第 2 章
实时系统与嵌入式系统概述

Robert Oshana

2.1 实时系统

实时系统是指在规定的时间间隔内必须对由外部刺激（包括物理时间的改变）做出快速反应的操作系统。《牛津字典》这样定义实时系统：

实时系统指一切以输出时间为关键的系统。

这通常因为在物理世界输入对应某些运动，并且输出不得不适合相同的运动。为了遵守时限，从输入到输出的延迟时间必须足够小。另一方面可以把实时系统理解为可由任何信息处理活动组成，必须在限定的时间内对外部的输入产生应答的系统。一般来说，实时系统是一个可以持续且及时与周围环境交互的系统（图 2-1）。

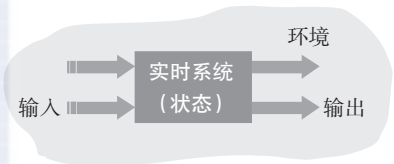

图 2-1 实时系统对来自于环境的输入做出反应并且产生影响环境的输出

2.1.1 软实时和硬实时系统

运算的正确性不仅依赖于它的结果，还依赖于它输出数据的生成时间。实时系统必须满足限定的应答时间，否则系统会遭受严重的后果。如果这个后果包含性能的退化，而不是完全出错，系统则被认为是软实时系统。如果这个后果是系统错误，那么这个系统则被称为硬实时系统（例如，汽车中的防抱死系统）。

当且仅当系统功能（硬件、软件或者二者的结合）对于完成一个动作或者任务具有硬时限，才能认为是实时的。这个时限必须永远被遵守，否则任务会有错误。系统可能包含一个或者更多的硬实时任务，以及其他的非实时任务。这是可被接受的，只要系统可以适当调度这些任务，并且硬实时任务总能遵守时限。硬实时系统多见于嵌入式系统。嵌入式系统是更大系统中的特定部分。在本章的后面部分我们将针对嵌入式系统进行更加具体的研究。

2.1.2 实时系统和分时系统的区别

实时系统和分时系统之间的基本区别在于三点（表 2-1）。

- 系统能力
- 响应能力
- 负载能力

表 2-1 实时系统具有高度的可调度性，必须以高度的资源利用率下满足系统的时序需求必须

特征	分时系统	实时系统
系统能力	高吞吐量	可调度性和系统任务遵循所有时限的能力
响应能力	快速平均响应时间	能够确保的最坏情况下的延迟，即在最坏情况下对事件的响应时间
负荷能力	公平处理所有任务	稳定，当系统超负荷，重要任务必须遵守时限，同时其他任务可能等待

实时系统也要确保最差情况下的时延，保证系统在最差情况下对事件的响应时间。实时系统也提供瞬间超负荷的稳定性——当系统对事件响应已经超负荷，不可能遵守所有的时限，被选择为最高优先级任务的时限仍然必须保证。

2.1.3 DSP 系统是硬实时系统

DSP 系统通常限定为硬实时系统。例如，假设模拟信号被处理为数字信号。需要考虑的第一个问题是，为了在数字域中准确地表征一个模拟信号，应该多久进行一次采样或者如何测量。正如第 1 章所讨论的，采样速率是指一个模拟信号（比如声音）在数字领域每秒采样的次数。现在我们知道信号采样率必须不小于希望保留的最高频率的 2 倍。如果信号在 4kHz 包含重要内容，则采样频率必须至少应为 8kHz。采样周期则等于：

$$T = 1 / 8000 = 125 \text{ ms} = 0.000125 \text{ s}$$

这些告诉我们，这个信号的采样率决定，在下次采样之前，我们将要有 0.000125s 去执行所有必要的处理。当发生连续采样，系统不能在处理过程中落后，并且必须产生正确结果，这就是硬实时系统。

硬实时系统

硬实时任务的集体时效性是二元的，所以它们要么满足它们的时限（在一个正常运行的系统中），要么无法满足（在这种情况下系统是不可行的）。在所有硬实时系统中，集体时效性是决定性因素。这个决定性并不意味着单个任务的实际完成时间，或者任务的执行顺序需要预先知道。

