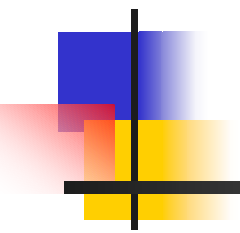


TI C6000 DSP/BIOS

实时操作系统RTOS(1)

- 实时系统最大特点以及与普通操作系统的区别
  任务要按时完成；deadline；
  由于大多数实时系统是嵌入式，所以又常常称为嵌入式实时系统.
- 实时操作系统的作用：
  将CPU时间、中断、I/O、定时器等资源都包装起来，留给用户一个标准的API，并根据各个任务的优先级，合理地在不同任务之间分配CPU时间。

实时操作系统RTOS(2)

- 实时操作系统的基本功能模块：
  任务管理、定时器管理、存储器管理、资源管理、事件管理、系统管理、消息管理、队列管理、旗语管理等
  管理功能是通过内核服务函数形式交给用户调用的，也就是RTOS的API
- 使用实时操作系统的优点：
  标准化、可移植性强（因为与具体的硬件隔离开来，由RTOS去管理）
  其它：任务调度，任务间通讯，消息传递，资源保护等功能，比自己直接开发的要完整。

基于DSP/BIOS的软件开发和传统软件开发的不同

- DSP/BIOS中，硬中断处理尽量快，一般不允许中断嵌套
  中断延迟：中断的关闭时间. 这是实时系统的一个重要指标。
- 所以传统的在中断中处理的任务一般要划分为两个部分：一个是控制部分，花时间少，放在HWI函数中；另外一部分是处理部分，放在SWI函数或任务处理中

基于DSP/BIOS的软件开发和传统软件开发的不同（2）

- DSP/BIOS 提供标准的时标，供整个应用程序参考。
  如：任务的切换是以时标为单位；
  超时等待的判断也是以这个时标为单位；
  (超时等待在传统模式中也可实现)

- DSP/BIOS 的时钟：
  高、低分辨率计时和系统时钟；
  时钟HWI将低分辨率计时加1；（CLK对象在时钟HWI中执行）
  系统时钟在在配置工具中设置：默认低分辨率计时和系统时钟系统，系统时钟还可由其它时间驱动。系统时钟驱动PRD模块。
  高分辨计时是按照定时器的计数寄存器的速率加1的，所以高分辨时间等于计数寄存器的累加次数。

基于DSP/BIOS的软件开发和传统软件开发的不同（3）

- DSP/BIOS 提供的实时监测手段丰富，虽然传统方式也可做记录，但是DSP/BIOS有的 功能还是不能实现：
  实时监测的数据在主机端格式化处理；
  CPU负载显示；
  4种API实现实时监测数据的采集：
  LOG, STS,HST,TRC

