doi:10.6041/j.issn.1000-1298.2013.04.002

# 电控机械式变速器控制系统 UML 建模与实现*

綦声波[1]　纪风磊[1]　于敬东[2]
(1. 中国海洋大学工程学院, 青岛 266100; 2. 烟台森科特智能仪器有限公司, 烟台 264100)

**摘要**: 引入面向对象的分析方法,采用统一建模语言 UML 对电控机械式变速器控制系统的功能、对象结构、行为和活动以及实现方式等建立了一整套模型。UML 模型经过适当转换,可以基于量子平台实现从模型到代码的自动生成。该方法在开发、维护和代码重用等方面较传统的前后台系统开发模式具有明显优势。

**关键词**: 电控机械式变速器　统一建模语言　量子平台
**中图分类号**: TP23; U463.6　　**文献标识码**: A　　**文章编号**: 1000-1298(2013)04-0008-07

# Modeling and Implementing of AMT Electronic Control System Based on UML

Qi Shengbo[1]　Ji Fenglei[1]　Yu Jingdong[2]
(1. *College of Engineering*, *Ocean University of China*, *Qingdao* 266100, *China*
2. *Yantai Sencott Intelligent Instruments Co.*, *Ltd.*, *Yantai* 264100, *China*)

**Abstract**: Object-oriented analysis methods were introduced and a set of models were built based on the unified modeling language (UML), such as the function of the system, object structure, behavior and activities, as well as implementation of the system. After appropriate conversation, and with QM tool, the code could be automatically generated directly from UML model based on quantum platform. Compared to the usual development method, there were obvious advantages of the proposed method in development, maintenance and code reuse, etc.

**Key words**: Automated mechanical transmission　Unified modeling language　Quantum platform

## 引言

自动机械式变速器(AMT)控制系统为典型的嵌入式汽车电子系统。传统方法一般使用前后台系统进行开发[1]。在实际的开发中,AMT 系统需要作大量的实车测试和性能分析,并根据测试结果优化算法,重写代码。AMT 系统对实时性要求高,需精确的时序控制。在前后台系统中,微小的代码改动都可能改变整个系统的时序,从而需要对全部的功能模块重新测试。实践表明,前后台开发方法耗时长,效率低,代码可维护性差,功能扩展困难。对于高实时性、高可靠性的嵌入式系统,亟待新的开发方法[2]。

文献[2]提出了"五层/三总线/一框架"的开放性总体系统结构,提出了软硬件协同开发的方法。文献[3]提出了用量子框架来构建开放式汽车电控系统体系结构的思路,但没有提出一个具体的解决方案。文献[1~2]都是对新方法的有益探索,但对加快开发速度和提高编码质量,都未能提供切实有效的具体方法。

本文引入面向对象的分析方法,基于 UML 建立 AMT 控制系统的完整模型。通过量子平台对活动对象和状态机的支持,实现从 UML 模型自动生成嵌入式系统中使用的 C 代码。

## 1　基于 UML 的 AMT 模型

统一建模语言(UML)是一种规范化的建模语言,可以用来构建软件系统完整且精确的蓝图。UML 用一系列的图来描述系统的静态构成和动态行为。其中用例图主要用来进行功能描述;类图用

收稿日期: 2012-04-08　修回日期: 2012-12-08
*国家自然科学基金资助项目(50575129、51075241)
**作者简介**: 綦声波,副教授,博士,主要从事嵌入式技术及汽车电子研究,E-mail: qishengbo@ouc.edu.cn

来描述系统的静态结构；序列图、活动图和状态图用来描述系统的动态行为。采用 UML 分析和实现 AMT 控制系统的建模应遵循从抽象到具体，从整体到局部的步骤，具体为[4]：①分析需求，建立系统功能模型用例图。②类的抽象与对象分析，建立简单类图。③系统活动分析，建立系统的工作流模型。④类的交互分析，建立顺序图或协作图。⑤类的行为分析，建立类状态图模型。⑥从系统动态模型中分析类的详细结构，建立系统静态模型。⑦用部署图和组件图构造系统。⑧编程实现 AMT 系统的控制要求。

