第25卷 第3期
2010年6月

天津科技大学学报
Journal of Tianjin University of Science & Technology

Vol.25 No. 3
Jun. 2010

基于MSP430单片机的软件代码快速开发

刘玉良[1]，贾子申[1]，刘丽辉[2]，杨伟明[1]
(1. 天津科技大学电子信息与自动化学院，天津 300222；2. 天津医科大学总医院，天津 300052)

摘 要：为实现高效的代码编写和方便的系统维护，在前期研究工作的基础上，利用 Matlab 7.x 的 Stateflow 工具箱，实现了基于 MSP430 单片机的嵌入式系统软件开发. 首先，对嵌入式系统进行基于事件驱动的建模与仿真. 然后，在状态图代码生成器(Stateflow Coder)中把状态图(State chart)翻译成高效的C语言代码. 最后，用基于MSP430单片机的目标系统底层驱动函数与翻译的 C 语言空函数进行简单替换，实现了嵌入式系统软件代码的快速生成. 对比实验表明，用本文方法生成的软件代码，在嵌入式目标系统中运行状况良好，其易维护性和执行效率等方面均优于手工编写的代码.

关键词：嵌入式系统；Matlab；状态图
中图分类号：R333.8　　**文献标志码**：A　　**文章编号**：1672-6510(2010)03-0061-05

Rapid Development of Software Code Based on MSP430

LIU Yu-liang[1], JIA Zi-shen[1], LIU Li-hui[2], YANG Wei-ming[1]
(1. College of Electronic Information and Automation, Tianjin University of Science & Technology, Tianjin 300222, China;
2. Tianjin Medical University General Hospital, Tianjin 300052, China)

Abstract: In order to get high efficiency of code design and convenience of software debugging, based on the analysis of relevant research before, the development of the embedded system software of MSP430 was realized. Firstly, using Matlab/StateflowToolbox to model and simulate the complex event drive in the embedded system was introduced. Then using coder the stateflow was translated into high efficiency C code. By making some simple replacement to the blank C function generated with the low-layer function of the target system based MSP430, the code rapid generation of the embedded software was finished. Experiment show that the automatically code generation works well in the embedded target system. It has good performance and high code efficiency than those code generated by hand.

Keywords: embedded system; Matlab; state chart

随着嵌入式系统复杂性的提高，嵌入式软件的开发与后期维护工作也越来越复杂. 代码自动生成工具在嵌入式系统开发中的应用为解决此难题带来了希望[1-5]. 很多基于统一建模语言 UML(Unified Modeling Language)及其开发理念 MDA(Model-Driven Architecture)的设计工具产品相继问世，如 Telelogic 公司的 UML2.0 工具 Tau Developer，Mirabilis 公司推出了基于 UML 的 System C Modeler，Metrowerks 公司针对 Linux 操作系统的嵌入式系统构建工具 PCS(Platform Creation Suite)等，Mathworks 公司也在 Matlab 中提供了 Stateflow 工具箱，以支持基于状态图的建模与仿真[6]. 嵌入式系统代码自动生成在很大程度上提高了代码的开发和后期维护的效率.

本文在前期基于 MSC1212 的嵌入式软件代码快速生成研究工作[7]的基础上，尝试把基于模型驱动的软件设计方法应用于基于 MSP430 系列单片机的系统软件开发过程. 利用 Matlab 的 Simulink/Stateflow 工具箱，实现嵌入式 C 语言代码的自动生成，并与手工编写代码的代码量和运行效率进行了比较.

收稿日期：2009-08-31；**修回日期**：2009-12-13
基金项目：天津市高等学校科技发展基金资助项目(20080808)
作者简介：刘玉良(1972—)，男，河北迁安人，讲师，博士，ylliu@tust.edu.cn.

1　开发方法

系统软件代码自动生成主要包括三个过程：建模与仿真、代码生成和平台移植.

1.1　建模与仿真

Simulink 模块和状态图工具箱 Stateflow Toolbox 是 Matlab 常用的系统建模工具[8]. 为了描述嵌入式系统的逻辑行为，首先需要分析对象在不同时刻的行为特点，然后在 Simulink 环境中用多个状态图模块建立嵌入式系统软件模型.