DSP/BIOS 概述

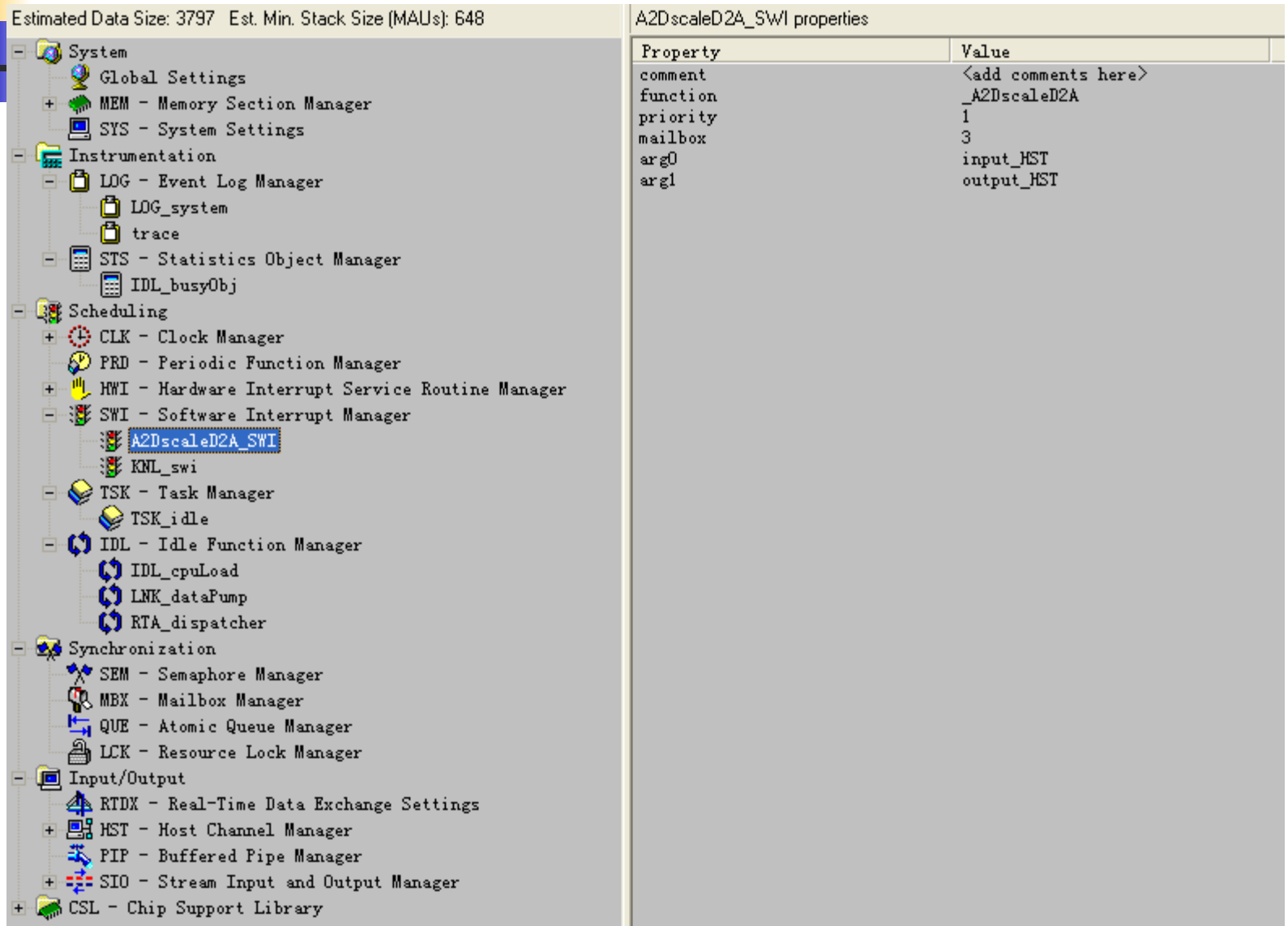
- 设计目的：
  为需要实时调度和同步，主机－目标系统通讯和实时监测的应用而设计的
- DSP/BIOS组件：
  抢先式多任务内核，硬件抽象层，实时分析工具和配置工具

