TOPPERS中级实装讲座

（H8/3069F版：实时系统的构建）

应用程序实习篇：第1天

TOPPERS工程

培训工作组

关于本教材的注意事项

- **关于本教材的使用请注意以下几点。**

1. 关于著作权的表述

2. 关于本资料如果有任何意见、建议、想法或疑问，请通过电子邮件与TOPPERS办事处进行联系。

3. 关于本资料的内容，出于调整和改善的目的，可能会对内容进行修订，且不予通告。

日 程 表

- 第1天

1.构建应用RTOS的实时系统	1.5小时
2.介绍实习课题	0.5小时
3.确认开发环境	0.5小时
4.制作话题烧水壶1	3.5小时
4－1.制作提示1	(0.5小时)
4－2.制作提示2	(0.5小时)

- 第2天

1.制作话题烧水壶2	1.5小时
2.评审 设计、实装的评审	0.5小时
3.提高实时性的方法	1.0小时
4.任务负荷的检验与对应	0.5小时
5.话题烧水壶中断的对应	2.0小时
总结	0.5小时

应用程序实习篇的目标

- 学习实时系统的设计技术。
- 通过“话题烧水壶”样机的软件开发，学习实时系统的实装技术。
- 学习实时调度理论、以及使用中断控制提高实时性的实装技术。

构建应用RTOS的实时系统

1. 构建应用RTOS的实时系统
2. 介绍实习课题
3. 确认开发环境
4. 制作话题烧水壶1

对应用RTOS的实时系统的构建进行说明。

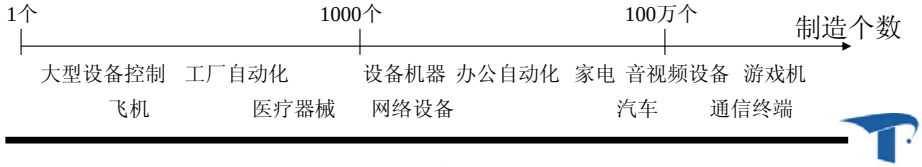
嵌入式系统复习1
系统的多样性

目前还没有多样且明确的技术培训

- 嵌入式系统无论是从规模还是特性那个方面来说都是非常多样性的。
- 目前还没有提出有效的分类标准。
- 左右嵌入式系统开发手法的公式：

总成本＝开发成本＋制造成本×制造个数

并且有必要再加上time_to_market的因素。
近年即使在大批量生产中也不能忽视开发成本。



2006/12/22 TOPPERS工程認定 6 TOPPERS

嵌入式软件开发的特征：

・与硬件密不可分的程序设计
各个系统的硬件构成及外围设备等都是不同的，所以在进行程序设计时要直接对应硬件。
在整个开发中，技术含量高的部分占有很大的比例，可以说开发是很难的。

・开发环境与目标环境的分离
不一定总能在目标系统内进行开发，有时必须考虑用交叉开发环境、远程调试、模拟等方式进行开发。
而且，有的系统需要在系统未停止的状态下进行验证、调试。

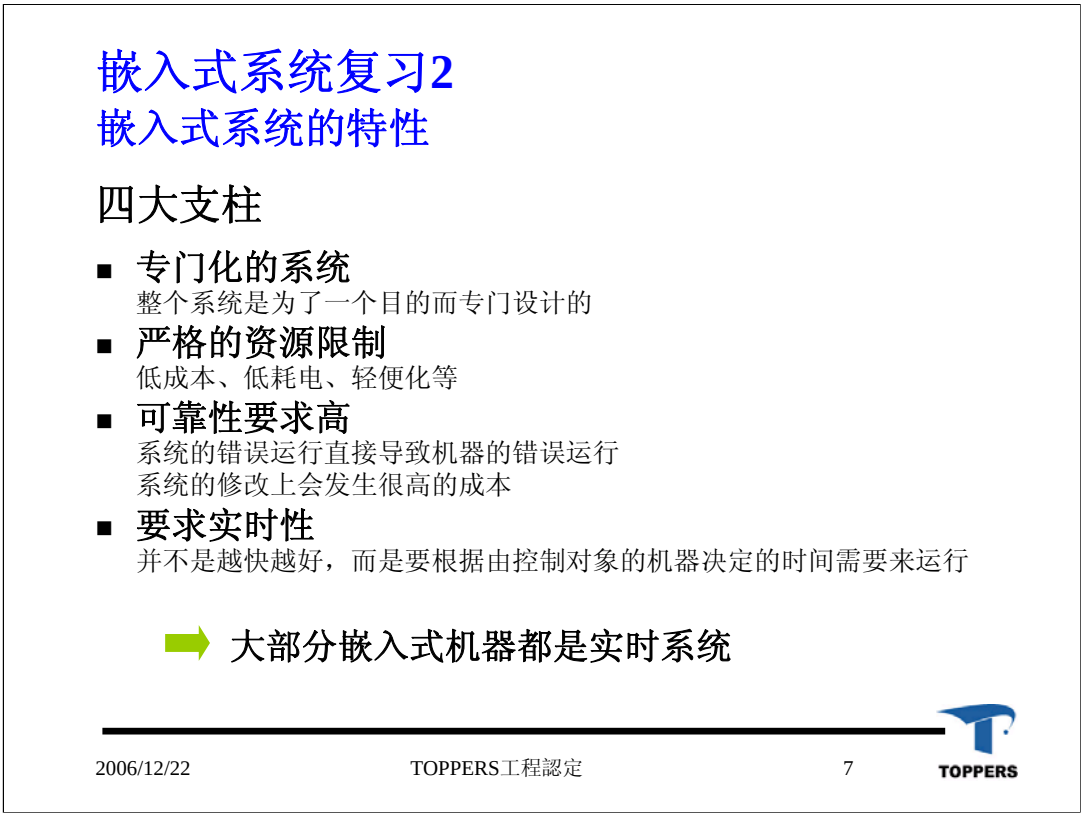
・与硬件的协调设计・并行开发（的可能性）
慢慢实现在完成系统前可以进行验证、调试。

・多样化平台（硬件、OS）
很多系统都针对硬件或OS进行了优化。

・验证成本高
由于要在高可靠性以及由时机问题带来的不可预测性的前提下运行控制对象的机器，所以检证成本很高。

・有时也需要面向异常处理的软件设计。

・前提是系统内的软件值得信赖。



从嵌入式系统的多样性中可以总结出以下几个特性。

- 专门化的系统

  整个系统是为了一个目的而进行设计的。

- 严格的资源限制

  批量生产时这一特性尤为显著，严格要求降低成本。

  其中包括低耗电、运行环境（温度、运行环境等）、轻便化等特殊限制。

- 高可靠性

  通常都是系统的错误运行直接导致机器的错误运行，所以得不到保证是不行的。

  有时也会成为PL法的对象。

  有时在系统的修改上需要很高的成本。

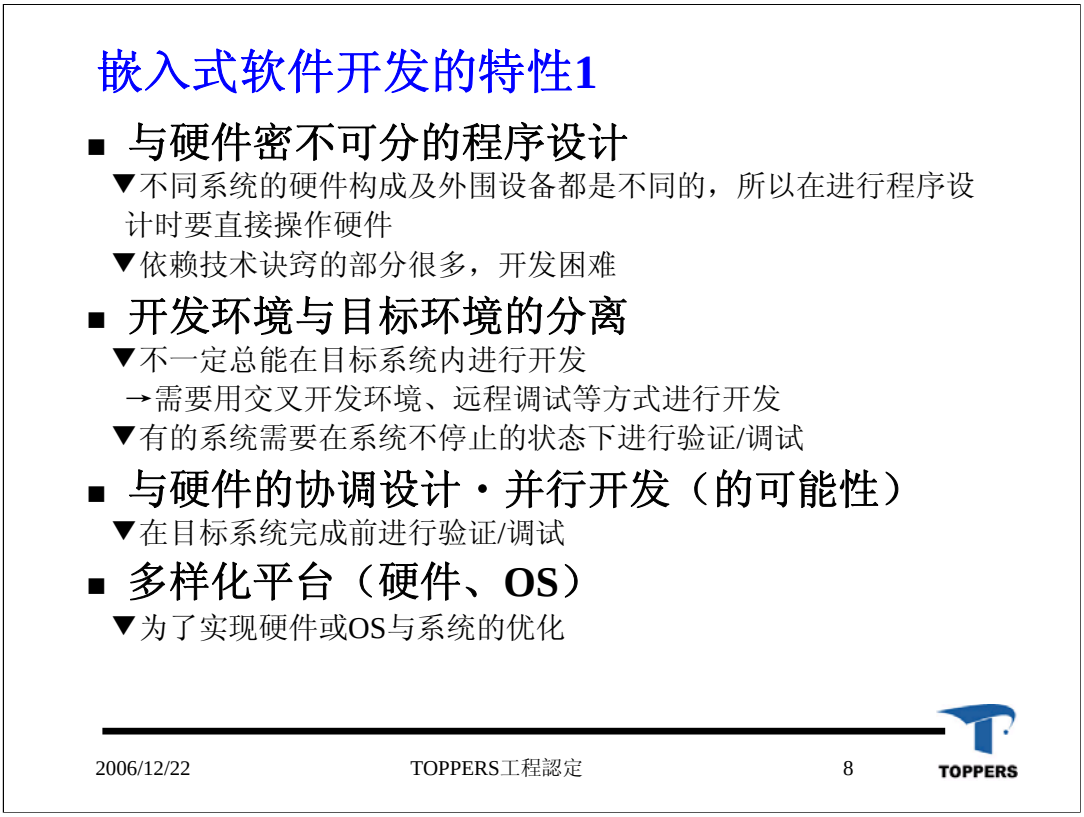
  用户会对机器的可靠性抱有很高的期待。

- 实时性

  需要根据由控制对象的机器决定的时间因素运行。

  并不是越快越好。

  能够实现这一点的系统就是实时系统。



嵌入式软件开发的特征：

1）与硬件密不可分的编程

各个系统的硬件构成以及外围设备等都是不同的，要开发的程序应该是直接控制这些硬件。在程序开发中不仅需要程序设计开发能力，硬件的控制技术、对机器的特殊技能也是十分必要的。这些基本都要借助于经验以及技能等，开发就会有一定的难度。

2）开发环境与目标环境的分离

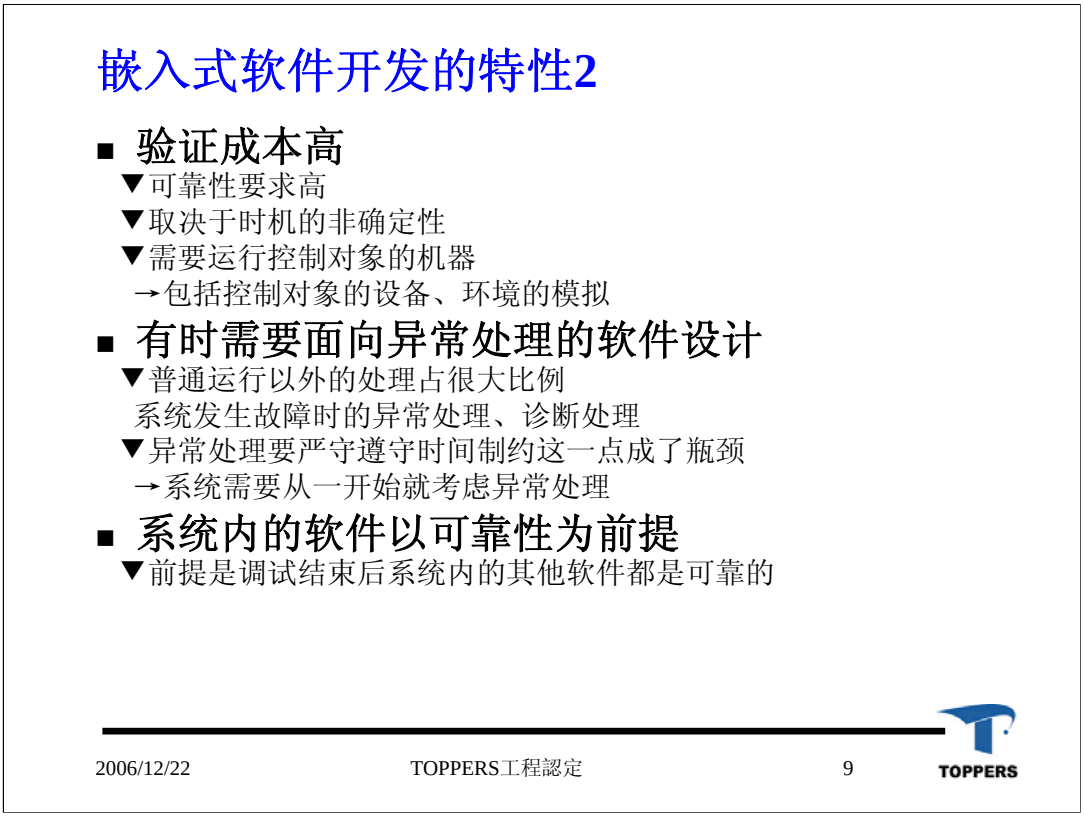
通常系统的运行环境与开发环境都是分开的。那么，为了对产品进行管理，就必须事先准备好交叉开发环境，并考虑好怎样才能实现已构建的运行方式在目标环境下的运行、调试。而且，在进行验证、调试的时候，控制对象的机器可能会异步运行，或者在多任务环境下一一停止任务来进行调试时控制对象的机器可能会不正常运行等等，这就需要系统能在不停止的状态下进行验证、调试。

3）有时要进行与硬件的协调设计・平行开发

有时在目标系统完成前就必须开发嵌入式软件。此时就需要在模拟器环境下进行开发，或在工作站上模拟运行硬件来开发软件。硬件的协调设计系统成本很高，所以还没有普及。

4）多样化平台

嵌入式系统是十分多样的，所以需要在对应各个系统的多样化平台（这里指硬件、OS）下进行开发。



5）验证成本高

主要原因：

・嵌入式系统追求高可靠性

・必须考虑控制对象的机器以及其运行环境等引起的各种时机事件，有很多不可预测的因素

・如果不运行控制对象的机器，就无法测试

6）有时需要面向异常处理的软件设计

一般在嵌入式系统中普通运行以外的处理占有很大的比例。如系统发生故障时的异常处理、诊断处理、以及与之相伴的特殊设定、用户（维修人员）接口等。在进行诊断处理时，用于判断异常运行时的运行状况的获取LOG的功能、监视运行的处理等程序就会成为系统负荷。在进行系统设计时要考虑到这些因素。

7）前提是系统内的软件值得信赖

前提是系统的调试、验证结束后系统内的其他软件值得信赖。

嵌入式软件的开发环境

下面介绍一下嵌入式软件的开发环境。

嵌入式软件的开发环境
介绍一下嵌入式软件开发中独特的工具

- 交叉开发环境与开发工具
  - ▼目标系统与开发用的系统（主系统）分离的软件开发环境叫做交叉开发环境
  - ▼交叉开发环境的开发工具
    - ・交叉编译器
    - ・交叉汇编程序
    - ・交叉连接程序
    - ・远程调试器……各种方式
- 测试/验证环境
    - ・不使用实机的测试 / 验证环境

・交叉开发环境

目标系统与开发用的系统（主系统）分离的软件开发环境叫做交叉开发环境。
TOPPERS/JSP的开发环境就是交叉开发环境。而GNU为以下对应。

・交叉编译器	--- XXX-XXX-gcc　XXX-XXX-g++
	gcc是C语言的编译器、g++是C++语言的编译器
	从各语言生成中间对象(*.o)
・交叉汇编程序	--- XXX-XXX-as
	从对应各CPU的汇编语言生成中间对象
・交叉连接程序	--- XXX-XXX-ld
	将中间对象重新配置到实际的ROM、RAM上
・远程调试器	--- GDB

不使用实机的验证环境有TOPPERS/JSP的Windows版的设备模拟器、TORNADE的Windows模拟器等。这些都可以不使用实机进行嵌入式软件的验证。

面向嵌入式软件的开发工具的特征1
编译器的优化

- 以减少代码容量为目的的优化
  - ▼在编译器的优化技术中，既有能同时实现高速化与减少代码量的技术也有两者矛盾的情况
    - 前者的例子：去掉共通部分公式
    - 后者的例子：循环展开
  - ▼有些编译器可以在编译器选项中指定是重视高速化还是重视降低代码量
- 以减少耗电量为目的的优化
  - ▼正处于研究课题的阶段
  - ▼与高速化的优化基本一致

刚开始在嵌入式领域内使用C语言时，与汇编语言相比，当时的C语言在编译后的执行大小以及执行速度等方面存在问题。后来，多次改善编译器，利用优化技术实现了接近汇编语言在编译时的执行大小以及执行速度。优化包括高速化和代码量的减少。此功能既有能同时实现的也有两者矛盾的。

能同时实现的例子：

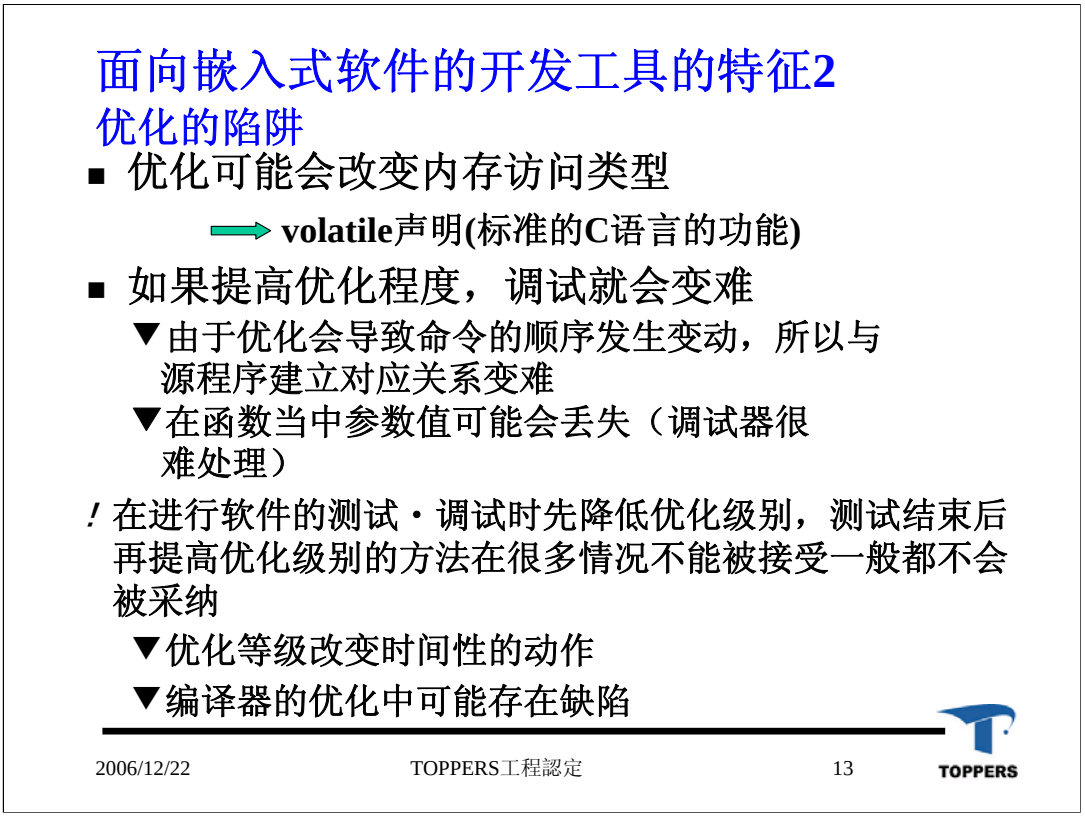
```
{
        int i, j;
    i = 10;
    j = 20;
    i = 30;
        ・・・
}
```

在上述记述中，由于第一个“i=10;”没有意义，所以在进行优化时要将其删除再生成代码。此时，由于删除了没有意义的执行代码，所以能够同时实现高速化与代码量的减少。

矛盾的例子：{

```
    int i;
    int exp[100];
    for(i = 0 ; i < 100 ; i++)
      exp[i] = 0;
}
```

在上述记述中，如果像exp[0] = 0 ; exp[1] = 0; ・・・exp[99]=0;这样进行优化，那么就不需要管理i，所以虽然实现了高速化但代码量却增大的。