作为硬实时系统的计算机系统，意味着与时限的长短无关，其可能是微秒级的，也可能是几周。“硬件实时”这个词的使用可能有些混乱。有人认为硬实时响应时间低于极低的阀值，如 1ms。这并不正确。许多这样的系统属于软实时。这些系统应该正确称为“实时快速”或“实时可预测”，但实际并非硬实时。

硬实时计算的可行性和成本（例如系统资源方面来）依赖于已知，首先是与任务和执行环境有关的预期行为特征。这些任务特征包括：

- 时限参数，例如到达周期或者最大值。
- 截止时间
- 最差执行时间

- 就绪和挂起时间
- 资源利用属性
- 优先权和排斥约束
- 相对重要性

下面举例执行环境相关的重要特征：

- 系统加载
- 资源相互作用
- 队列规则
- 仲裁机制
- 服务延迟
- 中断优先级和时间
- 缓存

在硬（软）实时计算中，决定集体任务的及时性需要掌握相关任务的预期特征和执行环境，即环境完全可知。这些特征性的常识会被用来预先分配资源，使得所有时限被遵循。

通常，任务的预期特征和执行环境必须调整以确保调度和资源分配满足所有时限。遵循所有时限的不同算法或者调度通过其他因素被评估。在很多实时处理应用中，首要因素是使处理器的利用率最大化。

硬实时计算的分配可以使用多方面的技术。有些技术包含线性列举法查找静态调度，这可以确保满足所有的时限。调度算法包括分配不同系统任务优先级。这些优先级可以被分配，或通过应用程序离线，或通过应用程序或操作系统软件在线。任务优先级分配或是静态（固定），如使用单调率的算法，或者是动态的（可变），如使用最快时限优先算法。

2.1.4　实时事件特征

实时事件包括三个类别：

- 异步事件是完全不可预知的。举个例子，一部手机拨打电话到一个基站。虽然基站是可以预知的，但是拨打电话的动作是无法预知的。
- 同步事件是可以预知事件，并且按照精确的规则出现。比如，摄像音频和视频是同步。
- 等时事件的规律性发生基于给定的时间窗口。比如，在网络多媒体程序中的音频数据，当对应的视频码流到达时，必须在给定的时间窗口出现。等时是异步的一个子类。

在许多实时系统中，任务和预期执行环境特性很难预测。这使得硬实时调度不可行。在硬实时处理过程中，集体事件及时性的标准由实际的需求决定。满足该要求的必要方法是确定性任务和执行环境特征情况的静态（即优先）调度。为了实现离线调度和资源分配，要求预知每个系统任务以及它们的预期执行环境，而这种需求极大地限制了硬实时处理的适用性。

2.2 高效运行和运行环境

实时系统是时间关键的系统，相对于其他系统，系统执行的效率会更为重要。效率可以分别按处理器、内存和电源分类。这些约束可能影响从处理器到编程语言的任何选择。使用高效语言的主要益处之一是可以让程序员剥离实现的细节，专注于解决问题。但是在嵌入式领域中，并不总是如此。有些高级语言所用的一些指令会比汇编语言的命令慢一个数量级。但是，通过正确的技巧，高级语言可以高效地在实时系统当中使用。我们将在第 11 章详细讨论这个话题。

资源管理

只要系统可以让时间关键的进程在可接受的时限内完成，该实时系统就能实时运行。“可接受的时限”被定义为行为模式的一部分或者系统的非功能需求。这些需求必须是客观、可计量的，因此是可测量的（比如系统必须很快，这样的描述是无法量化的）。如果系统包含一些实时资源管理（为了确保系统运行的实时性，必须要显式管理这些资源），那么可以说该系统是实时的。如前所述，资源管理能以静态离线或者动态在线方式完成。

实时资源管理带来一定开销。系统需求的实时处理程度不能仅仅通过硬件的超强功能达到（比如，使用更快的 CPU 实现的高速处理器）。实时系统资源管理也要用来节省开销。必须实时态运行系统由实时资源管理和硬件资源容量构成。带有物理设备的交互系统需要更高程度的实时资源管理。这些计算机是指我们之前所谈的嵌入式系统。许多这类嵌入式计算机使用很小的实时资源管理单元。通常使用的资源管理是静态的，并且需要系统环境优先级的分析先于在其环境当中的执行。在实时系统中，为了实时资源管理，物理时间（相对于逻辑时间）是精确关联的事件及其发生时间的必需。当进程运行到结束时，物理时间对于运行时间的约束和测量实际消耗的时间都非常重要。物理时间还可以用于记录历史数据。

