冶金自动化研究设计院硕士研究生论文

基于模型的设计方法在冶金电炉智能控制中的应用

Application of Model-Based Design to Intelligent Controller on Metallurgic Electric Furnace

专 业 名 称：控制理论与控制工程

研 究 方 向：基于模型的设计

指导教师姓名：房庆海

学 生 姓 名：张德龙

申请学位级别：硕士

论文提交日期：2010年1月

原 创 性 声 明

本人郑重声明：本人所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

作者签名：张德龙　　日　　期：_____

关于学位论文使用授权的说明

本人完全了解冶金自动化研究设计院有关保留、使用学位论文的规定，即：自动化院有权保留送交论文的复印件，允许论文被查阅和借阅，可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后遵守此规定）

作者签名：张德龙　导师签名：_____

日　　期：_____　日　　期：_____

摘　要

该课题旨在研究基于模型的设计方法的实现技术及其在自主研发的冶金电炉智能控制板固件开发中的应用，着重研究自动代码生成的冶金电炉智能控制板固件架构。研究在 MATLAB/Simulink 软件平台上实现的基于模型设计的过程中具体而关键的技术细节，为 863 课题“冶金电炉智能控制系统”技术研究的核心理论成果转化为工业产品提供技术方案。

文章首先综述了基于模型设计方法在嵌入式固件开发中的优势并分析了冶金电炉智能控制系统的特点。接下来研究了这种设计方法在 MATLAB/Simulink 软件平台上实现的技术细节以及对自主设计的开发板进行固件开发的详细流程。然后讨论了基于模型设计方法在冶金电炉智能控制系统设计上的具体应用，以此方法实现了智能温度预报功能模块的设计开发。最后对课题的研究情况进行总结并对后续工作进行展望。

该项研究为在 MATLAB/Simulink 软件平台上对自主研发的 DSP 开发板固件设计提供实例参考，可为工业自动化嵌入式系统研发提供借鉴产品。研究表明，基于模型的设计方法加速了系统开发进程，具有很强的可移植性，能有效保证预期设计目标的全面实现，节约开发成本，最终产品功能易于拓展、便于维护和升级。

关键词：基于模型的设计；嵌入式实时固件；DSP；Simulink；智能控制器

ABSTRACT

The subject is intended to study technical realization of model-based design method and its application to independently developed intelligent controller on metallurgic electric furnace. Detailed and specific techniques engaged in the workflow of model-based design implemented on MATLAB/Simulink software platform, which are crucial on the process of transfer between theoretical research results of core technologies and industrial products, are to be used on developing firmware on the controller as guideline on working out practical scheme.

First of all, advantages of developing embedded firmware with model-based design methodology were summarized and specialties of intelligent controller on metallurgic electric furnace were analyzed. Second of all, detailed realization techniques of model-based design on MATLAB/Simulink software platform and specific workflow were studied. And then, application of this design method on intelligent controller on metallurgic electric furnace were discussed, intelligent temperature prediction module was developed with this method. Finally, the study was summarized and the following task was prospected.

The techniques studied in the subject provided practical case of targeting custom boards based on TI's DSPs. It has been proved that the application of model-based design implemented in MATLAB/Simulink soft platform on the intelligent controller firmware has provided advantages as speeding up development, guaranteeing whole achievement of pre-design expectations, reducing costs. Besides, the embedded product is benefited from features of maximum portability and configurability, easy extensions and upgrade, convenient maintenance, etc.

Keywords: Model-Based Design; Embedded real-time Firmware; DSP; Simulink; Intelligent Controller

目 录

第一章 绪论

1.1 概述

本课题根据《冶金电炉智能控制板规格设计书》及《冶金电炉智能控制系统功能规格设计书》，研究基于模型的嵌入式系统固件（Embedded System Firmware）设计方法在开发智能优化控制系统的冶金电炉智能控制板中的应用技术，把863项目“冶金电炉智能控制系统”的核心技术研究成果实现在冶金电炉智能控制系统硬件平台上，应用到工业生产现场，满足技术和经济指标的要求。

冶金电炉智能控制系统应用领域的特殊性要求冶金电炉智能控制板具有普遍性、特殊性以及易于拓展升级的特点。钢铁企业的电炉运行环境大致相同，冶金电炉智能控制系统在宏观架构上一致。冶金电炉智能控制板完全嵌入冶金电炉智能控制系统内部，运行预定义的任务，实时采集现场数据并给出控制信号，完成与上位机以及热备PLC系统的通信等。但不同的电炉现场参数略有差异，某些实现细节上需要根据具体对象做出适当调整，即每个电炉系统均有其特殊性。冶金电炉在钢铁冶炼过程中长期连续运行，其间钢铁企业对电炉进行维护改造，需要系统能够易于维护和升级扩展。冶金电炉智能控制板的上述特点对系统固件开发提出很大要求，加重了设计的复杂度及难度。与此同时，希望开发文档易于管理和传承，研发的技术成果及产品不会因技术人员的流动而变得难以继承和维护[1][2]。

1.2 基于模型的设计方法

随着硬件和存储器的价格日益降低，功能日益强大，嵌入式系统更加普及。但与此同时，系统设计的复杂度和难度也越发提高。传统设计方法中，设计结果和预期功能和性能之间的误差只有在研发后期系统集成的时候才能被发现，而这些误差确是在设计初期引入的。面对这些挑战，嵌入式系统工程师们必须采用更高效的软件和硬件开发方法。基于模型的设计就是解决嵌入式系统设计的一种有效的方法论[3]。

1.2.1 什么是基于模型的设计方法

基于模型的设计（Model-Based Design），作为高效嵌入式软件设计的方法论，借助通用开发环境中的一套工具实现设计、验证、程序功能划分以及自动生成代码。

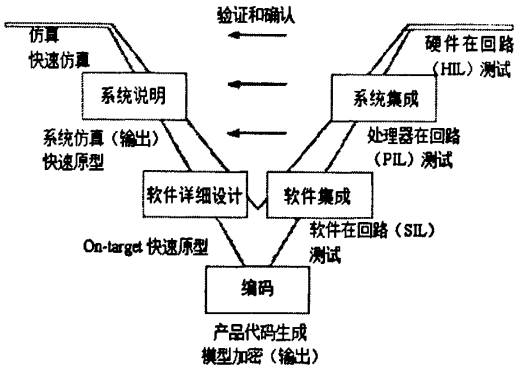


图 1 基于模型设计的各组成部分及系统开发的 V 模型

图 1 表示基于模型设计方法的组成部分以及使用该方法进行嵌入式系统开发过程的 V 模型[17]。基于模型的设计方法以模型为核心，首先建立系统功能的概念模型；通过仿真改进概念模型；一旦模型经验证完全实现了系统目标，就形成“可执行的规范”，成为后续工作的基础；最后一步使用自动代码生成功能，将模型在硬件上实现。开发过程的每一步，都可验证模型是否实现了性能指标[6]。

V 模型充分对该实现过程进行了说明。V 左侧和代码生成步骤有关，包括需求分析、系统总体设计、详细的软件设计以及编码。右侧部分则集中在左侧相关步骤的验证和测试，包括软件集成和系统集成。该模型涵盖了传统系统工程的各个环节，包括定义需求、系统级设计规范、子系统设计、软件实现、子系统集成及测试、系统级集成及测试以及全部集成和测试。但 V-模型的每个环节均覆盖传统设计方案中的多个环节，并且，V-模型中尤其强调对设计的验证过程[4][7][4]。

1.2.2 基于模型的设计方法在多领域的应用

基于模型的设计方法（Model-Based Design）采用数学的和可视化的技术手段，解决复杂控制系统设计相关的问题，是作为解决控制系统设计固有的难度和复杂度的手段而出现的，广泛应用于航天、国防、汽车、通信、消费类电子和医疗电子设备。下面列举某些实例，简要说明基于模型的设计方法在某些领域的应用及取得显著效果。

1.2.2.1 汽车电子领域

基于模型的设计方法以及成为开发汽车嵌入式软件的首选方法，尤其是在汽车电控单元（ECU）的开发方面。它改善了技术规格定义阶段、设计阶段和实现阶段。汽车工程师采用基于模型的设计方法，将模型用作可执行的技术规范，设计算法并用模型分析系统的动态行为，模拟系统部件与环境条件。降低了对造价昂贵的实物原型的需求。追踪并验证功能是否被实现并满足系统要求，对模型的覆盖进行分析，从设计结构角度进行完整性评估。评估从这些模型中自动生成代码已经成为了公认的 ECU 软件开发方式[7]。

通用汽车公司 (GM) 已使用 The MathWorks 的基于模型的设计 (Model-Based Design) 工具开发出双模式混合动力总成控制系统。GM 利用科学计算和建模仿真的软件工具（包括 MATLAB 和 Simulink），在 9 个月内设计出了动力总成原型，这比预期开发时间短了 24 个月。

东风电动车辆股份有限公司(DFEV)使用 The MathWorks 的 Model-Based Design（基于模型的设计）工具，历时 18 个月开发出了一种电池管理系统。这种新型电池管理系统已安装到东风 EQ6110 混合动力电动城市公交客车上。与普通城市公交车相比，这种车辆降低了 30% 的油耗，同时减少了废气排放。

1.2.2.2 通信领域

软件无线电（software radio）在一个开放的公共硬件平台上利用不同可编程的软件方法实现所需要的无线电系统，简称 SWR。在目前的条件下可实现的软件无线电，称作软件定义的无线电（Software Defined Radio，SDR）。

大多数 SDR 设计工程师一直使用传统的设计方法：将系统架构师定义的规范细化为文档，用这些文档指导专注于信号处理或射频工程领域的项目团队，然后由这些团队定义硬件、设计电路、编写软件、运行仿真、测试并生成大量数据，这些数据被用来验证最终实现结果是否满足规范要求。尽管软件无线电(SDR)的指导原则是“开发一次，随处运行”，但是每当硬件发生变化时，所有开发经常要重新开始。基于文本规范的传统开发模式已无法满足 SDR 硬件和软件的可移植性要求。

部分 SDR 设计工程师采用基于模型的设计思想的设计新方法，通过建立可执行的规范、IIM 和 ISM 模型，并维护原始波形规范的可跟踪性、确保在整个开发过程中不断验证，实现了在不同的硬件、软件以及 SCA 核框架平台上支持自动代码生成和代码可移植性，解决了这些挑战并充分发挥 SDR 的潜能。

1.3 冶金电炉智能控系统

冶金电炉是流程工业中的重要设备，也是大型能耗设备。对提高钢铁质量、产量以及钢铁行业的节能减排都具有很大的意义。通过分析目前国内外电炉冶炼工艺现状，得知我国的电炉工艺模型开发及应用状况与国外还有一定差距，这集中体现在冶金电炉的控制系统上。

课题“冶金电炉智能控制系统（863 计划）”研究了冶金电炉的核心技术[2]：基于专家规则与计算智能的冶炼过程温度预报技术，电极升降复合智能控制技术，以及专家系统与分析模型结合的能量输入优化技术。在以上核心技术基础上，进一步研究其实现技术，开发了智能优化控制系统。控制系统架构中，原系统上位机及下位机均采用工控机，两级之间通过 TCP/IP 方式实现通信连接，结构复杂，体积庞大，成本较高。新设计的冶金电炉智能控制板是基于 TMS320C6713 的硬件平台，采用 CPCI 形式与上位机联结，通过 HPI 进行数据通信，实现原下位机的功能。采用智能控制板后系统结构更为紧凑，降低了成本，更提高了整体性能，对加速控制方案产品化及运行维护具有重大意义。

1.3.1 冶金电炉智能控制板固件功能

固件功能主要有：

(1) 电炉仿真器

电炉仿真器是一个采用延时网络技术（TDNN）的 BP 神经网络模型（NN），根据 EAF 的过去及当前状态，预测下一时刻 EAF 状态，是电炉调节器的基础。

(2) 电炉调节器

电炉调节器产生智能调节器系统的信号输出，实现电极升降的优化控制，采用标准的三层前向 BP 神经网络。

(3) 复合优化综合控制

为了提高设定点的跟踪精度，降低三相功率的不平衡度，减少液压阀的动作频率，本系统采用人工神经元网络和模糊控制有机结合的电极升降控制算法，同时也引进电弧电极升降控制的专家规则，对控制系统的最终输出进行综合判定。

(4) 高速数据采集与处理

采集现场数据并进行处理，降低噪声。主要的处理数据包括：电压/电流有效值；

- 功率因数；
- 有功功率/无功功率/视在功率；
- 累计电耗；
- KW 因子；
- PERSSION 因子；
- 谐波因子；
- 其它所需的参数。

(5) 数据通信

主要负责与热备 PLC 之间的数据通信。

1.3.2 冶金电炉智能控制板硬件结构

冶金电炉智能控制板以 32 位浮点运算处理器 TMS320C6713 DSP 构建硬件平台，采用 CPCI 方式与 X86 计算机（上位机）连接，与上位机的通讯单元主要包括 CPCI 总线热插拔控制模块以及 CPCI 总线桥接模块。系统设置 2M 容量的 FLASH 存储程序，16M 容量的 SDRAM 运行程序和暂存数据。与外部接口单元主要包括数字量输入输出模块、模拟量输入输出模块 RS232 以及以太网通信模块。

1.4 各章内容概述

该课题研究冶金电炉智能控制板固件开发使用的技术，研究基于模型的设计方法在 MATLAB/Simulink 软件平台的实现。但这是一项综合而且复杂、涉及多学科、多专业的问题，本文难以面面俱到。课题研究的目标在以下几个关键点上：

第一章介绍该课题背景来源，阐明冶金电炉智能控制板固件开发采用基于模型设计方法的缘由，确定采用的软件平台。进一步介绍了冶金电炉智能控制板的功能及硬件结构。

第二章研究采用基于模型的设计方法、应用 MATLAB/Simulink 软件平台、以 DSP 为处理器的硬件平台的固件开发环境以及开发平台各部分的主要功能和作用；自主研发的 DSP 产品原型在该综合开发平台上的设计步骤和方法，以及关键参数的设置和意义；讨论从模型自动生成的代码的运行方式、结构和文件组成，用户代码

与自动生成代码的集成，研究基于模型设计的验证技术，在设计过程中的应用，研究实时嵌入式系统中关键概念在该综合开发平台中的实现方式，如任务、异步事件处理以及调度等；

第三章内容讨论该项技术在冶金电炉智能控制板固件开发中的应用情况，应用该项技术建立了智能温度预报模型；

第四章内容对课题的研究情况进行总结，对后续工作进行展望。

第二章 基于模型设计方法的实现

嵌入式系统功能越来越复杂，软件开发已经成为一项浩大的工程。从开发方案的制定、功能设计到具体实施，每一环节都必须严格把关，否则研发项目就会以失败而告终。软件开发的过程是由方法论和实施工具构成的。基于模型的设计方法是嵌入式软件设计的方法论，能满足我们产品研发的需要。实施工具的选取也是非常重要的，优秀的开发平台是研发任务顺利完成的必要保障。

2.1 开发平台和流程

MathWorks 用于基于模型设计的产品提供可视化的交互环境，在该环境中可以建立、管理以及仿真模型。通过基于模型的设计，项目开发组成员可以不使用快速原型产品或实时目标就能开始评估软件设计。MathWorks 基于模型设计的环境允许工程师建立物理系统的数学模型，设计软件、建模其行为，然后仿真整个系统预测并优化其性能。仿真在硬件和软件建立之前进行，验证设计确实产生了正确的结果。模型的本身就成为一种规范，可以从模型生成实时代码，用于测试、快速原型以及嵌入式实现。因此，避免了手写代码的工作，从而减少了潜在的错误[8][9]。

该环境图形化、结构化的特性项目开发组成员建立动能性的模型，准确地建立文档，并且有效地沟通他们的设计成果，减少了由于误解和误会而带来的风险。系统要求和规范的改变以及修正可以便利地纳入模型中，通过仿真完整地评估，自动地反映到最终实时嵌入式软件中。