限于篇幅，本文简要描述 AMT 系统的建模过程，建模工作都是用开源的 CASE 工具 StarUML 完成的。

### 1.1　AMT 系统功能分析

如图 1 所示，AMT 系统主要工作原理是：AMT 控制单元 TCU 通过传感器测量车辆参数，如车速、加速度、发动机转速、油门踏板深度、刹车信号、操纵杆信号等。根据程序设定的换挡算法计算出目标挡位，并控制直流电动机完成换挡动作。换挡期间，TCU 通过控制节气门的开度，避免发动机超速或熄火。

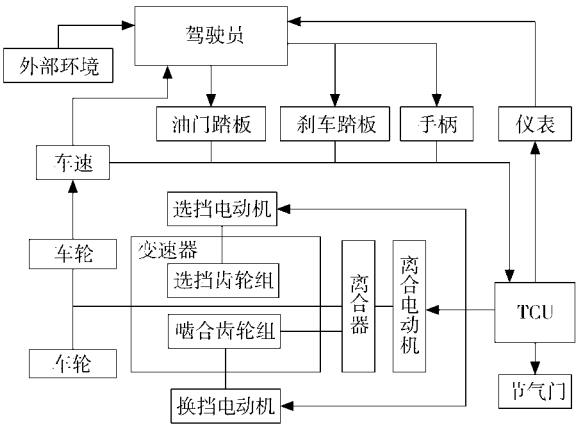



图 1　AMT 控制系统示意图

Fig. 1　Control system of AMT

#### 1.1.1　AMT 系统功能划分

AMT 系统的主要功能由下 4 个部分组成。

(1)驾驶相关部分

对驾驶员的操作请求做出反应。可以进行模式选择，驾驶员可以选择手动/自动模式，也可以在经济/普通/动力 3 种换挡策略中选择其中一个。可以进行点火、起步和倒车。在手动模式下，可以通过操纵杆来进行手动换挡；在自动换挡模式下，主要通过油门踏板和刹车发出请求，控制器会根据当时的车速、加速度和发动机转速等情况自主决定换挡时机和挡位。

(2)控制相关部分

即控制器的自主行为。信号采集即通过采集传感器的信息来获得汽车当前的车速、挡位和发动机转速等信息。在线学习是通过对驾驶员的操作进行记录和分析，计算出驾驶员的驾驶风格，选择相匹配的换挡策略。执行器控制算法对执行器(选挡、换挡和离合运动所用的电动机)进行运动控制。

(3)系统安全相关部分

系统安全部分包括故障诊断和应急策略两部分。故障诊断可对 AMT 系统的硬件、软件和机械结构进行在线实时监测，在系统功能失效的情况下(如部分传感器故障)，使用应急策略维持系统主要功能的运行。

(4)调试接口

这是提供给开发者或者维修人员使用的接口。离线的故障诊断用来判断系统的错误，提出维修建议。固件升级可以更新系统的固件。系统评价可以帮助开发者确定系统性能是否达到预期目标。程序员可以通过调试接口读取或修改系统的关键参数。

#### 1.1.2　AMT 系统的用例图

功能需求在 UML 中是由用例图描述的[4~6]。用例图(图 2)是站在用户的角度，对系统的功能需求进行分析。AMT 系统的主要使用者是驾驶员、开发者和维修人员。驾驶员会使用点火、换挡等与驾驶相关的操作；开发人员会使用调式接口；维修人员会使用故障诊断等功能。

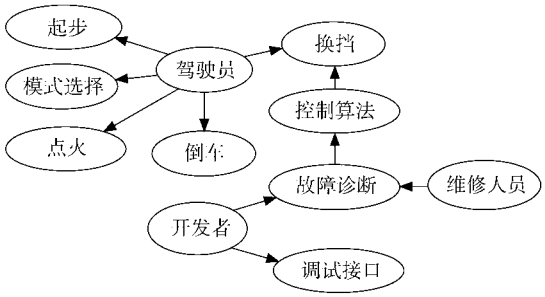



图 2　用例图

Fig. 2　Diagram of use case

#### 1.1.3　用例细化

每个用例都需要进行细化，以描述该用例的细节。UML 对用例的细化并没有规范。可以用任何 UML 元素描述各个用例的细节。限于篇幅，仅对起步用例和倒车用例进行细化。为了体现系统各个单元在时间轴上的交互，这些细化用例采用序列图表示，如图 3 ~4 所示。

