HZ BOOKS
华章科技

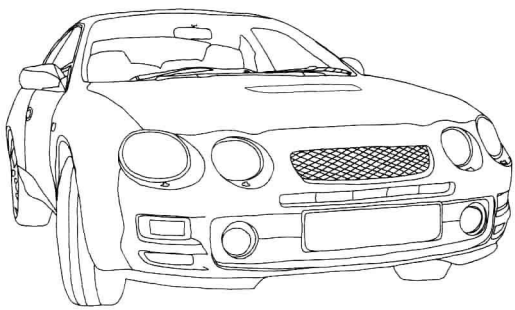
国内首本讲述用于汽车电子的、带有接口的开放式系统，它将计算机嵌入式软件理论与国内外最前沿的汽车电子开发技术紧密结合，使读者能够掌握先进的汽车电子软件开放式系统架构方法，熟练掌握软件开发环境的使用方法。

Automotive Software Open System Architecture

汽车软件开放式系统架构

张晋东◎著

机械工业出版社
China Machine Press



Automotive Software Open System Architecture

汽车软件开放式系统架构

张晋东◎著
秦贵和◎审

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

汽车软件开放式系统架构 / 张晋东著 . —北京：机械工业出版社，2013. 4

ISBN 978-7-111-42152-8

Ⅰ . 汽… Ⅱ . 张… Ⅲ . 汽车 – 电子系统 – 应用软件 – 高等学校 – 教材 Ⅳ . U463.6

中国版本图书馆 CIP 数据核字（2013）第 074504 号

本书讲述了一种汽车软件开发技术——汽车软件开放式系统架构。内容依托于欧洲汽车行业制定的 OSEK/VDX 开发规范。全书共 10 章，主要内容包括：系统架构的各个组成部分、实现启动和结束、开发支持的调试技术、任务状态模型、基于时间的服务、中断处理、任务间通信、资源管理、调度策略以及应用系统的设计；并讲述了如何使用由 Elektrobit 公司开发的 ProOSEK 环境来构建汽车软件开放式系统架构，使读者了解其基本原理、开发过程和相关开发工具的使用方法。

本书可供从事汽车电子软件开发的人员使用，也可作为计算机专业或软件工程专业相关课程的参考书。

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）
责任编辑：陈佳媛
北京诚信伟业印刷有限公司印刷
2013 年 6 月第 1 版第 1 次印刷
147mm × 210mm · 5.375 印张
标准书号：ISBN 978-7-111-42152-8
定 价：35.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

前　言

汽车软件开放式系统架构一词来自于OSEK的中文意译。OSEK这个词最早出现于20世纪90年代，德国的汽车制造商BMW、Bosch、Daimler-Chrysler、Opel、Siemens和VW旨在通用的实时分布式操作系统规范方面进行合作。参与项目的德国Karlsruhe大学提出了OSEK，全称为“Offene Systeme und deen Schnittstellen fur die Elektronik im Kraftfahrzeug”，中文意译为用于汽车电子的、带有接口的开放式系统。

本书共10章，主要内容包括：系统架构的各个组成部分、实现系统的启动和结束、开发支持的调试技术、任务状态模型、基于时间的服务、中断处理、任务间通信、资源管理、调度策略，以及应用系统的设计；并在各章中讲述了使用ProOSEK环境来构建汽车软件开放式系统架构的方法。通过阅读本书读者可以了解汽车软件开放式系统架构的基本原理、开发过程和相关开发工具的使用方法。

本书编写过程中得到了EB汽车软件（上海）有限公司的周建锋博士（Dr. Jeff Zhou）及李海等相关人员的大力支持和帮助，在此表示感谢！

感谢秦贵和教授在百忙之中审阅本书，并提出许多宝贵意见和建议，在此表示感谢！

由于本书编者知识积累有限，尤其是在汽车电子软件方面，所以本书难免存在错漏，敬请读者批评指正。

编者

2013年2月

目　录

前言

第1章　汽车软件开放式系统架构简介.....1
1.1　开放式系统架构的优势.....2
1.2　系统架构组成部分.....2
1.3　系统架构工具链.....3
1.4　操作系统的体系结构.....4
1.5　ProOSEK开发环境.....6

