关于实时设备驱动程序自动生成的研究*

范植华 邢国光

(中国科学院软件研究所 北京 100080) (北京系统工程研究所 北京 100101)

摘要 软件系统的自动生成是 90 年代软件工程的主流. 由于专用实时设备的高效管理与灵活加接是任何计算机专用实时系统不可或缺的基本功能,加之它们的类别和型号、数量和质量的与日俱增,如何克服专用外设驱动模块的易变性成为任何专用实时系统自动生成的关键. 本文介绍作者在专用实时设备驱动程序自动化研究中的初步成果,重点是他们自行研制的专用实时设备驱动程序半自动生成器 RTDG(real time device generator)的内部构造与实现技术,以及一个经过实践考验的应用实例. 此项实时系统自动生成研究过程中的阶段性成果已通过部委级鉴定,并获部委级三等奖.

关键词 专用实时母系统,实时应用子系统,实时外设驱动程序,菜单驱动与控制系统,多窗口编辑系统,实时设备数据结构生成系统.

软件系统的自动生成是 90 年代软件工程的主流.[1]作为实时系统自动生成过程中的一个层次产品,在计算机专用实时母系统[2]的总体框架中,借助于具体模型与抽象模型,数据模型与操作模型[3]而构筑起来的实时设备驱动程序半自动生成器 RTDG,有能力在实时应用工程师的配合之下,为专门的实时应用子系统交互式地生产紧凑而高效的专用外设驱动程序. 在外部,它通过自备的编程环境与来自 DEC 的程序开发与运行环境相结合,形成交互式的支持环境.[3]在内部,它划分为菜单驱动与控制系统、EPASCAL 多窗口编辑系统和实时设备数据结构生成系统等 3 个相对独立的部分,各司其职,构成完整的半自动生成器. 下面分述各部分的功能与结构.

1 菜单驱动与控制系统

RTDG 的命令经由菜单方式输入. 用户于光标所在处拍入驱动程序名和菜单选择项,菜单驱动程序负责把参数传送给控制程序. 后者分析这些参数,根据不同的参数值启动不同的子系统.

菜单管理子系统是利用 DCL(digital command language)[4]命令过程 VAX Ada FMS (form management system)[5]实现的.

---

* 作者范植华,1942 年生,研究员,主要研究领域为并行处理,实时处理. 邢国光,女,1955 年生,副研究员,主要研究领域实时数据处理,专用实时母系统.

本文通讯联系人:范植华,北京 100080,中国科学院软件研究所

本文 1994－08－15 收到,1994－12－31 定稿

(1) 建立菜单的方法

我们采用FMS来建立RTDG的屏幕格式及其格式属性.RTDG共有15幅不同格式的画面,均由下列操作建立起来:

① 执行命令"$FMS/EDIT 格式名"进入格式编辑状态,交互式地指定格式属性,建立实际格式,为菜单打印背景文件,建立域,设计显示属性和域属性,进行格式测试;

② 创建格式库,并将格式存入格式库;

③ 利用Ada语言编写应用程序,旨在根据用户需要调用FMS的格式驱动子程序.例如一个显示主菜单并请求用户输入实时设备驱动程序名和选择项的Ada应用程序,需要调用附加终端和附加工作区间,打开库通道,清屏并显示一个格式等事务的格式驱动子程序,其流程如图1所示;

④ 建立可执行的文件,然后,为运行该文件所需要的操作流程如图2所示.

(2) 管理菜单的方法

我们采用DCL命令过程来管理菜单.DCL命令过程是由DCL命令串组成的文件,它含有诸如数字和字符串变量、函数、程序控制语句、子过程、出错处理、I/O处理等等通常仅属于高级语言的特性,凭借这些特性,管理RTDG各子系统的任务即可完成.比如说,通过DCL命令过程中的文件操作功能,打开并读入一个外部文件mm.dat的程序段如下:

```
open/read inputfile mm.dat                打开文件 mm.dat,给它分配一个逻辑名 inputfile
read_loop:
read/end_of_file=endfile inputfile num    从与指定逻辑名 inputfile 相等价的文件 mm.dat 中读入一个记录并赋予
                                          num,遇到文件结束符,控制则转至 endfile
  first_char:='f$extract(0,1,num)         取首字符
  if first_char.eqs. "1" then bh:=1
    goto read_loop                        继续读
  endfile:
    close inputfile
    goto 'bh'                             转移到标号所在命令行
```

其中的mm.dat,由下面的Ada程序创建并填入选择项:

```
      ......
get(i,file,terminal,"option");取用户键入的选择项
create(output_data,out_file,"mm.dat");建议外部文件
put(output_data,file);写
close(output_data);关闭
      ......
```

简言之,如图3所示,依靠对同一外部文件的读写操作,我们能够将用户键入的选择项读入系统,分析后启动相应的子系统.