2.1.1 开发平台

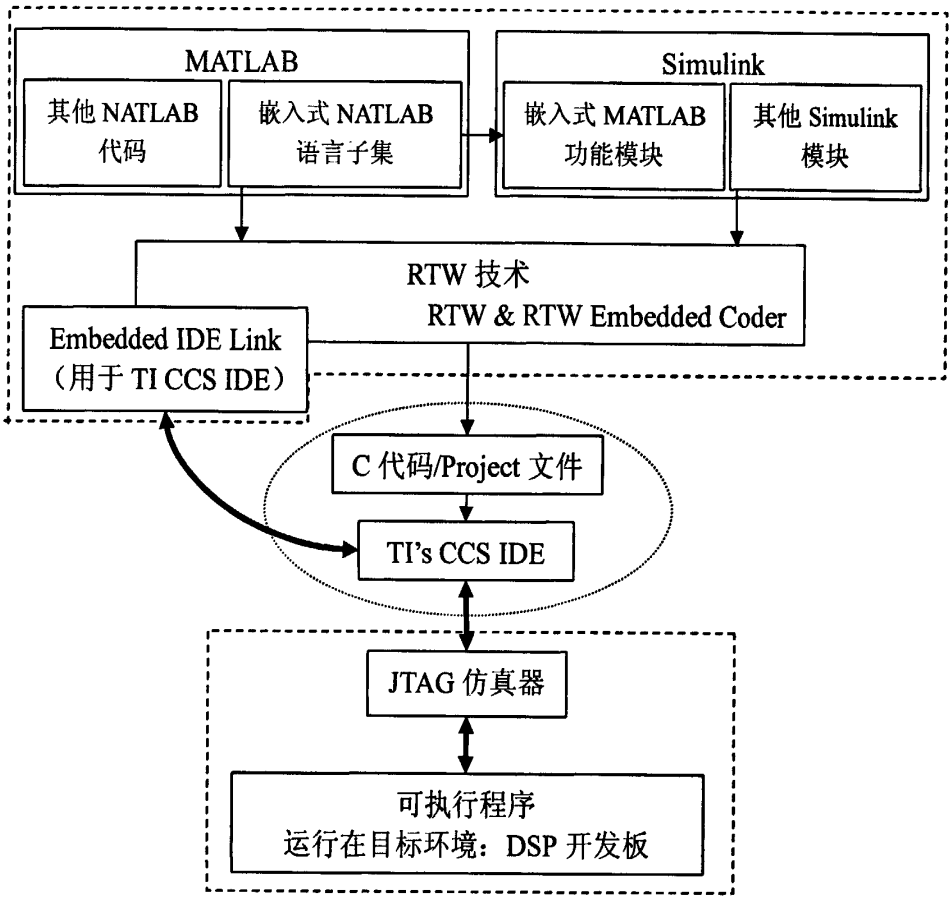


图 2 MATLAB 辅助 DSP 系统固件设计综合开发平台

上图是在 MATLAB/Simulink 平台上采用基于模型设计方法辅助由 TI DSP 为处理器的硬件平台进行固件开发的综合开发平台示意图。该综合开发平台由三部分构成：MATLAB/Simulink 软件平台，TI 的 CCS 集成开发环境，以及由 JTAG 和目标板组成的硬件平台。在 MathWorks 提供的软件平台上进行算法开发，通过仿真验证确定开发的算法模型满足系统性能的要求；采用 RTW 技术（RTW 及 TRW Embedded Coder 产品）结合 Target Support Package 生成算法模型的嵌入式 C 代码和片上外设的驱动程序；Embedded IDE Link（For Use With TI's CCS IDE）起到 MATLAB/Simulink 和 CCS IDE 之间的连接作用，并支持将 RTW 生成的 C 代码自动创建 CCS IDE 中的项目文件。通过 Embedded IDE Link（For Use With TI's CCS IDE），

还可以从 MATLAB 环境中编译、调试 CCS 中的项目并控制目标开发板上实时嵌入式程序的运行，通过 CCS 与目标开发板交换数据等操作。该项技术大大简化了 DSP 嵌入式软件开发。

2.1.1.1 硬件平台

硬件平台由 JTAG 仿真器和开发板以及外围设备组成。在为产品开发的硬件原型可用之前，一般先选用和产品相似的开发板进行软件开发和调试。也可以用软件仿真器进行一些功能性的验证和调试。

2.1.1.2 Code Composer Studio（CCS）IDE

CCS IDE 是用于 TI DSP、微处理器和应用处理器的集成开发环境，包含了一整套用于开发和调试嵌入式应用程序的工具。CCS IDE 包含适用于每个 TI 器件系列的编译器，源代码编辑器，项目开发环境，调试器，优化器，仿真器以及许多其它的功能。CCS IDE 提供的单用户界面能使用户完成应用程序开发流程的每一步。CCS 主要部分为：

(1) CCS 代码生成工具

(2) CCS 集成开发环境

(3) DSP/BIOS 插件程序和 API

(4) RTDX 插件、主机接口和 API

2.1.1.3 MATLAB/Simulink 软件平台

MATLAB/Simulink 为综合开发平台的主要组成部分，它包含许多领域的多种工具箱和模块库，可根据特定项目选用适当的产品组件进行算法开发，具体信息可查阅相关文献。此处仅对该项技术中的部分关键产品的功能和祈祷的作用作简要说明：

(1) RTW 技术

RTW 技术是 Simulink 的重要组成部分，是 Simulink 代码生成的基础，为在 Simulink 环境中建立的图形化算法模型或用嵌入式 MATLAB 语言子集建立的程序化的算法生成 C 或 C++代码和可执行程序。整个算法模型或单个子系统可以生成符

合 ASNI/ISO 标准的 C 代码。根据系统目标文件（System Target File）选项指定的目标系统，生成的代码能够运行在相应的微处理器或实时操作系统上。RTW 还能将生成独立的 C 代码编译成可执行的程序，用于算法的开发和测试。生成的代码还可用于许多实时的和非实时的应用程序中，包括加速仿真、快速原型以及硬件在环测试（hardware-in-the-loop testing）。用 Simulink 模块及内置的分析工具可调节或检测生成的代码，或者在 MATLAB 及 Simulink 环境之外运行代码并与之交互[10][17][18]。

Real-Time Workshop Embedded Coder 扩展了 RTW 代码生成的功能，扩展的这些功能对嵌入式软件开发尤为重要。使用 RTW Embedded Coder 附加产品，不仅可以使用 RTW 技术的各个方面，还可以生成清晰、高效的专业水平代码[19]。

RTW 的相关参数在模型的配置参数中进行设置：**Configuration Parameters\ Real-Time Workshop**。该选项卡下面有 Report、Comments、Symbols、Custom Code、Debug、Interface、Code Style、Templates、Data Placement、Data Type Replacement、Memory Sections 以及 Embedded IDE Link 等子项。

(2) Embedded IDE Link（TI CCS）[20]

Embedded IDE Link 的功能是将 MATLAB/Simulink 和 DSP 嵌入式软件的集成开发环境联接起来。通过 Embedded IDE Link，可以生成、编译、测试以及优化将部署到用于原型研制或实际产品中的嵌入式代码。Embedded IDE Link 自动完成调试、项目生成以及运行在嵌入式处理器或集成开发环境（IDE）提供的指令集模拟器上的目标代码验证等任务。此外，可以在 MATLAB 和 Simulink 中创建测试工作台（test bench），用于处理器在环测试方式（Processor-in-the-Loop testing）验证手写代码及生成的代码。

Embedded IDE Link/For use with TI's CCS 专门针对 TI 的 DSP 系列产品，它与 RTW 及 RTW Embedded Coder 联用，能在 Simulink 中以图形化方式创建的算法模型顺利地转换为 CCS 中完整的项目（projects）文件，并能进一步编译成在嵌入式处理器上可执行的实时独立程序。所生成项目不仅包含生成的应用程序的 C 代码，还有嵌入式软件架构需要的初始化、调度以及处理器实时执行的应用程序代码的管理。硬件平台改变时，只需要改变目标选择模块中处理器及开发板的相应参数，即可重新生成适合新硬件平台的嵌入式 C 代码，几乎不需手动改写代码，具有极强的可移

植性。Embedded IDE Link 主要功能有：

1) 用 MATLAB 和 CCS 环境中的项目进行交互

MATLAB 与 CCS 集成开发环境的自动接口（Automation Interface），仅与 MATLAB 单独使用（不用 Simulink），即应用 MATLAB 语言编写脚本文件，自动调试和分析 CCS 中的项目，在 MATLAB 环境中察看运行时的数据并通过 MATLAB 改变目标板上运行程序的参数。Embedded IDE Link 支持所有 CCS 支持的 TI 处理器，接口包含两种方式：调试模式（Debug Mode）和 RTDX 模式。

2) 与 Simulink 联合仿真

应用 Embedded IDE Link 和 RTW Embedded Coder，可以将 MATLAB 和 Simulink 的算法运行在目标处理器或集成开发环境提供的指令集仿真器上、仿真模型作为测试工作台进行联合仿真。算法部署到目标平台，PIL 仿真功能可以让开发者验证算法代码，而不必手工搭建一个嵌入式测试平台。

3) 用 RTW 进行项目生成和代码优化

项目生成（Project Generator）使用 RTW 以及 RTW Embedded Coder 将 Simulink 模型生成完整的 CCS 项目。这些产品生成应用程序的 C 源代码，并根据需要包含汇编代码以及链接描述文件。应用 RTW Embedded Coder，可以在 MATLAB 支持的 C2000/C55x/C6000 系列的 DSP 上生成代码并进行 PIL 测试。RTW 从 MATLAB 和 Simulink 模型生成与目标无关的算法 ANSI C 代码，与 RTW 联用时，Embedded IDE Link 为算法代码添加上实时的嵌入式框架代码，包括在目标处理器上实时运行的必备调度程序和中断处理机制。Embedded IDE Link 还添加处理器专用的优化表，自动将 ANSI C 代码尽可能地替换为优化的内部对象，减少手工修改代码，提高代码执行效率。

对于 MATLAB 及 Simulink 不能生成应用程序的项目及代码的处理器，RTW 仍然支持，但必须手工完成外设软件集成及项目创建的工作。

Embedded IDE Link 选项在模型的配置参数中进行设置：**Configuration Parameters\ Real-Time Workshop\ Embedded IDE Link。**

(3) Target Support Package[21]

Target Support Package 用于将 MathWorks 产品生成的实时程序代码部署到嵌入

式微处理器，微控制器，以及 DSP 上。以实时方式在特定目标上执行 RTW 生成的代码需要 RTW 生成针对特定目标的专用代码，包括 I/O 设备驱动程序和中断服务程序（ISR）。它使外围设备和实时操作系统与用 Simulink 模型、Stateflow 流程图以及嵌入式 MATLAB 语言子集结合起来，不必手写底层驱动程序和运行时代码。合成的可执行程序可以被部署到嵌入式硬件上，用于目标板上快速原型、实时性能分析以及现场生产。

Target Support Package/For Use with TI's C2000、C5000、C6000 模块集分别支持 TI C2000 系列，C5000 系列以及 C6000 系列的 DSP。MATLAB 对 TI 的 DSP 支持主要有以下几个方面：

1) 对 DSP 芯片级的支持。除了前面提到的特定硬件开发板，也支持使用 C2000、C55xx 以及 C6000 处理器的定制开发板（custom boards）。对定制开发板的支持需要使用 Embedded IDE Link 和 Texas Instruments® Code Composer Studio adaptor。只要采用 CCS 支持的 DSP 芯片，MATLAB 均能支持。内核支持库（Core Support library）提供输入/输出、通信以及访问 DSP 外设或板上外设的模块，比如 CPU Timer 和 Hardware Interrupt 模块。但板上外设的驱动程序需要用户自己开发，集成到模型中，在编译时一同转化为项目的代码。
2) 开发板级的支持。如 Avnet S3ADSP、DM6437、 C5510 DSK 和 C6713 DSK，不仅支持 DSP 芯片及片上资源，还支持板上外设，能直接生成专用的驱动程序，将模型生成的代码直接运行在目标板上。
3) 提供 TI 某些 DSP 专用的优化算法库，如 C28x 的优化模块 C28x IQmath 库等，IQMath 库可以通过 Simulink 的对话框修改这些模块的参数，生成高效优化的代码。
4) 用于 MATLAB 平台和 CCS 集成开发环境之间的联系，如 C2000 系列的 RTDX 以及 Host SCI Blocks。对选定的处理器系列支持片上外设和板上的外设以及实时操作系统。

2.1.2 开发流程

基于模型的设计的各组成部分。基于模型的设计中，开发过程围绕系统模型展

开——从需求获取及设计到实现和测试。该设计方案的核心是模型，在产品整个开发周期内强调验证的重要性。这种方案的简要开发流程如下[18]：

(1) 建立需求文档，验证需求

(2) 以书面形式确切表达解决方案的功能，建立“需求文档”，规定系统任务，以便复查和验证（需求文档影响代码生成过程）

(3) 开发模型可执行说明

(4) 开发详细软件设计

(5) 生成应用程序代码

(6) 集成并验证软件

(7) 集成、验证并校准系统组件

2.1.3 定制板的固件开发

冶金电炉智能控制板使用的 TMS320C6713DSP，可以使用 Target Support Package 为该目标板生成代码。C6000DSP 库中外设模块，比如 C6713 DSK DAC 模块，是其硬件特有的，在用户自主研发的硬件平台上不能运行。该工具箱提供的所有与硬件平台相关的模块都不能在用户自主研发的硬件平台上运行[12][21]。

2.1.3.1 典型目标板开发过程

通常，开发环境要求完成一系列的过程，以建立模型为开始，以生成适合目标的代码而结束。

(1) 建立转换为目标代码的算法或程序的 Simulink 模型：开发算法的方式有以下几种：

1) 用 MATLAB 代码，使用嵌入式 MATLAB 语言子集；

2) Simulink 模型；

3) 集成到 Simulink 模型中的代码

(2) 添加 Target Preference 模块到模型中。选择 Custom Board C6000 Target Preference 模块。

(3) 配置目标参数选项（Target Preference）模块。**Board type** 项输入 Custom，告诉系统正在开发 Target Support Package 不明确支持的开发板。映射存储

器段（segments），分配代码段（sections）和数据段（sections）到存储器段中，并设置其他特定目标选项。

(4) 设置模型的 Simulink 配置参数。注意，添加 Target Preference 模块到模型之后再进行设置。设定的参数对仿真及生成的代码都有影响。

(5) 将模型编译到目标。

2.1.3.2 对定制板的支持

自主研制的冶金电炉智能控制板并非 MATLAB 软件平台直接支持的开发板，需向模型中添加 Custom Board C6000 Target Preferences 模块。配置该模块的参数告知 Simulink 软件、Target Support Package 软件以及 RTW 软件目标处理器的类型和生成在目标上运行的代码的方式。对所有 C6000 处理器硬件平台目标的支持是通过 Target Preferences 模块来实现的。

表 1 可用于 TMS320C6713 硬件平台的目标参数选择模块

Target Preferences 模块	描述
Custom Board C6000	提供对以任何 C6000 处理器平台为目标的硬件配置访问。注意：该模块没有设置任何默认值。添加该模块到模型后，必须设置每个子窗口上的所有选项——Board Information，Memory Mapping，以及 Section Layout。
C6713DSK	设置以 C6713DSK 为目标的默认值。添加该模块到模型后，根据需要修改默认值。该模块的参数设置匹配开发板属性。

对那些不支持的开发板，这些 Target Preferences 模块提供了直接操作方式。由于某些和存储器映像特点及其他一些与处理器相关的属性，Custom Board C6000 仅与适用于 C6000 DSP。使用 Custom Target 和 Custom Target Preference 模块时，有些影响目标配置的选项：

(1) 用 C6000 Target Preference 对话框中 Memory 和 Sections 子窗口指定存储器分配（存储器映像）。设置与硬件最佳匹配的存储器映像。目标开发板采用 C6713 处理器，存储器的配置与系统支持的 C6713DSK 的配置大体一致，比如存储器大小、相同的 EMF 设置、相同的存储器段、以及相同的 Cache 配置。

(2) 若只用目标板片上存储器，在 TI C6000 Compiler 选项中为 Memory Model

选择 Near_Calls 选项。若使用特定目标板的外部存储器，为 Memory Model 选择 Far_Calls 选项。Memory Model 列表中的其他选项对数据和综合数据提供 Near 和 Far 分配的组合方式。