汽车起步用例的初始条件是在发动机转速大于或等于怠速，汽车速度为零。触发条件是驾驶员挂入 D 挡，松开手刹，踩下油门，手动模式下还需要驾

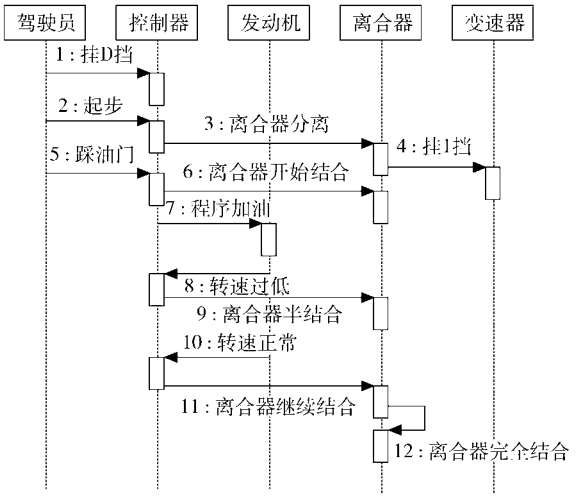



图 3 汽车起步用例序列图

Fig. 3 Sequence diagram of vehicle starting use case

驶员挂入 1 档。主流程是:离合器分离,挂入 1 挡,程序控制离合器结合,起步完成。在离合器结合的过程中,需要程序加油,避免发动机速度过低而熄火。如发动机转速仍然过低,需控制离合器退回到半结合点。用例的结束条件是离合器完全结合,汽车速度大于零,发动机转速大于怠速、当前挡位为 1 挡。

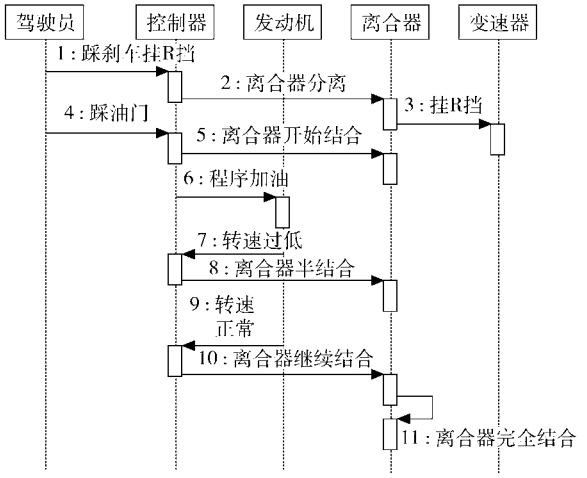



图 4 汽车倒车用例序列图

Fig. 4 Sequence diagram of reverse starting use case

倒车用例的初始条件是发动机转速大于或等于怠速,汽车速度为零。触发条件是驾驶员踩刹车,挂入倒挡。主流程是程序控制离合器结合。离合器控制细节类似于起步控制。结束条件是汽车速度大于零、发动机转速大于怠速、当前挡位为倒挡。

### 1.2 AMT 系统的静态模型

对用例图进行分析,可以分析抽象出系统中的类,以及类与类之间的关系[6]。系统的简单类图如图 5 所示。

驾驶员通过显示模块获得当前的挡位信息,并使用操纵杆、刹车踏板、加速踏板发送换挡请求。策略类分析来自驾驶员的命令,综合当前的车速和加速度,完成目标挡位的决策。协调类会根据策略类和安全类的信息,分发换挡指令给发动机、离合器和变速箱,完成换挡动作。其中,安全类作为具有辅助功能的类,也可向协调类发送特定的请求。发动机类中包含节气门类。变速器类中包含换挡类和选挡类。

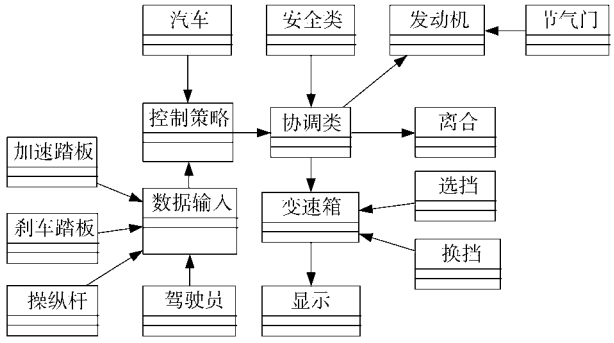



图 5 AMT 系统静态模型

Fig. 5 Static model of AMT system

在图 5 中,每个类元素都封装了自己的属性和方法,相当于一个子系统。面向对象的分析方法,可以将一个大系统分解成若干个功能相互独立子系统,降低了系统耦合度,便于系统分析和设计。每个类元素都可以独立测试,从而减少了测试的次数,缩短了测试周期。