DSP/BIOS 模块

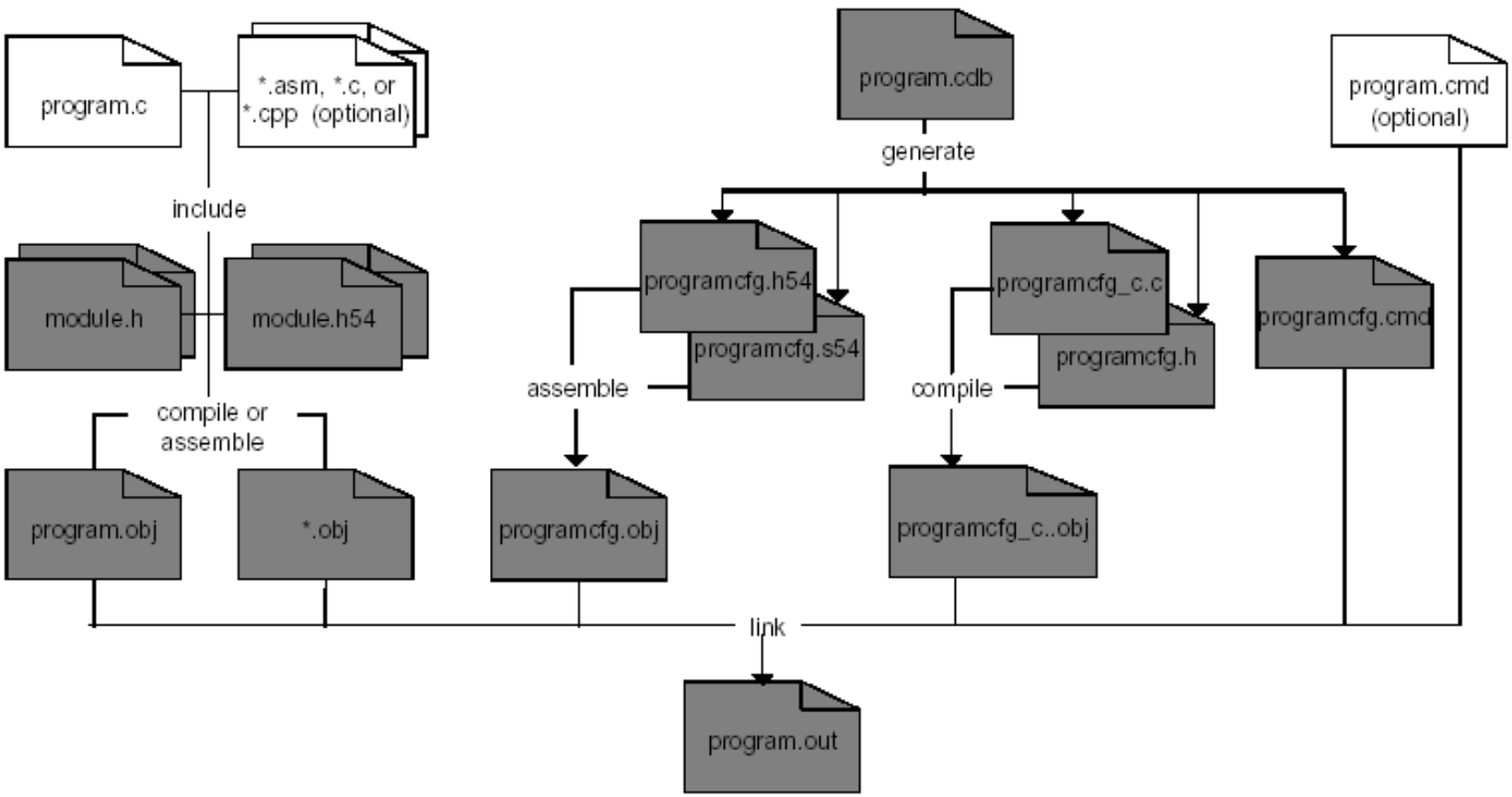
DSP/BIOS API被划分为多个模块。根据应用程序模块的配置和使用情况的不同，DSP/BIOS的代码大小从500字到6500字不等。应用程序通过调用API来使用DSP/BIOS，所有的DSP/BIOS API都是按C可调用的形式提供的。只要遵从C的调用约定，汇编代码也可以调用DSP/BIOS API。

模块名称	说明
ATM	Atomic functions written in assembly language
C54, C55, C62, C64	Target-specific functions, platform dependent
CLK	Clock manager
CSL	Chip Support Library;
DEV	Device driver interface
GBL	Global setting manager
HST	Host channel manager
HWI	Hardware interrupt manager
IDL	Idle function manager
LCK	Resource lock manager
LOG	Event log manager
MBX	Mailbox manager
MEM	Memory segment manager
PIP	Buffered pipe manager
PRD	Periodic function manager
QUE	Atomic queue manager
RTDX	Real-time data exchange settings
SEM	Semaphore manager
SIO	Stream I/O manager
STS	Statistics object manager
SWI	Software interrupt manager
SYS	System services manager
TRC	Trace manager
TSK	Multitasking manager

DSP/BIOS 配置工具



DSP/BIOS 文件


program.c
*.asm, *.c, or *.cpp (optional)
include
module.h
module.h54
compile or assemble
program.obj
*.obj
program.cdb
generate
programcfg.h54
programcfg.s54
assemble
programcfg.obj
programcfg_c.c
programcfg.h
compile
programcfg_c..obj
programcfg.cmd
program.cmd (optional)
link
program.out

