第 33 卷 第 6 期 计 算 机 工 程 2007 年 3 月
Vol.33 No.6 Computer Engineering March 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)06—0280—03 文献标识码：A 中图分类号：TP311

智能家电嵌入式软件的源码构件设计方法

邓 勇，桑 楠，罗克露，高 卓
(电子科技大学嵌入式实验室，成都 610054)

摘 要：通过对家电控制器常用 MCU 体系结构、程序设计语言、家电功能及外设驱动源码研究，提出了一种形式化的嵌入式软件的源码构件和程序代码的设计方法。介绍了源码构件层次体系和逻辑模式；定义了源码构件运算符；给出了构件及应用代码的形式化生成；并以重用因子 K 为标准，辅助衡量源码构件的代码片段划分和设计的合理性。该方法具有很好的实用性、可扩展性和通用性。

关键词：智能家电；源码构件；代码片段；构件描述块；重用因子

Design Method of Code Component for Embedded Software in Smart Appliances

DENG Yong, SANG Nan, LUO Kelu, GAO Zhuo
(Embedded Lab, University of Electronic Science and Technology of China, Chengdu 610054)

【Abstract】 The paper puts forward a formal method for designing code component and program of embedded software, through the research of MCU architecture, programming language, appliance functions and drivers of peripheral equipment. This paper introduces the layer structure and logic model of a code component, defines the operator among the code components, and gives the formal generation of complex component or application code. Furthermore, the reuse factor K is defined to give support to weigh the rationality of detaching and designing the code segment of a code component. The method is featured by practicability, expandability and universality.

【Key words】 Smart appliances; Code component; Code segment; Component description block; Reuse factor

在嵌入式应用领域，由于嵌入式平台的多样性及其内存消耗、实时特性、可靠性和稳定性等重要的非功能性因素，国内外研究机构专门推出了一系列嵌入式构件模型，包括比利时IWT协会赞助的SEESCOA项目的CCOM模型[3,5]、飞利浦公司用于消费电子的Koala构件模型[3,5]和ABB等公司用于现场设备技术的Pecos构件模型[3,5]等。这些模型的共同特点是：基于源码级的构件复用，并有相应的CASE工具支持，其缺点是不支持无嵌入式操作系统的智能家电嵌入式软件的开发。

针对智能家电嵌入式软件开发所面临的常见问题是：软件工程师编写的程序代码中有大量的可重用片段，他们的离职常常导致技术无法继承，本文基于智能家电控制器常用MCU体系结构、程序设计语言、常用家电功能及外设驱动源码研究，提出了一种适用于智能家电嵌入式软件构件化开发方法[1,2,4]，介绍了该方法采用的构件规范及构件运算模式，说明了构件及应用代码的形式化生成过程，并且给出评价源码构件的代码片段划分和设计合理性的方法。

1 源码构件设计方法

1.1 源码构件规范

定义 1 设计的源码构件必须符合以下规范：

源码构件 ::= { 复合构件 | 原子构件 }

复合构件 ::= {复合代码片段 + 构件描述块}

原子构件 ::= { 原子代码片段 + 构件描述块 }

复合代码片段 ::= {∑(代码片段 + 插入点) }

插入点 ::= { BEGIN + 复合代码片段 | 原子代码片段 + END}

原子代码片段 ::= { 简单赋值语句 | 算术逻辑运算语句 | 内部跳转语句 | 条件语句}

代码片段 ::= { 简单赋值语句 | 算术逻辑运算语句 | 跳转语句 | 条件语句 | 空语句 }

跳转语句 ::= { 内部跳转语句 | 外部跳转语句 }

内部跳转语句 ::= { 语句的跳转目标的标号在代码片段内部 }

外部跳转语句 ::= { 语句的跳转目标的标号在代码片段的外部 }

构件描述块 ::= { 源码构件属性描述信息集 }

按此定义，原子构件是源码构件的最小单位，不可再分割，但原子代码片段可以插入到其它非原子源码构件中；复合构件提供的插入点处，可以插入与该插入点功能描述相同的复合代码片段或原子代码片段，则实现了构件之间的嵌套插入；而源码构件则是原子构件和复合构件的总称。复合构件提供的插入点采用保留字 BEGIN 和 END 定义，与源码构件相关的所有属性信息均存在于构件描述块中，是源码构件存在的唯一标识。