优化存在几个陷阱。

```
{
  unsigned char *port5 = (unsigned char *)H8_P5DR;
    *port5 = 0x02;
    soft_delay(100);
  *port5 = 0x03;
    ・・・・
}
```

在刚才的优化的例子中的上述记述中，当向端口5写入0x02后又想写入0x03时，即使删除了最初的0x02的写入，但硬件也无法正常运行，此时，如果使用volatile声明可以回避此问题。

如果提高优化等级，调试就会变难。优化会导致编译时生成的汇编程序命令的顺序发生替换，这样就会很难与源程序相对应。而且还可能会导致函数正在执行的时候没有引用的参数值丢失，这样就无法实现具有参数值显示功能的调试器的功能。

有一种方法是在测试・调试时先降低优化等级（在调试模式下生成代码），测试结束后再提高优化等级。但此方法一般都不会被采纳，其主要原因如下所示：

1）关于优化等级，如果程序的Runtime改变，控制机器的运行时机就会改变，这会成为新的障碍。

2）编译器的优化中可能存在缺陷，或可能导致缺陷。

基础知识：volatile声明

- **volatile声明是用来修饰值可能发生变更的任意类型（volatile=挥发性的）**

```
char *device_register = (char *)0xfffe0020;
while ((*device_register & 0x01) == 0);
```

陷入无限循环！

```
volatile char *device_register = (volatile char *)0xfffe0020;
while ((*device_register & 0x01) == 0);
```

！volatile声明限制优化

注意）指针的**volatile**声明

volatile char *device_register
→ device_register指向的对象为 volatile

char *volatile device register
→ device_register本身为volatile

・volatile声明

表示声明的值为不明的类型，可能会发生变化。volatile的意思是“挥发性的”。device_register指向输入用的端口，想在内容变为0x01之前添加等待时，通过没有volatile声明的device_register记述，判断为最初引用的值没有发生变化，编译器的优化可能生成只引用一次的代码。所以只引用最初访问的第0位，如果该值为0则后面的都不再引用，就可能会进入永久等待状态。在这种情况下，如果先进行volatile声明，编译器就不会成为优化的对象，而且while语句可以正确引用，这样在device_register的BIT0的值变为1时进入下一个步骤。

对指针进行volatile声明时，像上述例子那样，

1）放在char*的前面时，表示指针引用的对象为不明的类型，会发生变化。

2）放在char*的后面时，表示指针本身为不明的类型，会发生变化。

面向嵌入式软件的开发工具的特征3

- 直插式汇编程序功能
  - ▼在C语言的源文件中，直接写汇编程序命令的功 能
  - ▼在使用特殊的寄存器、命令时是必要的
- 内存分配的指定
  - ▼详细指定将程序空间、数据空间等放到哪个内存地址的功能是必要的
    - ⇦ 想使用不同性质的内存
      例）*高速的SRAM,DRAM*
      电池备份的RAM
  - ▼用编译器（或连接程序）指定放置变量的区域（段）
  - ▼对连接器指定各区域的链接方法以及其首地址

・直插式汇编程序

直插式汇编程序是在C语言的记述中直接记述汇编程序的功能。下面是H8的直插式汇编程序的例子，是从config/h8/cpu_insn.h的记述中摘录的。在Asm();中记述汇编代码。如果这样记述，C语言的源代码就会取决于CPU，所以一定要注意。

```
/*
 *读取条件码寄存器（CCR）的当前值
 */
Inline UB
current_ccr(void)
{
    UB              ccr;
    Asm("stc.b ccr,%0l" : "=r"(ccr));
    return(ccr);
}
```

・内存分配

为了使嵌入式程序在ROM、RAM上正确运行，需要详细指定程序的空间、数据领域的功能。在本讲座使用的GNU的环境下指定区域，并通过连接程序(ld)将其地址进行再配置，这样就实现了此功能。

config/h8/akih8_3069f/debug.ld是用于再配置为调试用的指定文件。本讲座的所有运行环境都是在RAM上，所以debug.ld将所有地址都设定为在RAM上。

面向嵌入式软件的开发工具的特征4

- 生成可ROM化的代码
  - ▼将程序放到ROM上运行的功能
  - ▼程序代码与常量数据放在ROM上
    - 常量数据：C语言const声明的变量
      字符串常量（用“”括起来的字符串）
    - 注意）指针的const声明与volatile声明一样需要注意
  - ▼将初始化的数据放到ROM上，在运行程序前将其拷贝到RAM上使用
    - ！应该注意数据的地址与运行时应该引用的地址是不同的
      - ⇒ 启动模块的作用

2006/12/22 TOPPERS工程認定 16 TOPPERS

・ROM化的代码指定

在进行实际的商品化时，必须将程序、常量数据配置到ROM上。如果不这样做，每次都需要用下载工具来下载程序等。配置到ROM上后，用RAM写入工具进行写入，这样不用下载就可以运行了。但是，存在下述初始化变量时，如果将其配置到ROM上，配置后就不能改写数据了。

```
#define INITIAL_MODE 10
int Start_Mode = INITIAL_MODE;
```

这些数据在ROM上准备了初始值，在执行时（接通电源时或复位时）需要将其内容拷贝到实际执行的数据的地上。这是启动模块的功能。本讲座教材的H8用TOPPERS/JSP的启动模块记述在onfig/h8/Start.S中。大概是在第100行附近进行拷贝处理。

嵌入式软件的调试环境1
什么是调试器（debugger）

- 协助发现缺陷（**bug**）的工具
- 主要的功能（源代码调试器）
  - ▼**程序的下载**
  - ▼**程序的运行与停止**
  - ▼**程序的读取**（显示对应的源代码）
  - ▼**数据的读取、改写**（变量、内存、寄存器）
  - ▼**函数（或子程序）的调用**
  - ▼**断点**（执行中断、访问中断）
  - ▼**单步执行**（源代码级别、机器语言级别）
  - ▼**堆栈状态的读取、跟踪**

调试器的主要功能：

1）程序的下载

使用调试器的通信功能将开发系统中的程序下载到目标系统的RAM（有时是ROM）上。

2）程序的运行与停止

从指定的位置开始运行下载的程序，一直到指定的位置结束。

3）数据的读取、改写

对目标系统中的数据进行读取、改写。并对处于停止状态的CPU寄存器进行读取、改写。

嵌入式软件的调试环境2

- **ROM监视器**
  - ▼先放到目标系统的**ROM**上，再用串行接口等将目标系统作为终端进行连接使用的调试器
  - ▼没有源代码级别的调试功能
- 远程调试器
  - ▼用主系统的调试器实现高端的调试器（源代码级别的调试功能等）
  - ▼目标系统上仅实装用于查看・设定目标系统的状态的最低限度的软件（调试接口进程、调试监视器）
    - ・寄存器 / 内存的读取 / 写入
    - ・运行的开始 / 重新开始、单步运行
    - ・程序的强制结束（复位）

以下是本实习中使用的苫小牧高等专科版H8简易监控程序的命令。

ld	将摩托罗拉S记录格式的调试模块下载到RAM上 。
go [<addr>]	从地址<addr>开始执行调试模块。 如果省略<addr>，则从指定为复位异常处理向量地址的地址开始执行。
dw [<addr> [<len>]]	从地址<addr>输出长度为<len>Word（2字节）的数据。 如果省略<addr>，则从上一次输出的数据的下一个开始输出256字节。 但如果是第一次执行，则从内置RAM的开始地址开始输出256字节。
db [<addr> [<len>]]	从地址<addr>输出长度为<len>字节的数据。 如果省略<addr>，则从上一次输出的数据的下一个开始输出256字节。 但如果是第一次执行，则从内置RAM的开始地址开始输出256字节。
ew [<addr>]	从地址<addr>开始为Word数据的写入模式。 如果省略<addr>，则从上次最后写入的地址的下一个开始为写入模式。 在写入模式下不进行任务输入直接换行时，不会对显示的内存进行写入。如果只输入“.”，则结束写入模式。
fb [<addr> [<len> [<init> [<inc> [<repeat>]]]]]	从地址<addr>开始写入长度为<len>字节的数据。 <addr>的省略值是内置RAM的开始地址，<len>的省略值是256字节。 <init>是开始写入的数据类型的初始值，<inc>是开始写入的数据类型的增加值，<repeat>是写入的重复次数。
?	输出简单的命令的格式。

嵌入式软件的调试环境3

- ICE（In-Circuit Emulator）
  - ▼用连接ICE（一般是在处理器的socket上插入探针）的方式来代替电路板的处理器，ICE模拟处理器来监视运行
- ROM模拟器
  - ▼不在处理器上而是在ROM socket上插探针的方式来监视处理器的运行，停止/改变处理器的运行时使用NMI
- ICE的问题点
  - ▼由于必须针对每个处理器进行开发，所以造价高
  - ▼单芯片化以及高速缓存等导致无法从芯片外部总线无法监视
  - ▼处理器的高速化导致一插入探针就不运行了／运行不稳定

2006/12/22 TOPPERS工程認定 19 TOPPERS

・ICE（In_Circuit Emulator）
用连接ICE的探针的方式来代替电路板的处理器，ICE模拟处理器的处理。这样就可以从ICE内部实时
监视处理器的运行，而且可以在需要时停止或改变运行。这样，就可以实现调试器无法实现的在不停止系统的状态下进行的调试作业。而且，通过获取问题发生之（前）后的运行记录，还可以解析普通运行中发生的障害。好像用户接口（或用户界面）一般都是在主机上运行。也可以看成是通过切断处理器和基板之间的连接而实现了强大的调试支援功能。

・ROM模拟器
是不使用程序处理机，而是通过在ROM socket上插入探针来监视处理器的指令功能的模拟器。
在停止或改变处理器的运行时使用NMI。

・ICE的问题点
ICE存在很多问题，对应了单片调试功能的ICE是现在的主流。由于ICE必须对每个处理器进行开发，所以价格很高。
→ROM模拟器
单片化、高速缓冲存储器等导致无法从芯片外部监视总线。
→评价芯片（Evaluation Chip）
→单片调试功能
处理器、总线等的高速化导致有时一插入探针就不运行了，或运行变得不稳定。
→单片调试功能

嵌入式软件的调试环境4

- 评估芯片（Evaluation Chip）
  - ▼已将处理器内部的总线引到外部的调试专用处理器
- 芯片上调试功能
  - ▼协助调试的电路实装在芯片内
  - ▼也尝试芯片上调试功能的标准化
    EJTAG、nWIRE、NEXUS等
- 模拟器
  - ▼广泛应用于硬件完成前的软件测试・调试
  - ▼权衡模拟器速度与时间精度
    ⟹在速度・精确度・价格上，有各种各样的模拟器
  - ▼联合模拟器（同时模拟硬件与软件的技术）

・评价芯片（Evaluation Chip）
已将处理器内部的总线引到外部的调试专用芯片
・单片调试功能
请想象协助调试的电路已实装在芯片内，ICE（的子集）已装在芯片内。也尝试了单片调试功能的标准化。
・模拟器
是想在硬件完成前开发并调试软件时被广泛使用的方法。权衡模拟器速度、时间精确度、价格，有各种各样的模拟器。
・Cycle Accurate模拟器（CAS、低速）
・指令集级别的模拟器（ISS、中速）
・C语言级别的模拟器（高速、忽略时间精确度）
也有像联合模拟器这样同时模拟硬件和软件的技术。

实际的调试方法

- 通过输出记录进行调试
  - ▼使用**print**语句或**syslog**语句，将运行记录显示到控制台上，一边确认运行状态、顺序等一边进行调试
  - ▼因为不会停止任务的运行，所以可以按照控制对象设备的运行步骤进行确认
- 使用分析仪进行调试
  - ▼**ICE**等机器的主要目的是调试硬件以及与硬件的接口部分
    - ⇒与调试功能相比，更重视跟踪功能
  - ▼在发生问题的环境下，将分析仪连接到端口、内存上来获取问题发生前后的访问、数据等的记录进行解析

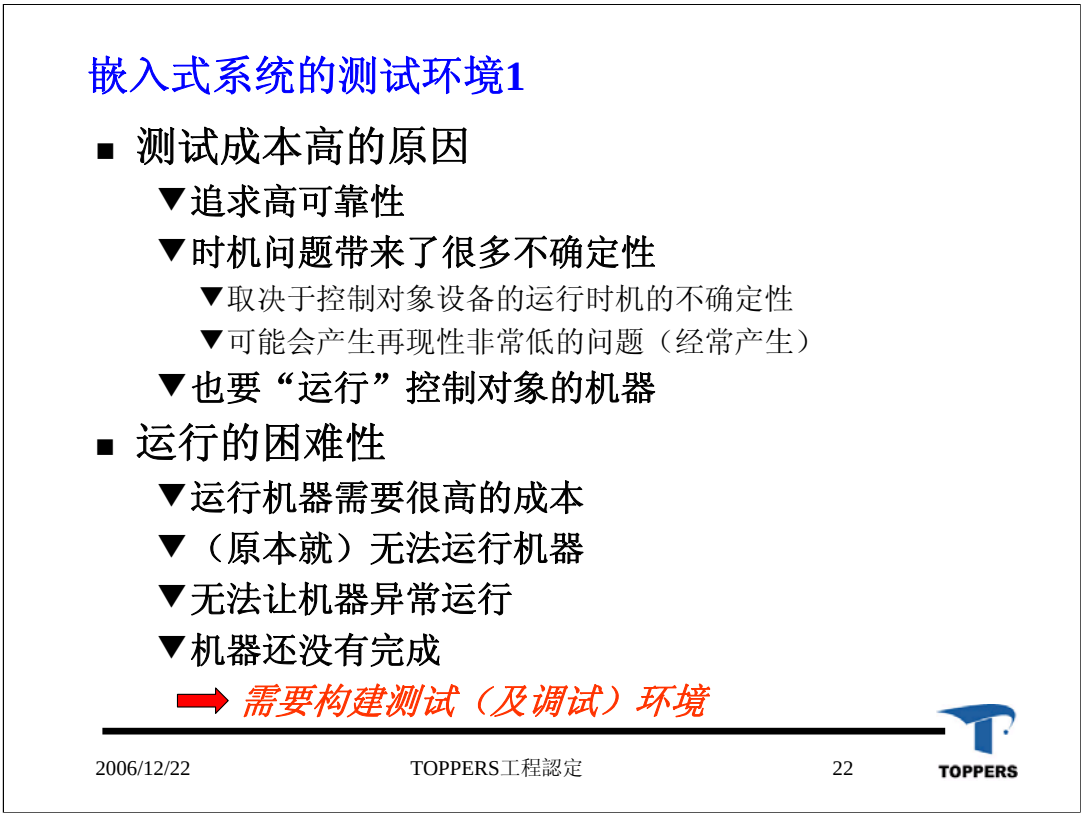
・通过输出记录进行调试

在多任务环境下进行调试时，如果用调试器停止对象任务，对机器的控制顺序就会被打乱，这样就无法正确控制。所以，一般都是使用syslog语句、printf语句来显示运行记录关进行调试，此方法的调试效率比较高。尤其是在下述开发环境下：

1）在LINUX、Windows这样的开发环境下进行模拟开发时，如果要在目标环境下运行可以引用已确认运行的源代码的中间件，可以一边确认记录的输出一边确认运行。

2）在中断、周期句柄这样的非任务语境下，有时无法用调试器进行中断。此时，通过syslog语句输出的记录进行确认是非常有效的方法。

在本讲座的开发中，通过输出记录进行开发是非常有效的方法。



・嵌入式系统的测试环境

嵌入式系统的测试成本高的主要原因：

1）嵌入式系统的错误运行经常会直接导致机器的错误运行，而且机器的错误运行有时会关系到人命。另外，如果大量出售，维修的成本也会很高。所以，有很多追求高可靠性的嵌入式系统。

2）有时需要嵌入式系统能够对由机器的运行时机、各种机器的组合运行等引起的不可预测性的因素进行控制。在这种情况下，经常会产生再现性很低的障碍，作为验证方法需要黑盒测试。在嵌入式系统中，由时机问题带来的不可预测性因素有很多。

3）在测试时，也需要运行控制对象的机器。如果该机器为样机，通常也会价格很高。

在嵌入式系统下，需要运行控制对象的机器来进行测试。在进行测试（包括调试）时，需要有计划地构建测试环境。但有时在测试或调试时运行机器这件事本身就是很困难的。

1）运行机器的成本就很高。

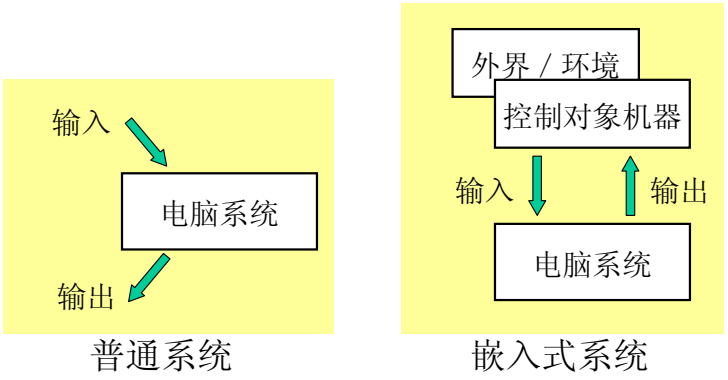
2）如果不具备各种条件，可能无法运行机器。

3）为了使机器异常运行，有时需要改造机器本身。有时为了测试还需要准备特殊的机器。

4）有时机器可能还没有完成。

嵌入式系统的测试环境2

- 需要“运行”控制对象的机器
  - ▼控制对象的机器（机械方面、电子方面、物理方面、化学方面等等）的系统也很复杂
  - ▼只是在输入之后确认输出是无法进行测试的



在企业版的系统中，一般是开发环境＝运行环境，所以比较容易构建调试、测试环境。但在嵌入式系统中，需要根据控制对象的机器（有时会根据外界、环境进行特殊的运行）的输入实时给机器提供输出数据。有时控制对象的机器的系统也很复杂。

应用实时OS的系统设计的方针

应用实时OS的软件开发1

上游开发

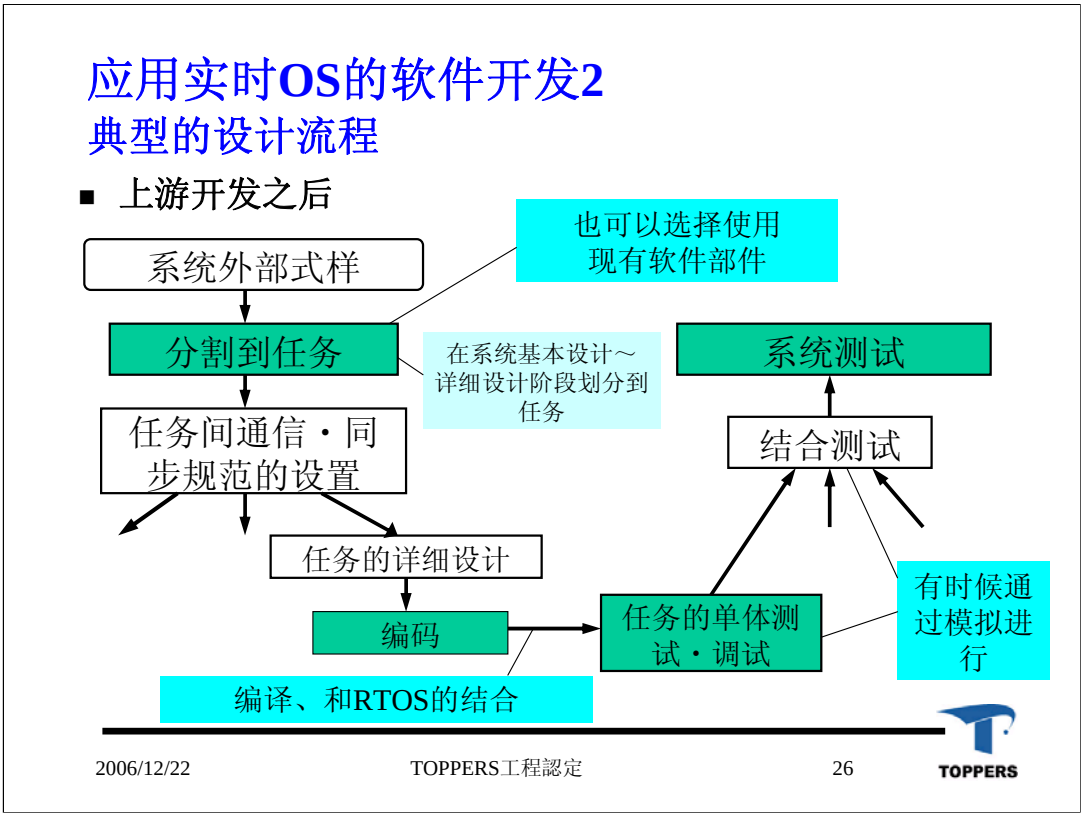
- 根据计划式样书、需求式样书等需求来分析系统
- 分析方法中包括结构化分析、面向对象分析、公司独特的方法等各种各样的方法
- 创建基于UML的模型的事例越来越多