采用 Simulink/Stateflow 建立嵌入式系统软件模型之后，首先进行语法检查，并在 Model explorer 中为状态图中使用的数据指定数据类型，为事件确定属性.

仿真过程中可打开状态图的调试窗口，通过选择 Event Broadcast 中断和点击 Step，可以使 alarm 模块的内部逻辑一步一步执行，分析状态图的变迁过程，查找错误. 每一次外部事件发生时都会重复这个过程.

调试窗口的 Coverage 是模型代码覆盖率，用来发现模型中的冗余代码. 当所有状态都执行过以后，覆盖率应为 100%. 系统模型仿真通过之后，可以选择把系统模型翻译为某种程序语言，通常选择C语言.

1.2　代码生成

由于现在的许多单片机开发环境(IDE)均可以使用 C 语言作为编程语言，所以选择 ANSI-C 作为模型的生成语言具有较好的通用性和可移植性. 将状态图翻译为 C 语言的工作由 Stateflow Coder 来完成，它是 Stateflow Toolbox 的一个辅助工具.

由 Stateflow Coder 生成的嵌入式 C 语言代码，在结构上是非常规范和紧凑的，使用了最小的 RAM 区，但对于一个实时嵌入式应用软件来讲，它还缺少一些关键性的框架代码，如：多个事件的调度代码和中断例行程序代码. 针对特定的单片机系统，还需要注意编译器方面的要求.

1.3　平台移植

完成了嵌入式系统的建模、仿真，产生了规范的 C 代码以后，接着要将这些代码移植到特定的单片机芯片中，在相应的 IDE 开发环境下，实时验证程序的正确性和稳定性.

因为由 Stateflow Coder 生成的程序代码是标准的 C 语言代码，适用于任何一款提供 C 语言编译器的单片机，所以只要适当修改一些底层代码，就可以顺利地移植到任何一款单片机上，在一定程度上解决了代码跨平台移植问题.

把手工编写的嵌入式 C 语言代码在 IAR 公司的集成开发环境下编译通过后，下载到 MSP430F413 单片机中运行，实时监测系统的功能和稳定性.

2　开发实例

基于德州仪器公司 MSP430 单片机，进行代码自动生成的测试，图 1 是嵌入式系统的软件模型.

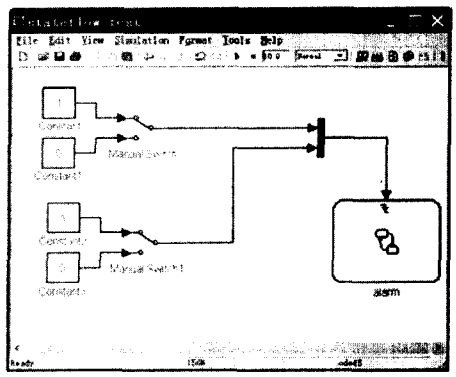

图 1　嵌入式系统软件模型

Fig.1　Software model of the embedded system

状态图可以很好地描述串行事件的执行过程；而对于并行事件，状态图则使用'伪'并发机制来模拟. 对于实时性不是非常严格的单处理器嵌入式系统，该机制是一个可行的解决方案.

在图 1 的模型中，从 Constant 到 Constant3 是 4 个常数模块，用于产生状态图模块 alarm 所需要的 2 个外部事件. 当发现有满足条件的事件后，激活 alarm 状态图，使其运行一次. 在基于 MSP430 单片机的嵌入式系统中，alarm 状态图模块所描述的逻辑就是单片机内部的软件功能. 两个外部事件可以认为是由中断例行函数处理过的外部信号输入标识. 中断函数每执行一次，说明有一次外部信号输入，就标识一次，然后主程序将响应一次这个标识，执行一次动作. 如果两个事件同时发生，系统按设计时确定的顺序分别执行. 图 1 中由 Constant 和 Constant1 构成的事件先被执行. 需要特别指出的是，中断函数没有包含在图 1 中，在最终产生的代码中也不存在，需自行编写. 可以通过 alarm 模块窗口查看状态图的内部逻辑描述. 对于图 1 所示模型，存在两个线程 thread1 和 thread2，并发执行. 在实际运行时，每个线程都会对事件进行检查.