### 1.3 AMT 系统的动态行为

AMT 系统的动态行为包括[4~6]:①系统的业务流,通常使用活动图来表示。②类和类之间的关系,包括序列图和协作图。序列图用来表示以时间顺序来进行的对象间的交互,协作图则用来表示对象间的交互及协作关系。③对象的行为,包括对象状态和一定条件下的状态转移,主要用状态图来表示。

图 6 表示了 AMT 系统的基本业务流。点火后,挡位回空挡,然后启动发动机。AMT 系统是手自一体的,驾驶员可以选择手动模式或自动模式。手动模式通过操纵杆决定升挡和倒挡,自动模式则通过 TCU 内置的算法自主决定目标挡位和换挡时机。发动机启动后,汽车可以倒挡起步,也可以挂 1 挡起步。一旦起步成功,汽车可以在换挡策略的控制下,升挡、降挡、回空挡和倒车,直至发动机熄火。除此之外,系统还应包括错误检测、错误处理、程序升级等业务流程,这些本文不再详述。

类和类之间根据时间顺序而进行的交互可以用序列图表示[5~6]。图 7 表示换挡过程的序列图。策略类计算出目标挡位,向协调类发出换挡指令;协调类负责分配换挡动作,首先使离合器分离,然后使用电控直流电动机控制变速器换挡;与此同时,离合器

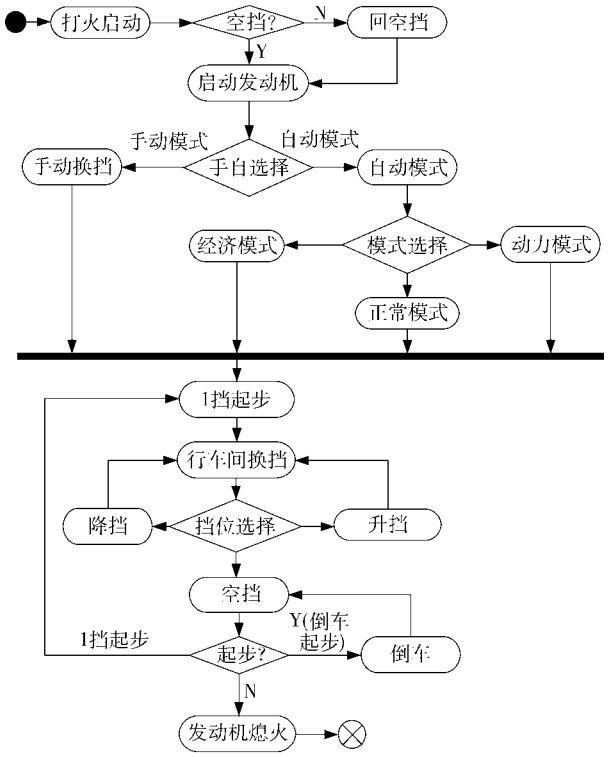



图 6　AMT 系统业务流

Fig. 6　Business flow of AMT system

分离使发动机的负载转矩减小，此时应减小节气门开度，避免发动机超速。换挡结束后，离合器开始结合；为避免离合器结合过程中发动机熄火，此时应加大节气门的开度。

图 7 的序列图表述了换挡过程的基本动作序列。实际过程中还应包括超时处理和故障处理等。每个类都有自己的控制细节，如节气门的控制算法和电控电动机的控制流程等，这些也可通过状态机实现。

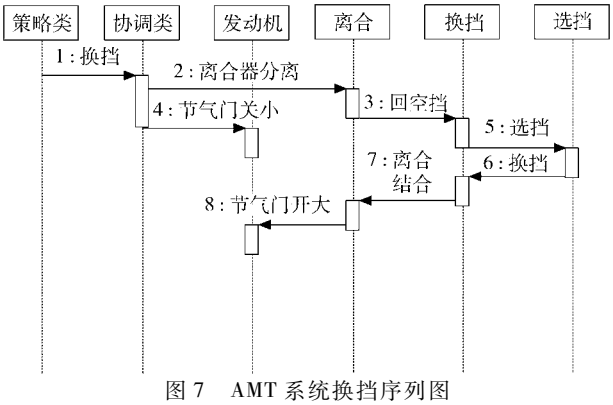



图 7　AMT 系统换挡序列图

Fig. 7　Shift sequence diagram

### 1.4　AMT 系统实现方式的建模

AMT 系统的实现方式建模，主要是通过组件图和部署图来实现的。图 8 表示了 AMT 组件图。

AMT 系统的程序代码由系统代码和用户代码两部分组成。系统代码封装了硬件细节，对用户代码提供统一接口，并且提供一个模型驱动框架。用户代码是和具体功能相关的，是 UML 模型的实例化。

