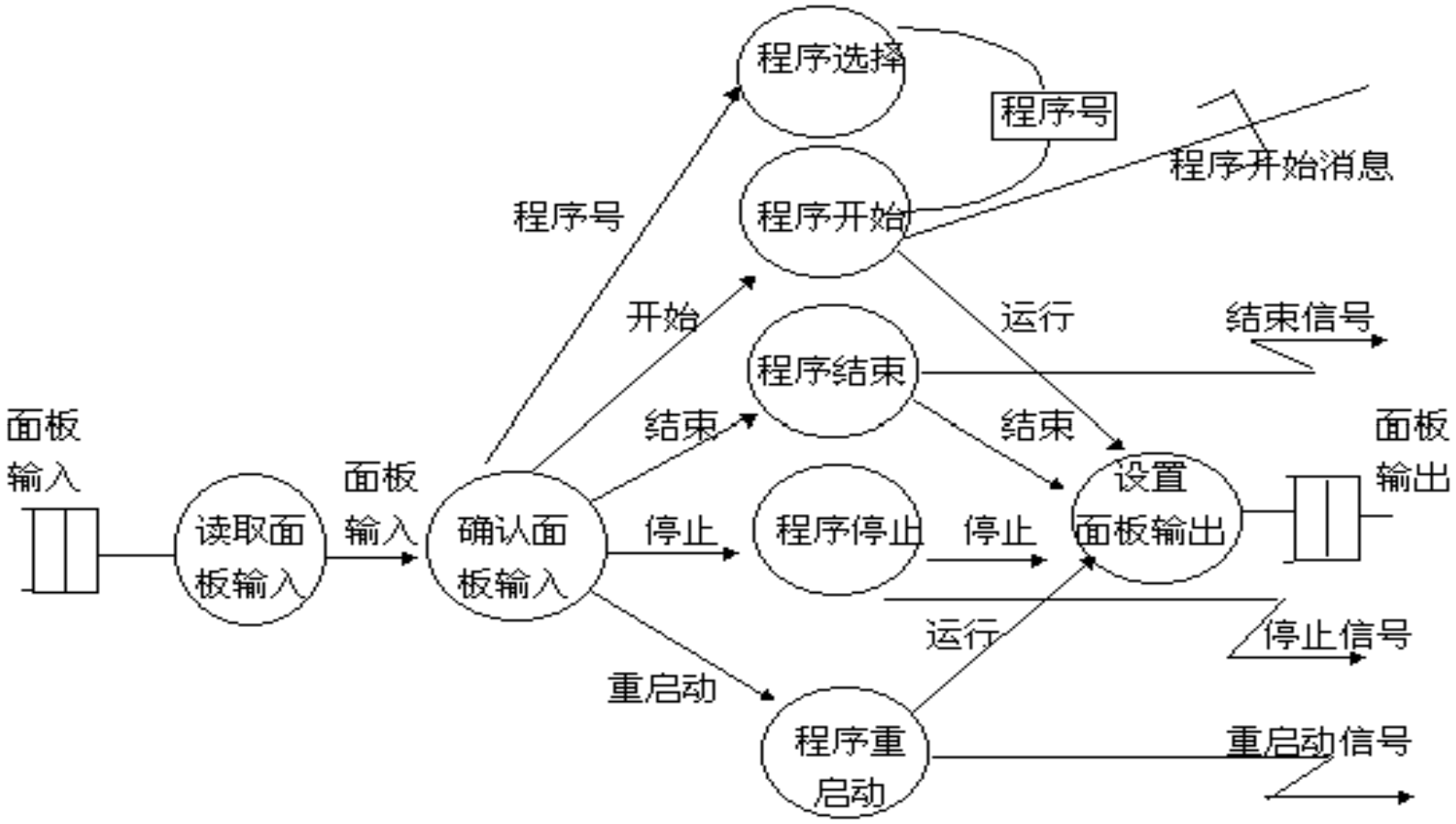