第2章　实现系统的启动和结束.....11
2.1　系统的启动.....11
2.2　系统的应用模式.....14
2.2.1　应用模式的适用范围.....15
2.2.2　系统的启动性能.....15
2.2.3　支持的应用模式.....16
2.3　系统的结束.....16
2.4　应用系统开发示例.....19
2.5　实现文件的管理.....23
2.5.1　创建与保存新项目.....23
2.5.2　打开与关闭项目.....24
2.5.3　项目向导.....25

第3章　应用程序开发的支持.....27
3.1　错误处理.....28
3.2　错误管理.....29
3.3　错误钩子例程.....31
3.4　调试分类.....31
3.4.1　系统架构运行时接口ORTI.....32
3.4.2　堆栈检查.....32
3.4.3　跟踪缓冲区.....33
3.5　钩子.....34
3.5.1　启动钩子.....34
3.5.2　前任务钩子.....35
3.5.3　后任务钩子.....35
3.5.4　错误钩子.....36
3.5.5　COM错误钩子.....36
3.5.6　关闭钩子.....36
3.6　系统的配置.....37

3.6.1 生成系统 37
3.6.2 验证配置 38
3.6.3 配置设置 38
3.6.4 获取内存需求 41
3.6.5 命令行模式 41

第4章 任务 43
4.1 任务状态模型 44
4.1.1 扩展任务 45
4.1.2 基本任务 47
4.1.3 任务类型的比较 48
4.2 激活任务 49
4.3 任务的切换机制 50
4.4 任务的优先级 50

第5章 报警器 53
5.1 报警器的配置 54
5.2 计数器 55
5.3 报警器的管理 56
5.4 报警回调例程 57
5.5 应用系统开发示例 58

第6章 中断和中断服务程序 62
6.1 中断分类 62
6.2 中断服务程序 64
6.3 应用系统开发示例 65

第7章 通信 67
7.1 事件 67
7.2 消息 76
7.3 应用系统开发示例 85

第8章 资源 92
8.1 资源管理 94
8.2 对被占用资源的访问 95
8.3 使用资源时的限制 96
8.4 调度作为资源 96
8.5 同步机制的一般问题 96
8.5.1 优先级反转 96
8.5.2 死锁 97
8.6 优先级上限协议 98
8.7 扩展中断级别的优先级上限协议 100
8.8 内部资源 103

第9章 调度 105
9.1 调度策略 106
9.1.1 完全抢占式调度 106
9.1.2 非抢占式调度 107
9.1.3 任务组 108
9.1.4 混合抢占式调度 109
9.1.5 选择调度策略 110
9.2 终止任务 110

第10章 系统设计 113
10.1 系统设计准则 113
10.2 系统设计目标 115
10.3 高级别的系统设计 115

10.3.1 管理对象 116
10.3.2 管理属性和参数 ...118
10.3.3 任务的设计 124
10.3.4 报警器的设计 128
10.3.5 中断的设计 133
10.3.6 通信的设计 134
10.3.7 资源的设计 139
10.3.8 调度的设计 140
10.3.9 事件的设计 140
10.4 低级别的系统设计 142
附录 API 参考 144
参考文献 166

CHAPTER 1

第1章　汽车软件开放式系统架构简介

目前，汽车上的电子控制单元（ECU）不断增加，致使汽车软件的质量和安全的重要性不断增加，需要一种新的软件开发技术标准来提高软件质量，使其具有更好的可维护性，更高的软件可移植性。汽车软件开放式系统架构（Automotive Software Open System Architecture）可以减少汽车软件在非应用方面的各种管理和开发的开支，还可以解决由于不同的接口协议导致的不同厂家生产的控制单元不相容的问题。汽车软件开放式系统架构支持应用软件的可移植性和可重用性，是尽可能独立于应用程序的抽象接口规范，包括：实时操作系统、通信和网络管理。它是一个硬件与网络相互独立的用户接口规范。它可以进行功能配置和扩展，并可对存在问题的应用进行体系结构的优化与调整，是一种高效的设计架构。使用该架构可对项目进行原型开发和功能核查。

1.1 开放式系统架构的优势

通过本书描述的汽车软件开放式系统架构可以实现不同体系设计的控制单元的接口标准化，从而节省了系统开发成本和开发时间。并且提供了一种没有规定具体实现的可单独执行的独立性规范，这样大大增强了汽车生产商控制单元的软件质量，有利于重用汽车中现有的分布信息资源，在不增加硬件的条件下提高整个系统的性能。