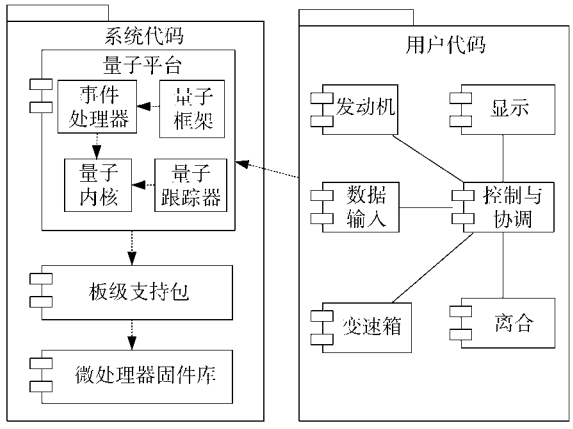



图 8　AMT 系统组件图

Fig. 8　Component diagram of AMT system

为使代码有很好的移植性，系统代码分为 3 层，即处理器抽象层、板级驱动层和量子平台层。层与层之间依靠封装好的标准 API 函数进行调用。处理器代码库包含了处理器的启动代码、时钟配置、处理器外设的驱动函数等。固件库封装了芯片的寄存器细节，使用户不用直接操作寄存器，而是使用 API 接口函数操作硬件。代码一般由处理器的生产商提供，具体实现细节依赖于芯片型号和固件库的版本。板级支持层(BSP)是和硬件相关的，包括电路板的驱动程序和 API 接口，使板级电路的实现细节对用户是透明的。板级支持包一般由硬件工程师设计、编写和测试。量子平台是一个轻量级的状态机引擎，适合在嵌入式系统中使用。

用户代码是 UML 模型的具体实现。量子平台中，用户代码是以活动对象为基本单元进行组织的[7]。用户代码中的活动队形包括发动机、显示模块、用户输入、变速箱和离合器等。

## 2　AMT 系统模型的实现

如 1.4 节所述，AMT 模型是通过量子平台实现的。量子平台 QP 是一个轻量级、开源状态机(UML 状态图)引擎。量子平台主要为实时嵌入式系统设计，使软件开发人员能够使用状态机快速构建结构合理，事件驱动的应用程序[7]。量子平台框架中使用 RTC 原则，从根本上杜绝了死锁状态的产生[7]。量子平台由可剥夺的实时内核 QK 或协作式内核 Vanilla、事件驱动框架 QF、层次状态机处理器 QEP 和实时代码追踪器 QS 组成。量子平台系列有 QP/C、QP/C++和 QP-nano 3 种架构。在嵌入式中，支持 C 语言的编译器容易获得，因此一般采用 QP/C

框架。QP/C 框架的简单结构如图 9 所示[7]。

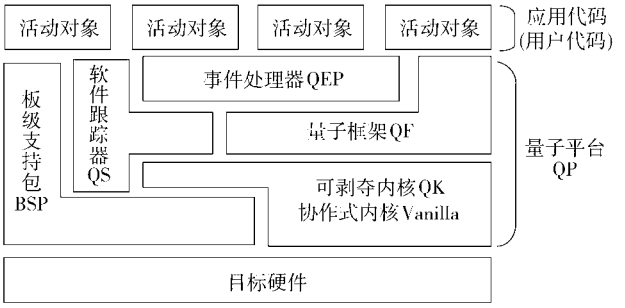



图 9 量子平台的层次结构

Fig. 9 Hierarchy of quantum platform

## 2.1 从 UML 模型到活动对象

从图 9 可以看出,QP 平台运行的基本单元为活动对象(active object)。活动对象是用户编写的可以在量子平台上直接运行的类。活动对象之间相互独立,可以通过 QF 框架提供的消息机制相互通信[7]。