1.2 源码构件的层次体系

根据上述规范，嵌入式软件的源码构件、复合构件、原子构件、代码片段和构件描述块的关系如图 1 所示。

图 2 体现了源码构件的嵌套关系，其中，叶子结点(实心)表示原子构件，中间结点和根结点(空心)表示复合构件，中间结点的分支数表示复合构件提供的插入点数，树的高度表示构件之间的嵌套度。由这些树形成的一片森林则表示了将由若干源码构件组装成的一个嵌入式软件的源程序。

基金项目：国家“863”计划基金资助项目(2005AA1Z2120)
作者简介：邓 勇(1982－)，男，硕士生，主研方向：嵌入式软件及构件技术；桑 楠，副教授；罗克露，教授；高 卓，硕士生
收稿日期：2006-05-30 **E-mail：**dy19820525@163.com

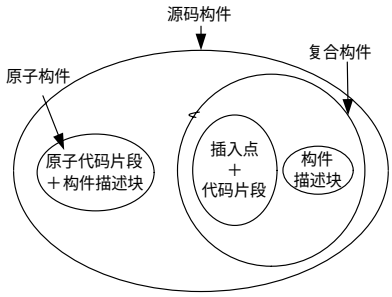

图 1 构件层次结构关系

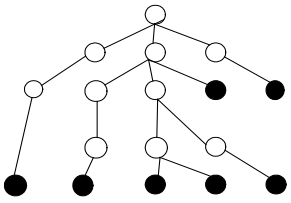

图 2 构件树型嵌套关系

2 构件的逻辑模式

源码构件在逻辑上可以抽象成一个二元组：C = (A，S)，其中 S 为源码构件的代码片段集合，当 C 为复合构件时，将其复合代码片段中包含的插入点依次记为第 1 个插入点，……，第 n 个插入点；A =(N，L，I，E)，表示构件描述块，完成代码片段的封装，唯一标识一个二元组 C，其中：

(1)N = (ID，Name，Type，Reuse)，为构件标识信息。

1)ID：构件 ID，每个源码构件具有唯一的构件 ID。

2)Name：构件名，规定构件名以字符或下划线开始，由字符、数字和下划线组成的字符串，构件名的字符数至少为 6，但不超过 128。

3)Type：构件类型，为一个整型变量，0：原子构件；>0：复合构件，其值表示该复合构件具有的插入点数。

4)Reuse: 构件重用度，即构件重复使用的累计次数，为一个整型变量，初始化为 0。

(2)L =(Mcu，Solve，KeyWord)，为构件检索信息。

1)Mcu：MCU 型号，代码片段所面向的 MCU 型号。

2)Solve：解决方案，代码片段所面向的具体应用。

3)KeyWord：功能关键字，源码构件完成功能的相关关键字。

(3)I = (In，Out)，为构件接口信息，给源码构件及应用代码的生成提供支持，方便源码构件及应用代码的形式化。

1)In：入接口，提示源码构件引用的场合、使用的参数(包括变量和常量)。

2)Out：出接口，描述构件对外界的请求。当构件完成某一功能时，可能需要其它构件的协作，出接口指明了这种对外关系。

(4)E =(Function，Parameter，Note)，为构件附加信息。

1)Function：功能描述信息。

2)Parameter：可配置参数，将源码构件内相关寄存器或存储器变量设置为可配置变量，构件组装时实现配置变量的实例化，达到源码构件的可配置性，解决源码构件组装时存在的资源的共享和互斥问题。

3)Note：构件注释信息，提供与构件库管理相关的信息的其它性质，这些性质是构件库管理所必需的信息，包括构件版本号、构件作者、入库时间、修改时间和修改影响。

3 源码构件的运算

3.1 构件运算符

定义 2 源码构件的运算符定义如下：

(1)‘.’运算：该运算符实现提取构件 C 的内容，如 C.S 表示源码构件 C 的代码片段 S，C.A.I.In 表示源码构件 C 的入接口。

(2)‘∪’运算：当构件C_1与C_2满足条件：

1)C_1.A.I.Out = C_2.A.I.In ；

2)C_1.A.L.Mcu = C_2.A.L.Mcu，C_1.A.L.Solve = C_2.A.L.Solve 时，C = $C_1 \cup C_2$ = (A, S)，其中，