语法检查工作完成后，需要在 Model explorer 中

为状态图使用的数据指定数据类型，为事件确定属性. 因为模型中没有中断函数，所以用图 1 中的两个 Manual Switch 模块进行模拟输入. 只要双击 Manual Switch 模块，就可以拨动开关. 通过设置 Configuration Parameters 选项卡，完成仿真前的准备工作. Stop time 项应设为 inf；Type 项应设为 Fixed-step；Solver 项应设为 discrete(no continuous states)，其他保持默认.

完成上述设置后，开始仿真. 仿真过程中保持 alarm 模块为打开状态. 双击 Manual Switch 模块，产生外部事件激活 alarm 状态图，进入主程序.

在仿真过程中，还可以打开状态图的调试窗口，如图 2 所示. 通过单步执行，可以使 alarm 模块的内部逻辑单步执行，分析状态图的变迁过程，查找错误. 每一次外部事件发生都会重复这个过程.

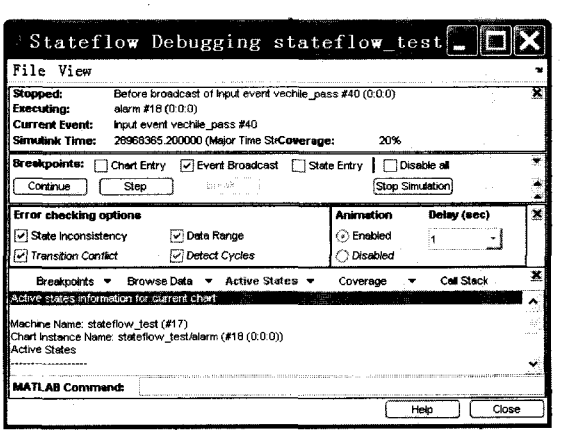

图 2　状态图调试窗口

Fig.2　Stateflow debugging window

当所有状态都执行过以后，调试窗口中的模型代码覆盖率应为 100%. 系统模型仿真通过后，选择把系统模型翻译为 C 语言.

选择 ANSI-C 作为模型的生成语言具有较好的通用性和可移植性，因为现在的许多单片机开发环境(IDE)均可以使用 C 语言作为编程语言. 状态图翻译为 C 语言的工作由 Stateflow Coder 来完成，它是 Stateflow Toolbox 的一个辅助工具.

此外，还可以打开自定义目标编辑窗口，进行自定义设置. 在 alarm 模块的界面中，选择 Add 下拉菜单中的 Target 选项按钮，打开图 3 所示窗口.

将 Target Name 一栏改为自定义的名称，例如：my_prog. 在 Target Language 一栏中选择 Generate code only(non-incremental)，然后点击 Generate 按钮，Stateflow Coder 打开图 4 所示代码生成窗口.

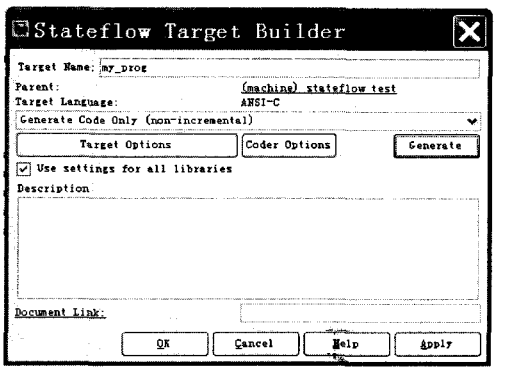

图 3　自定义目标编辑窗口

Fig.3　Stateflow target builder window

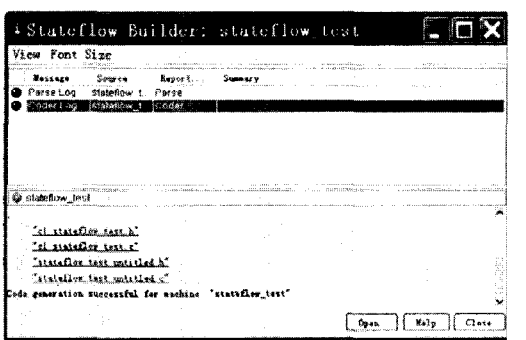

图 4　代码自动生成窗口

Fig.4　Stateflow builder window

分别点击窗口下方的 4 个文件，包括两个“.h”头文件和两个“.c”程序文件，在 Matlab 的 Editor 中打开. 观察可以发现，在 c1_stateflow_test.c 文件中包含了 alarm 状态图的完整代码，在其文件的头部包含了两个“.h”文件：

```
/* include files */
#include "stateflow_test_untitled.h"
#include "c1_stateflow_test.h"
```

分别定义了状态图所使用的数据和事件.