(3) 不要使用现有的 ADC、DAC、DIP 开关或 LED 模块，除非确定硬件平台与相应的 EVM 和 DSK 在所有关键部件是一致的。通常，ADC、DAC 以及其他目标专用模块是专为制定目标板设计的，用在不同的硬件平台上，会产生有些问题，不能正常工作。

(4) 使用超时运行通知（Overrun Notification）功能时，设置 TI C6000 runtime 的 Overrun notification method 为 Print_message。

(5) 使用 C6713 为处理器的定制目标，首先创建模型，添加并配置定制板 C6000 Target preference 模块，然后打开模型的 Configuration Parameters 对话框。

2.2 自动代码生成

本章的研究内容从宏观层级上把握开发完整功能的嵌入式应用程序的技术及简要的实现方案。该部分首先阐述从开发的模型到在目标板上可运行的程序的完整转换过程，进而分析了自动生成代码的结构及其运行机制和代码优化技术，接下来讨论了将用户代码集成到自动生成代码和项目中的实现方式，最后介绍了数据存储管理。

2.2.1 从模型到项目的实现

模型经仿真验证无误后，进一步生成代码。下图展示的是从模型到生成的源代码的过程[22]：

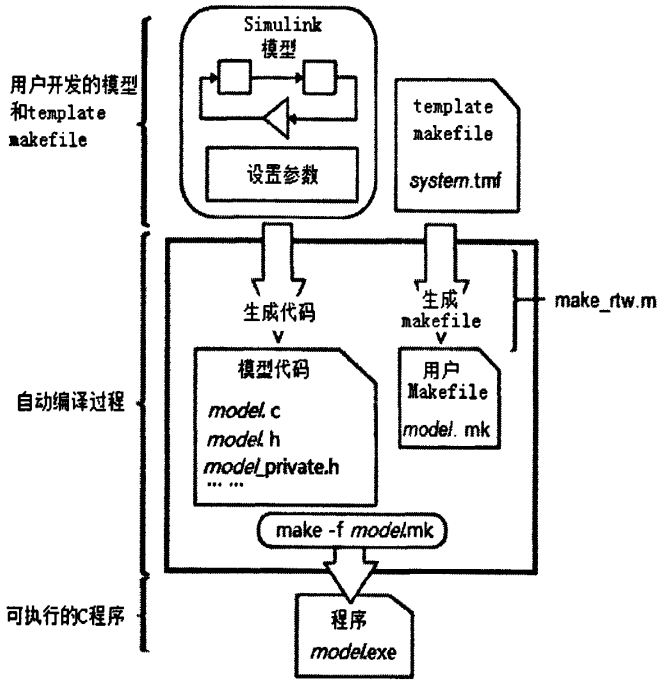


图 3 从模型到嵌入式固件的实现过程

从模型到源代码需要经过两个步骤——编译模型（Model Compilation）和生成代码（Code Generation）。构造（Build）过程以 Simulink 软件编译（Compile）模块图开始。在该阶段，Simulink

- 计算仿真参数及模块参数的值；
- 传导信号宽度及采样时间；
- 确定模型中各模块的执行顺序；
- 计算 work 矢量长度，比如在 S-函数中应用的 work 矢量；

Simulink 处理完这些，就编译一个代表这个模型的过渡文件。该模型的过渡描述文件以与语言独立的格式存储在 ASCII 文件 *model*.rtw 中。*model*.rtw 文件在格式上与 Simulink 模型(.mdl)文件相似，但只用于自动代码生成。*model*.rtw 文件是构造(build)过程下一阶段的输入。

RTW 代码生成程序（Code Generator）使用目标语言编辑器（Target Language Compiler，TLC）以及支持 TLC 的功能库将存储在 *model*.rtw 文件中过渡模型描述转换成针对特定目标的代码。TLC 编译的目标语言是一种解释性的编程语言，这种语言设计的目的是将模型描述转换为代码。TLC 执行一个有多个目标文件（target

files，.tlc 脚本程序)。TLC 脚本文件决定从模型生成代码的方式，以 *model*.rtw 文件作为输入文件。TLC

(1) 读取 *model*.rtw 文件；

(2) 编译并执行系统目标文件中的命令；

(3) 编译并执行模块级别目标文件中的命令；

(4) 记录 Simulink 模块图的源代码版本。

Embedded IDE Link/For Use With TI's Code Composer Studio 的项目生成程序为开发项目及生成代码提供了以下功能：

(1) 支持 TI CCS 软件自动项目构建，可以从 RTW 和 RTW Embedded Coder 产品生成的代码创建项目。创建的项目自动装载到 CCS 开发环境中的 CCS 项目中；

(2) 应用模型配置参数选项和处理器首选项模块参数配置代码生成；

(3) 从两个系统目标文件（ccslink_grt.tlc，ccslink_ert.tlc）中选择选择专门针对特定处理器的专用代码；

(4) 配置代码构建过程；

(5) 在处理器上动装载并运行生成的项目；

为 TI DSP 硬件平台生产项目，须向系统模型中添加 Target Preference 模块，自主研发的开发版选用 Custom Board，根据具体的处理器型号和板上资源配置相应的参数。向模型中添加该模块以后，Configuration Parameters 相关的参数也会自动设置，在 Real-Time Workshop 选项卡中自动将 System target files 配置为 ccslink_ert.tlc（需要 RTW Embedded Coder 产品的支持）或 ccslink_grt.tlc。

2.2.2 生成的程序结构及代码模块

为嵌入式目标生成代码需要有 RTW Embedded Coder 软件的支持，不同的应用目的生成的代码结构是不同的，如快速原型（Rapid Prototyping）的代码结构和嵌入式软件代码结构有所区别。RTW Embedded Coder 产品为嵌入式应用程序提供的结构如下[18][19]：

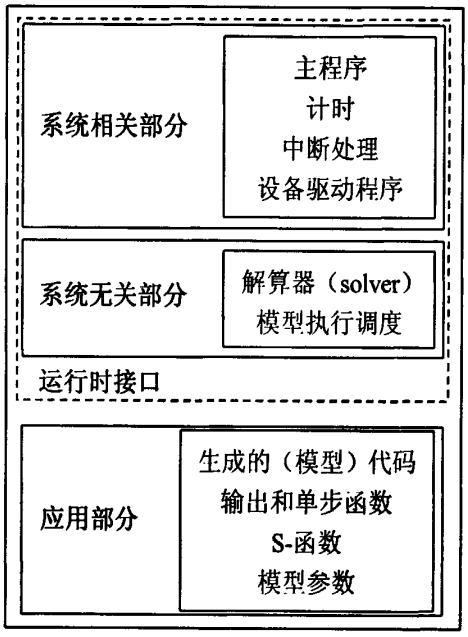


图 4 嵌入式程序结构

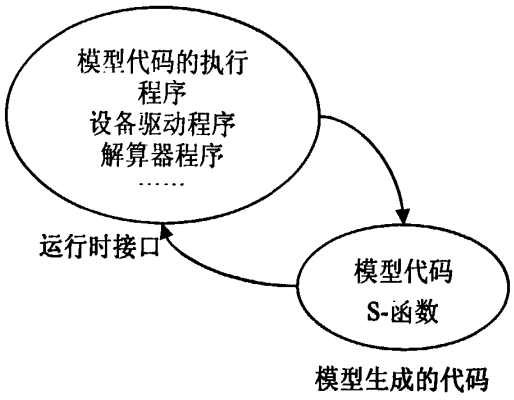


图 5 嵌入式程序运行概貌

而图 4 展示了生成代码的结构及组成部分，而图 5 从总体上以面向对象的视点展示了嵌入式程序的运行概貌。生成的代码由运行时接口（Run-Time Interface）和应用部分组成。运行时接口提供模型执行的驱动程序（model execution driver），由和系统相关的部分和与系统向独立的部分组成：

- 初始化（**Initialization**）：*model*_initialize 初始化运行时接口代码和模型代码；
- 模型输出（**ModelOutputs**）：*model*_output 调用模型中所有当前时刻运算的模块，产生模块的输出。
- 模型更新（**ModelUpdate**）：*model*_update 调用模型中所有当前时刻运算的模块，更新模块的离散状态或相似类型的对象。
- 模型终止（**ModelTerminate**）：*model*_terminate，在嵌入式应用软件中，该程序一般运行不到。但可以根据需要，设置该程序的行为。

应用部分是由模型生成的代码，是嵌入式应用软件运行的算法是集中体现。该段代码由运行时接口程序调度反复执行。

RTW Embedded Coder 软件在当前工作目录下创建存储生成源代码的构建目录（build directory），在系统目标文件为 ccslink_ert.tlc 时，文件夹名称为 slprj 和 *model*_ticcs。该编译目录还包含目标文件（object files），makefile，以及在代码生成

过程中生成的文件。生成的文件及文件内容的描述如下表所示：

表 2 RTW Embedded Coder 文件包

生成文件	内容描述
model.c 或 .cpp	包含实现模型算法代码的入口函数（例如，*model*_step, *model*_initialize,以及 *model*_terminate）
*model*_private.h	包含模型及子系统必须的局部宏（local macros）和局部数据（local data）。该文件包含在模型生成的源文件中。将手写代码连接到模型时，不需要包含 *model*_private.h。
model.h	声明模型数据结构和模型入口函数及数据结构的公共接口。同时，提供用于实时模型数据结构接口的访问宏（accessor macros）。model.h 包含在模型子系统.c 或.cpp 中。 如果需要将手写代码连接到一个或多个模型生成的代码，需要在手写代码中包含要连接到的每个模型的 *model*.h 文件。
*model*_data.c 或.cpp (有条件的)	*model*_data.c 或.cpp 是有条件生成的。它包含参数数据结构、常数模块 I/O 数据结构、以及任何模型结构数据类型的零值声明。如果模型中没有使用这些数据结构和零值，不会生成 *model*_data.c 或.cpp。注意：这些结构和零值 在 *model*.h 中不是声明为 extern 的。
*model*_types.h	为实时模型数据结构和参数数据结构提供。可重用函数的函数声明可能需要这些。同时，还提供模型使用的用户定义类型的类定义。
rtwtypes.h	定义 RTW Embedded Coder 生成代码需要的数据类型、结构和宏。大多数其他生成的代码模块需要这些定义。
MW_c6xxx_csl.c	初始化目标板的程序代码，中断及定时器设置等代码。
MW_c6xxx_csl.h	目标板初始化程序的头文件及函数原型声明。
ert_main.c 或.cpp （可选）	只有当开启 **Generate an example main program** 选项时才会生成此文件。（该选项默认状态为开启）。

注：*model* 表示模型的名称，被具体模型名称替代。

2.2.3 集成外部代码到生成的 C 代码中

外部代码，也称用户代码，区别于借助于 RTW 技术从模型自动生成的代码。代码集成就是将外部代码和自动生成的代码集成到一个系统中，完善系统功能。代码集成可以归于以下两类情况：

（1） 将 RTW 生成的代码集成到更大系统的代码库中。比如，为一个现有系统生成一个新的算法代码，正确使用生成代码的入口函数并包含相关的头文件，可以方便地实现此功能；

（2） 将现有代码集成到RTW软件生成的代码中。用模型开发了系统的大部分功能，嵌入式系统上设备的驱动程序或者已有的经过测试的算法代码都可以集成到RTW从模型生成的代码中，实现系统的完整功能。

冶金电炉智能控制系统固件开发属于第二种情况。这些手写代码既可以是程序文件，也可以是简单的几行语句。RTW有多种方法将现有代码和用户开发的代码整合到生成的代码中，主要有基于模块的方式和基于模型的方式。在具体使用时，根据应用程序的需要选用恰当的实现机制。

2.2.3.1 基于模块的代码集成方式

根据应用程序不同的要求，可以采用不同的机制实现基于模块的代码集成：

（1） 使用用户编写的 S-Function 模块

（2） S-Function Builder 模块

（3） Real-Time Workshop Custom Code 模块

具体选用哪种方法，视需要而定。比如，需要应用像表达式折叠这样的代码生成优化选项，需要使用用户编写的 S-Function 模块；而如果要使集成的代码不影响仿真，则使用 Real-Time Workshop Custom Code 模块。总的来说，S-Function 模块提供了代码集成功能最全面、最灵活的手段，并且可以指定额外的编译信息。

2.2.3.2 基于模型的代码集成方式

基于模型的或基于目标的代码集成方式可以采用以下两种实现机制：

（1） Configuration Parameters 对话框中的 **Custom Code** 子窗口；

（2） 自定义的 target template makefile

2.2.4 数据类型和变量

2.2.4.1 Simulink 支持的数据类型

大多数编程语言要求在使用数据和函数之前进行声明（declare）。声明指定：数据的范围、生命周期、数据类型以及初始化[15]。Scope 和 Duration 合起来为存储类型，如果没有提供初始值，多数编译器赋 0 值或 null 指针。

表 3 变量声明的作用

范围（Scope）	有权使用数据的程序区域
生命周期（Duration）	数据驻留在存储器中的时间
数据类型（Data Type）	分配给数据的存储器数量
初始化（Initialization）	数值，指针，或 NULL

Simulink 支持的数据类型有 double,single,int8,uint8,int16,uint16,int32,uint32 以及 Fixed-Point（定点数，8 位、16 位或 32 位字长）。Simulink 支持除 int64 和 uint64 之外的所有内置 MATLAB 数据类型。“内置的数据类型”指由 MATLAB 自己定义的而非 MATLAB 用户定义的数据类型。除非另作说明，Simulink 文件中的术语“数据类型”指内置的数据类型。除了这些内置类型，Simulink 定义了布尔（1 或 0）类，布尔类的实例在系统内部是用 uint8 数值来表示的。

Simulink 支持多种数据存储类型，除了常用的数据存储类型以外，还有很多便于和外部代码交互的特性。借助这些数据存储类型，可以使用用户程序访问自动生成代码中的数据变量，也可以让自动生成源代码中的变量使用用户代码中的数据。Simulink 支持的数据存储类型及其说明如下表所示：

表 4 支持的数据存储类型

名称	描述	模型参数	模型信号	数据类型
Const	声明时使用 const	支持	不支持	所有
ConstVolatile	声明时使用 const volatile	支持	不支持	所有
Volatile	声明时使用 volatile	支持	支持	所有
ExportToFile	生成并包含文件，使用用户指定的名称，包含全局变量的声明和定义	支持	支持	所有
ImportFromFile	引用包含全局变量的头文件	支持	支持	所有
Exported Global	声明并定义全局范围的变量	支持	支持	所有
Imported Extern	导入模型外部定义的变量	支持	支持	所有
BitField	在命名的位域中嵌入布尔数据	支持	支持	布尔型
Define	用#define 宏定义参数	支持	不支持	所有
Struct	向命名的结构中插入数据实现数据封装	支持	支持	所有

2.2.4.2 控制 Simulink 模型的数据

有两种方法可用于 Simulink 中声明数据：数据对象（data objects）和直接说明（direct specification）。两种方法都允许对数据类型（data type）和存储类型（storage class）进行完全控制。在一个模型中可以同时使用这两种方法。

三类数据对象（data object）：

- Signal
- Parameter
- Bus

代码生成器使用 MATLAB 基础工作间中的数据对象。可以创建并检查这些数据对象，通过在 MATLAB 命令窗口中输入命令或使用 Model Explorer。数据对象包含 active 域和 descriptive 域。Active 域影响仿真或代码生成。Descriptive 域不影响仿真和代码生成，但用于数据词典和模型检查工具。

2.2.4.3 增加新的数据对象

RTW 产品定义了可调参数的四种存储类型。在 RTW 的编译过程中，信号、可调参数、模块状态或数据对象的存储类型指的是该实体在生成的代码中的声明、存储以及表示方式。在 RTW 编译过程背景下的术语“存储类型”和诸如用在 C 语言中的“存储类型说明符”并非同样的意义。

RTW 软件定义了用于所有目标（target）的四种内置存储类型，必须将可调参数声明为以下四种存储类型之一：

- SimulinkGlobal（Auto）：这是默认的存储类型。RTW 产品存储该参数作为 *model*_P 的一个成员。在代码生成时，*model*_P 的每个成员均被初始化为对应的 workspace 变量数值。
- ExportedGlobal：生成的代码 instantiates 并初始化该参数，*model*.h 将它作为全局变量 exports。exported 全局变量和 *model*_P 数据结构无关。在代码生成时，每个 exported 全局变量均被初始化为对应的 workspace 变量数值。