2 EPASCAL多窗口编辑系统

这是RTDG最基本的组成部分之一.它交互式地向用户提供实时设备驱动程序外层的结构知识,引导用户编辑出语法正确的驱动程序.该系统的输入是由实时设备数据结构生成系统产生的数据模型.第1次输出驱动程序的抽象模型程序,语法正确但不完整,供用户编辑修改.此刻,系统在提示区显示:

Do you have the interrupt_service, yes or not?

如果你的驱动程序结构需要中断服务程序,则拍yes,系统自动切换到一种三窗口的屏幕状

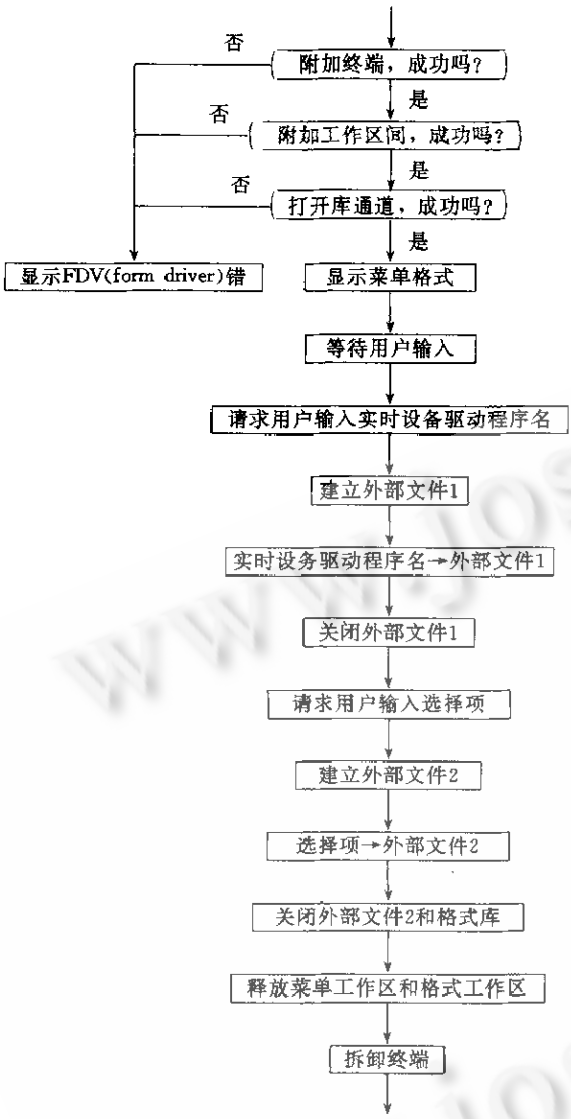

图1 格式驱动子程序流程

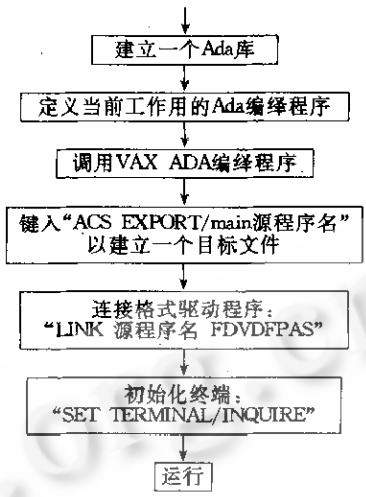

图2 运行可执行文件的操作流程

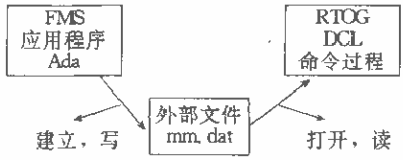

图3 选择项输入流程示意

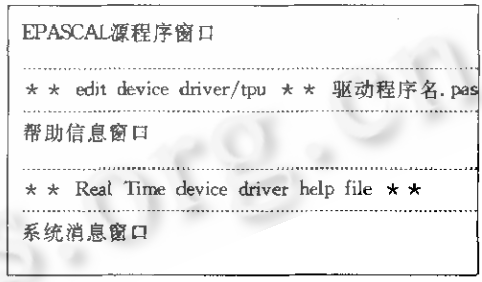

图4 一种三窗口编辑状态

态(图 4)，光标指向上窗口驱动程序抽象模型的末尾. 当用户发现缺少模块成分时，键入诸如 procedure，process_block 那样的成分名，系统即把相应的结构拷贝到光标所在的位置. 当用户深入某一过程内部，需要某些核心服务时，通过类似的作法，指定的核心服务迅即插到光标所指位置. 与此同时，关于该核心过程的帮助信息在中窗口自动滚出. 利用功能键，用户尚可获得联机帮助，比方说显示若干实时设备驱动程序的样板，辅助用户进行初始化操作等. 类似地，用户还能控制光标，以及窗口的切换与滚动. 上述工作方式，为用户开辟一个友好的编程环境，使得用户无需查阅手册，即可熟练地使用有关的核心过程，少犯甚至不犯程序结构方面的语法错误.