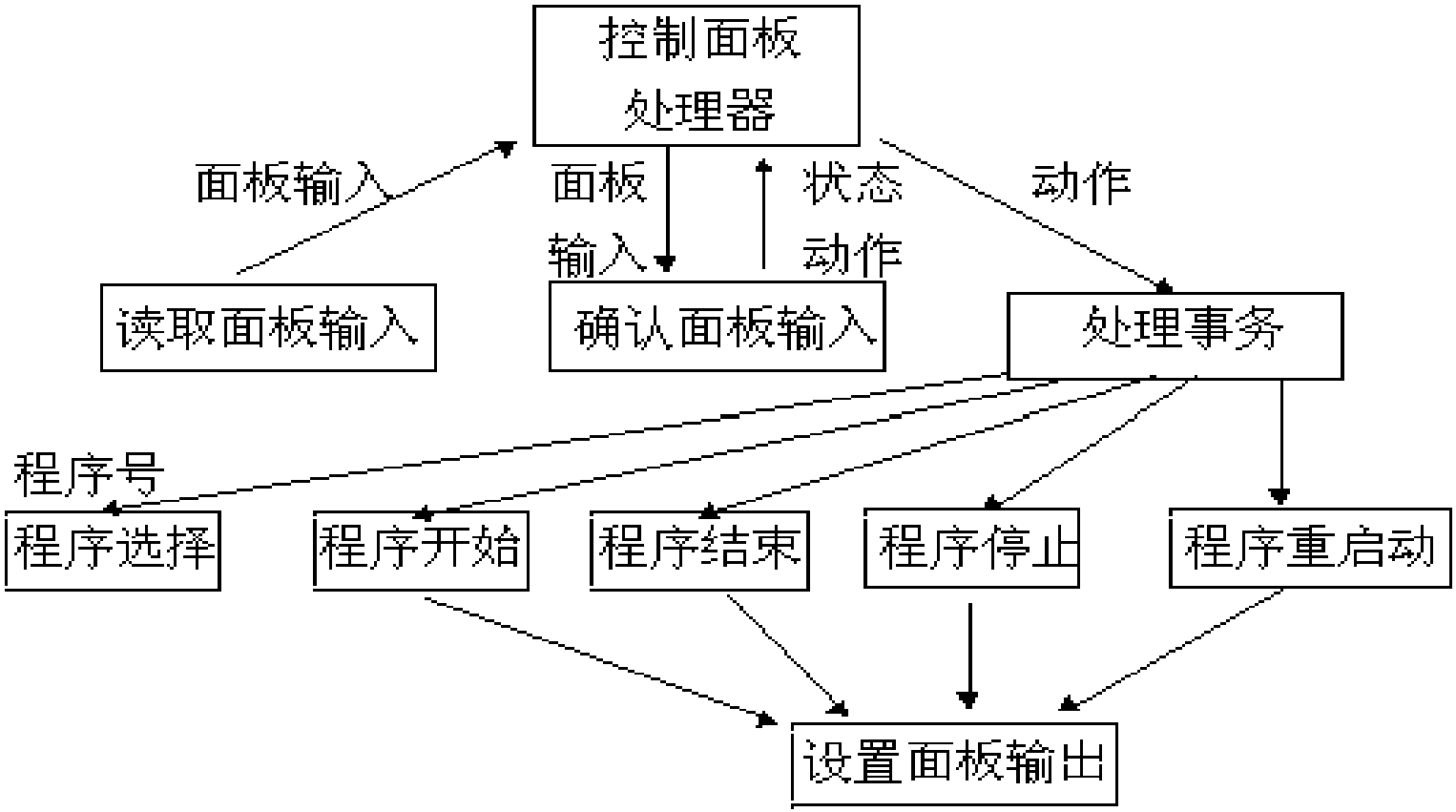