在 c1_stateflow_test.c 文件中还包含了对于事件和状态的定义.

对于变量的定义则在 c1_stateflow_test.h 文件中，而且采用了结构描述方法.

对于两个外部事件的使用，Stateflow Coder 产生了两个函数：void broadcast_c1_stateflow_test_vechile_pass(void)和 void broadcast_c1_stateflow_test_vechile_pass_next (void)，它们的声明在 c1_stateflow_test.h 文件中，它们定义在 c1_stateflow_test.c 文件最后：

```
void broadcast_c1_stateflow_test_vechile_pass
(void)
{
uint8_T b_previousEvent;
```

```
...
_sfEvent_stateflow_test_ = b_previousEvent;}
void broadcast_c1_stateflow_test_vechile_pass_
next(void)
{
uint8_T b_previousEvent;
...
_sfEvent_stateflow_test_ = b_previousEvent;}
```

而真正的主逻辑函数，即由 alarm 状态图产生的代码，被置于 void c1_stateflow_test(void)函数中，定义在 c1_stateflow_test.c 文件的中部：

```
/* Function Definitions */
void c1_stateflow_test(void)
{
 uint8_T b_previousEvent;
 ... }
```

外部事件函数：

```
void broadcast_c1_stateflow_test_vechile_pass
(void)
void broadcast_c1_stateflow_test_vechile_ pass_
next(void)
```

通过调用c1_stateflow_test()函数，实现事件驱动：

```
void broadcast_c1_stateflow_test_vechile_pass
(void)
{
 uint8_T b_previousEvent;
 b_previousEvent = _sfEvent_stateflow_test_;
 _sfEvent_stateflow_test_ = event_vechile_pass;
 c1_stateflow_test();
_sfEvent_stateflow_test_ = b_previousEvent;}
void broadcast_c1_stateflow_test_vechile_pass_
next(void)
{
uint8_T b_previousEvent;
b_previousEvent = _sfEvent_stateflow_test_;
_sfEvent_stateflow_test_ = event_vechile_pass_
next;
c1_stateflow_test();
_sfEvent_stateflow_test_ = b_previousEvent;}
```

用 MSP430 的驱动函数替换 Stateflow Coder 生成的空函数，代码自动生成工作完成.

建模、仿真并产生了规范的 C 代码以后，就要将这些代码移植到特定的单片机芯片中，在相应的 IDE 开发环境下(IAR 公司)，实时验证程序的正确性和稳定性. 为了能有效地使用 Stateflow Coder 产生的 C 代码，使用了如下的程序模板：

```
//******通用程序模板******
//变量声明
...
//常量定义
...
MAIN
{
initialize_MCU()//底层函数，手动编写
initialize_RAM()//底层函数，手动编写
sfevent = CALL_EVENT
loop
sleep()//程序休眠指令
SCH() //调用 SCH 图的代码
goto loop}
//*******调度程序******
SCH()
{
Deal_Event_A();
Deal_Event_B();
... }
mian_logic()
{ ... }
//******中断服务程序******
Interrupt TASK_x
{
TASK_x();}
...
```

用 Stateflow Coder 对状态图模块进行翻译后，分别得到两个*.c 源文件和两个*.h 头文件. 对应上述程序模板，将 4 个文件中的程序代码通过人工方法装配到程序模板中，函数的对应关系是：

(1) 变量声明和常量定义，分别对应/* Named Constants */和/* Type Definitions */下的内容. 但是，为了方便在汇编语言源程序中使用 C 语言的变量，需要把 Stateflow 生成的 C 代码中的结构体变量改成独立的多个变量，分别定义.