所有实时系统都要在性能与调度开销之间作出取舍，从而在优化调度规则的实时部分与离线调度性能的评估、分析之间达到某种合理的平衡，以获得一种可接受的时限。

反应式和嵌入式实时系统

有两种类型的实时系统：反应式和嵌入式。反应式实时系统涉及与环境长时间连贯交互（比如驾驶员控制飞机）。嵌入式实时系统用于控制安装在某个更大系统内部的专用硬件（比如控制汽车防抱死微处理器）。

2.3 实时系统设计的挑战

实时系统的设计为设计者带来巨大的挑战。其中最重要的挑战来自于这样的事实：实时系统必须与环境进行交互。环境是复杂而多变的，因此这些交互会变得非常复杂。许多实时系统在环境中不只与一个实体交互，每个实体都有不同的特征，要求不同的交互速度。

例如，手机基站必须能够在同一时间处理数以千计的手机用户打来的电话。每个电话可能有不同的处理需求。所有的这些复杂处理必须被管理和协调。

2.3.1　响应时间

实时系统在约定的时间内必须对环境中的外部交互做出相应反应。这些系统必须在约定的时间内产生正确的结果。这意味着响应时间与产生正确的结果一样重要。实时系统必须设计来满足这些响应时间。必须设计硬件和软件来支持这些系统响应时间的需求。为硬件和软件优化系统需求的分区也同样重要。

实时系统必须遵循响应时间的要求。使用硬件和软件部分的组合，工程决定构架，比如系统处理器内部链接、系统链接速度、处理器速度、内存大小和输入输出带宽。需要回答的主要问题包括：

- 架构是否合适？为遵循系统响应时间的要求，系统可以被构架使用一个强大的处理器或几个较小的处理器。应用程序是否可以划分到几个更小的处理器上同时不会导致整个系统出现大的通信瓶颈？如果设计者决定使用一个强大的处理器，该系统能否满足其电源需求？有时一个简单的架构可能是更好的方法——更多复杂性可能导致不必要的，引起响应时间问题的瓶颈。
- 处理元器件是否足够强大？一个利用率高（大于 90%）的处理单元将会导致不可预测的运行时行为。在这个利用水平当中，在系统中低优先级任务可能会死掉。作为一般规则，加载在 90% 的实时系统将花费大约两倍的时间来开发，这是优化周期和系统整合问题决定的。使用率为 95%，由于相同的问题，系统将占用三倍的时间来开发。使用多个处理器会有帮助，但是处理器间通信需要管理。
- 通信速度是否足够快？通信和输入输出是一种在实时嵌入式系统常见的瓶颈。许多响应时间问题不是来自超负荷的处理器，而是出在系统获取和输出数据的延迟上。在其他情况下，超负荷的一个通信端口（大于 75%）会导致不同的系统节点不必要的队列，引起在整个系统消息传递的延迟。
- 是否使用正确的调度系统？在实时系统，处理实时事件的任务必须采取更高的优先级。但你如何调度多个是处理实时事件的系统任务呢？有几种可用的调度方法，另外工程师必须设计调度算法以适应系统优先级，满足所有实时的时限。因为外部事件可能发生在任何时候，调度系统必须能够抢占当前运行的任务，并允许运行更高优先级的任务。调度系统（或实时操作系统）不能引入大量的开销到实时系统中。

2.3.2　从故障中恢复

实时系统与环境的相互作用本质上是不可靠的。因此实时系统必须能够在环境中检测并克服失败。而且，由于实时系统也嵌入到其他系统，并且可能很难触及（如宇宙飞船或卫星），同样必须能够检测并克服内部故障（用户没有容易触碰的“重置”按钮！）。而且，由于环境中的事件是不可预测的，在该环境中，为每一个可能的组合和事件序列进行测试，