- ImportedExtern：*model*_private.h 将该参数声明为 extern 变量。你的代码必须提供适当的变量定义和初始化程序。
- ImprotedExternPointer：*model*_private.h 将该变量声明为 extern 指针（pointer）。如果有的话，你的代码必须提供适当的指针变量定义和初始化程序。

这些数据类型对生成的用于引用数据的代码形式提供有限的控制。例如，存储类型为 Auto 的数据，典型地，作为结构的一个元素被声明或访问；而具有 ExportedGlobal 存储类型的数据却作为非结构体的全局变量被声明或访问。

Simulink 的内置存储类型适合许多应用场合，但嵌入式系统的设计者通常需要更深入地控制数据的表示形式。RTW Embedded Coder 的自定义存储类型（CSCs）扩展了 RTW 提供的内置存储类型。CSCs 提供了在特定程序中控制嵌入式算法要求的数据结构的技术。例如，可以用 CSCs：

- 定义用于存储参数或信号数据的结构体
- 存储布尔数据在位字段中以保留内存
- 将生产的代码与无法修改接口的遗留软件结合
- 生成符合公司高安全性代码软件工程准则的数据结构和数据定义

CSCs 只能用于 ERT 目标中。使用 GRT 目标时，CSCs 的名称有时会出现在对话框中，但除非是在 ERT 目标中，指定 CSC 没有影响。在没有 ERT 的目标中指定 CSC 等价于指定 Auto 类型。

2.3 基于模型设计方法中的验证技术

从基于模型设计方法开发周期的 V 模型中可以看出，这种方法在开发过程中尤其重视验证。实际上，验证技术的有效应用是最终设计能够实现预期设计功能的必要保障手段。本章的研究内容是采用 MATALB/Simulink 实现基于模型设计方案的过程中可以采用的验证手段和工具。总体来说，在综合开发平台上针对 DSP 目标固件开发实施的验证技术涉及两个环境、三个方面：主机环境、目标环境以及主机仿真目标环境。主机环境下的仿真主要是验证模型是否满足产品功能需求，一般按照从局部到整体的测试及验证方式。主机仿真目标环境主要是为了进一步改进模型、进

行系统级模型的功能及性能评估，评估系统对硬件平台的资源需求。以上的仿真验证大多基于非实时的操作系统环境。在目标上的验证是实时性的，RTW 为组件模型生成实时嵌入式代码，同时需要集成到可执行的软件框架之中，部署到目标开发板上。不同的验证技术都有特定的作用和实现目标，在产品开发整个生命周期内特定阶段需要特定验证技术实现阶段性目标。典型的开发过程及相关验证技术如下图所示[20] [25] [26]：

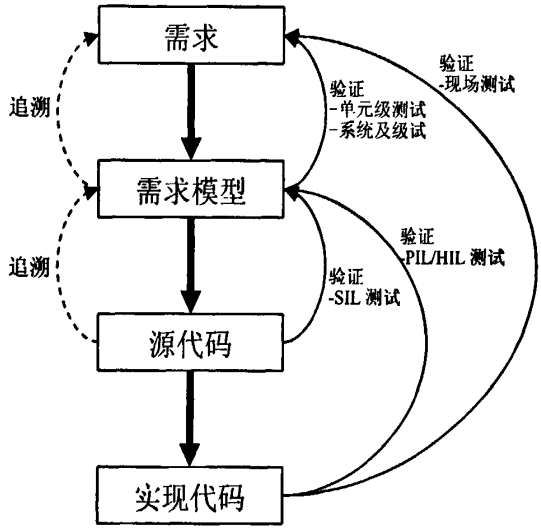


图 6 典型的开发过程与验证技术

2.3.1 早期验证解决系统设计面临的挑战

系统设计过程中，错误的产生更多地出现在更多的集中在设计的早期。但是在传统的系统开发方案中，这些错误要在设计晚期做系统集成的时候才能发现——显然，这种情况使得传统开发方案效率较低。

基于模型的设计方法通过仿真技术对功能模型进行分析，验证概念模型是否实现了对性能指标的要求，并进一步改进完善。设计过程中的错误在开发初期即可发现并得到校正，不会使错误向后传播。

传统设计过程中对功能模块的验证需要手工编写测试代码，这些测试代码往往是专用的，针对特定功能模块的，编写起来繁琐并且可重用性较差，在开发过程中耗费时间和精力。基于模型的设计方法在统一的环境中借助系统模型对组建模型进行验证，利用自动代码生成技术将组件模型生成的代码集成到嵌入式实时程序的架构中，避免了手工编写测试环境代码的繁杂工作，并且具有很强的可移植性。

2.3.2 测试并改进概念模型

概念模型（Concept Models）是在开发过程中最初实现系统功能的模型，在此基础上逐步改进优化，最终确定最优模型形成最终产品。

2.3.2.1 基于主机的仿真

可在设计初期应用 RTW 技术，以基于主机（Host-Based）的仿真验证产品需求是否得到满足。在整个产品研发的生命周期内，可以应用 Simulink 创建、仿真、分析并改进算法和控制器模型及其控制的运行环境（被控对象的模型）。有应用程序需求的说明文档以及开发的概念模型，就可以用基于主机的仿真实现以下目的：

- 在非实时的主机环境下验证模型的功能；
- 改进算法的概念模型；
- 测试概念模型；

主机（host computer）在非实时操作系统环境下运行 MATLAB 和 Simulink。模型的运行不会按照模型设定的计时要求实时地运行，Simulink 引擎会根据主机系统可用资源全速运行模型。比如 ，嵌入式系统模型中的两个任务之间有短暂的时间间隔，在嵌入式系统上实时运行时，该间隔时间被保留、没有任务运行，而在主机上仿真时，该时间间隔不会被保留、执行完一个任务就会立即执行下一个任务。

正常（Normal）模式的仿真是以解释模式（interpretive mode）运行的，允许开发者访问、显示并调节数据和参数。该模式在实验阶段和模型开发初期非常有用。但随着模型及仿真脚本在大小和复杂度上的增长，仿真速度会变慢。可以使用供替代的方式——Accelerator 模式和 Rapid Accelerator 模式——改善仿真性能。这两种模式用编译的目标代码替代解释性的代码，从而提高 RTW 技术执行效率。但这两种加速仿真模式生成的代码仅用于加速模型仿真的目的，必须使用 RTW 产品生成其他用途的代码。

2.3.2.2 独立运行的快速仿真

建立好模型后，可采用独立运行的快速仿真（Standalone Rapid Simulations）这种方式检验模型的性能。在模型配置参数对话框的 RTW 子项中，将目标选定为

rsim.tlc（Rapid Simulation Target），配置其他相关选项，编译模型，生成代码。快速仿真生成的独立运行程序可以在 MATLAB 环境外以 External 模式运行，与 Simulink 模型交互数据和参数。

使用 RSim 目标可以在仅编译一次情况下就运行多项仿真，研究各种参数设置和输入信号对模型的影响，适合批量仿真（batch simulations）。可以在 Windows 平台上创建一个.bat 文件设置运行多次仿真。新的输出文件名可以重新运行仿真而不会覆盖先前的仿真结果。

2.3.2.3 用快速原型技术测试并改进模型

快速原型（Rapid Prototyping）是一种用于测试新想法和研究的应用程序开发技术。使用快速原型可以生成模型的代码，运行代码并调整。

在廉价的 PC 机或非目标硬件平台上（包括现成的 I/O 卡）对改进的概念模型使用快速原型技术。控制器的代码可以部署到和系统硬件（被控制的物理设备）相连的实时仿真器上。

快速原型技术不甚强调代码效率和 I/O 等待时间（latency），其首要目标在于：

- 组合系统设计中的算法、软件、以及硬件设计阶段，消除潜在的障碍；
- 迅速地通过快速原型反复修正算法，从而改进模型的设计；
- 验证控制器在非实时情况下能否满足控制物理系统的需要；
- 在设计硬件前评估系统系能，编写系统软件代码，或者敲定固定的设计；
- 在构建昂贵硬件前通过迅速地反复研究问题解决方案以缩减开发费用；

下图展示了传统方法和快速原型方法之间的区别：

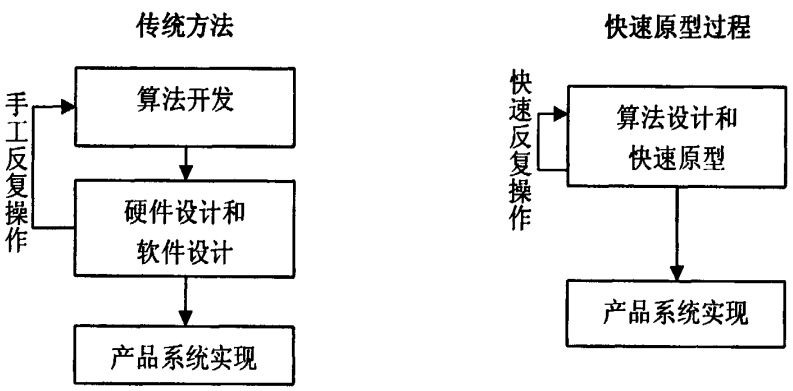


图 7 应用程序的传统设计方法与快速原型过程的对比

(1) 快速原型系统仿真

所谓系统仿真，就是将一个或多个组件的模型集成到表示这些组件最终运行环境的系统模型或环境模型中去。快速原型实现系统仿真是非实时的。系统仿真的目的在于：

- 改进组件模型
- 改进系统模型
- 在非实时环境中验证组件模型的功能
- 测试概念模型

(2) 快速原型实时系统

实时快速原型要求使用实时仿真器，连接到被控系统硬件上。生成代码，部署到实时仿真器或嵌入式微处理器上，代码运行时进行调整。这一设计步骤是非常重要的，验证一个组件是否能够满足系统的控制要求，允许开发人员评估代码、查看并优化代码。实时快速原型的目的在于：

- 通过迅速地反复操作改进组件模型和环境模型的设计；
- 验证组件在实时环境中能否满足物理系统的控制要求；
- 在设计硬件前评估系统系能，编写系统软件代码，或者敲定固定的设计；
- 测试硬件；

2.3.3 生成代码和系统要求的追溯

应用系统的要求记录在文档中，如电子表格、数据库或需求管理工具，采用像 Simulink Report Generator，Microsoft Word，Microsoft Excel 等工具，可以实现交互式追踪和追踪报告验证生成的代码是否满足记录在文档中的要求。这些机制提供了一种方式，能够追溯生成的代码到文档中的需求，并且生成追溯报告。

生成代码和需求追溯可实现的目标有：

- 将需求文档和概念模型中的对象连接到一起，生成与该模型相关联的需求报告。这样，系统需求在模型中得到充分的体现，并且可以方便地检查各个需求是否得到实现。
- 从模型中的模块或子系统追溯生成代码，或者从生成的代码追溯与之相连

的模型中的模块或子系统。这种机制可以在代码生成报告中查看与某段源代码对应的模块或子系统，同样，也可以从模型中查看某个模块或子系统对应的模块。从而确认了由模型到代码的完全转换，进而可以确定生成了需求文档中记录的功能需求的代码。

2.3.4 验证生成的源代码

通过基于主机的仿真和快速仿真验证了开发的模型的功能，借助 RTW 技术自动生成代码后，代码是否忠实体现系统模型的功能，就需要对生成的源代码进行验证。

2.3.4.1 代码正确性检查

检查代码的正确性就是核实源代码中没有编译时错误，没有链接时错误，并且没有运行时错误（比如溢出，除零，或者数组访问越界）。

编译时错误和链接时错误的检测可以依赖于集成开发环境工具，并在这些工具的帮助下校正错误。但是运行时错误的检查和校正困难得多。某些方法可以检测运行时错误：

- 运行可执行程序，分析运行结果；
- 插入输出信息之类的代码
- 编写测试代码并运行

检查由 RTW 技术生成的代码的正确性时，还可以选用 PolySpace®产品。产品 PolySpace 是建立在验证技术基础上的，应用形式方法检测、算术地证明各类运行时错误是否存在，比如溢出。此外，PolySpace Model Link™ SL 产品允许开发人员追踪 PolySpace Client 报告的 C/C++结果到 Simulink 模型中。

2.3.4.2 软件在环仿真验证生成的源代码

软件在环（Software-in-the-Loop）仿真是在 MATLAB 主机系统上模拟模型在目标上的行为。借助 RTW 产品可以用 S-函数包装运行软件在环仿真。此外，如果主机和目标平台的字长不同（如，32 位的主机和 16 位的目标机），可配置模型，使用硬件仿真（hardware emulation）在主机平台上仿真目标机的性能。

硬件仿真能保证整数和定点数运算在 MATLAB 主机系统运行的仿真结果和在目标系统上运行的生成代码产生的结果的一致性。生成的代码仿真在目标机上的性能。产生的仿真代码可能会包含附加代码以确保性能与目标环境一致，比如，类型转换代码。

要配置模型应用硬件仿真进行 SIL 仿真，在模型的配置参数选项卡中进行相应的设置：**Configuration Parameters>Hardware Implementation**。在 MATLAB 主机系统上运行 SIL 仿真，要生成部署到目标上的代码，需将 **Hardware Implementation>Emulation hardware** 中的选项 **None** 勾选，重新生成代码。

2.3.5 目标环境上的组件验证

有了算法模型的代码，还需要将其集成到软件框架中经过编译、链接生成可执行应用程序才能部署到嵌入式系统运行。应用 Embedded IDE Link 中的模块以及 DSP 的 Target Support Package 模块库中提供的模块，根据需要集成用户手写的设备驱动程序，然后可以快捷地生成可以部署到目标 DSP 板的完整的应用程序。

嵌入式验证主要是在处理器上运行生成的代码，证明代码确实实现了期望的功能。Embedded IDE Link 的组件组合起来提供工具在开发过程中验证代码，允许仿真的一部分在硬件上运行，优化代码执行。

2.3.5.1 PIL 联合仿真技术

处理器在环（Processor-in-the-Loop，PIL）联合仿真是一种帮助开发者评估算法在选定的硬件平台上运行状态的技术。PIL 需要 RTW Embedded Coder 的支持。完整的软件系统应该包括：

- 软件算法
- 调度算法
- 设备驱动程序

组件模型生成的代码即软件的算法，此处称为 PIL 算法。调度算法和设备驱动程序合起来视为 PIL 应用程序，借助 Embedded IDE Link 和 Target Support Package 产品的支持，可由 RTW Embedded Coder 生成适合目标处理器平台的应用程序架构。

为了实现组件验证的目的，除了将组件模型转换为嵌入式代码并集成到软件架构中部署到目标平台，还需要由测试环境提供组件算法的驱动信号并对其产生的响应进行分析，也就是需要为 PIL 应用程序提供测试环境。测试环境与目标之间的联结需要使用 PIL 模块。相关概念的定义如下：

(1) PIL 算法

PIL 联合仿真过程中测试的算法代码。PIL 算法以编译过的对象形式参与联合仿真，确保验证是在对象层级上的。

(2) PIL 应用程序

运行在处理器硬件平台的可执行应用程序。Embedded IDE Link 扩充算法代码，添加 PIL 执行程序框架，创建 PIL 应用程序。执行程序框架代码继而被编译成嵌入式应用程序的一部分。

执行程序框架代码包含 string.h 头文件，PIL 应用程序可以使用 memcpy 函数。PIL 应用程序使用 memcpy 在 Simulink 模型和联合仿真的处理器之间交换数据。

(3) PIL 模块

开发者从模型的子系统创建的模块。运行仿真时，PIL 模块起到模型和运行在处理器上的 PIL 应用程序之间接口的作用。

2.3.5.2 PIL 的实现

联合仿真需要分工：Simulink 软件建模被控对象或测试平台，由模型中算法生成的代码运行在处理器硬件平台上。

在 RTW Embedded Coder 软件生成代码的过程中，可以将 Simulink 软件中组件之一——包括模型、模型中的子系统、或库中的子系统——创建 PIL 模块。然后将生成的 PIL 模块放在用作测试平台的 Simulink 模型中，运行测试，评估针对特定处理器的代码执行状态。