(2) initialize_MCU()和 initialize_RAM()采用汇编代码实现.

(3) sleep()函数是针对特定单片机系统的一个函数，没有通用性. 不同的单片机休眠的方式不同，建议用汇编代码编写.

(4) 在 SCH()函数中，事件响应函数 Deal_Event_A()和 Deal_Event_B()用 void broadcast_c1_stateflow_test_vechile_pass(void)和 void broadcast_c1_stateflow_test_vechile_pass_next(void)替换，如有多个事件，则依次列出所有的事件响应函数.

(5) mian_logic()函数用状态机产生的 void c1_stateflow_test(void)函数替换.

(6) 中断服务程序是需要人工编写的，可以用汇

编语言或 C 语言.

因为由 Stateflow Coder 生成的程序代码是标准的 C 语言代码,适用于任何一款提供 C 语言编译器的单片机,所以只要适当修改一些底层代码,就可以顺利地移植到任何一款单片机上,在一定程度上解决了代码跨平台移植问题.

把手工装配好的嵌入式 C 语言代码在开发环境下编译通过后,下载到 MSP430F413 单片机中运行,可以实时检查系统的功能和稳定性.

3 结果与讨论

为了比较本文方法与手工编写代码的效率,进行了简单的对比实验. 用普通的开发流程图方式开发楼宇监控系统软件,控制逻辑需要反复修改,耗时半个月,每天投入 6 h,总共 80 h. 用本文方法进行开发,虽然还是需要修改,但是由于图形编程逻辑直观,一目了然,所以只用18 h 就可以完成,对比结果见表 1.

表 1　与手工代码开发的比较结果

Tab.1　Comparison with coding by hand

开发方法	开发周期/h	移植性	代码稳定性	可维护性
自动生成	18	好	高	高
手工编写	80	差	低	低

通过对比可见,与手工编写程序相比,自动生成代码方法开发嵌入式程序的开发周期大幅缩短,软件的可移植性、可维护性和稳定性都大幅提高.

4 结　语

本文利用 Simulink/Stateflow 工具箱,实现了在 MSP430F413 硬件平台上的嵌入式系统软件代码自动生成方法. 由状态图模型自动产生的代码与手工编写的代码相比,具有结构紧凑、形式规范、易于维护等特点. 把基于模型的软件开发方法应用于低端嵌入式系统的开发,可以加快开发进度,节约时间成本,降低软件工程师的开发和维护强度. 虽然还不能做到 100%代码自动生成,但是其已对嵌入式系统应用提供了较大的便利.

参考文献:

[1] 刘然,陈英,赵小林. 基于 UML 的 CASE 平台的代码自动生成[J]. 北京理工大学学报,2002,22(2):196-200.

[2] Selic B. Using UML for modeling complex real-time systems[J]. Lecture Notes in Computer Science, 1998(1474):250-272.

[3] Basso F P,Oliveira T C,Becker L B. Using the FOMDA approach to support object-oriented real-time systems development[C]//Proceeding of 9th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. Piscataway:IEEE,2006:374-381.

[4] Khan M U,Geihs K,Gutbrodt F,et al. Model-driven development of real-time systems with UML 2. 0 and C [C]// Proceedings of 13th IEEE International Conference on Engineering of Computer Based Systems. Piscataway:IEEE,2006:33-42.

[5] Lu Shourong,Halang Wolfgang A,Zhang Lichen. A component-based UML profile to model embedded real-time systems designed by the MDA approach[C]// Proceedings of 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. Piscataway:IEEE,2005:563-566.

[6] Bastian Florentz,Martin Mutz. Avoiding unpredicted behaviour of large scale embedded systems by design and application of modeling rules[C]// Proceedings of 2004 1st International Workshop on Model, Design and Validation,SIVOES-MoDeVa 2004. Piscataway:IEEE, 2004:59-66.

[7] 刘玉良,李刚,康凯. 基于 MATLAB 的嵌入式系统软件开发[J]. 天津大学学报,2008,41(5):593-596.

[8] 邹晖,陈万春. Stateflow 在巡航导弹仿真中的应用[J]. 系统仿真学报,2004,16(8):1854-1856.