图 2-9　控制面板处理器任务数据流图

任务设计



控制面板处理器任务的模块结构

模块构筑

- 进行每个模块的详细设计，直到每个具体的函数
- 函数设计：含11项
  - 函数描述　给出对该函数的简要描述，说明设计目的、意义以及特点
  - 功能　说明该函数应具有的功能，可采用**IPO**图（输入—处理—输出图）形式
  - 性能　说明对该函数的性能要求，包括精度、灵活性和时间特性等
  - 输入　定义每个输入项的特性，包括名称、标识、数据类型和格式、取值范围、输入方式、数据来源、保密方式等
  - 输出　定义每个输入项的特性，特征同输入

函数设计（续）

- 算法　详细说明本函数所选用的算法，具体的计算公式和计算步骤
- 流程　用流程图辅以必要的说明来表示本函数的逻辑流程
- 接口　说明本函数与其他函数的调用关系，包括说明参数赋值和调用方式以及相关数据结构（如数据库、文件）。
- 存储分配　说明本函数的存储分配
- 限制条件　说明本函数运行所受限制
- 测试计划　说明对本函数的测试计划，包括技术要求、输入数据、预期结果、人员安排等

任务与系统集成

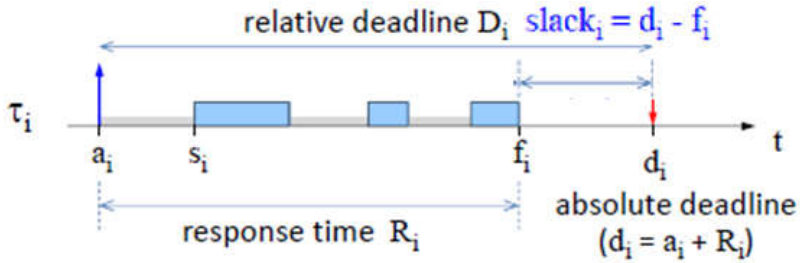
- 模块逐个连接、测试以构成任务
- 任务被逐个连接和测试形成最终系统
- 可分两步集成
  - 在宿主机上模拟集成（软集成）
  - 在目标机上集成

CODARTS

- 结合了COBRA(Concurrent Object Based Real-time Analysis)和DARTS的优点
  - 采用COBRA方法对系统进行分析建模
    - 将系统分解为多个子系统，并将子系统定义为一组由若干对象和功能支持的服务。
  - 在系统划分完成后，使用DARTS方法将任务结构化
    - 确定并发任务，定义任务间接口，使用事件序列图对整个系统建模。
- 支持设计方案的性能分析和软件增量式开发

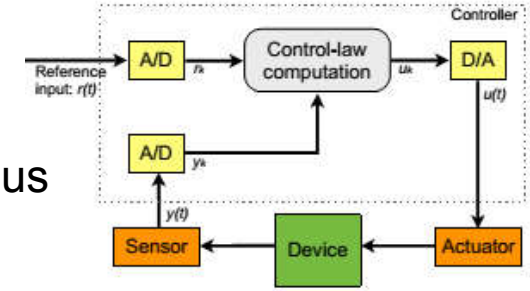
Temporal Constraints编程

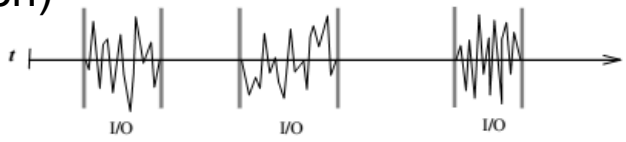
- Types of Temporal Constraints
  - Logical time：impl-independent logical sync & qualitative timing
  - Physical time：impl-dependent physical time consumption
    - Absolute temporal constraints
      - Measured with respect to a global clock：2018.9.18，8:00
    - Relative temporal constraints
      - Measured with respect to a local clock：1ms（real time）
- Type of Specification Temporal constraints
  - Program-based temporal constraints
    - Programmed in the target language(Implementation language)
  - Specification-based temporal constraints
    - Specified in a separate language(Coordination language)
- Granularity of Temporal constraints
  - Statement level
  - Task level

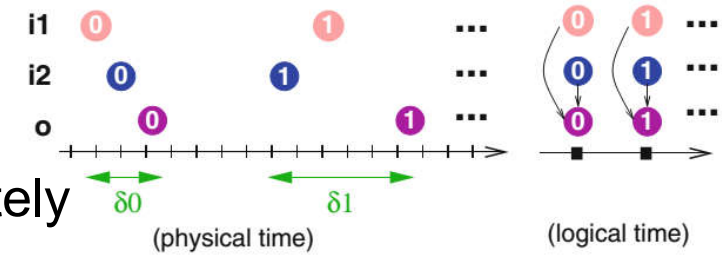


Programming style used in practice

- Synchronous approach：控制工程师采用
  - Berry's synchrony hypothesis
    - Activities take no time，or atomic，or instantaneous
      - an infinitely fast machine
  - Concurrency：logic
    - (asynchronous implementation models also have)
  - Determinism (not all)
    - outputs happen simultaneously and synchronously with inputs
  - Temporal primitive executed accurately(?)
    - Real-Time（环境） is not Logical Time（系统）
  - Easy to do verification test
    - suitable for RT-Analysis (e.g., rms or time-driven)
  - Distribution：异步性，时钟难于同步
- Asynchronous approach：计算机专家采用
  - 异步假设：“同时发生是不可能的”??
  - 基于信号量的Hoare-style通信方法
  - Reaction compete with each other
  - Temporal primitives executed inaccurately