下图是联合仿真的结构示意图：

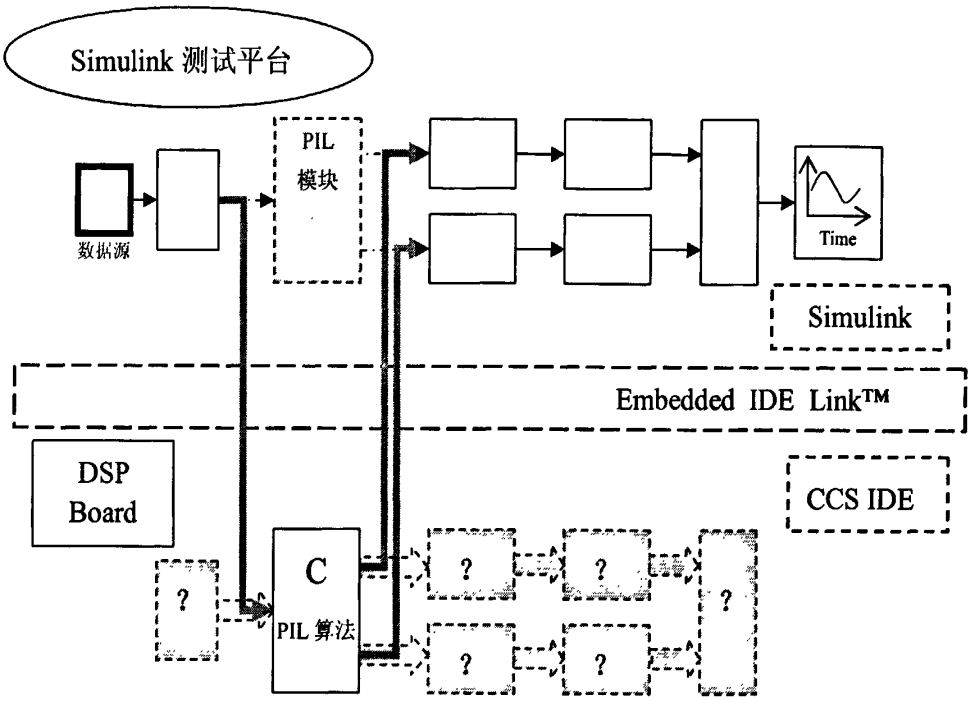


图 8 借助系统模型应用 PIL 仿真进行早期验证

联合仿真过程中，主机运行的 Simulink 仿真模型通过 PIL 模块与目标环境中运行 PIL 应用程序联系。在测试环境中，PIL 模块代表 PIL 算法但仅起到与 PIL 应用程序中的 PIL 算法联系的作用。PIL 模块将输入数据通过 Embedded IDE Link 发送到 CCS 集成开发环境，进而通过 JTAG 仿真器送给目标板上运行的 PIL 应用程序。PIL 算法使用接收到的数据进行实时运算，运算结果通过反向途径传回 Simulink 模型中，由 PIL 模块输出，驱动后续功能模块。甚至可以在 Simulink 测试环境中同时运行组件模型，将 PIL 算法运算结果和 PIL 组件模型运算结果进行对比。通过分析联合仿真的结果，评估组件代码在目标环境的性能，并可进一步分析代码的可移植性。

2.4 实时嵌入式系统功能的实现

2.4.1 同步任务管理

任务是嵌入式系统架构中的重要概念[30]。固定步长的 Simulink 模型有两种执行模式：单任务和多任务。这两种模式仅对固定步长的求解程序有效。

同步任务，不论是只有一个任务还是多个任务，都是由定时器中断协调运行的。

定时器中断决定某个任务在某个特定的时刻运行。

2.4.1.1 任务分配机制

在 Simulink 模型中，嵌入式系统任务的分配是通过模块的“Sample Time”参数实现的。模型中模块的“采样时间”根据任务需要来设定，根据这个参数的设定可把模型分为单速率模型和多速率模型。相关参数可在 Solver 选项卡中设置，其中的固定步长，也就是基本采样时间必须是所有采样时间的公约数。采样时间越小的模块组成的任务运行的速度越快，优先级也越高。在模型的参数配置对话框 Solver 面板中 **Tasking mode for periodic sample times** 菜单选择周期性任务的运行模式。再根据模型中模块的 Sample Time 参数设置，可出现下表列出的组合方式，反映到生成的代码上，即单任务或多任务的应用程序。

表 5 RTW Embedded Coder 目标模型允许的解算器模式

模式	单速率	多速率
单任务	允许	允许
多任务	禁用	允许
自动	允许（默认为单任务）	允许（默认为多任务）

2.4.1.2 处理不同速度模块的联结

具有不同 Sampling Time 的模块在连接时，就存在速度变化（Rate Transitions）。模型中会出现以下两种周期性采样速度变化的情况

- 速度快的模块驱动速度慢的模块
- 速度慢的模块驱动速度快的模块

速度变化会使模块运行顺序出错，从而引起计算数据的错误。不同速度的模块连接到一起时，需在它们之间加入 Rate Transition 模块。该模块可以解决以不同速度运行的模块之间传输的数据的完整性可确定性问题。下面就这两方面的问题进行解释：

(1) 数据完整性

当模块的输入在模块执行时发生变化就会产生数据完整性的问题。数据完整性问题会有抢占引起：

- 快速模块向慢速模块提供输入
- 慢速模块从快速模块读取输入值V_1并用该值进行计算
- 计算过程被快速模块的运行抢占，快速模型计算的结果输出值为V_2
- 数据完整性问题就出现了：慢速模块恢复运行的时候，继续中断的运算，现在使用新的输入值V_2

这种数据传输方式称为“不受保护的”。在“受保护的”数据传输中，快速模块的输出V_1被保持知道慢速模块完成运行。

(2) 可确定性与不可确定性

在确定性的数据传输中，数据传输的时间时完全确定的，由模块的采样时间来确定。不可确定性的数据传输，其时间依赖于数据的可用性、模块的采样时间、以及接收数据的模块相对于提供数据的模块的执行时间。

2.4.2 异步事件处理

同步任务按设定好的方式在系统时钟的控制下周期性地运行，但这种运行方式对某些系统来说不够灵活，如控制系统或通信系统，它们要求对异步事件做出实时响应。这类事件的发生是随机的，这就要求系统具有异步事件的实时处理能力。

2.4.2.1 异步事件任务的创建

RTW 能够产生用于实时处理异步事件的代码，而 Embedded IDE Link 也提供了硬件中断和管理的模块[18][19]：

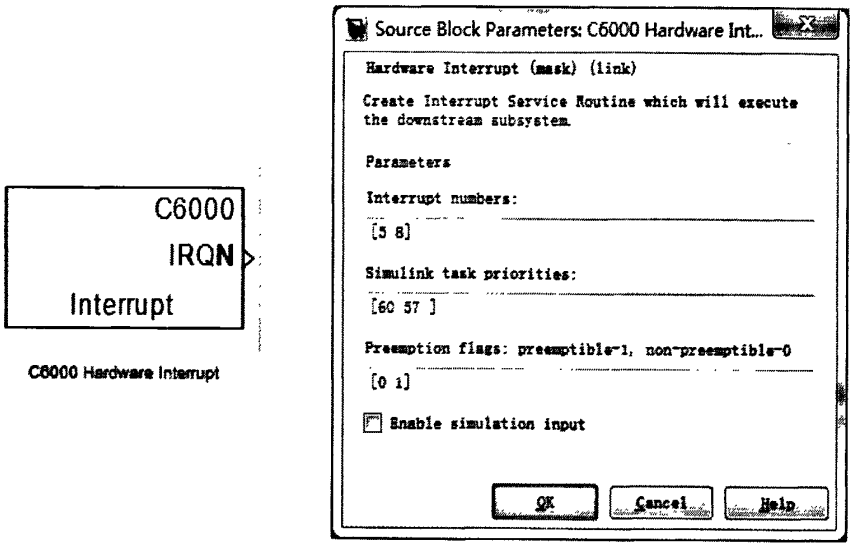


图 9 Embedded IED Link 库中提供的 C6000 硬件中断模块

左图：硬件中断模块　　右图：中断模块参数配置对话框

而如果采样 DSP/BIOS 功能，Target Support Package 模块库中还提供了相应的功能模块：

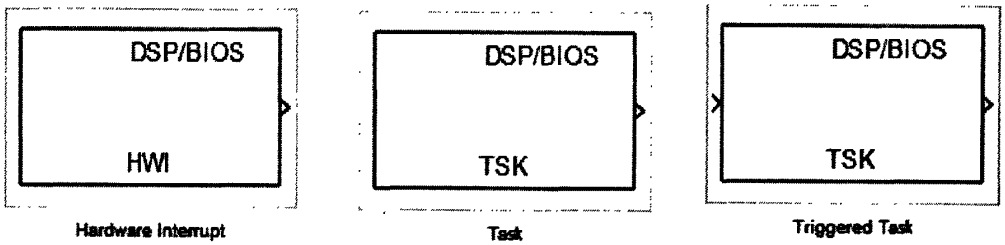


图 10 Target Support Package 模块库中提供的对 DSP/BIOS 功能支持模块

异步事件定义为子系统，由硬件中断模块驱动。中断发生时，对应于该中断号的子系统被激活运行，相应想异步事件得到处理。每个模型只有一个硬件中断（Hardware Interrupt）模块，多个中断源以向量的形式在对话框的 Interrupt nubers 选项中设置，模块的输出也为向量，顺序与该处设定的中断号码对应。Simulink task priorities 选项设置与之对应的任务优先级，而 Preemption flags 选项设置相应的中断事件在运行过程中是否允许被其他事件中断。

C6713CPU 内核支持 16 个不同优先级的中断信号，其中前四个中断为非屏蔽中断，对应的中断源是固定的。其他 12 个中断源是可屏蔽的，每个中断源都有 32 个可能来源（中断事件）。

2.4.2.2 中断映射及使能

调度模块会为会生成响应中断的中断服务程序，但在生成的代码中并不使能中断。此外，该模块并不将中断映射到模块对话框中指定的中断服务程序。因此，要使能并映射中断程序，需提供执行该功能的代码。如下例：

IRQ_map(IRQ_EVT_EXTINT5,5); 将中断号 5 映射到中断源（中断事件）ext.int.5

IRQ_set(IRQ_EVT_EXTINT5); 使能中断号

上面使用的宏代码是 TMS320C6000 CSL API 函数，具体应用参照《TMS320C6000 API Reference Guide》[28]。根据前面讨论的添加用户代码的方法，可以添加一个 System Outputs 模块（Real-Time Workshop\Custom Code 模块库）。

2.4.3 任务调度

调度是嵌入式系统架构中的核心概念。RTW 生成的代码支持同步调度和异步调度。如前所述，同步任务的调度是借助任务标识 tid（task identification）机制在系统时钟的协调下进行的。使用 Embedded IDE Link 模块库中的调度模块可以实现异步调度，定义中断以及中断任务。用这些调度模块可以实现多个任务的异步运行模式。

2.4.3.1 同步调度

通过定时器中断的代码同步运行需要定时器的中断服务程序（ISR）来协调。ISR 响应中断申请后重复运行一次模型。对 Embedded IDE Link 生成的代码使用一个 CPU Timer，TMS320C6713 使用 Timer 1，中断号为 5。

软件自动配置定时器，生成代码的基准速度采样时间等于中断速度。Embedded IDE Link 计算并配置定时器周期，确保实现期望的采样速度。

可实现的最小基准采样时间依赖于两个因素：算法的复杂度和 CPU 时钟速度。可实现的最大值依赖于最大的定时器周期值和 CPU 时钟速度。

如果模型中的所有模块继承采样时间数值并且没有明确地定义采样时间，Simulink 指定 0.2 秒的默认采样时间。

（1） 实时单任务调度

下图阐明了单任务的执行

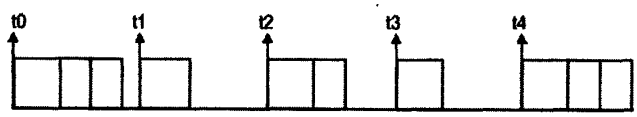


图 11 单任务运行模式

下面的伪代码是实时单任务系统模型的执行：

```
rtOneStep()
{
  Check for interrupt overflow
  Enable "rtOneStep" interrupt
  ModelOutputs    -- Major time step.
  LogTXY          -- Log time, states and root outports.
  ModelUpdate     -- Major time step.
  Integrate       -- Integration in minor time step for models
                  -- with continuous states.
    ModelDerivatives
    Do 0 or more
      ModelOutputs
      ModelDerivatives
    EndDo (Number of iterations depends upon the solver.)
    Integrate derivatives to update continuous states.
  EndIntegrate
}
main()
{
  Initialization (including installation of rtOneStep as an
  interrupt service routine, ISR, for a real-time clock).
  While(time < final time)
    Background task.
  EndWhile
  Mask interrupts (Disable rtOneStep from executing.)
  Complete any background tasks.
  Shutdown
}
```

首先是初始化阶段，包括初始化模型状态及设置执行引擎（execution engine）。然后，模型开始执行，每次执行一步。rtOneStep 函数是由周期性时钟驱动的。在程序基准采样时间规定的时间间隔上，ISR 抢占背景任务执行权，执行模型代码，必须在下一个时间间隔（即下一次定时器中断）之前完成。程序控制再交还给背景任

务。

（2） 多任务调度

下图阐明了 RTW 软件在多速度环境、伪多任务环境以及单任务环境下处理多速度系统中任务的计时的方式。

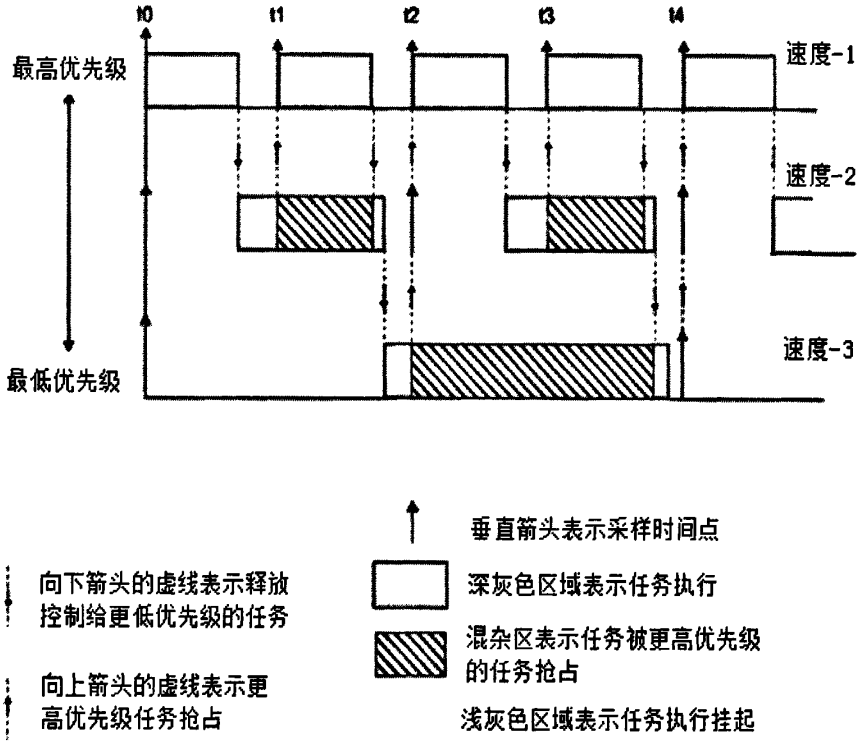


图 12 多任务运行模式

下面的伪代码是实时多任务系统模型的执行：

```
rtOneStep()
{
  Check for interrupt overflow
  Enable "rtOneStep" interrupt
  ModelOutputs(tid=0)        -- Major time step.
  LogTXY                     -- Log time, states and root outports.
  ModelUpdate(tid=0)         -- Major time step.
  Integrate                  -- Integration in minor time step for
                             -- models with continuous states.
      ModelDerivatives
      Do 0 or more
        ModelOutputs(tid=0)
        ModelDerivatives
      EndDo (Number of iterations depends upon the solver.)
      Integrate derivatives and update continuous states.
  EndIntegrate
    For i=1:NumTasks
      If (hit in task i)
        ModelOutputs(tid=i)
        ModelUpdate(tid=i)
    EndIf
  EndFor
}
main()
{
  Initialization (including installation of rtOneStep as an
  interrupt service routine, ISR, for a real-time clock).
  While(time < final time)
    Background task.
  EndWhile
  Mask interrupts (Disable rtOneStep from executing.)
  Complete any background tasks.
  Shutdown
}
```