结构化分析	面向对象指向分析
事件列表	语境图
语境图	用例、用例描述
数据字典	类图、对象图
DFD0（基本分析）	状态迁移图、时序图、事件列表
非功能需求的调查	

以下是代表性的例子

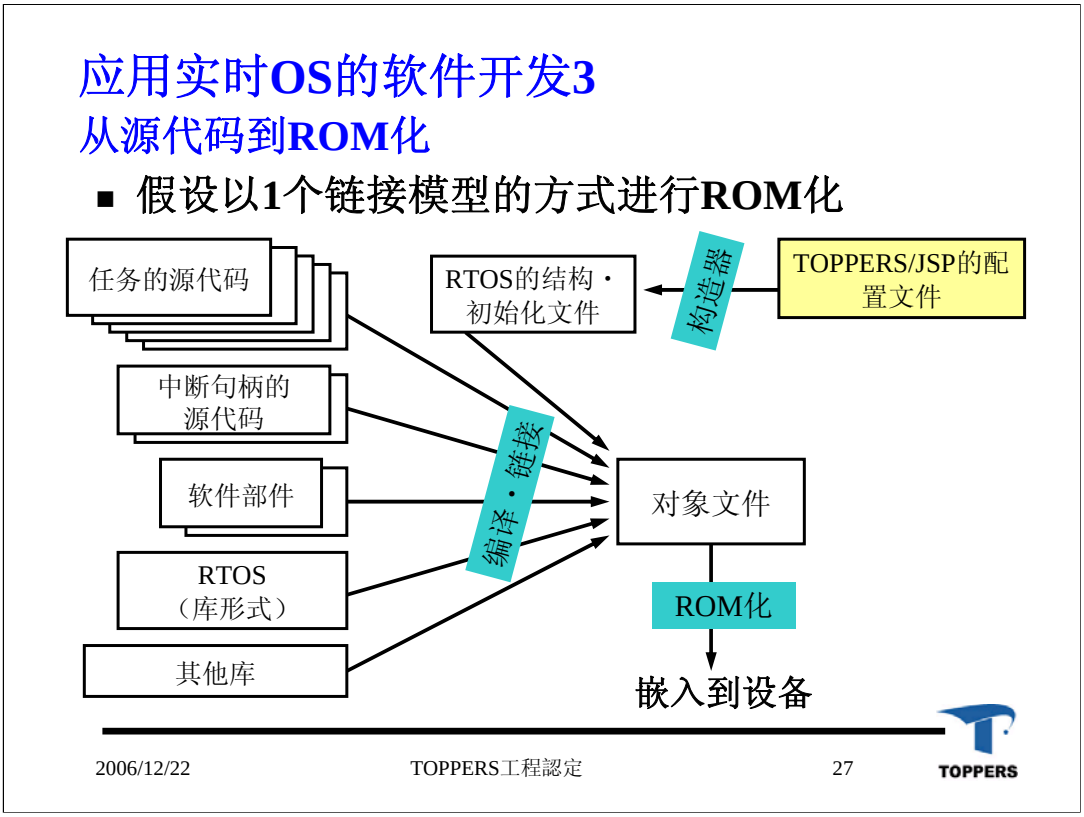
上层开发

根据计划式样书、需求式样书等进行系统分析。在之前的讲座中都是使用结构化分析来进行分析的，但在实际中其实有很多种方法，包括公司、开发团队的独特的方法。在上述例子中，列举了结构化分析时的图、面向对象分析中用模型来表示软件时的UML图。



根据分析的结果来设定实际运行的程序。应用实时OS时，需要将系统分成几个任务。划分的方法会在后续章节中进行说明。然后确定任务间的通信式样。软件开发方法有瀑布型和螺旋型。瀑布型是按照最初确定的式样创建程序，一直到最后的测试。而螺旋型是将开发分成几个迭代来进行。先在第一个迭代中设计出具备重要的基本功能的任务，在确认已达到功能、性能要求后，再在后面的迭代中扩展功能。在系统中有很多不确定的事项或规模太大时，螺旋型开发比较适合。

划分了任务并确定了通信方法后，对各个任务进行详细设计、编码。对已完成的任务进行了单体测试后，将多个任务结合起来进行结合测试。系统测试是根据外部式样书对各式样进行确认测试。



在应用普通的实时OS的系统中，创建1个链接模型来进行ROM化。如果系统规模很大，也可能会根据需要下载进程化的程序来运行，在这里就不进行说明了。

上图是实时OS的1个链接模型。详细内容在TOPPERS/JSP的各开发例子中进行说明。对任务的源代码、中断句柄的源代码、软件部件（也有库的形式）、TOPPERS/JSP（多数都是库的形式）、其他库、OS对象的初始化以及结构等进行源代码化后形成的内容进行编译、链接、ROM化，然后嵌入到机器中。在TOPPERS/JSP中，如果创建配置文件，就要使用构造器（将配置文件转换成源代码的程序）将其转换成具有OS对象的初始化以及结构的源代码。在使用了本实习中的TOPPERS/JSP的系统中，将链接后的对象转换成摩托罗拉的S格式，并使用ROM监视器下载到H8/3069F的板上的SRAM中运行。运行ROM时，同样可以使用工具将S格式的数据写入到单片CPU上的闪存ROM中运行。

应用实时OS的软件开发4
RTOS的配置

- 指定RTOS的结构、初始状态的步骤
- 什么样的结构可以变更取决于各RTOS，如：
  - ▼使用/不使用的功能
  - ▼各对象（任务、信号量）等的最大数
  - ▼任务优先级的层次
  - ▼内存的分配/配置
  - ▼各对象的生成・定义信息
    （静态生成对象时）
- 配置的机制
  - ▼根据配置改变RTOS代码的机制
  - ▼配置结果是转换成一个（或多个）源代码的方法

μITRON4.0规范中的配置

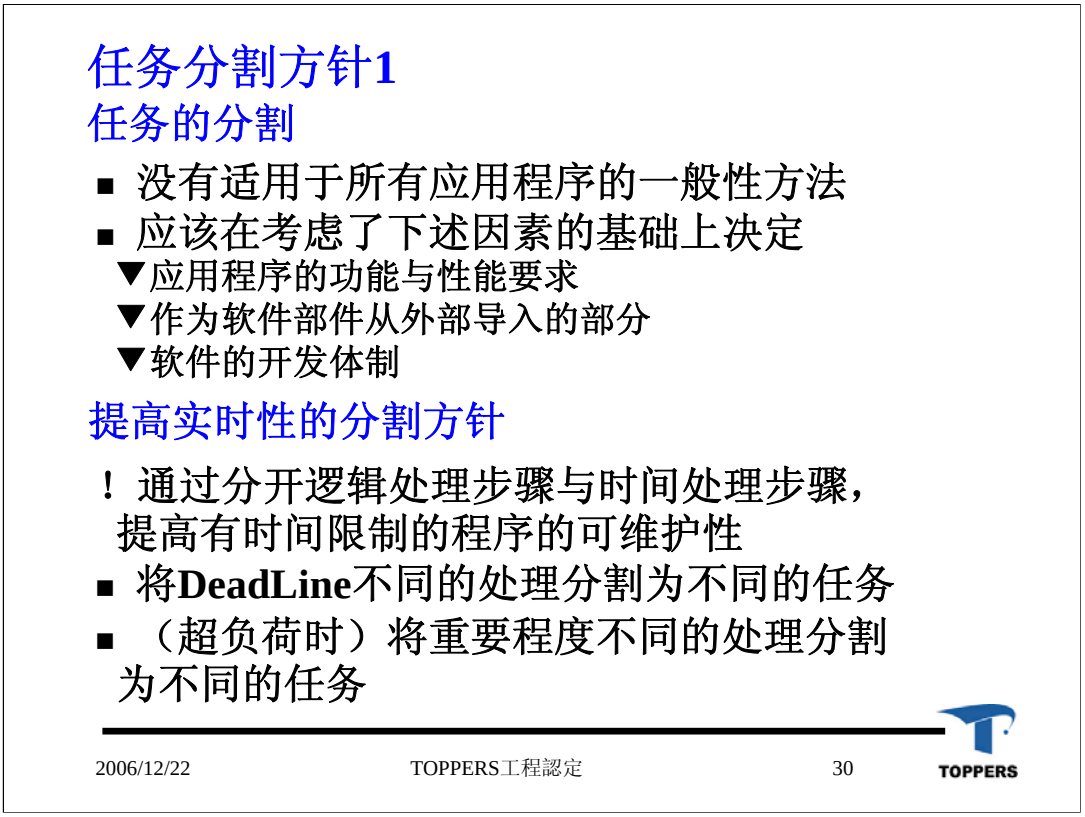
- 静态生成内核对象
  → 把对象生成信息的配置文件中的描述标准化（静态API）
- 静态API的例子

```
CRE_TSK ( tskid, { tskatr, exinf, task, itspri,
                   stksz, stk });
ATT_ISR ({ intatr, exinf, intno, isr });
```

- 可以加上软件部件的静态API一起记述
- 优点
  ▼很容易将软件移植到别的内核上
  ▼很容易嵌入软件部件
  ▼不需要分别记住服务调用与静态API

在基于μITRON4.0的标准文件的实时OS下，使用构造器将记述了静态API的配置文件转换成源代码。在ITRON TCP/IP式样中也有用于生成通信端点的静态API，它生成协议栈的对象。

普通的结构是以菜单的形式设定结构。所以，如果OS的厂家或CPU改变了，就需要重新设定。但用静态API记述的结构是标准式样，所以它不受厂家、CPU的影响，可以进行再利用。在直接用C语言记述RTOS的初始化、对象时，需要实装环境、实时OS的知识。但如果使用静态API记述，那么此技术就会隐藏在构造器中，由构造器的创建者完成实装的部分。所以实时OS的使用者不需要进行实装的设计。



嵌入式系统的机器范围很广。根据非功能要求、实装的构架等，任务的设定会受到很大的影响。所以，不存在能够对应所有应用程序的一般性任务分割方法。

在划分任务时请考虑下述因素：

1）应用程序的功能和性能的要求

2）作为软件部件从外部导入的部分

3）软件的开发体制

实时OS的任务功能的特点是可以将逻辑处理步骤与时间处理步骤分离。比如说，结构化设计、面向对象设计是通过对压缩度、耦合性进行优化来实现模块化、对象化的方法，它不包括分开逻辑处理步骤与时间处理步骤的机制。所以它有任务分割与模块、对象不能正确对应的一面。在为了提高实时性而划分任务时，请注意以下几点：

1）将最后时间不同的处理划分为不同的任务

最后时间是该功能被规定的结束时间，在实时系统中必须在结束时间内结束处理。

将最后时间不同的处理划分为不同的任务可以使优先级的设定等容易对应最后时间。

2）（超负荷时）将重要程度不同的处理划分为不同的任务

由任务化产生的重要程度不同的任务通过以不同的优先级运行，能够实现超负荷的对应。

任务的分割方针2

模块化的分割方针

！为了实现软件的结构化・开发容易化而进行分割

- 将不同的功能模块、不同设备的操作分割为不同的任务
- 将不同类型的处理分割为不同的任务
  - ▼分割为I/O任务与内部处理任务
  - ▼将系统的监视・维修・诊断用的任务分割为不同的任务
  - ▼也有将发生异常时、故障时的处理分割为不同任务的方法
- 将容易进行再利用的单位分割为单独的任务
- 将完整的处理单位分割为同一个任务
  - ▼将一个状态迁移、输出一个结果的处理与数据流关闭的处理分割到同一个任务中
  - ▼也有根据各种状态（模式）分割为不同任务的方法

→ C规范

本次是以软件的结构化、开发容易化为方针来考虑任务的划分。结构化分析、面向对象分析的结果是划分的主要因素。

1）将不同的功能模块、不同设备的操作划分为不同的任务。

2）将不同种类的处理划分为不同的任务。如下述方针：

・划分为I/O任务与内部处理任务。

・将系统的监视・维修・诊断用的任务分割为不同的任务。

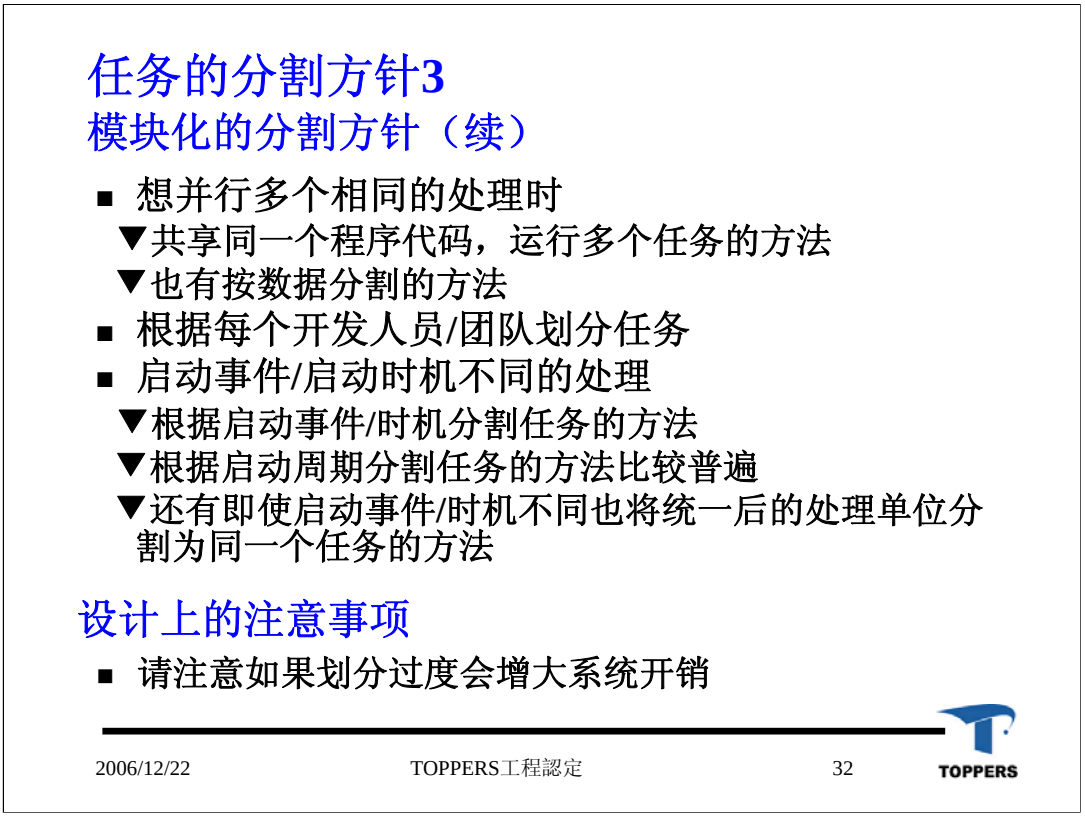
・也有将发生异常时、故障时的处理划分为不同的任务的方法。但也有不划分为不同的任务的方法。

3）将容易进行再利用的单位划分为不同的任务，这样在功能升级时比较容易对应。

4）将统一后的处理单位划分为同一个任务。

・将获取一个状态迁移、一个结果的处理与数据流关闭的处理划分到同一个任务中。

・也有根据各种状态（模式）划分不同任务的方法。例如，通过结构化分析进行结构化的设计时使用C式样的方法。



5）想并行多个相同的处理时

・对同一个程序代码进行任务化，使其对应不同的数据来并行执行

・上述方法要占用很大的系统开销、内存资源，所以也有将不同的数据状态迁移到一个任务中的方法。

6）根据开发人员、团队划分任务，此方法比较容易开发。

7）根据启动事件、启动时机不同的处理进行划分。

・有根据启动事件、时机划分任务的方法。

・根据启动周期划分任务的方法比较普遍。

・还有即使启动事件、时机不同也将统一后的处理单位划分为同一个任务的方法。

注意：

如果系统中的任务分割过度，就会加重下述问题。

・系统开销增大

・内存效率低下

・排他控制复杂化

任务优先级的指定

优先级的确定方法（在明天的讲座中会详细说明）

- 在满足时间限制的范围内，着急的任务（限定时间短的任务）优先
- 暂时超负荷时，如果无法满足时间限制，那么重要的任务（无法满足时间限制时受影响较大的任务）优先

如何判断是否能满足时间限制？

- 可以应用实时调度理论

暂时超负荷时怎么办？

- 将重要性较低的任务停止/降低精确度。
- 有各种技术（与以前的内容是否矛盾？）

任务优先级的相关内容会在明天的讲座“实时性的提高方法”中进行说明。

任务间通信・同步的设计
共享内存vs消息通信

- 原则上是用其中一种方式能够实现的用另一种方式也能实现
  - ▼一般是共享内存通信的方式系统开销比较小
  - ▼在系统验证方面，通信密度大且能够进行在**RTOS**层进行监视的消息通信比较容易处理
    - →如：很容易进行问题的划分
- 根据状况的不同，可能某个更有优势/更方便
  - ▼信息的流程为**1:n**（包括广播传输方式）或需要信息的时机不确定等情况，共享内存通信的方式比较有优势
  - ▼需要信息排队等待的情况消息通信的方式比较方便

设计上的注意事项1

- 需要注意如果同时使用两种方式，系统设计就会很复杂，而且可能会很难看懂
- 关于信号量/互斥体的排他控制，要注意死锁和优先级倒置
- 在消息通信中，对服务器任务提出处理请求时也可能出现优先级倒置的问题