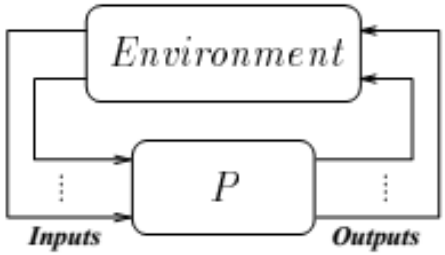




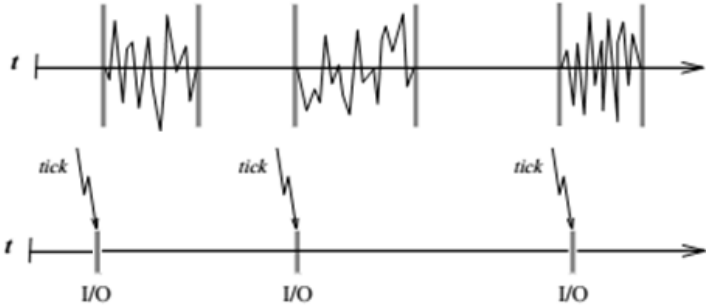
programming language

- RT Concurrent programming language
  - Ada [1983]
    - Based on CFSM(sync message passing) and processes
    - Nondeterministic：未指定进程执行顺序
    - Time-handling primitive is somewhat vague
    - Execution time is unpredictable
  - PEARL[1978]：一种工业过程控制语言
- Synchronous programming language
  - Esterel [1988]
    - Control-oriented language：A imperative language
    - SyncCharts [1996], a graphical version of Esterel
  - Signal [1991] and Lustre [1991]
    - Signal-processing language：Data-flow + time

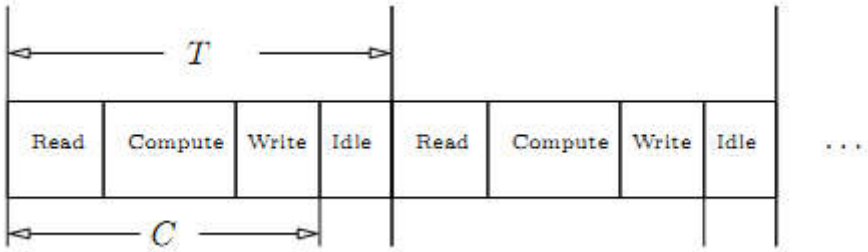
同步语言Synchronous reactive formalisms



- 基于一种抽象的（逻辑的）时间概念
  - 程序的执行过程被划分成无限的顺序的离散的宏反应步
  - 宏反应步由有限个微步组成。
  - 每个微步中，程序读入新的输入，计算相应的输出。
- 之所以称为“同步”
  - 计算所有的输出在一步中以零时间开销瞬间完成；
  - 通过语言的语义，所有并行组件的反应步同步。
    - a *single global clock*
    - requires a fast broadcast mechanism



- 因此，自动机的同步组合是确定的
  - 基于Mealy机
  - 利于系统设计的可预测性、验证和代码生成。
- 提供了RTE系统的高层描述
  - modeling & programming lang
  - 满足MBD设计流程的相关需求。