3) C.S = C_1.S ∪C_2. S；

4)C.A.I.In = C_1.A.I.In, C.A.I.Out = C_2.A.I.Out；

5) C.A.N.Reuse = 0；

6) C.A.N.Type = C_1.A.N.Type + C_2.A.N.Type；

7)C.A.L.Mcu = C_1.A.L.Mcu，C.A.L.Solve = C_1.A.L.Solve。

因此，∪运算实现源码构件的代码片段顺序链接。

(3) ‘< >’运算：当构件C, C_1, …, C_n满足条件：

1) C.A.N.Type ≠ 0；

2) C.A.L.Mcu = C_1.A.L.Mcu = … = C_n.A.L.Mcu, C.A.L.Solve = C_1.A.L.Solve =… = C_n.A.L.Solve时，C< C_1, …, C_i, …C_n>，其中n =C.A.N.Type，表示在复合构件C的第i个插入点处插入源码构件C_i的代码片段，构件C_1,…，C_n的插入顺序与复合构件C的插入点顺序一一对应，则

①C.A.N.Reuse = 0；

②若C_i为空语句，则表示在源码构件C的第i个插入点处无 插入；

③C.A.N.Type = $\sum_{i=1}^{n} C_i$.A.N.Type，其中C_i为空时，C_i.A.N.Type = 1。

构件 C 的构件描述块中其它信息不变。并且，该运算符可以进行构件嵌套运算。

3.2 构件的生成

采用源码构件运算符即可实现源码构件的形式化生成，复合构件C = $C_1 \cup C_2$ < C_3, C_1,C_4< C_3,C_5 >>∪C_6 ，其生成过程为：

(1)源码构件C_3，C_5的代码片段分别依次插入到源码构件C_4的前 2 个插入点处。

(2)源码构件C_3、C_1和C_4的代码片段分别依次插入到源码构件C_2的第 1、2、4 个插入点处。

(3)源码构件C_1、C_2和C_6的代码片段顺序链接形成源码构件C的代码片段。

(4)设定源码构件C的构件描述块信息：比如C.A.I.In = C_1.A.I.In, C.A.I.Out = C_2.A.I.Out ;C.A.N.Reuse = 0 ;C.A.L.Mcu = C_1.A.L.Mcu ,C.A.L.Solve = C_1.A.L.Solve。

3.3 应用代码的生成

定义 3 源代码 := {头文件+∑(宏定义 | 寄存器/变量定义 | 源码构件) }

采用符合本设计方法的源码构件生成面向具体应用的源代码时，需要引入应用代码的头文件，以及对源码构件中所使用寄存器、变量或宏进行定义。采用 Src 表示源代码，H 表示头文件，D 表示宏定义集合，V 表示寄存器/变量定义集合，则定义 3 可以表示为

Src = H + ∑(D | V | C)

因此，使用源码构件C_1、C_2、C_3、C_4、C_5、C_6生成源代码时，若运算式为

Src = H + D ∪ V_1 ∪V_2 ∪V_3 ∪ C_1 ∪ C_2 ∪C_6< C_3,C_4< C_5 >>。

其形式化生成过程的解释与构件生成过程的解释相似。

当使用源码构件C_3、C_4、C_5、C_6生成复合构件C = C_6 < C_3,C_4< C_5 >>时，则上述应用代码的形式化生成过程可以表示为

Src = H + D ∪ V_1 ∪V_2 ∪V_3 ∪ C_1 ∪ C_2 ∪C

4 构件设计的合理性

定义 4 构件使用频度 M 代表源码构件 C 在实现构件生成或应用代码生成过程中的重复使用次数。

根据定义 4，则在$C_1 \cup C_2 < C_3, C_1, C_4 < C_3, C_5 >> \cup C_6$的构件生成运算过程中，构件$C_1, C_2, C_3, C_4, C_5, C_6$的使用频度分别为$M_1 = 2, M_2 = 1, M_3 = 2, M_4 = 1, M_5 = 1, M_6 = 1$。因此，在完成构件运算之后，根据构件使用频度修改构件描述块中C.A.N.Reuse的值，C.A.N.Reuse = C.A.N.Reuse + M。