这几乎是不可能的。这一实时软件的特征，在某种意义上会有些不确定性。在一些实时系统中，基于不确定性行为环境去预测多条执行路径几乎是不可能的。

由实时系统检测和管理内外故障的例子包括：

- 处理器故障
- 板级故障
- 链接失败
- 外部环境无效行为
- 内部连接失败

2.4 分布式和多处理器构架

实时系统正变得十分复杂，应用程序在分布式多通信系统中的多处理器系统中执行。设计师面临的挑战涉及多处理器系统中的应用划分。这些系统将涉及多个不同节点上的处理。一个节点可能是一个 DSP，另一个可能是一个更通用的处理器。一些甚至可能是专用的硬件处理单元。这将为项目团队带来很多设计挑战。

2.4.1 系统初始化

初始化一个多处理器系统是很复杂的。在大多数多处理器系统中，软件加载文件驻留在通用处理节点。直接连接到通用处理器（例如一个 DSP）的节点先要进行初始化。在这些节点完成加载和初始化后，与其相连的其他节点可能会经历同样的过程，直到系统完成初始化。

2.4.2 处理器接口

当多个处理器必须相互通信时，必须注意在处理器间发送的消息，确保其定义明确且与处理元素一致。消息协议的差别包括字节顺序、字节排序和其他填充规则，这会让系统集成更加复杂，尤其是系统要求向后兼容时。

2.4.3 负载分配

正如前面提到的，多个处理器导致了应用分配的挑战，可能要求开发应用程序以支持在处理单元之间的有效应用分配。应用程序的分配错误可以导致系统瓶颈，使得某些处理单元超负荷运行，而其他处理单元闲置，从而降低了系统整体的功能。应用程序开发人员设计的应用程序在处理单元之间必须进行有效分配。

2.4.4 集中的资源分配和管理

在多个处理单元的系统中，还有一组需要管理的公共资源，包括外设、交叉开关记忆和内存。在某些情况下，操作系统可以提供信号量机制等来管理这些共享资源。在其他情

况下，可能会有专门的硬件来管理资源。不管怎样，在系统中重要的共享资源必须被管理以防止更多的系统瓶颈。

2.5　嵌入式系统

嵌入式系统是一个专门的计算机系统，通常会集成到一个更大的系统中。嵌入式系统由硬件和软件组件组成一个计算引擎，实现某种特定的功能。不像桌面系统用来执行普通功能，嵌入式系统限制在其应用程序中。如前所述，嵌入式系统通常有时间约束，需对环境做出反应。一个嵌入式系统粗略的划分包括：为应用程序提供性能所必需的硬件（和其他系统属性如安全）和在系统中供了大部分功能和灵活性的软件。一个典型的嵌入式系统如图 2-2 所示。

图 2-2　典型嵌入式系统组件

一些典型的嵌入式系统组件包括：

- 处理器核：在嵌入式系统的核心位置。从简单便宜的 8 位微控制器到更复杂的 32 或 64 位处理器都可能成为处理器核。嵌入式系统设计人员必须为应用程序选择最节省成本的设备，也要满足所有的功能性和非功能性（定时）需求。
- 模拟 I/O：D/A 和 A/D 转换器用于从环境中获取数据并把它返回到环境中。嵌入式系统设计人员必须了解来自环境的、需要的数据类型，该数据的精度要求和输入 / 输出数据速率的精度，选择正确的应用程序转换器。外部环境驱动嵌入式系统的反应类型。嵌入式系统的速度至少要能够跟上环境。比如模拟信息，如光或声音的压力、加速度，被检测到并输入到嵌入式系统当中。
- 传感器和执行器：传感器是用来从环境中感应模拟信息。执行器用于以某种方式来控制环境。
- 嵌入式系统也有用户界面：这些界面可以简单如一个闪烁的 LED 灯，可以复杂，如手机或数码相机界面。
- 应用程序特别通路：硬件加速如 ASIC 或 FPGA，用于在有高性能需求的应用程序中加速特定的功能。嵌入式系统设计人员必须能够恰当集中或划分应用程序，使用有效的加速器以获得最佳应用。
- 软件：嵌入式系统发展的一个重要组成部分。在过去的几年中，嵌入式软件的快速的增长已经超出了摩尔定律，大约每 10 个月增加一倍。嵌入式软件通常是以某种方式优化（性能、内存，或电源）。越来越多的嵌入式软件使用高级语言如 C/C++，一些至关重要的代码仍使用汇编语言编写。
- 内存：嵌入式系统的重要组成部分。根据不同的应用程序，嵌入式应用程序可以根据应用程序耗尽 RAM 或 ROM。嵌入式系统还使用了很多易失性和非易失性的存储器，我们将在本章后面更多谈论这个方面。