量子平台系统定义和实现了一些基本类,包括 QActive、QEvent 和 QTimeEvt 等。QActive 在量子框架中是一个封装的线程,并包含一个 QFsm 状态机;QEvent 是定义事件的一个基础类,用来产生一个事件信号;QTimeEvt 则是一个时间事件的基础类。

量子平台对 UML 类元素的支持是通过继承机制实现的。UML 类通过继承 QActive 类可以编程为一个活动对象,从而能在量子平台上运行。UML 类还可以通过继承 QEvent 类获得量子平台事件驱动支持,通过继承 QTimeEvt 获得量子平台时间事件的支持。

量子平台只提供了对状态机和类的支持,不支持 UML 中的其他元素。UML 模型的动态行为必须转换成在逻辑上等价的层次状态机,才可以在量子平台上运行。另外,类与类之间的关联可以通过量子平台提供的消息机制实现。

以发动机(Engine)为例(图 10),简述 UML 模型的 Engine 类在量子平台下的实现。Engine 类通过继承 QActive 成为活动对象。Engine 类通过间接继承 QHsm 类获得量子平台对状态机的支持,其动态行为需要使用状态机重新描述。Engine 类通过继承 QEvent 类获得量子平台对消息机制的支持,和其他类的关联通过消息机制实现。

## 2.2 UML 模型在量子平台下实现

StarUML 作为一款免费开源的 CASE 工具,具有自动生成代码的功能,能够生成 UML 静态模型的 C++代码。图 11 描述了从 UML 模型实现的基本步骤。UML 模型通过 StarUML 工具生成 C++代码。UML 模型中动态行为可以通过状态图重新描述。

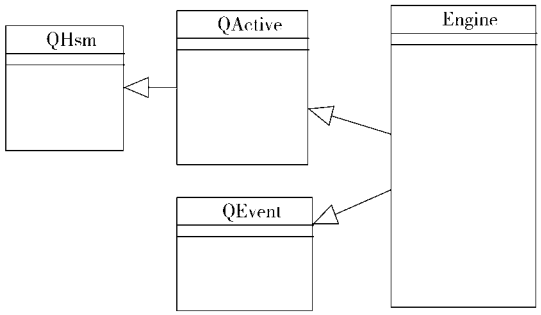



图 10 UML 类转化成活动对象

Fig. 10 UML class transformed into active object

在本文中,状态图的建模全部是手工完成的。文献[8]表明,通过借鉴 BK 算法核心思想,可以实现 UML 序列图合成符合 UML 标准的状态图[8],所以这部分工作将来能够实现自动化。活动对象可以在 QM 工具下完成建模,并通过 QM 工具自动生成代码。

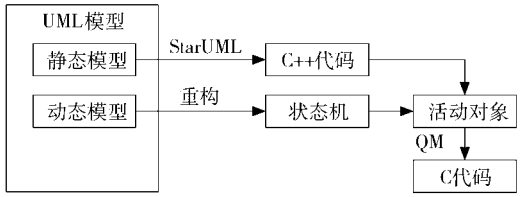



图 11 UML 模型的实现步骤

Fig. 11 Implementation steps of UML model

以 UML 类发动机为例,简述将 UML 语言中的类转换为量子平台下的活动对象。Engine 类在 StarUML 工具的自动生成的代码为

```
class Engine {
public:
void get_seed();
void set_speed();
 ⋮
private:
int speed;
int torque;};
```

如图 12 所示,在 QM 工具下,可以创建活动对象 Engine, Engine 类的属性和方法可以手动添加。

Engine 类的动态行为可以用状态图重新描述。Engine 状态包括初始状态、正常状态、怠速状态和过渡状态。正常状态下,驾驶员通过油门踏板直接控制油门;怠速状态下油门踏板完全松开,发动机怠速;在过渡状态下通常出现在换挡过程中,此时油门完全是由 TCU 程序控制的。

活动对象 Engine 可以使用 QHsm 基础类提供的 QHsm_ctor 方法,将自己链接于层次状态机的初始状态 Engine_initial 上。代码如下

```
QHsm * Engine_ctor(void) {
```