死锁

- 两个以上的任务互相等待对方处理进行的状态

▼典型的是在以各任务不同的顺序锁定两个以上的信号量/互斥体时发生死锁

→如果能指定锁定的顺序就可以解决

在设计上有以下3项注意点：

1）如果同时使用两种方式，系统设计就会很复杂，而且可能会很难看懂，所以一定要注意。

2）关于信号量、互斥体的排他控制，要注意死锁和优先级倒置。

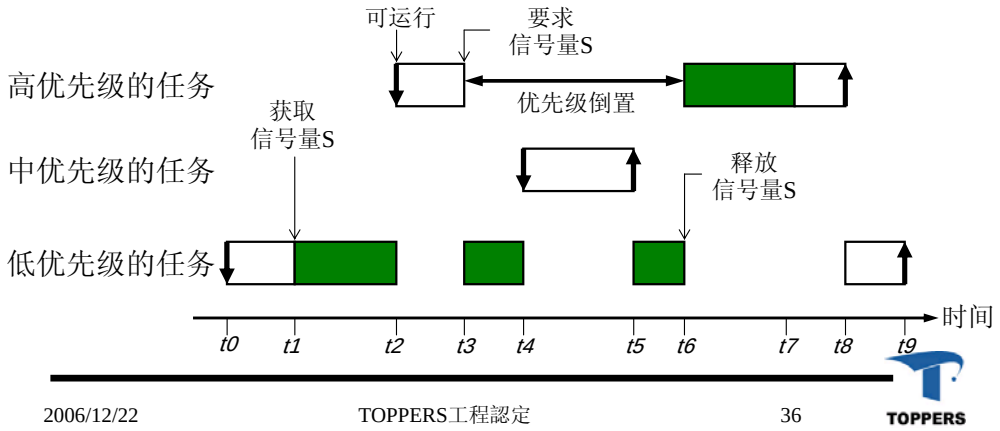
3）在消息通信中，对服务器任务提出处理请求时也可能出现优先级倒置的问题。

（比如，数据队列中优先级高的任务对优先级低的服务器任务提出处理请求时，如果队列已满，优先级高的任务就会进入等待状态）

设计上的注意点2

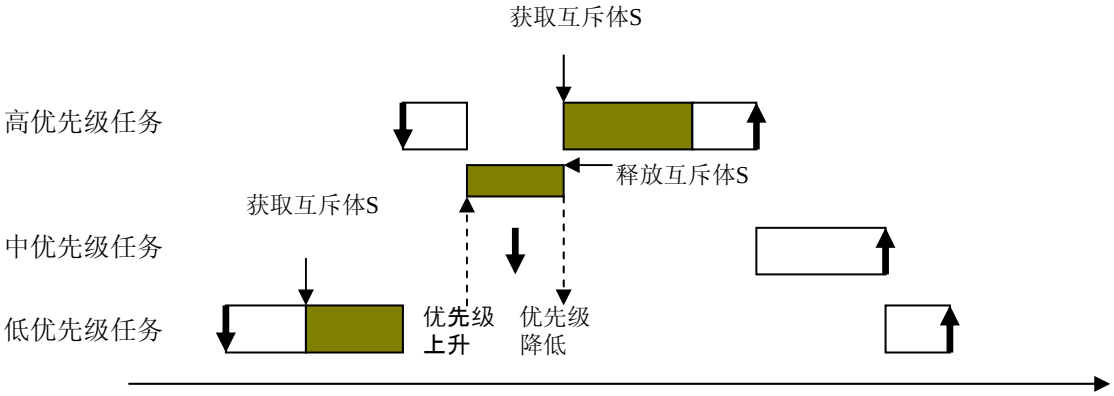
优先级倒置（Priority Inversion）

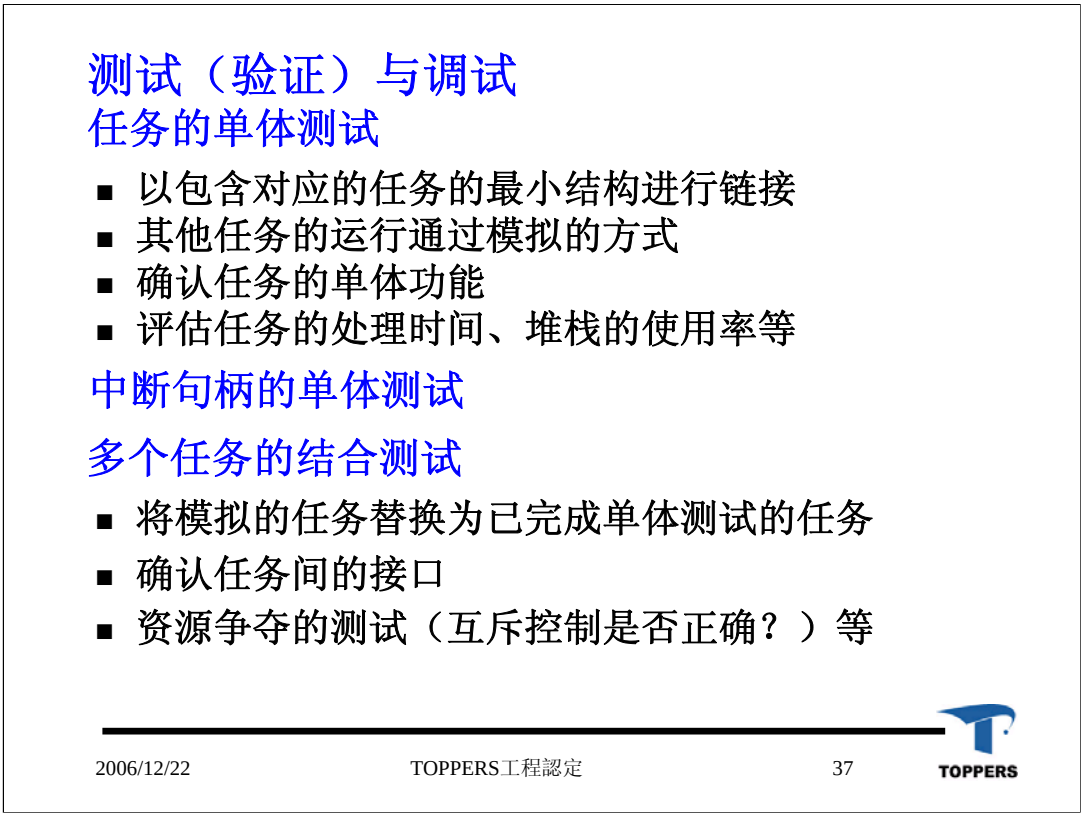
- 高优先级的任务和低优先级的任务资源共享，并通过信号量S实现对资源的排他控制时，低优先级的任务使高优先级的任务进入执行等待状态



2006/12/22 TOPPERS工程認定 36 TOPPERS

不使用信号量，而是通过设定了优先级继承协议的互斥体进行上述操作时，排他控制的运行是不一样的。如果设定优先级继承协议，那么在高优先级的任务进入等待状态时由低优先级的任务锁定的互斥体就会被释放，低优先级的任务的优先级会被提高到高优先级的任务的优先级，所以不需要等待中优先级的任务执行结束。





・测试（验证）与调试

任务的单体测试

进行单体测试时，对应对象任务的功能、输入会有什么样的输出？会进行怎样的运行？等等，总结这些项目。添加能够确认对象任务程序的输入、输出等的程序，并根据各项确认任务的运行、输出。任务的处理时间、堆栈的使用率等也要事先测定。

中断句柄的单体测试

中断句柄与任务的单体测试一样，在确定了要确认的项目后在模拟环境下进行单体测试。

多个任务的结合测试

在结合测试中，比较普遍的步骤是从重要的功能开始确认，并一边追加附加的功能一边验证整体的运行是否正确。所以，要先创建实现重要功能的任务进行验证，而具有附加功能的任务等由有接口的伪程序代替执行，或不进行结合，先完成创建进行结合功能测试。此时，也要进行任务间接口的确认、资源争夺的测试等。具有附加功能的任务测试结束后，与正在创建的伪程序进行替换，或追加接口来进行结合。进行增加了新功能的结合测试，在修改有问题的部分的同时，再反复进行符合最终式样的创建和结合测试。

调试环境的支持RTOS的功能

- 支持RTOS的调试环境（软件调试器、ICE等）具有以下功能：
  - ▼读取对象状态
    显示任务、信号量等的状态
  - ▼取出任务的语境
    显示处于等待状态的任务的寄存器等
  - ▼发系统调用
    通过系统调用来改变对象状态
  - ▼识别任务的断点
    在指定的任务执行到指定的地址时，只停止相应任务的功能等
  - ▼获取并显示RTOS的运行记录
    记录任务切换等信息，然后显示

在市面上出售的实时OS中，好像多数都提供了支持实时OS的调试环境。在该教材中，作为辅助内容还提供了任务监视器。下面结合其内容进行说明。

1）读取对象状态

只有任务可以参照：DISPLAY TASK命令

2）取出任务的语境

寄存器的参照：DISPLAY REGISTER命令

3）发系统调用

act_tsk： TASK ACTIVATE命令

ter_tsk： TASK TERMINATE命令

sus_tsk： TASK SUSPEND命令

rsm_tsk：TASK RESUME命令

chg_pri： TASK PRIORITY命令

4）识别任务的断点

不支持

5）获取并显示RTOS的运行记录

不支持

实习课题的说明

1. 构建应用RTOS的实时系统
2. 介绍实习课题
3. 确认开发环境
4. 制作话题烧水壶1

下面介绍一下实习课题：

・介绍实习中设计、制作的“话题烧水壶（GOMA_1015型）”

・运行环境是使用了netDevice的模拟器环境，下面对此环境以及限制事项进行说明。

SESSAME：话题烧水壶的实装
由于使用H8/3069，所以硬件式样有变更

- 话题烧水壶令需求式样书：根据第3版制作样机。
- 关于样机壶的制作，不是制作专用的LED，而是制作16个字符*2行的LCD模块来显示温度、水位、定时器。

・话题烧水壶（CDMA_1015型）需求式样书第3版
GOMA_1015型是在SESSAME的教材中进行制作的。

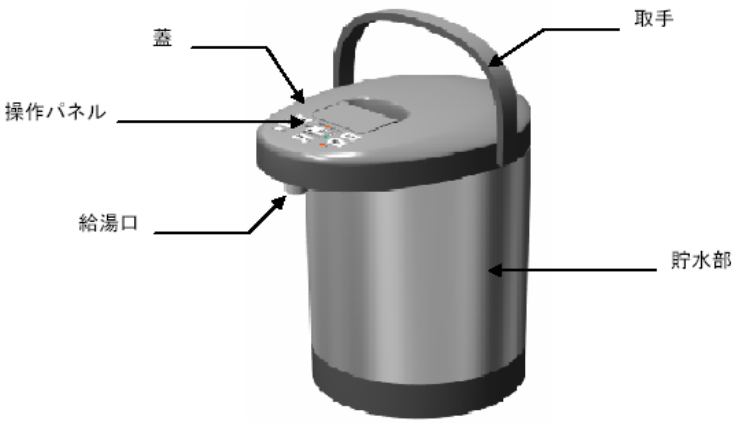
1、功能要点

本次设计的烧水壶是给用户提供下述功能的家电产品。

・烧开并保温壶内的水的功能
・提供壶内的水的功能
・到了用户指定的时间就鸣蜂鸣器进行通知的厨房定时功能

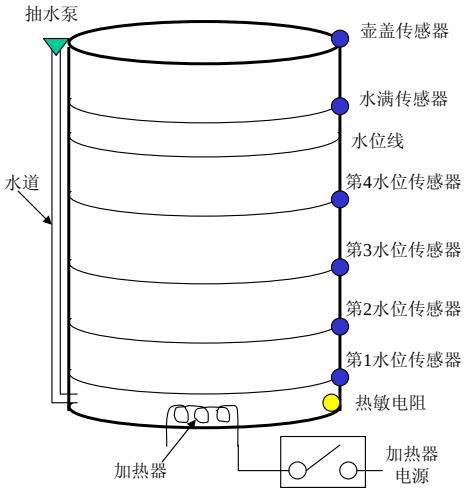
2、壶的外观与名称

此处的部位名称、功能都是一般性的内容，所以省略了说明。



硬件部分的结构
介绍一下贮水部的结构

- 贮水部的结构如右图所示
- 在加热器和电源供给部分有开关
- 精确的温度用加热器开关进行控制
- 检测温度用的热敏电阻装在贮水部的底部，通过AD转换器转换成温度



2006/12/22 TOPPERS工程認定 41 TOPPERS

各部位的术语说明：

1）水满传感器
检测水位是否超出该壶的允许上限。此传感器处于ON状态时说明水位已超出允许上限。此时必须切断加热器。

2）第n水位传感器
检测水位。各传感器处于ON状态时说明水位超出此位置。

3）壶盖传感器
检测壶盖是否开着。壶盖关着时为ON状态。

4）热敏电阻
检测壶内的水温。温度是通过电压输出。使用A/D转换器将此电压转换为温度的数值。

5）加热器
加热壶内的水。

6）加热器电源
给加热器提供电力。一般是ON状态，但在加热器发生异常时关闭并切断电力。

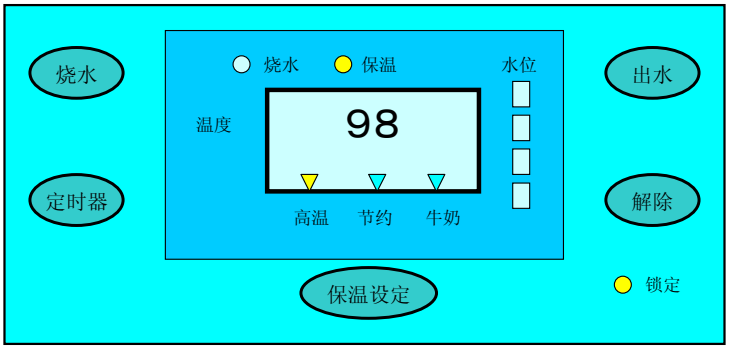
7）水位线
使用户知道此壶可以装入的水量上限的标记。在水满传感器下面一点的位置。

8）抽水泵
向上抽出壶内的水，从出水口排出去。

9）水道
抽水泵抽水时的通道。

操作面板1
样机壶上的面板

- 下图是样机壶上的操作面板
- 显示温度和水位的部分只是虚设，实际是在H8/3069F电路板的LCD上显示



2006/12/22 TOPPERS工程認定 42 TOPPERS

样机壶的面板由5个按钮和7个灯构成。

・按钮开关

1）定时按钮

按下这个钮就会启动定时器，每按一下都会加1分钟。

2）保温设定按钮

按下这个钮后，可以将保温模式设定为高温（98度保温）、节约（90度保温）或牛奶（60度保温）模式。每按1下，都会在高温→节约→牛奶→高温之间进行模式切换。

3）解除按钮

进行出水口的锁定与解除。在锁定状态下，即使按下出水钮也不会出水。

如果在锁定状态下按下此钮就会解除锁定，如果在解除状态下按下此钮就会锁定出水口。

正在出水时不能进行锁定。

4）出水按钮

如果按下这个钮，就会启动抽水泵，从出水口排出水。按着时出水，如果手离开按钮使按钮抬起就会停止出水。

5）烧水按钮

按下这个钮后就会烧壶中的水并去除漂白粉。如果正在烧水时按下这个钮，就会中断烧水，并进入保温状态。每按1下都会在烧水→保温→烧水之间切换。

・灯

1）温度・模式灯

“高温”“节约”“牛奶”中，哪一个上方的▼灯亮，就表示为哪种模式。

2）锁定灯

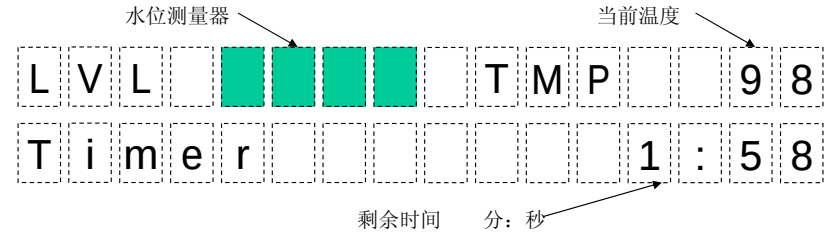
表示出水口是否被锁定。出水口锁定时灯亮。

3）烧水、保温灯

烧水灯在烧水时亮、保温时熄灭。保温灯正好相反。

操作面板2
H8-3069F板上的LCD

- 16个字符*2行的LCD模块的第1行显示水位高度(4个高度)、温度
- 第2行显示厨房定时器剩余的“分：秒”
- 定时器没有启动时第2行不显示内容



・LCD的显示

1）水位测量器

显示壶内的水位。

水位测量器和各水位传感器之间的关系如下所示：

第8个字符→□→对应第4水位传感器

第7个字符→□→对应第3水位传感器

第6个字符→■→对应第2水位传感器

第5个字符→■→对应第1水位传感器

2）当前温度

显示当前温度。

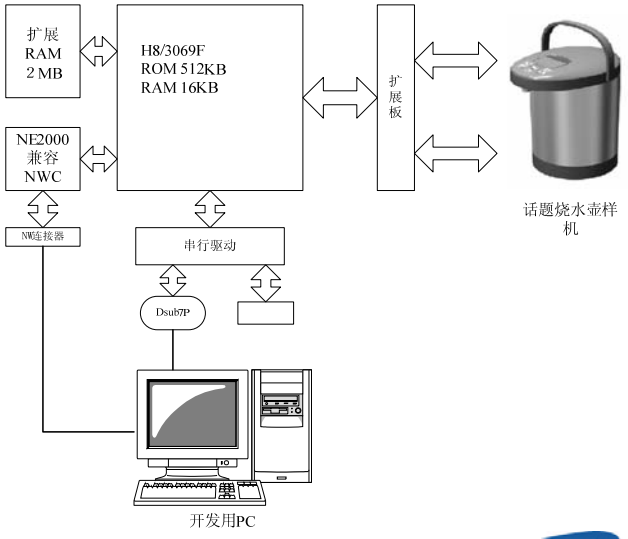
3）显示定时器剩余时间的窗口

用“分：秒”来显示到规定的时间还有多长时间

定时器没有启动时第2行不显示任何内容。

H8/3069F概要1
基本构成

- 将H8/300H作为CPU 核心的单片机
- 连接了扩展RAM和网络控制器



2006/12/22 TOPPERS工程認定 44 TOPPERS

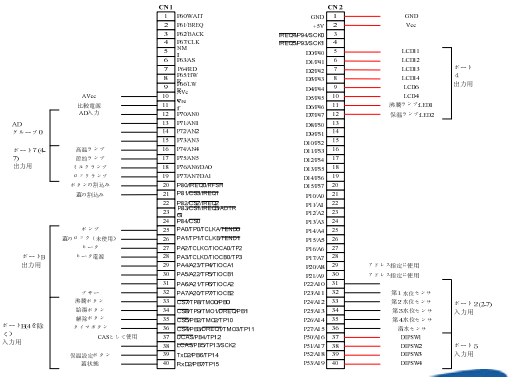
・ H8/3069F是将H8/300H作为CPU Core的单片机。

1）闪存ROM 512KB
2）RAM 16KB
3）中断控制器
4）16位集成定时器部件
5）DMA控制器
6）更新控制器
7）监视定时器
8）串行通信接口X2
9）A/D转换器
10）D/A转换器

・扩展RAM 2MB
・安装NE2000兼容网络控制器、RTS8019AS
・先向闪存ROM中写入TOPPERS/JSP程序可以下载的ROM监视器。监视器使用“开发用PC”的串口连接到板上。“开发用PC”可以用ETHER NET进行通信。关于“话题烧水壶样机”，使用扩展板上连接的2个通道CN1和CN2，通过各端口连接到样机设备上。
（实际上是通过网络与话题烧水壶模拟器进行连接的）

与样机壶的连接
与H8/3069电路板的连接

- 对扩展板的CN1和CN2的空闲端口分配各设备的接口
- 端口4、5直接使用现有的扩展板



TOPPERS

・扩展板上的2个通道CN1和CN2与“话题烧水壶样机”的各设备之间的连接如下所示：

1）作为中断使用CN1的20、21号的IREQ0和IREQ1。

2）为了用A/D转换器获取温度，将组0的AN0连接到传感器上，并从Vref获取取样电压。

3） LCD直接使用扩展板附带的。LCD的控制使用端口4作为输出。

4）抽水泵、加热器、蜂鸣器的控制使用端口A作为输出。

5）各按钮、壶盖的状态获取使用端口B作为输入。

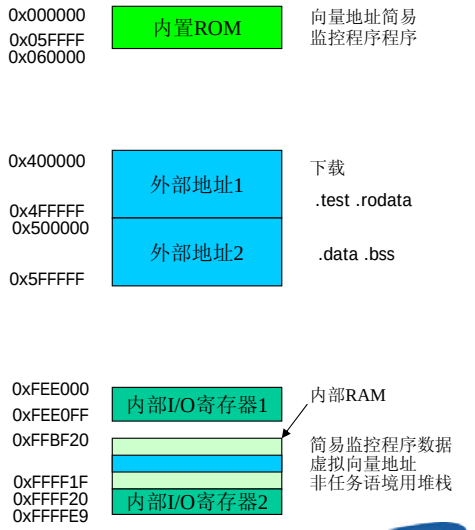
6）灯的显示使用端口4和7作为输出。

7）端口2作为输入连接到各传感器上。

8）扩展板上连接了DIP_SW的端口5直接使用。

H8/3069F概要2
CPU运行模式与地址空间

- 本次开发的**H8/3069F**在7个运行模式中的模式5下运行
- **ROM**上已写入简易监控程序，将包含**JSP**的程序下载到右图中淡蓝色的部分就可以运行



H8/3069F具有外部RAM、LAN驱动器，为了使用这些必须在运行模式的模式5下运行。而且几个端口将不能使用。

在本实习中，H8/3069F的闪存ROM上已写入苫小牧高等专科的阿部先生创建的ROM监视器h8mon。此监视器可以下载在H8用TOPPERS/JSP的调试模式下创建的摩托罗拉S格式的执行程序并运行。下载的程序要写入到上述存储布局的淡蓝色部分。其中，vector部分的中断向量也要从OXFFCOOO写入到OXFFCOFF，虚拟实现可以作为中断向量运行，所以不用改写闪存ROM就可以运行包含TOPPERS/JSP的程序。