模型中的模块具有不同的采样时间时，Simulink 引擎为每个模块指定一个任务标识符 tid（task identifier），任务标识符将模块与以该模块的采样速度执行的任务关联到一起。

周期性任务运行在多任务模式时，系统默认具有最快采样速度的模块由最高优先级的任务执行，采样速度次之的模块由优先级次之的任务执行，以此类推。基准采样时间在模型中是最快的。处理高优先级任务之间的空闲时间用于处理低优先级

的任务。这样，程序的执行效率更高。在多任务环境中（即，在实时操作系统下），能定义各自独立的任务并指定它们的优先级。在 bare-board target（即，没有实时操作系统）中，不能创建各自独立的任务。

这就意味着一个中断可以在另一个中断正在进行处理的过程中发生。在这种情况下，当前中断被挂起，浮点运算单元（FPU）环境变量被保存，更高优先级的中断执行其高优先级（也就是更快的采样时间）的代码。一旦完成，控制权交给被挂起的中断服务程序。

2.4.3.2 异步调度

Embedded IDE Link 模块库和 Target Support Package 为 C6713 处理器提供了使用中断资源的模块，应用 RTW 技术，可以为系统使用的中断产生中断服务程序。这些异步调度模块有：

- Bare-Board 代码生成模式的 C5000/C6000 Hardware Interrupt 和 Idle Task
- DSP/BIOS 代码生成模式的 DSP/BIOS Hardware Interrupt，DSP/BIOS Task，和 DSP/BIOS Triggered Task

C6000 Hardware Interrupt 模块能够使能选用的 TMS320C6000 DSP 硬件中断，生成相应的中断服务程序，并将中断服务程序的入口地址和中断服务向量表连接到一起。 Hardware Interrupt 和 Idle Task 模块驱动 Function Call Subsystem，也就是中断服务程序。中断事件发生后，生成的子系统的代码就被中断服务程序调用。

C6000 Idle Task 模块在生成的代码中指定一个或多个函数作为后台任务运行。这些后台任务充分利用实时系统任务执行的“缝隙时间”，完成优先级较低、不甚紧急的任务。这些函数是从 Idle Task 连接的函数调用子系统生成的。

DSP/BIOS 代码生成模式的应用需要在目标参数选项中选定使用 DSP/BIOS 功能，然后采用相应的 DSP/BIOS 异步调度模块设计模型。应用 DSP/BIOS 功能，可以在应用程序中使用DSP/BIOS实时操作系统(RTOS)的特性。作为TI eXpressDSP™技术的一个组成部分，TI 设计的 DSP/BIOS 包含三个部分：

- DSP/BIOS 实时分析工具
- DSP/BIOS 配置工具

➢ DSP/BIOS 应用程序接口

模型中应用了 DSP/BIOS 功能，就可以将这些组件连接到应用程序中，根据需要使用必要的工具。

使用异步调度的模型中，各个任务运行的优先级不在严格受到采样时间的限制，中断服务具有较高的优先级，并且，在中断模块的选项卡中可以设置各中断的优先级。

第三章 冶金电炉智能控制系统开发研究

冶金电炉智能控制系统由上位机和下位机（冶金电炉智能控制板）组成，本章就系统的功能模块划分探讨基于模型设计的方法在该系统中的应用。然后以温度预报的神经网络模型为例，论述完整的开发过程以及在应用的技术。

3.1 冶金电炉智能控制系统实施方案

冶金电炉智能控制控制系统，从物理结构上来讲，主要有控制对象（冶金电炉系统）、控制器（冶金电炉智能控制板）和上位机（二级系统）三部分构成。该套智能控制系统从功能结构上来讲，上位机运行温度预报模块和功率优化模块，而下位机，即冶金电炉智能控制板，由电炉仿真器、智能调节器、高速数据采集和处理以及和上位机的数据传输等主要功能模块组成。下面显示的是智能控制系统结构示意图。

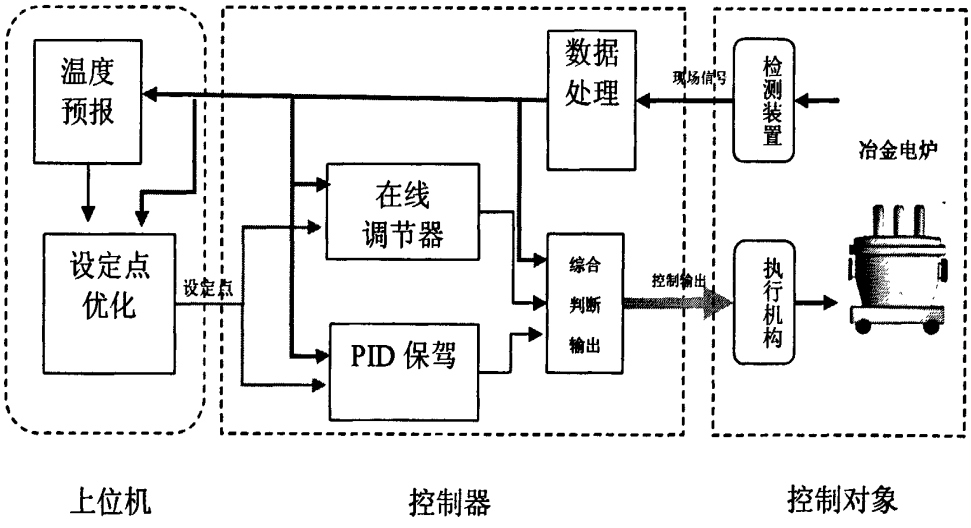


图 13 冶金电炉智能控制系统功能模块图

3.1.1 建立控制对象的数学模型

采用基于模型的设计方法，被控对象的数学模型是必要的。被控对象的数学模型既是控制器设计的基础，也是验证控制器功能及性能的所凭借的重要工具。

冶金电炉结构复杂，冶炼时内部物理化学变化及反应规律难以描述，机理分析

方法建模难度较大。可以充分利用冶金电炉的运行数据，建立等效物理模型。模型的准确性以及精度对后期工作有重要的影响，只有准确的在精度许可范围内的被控对象模型才能作为设计控制器模型的依据并用以检验控制器。

可以用现场输入数据输入模型，并把模型的输出与现场输出进行对比，调节模型参数，以验证模型的精准性。

3.1.2 上位机功能模块的开发

在 MATLAB/Simulink 环境下统一建模实现上位机的算法功能——智能温度预报和功率优化。这样设计的好处一是可以做系统级的功能验证，各功能模块之间做到无缝连接；二是可以将上位机的算法模型借助 RTW 技术生成 C 代码或编译成静态/动态链接库，便于 HMI 程序调用。

3.1.3 冶金电炉智能控制板固件开发

电炉仿真器和智能调节器是 863 项目“冶金电炉智能控制系统”理论研究成果之一，是也觉得了智能控制器的核心算法，该部分功能实现在冶金电炉智能控制板上。

冶金电炉智能控制板固件从总体功能结构上将控制器实时嵌入式应用程序分为三个部分：控制器的算法代码，设备驱动程序，以及调度程序和后台管理程序。如下图所示：

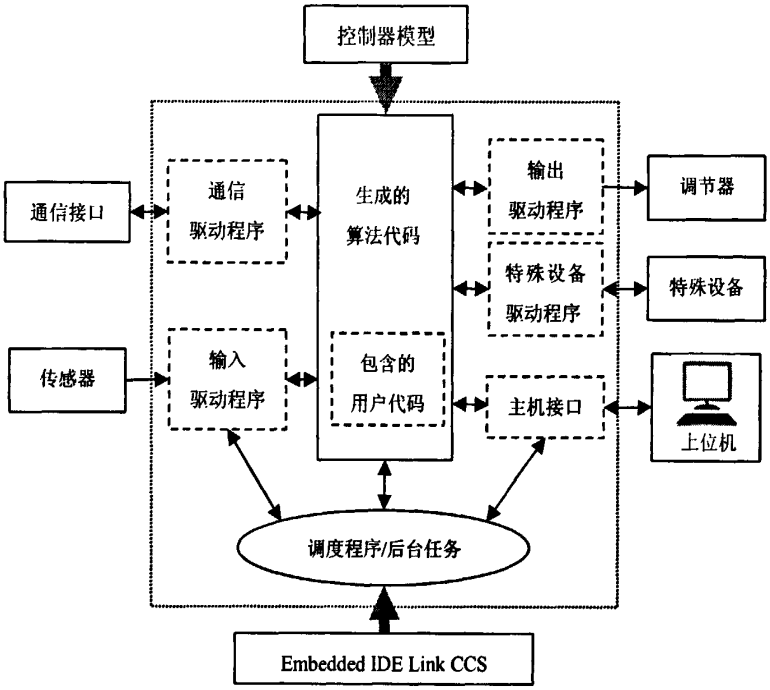


图 14 冶金电炉智能控制板固件结构

采用这种结构的目的在于最大限度地利用代码的可移植性以及代码模块的可集成性。为此，开发的代码只采用 ANSI/ISO C/C++标准[29]。

开发方案为用 MATLAB/Simulink 软件开发应用程序大部分功能代码，将手写代码集成到生成的项目代码中[31]。

控制器的算法代码通过两种途径开发，便于在 Simulink 上以模型方式或以嵌入式 MATLAB 语言子集开发的就采用基于模型的设计方法，然后生成算法代码；同时，将开发的经过验证的手写代码集成到控制器。设备驱动程序也分为两个部分，TMS320C6713 的芯片初始化部分由 Target Support Package 模块库中的模块及模型的参数配置选项共同生成，部分片上外设和板载设备的驱动程序主要手工开发，在代码生成环节集成到生成的项目代码中。调度程序和后台管理程序以及应用程序软件架构均借助于 Target Package Support 和 Embedded IDE Link 产品实现。代码转换时使用 RTW Embedded Coder 产品，生成在嵌入式系统上运行的实时代码。

在开发过程各阶段采用适当的验证技术对开发出的组件模型、系统模型从功能和性能各方面进行验证，保证最终产品能够实现预期设计目标。

3.1.3.1 设备驱动程序

控制器上的硬件主要有作为处理器的 TMS320C6713 DSP 芯片，与上位机联接的 CPCI 桥芯片以及热插拔控制芯片，基本外围电路有 2MB 的用来存储程序的 FLASH 存储芯片和 16MB 的 用来运行程序和暂存数据的 SDRAM，外部接口单元主要是 232、以太网通信模块，数字量输入输出模块及模拟量输入输出模块。

用基于模型的代码集成方式将现有的设备驱动程序集成到系统模型生成的项目中，同时，用基于模块的代码集成方式，使用 Custom Code 模块库中的模块将一行或几行设备驱动程序的函数调用语句（不必要写成函数的形式）将驱动程序的入口函数插入到系统模型生成的项目源代码文件中，这些代码要插入到初始化程序部分。系统模型生成的源代码文件 MW_c6xxx_csl.c 执行初始化功能。

设备驱动程序应用 C6000 系列芯片支持库应用程序接口函数（CSL API）来编写。TMS320C6000 CSL 是一组用来配置和控制片上外设的应用程序 C 语言接口函数。CSL 由分离的模块组成，这些模块被编译并归档到一个库文件中。除了几个模块提供通用编程支持外——比如含有中断管理应用程序接口（APIs）的中断请求（IRQ）模块，以及允许芯片整体设置的 CHIP 模块，每个模块均与单个外设相关。每个片上外设都有一个 APL 单元与之对应。CSL 具备的优势有：外设易于使用，压缩开发时间，可移植性[32][33]。

3.1.3.2 任务和调度

任务和调度是实时嵌入式应用程序软件架构的重要组成部分，任务是嵌入式系统功能的集中体现，而调度则是这些任务能否有效实现的重要保障，它们是嵌入式系统设计中的核心概念。

显而易见，冶金电炉智能控制器应用程序是一个实时多任务系统，在系统时钟的控制下执行着周期性的任务，并且能够实时地处理异步突发事件。设计模型时应充分考虑各任务和事件的明确地界定以及其优先次序，此外，还包括各任务在运行过程中是否能被更高优先级的任务抢占。下面对周期性任务和异步事件作简要地说明。

（1） 周期性任务

冶金电炉智能控制器在一个控制周期内（两个控制信号输出之间的时间）执行的任务主要有：

1) 控制信号计算及 ANN 模型在线学习，这是最高优先级的任务，控制周期为 1s；

2) 数据采集：每周期采集 100 点

a) 50Hz 交流供电，采集频率为 5K/通道，任务执行速度为：0.0002s；

b) 60Hz 交流供电，采集频率为 6K/通道，任务执行速度为：0.0002s；

3) 数据传输：与上位机之间采用 CPCI 方式连接，上位机通过 TMS320C6713 的 HPI 访问 CPU 的存储空间

a) 智能调节器到上位机

每秒钟更新一次，10 组数据；每组数据处理时间为 0.1s；

b) 上位机到智能调节器

每秒钟更新一次，处理时间为 1s；

其他周期任务的安排可以根据这些基本任务及其重要性来设定适当的运行周期，通过模块的 Sample time 参数选项来控制。为使模型层次分明、结构清晰，更为了模块重用以及模型升级方面的考虑，相关任务应做成专用的子模型或功能模块，必要时候可以封装起来。

（2） 中断资源

控制器的中断拓扑如下所示：

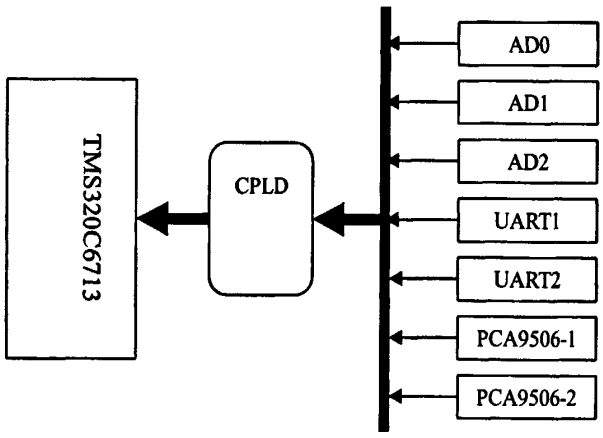


图 15 智能控制器中断拓扑

DSP 可提供的外部中断 NMI(非屏蔽)、GP4(EXT_INT4)、GP5（EXT_INT5)、GP6(EXT_INT6)、GP7(EXT_INT7)、GP1(HINT)。其中，HPI 中断，固定为 HINT(GP1)，板上其他所需中断概括如下。

- 输入扩展中断（1～2）
- AD 中断（3）
- 串口扩展（1～2）

异步事件的设计使用 Target Package Support 模块库中的模块，并且需要提供中断信号与中断源映射代码及中断服务程序使能代码。这些功能的实现可以通过 real-time Workshop 产品下的 Custom Code 模块库中的相应模块实现。在模块参数对话框中适当的位置插入执行此任务的代码行，即可在代码生成是将这些代码插入到生成的源代码文件中的适当位置。

异步事件的综合调试需要在硬件平台上来完成。代表异步事件相关的任务（也就是异步事件的中断服务程序）的子模型的功能和性能可在 Simulink 环境中验证。

（3） 异步调度

Target Support Package 支持的 TMS320C6713 DSP 异步调度有两种代码生成方式：Bare-Board 和 DSP/BIOS。该系统上优先选用 Bare-Board 模式，这是因为：

(1) DSP/BIOS 操作系统总是执行高优先级的任务。这种运行机制和某些其他的实时操作系统不同。在那些实时操作系统平台上，每个任务都可获得公平的处理时间。因此，基于这种情况，某些低优先级的任务可能永远不会被执行——因为更高优先级的任务不会被阻塞。