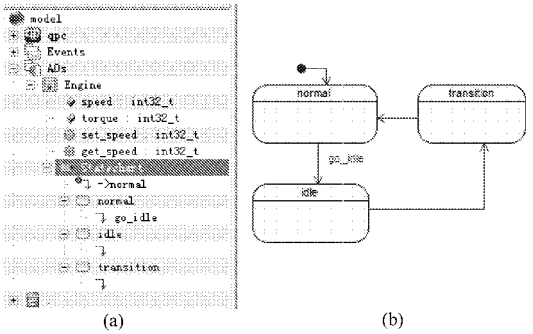
```
Engine * me;
    me = &l_engine;
    QHsm_ctor(&me->super,(QStateHandler)&Engine_initial);  /* superclass' ctor */
    return (QHsm *)me;
}
```

经过以上步骤，由 UML 模型换成 Engine、Clutch 和 Gear 等几个包含状态机的活动对象，详见图 11。多个活动对象只能使用 QF 提供消息机制的共享数据和资源[7]。另外，错误处理和安全策略等也可以通过活动对象来实现。

### 2.3　基于 QM 工具的代码自动生成

文献[13]表明，可以通过以 XML 语言为中介的 XMI + XSLT 技术和 QF 模板，实现活动对象的自动代码生成。实际上，免费的 QM 工具能够实现基于活动对象的代码自动生成。QM(QP modeler)是可以和 QP 状态机框架配合使用的状态图建模工具。QM 提供直观的图形环境并且能够自动生成非常紧凑的 C/C++代码，可 100% 追溯原始设计。QM 的使用，使得用户编写的代码量降到最少，加快编码的速度并提高代码质量。

图 12 以 Engine 为例，描述了一个通过 QM 建立的活动对象模型。图 12a 是活动对象 Engine 的属性和方法，这些可以手动添加，图 12b 是描述其动态行为的状态图。




(a)　　(b)

图 12　QM 工具的界面环境

Fig. 12　Interface environment of QM tools

通过 QM 自动生成的代码为

```
typedef struct EngineTag {
/* protected: */
    QActive super;
/* public: */
    int32_t speed;
    int32_t torque;
} Engine;
/* public: */
int32_t Engine_set_speed(Engine * me);
int32_t Engine_get_speed(Engine * me);
/* protected: */
QState Engine_initial(Engine * me, QEvent const * e);
QState Engine_normal(Engine * me, QEvent const * e);
QState Engine_idle(Engine * me, QEvent const * e);
QState Engine_transition(Engine * me, QEvent const * e);
⋮
```

## 3　系统测试

最终代码在 IAR 环境下编译通过，并下载到以 STM32F103 为主控芯片的 AMT 控制器上。系统代码编译后占用只读存储区 5 kB，读写存取区 620 B。用户代码编译后，占用只读存储区 20 kB，读写存储区 10 kB。

目标代码需要实车测试，以便发现代码缺陷，及时修正。QP 提供了追踪工具 QS，可以在占有最少内存和 CPU 运算时间的情况下，提供程序运行的关键数据[7]。QM 工具的使用，可以使程序员快速重构代码。QS 和 QM 联合使用，缩短了测试周期，提高了编码效率。

由于作者在前后台系统、基于 FreeRTOS 嵌入式操作系统和量子平台 QP 上进行过同一硬件系统的 AMT 控制系统测试，忽略对平台的掌握和熟悉时间，其大致的编程调试时间、程序量、易读性及维护性的对比见表 1。

**表 1　前后台系统、Free RTOS 和量子平台的比较**

**Tab. 1　Comparison of foreground and background system, Free RTOS and quantum platform**

| 项目 | 前后台 | FreeRTOS | QP |
|---|---|---|---|
| 只读/kB | 47.1 | 28.8 | 25.0 |
| 读写/kB | 21.3 | 19.2 | 10.6 |
| 开发时间/d | 180 | 120 | 90 |
| 可读性 | 差 | 容易 | 容易 |
| 可维护性 | 困难 | 一般 | 容易 |
| 可扩充性 | 困难 | 一般 | 容易 |

通过测试发现，UML 模型需求和最终设计有 90% 以上的一致性。量子平台避免了嵌入式实时操作系统经常发生的死锁现象，有着更强的稳定性和健壮性。系统的实时性和可靠性满足设计要求。