闪存ROM有保证写入次数，如果为了调试多次向闪存ROM进行写入，那么有时会无法重新写入。

H8/3069F概要 3
I/O端口

- 在运行模式5下运行时，有几个端口不能使用
- H8/3069F的扩展板通过端口4、5进行连接
- 样机壶使用端口2、4、7、A、B进行连接

端口	概要	端子	连接
2	8位的输入输出端口，输入Pull Up MOS内置	$P1_7$-$P1_0$ /A_7-A_0	输入、与水位传感器相连接
4	8位的输入输出端口，输入Pull Up MOS内置	$P4_7$-$P4_0$ /D_7-D_0	输出、与LCD和两盏灯相连接
5	4位的输入输出端口，输入Pull Up MOS内置	$P5_3$-$P5_0$ /A_{19}-A_{16}	输入、与DIP-SW相连接
7	8位的输入输出端口	$P7_7$/AN_7 $P7_6$/AN_6 $P7_5$-$P7_0$	输出、其他灯
A	8位的输入输出端口	PA_7-PA_0	输出、控制用
B	8位的输入输出端口	PB_7-PB_0	输入、按钮和壶盖

TOPPERS

为了运行模式5或LAN驱动，使用H8/3069F的下述端口。所以，为了与样机壶进行连接，必须使用这些端口以外的端口。

1）端口1

2）端口2的BIT0、1、2

3）端口3

4）端口5的BIT4到7（无）

5）端口6

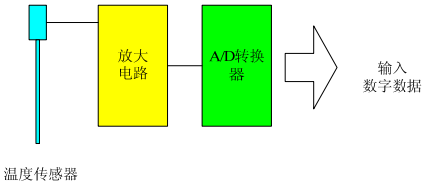
6）端口8的BIT2到3

7）端口9

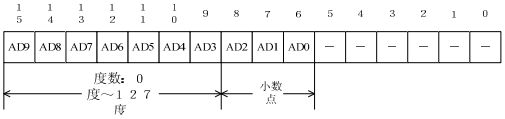
8）端口B的BIT4

H8/3069F概要4
A/D转换器

- 壶的温度传感器值增加，进入A/D转换器中
- 模拟输入将AN0的电压在ADDRA寄存器中转换为10位的数字值。此时，后3位变成小数值



A/D数据寄存器：ADDRA



H8/3069F中有10位8通道的A/D转换器。为了获取温度，只使用AN0的模拟输入。输入值进行A/D转换后存储到A/D数据寄存器A（ADDRA）中。如果可以读入此寄存器，则获取温度。在ADDRA中有效的10位中，前7位是度数，后3位是小数点值。

但是，实际上将传感器的值数字化后的值并不能直接作为温度的比例值进行获取。所以，要使用表等进行转换。在本实习中是使用模拟器，所以要创建能从传感器获取正确温度的程序。

连接端口表1

输出端口

- 对应于样机壶的硬件写入部分的端口连接布局如下所示：

端口	位	控制	端口	位	控制
4	0-3	LCD数据位	A	0	抽水泵：1为ON
	4	LCDRS信号		1	壶盖的锁定（未使用）
	5	LCD启动信号		2	加热器：1为ON
	6	沸腾灯：1为ON		3	加热电源：1为ON
	7	保温灯：1为ON		7	蜂鸣器：1为ON
7	4	高温灯：1为ON			
	5	节约灯：1为ON			
	6	牛奶灯：1为ON			
	7	锁定灯：1为ON			

为了与样机壶进行连接，使用了端口4、端口7、端口A三个端口。这些都是用于数据输出。

端口4的0到5位用于控制LCD。端口4连接在板上的2个LED和样机壶的烧水灯、保温灯上。端口7连接在剩下的4个灯上。

关于各灯，如果设置1则灯亮，如果设置0则熄灭。

端口A控制设备。

BIT0控制抽水泵，设置为0时停止，设置为1时抽水泵供水。

BIT1是壶盖的锁定，在本实习中不使用。

BIT2控制加热器，设置为0时关闭，设置为1时启动。

加热器电源关闭时加热器不会运行。

BIT3是控制加热器的电源，为0时切断电源，为1时打开电源。

BIT7是控制蜂鸣器，为1时打开蜂鸣器，为0时关闭蜂鸣器。

连接端口表2

输入端口表

- 对应于样机壶的硬件读入部分的端口连接布置如下所示：

端口	位	控制	端口	位	控制
2	3	第 1 水位传感器：0为ON	5	3	DIP-SW4：0为ON
	4	第2水位传感器：0为ON	B	0	烧水按钮：0为ON
	5	第3水位传感器：0为ON		1	出水按钮：0为ON
	6	第4水位传感器：0为ON		2	解除按钮：0为ON
	7	水满传感器：0为ON		3	定时器按钮：0为ON
5	0	DIP-SW 1 ：0为ON			
	1	DIP-SW2：0为ON		6	保温设定按钮：0为ON
	2	DIP-SW3：0为ON		7	壶盖：0为关闭

端口5用于DIP_SW的输入。DIP_SW1用于控制是否使用监视器。打开时插上电源启动监视器。关闭时不启动监视器，激活任务ID1的任务。

端口2、端口B作为样机壶的输入数据进行使用。端口2的BIT3到7对应于水位传感器，传感器打开时端口的值变为0。

端口B对应于按钮和壶盖的状态。

BIT0在烧水按钮打开时为0，关闭时为1。

BIT1在出水按钮打开时为0，关闭时为1。

BIT2在解除按钮打开时为0，关闭时为1。

BIT3在定时器按钮打开时为0，关闭时为1。

BIT4在保温设定按钮打开时为0，关闭时为1。

BIT7在壶盖打开状态下为1，关闭状态下为0。

中断设定
样机壶中使用2个

- **H8/309F的TOPPERS/JSP和TINET中使用了8个中断**

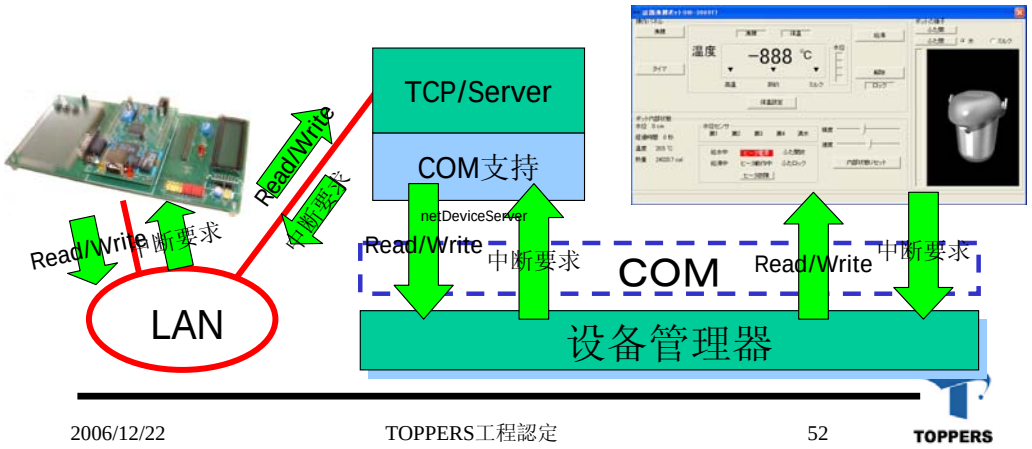
编号	标签	中断内容
13	IREQ1	通过壶的按钮的ON或OFF进行中断
14	IREQ2	通过壶盖的开与关进行中断
17	IREQ5	网络驱动IF_ED使用的中断
24	IMIA1	TOPPERS/JSP的系统定时器中断
52	ERI0	串口0的错误中断
53	RXI0	串口0的数据接收中断
54	TXI0	串口0的数据发送中断
56	ERI1	串口1的错误中断
57	RXI1	串口1的数据接收中断
58	TXI1	串口1的数据发送中断

在样机壶中，对应了IREQ1和IREQ2两个中断。按下或抬起5个按钮中的任何一个时，发生IREQ1中断。打开或关上壶盖时，发生IREQ2中断。

其他的中断是H8/3069F板上对应的中断。

介绍实习环境
介绍一下应用netDeviceServer的开发

- 开发使用**H8/3069F**
- **H8/3069F**电路板上就像安装了话题烧水壶用的扩展硬件一样，可以进行控制



2006/12/22 TOPPERS工程認定 52 TOPPERS

・netDevice

通过COM连接Window版TOPPERS/JSP用的设备管理器和网络服务器（netDeviceServer）。使用在H8/3069F的TOPPERS/JSP下运行的TINET连接到此服务器上，这样就可以将VB等创建的模拟器作为H8/3069F的扩展硬件来进行使用。在实时系统开发实习中，使用该系统将话题烧水壶模拟器作为H8/3069F的扩展硬件进行使用，实现话题烧水壶系统的开发。

netDeviceServer作为网络·服务器进行启动，但由于没有进行多线程化，所以客户端只有一个。

开发环境的说明

1. 构建应用RTOS的实时系统
2. 介绍实习课题
3. 确认开发环境
4. 制作话题烧水壶1

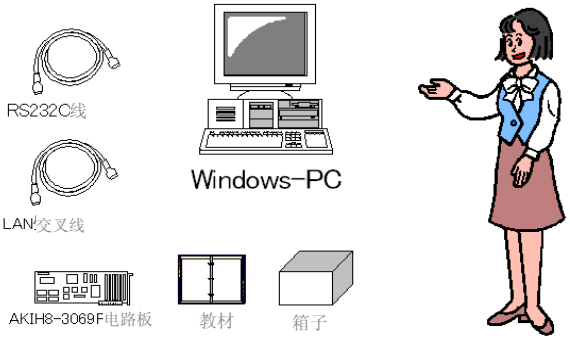
介绍话题烧水壶样机版的开发环境。

使用AKIH8/3069F的带按钮开关的扩展板的模拟环境进行解说。只要有该模拟环境，即使没有扩展板也能够实现碗面定时器的调试。

确认开发环境
机器是否连接

- **PC**
- **电路板**(AKIH8/3069F)
- **RS232C线**
- **LAN交叉线**
- **教材**
- **箱子**

电源	1 个
CD-ROM	1张
等	



2006/12/22 TOPPERS工程認定 54 TOPPERS

・PC　需要以下实装
　已安装PizzaFactory2或Cygwin
　编辑器（TeraPat）
　通信软件（Tera Term）
・AKIH8/3069F板　已装配完毕
　ROM监视器已写入到闪存ROM中
　（安装了普通软件的板已写入完毕）
・RS232C线　PC与板之间进行通信
　实现监视器的显示、程序的下载等
・ＬＡＮ交叉线　板与PC上的模拟器之间进行通信
・教材　该教材
・电源　给板提供电源
・ CD_ROM　写入了用PizzaFactory2、JSP进行开发的方法
・箱子　装AKIH8/3069F板、电缆、CD_ROM的箱子

确认开发环境的连接
与第1、2天的环境相同

- 通过LAN交叉线连接PC的LAN适配器与AKIH8/3069F电路板的LAN适配器
- 通过RS232C线连接PC的RS232C连接器与AKIH8/3069F电路板的RS232C连接器
- 启动Tera Term，打开电路板的电源，确认监视器的提示符
- 打开DOS窗口，通过ipconfig命令确认LAN的IP地址

是否是IP=192.168.11.6？

・LAN电缆

通过LAN连接板与PC。PC是服务器端，板是客户端，以此种方式进行设备的访问信息的通信。板上的程序通过通信的信息来控制PC上的扩展板模拟器。由于是交叉线，所以是1对1的通信。

・RS232C线

用RS232C串行的步骤连接板与PC。PC端通过通信软件TeraTerm实现板的监视器程序（ROM用和任务用）的控制以及状态的显示。使用ROM监视器与TeraTerm的通信功能来下载创建的程序。

・IP地址

在PC上确认IP的设定。连接电缆并接通电源，如果已设定IP，则可以通过ipconfig命令来确认设定的IP。没有设定时请通过“控制面板”的“网络连接”进行设定。netDevide通过IP地址指定服务器端的PC。

构建确认环境
运行netDevice版的碗面定时器

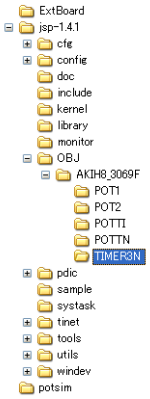
- 使用Netdevice+AKIH8/3069电路板制作“碗面定时器”
- 式样如下

接通电源后，为了确认运行，以1秒钟间隔闪烁LED3。

打开开关5启动定时器，关闭则停止定时器，初始超时时间为1分钟

在已启动定时器的状态下，打开开关4将超时时间延长1分钟。打开两次超时时间就变为3分钟。

为了确认定时器的运行，每10秒钟闪烁LED2一次，超时时闪烁30秒后停止。



普通的定时器由LCD的时间显示与蜂鸣器构成。在AKIH8/3069F mini板上只有LCD和LED设备，所以要用模拟器创建LED和按钮开关，来实现时间的通知和超时的通知。

关于时间的通知，每隔1秒钟闪烁LED3一次来进行通知。而定时器的通知是由LED2实现的。如果启动定时器，则每10秒钟闪烁一次。超时的通知是由LED2闪烁15秒钟来实现的。

开始定时器是打开开关5，而停止定时器是关闭开关5。开始时会设定1分钟的超时，每次打开开关4都会将超时延长1分钟。

netDevice版碗面定时器1
重新构建Kernel库

- 为了对应任务监视器，要使用编辑器追加_DMON定义，并重新创建Kernel库

  COPTS := $(COPTS) –DMON

```
$ cd ./OBJ/AKIH8_3069F
$ mkdir libkernel
$ cd libkernel
$ ../../../configure -C h8 -S akih8_3069f
$ make depend
$ make libkernel.a
```

2006/12/22 TOPPERS工程認定 57 TOPPERS

为了使设定任务监视器的任务时间的功能有效，要重新构建Kernel库。下面再介绍一下构建的方法：

1）在./OBJ/AKIH8_3069F中创建libkernel目录。

2）移到此目录下，用configure命令构建sample1的生成环境。

./../../configure_C h8_S akin8_3069f （构建h8/akih8_3069f用的sample1生成环境）

3）在/OBJ/AKIH_3069F/libkernel中的Makefile的第101行的COPTS选项中追加_DMON选项进行构建。

COPTS：＝$(COPTS)

COPTS := $(COPTS) –DMON

如下进行构建。

$ make depend

$ make libkernel.a

-D选项是在Kernel中追加任务的监视器链接功能的设定。

Netdevice版碗面定时器2
创建 （Build）

- 进入到开发环境TIMER3N下
- 按以下步骤创建jsp.srec

```
$ cd ../TIMER3N
$ ls
akih8_device.c     Makefile     timer3.c     timer3.h  tinet_timer3.cfg
Akih8_lcddevice.c route_cfg.c timer3.cfg tinet_app_config.h
$ make depend
$ make
```


参考用的碗面定时器程序保存在OBJ/AKIH8目录下。通过输入PizzaFactory2（Cygwin）的命令移到此目录下，再通过下述命令进行创建。

$ make depend

$ make

创建后生成jsp.srec。将其下载，并在板上运行。

该网络设备对应的碗面定时器程序与计算机上的服务器建立连接，在模拟器上进行通信。教材中计算机的IP地址是设定为192.168.11.6，但板上的IP地址的子网掩码是通用的，必须设定不同的IP地址。IP地址在tinet_app_conifg.h中有设定（在本教材中，IP地址：192.168.11.98，默认网关：192.168.1.1），为下述定义。

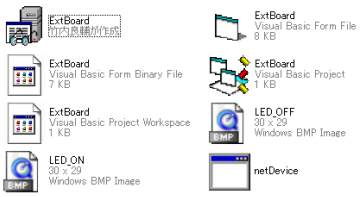
```
#define IPV4_ADDR_LOCAL          0xc0a80b62      /* 192.168.11.98 */
#define IPV4_ADDR_LOCAL_MASK     0xffffff00      /* 255.255.255.0 */
#define IPV4_ADDR_DEFAULT_GW     0xc0a80b01      /* 192.168.11.1 */
```

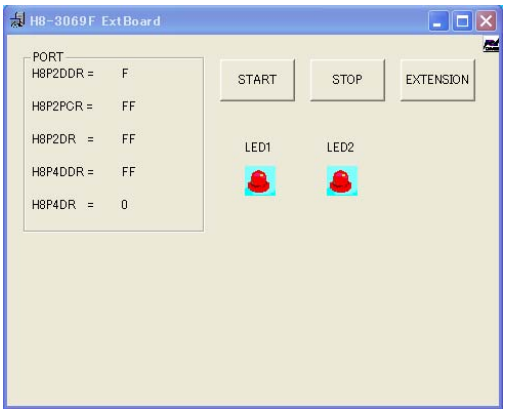
与PC的IP地址的子网对应的IP地址与教材中的不同时，需要对照板的IP地址重新设定。

netDevice版碗面定时器3
启动模拟器

- 按“开始→话题烧水壶→扩展板”启动“ExBbard”模拟器

ExtBoard用来模拟AKIH8/3069F的扩展板上添加了3个开关的电路板。即使没有扩展板也可以运行。





2006/12/22 TOPPERS工程認定 59 TOPPERS

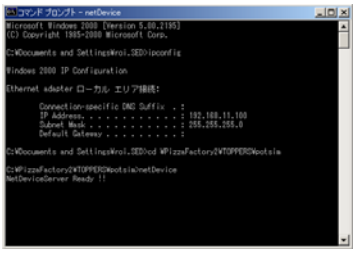
・AKIH8/3069F版的扩展板模拟器

是模拟由VB6创建的AKIH8/3069F的扩展板的程序。此程序与Windows版的TOPPERS/JSP是共用的，所以只要创建了Windows版的驱动，即使没有板也可以运行。从板上访问的端口基本有下述5个。

H8P2DDR	输出	端口2（H8P2DDR）访问模式设定
H8P2PCR	输出	端口2（H8P2DDR）的Pullup设定
H8P2DR	输入	按钮的输入端口
H8P2DDR	输出	端口4（H8P2DDR）访问模式设定
H8P4DR	输出	与LED的设定端口、AKIH8/3069F的扩展板的LED是相同的端口设定，在没有连接到netDevice时就可以控制扩展板的LED。

实习环境的启动步骤1
Device管理器的确认与netDevice的启动

- 确认TOPPERS模拟器用DeviceManager的安装（确认工具栏的图标）
- 在DOS窗口下移到netDevice目录，启动netDevice（服务器）
- 直接双击也可以启动



设备驱动图标

启动电路板时，请确认DIPSW 1 是否为ON

・实习环境

关于实习环境，需要先将TOPPERS/JSP Windows版的Device Manager安装到开发环境中。如果已经安装，启动H8版的话题烧水壶模拟器就会显示Device Manager图标，这样就可以进行确认了。

请先启动话题烧水壶与netDevice。至于启动的顺序，哪个在前都可以。在启动netDevice前，请从DOS窗口确认ipconfig命令的开发环境PC的IP地址。作为网络服务器启动netDevice，这样要想从板连接到话题烧水壶就必须指定服务器的IP地址。如果netDevice可以作为服务器正常启动，则会显示“netDeviceServer Ready！！”。

设备管理器的安装方法如下所示：

用了运行对应网络的模拟器，需要安装TOPPERS DevieManager与设备用Visual BASIC OCX控件。

因为是两个程序常驻型，所以创建的安装处的目录要无变更，请将ToppersDeviceManager中的下述文件拷贝到此目录下。

・devicemanager.exe

・devicemanagerps.dll

・device.dll

请从DOS窗口移到拷贝的目录并执行下述命令。

>devicemanager /REGSEVER(return)

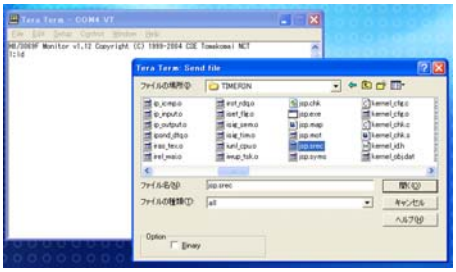
>regsvr32 devicemanagerps.dll(return)

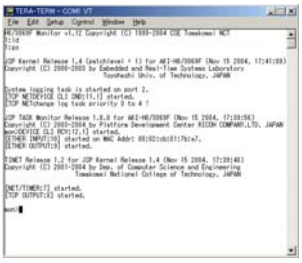
>regsvr32 devicedll(return)