ESTEREL

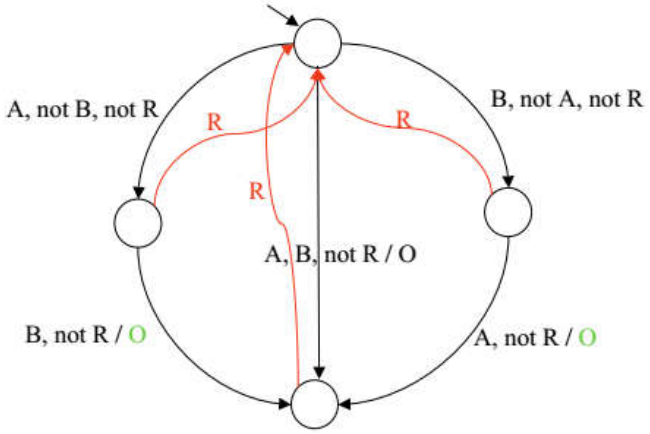
- It was originally designed as a language for programming robot car controllers in the 80's.
  - It is one of the first language based on the synchrony hypothesis. (Statecharts is another.)
    - An Esterel program specifies the control flow of a reactive system.
    - It is event-driven based on discrete signals.
  - An Esterel program is typically compiled (flattened) into a (Mealy) FSM.
- Reactions take no time.
  - That is, the reactive machine is infinitely fast.
  - Responses are synchronous (instantaneous) with stimuli.
    - Inputs are signals, which could be generated by interrupts or sampling, and outputs are produced instantaneously w.r.t. to inputs.
- Communication is by broadcast.
- Synchronous languages are deterministic.
- There is no built-in notion of time; a tick is just a type of signal.

ESTEREL：A Simple Example

- “Emit O whenever A and B, reset upon an input R.”

```
input A, B, R;
output O;
loop
        [ await A || await B ];
        emit O;
each R;
```

- An Equivalent FSM



异步实时语言

- 便于将一个工作划分为一系列任务（线程）
- 能在编译时计算出任务的WCET
  - 循环只能采用常量参数的for循环，而不能使用不确定的while循环和递归。
  - 流水线，Cache miss的影响，缺页，DMA周期窃取等突发事件的影响造成无法评估WCET。
- 计时，以便定义延迟、间隔时间、超时等
- 支持周期性事件描述
  - 如Ada：every（25ms）{…}
  - 可由此编译时计算系统负载的可调度性

PEARL：一种工业过程控制语言

- 每次中断*ready*发生时，任务*supplies*被激活
  - WHEN *ready* ACTIVTE *supplies*;
- 可以对某个任务的执行进行预约：
  - AT *12:0:0* ALL *1 SEC* UNTIL *12:15:0* ACTIVATE *measuring*;
  - AFTER *5 SEC* RESUME;
- 同步
  - REQUEST conveyor-belt;
  - RELEASE communication-buffer;

Edward Lee： 实时系统设计中存在的问题

The Real-Time Problem

- Programming languages have no time in their core semantics
- Temporal properties are viewed as “non-functional”
- Precise timing is poorly supported by hardware architectures
- Operating systems provide timed behavior on a best-effort basis (e.g. using priorities).
- Priorities are widely misused in practice