定义 5 重用因子 K 表示构件在给定时间周期内期望的重复使用次数。其值为一预先给定的整数，根据需要可以进行调整。

根据对构件重用度 C.A.N.Reuse 的统计，事先为重用因子 K 取定合理值，用以作为衡量构件使用频率高低的标准。

定理 源码构件 C 代码片段划分和设计的合理性判定定理。

当 C.A.N.Reuse < K 时，该构件的使用频率低，其代码片段的划分和设计不合理，需要从构件库中删除或者进行修改，以增强该构件的重用性。

当 C.A.N.Reuse ≥ K 时，该构件具有较高的使用频率，其代码片段的划分和设计合理，构件的重用性比较强。

因此，根据定理，可以使用 C.A.N.Reuse 的值来度量源码构件 C 的代码片段划分和设计的合理性，方便对构件库的管理。

5 结束语

本文在介绍源码构件层次体系和逻辑模式的基础上，定义了源码构件运算符，用以实现构件及应用代码的形式化生成。该模式具有很好的实用性、可扩展性和通用性。

本研究尚存在一些需要进一步完善的问题，如构件重用度 Reuse 的计算问题，将在后续的研究中解决。

参考文献

1 Yen I L, Goluguri J, Bastani F, et al. A Component-based Approach for Embedded Software Development[C]//Proceedings of the 5th IEEE International Symposium on Object-oriented Real-time Distributed Computing. 2002.

2 Angelov C. A Software Framework Component-based Embedded Applications[C]//Proceedings of the 11th Asia-Pacific Software Engineering Conference. 2004.

3 Rob van Ommering, Frank van der Linden, Kramer J, et al. The Koala Component Model for Consumer Electronics Software[J]. IEEE Computer, 2000, 33(3): 78-85.

4 古幼鹏, 熊光泽, 桑 楠. 基于构件的嵌入式软件仿真开发环境模型研究[J]. 系统工程与电子技术, 2004, 10(26): 1495-1499.

5 房红征, 赵贵根, 柳克俊. 嵌入式组件模型研究[J]. 微计算机应用, 2005, 26(5): 521-524.

(上接第 272 页)

一段时间，都要向系统提交用户的位置信息，该位置信息经过编码后递交到交互管理器，交互管理器调度图形生成模式，在地图显示页面中更新用户当前位置。交互管理器隐藏在运行时框架中，模式与运行时框架之间的通信通过 Document Object Model(DOM)事件来实现。

最后以一个用例来观察导游系统中的多模式交互的过程。首先假设用户当前位置在南京的鼓楼广场，GPS 接收器首先向系统提交用户的位置是鼓楼广场，交互管理器接收到 GPS 模式输入，调用图片输出生成器，在地图浏览器上标出用户当前位置，并且生成“用户当前位于鼓楼广场”的上下文信息，并准备鼓楼广场的文本和图片资料以备查看。然后，用户语音输入“到中山陵的路径”，经语音识别和整合当前位置上下文后的结果提交给交互管理器。交互管理器将数据处理模块计算出来的结果传给图片和音频文件输出生成器，在浏览器上显示合适的路径和公交线路图，并给出语音提示。用户可以根据系统的提示选择路径和到达方式。

4 结语

本文提出的城市移动导游系统还处于原型设计阶段，重点介绍了系统中人机交互模型的设计。在移动计算环境下，人机交互成为其中的关键问题，我们提出的上下文参与的多模式交互设计思想，增加了人机交互的自然性，并且有效提高人机交互效率，是一种可行的人机交互模型。

参考文献

1 王 悦, 岳玮宁, 王 衡. 手持移动计算中的多模式交互[J]. 软件学报, 2005, 16(1): 29-36.

2 董士海, 王 坚, 戴国忠. 人机交互和多模式用户界面[M]. 北京: 科学出版社, 1999.

3 岳玮宁, 王 悦, 汪国平. 基于上下文感知的智能交互系统模型[J]. 计算机辅助设计与图形学学报, 2005, 17(1).

4 Yang J, Yang W, Denecke M. Smart Sight: A Tourist Assistant System[C]//Proc. of the 3rd International Symposium on Wearable Computer. 1999.

5 Nigel Da, Keith C, Keith M. Using and Determining Location in a Context-Sensitive Tour Guide[J]. IEEE Computer Journal, 2001, 34(8).

6 Cheverst K, Davies N, Mitchell K. Developing a Context-aware Electronic Tour Guide: Some Issues and Experiences[C]//Proceedings of the 2000 Conference on Human Factors in Computing Systems. 2000.

7 Schilit W N. A System Architecture for Context-aware Mobile Computing[D]. New York: Columbia University, 1995.

8 Jim B. Aspect Communications, Multimodal Architecture and In-terface[Z]. 2005-04-22. http://www.w3.org/TR/2005/WD-mmi-arch-20050422/.

9 James A L, Raman T V, Dave R. W3c Multimodal Interaction Framework[Z]. 2003-05-06. http://www.w3.org/TR/2003/NOTE-mmi-framework-20030506/.