关于设备管理器，请参照TOPPERS/JSP的文档“window.txt”。

实习环境的启动步骤2
程序的下载与监视器的启动

- 请打开DIP_SW1
- 在TeraTerm中输入ld命令后， jsp.srec 和TeraTerm重叠后开始下载
- 通过File→SendFile……然后选择jsp.srec也可以进行下载
- 下载后，用go命令运行任务监视器





・H8板的启动与2个监视器

在H8板上，由苫小牧高等专科创建的H8用ROM监视器已写入到闪存ROM中。为了能够很容易连接到netDevice，需要初级实装讲座中使用的任务监视器。ROM监视器和任务监视器使用H8板附带的RS232C端口，可以通过命令进行操作。在RS232连接的PC上，通过以下设定启动Tera Term。

Baud rate: 38400bps
Data: 8bits
Parity: none
Stop: 1 bit
Flow control: none

请确认RS232的连接和DIP_SW的打开，并插上H8板的电源。

Teral Term上显示以下内容，已启动ROM监视器。

“H8/3069F Minitor v1.2 Copyright@1999_2004 CSE Tomakomai NET
1：
”

在下载创建的程序时使用ld命令。输入命令后，通过File→Sendfile…指定SREC文件就会开始下载。下载结束后，再一次显示“1：”提示符。运行已下载的程序时使用go命令。通过go命令，TOPPERS/JSP→TINET→任务监视器启动。

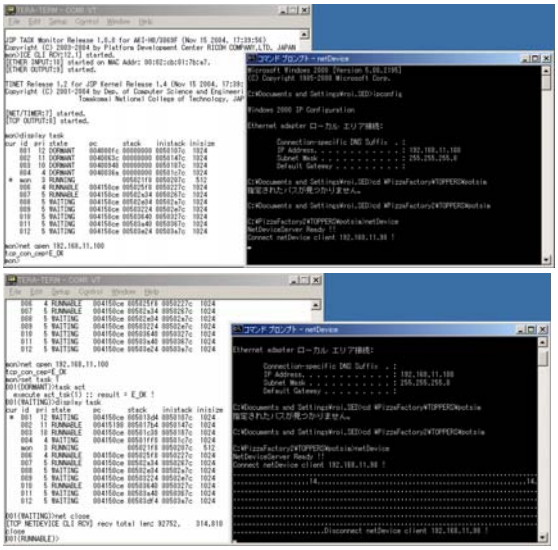
任务监视器是在H8的调试板的DIP-SW的第一个打开时启动，所以请确认DIP-SW。

通过任务监视器的提示符“mon>”在display task（return）中显示当前任务的状态。

DORMANT状态的任务是用户程序，其他任务是TINET用任务或监视器任务。

实习环境的启动步骤3
与netDevice进行连接以及解除连接

- 任务监视器启动后，通过net open<IP地址>连接到netDevice
- 通过net close命令解除与netDevice的连接
- 在非连接状态下将电路板复位，可以下载新的程序



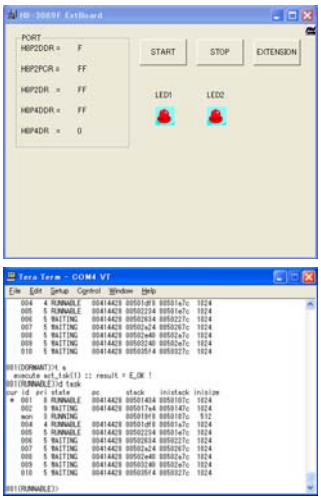
・与netDevice的连接

如果启动用户程序，就会从设备进行读取、写入。所以，最好从任务监视器启动用户任务。监视器启动时不会自动连接到netDevice。所以在启动时需要输入net open <IP地址>（return）的命令来实现与netDevice的连接。连接后显示“Connect netDevice client <IP地址>！”。解除连接时输入net close （return）命令来进行解除。此时会显示“Diconnect netDevice client<IP地址>！”。输入net close后会马上解除服务器的连接，但由于存在TINET的关闭超时时间，所以H8板要等待几分钟。如果重新下载程序，则不用等待超时，可以直接按复位键进行复位，然后从ROM监视器下载程序。不需要重启netDevice、话题烧水壶模拟器。

如果不连接netDevice直接开始用户任务，则程序会以H8的实板的端口为对象进行运行。这样就不会输入期望的数据，导致运行不正确，但程序不会失控。

系统构建方面的限制
由网络代替设备带来的限制

- 非任务语境下的网络上的端口无法访问
  这是实时系统设计方面一个很大的限制
- 因为TCP/IP、客户端应用程序等要被多个任务运行，所以比实际的系统的系统开销大些



在碗面定时器系统中，网络上使用6个任务

・非任务语境下的端口访问

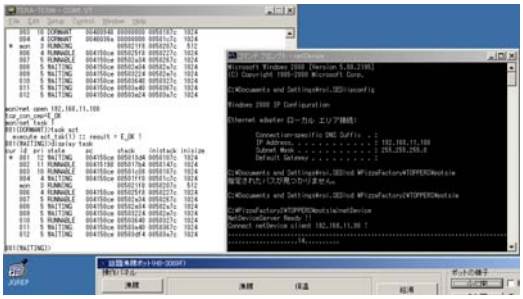
在Windows版的TOPPERS/JSP中，任务语境与非任务语境都要作为Windows的线程来运行，所以不管是哪一个都可以通过COM进行进程间的通信，实现设备的访问。但在使用netDevice的系统中，设备的访问是使用TCP/IP的协议栈来访问端口的，所以无法从非任务语境访问端口。如果从非任务语境对netDevice进行访问，就会出现“错误记录”，并对实设备进行访问。这样就无法正确访问模拟器。

在实时OS中，中断等非任务语境比任务优先执行。所以，即使在中断中对网络进行访问也无法正常访问端口，因为中断结束后会执行网络用的任务。而且，即使在任务中已模拟运行了中断程序，整个系统也会变得很复杂，不适合用于培训。此限制是系统设计方面一个很大的限制。

实习程序的运行
程序的运行与出错时的对应

- 与netDevice进行连接后，使用set task〈ID〉命令进入启动任务，并用task activate命令启动任务。将模拟器作为设备运行程序

发生无法预测的错误时，请结束所有程序



・用户任务的执行

关于话题烧水壶的Sample开发环境POT1和POT2，是由任务1来启动其他任务。所以，如果使用监视器的task命令启动任务，其他的用户任务就会启动。在用户任务正在执行时，即使使用net close （return）命令切断与netDevice的连接，也不会出现问题。

如果为了重新下载程序在与netDevide进行连接时通过H8板上的复位键将板复位，netDevice就会无法检测复位。此时，可以如下进行对应。

1）重启netDevide

2）取下LAN电缆

3）进行下载，用go命令启动任务监视器并重新进行net open。

不过，虽然3）可以重启，但不能保证。

・其他

重新运行时，请用话题烧水壶的“内部状态复位”按钮将模拟器复位。

另外，模拟器、netDevice的状态异常时请重启各程序。

在netDevice的限制事项中，在非任务语境下无法访问netDevice上的端口。

所以，如果在中断后用轮询法从任务读取按钮等，有时会发生延迟且读取错误。请慢慢按下模拟器的按钮等。

制作话题烧水壶1

1. 构建应用RTOS的实时系统
2. 介绍实习课题
3. 介绍开发环境
4. 制作话题烧水壶1

制作话题烧水壶。

话题烧水壶1
读式样书了吗？

- “话题烧水壶（GOMA_1015型）”式样书可以从SESSAME的主页下载

  http：//www.sessame.jp
- 本教材是以第3版为对象
- 在课前学习中读式样书了吗？

话题烧水壶2
目录式样

- 简要说明一下“话题烧水壶（GOMA_1015型）样机”的目录使用

壶内的水的烧开・保温功能
进行烧水设定并关上壶盖后，烧水3分钟

壶内的水的3种保温模式
烧开后可以进行高温（98度）、节约（90度）、牛奶（60度）保温

出水的锁定功能
按下出水按钮就能出水，按下锁定按钮则出水锁定，这样很安全

水温与水位的显示
水温和水量是一直显示的，所以很容易确认

在指定的时间鸣蜂鸣器的厨房定时器功能
可以每1分钟设定一次时间，通过蜂鸣器进行通知

根据“话题烧水壶（GOMA_1015型）”需求式样书制作的电子壶的4个特征已进行目录化。
话题烧水壶样机的目录式样与一般电子壶基本相同。

1、烧开·保温壶内的水的功能
按下烧水按钮并关上壶盖后自动切换到烧水模式，保持烧水状态3分钟并去除漂白粉。

2、壶内的水的3种保温模式
从高温（98度）、节约（90度）、牛奶（60度）中选择保温温度。正在烧水时，如果按下保温按钮则切换到保温模式。

3、出水的锁定功能
按下出水按钮，抽水泵自动进行供水。出水具有锁定功能，如果已进行锁定，即使小孩误按下出水按钮也不会出热水，所以请放心。

4、在指定的时间鸣蜂鸣器的厨房定时器功能
因为具有厨房定时器功能，所以即使没有“碗面定时器”，也可以马上知道方便面做好的时间。

话题烧水壶的结构化分析1
事件列表（除温度控制以外）

事件	stimulus	动作	响应
启动定时器	・时间	（1）启动定时器 （2）没有剩余时间时通知用户	（1）启动蜂鸣器并显示剩余时间 （2）启动蜂鸣器并显示剩余时间
追加定时器时间	・时间	向当前剩余时间追加指定的时间	・启动蜂鸣器
保温模式显示	・保温模式	・设定保温模式 ・变更温度控制	・启动蜂鸣器 ・模式显示
出水口的锁定	（无）	・锁定出水口	・启动蜂鸣器 ・锁定灯亮
出水口锁定解除	（无）	・解除出水口的锁定	・启动蜂鸣器 ・锁定灯熄灭
开始出水	（无）	・从壶内出水	・从出水口出水（启动抽水泵）
出水结束	（无）	・停止壶内出水	・停止从出水口出水（关闭抽水泵）
水位的变化	・水位	・显示	・更新水位显示

TOPPERS

事件列表（除温度控制以外）

・具有状态的功能（4个）

定时器剩余时间的管理功能

蜂鸣器的打开时间：按下按钮1秒钟，超时时3秒、出错时30秒

水位的监视时间：200MS

按钮与壶盖的轮询时间：50MS

・定时器处理

按下定时器按钮：启动了定时器、剩余时间追加60秒：1秒蜂鸣器

另外，60秒启动定时器：1秒蜂鸣器

定时器剩余时间结束：3秒蜂鸣器

启动了定时器时LCD上显示剩余时间

・出水处理

按下出水按钮：除以下状态以外，打开抽水泵

已锁定时、壶盖开着时

・保温模式处理

按下保温按钮：1秒蜂鸣器、每次按下都会以高温→节约→牛奶→高温…的顺序切换模式

显示对应的灯，如果正在烧水则切换到保湿模式

话题烧水壶的结构化分析2

事件列表（温度的控制关系）

事件	stimulus	动作	响应
开始烧水	（无）	（1）烧水 （2）烧水结束后进入保温状态	（1）・启动烧水灯 ・关闭保温灯 ・启动加热器 （2）・关闭烧水灯 ・启动保温灯
烧水中断	（无）	・进入保温状态	・关闭烧水灯 ・启动烧水保温灯
水满	・水位	・停止温度控制	・关闭烧水保温灯 ・关闭加热器
无水	・水位	・停止温度控制	・关闭烧水保温灯 ・关闭加热器
盖上壶盖	（无）	・如果可以控制温度就开始烧水	・启动烧水灯 ・关闭保温灯 ・关闭加热器
打开壶盖	（无）	・停止温度控制	・关闭烧水保温灯 ・关闭加热器
温度异常	温度异常检测	・停止温度控制 ・鸣蜂鸣器进行警告	・启动蜂鸣器 ・关闭加热器

2006/12/22 TOPPERS工程認定 69 TOPPERS

事件列表（温度控制）

・具有状态的功能（2个）

根据热敏电阻的检测、目标温度来打开或关闭加热器：100MS周期

温度错误状态的检测（高温异常、温度不上升、出错）：100MS周期

・温度控制

启动时插上加热器电源

壶盖关上时、按下烧水按钮时，切换到烧水模式，烧水3分钟

然后切换到保温模式

以下情况要关闭加热器

壶盖开着、水位为空、水满

高温异常、温度不上升、出现异常时，关闭加热器并切断加热器电源。

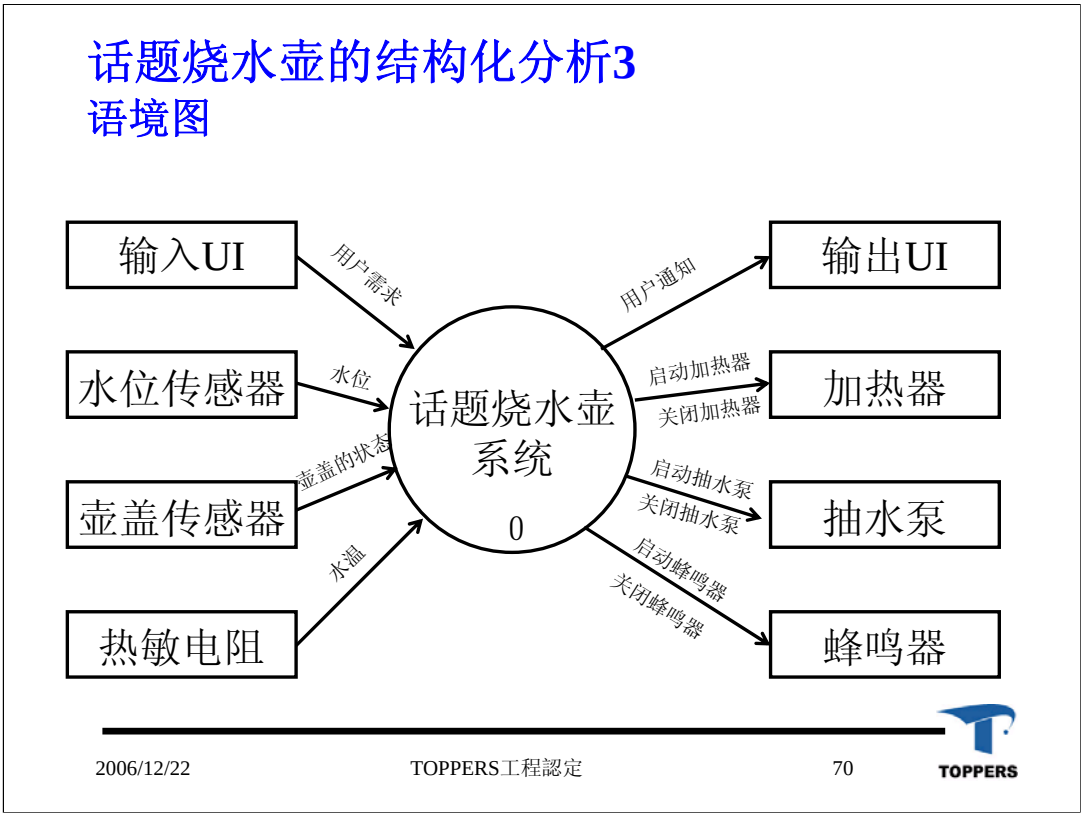
・温度错误检测

温度不上升、异常：在低于目标温度5度以上的状态下，水温在60秒内不发生变化时

高温异常：水温超过110度

・开发中的注意事项

netDevice的通信控制约占50%的CPU负荷。如果周期任务的数量超过4个，就会出现超过最后时间的情况。为了保证实时性，要限制任务的数量。



制作话题烧水壶的语境图。

语境图对系统以及与系统交换信息的实体之间的边界进行切分，并明确什么是分析的对象。

・系统

分析对象的系统，这里指话题烧水壶系统

・终端连接器

与其他系统、硬件等分析对象进行数据发送接收的实体

UI、水位传感器、壶盖传感器、热敏电阻、加热器、蜂鸣器

・数据流

表示流动数据的名称与流动的方向

打开加热器、关闭加热器、水位等

以下是输入与输出UI（用户接口）的详细内容：

・输入UI

出水按钮、解除按钮、烧水按钮、定时器按钮、保温设定按钮

・输出UI

灯：烧水、保温、高温、节约、牛奶、锁定

LCD（16*2）：温度显示、水位显示、时间显示

话题烧水壶的结构化分析4
数据字典1

用户需求 = [保温模式|出水口的锁定|出水口锁定解除|出水|烧水|定时器时间]
用户通知 = [保温模式显示|温度显示|烧水灯亮|烧水灯熄灭|保温灯亮
|保温灯熄灭|锁定灯亮|锁定灯熄灭|水位显示|定时器时间显示]
定时器时间显示 = 整数、单位：分、下限：0
保温模式 = [高温模式|节约模式|牛奶模式]
温度异常 = [高温异常|温度不上升异常]
水位 = [0|1|2|3|4|水满|异常]、单位：无
水温 = 整数、单位：℃
壶盖的状态 = [开|关]
出水 = [开始|停止]
烧水 = [开始|停止]、时间：3分钟
保温 = [开始|停止]
高温模式 = 与烧水基本一样效果的保温模式
节约模式 = 节约电量的模式
牛奶模式= 用于冲婴儿奶粉的模式

数据字典中定义了话题烧水壶使用的数据。

这里记载了BNF记法中各数据的结构。

例）

用户需求=[保温模式丨出水口的锁定丨出水口的锁定解除丨出水丨烧水丨定时器时间]

＝＞用户需求是指保温模式、出水口的锁定、出水口的锁定解除、出水、烧水、定时器时间中的任何一项。

话题烧水壶的结构化分析5
数据字典2

保温温度 = [**98**|**90**|**60**]、单位：℃

温度控制方式 = [PID控制方式|温度控制Table方式|目标温度ON/OFF控制方式]、
监视时间：**100MS**一次

高温异常温度 = ［110]、单位：℃

高温异常式样 = 高温异常温度 + 检测周期

温度不上升异常式样 = 与目标温度的差异 + 与前一次检测的温度的差异 + 检测周期

当前保温模式 = 保温模式

当前水温 = 水温

当前水位 = 水位

当前经过时间 = 整数、单位：秒、下限：0

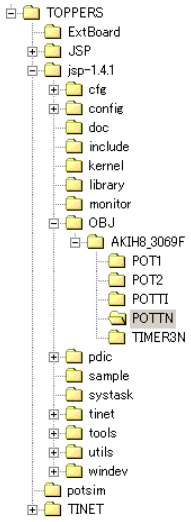
烧水温度式样 = 烧水温度 + 烧水加热时间

烧水温度 = 整数、单位：℃

锁定 = [锁定|锁定解除]

话题烧水壶开发工作空间1
POT1和POT2

- 请确认开发目录
- **POT1**在只有驱动的环境下创建所有的应用程序
- **POT2**已实装了驱动＋定时器功能。仅开发温度控制功能
- 请选择一个进行开发



2006/12/22 TOPPERS工程認定 73 TOPPERS

话题烧水壶样机的软件开发环境有POT1和POT2两个目录。请选择其一进行开发。

・POT1只准备了以前介绍过的驱动的源程序。请自己创建控制话题烧水壶的应用程序（几个任务）。但为了进行设备的初始化已准备了event_task。启动此任务时，进行设备的初始化init_pot（）。设备的初始化必须在连接了netDevice后进行，所以无法通过ATT_INI等服务调用进行初始化。