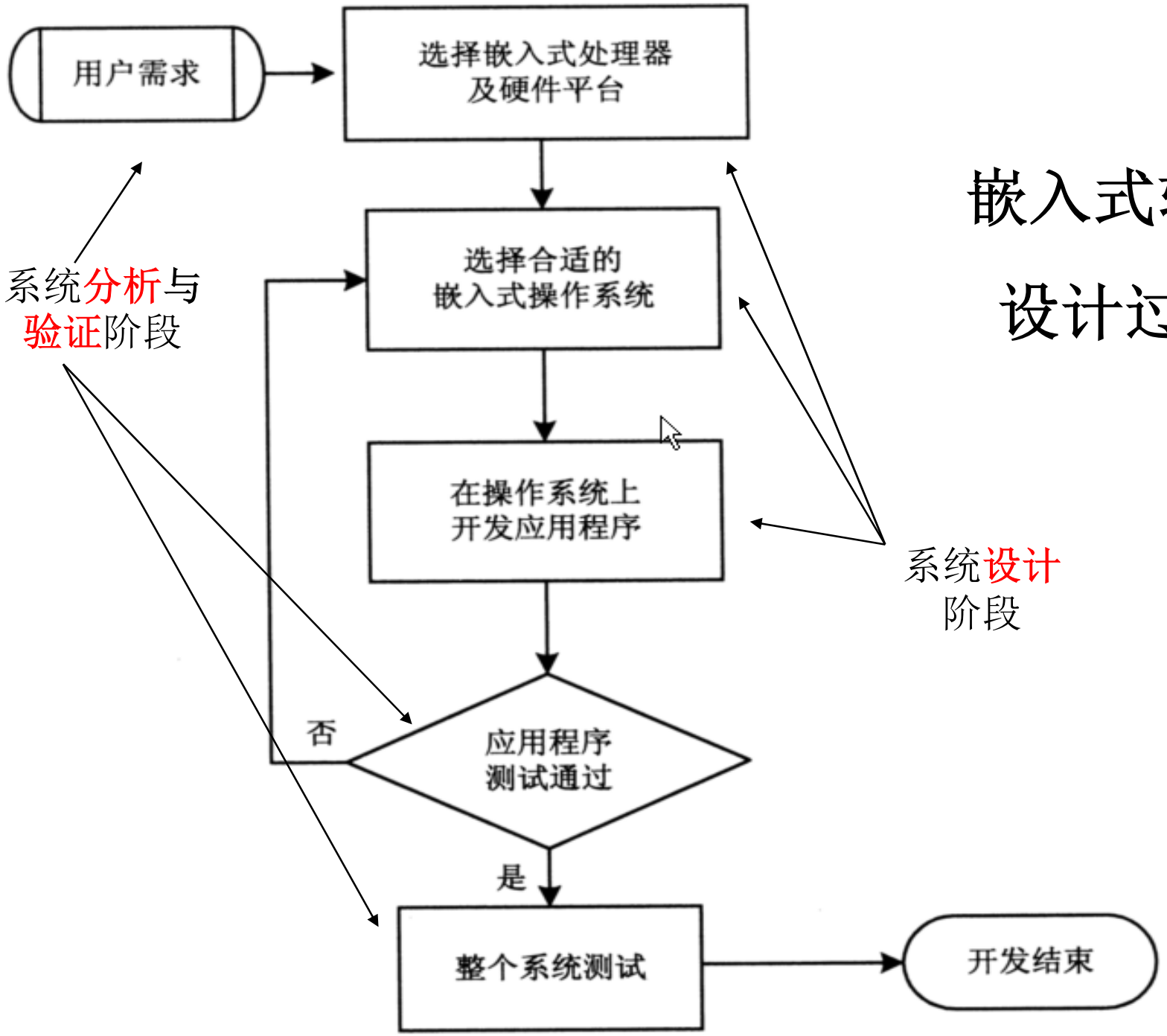
定时规范：Krishna《实时系统》

- 需规定：“没有一个语言做到了！”
  - 两个事件自己的持续时间不超过规定的最大值，且不小于规定的最小值
  - 分配到特定任务的最大执行时间。
    - 如果超过，产生异常，导致产生了一个异步的控制权转移。
      - Overrun Handling
  - 特定任务开始执行的绝对时间。
  - 指定某个任务完成n秒后开始的特定任务。
  - 发出一个消息后多长时间接受到。
  - 接收一个消息后多长时间必须由接收任务处理。
  - 指定一个任务的周期性时序安排
  - 处理每个循环所允许的最长时间
  - 任何动态数据结构大小的上限
    - 因此规定了在过程之间传递它们或分配内存所需的时限。
- Timing requirements can be related to relative or absolute time

Insup Lee2010：**Take Away Messages**

- Timing constraints in programming languages are a topic since at least 1968.
- What are the right abstractions?
  - (modules, tasks, statements)
- What is the right notion of time?
  - (zero, continuous, discrete time)
- Who checks timing constraints?
  - (offline, online)
- How do you specify timing?
  - (specification-based vs. programming)
- How to ensure timing constraints?
  - (verification, runtime checking, offline, online)

嵌入式软件设计过程


用户需求
选择嵌入式处理器及硬件平台
选择合适的嵌入式操作系统
在操作系统上开发应用程序
应用程序测试通过
否
是
整个系统测试
开发结束
系统分析与验证阶段
系统设计阶段

作业（选2）

- 选一：
  - 画出Helicopter Flight Control示例的任务图
  - 采用EA或DARTS方法定义Helicopter Flight Control中的多任务，并设计调度方案。
- 选一：
  - 调研一个实时系统的时间约束，并将时间预算分配到每个任务上的示例
  - CPU clock是Physical time还是Logical time？
  - 采用DARTS方法设计交通灯控制系统
  - DARTS方法为何基于数据流模型？
  - 同步语言有哪些特征？
  - Ada语言有哪些功能？
  - EA设计分析工具TimeWiz


Thank You