1.2 系统架构组成部分

汽车软件开放式系统架构的各个组成部分如图 1-1 所示，包括：操作系统（Operating System，OS）、任务（Task）、网络管理（Network Manage，NM）、I/O 层、通信管理（Communication Manage，CM）以及硬件物理层。其中，操作系统包括：基本功能、资源管理和事件机制。通信管理包括：任务与任务之间的通信。网络管理包括：网络的启动和关闭、监测网络节点。

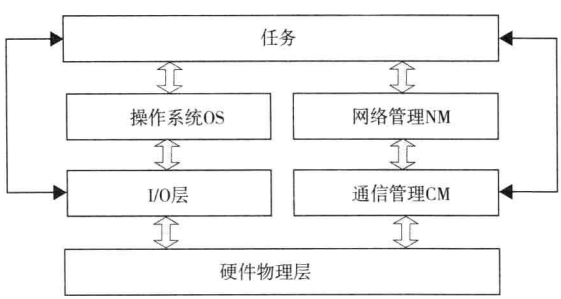


图 1-1 系统架构的组成部分

通过图 1-1 中各个部分的描述不难看出系统架构可以使用户不依赖

于单一的供应商，这样有助于覆盖一个大范围的系统实现目标平台。此架构中的操作系统为静态的，不能对系统资源进行动态处理，但可为每个任务应用程序生成单独的内核，支持高度独立的目标硬件。此架构所遵循的标准中没有定义动态内存管理和任何类型的输入 / 输出的服务，但是它支持任何类型数据的输入 / 输出。

1.3　系统架构工具链

在本书中，按照开放式系统架构的方法开发实现一个汽车上的软件程序需要运用相应的开发工具——ProOSEK。ProOSEK 融入了 tresosTM 启动环境，需要遵循 tresosTM 的系统架构工具链，如图 1-2 所示。

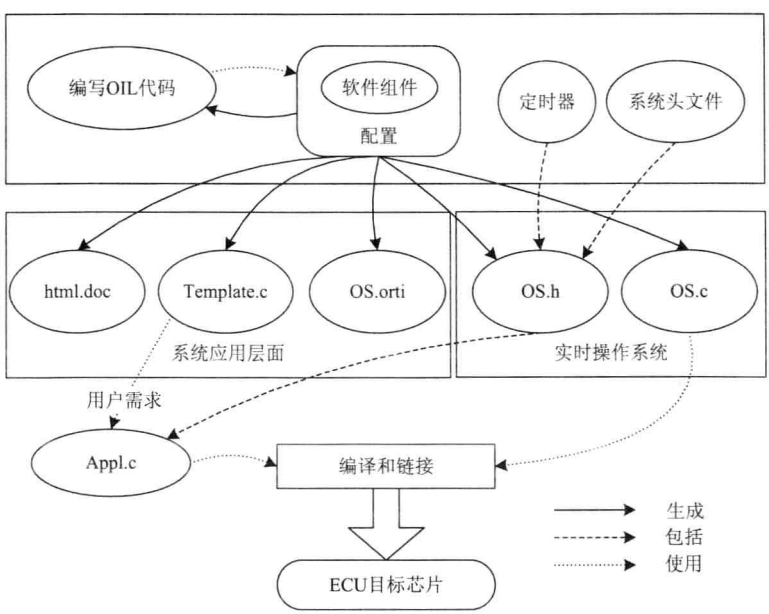


图 1-2　tresosTM 系统架构工具链

首先，需要在 ProOSEK 中对汽车软件开放式系统架构的各个组成部分进行配置（可使用 OIL 语言）并加载头文件和定时器，在编译完成后会生成 html.doc、Template.c、OS.orti、OS.h 和 OS.c 五个文件。然后结合 Template.c 和 OS.h 添加相应的应用代码得到 Appl.c。最后把经过编译 / 链接的 Appl.c 下载到目标芯片，从而实现汽车电子软件的开发。

按照 tresosTM 系统架构工具链进行开发可实现对内核的优化，可为应用程序选择足够的调度并衍生出对具体硬件的支持，可使用事件、计数器和多种定时器模块，从而可实现系统架构为每个应用程序生成单独的内核，可实现无需用户干预情况下的栈自动共享，可检查资源的一致性和可信性。tresosTM 系统架构工具链能够更好地适应实际的需求，其代码中只包含应用程序所需的功能，其数据结构都是静态的，可用于对服务进行管理。tresosTM 系统架构工具链中的配置工具支持确定的任务属性（如：类型、激活类型、优先级），能够实现对调度策略和错误检测级别的设置，能够连接具有中断源的 ISR，能够在应用程序的报警器到期时执行定义的动作，能够连接报警器和计数器，能够对应用任务中的事件和资源的可用性进行定义。