・POT2已安装了驱动的源程序和除温度控制以外的事件列表的功能。选择了POT2时，请仅开发温度控制功能。

设定话题烧水壶模式的event_task只对输出用户接口进行设定，所以已实现的温度控制功能与此任务之间的通信功能要由自己来实现，而且请变更event_task。

话题烧水壶开发工作空间2
POT1和POT2的文件说明

- pot1.c/pot1.h (pot2.c/pot2.h)
  - ▲主程序与引用程序，请在此处追加实装
- pot1.cfg (pot2.cfg)
  - ▲壶的配置文件
  - ▲描述任务定义、信号量定义等
- akih8pot_device.c/akih8pot_device.h
  - ▲**Windows**上的壶模拟器相关的设备驱动（**netDevice**用的驱动）
- akih8lcd_device.c
  - ▲**AKIH8/3069F**上的**LCD**相关的设备驱动器（扩展板用的驱动）

jsp-1.4-1

cfg	Kernel构造器 安装了将配置文件转换成源文件或进行检查等的程序
config	目标依赖部分
doc	文档
include	通用头文件 有μITRON4.0相关的声明
kernel	Kernel源文件
library	支持库源文件
monitor	任务监视器源文件（话题烧水壶用）
OBJ	教材用的程序开发环境（话题烧水壶用）
pdic	PDIC（设备驱动的OS非依赖部分）
sample	TOPPERS/JSP 样例程序
systask	系统服务源文件
tinet	TINET（TCP/IP协议栈）相关的文件
tools	取决于开发环境的目录
utils	实用程序
windev	Window设备管理器
potsim	VisualBasic工程（话题烧水壶用）
extborad	VisualBasic工程（碗面定时器用）

akih8pot_device.c是汇集了访问netDecive的驱动的源文件。此文件在SIL接口中访问硬件，而且是通过#include〈monsil.h〉访问任务监视器用的SIL接口。监视器用的SIL在连接到netdevice时访问netDevice服务器，没有连接时访问实机的硬件。

akih8pot_device.c是实机的驱动用的源文件，它引用了实机用的SIL接口用的引用文件〈sil.h〉。

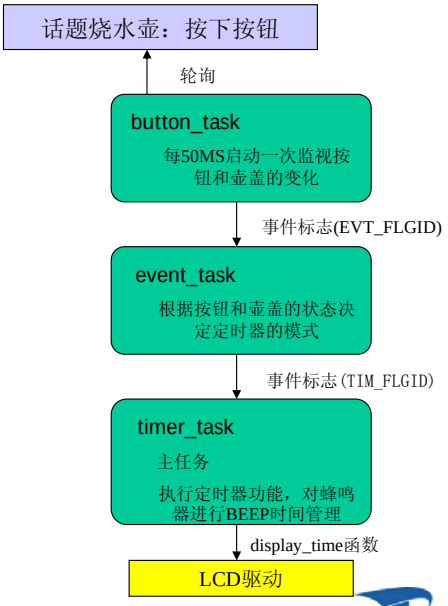
话题烧水壶开发工作空间3
POT1和POT2的文件说明

- tinet_app_config.h
  - ▲网络接口相关的定义
- tinet_pot1.cfg (tinet_pot2.cfg)
  - ▲TINET配置文件
- route_cfg.c
  - ▲路由表

・pot1.c/pot1.h(pot2.c/pot2.h)
初始状态下只实装了事件任务的主函数。
请在此处追加其他任务的主函数等。
pot2在此处实装了定时器功能、按钮事件的查看功能。
・pot1.cfg(pot2.cfg)
壶的配置文件
此处记述任务定义、信号量定义等。
・akih8pot_device.c/akih8pot_device.h
Windows上的壶模拟器相关的设备驱动器
如果使用此处的函数，则可以通过LAN操作壶模拟器的开关状态、壶的温度、壶的水位等。
・akih8lcd_device.c
H8/3069F板上的LCD相关的设备驱动
如果使用此处的函数，则可以直接操作LCD。
・tinet_app_config.h
网络接口相关的定义
・tinet_pot1.cfg(tinet_pot2.cfg)
TINET配置文件
・route_cfg.c
路由表

话题烧水壶开发工作空间4
POT2的定时器功能

- **POT2中实装了监视按钮按下的按钮任务（button_task）、解析按钮的事件任务（event_task）、处理定时器与蜂鸣器的定时器任务（timer_task）**
- **3个任务使用事件标志进行通信**



在POT2中，除驱动以外还实装了三个代表性的任务。

1）button_task

以50ms的周期启动监视话题烧水壶的按钮和壶盖的状态的任务。当按钮和壶盖的状态发生变化时，在event_task中使用事件标志（EVT_FLGID）进行通知。

2）event_task

指定话题烧水壶的模式。有定时器的启动要求或蜂鸣器的打开要求时，在timer_task中使用事件标志（TIM_FLGID）实现执行要求。想控制壶的温度时，需要创建此任务并扩展模式设定用的通信功能。

3）timer_task

以200ms的周期进行启动，实现定时器功能、蜂鸣器的开/关功能。

话题烧水壶开发完毕的固件1
驱动的说明

- 已创建了驱动，放在**akih8pot_device.c**和**akih8icd_device.c**两个源文件中
- 输入输出的参数设定在**akih8pot_device.h**中进行了定义
- 驱动需要用**initial_pot**函数进行初始化

函数	输入	输出	功能
initial_pot	无	无	实现LED与LCD的初始化、端口７与A的初始化、A/D转换器的初始化

使用驱动时需要用initial_por()函数进行初始化。实际的端口的初始化可以通过ATT_INI服务调用在任务启动前进行。但由于此开发环境需要经由网络进行初始化，所以在任务启动后再进行初始化。

驱动akih8lcd_device.c汇集了控制扩展板上的LCD的函数。aki8ild_device.c汇集了从应用程序调用的函数。在这些函数中，有访问netDevice与话题烧水壶进行通信的函数，也有调用akih8ild_device.c的函数访问LCD等的函数。

话题烧水壶开发完毕的固件2
输入驱动

函数	输入	输出	功能
get_switch	无	端口B的BIT设定	获取按钮与壶盖的状态 1为ON、 0 为OFF
get_temp	无	温度 0.125度单位	获取1/8度单位的温度 1度为8
get_level	无	水位 5位	以5位获取水位 第4位为ON时水满
get_simtime	无	模拟时间0.1秒单位	0.1秒的单位获取话题烧水壶模拟器的时间

输入用的驱动有上述4个函数。

get_switch函数获取话题烧水壶模拟器上的按钮和壶盖的状态。板B的BIT设定如下所示：

端口 B 设定(IN)

```
#define   PB_SWBOIL      0x01      /* SW BOIL BIT定义 */
#define   PB_SWPUMP      0x02      /* SW PUMP BIT定义 */
#define   PB_SWUNLOCK    0x04      /* SW UNLOCK BIT定义 */
#define   PB_SWTIMER     0x08      /* SW TIMER BIT定义 */
#define   PB_SWMODE      0x40      /* SW MODE BIT定义 */
#define   PB_COVER       0x80      /* COVER BIT定义 */
```

按钮是0为关闭，1为打开；壶盖是0为打开状态，1 为关闭状态。

get_temp函数获取话题烧水壶模拟器的当前水温。1对应0.0125度，所以1度对应8，2度对应16。

get_level函数获取话题烧水壶模拟器的当前水位。从0到4的5位对应传感器检测到的值，所有位都为0则表示没水，所有位都为l则表示水满。

get_simtime函数获取话题烧水壶模拟器的当前时间。单位是0.1秒。TOPPERS/JSP也有系统时间，所以应用程序也可以使用系统时间。话题烧水壶模拟器有时间栏，可以设定时间速度。为了对应时间栏，最好使用get_simeime。

话题烧水壶开发完毕的固件3
输出驱动

函数	输入	输出	功能
set_led	PotLedID req	无	设定灯(LED)的点亮、熄灭
set_control	PotCtlID req	无	设定抽水泵、锁定、加热器、加热器电源、蜂鸣器
display_temp	temp level	无	LCD上显示温度与水位
display_time	time	无	以1秒为单位显示定时器时间

输出驱动有以上4个函数。

set_led函数通过req设定灯的开/关。Rep=ON时灯亮，OFF时熄灭。灯的指定要依照下面的enum的指定。（LED_LEVEL的Req是显示BIT值）

```
typedef enum potledid{
   LED_BOIL=0,          /* BOIL ID */          烧水灯
   LED_KEEPWARM,        /* KEEPWARM ID */      保温灯
   LED_HIGH,            /* HIGH ID */          高温灯
   LED_ECO,             /* ECO ID */           节约灯
   LED_MILK,            /* MILK ID */          牛奶灯
   LED_LOCK,            /* LOCK ID */          锁定灯
   LED_LEVEL,           /* LEVEL ID */         等级灯（未使用）
  NUM_LED               /* Number of LED ID */
} PotLedID;
```

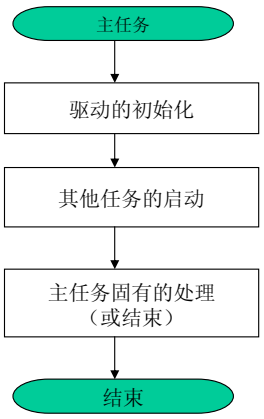
set_control函数通过rep的值对设备进行控制。rep=ON时启动，OFF时停止。可以控制的设备要依照下面的enum的指定。

```
typedef enum potctlid{
   CTL_PUMP,            /* PUMP ID */          抽水泵
   CTL_LOCK,            /* LOCK ID */          锁定（未使用）
   CTL_HEATER,          /* HEATER ID */        加热器
   CTL_HEATPOWER,       /* HEATPOWER ID */     加热器电源
   CTL_BUZZER,          /* BUZZER ID */        蜂鸣器
   NUM_CTL              /* Number of CTL ID */
} PotCtlID;
```

display_temp函数与display_time函数在LCD上显示设定值。LCD的细微的控制由驱动实现。time=0时不显示时间。

定时器的主要功能
任务的启动顺序

- 在**POT2**中，**timer_task**作为主任务实现设备的初始化、任务对象的初始化、其他任务启动
- 为了实现在连接**netDevice**后能从其他任务访问端口，请设计成从主任务启动其他任务



POT2的定时器任务兼任主任务，进行系统的初始化。当然，在创建了初始化任务这样的任务且进行了初始化和其他任务的启动后，也可以将其结束。但是，在像TOPPERS/JSP这样静态设定各Source的RTOS中用于初始化任务的堆栈空间在结束后不会被释放，所以在没有结束时以主任务的形式执行其他功能就可以有效地活用内存。

在此系统中，任务监视器首先启动。任务监视器调用akih8_device.c中的need_monitor（）函数。need_monitor函数查看扩展板上的DIP_SW，1为ON时返回TRUE。如果返回TRUE，监视器继续运行。DIP_SW的1为OFF时，启动TIMER_TASK，返回FALSE。如果返回FALSE，监视器就结束任务，所以将会出现无监视器任务的运行。通过改写need_monitor函数可以出现最初启动的任务的结构。但没有启动监视器时不会与netDevice进行连接，所以无法控制话题烧水壶模拟器，请注意这一点。

请开始实习

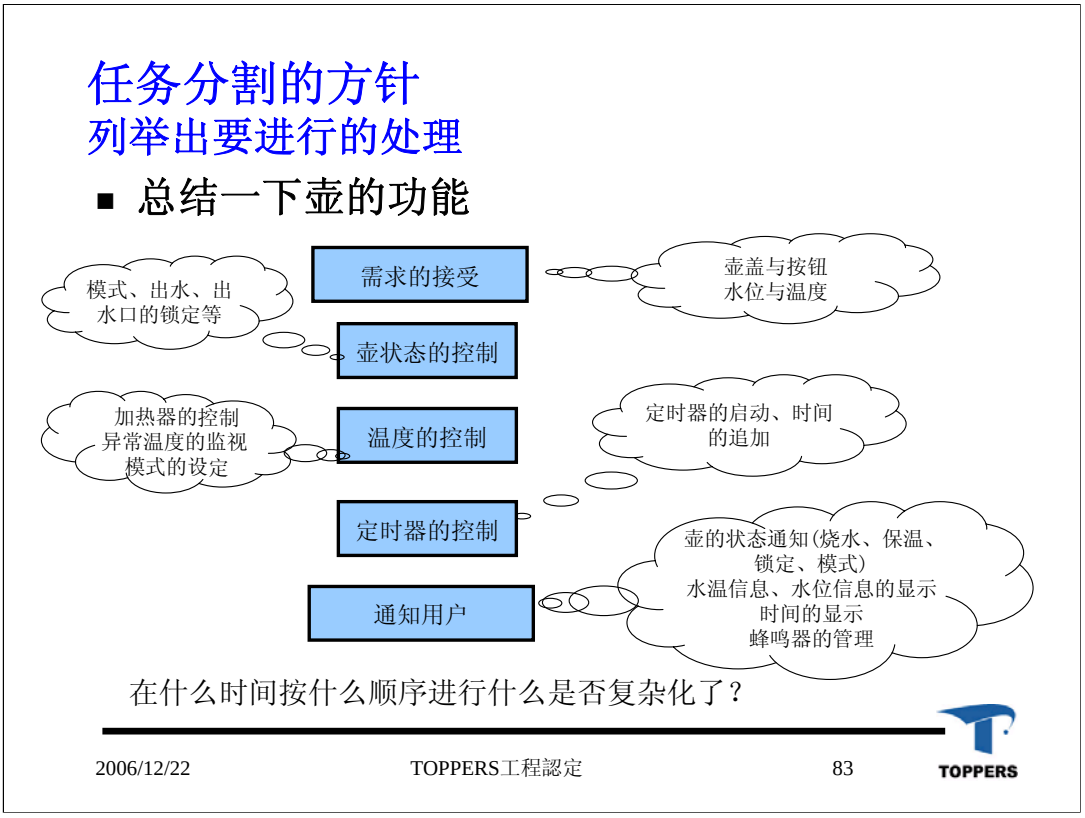
- 请结束**ExtBoard**模拟器，并启动话题烧水壶模拟器（**H8/3069F**版）
- **netDevice**服务器可以直接使用

- 实习开始后有几点开发的提示，请将此作为实习的参考

制作提示1

1. 构建应用RTOS的实时系统
2. 介绍实习课题
3. 确认开发环境
4. 制作话题烧水壶1

有几点制作的提示。



总结一下话题烧水壶的功能，可以总结出以下5点。

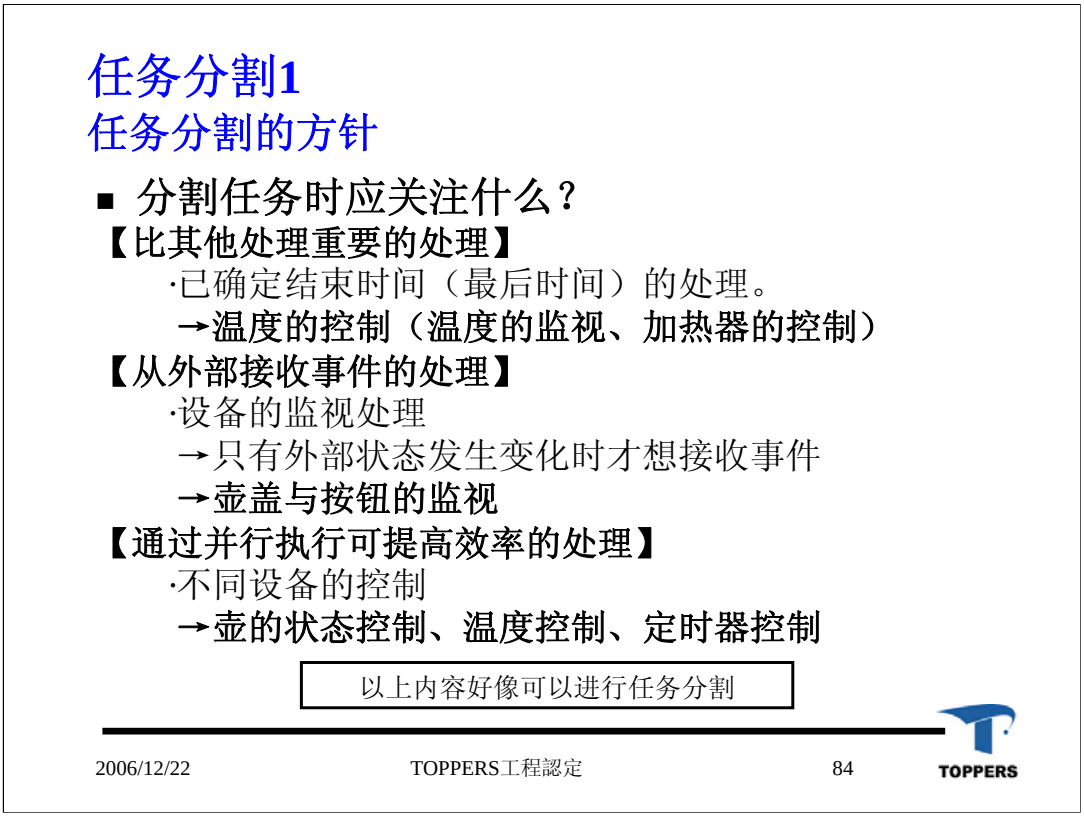
1）需求的接收：壶盖与按钮的状态的接收部

2）壶状态的控制：通过壶盖和按钮指定壶的模式（烧水、3种保温模式）
指定出水、出水的锁定
对定时器发出运行指示等

3）温度的控制：根据壶的模式、当前温度、当前水温指定目标温度，控制加热器。
检测出异常温度时，进行错误通知并停止加热器。

4）定时器的控制：发生定时器启动要求，对剩余时间的计算、显示、剩余时间结束后的蜂鸣器要求等进行控制。

5）通知用户：
壶的状态通知：烧水、保温、锁定等
当前水温、水位的显示
时间的显示、蜂鸣器的管理



任务的划分方针（其实只是划分的例子）

・比其他处理重要的处理：壶运行时的基本功能
已确定结束时间（最后时间）的处理
温度的控制→加热器的控制（100MS周期）：在100MS内指定目标温度，进行加热器的开/关。
（所以加热器的控制周期的单位为100MS）
→温度的监视（100MS周期）：以100MS周期检测温度的异常。
如果运行不正确，可能会引起火灾，所以至关重要。

・从外部接收事件的处理
设备的管理功能
→壶盖和按钮的状态监视：监视壶盖和按钮的变化，如果发生变化要通知给其他任务等。
要考虑响应，周期为50MS

・并行执行可以提高效率的处理
不同的设备的控制
→壶的状态控制：烧水、保温、出水、出水禁止等的判定
定时器启动、定时器延长等
→温度的控制：参考上面的内容
→定时器的控制：剩余时间的控制
→蜂鸣器的控制：在一定时间内鸣蜂鸣器

任务分割2
任务分割的例子

- 实际分割一下任务

任务名	任务说明	优先级
EVENT_TASK	壶的状态（烧水中、保温中等）的控制 状态通知	中
HEATER_TASK	根据水位以及水温控制加热器的ON、OFF	高
ERROR_TASK	监视水温，监视温度是否发生异常	高
TIMER_TASK	・控制厨房定时器 ・控制蜂鸣器	低
BUTTON_TASK	监视用户的操作(按下按钮、壶盖的开与关)	中

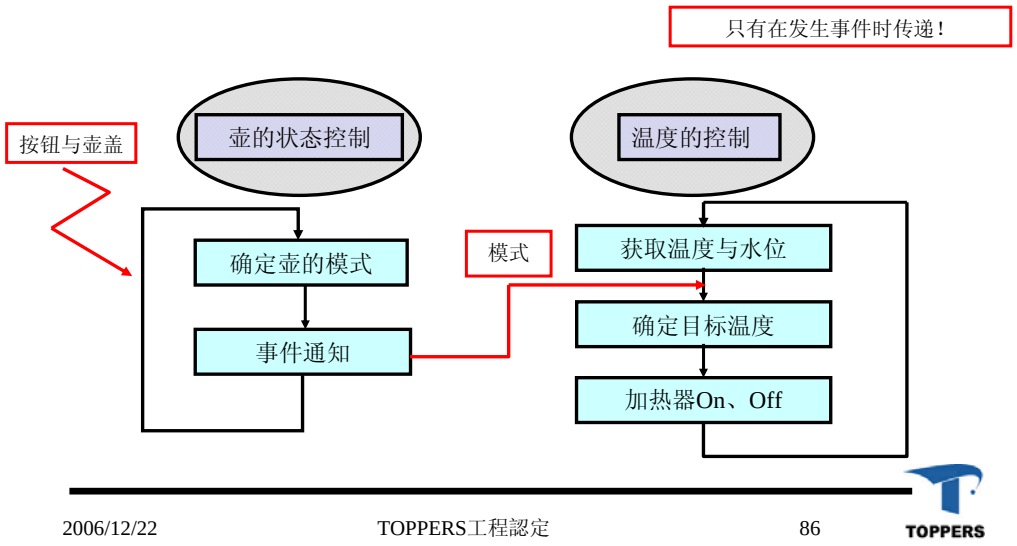
在这个例子中，参考POT2的定时器任务，在TIMER_TASK中嵌入了对蜂鸣器的控制。
将对蜂鸣器的控制分给别的任务更易于理解。