(2) 只有当任务执行的函数调用子系统包含的设备驱动模块受到阻塞，该任务才会受到阻塞。如果某一 DSP/BIOS 任务运行一个不包含任何设备驱动的子系统，而任务具有最高优先级，它将不会释放 CPU，实际上在应用程序中禁止了其他所有低优先级的任务。

从这两个角度考虑，为了冶金电炉智能控制系统的安全有效运行，优先考虑使用 Bare-Board 模式，生成异步任务调度程序。

（4） 控制器任务设计

根据系统需要，初步划分的任务如下表所示：

表 6 任务分配初步设计

进程名称	任务编号	任务
系统管理	T0	系统启停
数据采集	T11	数据采集
	T12	中间变量计算
	T13	RTDB 刷新
调节器输出	T21	智能 PID 计算
	T22	ANN 调节器计算
	T23	综合输出
	T24	进程通讯
ANN 参数优化	T31	炉子仿真 ANN 权值优化
	T32	调节器 ANN 权值优化
	T33	进程通讯
网络通讯	T41	HPI 主机通信

3.1.3.3 数据存储

冶金电炉智能控制器采用 2MB 的 Flash 存储程序，地址映射到 CE1 空间；16MB 的 SDRAM 用作程序的运行和数据缓存，映射到 CE0 空间；AD、DA 以及串口扩展等映射到 CE3 空间。存储空间需在 Custom Board 模块参数配置选项卡上进行相应的配置。

在设计系统模型时，需要自定义数据类型。在 Simulink 环境中建模时，除了可以使用系统提供的内嵌数据类型，如 Single、double 等，还允许用户定义自己的数据类型和数据对象，并且在自动代码生成时，可以控制数据对象在存储器中的位置。用户自定义的数据对象还可以用到模型的参数中和模块之间的连接信号上。

Embedded IDE Link 模块库中提供了以下两个存储器操作的模块：

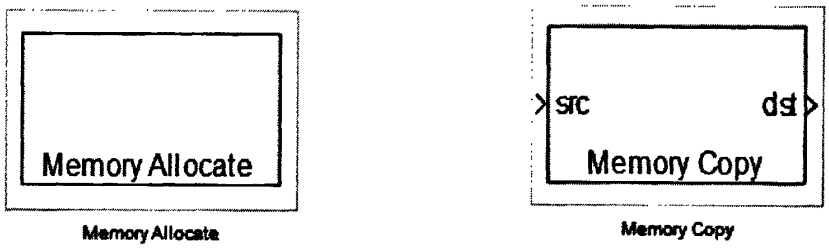


图 16 存储器操作模块

上图左侧模块是存储器分配模块，该模块控制 TI 编译器为定义的新变量分配存

储空间，该模块的参数可以用以指定变量名称、变量在存储器中的对齐方式、变量的数据类型以及完整定义存储器特征。右侧模块是存储器复制模块，该模块在定义的处理器的存储空间内复制变量或数据。配合使用这两个模块可以实现程序所需数据类型的定义和操作。

灵活的数据类型定义可以为冶金电炉智能控制系统模型的开发带来极大的便利。

3.1.3.4 在线调节器

在线调节器是由调节器和仿真器两部分组成的，都采用了含有一个隐含层的人工神经元（ANN）模型。调节器的计算结果输入仿真器，仿真器计算之后利用误差反向传播机制调整调节器的输出，反复进行这个过程直到调节器的输出达到精度要求为止。

该部分的核心是建立 ANN 模型，网络结构和权值训练可以借助 MATLAB 的 Neural Network Toolbox（神经网络工具箱）[27]来完成。神经网络工具箱是 MATLAB 提供的该专业领域的功能函数集，是基于 MATLAB 编程语言的。此外，与该工具箱对应的还有 Simulink 模块库用来建立神经网络结构模型。在 MATLAB 命令行键入：

```
>> neural
```

即可打开神经网络模块库，如下图所示：

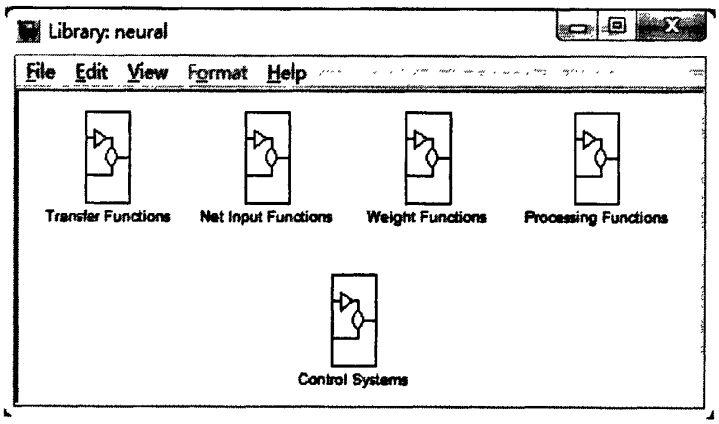


图 17 Simulink 的神经网络模块库

模块库中有：

- Transfer Functions（传递函数）：每个模块都可接受网络输入向量，产生相应的输出向量，输出向量的大小与输入向量的大小相同；
- Net Input Functions（网络输入函数）：每个模块都可接受任意数量的加权输入向量、权值层输出向量以及偏置向量，返回网络输入向量；
- Weight Function（权值函数）：每个模块均可接受一个神经元的权值向量，将该向量作用到输入向量（或者一层的输出向量）以产生神经元的加权输入值；
- Processing Functions（处理函数）：含有五组数据处理模块，每组有两个模块组成，分别用作数据的预处理和后处理，即输入输出数据的处理和逆处理；
- Control Systems（控制系统）：包含三个神经网络控制器模型——Model Reference Controller、NARMA-L2 Controller 以及 NN Predictive Controller，其参数可以调整。

在开发模型时，可以利用 Neural Network Toolbox 中的函数建立神经网络模型，用下面的函数和语法将神经网络对象转换成 Simulink 模型：

gensim(net,st)

其中第一个参数 net 是待转换为 Simulink 模型的神经网络对象，而第二个参数 st 是生成模型的 sampling time，是控制该模型运行的参数选项。

3.1.3.5 PID 保驾算法

PID 算法的应用是出于安全方面的考虑，如果智能调节器出于环境的影响而不能给出有效的控制，综合输出判断模块可裁定输出有效的控制。

PID 算法的实现可以借助 Simulink 中的 PID 控制器模块来实现。借助冶金电炉模型，在 Simulink 环境下可快捷地调节 PID 的参数，满足控制对象性能指标的要求。

3.1.3.6 数据采集和数据处理

数据采集和硬件设备相关，可分成两个阶段来实现。前期设计与硬件无关的数据处理算法，利用现场数据在 Simulink 环境下仿真验证、改进模型。后期集成设备

驱动程序，以快速原型或利用 HIL 方式验证该功能模块的性能。

3.2 钢液温度预报的设计实现

温度预报是冶金电炉智能控制系统的重要核心技术之一，神经网络的实现是温度预报模块建立的关键所在。以钢水温度预报模型[2]的建立过程说明基于模型的设计方法在冶金电炉智能控制系统设计中的具体实现方案。

3.2.1 钢液温度预报模型

钢水温度预报的模型是以神经网络为核心的，网络结构采用输入层、中间层和输出层的三层 BP 神经网络。根据对现场参数的分析，确定输入层的节点个数为 8，中间层的神经元个数为 5，输出层的神经元个数为 1。如下图所示：

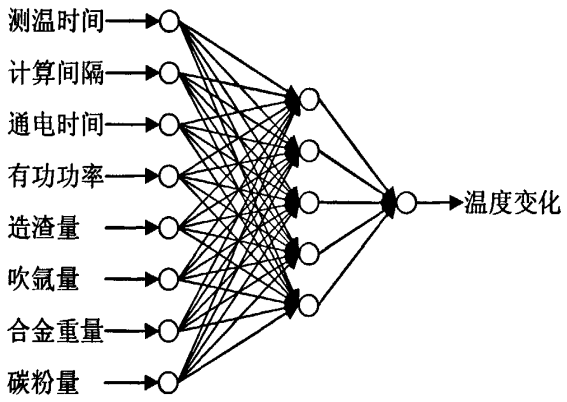


图 18 温度预报神经网络结构

为了提高预报精度，根据流程工艺分析出对温度有明显影响的因素，建立专家规则。主要包括以下几个方面：

（1） 钢液加热理论：升温时，

- 第一阶段：前 8 分钟，升温速度较缓，温升速度在 1.0℃/分钟左右；
- 第二阶段：第 8～12 分钟，温升速度大约在 1.0～2.0℃/分钟；
- 第三阶段：从第 12 分钟开始，温升速度可达 2.5～4.2℃/分钟；

（2） 钢液降温规律：热停时，吹氩强度不大时（＜100L/分钟）温降速度不大于 2℃/分钟；

（3） 钢包烘烤程度对钢液温度的影响：新包不超过 2℃/分钟，正常包不超过 6℃/分钟

（4） 加入合金对钢液温度的影响：

- 硅铁和铝块有利于增温
- 其他合金会引起钢液温降

专家规则子模型的建立主要分成两个部分：升温模型和降温模型。升温模型主要和二次电流设定点、钢包烘烤程度、钢包状态冶炼时间以及加入合金的种类有关，而降温模型主要和连续热停时间、吹氩量以及加合金的种类有关。结合专家规则，温度预报的模型结构如下所示：

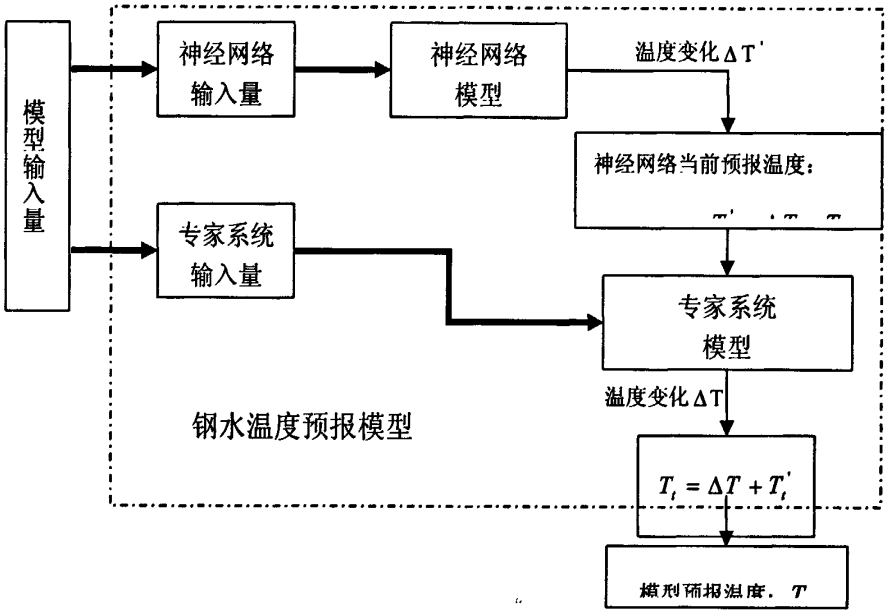


图 19 钢水温度预报模型功能结构

3.2.2 温度预报模型的建立

在 MATLAB/Simulink 软件平台上，建立温度预报模型。从前面的分析设计可以知道，钢水温度预报模型主要分成 BP 神经网络模型部分和专家系统模型部分。BP 神经网络部分利用现场采集数据，利用神经网络工具箱中的函数建立并训练网络参数，然后生成 Simulink 模型；专家规则部分直接在 Simulink 环境下利用 Simulink 模块库中的模块建模。两部分模型整合到一起，就形成钢水温度预报的智能模型。

（1） BP 神经网络模型

从现场数据中整理出具有代表性的输入（8 个变量）、输出参数（1 个变量）

集，建立 BP 神经网络模型：

```
tempANN = newff(P,T,5,{'logsig','purelin'}); %P为输入参数集，T为实测温度
                                              %中间层神经节点数为5
```

训练 BP 神经网络：

```
tempANN = train(tempANN, P, T);
```

此时，调用神经网络训练工具对该 BP 神经网络进行训练，如下图所示：

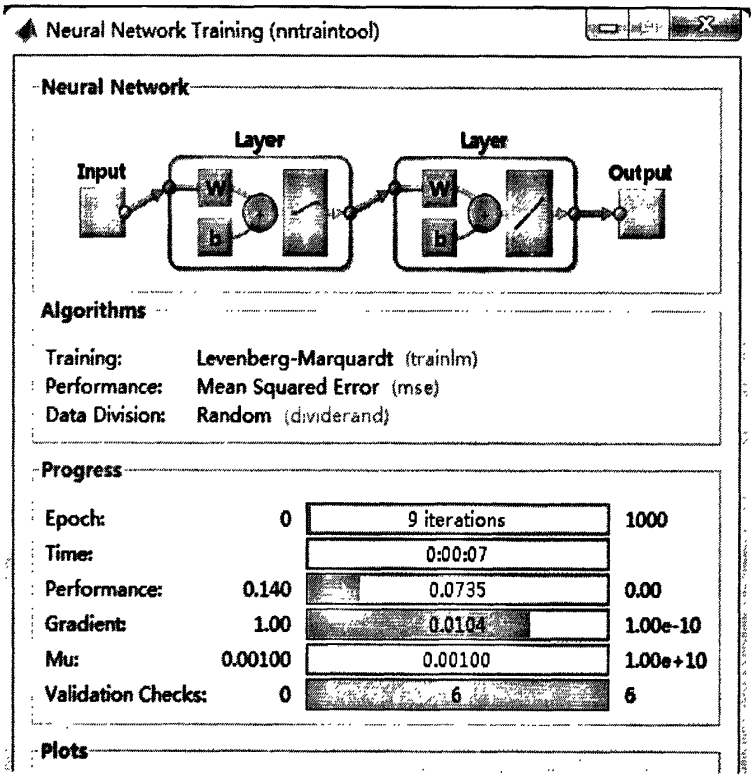


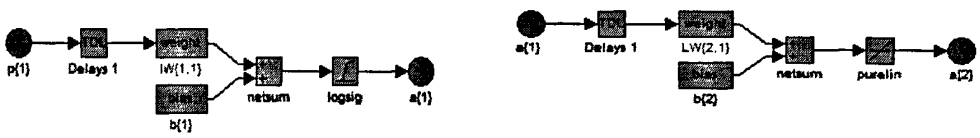
图 20 神经网络训练工具训练温度预报 ANN 模型

训练满足精度要求以后，模型连接参数存储在模型中。

将训练好的神经网络模型生成 Simulink 模型：

```
gensim(tempANN, 10);    %预报周期为10秒
```

生成的模型如下图所示：



左：ANN 中间层结构　右：ANN 输出层结构

图 21 BP 神经网络 Simulink 模型结构图

任选一炉初始钢水重量为 121.1 吨、钢种号为 2228K 的现场数据，测试温

度预报的 ANN 模型。模型的测试结果如下图：

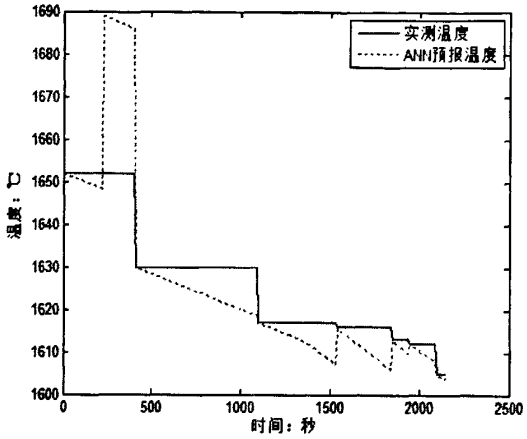


图 22 某炉次实测温度及 ANN 预报温度曲线

（2） 专家规则模型

专家规则模型主要是根据输入参数对神经网络模型的预测输出进行一些判断，在冶炼的具体阶段该 ANN 模型的输出是否合理，在 Simulink 模型的建立上，采用判断模块以及判断条件执行模块来实现。在升温规则部分和降温规则部分设定输出限幅模块：如果 ANN 输出在当前温度变化的许可范围内，直接输出 ANN 预测值；如果 ANN 输出的当前温度变化超过许可范围，则直接输出设定的上限值或下限值。专家规则模型如下图所示：