- 仿真和诊断：许多嵌入式系统中很难看到或拿到。需要有一个办法接入嵌入式系统用来调试它们。诊断端口如 JTAG（Joint Test Action Group，联合测试行动组）用于调试嵌入式系统。芯片仿真用来提供应用程序行为的可视性。这些仿真模块提供复杂的可视功能到运行时的行为和性能，实际上取代了有外部板级诊断能力的逻辑分析功能。

嵌入式系统是响应式系统

典型的嵌入式系统环境通过传感器响应环境，使用执行器（图 2-3）控制环境。这需要嵌入式系统的表现与环境相符。这就是为什么嵌入式系统称为响应式系统。一个响应式系统必须使用硬件和软件的结合，在确定的限制内响应环境中的事件。难题是，这些外部事件可以是周期性和可预测的，也可能是非周期和难以预测。在嵌入式系统中调度处理事件时，周期和非周期事件都必须考虑，最差情况下执行效率的性能必须被保证。这将是一个很大的挑战。

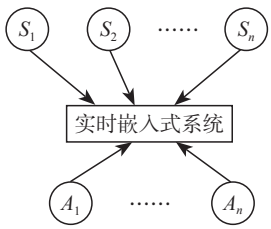

图 2-3　嵌入式系统中的传感器和执行器模型

嵌入式系统的关键特性如下。

- 监测和响应环境：嵌入式系统通常由输入传感器通过读取数据获得输入。有许多不同类型的传感器，检测环境中不同的模拟信号，包括温度、声音和振动。这个数据使用嵌入式系统算法来处理。结果可能以某种格式显示给用户，或仅用来控制执行器（如打开安全气囊并报警）。
- 控制环境：嵌入式系统可以产生和发送命令来控制执行器，如安全气囊，电机等。
- 信息的处理：嵌入式系统以某种有意义的方式处理来自传感器的数据，如数据压缩 / 解压缩、侧面碰撞检测等。
- 应用程序特征：嵌入式系统经常为应用程序设计，如安全气囊、数码相机或手机。嵌入式系统也可能被设计用于处理控制规则、限定状态机和信号处理算法。嵌入式系统还必须能够针对内部计算环境及周边系统检测故障并做出适当的反应。

2.6　总结

许多我们日常使用的项目以及交互设备包含嵌入式系统。嵌入式系统是一个是“隐藏”在交互设备中的系统。手机、应答机、微波炉、录像机、DVD 播放器、视频游戏机、数码相机、音乐合成器和汽车都含有嵌入式处理器。一种新型汽车可以包含多达 80 个嵌入式处理器。这些嵌入式处理器通过控制防抱死制动、气温控制、发动机控制、音频系统控制和部署安全气囊，等等，让我们享受安全与舒适。

嵌入式系统要承担迅速有效地反应外部“模拟”环境的额外负担。这可能包括一个按钮推动的响应，或在碰撞发生时一个传感器触发一个气囊，或者手机接到一个电话。简单

地说，嵌入式系统拥有时限，它可以硬实时或软实时。鉴于嵌入式系统自然的“隐藏”性质，它们还必须反应和处理没有人类干预的情况下不寻常的条件。

在嵌入式系统中，DSP 是主要用于信号处理。它能够执行复杂的信号处理实时函数，这是 DSP 超越其他类型嵌入式处理器的关键优势。DSP 必须实时响应环境中的模拟信号，将它们转换成数字形式，并且对那些数字信号进行加值操作，如果需要，处理后的数字信号转换成模拟形式返回环境当中。

我们仍然需要讨论允许 DSP 迅速高效执行实时嵌入式任务的特殊架构和技术。第 3 章将继续讨论这个话题。

与桌面或大型机不同的是，编写嵌入式系统的程序需要一种完全不同的方法。嵌入式系统必须能够响应外部事件，并利用一个可预测且可靠的方法。实时程序不仅要执行正确，也必须及时地执行。一个迟到的应答就是一个错误的应答。因为这个需求，我们将在本书的后面研究相关的问题，比如并发、互斥性、中断、硬件控制、多任务和处理。