作为人机接口的主体，EPASCAL 多窗口编辑系统提高了开发人员的工作效率. 它由总控程序及其附属模块组成(文献[1]图 5)，其总控程序负责：(1) 开辟 3 个窗口；(2) 根据前一部分(菜单系统)传递过来的驱动程序名，产生驱动程序模块首部；(3) 读入后一部分(数

据结构生成系统)产生的数据文件;(4) 按照"初始化,写,读"的顺序建立操作模型;(5) 控制转移到正文编辑器;(6) 分辨两种情况("CTRL/Z,EX"保存结果,"CTRL/Q"不留结果)退出.

制作这一部分的工具是 VAXTPU(一个高级的可编程正文处理机构).

```
        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
 1 |                          data type and variable definitions                  | 1
 2 |                                                                              | 2
 3 |                                                                              | 3
 4 |                                  option:                                     | 4
 5 |                                                                              | 5
 6 | +------------------------------------------------------------------------+   | 6
 7 | |  const definitions                                              01     |   | 7
 8 | +------------------------------------------------------------------------+   | 8
 9 | +----------------------------------------------------------------------------+ 9
10 | |  type definitions:  |                                                      | 1
11 | |                     | register                                    02       | 1
12 | |                     | interrupe service routine communtion region 03       | 1
13 | |                     | non_record type                             04       | 1
14 | |                     | data_area                                   05       | 1
15 | |                     | [aligned(1)] packed record                  06       | 1
16 | |                     | [word] packed record type                   07       | 1
17 | |                     | record with variants                        08       | 1
18 | |                     | record with array                           09       | 1
19 | |  -------------------+                                                      | 1
20 | |                                                                  10       | 2
21 | |                                                                  11       | 2
22 | +----------------------------------------------------------------------------+ 2
23 |                                                                              | 2
12345678901234567890123456789012345678901234567890123456789012345678901234567890
        1         2         3         4         5         6         7         8
```

图5 管理类型和变量定义的子菜单格式

3 实时设备数据结构生成系统

这一部分的主要功能是,通过多级选单方式,将用户键入的数据结构直观表示,转换成语法正确且完整的 EPASCAL 数据结构,从而形成特定设备的数据模型.

由于面向种类繁杂、型号各异[3]的专用实时外设,设计难度较大.主要难点:(1) 如何描述各类各型设备的数据结构,如何定义各种复杂的数据类型;(2) 对于某一确定的类型,如何把用户键入的参数转换成 EPASCAL 描述的数据结构.当用户不需要某张类型定义子表或某些域时,如何砍掉它们;(3) 如何组织类型定义子表.

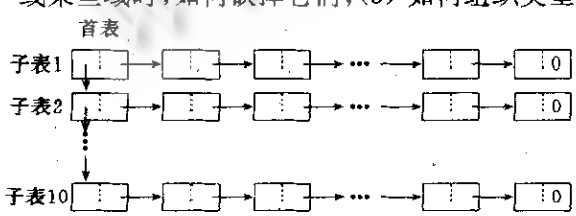

图6 子表的链接关系

为确保数据模型的灵活性与伸缩性,我们的实现策略如下:(1) 预制一批(10张)常用的类型定义子表,它们是在归纳消化现有各类各型实时设备驱动程序所用数据结构的特性后提炼而来.用户能够任选其中若干张,借以构造自己的数据结构;(2) 另设任意类型定义子表,供用户处理预定义类型之外的情况;(3) 提供管理类型定义和变量定义的子菜单,格式如图 5 所示,交用户自由地组织各种子表,并设置一个专门的

程序去管理它们的链接. 子表的链接关系如图 6 所示. 图 6 中，纵向链接不同的类型定义子表，横向链接相同类型子表的多次使用. RTDG 将同种类型子表所对应的数据结构，按照用户自定的先后次序和链接指针所指文件表位置，逐个并入一个数据文件之中. 实现这一部分的语言工具主要是 Ada.

4 实　例

我们利用 RTDG，快速地生成了一个高效的数模转换驱动程序. 其硬件环境如下：(1) 宿主机 MicroVAX II；(2)数模转换器通过一块数模转换接口板与主机连接. 配备该驱动程序，我们在四笔记录仪上加以检测(图 7)，成功地画出正弦曲线与直方图(图 8).