1.4 操作系统的体系结构

汽车软件开放式系统架构中的操作系统与我们常见的操作系统一样，都是为相互独立的应用程序提供了一个建立在处理器之上的基础环境。本书所描述的汽车软件开放式系统架构中的操作系统可控制多个并行进程的实时执行。此操作系统为用户提供了一个接口集合。这些接口

由竞争处理器的实体所使用。具体来说，分为两种类型的实体：任务（基本任务和扩展任务）和由操作系统管理的中断服务例程。控制单元的硬件资源可由操作系统服务管理。这些操作系统服务可以由应用程序或操作系统的内部接口调用。系统架构规范定义了三个处理级别：任务级别、调度逻辑级别和中断优先级别。根据用户分配的优先级来执行任务的调度（非 / 完全 / 混合抢占式调度），如图 1-3 所示。在执行开始时运行时间上下文被占用，任务完成后，将被释放。

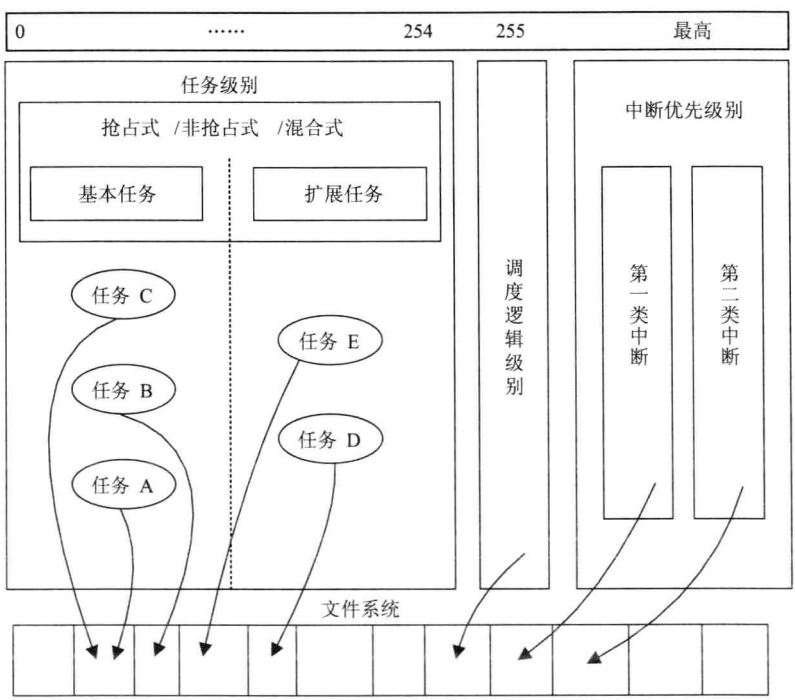


图 1-3　操作系统处理级别

按照图 1-3 的描述，相应地制定了以下优先级规则：（1）任务的优

先级是静态的，并且由用户指定；（2）中断比任务的优先级高；（3）对于任务优先级和资源上限优先级顺序，数字越大优先级越高；（4）中断服务程序根据具体实现和硬件体系结构分配中断优先级；（5）中断服务例程包含静态分配的中断优先级；（6）中断处理优先级由一个或多个中断优先级组成。在汽车软件开放式系统架构中为了处理任务和中断例程，而把处理级别定义为一个连续范围内的数值，具体执行过程中必须通过操作系统优先级映射硬件优先级来实现。

值得注意的是，调度的优先级分配仅仅是一个逻辑上的概念，它可以被执行而不用参考优先级。此外，系统架构规范中没有规定任务优先级与一个特定的微处理器架构的硬件中断级别有关的任何规则。

1.5 ProOSEK 开发环境

ProOSEK 是由 Elektrobit 公司开发的 OSEK OS 系统开发环境，包括图形化配置界面和整合了 ProOSEK 解决方案的文件结构。为了在使用环境中进行开发，需要了解不同工具是如何在一起工作的。此外，用户需要熟悉 ProOSEK 自带的一个预定义的目录结构。另一方面，有经验的用户可以更改默认设置的工作方式，以满足自身需求。ProOSEK 图形化配置器不再是一个独立的应用程序，它融入了 tresosTM 启动环境，支持 ProOSEK 插件，这是 tresosTM 基本安装所附带的。