## 4　结束语

此方法将 UML 模型和量子平台结合起来，初步实现了 AMT 控制系统从 UML 模型到目标代码的自

动转换,并提供了一系列方法,涵盖需求、系统分析、实现和测试各个阶段。UML 的使用,让设计者能够在设计初期构建系统的精确蓝图,减少开发的盲目性,项目开发具有完整且规范的文档,便于项目管理。QM 工具的使用,显著提高了编码速度,编码工作基本实现是图形化的,仅仅少量代码需要手工编写,大部分代码可以通过工具自动生成。QS 工具提供了友好的调试接口,大大缩短了系统测试的时间。测试证明,此方法保证需求和最终设计的一致性,加快了开发速度,同时提高了代码质量。通过 StarUML 工具建立的 UML 模型,为后期的开发维护提供了可靠的文档,便于工程的维护、后期的扩展和优化。图形化编程,将开发者从繁琐的,容易出错的手工编码中解放出来,更专注于产品的功能和可靠性。

## 参 考 文 献


1 冯能莲, 李克强, 连小珉, 等. 液力机械传动车辆自动换挡控制系统[J]. 农业机械学报, 2004, 35(1): 8 ~ 12.
Feng Nenglian, Li Keqiang, Lian Xiaomin, et al. Study on automatic-shift control system of a vehicle equipped with hydro-mechanical transmission[J]. Transactions of the Chinese Society for Agricultural Machinery, 2004, 35(1): 8 ~ 12. (in Chinese)

2 綦声波, 张承瑞, 罗映. 基于 SOPC 和量子框架的电控机械式变速器电控系统[J]. 农业机械学报, 2011, 42(10): 13 ~ 19.
Qi Shengbo, Zhang Chengrui, Luo Ying. AMT electronic control system based on SOPC and quantum framework[J]. Transactions of the Chinese Society for Agricultural Machinery, 2011, 42(10): 13 ~ 19. (in Chinese)

3 李洪斌, 张承瑞. 基于量子框架的开放式汽车电控系统体系结构[J]. 吉林大学学报, 2006, 36(2): 166 ~ 171.
Li Hongbin, Zhang Chengrui. Open architecture of automotive E/E control system based on quantum frame[J]. Journal of Jilin University, 2006, 36(2):166 ~ 171. (in Chinese)

4 黄咏, 庄诚, 高东杰. 输送系统的 UML 建模、优化与实现[J]. 控制工程, 2006, 13(1): 55 ~ 59.
Huang Yong,Zhuang Cheng,Gao Dongjie. Modeling, optimizing and implementing of transportation system with UML[J]. Control Engineering of China, 2006, 13(1):55 ~ 59. (in Chinese)

5 徐宝文, 周毓明, 卢红敏. UML 与软件建模[M]. 北京:清华大学出版社,2006.

6 Jim A, Ila N. UML 2 and the unified process: practical object-oriented analysis and design [M]. New Jersey: Pearson Education, 2005.

7 Miro S. Practical UML statecharts in C/C + +: event-driven programming for embedded systems [M]. 2nd ed. Boston: Newnes, 2008.

8 褚华, 李青山, 陈平,等. 一种基于 UML 序列图的状态图合成方法[J]. 系统工程和电子技术,2005,27(3):524 ~ 528.
Chu Hua, Li Qingshan, Chen Ping, et al. Approach of statechart synthesis from UML sequence diagrams[J]. Systems Engineering and Electronics, 2005, 27(3): 524 ~ 528. (in Chinese)

9 Qi Shengbo,Zhang Chengrui, Qi Zhenjun, et al. SOPC technology applied on automatic mechanical transmission control unit design [C]// Proceedings of the 8th World Congress on Intelligent Control and Automation. China, 2010: 5 371 ~ 5 375.

10 李晓军, 张承瑞, 杨泉. 软件总线在 AMT 电控单元中的应用研究[J]. 汽车工程, 2009(4): 304 ~ 307.
Li Xiaojun, Zhang Chengrui, Yang Quan. A study on the application of software bus to AMT ECU[J]. Automotive Engineering, 2009(4): 304 ~ 307. (in Chinese)

11 张俊智, 薛俊亮, 潘凯. 混合动力系统控制软件的开发[J]. 机械工程学报, 2009, 45(5): 115 ~ 120.
Zhang Junzhi, Xue Junliang, Pan Kai. Development of hybrid powertrain control software[J]. Chinese Journal of Mechanical Engineering, 2009, 45(5): 115 ~ 120. (in Chinese)

12 赵家强. 基于 UML 的 AMT 系统建模和实现[D]. 合肥: 合肥工业大学,2007.

13 朱伟. 基于量子框架 AMT 系统的代码自动生成技术研究[D]. 济南: 山东大学,2008.