DSP/BIOS的核心概念：线程

- 许多实时DSP应用都需要同时执行许多不相关的功能，这些功能一般是对外部事件的响应。这些功能称为线程。
- DSP/BIOS 支持4种：
  （1）硬件中断（HWI）：频率可达200KHz（5us），处理时限在2us～100us，包括CLK函数
  （2）软件中断（SWI）:时限100us以上，SWI允许HWI将一些非关键处理在低优先级上延迟执行，这样可以减少在中断服务程序中的驻留时间
  （3）任务TSK：任务与软件中断不同的地方在于在运行过程中可以被挂起。DSP/BIOS提供了一些任务间同步和通讯的机制，包括队列、信号灯和邮箱。
  （4）后台线程（IDL）： MAIN->空闲循环：运行那些没有执行期限（deadlines）的功能

线程的选择：

- 1.是否有时限要求
  1.1 是
  1.2 一部分有
  1.3 不如其它线程要求的时间严格
  1.4 没有时限要求：idle线程

线程的选择：1.1 有时限要求

- 线程是否需要时钟中断或其它硬中断触发？
  1.1 时钟中断触发，则可选
    （a）CLK线程
    （b）PRD函数
    （c）task/SWI：当线程需要挂起的时候
  1.2 其它中断触发，选HWI线程
  1.3 应用程序运行到一定的时候满足条件才运行：
    （a）一段时间expiration才运行，选PRD函数
    （b）其它程序条件：task/SWI
    （c）硬件中断是条件：处理分为2个线程，
      HWI+task/SWI

线程的选择：task or SWI

- （1）　需要函数在等待资源时能够挂起吗？
- （2）　函数和其它线程有复杂的关联（相互依赖）或复杂的数据共享吗？
- （3）　希望函数有自己的栈吗？（不是共享系统栈）
- （4）　希望线程使用LCK, MBX, or SEM modules吗？

（5）　希望线程在created, deleted, exited, made ready to run, or becomes the current thread时运行一个函数吗？（任务线程支持这些事件的全局函数挂钩（global function hook），例如，希望所有任务都调用一个函数来保存/回复外部硬件寄存器）。

（6）　这个线程比一个线程（上面的回答至少一个yes）优先级低的线程

有一个回答yes，选task，全no，选SWI;

Task和SWI的区别

- （a） 资源不能得到时，任务可以在执行时挂起，软件中断不行；
- （b） DSP/BIOS提供大量的结构用于任务间通讯和同步：信号，消息邮箱，资源锁；这些结构软件中断不能使用；
- （c） 每个任务有其自己的栈，软件中断共享系统栈
- （d） 在任务created, deleted, exits, or switches thread context时，和特定系统有关（System-specific）的函数可以被调用，这些函数用于**extend a task's context beyond the basic processor register set**。
- （e） 任务的优先级低于软件中断，高于背景idle线程，共16个优先级，用户使用1～15， 0好任务running the DSP/BIOS idle loop.

线程的选择(续2)

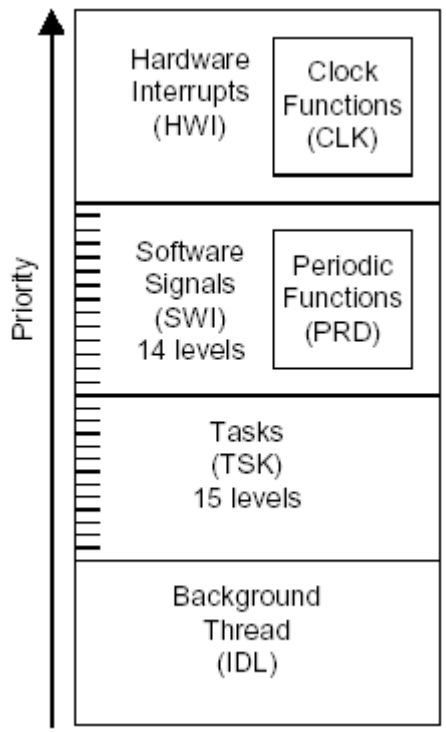
- 一部分有时限要求：
  把处理分成2个线程：HWI+TASK/SWI (同前)
- 不如其它线程的时限要求严格
  什么触发线程：是一段时间间隔完（time interval expiration），还是一些其它程序条件。 同前‘有时限要求1.3’