ProOSEK 内部所具有的操作系统是一个面向静态的标准系统。由于没有动态创建系统对象的方法，所以，系统资源必须静态分配。一旦明确了所需的所有系统对象，将自动生成 ProOSEK 系统并与应用程序进行链接。配置提供了所有类型的系统对象以及加载，并用 OIL 格式保存配置。配置为所有类型的系统对象提供了图形化编辑器。

此开发环境为现有的 OIL 配置生成器提供以下功能：(1) 能够生成一个 HTML 配置文件；(2) 能够对所实现的配置进行一致性和完整性验证；(3) 能够自动生成应用程序的模板；(4) 能够自动计算出系统中数据结构需要的 RAM 空间大小；(5) 能够自动生成系统架构运行时接口 ORTI 数据；(6)能够根据用户的配置自动生成一个 ProOSEK 内核。此外，开发环境的生成器也被集成到 tresosTM 启动环境中，也可以通过命令行接口进行操作。ProOSEK 的开发流程如图 1-4 所示。

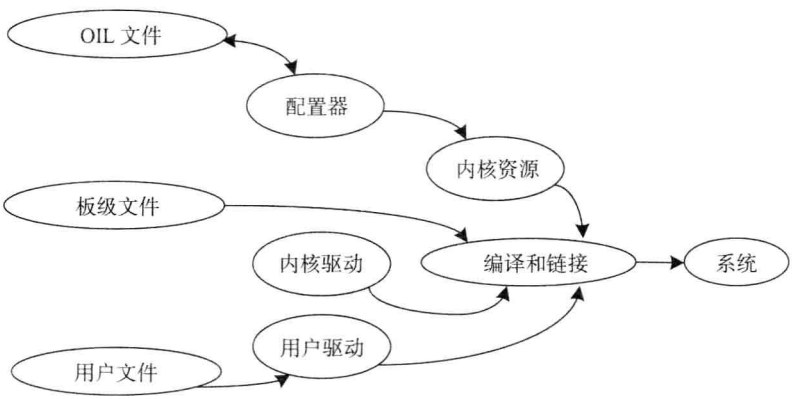


图 1-4　ProOSEK 开发流程

对于图 1-4 里涉及的 OIL 文件、板级文件和用户文件都可以根据具体的开发需求由有经验的用户对其结构进行修改。所涉及的环境变量 OSEK BASE 位于根目录 TRESOS BASE/ ProOSEK 里。图 1-4 中的配置器是开发过程中用到的一个主要工具，它是一个图形化的工具，允许系统根据设置的要求进行灵活配置，在完成配置之后可以使用 OIL 语言对其详细内容进行保存。此配置器也可以用来检查一个具体应用程序中所有配置的一致性并生成模板文件，这些功能在编写应用程序时非常有用。