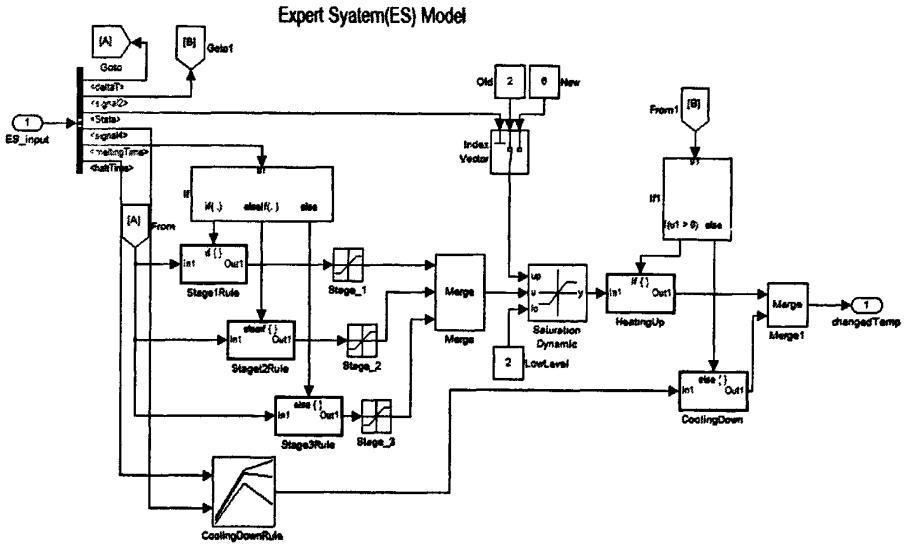


图 23 专家规则模型

3.2.3 温度预报模型的代码实现

将温度预报的神经网络模型和专家规则系统整合到一个模型中，得到冶金电炉钢液温度预报的智能模型，如下图所示：

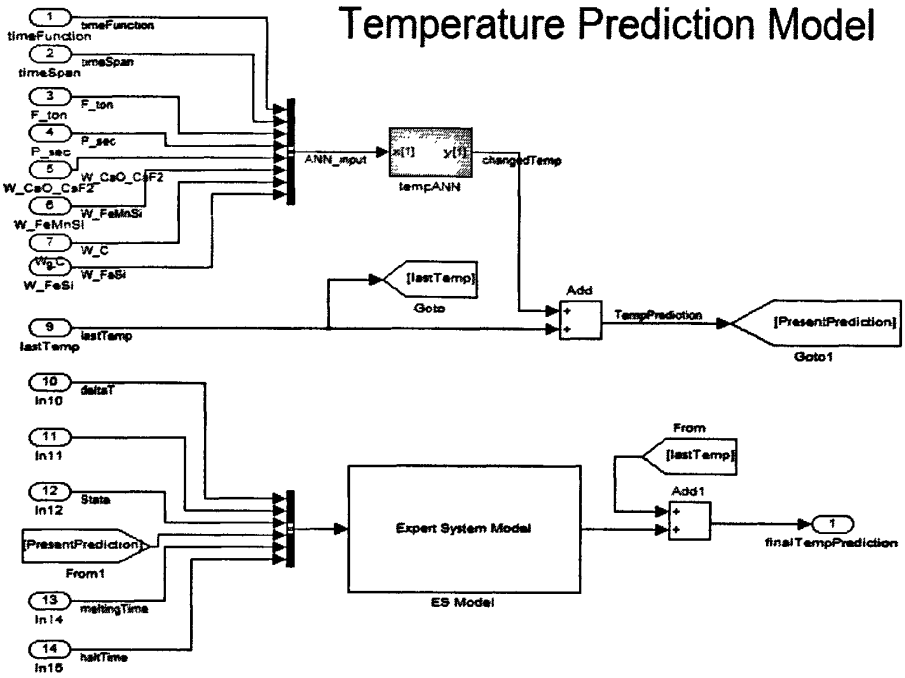


图 24 冶金电炉钢水温度预报智能模型

使用验证温度预报的 ANN 模型的现场数据测试该模型，得到的 ANN 模型以及整个智能预报模型的温度预报结果连同实测的温度曲线如下图所示：

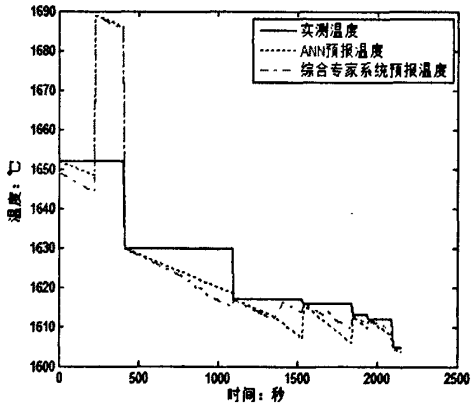


图 25 温度预报智能模型运行曲线

设计好的模型可以利用 RTW 技术（产品组件）将其自动生成目标平台的 C 代码，有效避免了手写代码的繁重任务。在 Simulink 模型的 **Configuration Parameters**

选项卡中配置适当的参数，选择系统目标文件，点击 Incremental Build 按钮，生成模型的 C 代码程序。

第四章 总结与展望

4.1 研究总结

本研究是冶金电炉智能控制系统产品研发项目中的一个环节，完成了预期研究目标：确定冶金电炉智能控制系统的开发方案，把握开发过程中的关键技术，在目前阶段、以现有条件研究基于模型设计在冶金电炉软件系统设计中的应用。根据研究结果，选用基于模型设计方法作为冶金电炉智能控制系统嵌入式固件开发方案。研究过程中分析了该项技术具体实现流程，掌握了关键的实现技术。为研发团队建立基于模型设计方法的文化打下基础[34]。

基于模型的设计方法起源于控制系统领域的应用，作为解决控制系统分析与综合过程中固有的难度和复杂度而出现的一种有效手段。这是一种解决问题的方法论，能够有效解决嵌入式系统软件设计中的许多难题。

软件设计是方法论和工具的统一。MATLAB/Simulink 软件平台是实现基于模型设计方法优秀的工具。在本课题研究范畴内，结合 TI 公司的 CCS 集成开发环境，可以实现基于模型设计方法的全部环节。从产品功能定义及性能要求，到概念模型的开发、自动代码/项目生成，以及最后的综合调试，提供的各种验证手段能充分保证预期功能和性能的实现。

对自动生成代码结构以及运行机制细致深入地研究可以使冶金电炉智能控制系统研发过程中采用更灵活的方式控制代码的生成，并实现现有资源与自动生成源代码的无缝集成。

相对于传统的产品研制过程，基于模型的设计在嵌入式系统开发中具有独特的优势。基于模型的设计方式使开发者在项目的开始就可以进行设计的验证和确认，用快速原型和自动代码生成（产品代码）手段测试系统中部分模块。这些特点提供给开发者明显的优势。从设计/项目管理角度来看，开发过程中任何时刻都能验证设计使得能够更精确地管理预期研发时间和费用。基于模型的设计还引入了“设计重用”的概念，曾经设计并通过验证的模块可在以后设计中重用，进而提高了效率。基于模型的设计能非常便捷地更换系统中的模块，修改或添加系统功能，便于系统

升级和维护。

自动代码生成能够生成具有精湛技术程序员水平的代码并根据系统的应用需要进行代码优化，减轻了嵌入式系统开发者的强度。硬件平台升级后，只需要付出最小代价设计新硬件平台设备驱动程序，在代码生成时修正目标环境参数就可升级软件系统，便于系统升级与维护。

4.2 后续工作展望

由于本人对冶金电炉智能控制板实现的功能及要求的性能指标并不十分了解，而目前尚无详实的技术说明文档可供参考，对基于模型的设计方法在冶金电炉智能控制系统中的具体应用技术的研究未能具体到工程实现层面。条件的限制使本课题对该项技术的研究更多集中在自主研发的基于 DSP 的智能控制器代码实现上，未能对模型建立过程作过多地探讨。

就基于模型设计这项技术而言，研究表明其较传统设计方法具有非常明显的优势。应该继续深入研究，在嵌入式系统软件开发过程中有计划的应用。MATLAB/Simulink 作为该项技术的实现工具，功能强大，已经广泛应用到航空航天、国防、电力、汽车电子等多个领域。但该工具涉及多个学科的知识，要想充分发挥技术方案的优势，必须灵活掌握工具软件的应用。为项目开发进展顺利，需要组建一个技术团队。

本课题在研究过程中对冶金电炉智能控制系统的设计中关键技术的实现进行了探索，取得一定的成果。但是具体的实现细节，尚需在项目进入实施阶段继续深入探索研究。

参考文献

[1] 冶金自动化研究设计院，冶金电炉智能控制板规格设计书，V 0.1，2009 年 6 月；

[2] 冶金自动化研究设计院，冶金电炉智能控制系统功能规格设计书，V2.0，2008 年 1 月；

[3] Jerry Krasner，Model-Based Design and Beyond: Solutions for Today's Embedded Systems Requirements，EMBEDDED MARKET FORECASTERS, American Technology International. January 2004.

[4] Ali Behboodian，Model-based Design，DSP magazine，52-55，May，2006；

[5] Jerry Krasner, Model Driven Development of Certifiable Software: A Best Practice for Safety-Critical Applications，EMBEDDED MARKET FORECASTERS，American Technology International. December 2008.

[6] Dick Benson, The Design and Implementation of a GPS Receiver Channel, DSP magazine, 50-53, October, 2005;

[7] Jim Tung, Using model-based design to test auto embedded software, http://www.eetimes.com/showArticle.jhtml?articleID=202100792 ,09/24/2007;

[8] 汪洋，郭丽丽，樊丽萍，一种基于 Matlab 的 DSP 开发思路研究，控制工程，2006 年，第 13 卷，增刊，123-126；

[9] 段国强，陈月云，MATLAB 辅助 DSP 设计的研究与实现，微计算机信息，2007 年，第 23 卷，第 7-2 期，130-132；

[10] 刘剑，李凌，MATLAB/RTW 环境下 PIC 微控制器开发平台的研究与应用，科技信息，2009 年，第 5 期，67-68；

[11] 冷斌，李学勇，刘建华，一种基于 Matlab 的 DSP 调试及直接代码生成方法，现代电子技术，2008 年，第 20 期，68-71；

[12] 淮文军，MTALAB 软件在 DSP 程序开发过程中的应用，苏州职业大学学报，2008 年，第 19 卷，第 1 期，92-94；

[13] 齐星刚，赵刚，李原，在 MATLABMATLAB/Simulink 平台上 DSP 代码

的自动生成，中国测试技术，2005 年，第 31 卷，第 1 期，87、88-106；

[14] 李真芳，苏涛，黄小宇，DSP 程序开发——MATLAB 调试及直接目标代码生成，第一版，西安电子科技大学出版社，2003 年 10 月，395 页；

[15] The MathWorks, Simulink® 7 User's Guide, Version 7.4, the MathWorks(Online Only),September 2009, 1528;

[16] The MathWorks, Embedded MATLAB™ User's Guide, Revised for Release 2009b, the MathWorks(Online Only),September 2009, 287;

[17] The MathWorks, Real-Time Workshop® 7 Getting Started Guide, Version 7.4, the MathWorks(Online Only),September 2009, 129;

[18] The MathWorks, Real-Time Workshop® 7 User's Guide, Version 7.4, the MathWorks(Online Only),September 2009, 1159;

[19] The MathWorks, Real-Time Workshop® Embedded Coder™ 5 User's Guide, Version 5.4, the MathWorks(Online Only),September 2009, 1000;

[20] The MathWorks, Embedded IDE Link™ 4 User's Guide: For Use with Texas Instruments' Code Composer Studio™, Version 4.0, the MathWorks(Online Only),September 2009, 471;

[21] The MathWorks, Target Support Package™ For Use with TI's C6000™ 4 User's Guide, Version 4.0, the MathWorks(Online Only),September 2009, 493;

[22] The MathWorks, Real-Time Workshop® 7 Target Language Compiler, Version 7.4, the MathWorks(Online Only),September 2009, 451;

[23] The MathWorks, Real-Time Workshop® Embedded Coder™ 5: Developing Embedded Targets, Version 5.4, the MathWorks(Online Only),September 2009, 253;

[24] The MathWorks, Simulink® 7 Writing S-Functions, Version 7.4, the MathWorks(Online Only),September 2009, 889;

[25] 肖兵，蔡一波，梁瑛琳，MATLAB 和 DSP 的滤波器硬件在环实时仿真，理论与方法，2007 年，第 26 卷，第 10 期，10-13、20；

[26] Horgan, J. .Hardware/Software Co-verification, EDA Café Weekly. March 29, 2004.

[27] Howard Demuth, Mark Beale, Martin Hagan, Neural Network Toolbox™ 6 User's Guide, Version 6.0.3, the MathWorks(Online Only),September 2009, 901;

[28] Texas Instruments Incorporated，TMS320C6000 Chip Support Library API Reference Guide，Version J，August，2004，1108；

[29] Texas Instruments Incorporated，TMS320C6000 Programmer's Guide，Version I，March，2006，440；

[30] 罗蕾主编，嵌入式实时操作系统及应用开发，第二版，北京航天航空大学出版社，2007 年 3 月，403；

[31] [美]Ed Sutter 著，张晓林等译，嵌入式系统固件揭秘，电子工业出版社，第 1 版，2003 年 6 月，328；

[32] 合众达，SEED-DEC6713 用户指南，版本号：C，2007 年，126 页；

[33] Texas Instruments Incorporated（著），卞红雨等编译，TMS320C6000 系列 DSP 的 CPU 与外设，第 1 版，清华大学出版社，2007 年 12 月，512 页；

[34] Paul F. Smith, Sameer M. Prabhu, Jonathan H. Friedman，Best Practices for Establishing a Model-Based Design Culture，SAE，April 2007

攻读硕士学位期间主要工作和发表论文

攻读硕士学位期间的主要研究工作有：

1. 863 课题“冶金电炉智能控制系统”中基于专家规则与计算智能的冶炼过程温度预报技术研究：研究冶金电炉智能温度预报模型，实现冶炼过程中满足精度要求的实时温度预报技术并建立模型。
2. 冶金电炉智能控制板的算法开发及固化技术研究：研究冶金电炉智能控制板上运行算法/任务的代码实现技术。

发表的论文：

1. 张德龙,刘海东,卢福勇。基于 MATLAB 平台的 DSP 嵌入式应用程序设计的研究。工业计量。

致 谢

在本研究论文完成、硕士研究生学习即将结束之际，我诚挚地对在此期间给我帮助的老师、同学、朋友以及亲人致以深深的谢意！

首先感谢我的导师房庆海博士：他以广博的学识、深邃的学术洞察力和丰富的实践经验循循善诱地把我引导进入嵌入式系统固件研发领域并指点正确的研究方向，使我综合技术水平得到显著提高，为以后工作打下坚实基础。房庆海博士严谨的治学态度、精益求精的工作要求和淡泊的心境是我一生的榜样，激励我在工作和生活中自强不息。

还要感谢冶金自动化研究设计院的领导对我无微不至地关怀，尤其是人教部庞云访老师以慈母般的情怀时刻关注着我的学习和生活。她的鼓励和支持使我在重重困境中坚强下来，在自动化院成长与进步。

吴少波博士、金传富博士、石志学、王金峰以及吉文杰等都曾就我的研究课题对我加以悉心指点或提供热情的帮助，没有他们我难以取得现在的高度，在此对他们深表感谢！

我的亲人、同学和朋友都在背后关注和支持我，给我生活和研究的动力，他们支持我克服了很多困难，给我生活的勇气和希望。我以努力学习、积极地生活回报他们对我无私的奉献。

最后，衷心地感谢参加论文评阅和学位答辩的各位专家、学者和老师，感谢你们的辛苦劳动和提出的宝贵意见！

硕士研究生阶段的学习即将结束，我的生命又翻开一个新的篇章，带着各位尊敬的老师、亲爱的同学和朋友以及亲人的祝福，我将以积极的心态、勤恳的工作态度去迎接新的挑战、把握新的机遇！

基于模型的设计方法在冶金电炉智能控制中的应用

作者：张德龙
学位授予单位：冶金自动化研究设计院