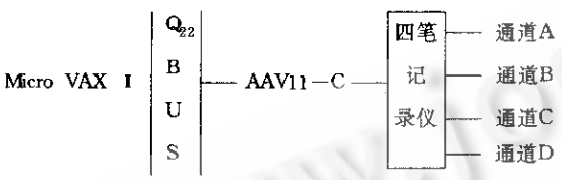

图7 数模转换驱动程序的检测环境

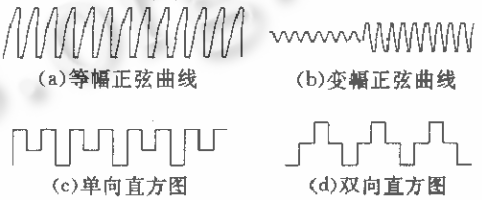

图8 四笔记录仪的输出

5 结束语

遵循本文介绍的技术途径，一个真实的实时设备驱动程序半自动生成器 RTDG 已于 90 年代初在 VAX 和国产太极机序列上开发成功，同时为某测控系统实际产生了一个数模转换器驱动程序，顺利通过部委级鉴定，获部委级三等奖. 根据我们的实践经验，使用RTDG 生成实时设备驱动程序有两大优点：

(1) 开发效率高

① 交互式地生成驱动程序的抽象模型，对用户尤其是初学者非常方便. 在 RTDG 循序渐进的引导下，用户轻而易举地达到驱动程序的一个抽象模型，进而借助于 EPASCAL 多窗口编辑系统提供的结构成分扩充功能，即可获得一个结构完整的框架；

② 在对抽象模型逐步细化的过程中，核心服务的引用与显示功能使得用户很容易掌握其用法. 此外，用户通过多功能键，还能够灵活地编辑与修改程序.

③ 围绕用户键入的一个驱动程序名，用户以选单方式指挥编译、连接、系统构造，直至装入操作自动地进行. 当编译出错时，尚可自动切换到编辑状态. 所谓“交互式”，仅指用户的选单动作.

(2) 构造出来的实时设备驱动程序效率高

① 高级语言 EPASCAL 配有面向字节的 I/O 操作，让用户得以进行快速的设备处理. 例如，为了按指定的要求把数据写入内存或从内存读出，驱动程序须置/清设备的控制状态寄存器(CSR)里的状态值. 通用外设一般求助于多层系统服务去调用写寄存器的核心过程 Write_register 来实现[6]，系统开销大，不适合实时外设，尤其是响应时间限于微秒级的强实时外设的需要，而 RTDG 则向用户提供跃过若干中间环节的手段.

② 专用实时的应用程序有能力直接初始化 I/O 请求并读写外设，无需绕道运行系统的

开销.

综上所述,RTDG是我们在专用实时程序设计自动化领域的一项阶段性成果,从中显露出全自动生成实时设备驱动程序的诱人前景:用户只需提交关于设备特性的书面报告和功能要求(Specification),RTDG据此即可全自动地为之生成高效的驱动程序.这正是我们下一步的努力目标.

参考文献

1 白光野,徐崇,范植华等.从软件工程的发展看软件自动化.软件学报,1995,6(增刊):292～300.

2 范植华,邢国光,李小白等.计算机专用实时母系统在VAX/VMS环境下的实现.电子学报,1993,21(5):85～87.

3 范植华,邢国光.专用设备驱动程序交互式生成器的总体设计.第二届专用实时操作系统研讨会论文集,北京,1989.

4 DEC Inc. VAX/VMS DCL Dictionary. 1985.

5 DEC Inc. VAX/FMS Introduction to FMS. 1985.

6 Andrew S. Tanenbaum. Operating systems design and implementation. Prentice—Hall Inc., 1988.(中译本:陆佑珊,施振川译.操作系统教程MINIX的设计与实现.世界图书出版公司,1990.)

A STUDY ON AUTO—GENERATION OF REAL—TIME DEVICE DRIVERS

Fan Zhihua

(*Institute of Software The Chinese Academy Sciences Beijing* 100080)

Xing Guoguang

(*Beijing Institute of System Engineering Beijing* 100101)

Abstract Automatic generation of software systems is the stream of the 90s in software engineering. Efficient management and flexible addition of special real—time devices is the basic function indispensable to any special real—time computer system. Moreover, their types and classes, and their quantity and quality are increasing with each passing day. How to overcome special real—time peripheral drivers' liability to variation is the key to automatic generation of any special real—time system. The paper relates to the preliminary result of our study of automated drivers of special real—time devices, with focus on the inner construction and implementation of the semi—automatic generator of special real—time device drivers, RTDG (real time device generator), which the authors develop on their own, and on a practice—tested application example. The stage result from our study of automatic generation of special real—time peripheral drivers has already passed the authentication on ministry level.

Key words Special real—time parent system, real—time application sub—system, real—time peripheral drivers, menu—driven and —controlled system, multi—window editor, generator of real—time peripheral data structures.