在 ProOSEK 中默认使用的文件结构有以下 7 种：

（1）TRESOS BASE/ProOSEK/src 用于保存某些体系构建应用程序所需的额外文件。

（2）TRESOS BASE/ProOSEK/boards 目录里保存着板级的具体信息。

（3）TRESOS BASE/ProOSEK/data 目录包含子目录，用于保存配置数据文件。

（4）TRESOS BASE/ProOSEK/include 目录包含产生内核和板级信息所需头文件的子目录。

（5）TRESOS BASE/ProOSEK/make 目录存放一些基本的 makefile，它们也被用来构建示例。

（6）TRESOS BASE/ProOSEK/bin 目录包含配置和 make 工具，存放用于构建示例应用程序所需的一些可执行文件。

（7）TRESOS BASE/ProOSEK/demos 包含一些示例代码。这些示例对于学习如何使用 ProOSEK（或编写应用程序）很有帮助。

ProOSEK 开发环境在首次启动时会显示 tresos 启动窗口，见图 1-5。

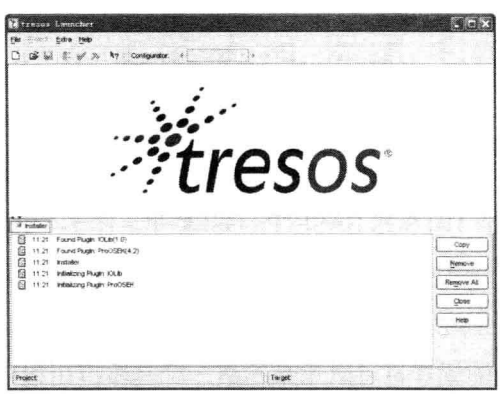


图 1-5　tresos 启动窗口

在 ProOSEK 开发环境的配置器窗口顶部可以找到应用程序开发时所需的菜单栏和工具栏，见图 1-6。菜单栏提供了访问各种配置器和生成器的元素。工具栏为最常用的功能提供了快捷方式。状态栏位于开发环境窗口的底部，用于显示当前正在配置的目标系统和目前正在编辑的 OIL 文件名称等具体内容。

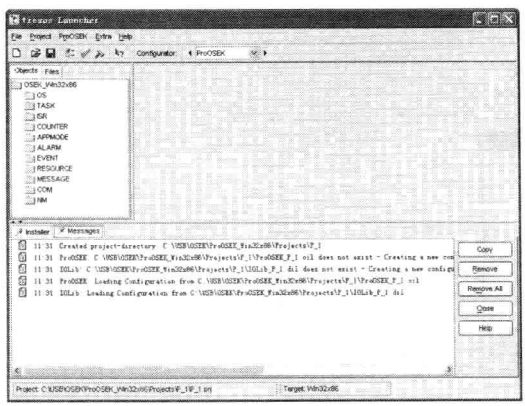


图 1-6　配置器主窗口——对象（Objects）视图

在图 1-6 中显示了配置程序的窗口，主要用于 OIL 文件加载或创建一个新的 OIL 文件。此窗口分为两个窗格，左窗格中可以选择两种视图——对象（Objects）和文件（Files）。在图 1-6 中显示的是对象视图，该对象视图显示具体架构提供的所有对象，通过双击这些对象，可以看到正在编辑的 OIL 文件中定义的所有实例。如果当前没有实例，则可以通过右击对象来创建一个新的实例。而在图 1-7 中显示的是文件视图。

文件视图可分成多个 OIL 文件系统，这个视图显示这些文件的布局。例如，在图 1-7 中显示的是 OIL 文件 ProOSEK_P_1.oil。每个 OIL 对象文件内定义了各种的 OIL 对象。这与在一个 OIL 文件中定义所有的 OIL

对象是相同的。OIL 对象文件能够提高所开发的应用系统的直观性和可理解性。主显示区的右窗格用来显示和编辑对象的属性。在本书的后续章节中将对各种对象的属性进行详细描述。

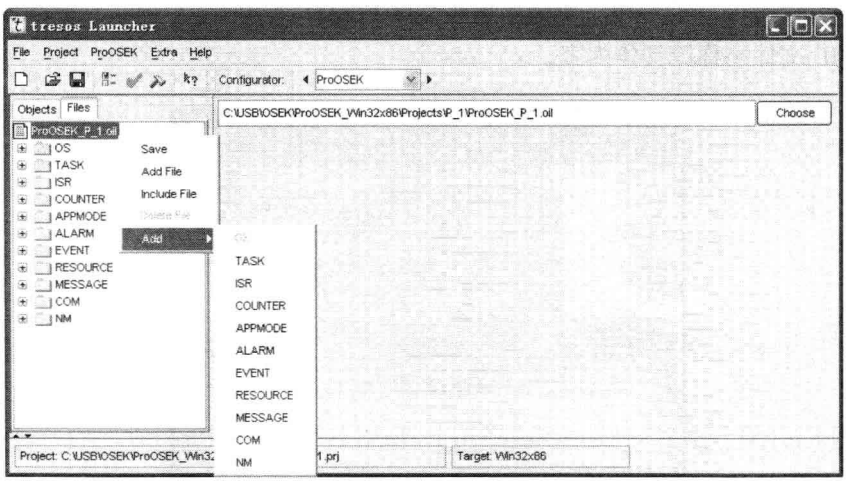


图 1-7　配置器主窗口——文件（Files）视图

在配置器窗口底部可以找到消息（Messages）窗口。在具体应用程序开发时所有执行动作的输出都被写入这个窗口，如：错误消息、有关生成的文件或一般的系统状态信息等。