任务分割的例子
作为实例尝试进行任务分割
1.EVEVT_TASK：作为event_task实装在pot2.c中。为了实现与HEATER_TASK的通信需要扩展。
2.HEATER_TASK ：未实装
3.ERROR_TASK ：未实装
4.TIMER_TASK ：作为timer_task实装在pot2.c中，具有定时器的功能和蜂鸣器的功能。
5.BUTTON_TASK ：作为button_task实装在pot2.c中。

任务分割3

任务分割的方针

- 将壶状态控制任务（EVENT_TASK）与温度控制任务（HEATER_TASK）作成图



考虑一下EVEVT_TASK与HEATER_TASK之间的通信。

HEATER_TASK每100MS获取一次水温和水位来确定目标温度，并指定100MS周期的加热器的开/关。所以，停止水位（满壶、没水）的加热器的功能由HEATER_TASK进行判断。
EVEVT_TASK根据壶盖和按钮的状态来指定模式，HEATER_TASK相关的控制通过事件标志来传递比较好，所以作为传递事件的例子可以考虑一下下述内容。

1）HEATER_CHECK：状态的变化（各种壶的状态发生变化）

2）HEATER_BOIL+HEAT_BOILCHG：烧水要求

仅HEATER_BOILCHG：烧水要求的变化、烧水时切换到保温

3) HEATER_ERROR:发生温度异常

程序的编写
配置文件的记述

- 将任务的生成记述追加到配置文件中

温度控制任务	使用加热器，控制水的温度
壶的状态控制任务	监视壶系统请求的事件

在源代码中是怎样进行表达的呢？

在pot1.cfg或pot2.cfg中声明生成任务

```
CRE_TSK
( HEATER_TASK , {TA_HLNG , (VP_INT) 0 , heater_task , HEATER_PRIORITY , STACK_SIZE , NULL })

CRE_TSK
( EVENT_TASK , { TA_HLNG,  0 , event_task ,  EVENT_PRIORITY , STACK_SIZE , NULL})
```

```
CRE_TSK
( 任务名 , {TA_HLNG , (VP_INT) 0 , heater_task ,任务的优先级, STACK_SIZE , NULL })
```


新建HEATER_TASK时，需要创建任务函数并使用CRE_TASK静态API登录到配置文件。

源文件另做处理时，在此之上还要修改Makefile。比如说，叫做heater.c的新建源文件进行追加时，请用对象名追加到第180行的UTASK_COBJS中。UTASK_DIR = $(SRCDIR)/library:$(NETAPP_DIR)

UTASK_ASMOBJS =

UTASK_COBJS = $(UNAME).o akih8pot_device.o akih8lcd_device.o $(NETAPP_COBJS)

UTASK_CFLAGS =

UTASK_LIBS =

↓

UTASK_DIR = $(SRCDIR)/library:$(NETAPP_DIR)

UTASK_ASMOBJS =

UTASK_COBJS = $(UNAME).o heater.o akih8pot_device.o akih8lcd_device.o $(NETAPP_COBJS)

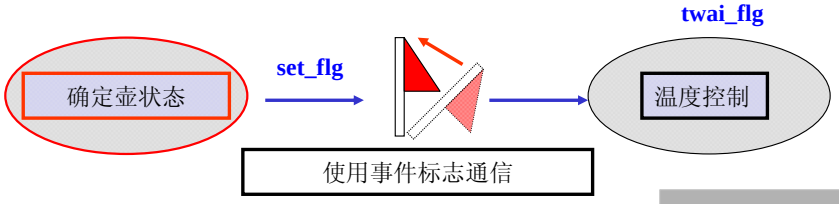
UTASK_CFLAGS =

UTASK_LIBS =

之后，重新进行make depend和make。

任务间的通信
使用事件标志进行任务间的通信①

- 壶状态任务与温度控制任务之间的通信是使用事件标志



详细内容请参考
μITRON的式样书

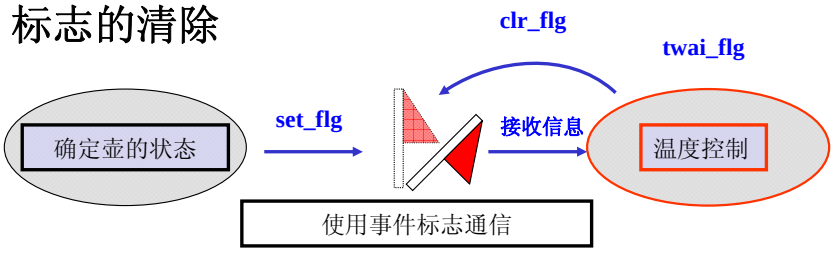
```
//创建标志(写在**.cfg文件中就可以)
  CRE_FLG  ( HEAT_FLGID, { TA_TFIFO | TA_WSGL  ) , 0 } ) ;
  CRE_FLG  (事件的种类,{(标志的属性),标志的BIT类型初始值});
//通过壶的状态任务通知温度控制任务(通知迁移烧水模式时)
  set_flg   ( HEAT_FLGID , HEAT_BOIL ) ;
  set_flg   (标志的ID,想传达的信息);
//在温度控制端等待设置标志
  twai_flg   ( HEAT_FLGID , EVT_HEAT , TWF_ORW , &flgptn , HEATER_CYCLE ) ;
  twai_flg   (标志的ID,事件的种类,等待模式,&flgptn,超时周期);
```


TOPPERS

```
/* 按下烧水按钮 */
        if(!(event & BUTTON_BOIL) && (sw & BUTTON_BOIL)){
                set_flg(TIM_FLGID, TIM_1SEC_BUZZER);
                set_flg(HEAT_FLGID, HEAT_BOIL);
```

任务间的通信
使用事件标志进行任务间的通信②

- 标志的清除




```
//在温度控制端清除接收的标志
  clr_flg ( HEAT_FLGID , ~flgptn );
  clr_flg  (标志的ID,清除的BIT类型);
```



```
ercd = twai_flg(HEAT_FLGID, EVT_HEAT, TWF_ORW, &flgptn, HEATER_CYCLE);
if(ercd == E_OK && state != STATE_ERROR_HEATER){
        clr_flg(HEAT_FLGID, ~flgptn);
        if(flgptn & HEAT_ERROR){
```

通过set_flg创建flgptn的对应位。
如果是HEAT_BOIL则从define的定义开始flgptn = 0x02 (0000 0010)
所以，clr_flg的清除的位类型如下所示：

~flgptn = 1111 1101

通过Flgptn和~flgptn和的逻辑积，清除HEAT_BOIL的位。

任务间的通信

使用事件标志进行任务间的通信③

- 两个任务间通信的实装例子

```
//事件任务端                                          标志等待
    for(;;){
        ercd = wai_flg(EVT_FLGID, EVT_EVENT, TWF_ORW, &flgptn);
    //通过其他事件进行处理
        . . . . . .
    //按下烧水按钮
        set_flg(HEAT_FLGID, HEAT_BOIL);
    }
//加热器任务端                                        标志等待
    for(;;){
      ercd = twai_flg(HEAT_FLGID, EVT_HEAT, TWF_ORW, &flgptn,HEATER_CYCLE);
      if(ercd == E_OK && state != STATE_ERROR_HEATER){
        clr_flg(HEAT_FLGID, ~flgptn);          标志清除
          . . . . . }
    }
```

制作提示2

1. 构建应用RTOS的实时系统
2. 介绍实习课题
3. 确认开发环境
4. 制作话题烧水壶1

温度控制任务的实装例子1
名称是加热器任务（heater_task）

- 任务与源代码的对应

加热器任务	使用加热器，控制水的温度	

在加热器任务内记述实际的处理

```
void heater_task(VP exinf)
{
  set_control(CTL_HEATPOWER, ON);          //打开加热器电源
  for (;;) {                               //无限循环
            模式的获取、或监视时间的超时
            根据模式与水位设定温度
            当前温度的比较
            加热器的加热、加热停止
            温度与水位的显示
  }
}
```

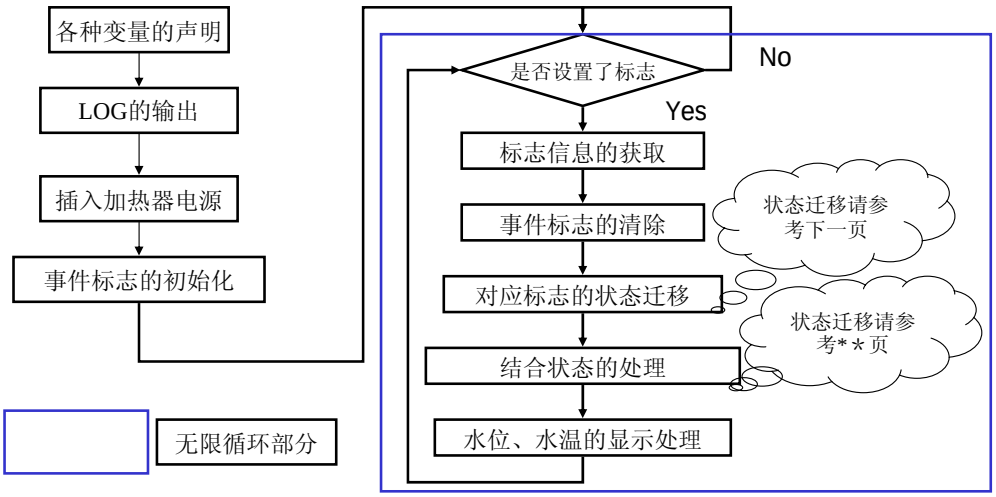
不要马上就开始写源代码，而要根据状态迁移图、流程图等内容进行实装。

实际上，如果只剩下源代码，在此之后发生式样变更、错误等时可能会很难维护。

（多数情况都无法理解本人的意图。）

所以为了也能让第一次担当的人员很容易维护开发的成果物，保留设计文档以及更新文档是非常重要的。

温度控制任务的实装例子2
画流程图



TOPPERS

＜通信用事件标志的例子＞

```
/*
 * HEAT_FLGID的位定义
 */
#define         HEAT_CHECK              0x01            /* 状态变更 */
#define         HEAT_BOIL               0x02            /* 烧水指定 */
#define         HEAT_BOILCHG            0x04            /* 烧水按钮 */
#define         HEAT_ERROR              0x10            /* 出错 */

#define         EVT_HEAT                (HEAT_CHECK|HEAT_BOIL|HEAT_BOILCHG|HEAT_ERROR)
```

＜状态的enum设定的例子＞

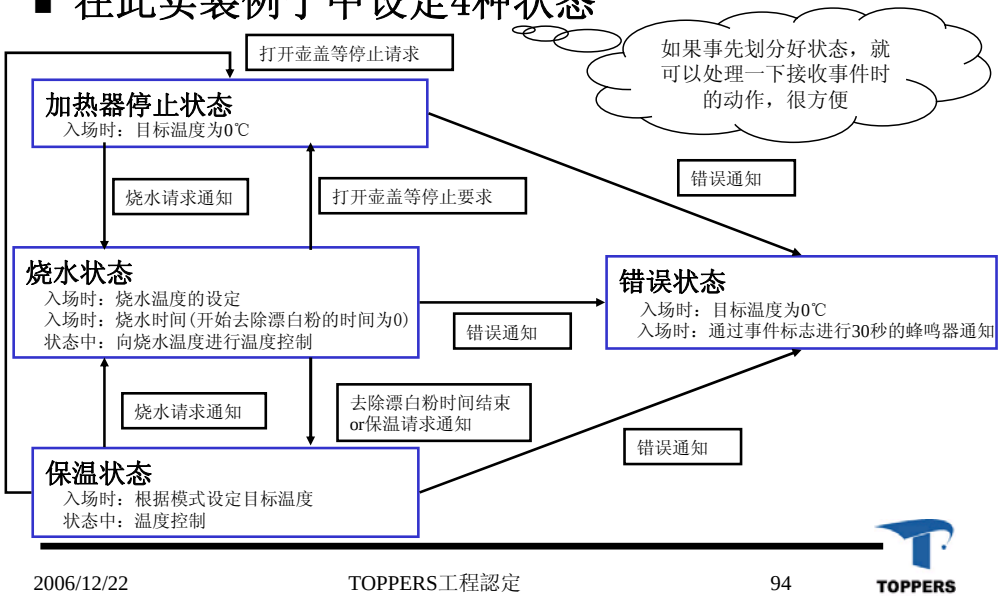
```
/*
 * 加热器任务的状态
 */
typedef enum heat_state {
        STATE_STOP_HEATER,              /* 加热器停止状态 */
        STATE_BOIL_HEATER,              /* 烧水状态 */
        STATE_KEEP_HEATER,              /* 保温状态 */
        STATE_ERROR_HEATER              /* 出错状态 */
} HEAT_STATE;

void heater_task(VP_INT exinf)
{
  SYSTIM    boil_time;                  系统时间型、作为烧水时间使用
  ER        ercd;                       错误信息型、错误信息（使用事件标志时使用）
  FLGPTN    flgptn;                     标志类型型、保存收到的事件值
  HEAT_STATE state = STATE_STOP_HEATER; enum型、保存heater_task的状态的函数
  H         oldtemp;                    保存100ms前的温度（1度＝８）
  UB        potlevel, oldlevel;         保存当前水位和100ms前的水位的变量
  int       count;                      循环的次数
```

温度控制任务的实装例子3

温度控制任务的状态

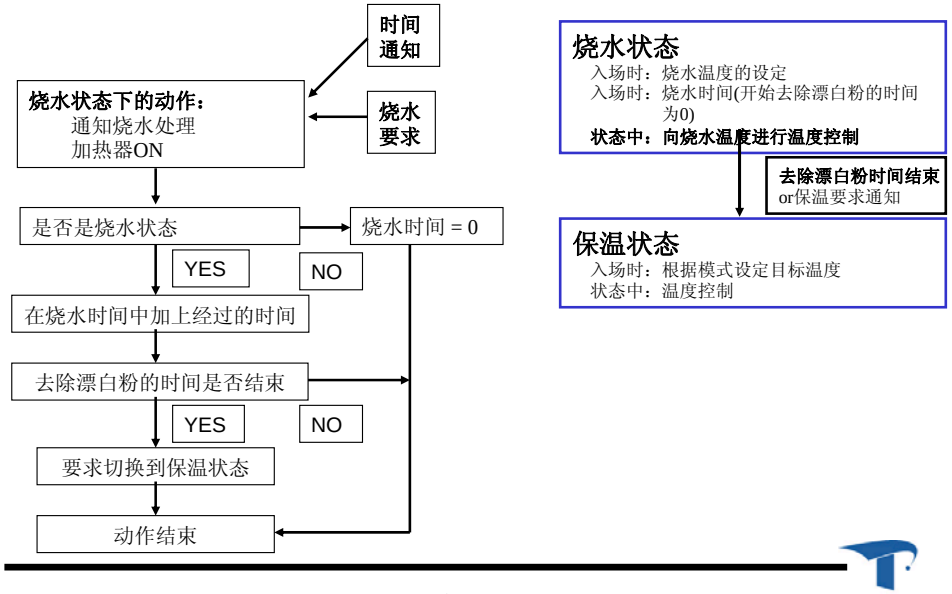
- 在此实装例子中设定4种状态



＜SampleList续 1 ＞

```
syslog(LOG_INFO, "Pot heater task starts (exinf = %d).", exinf);
pottemp  = get_temp();                          /* 初始状态的温度设定 */
oldtemp  = pottemp - 1;
potlevel = get_level();                         /* 初始状态的水位设定 */
oldlevel = potlevel - 1;
count  = 0;                                     /* 计数器的初始化 */
set_control(CTL_HEATPOWER, ON);                 /* 插上加热器电源 */
clr_flg(HEAT_FLGID, 0);
for(;;){
  ercd = twai_flg(HEAT_FLGID, EVT_HEAT, TWF_ORW, &flgptn, HEATER_CYCLE);
  if(ercd == E_OK && state != STATE_ERROR_HEATER){
    clr_flg(HEAT_FLGID, ~flgptn);
                        //以下是状态迁移的位置
    if(flgptn & HEAT_ERROR){
      state = STATE_ERROR_HEATER;
      targettemp = 0;
      set_flg(TIM_FLGID, TIM_30SEC_BUZZER);
    }
    else if(potlevel > 15 || potlevel == 0 || (potstate & COVER_OPEN)){
      state = STATE_STOP_HEATER;
      targettemp = 0;
    }
    else if(state != STATE_BOIL_HEATER){
      int i;
      i = (potstate & MODE_MASK) - 1;
      if((flgptn & HEAT_BOIL)){
        targettemp = temp_table[i][0];
        state = STATE_BOIL_HEATER;
        boil_time = 0;
      }
      else{
        targettemp = temp_table[i][1];
        state = STATE_KEEP_HEATER;
      }
    }
  }
```

温度控制任务的实装例子4
烧水状态

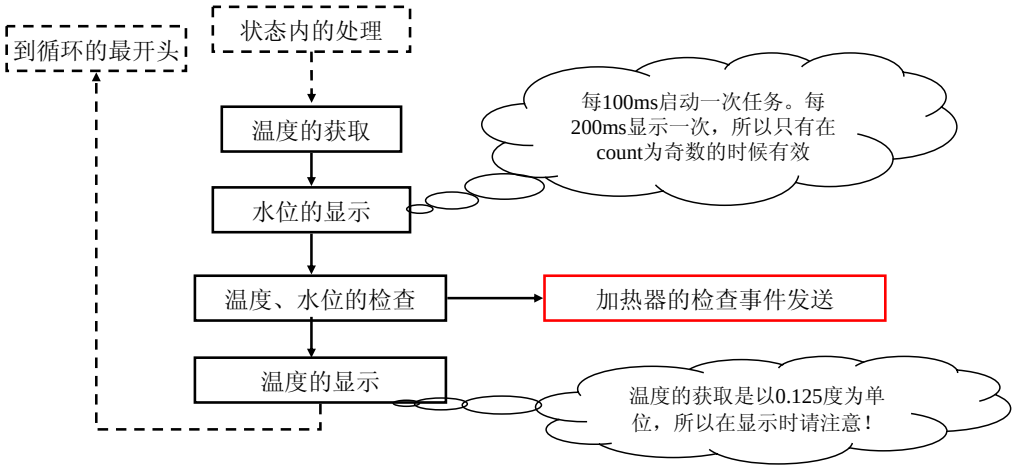


<SampleSourceList续 2 >

```
    switch(state){
  case STATE_STOP_HEATER:                    /* 加热器停止状态 */
    set_control(CTL_HEATER, OFF);
    set_led(LED_BOIL, OFF);
    set_led(LED_KEEPWARM, OFF);
    break;
  case STATE_BOIL_HEATER:                    /* 烧水状态 */
    set_led(LED_BOIL, ON);
    set_led(LED_KEEPWARM, OFF);
    set_control(CTL_HEATER, ON);
    if(boil_time){
      if((get_simtime() - boil_time) >= 30*60){
        state = STATE_KEEP_HEATER;
        set_flg(HEAT_FLGID, HEAT_CHECK);
      }
    }
    else if(pottemp >= targettemp)
      boil_time = get_simtime();
    break;
  case STATE_KEEP_HEATER:                    /* 保温状态 */
    set_led(LED_BOIL, OFF);
    set_led(LED_KEEPWARM, ON);
    if(pottemp > targettemp)
      set_control(CTL_HEATER, OFF);
    else if(pottemp < targettemp)
      set_control(CTL_HEATER, ON);
    break;
```

温度控制任务的实装例子5
不取决于状态的处理

- 温度的水位显示



<SampleSourceList续 3 >

```
    default:                                                    /* 出错状态 */
        set_control(CTL_HEATER, OFF);
        set_control(CTL_HEATPOWER, OFF);
        set_led(LED_BOIL, OFF);
        set_led(LED_KEEPWARM, OFF);
        break;
    }
    pottemp = get_temp();                                       /* 温度的获取 */
    count++;                                                    /* 向上计数 */
    if(count & 1){                                              /* 每200ms的处理 */
      potlevel = get_level();
      set_led(LED_LEVEL, potlevel);
    }
    if(pottemp != oldtemp || potlevel != oldlevel){
      set_flg(HEAT_FLGID, HEAT_CHECK);
      display_temp((pottemp+5)>>3, potlevel);
      oldtemp  = pottemp;
      oldlevel = potlevel;
    }
  }
}
```