线程的选择小结

- 选线程类型：1. 看是否和中断有关， 2. 是否周期 3. 是否时限限制。
- （1） 选CLK 对象：时钟中断触发
- （2） 周期PRD对象：周期运行的功能
- （3） HWI对象：由硬件中断触发且有时限要求
- （4） SWI对象：需要执行完的任务，但时限要求没有HWI高
- TASK对象：那6个问题, 看是否需要：等待资源时挂起；数据共享和复杂关系；和其它线程通讯；有自己的栈（这样的好处）；任务切换时调用同一个函数；比task优先级低；

线程优先级

- 在DSP/BIOS中，硬件中断有最高的优先级，然后是软件中断，软件中断可以被高优先权软件中断或硬件中断抢先。软件中断是不能被阻塞的。任务的优先权低于软件中断，共有15个任务优先权级别（加上TSK_idle应该16个）。任务在等待某个资源有效时可以被阻塞。后台线程idle_loop是优先级最低的线程。

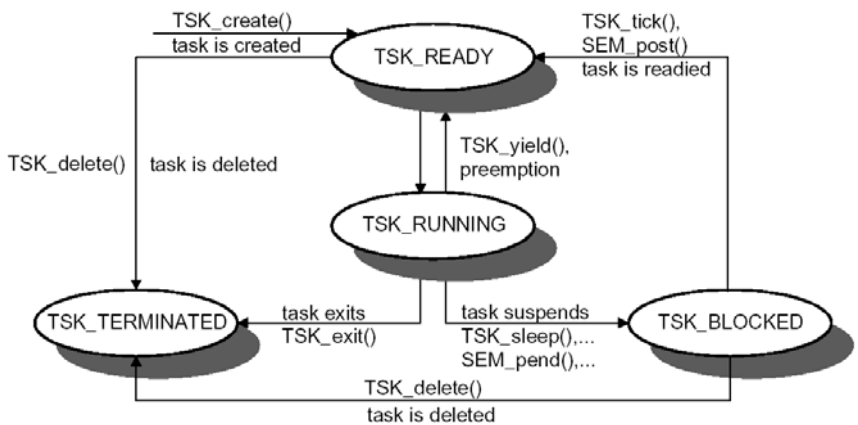


线程的抢占和让出

触发的线程（Thread Posted）	当前运行线程（Thread Running）			
	HWI	SWI	TSK	IDL
使能的 HWI	抢占	抢占	抢占	抢占
禁止的 HWI	等待重新允许	等待重新允许	等待重新允许	等待重新允许
使能的高优先级 SWI	——	抢占	抢占	抢占
禁止的 SWI	等待	等待重新允许	等待重新允许	等待重新允许
低优先级 SWI	等待	等待	——	——
使能的高优先级 TSK	——	——	抢占	抢占
禁止的 TSK	等待	等待	等待重新允许	等待重新允许
低优先级 TSK	等待	等待	——	——

任务的调度

- 运行态（Running），代表任务是处理器当前正在执行的线程，以TSK_RUNNING表示。
- 就绪态（Ready），代表任务一旦获得处理器的处理时间就可以执行，以TSK_READY表示。
- 阻塞态（Blocked），代表任务必须等到某个事件的发生才可以运行，以TSK_BLOCKED表示。
- 终止态（Terminated），代表任务已经结束，不会再运行，以TSK_TERMINATED表示。



expressDSP™ 算法标准

- eXpressDSP™ 算法标准使得按照算法标准开发的算法可以独立于具体的应用系统，主要是算法模块的资源分配和调度与算法分离，使得算法不经修改或只需很少修改就可以在不同的系统中重复使用。
- 算法标准主要规定了代码的编写规则，这包括适用于所有TI DSPs的一般规则和针对TI具体DSPs系列的一些特殊规则。此外，还规定了算法必须遵守的API标准