2000系列
软件资料

VAX-11FMS
语言接口手册

2000系列资料出版中心

前　言

美国DEC公司的VAX—11系列机是举世公认的优秀的32位超级小型机系列。它不但具有功能完善的指令系统、灵活巧妙的寻址方式、多种数据类型以及虚拟型存贮等特点，并且其结构面向操作系统，具有丰富的软件支持。

自七十年代末以来，VAX系列机已在世界各地得到广泛应用，我国已有不少单位引进该系列的各种机型，它具有广阔的发展前景。为推动国内计算机事业的发展，并考虑到VAX用户及高等院校教学的实际需要，主管部门组织电子工业部华北计算所、中国科学院沈阳计算所、中国科学院高能物理所、成都电讯工程学院、暨南大学、北京信息工程学院、航天工业部一院十二所等单位，成立了VAX系列机资料出版中心，组织经验丰富的软件专业人员翻译出版全套的VAX随机软件资料。

这套资料的第一批包括VAX／VMS（3.6版）十卷38册（VAX／VMS的一般介绍、命令语言和系统信息、文本编辑和格式化程序、程序开发工具、系统服务和I／O、运行时间库、VAX—11记录管理、兼容方式、系统程序设计、系统管理及操作），网络一卷1册、可选的VAX／VMS选件八卷23册（包括：FORTRAM、BASIC、PL／I、COBOL、BLISS—32、C、PASCAL、CORAL—66等语言），DBMS（数据库）一卷12册，CDD（公共数据字典）一卷3册，数据检索一卷6册，总共二十二卷85册。

第二批资料包括VAX Rdb／VMS（七册）、VAX—11 LISP语言（三册）、VAX—11 CEP／VMS（五册）、VAX Cluster（一册）、VAX—11 RGL（四册）、Micro VMS（五册）。

第三批资料包括VAX FMS（六册）、VAX TDMS（八册）和VAX GKS（四册）三卷共十八册。现已全部出版。

迄今为止，这是一套最完整、最系统、最丰富、最实用的软件资料。这套资料对VAX系列各档机的用户是必读资料，对从事计算机研制的各单位以及高等院校计算机工程系的教学是重要的参考资料；对各大专院校，各省、市图书馆也是珍贵的馆藏资料。

《**VAX—11 FMS语言接口手册**》（序号：AA—N209A—TE；操作系统及版本：VAX／VMS V3.2，软件版本：VAX—11 FMS V2.0 V1.5）是由电子工业部华北计算所陈国钦翻译，宗拔梅审校，由《**小型微型计算机系统**》编辑部编辑、出版、发行。

由于时间仓促、水平有限，因此一定有不少错误和不妥之处，敬请广大读者批评指正。

2000系列资料出版中心

一九八八年一月

目　　录

序　　言

本手册介绍FMS和VAX—11程序设计语言之间的语言接口。着重讨论特定语言中与FMS应用程序设计有关的问题。同这些问题有关的例子和注意事项都提供了。对本手册里所介绍的每一种语言，都编写了一个样品应用程序。样品应用程序(SAMP) 的软件，是FMS Version 2配给工具包的一个组成部分。本手册中每一种语言占一章，章末提供了程序代码，这是为了帮助你理解FMS，帮助你编写你自己的程序。取自样品应用程序的例子遍布全文。

读者

本手册适合于利用FMS写程序的程序员。可以用任意一种VAX—11程序设计语言来编写FMS程序。我们假定程序员对他为应用程序所选定的那种语言是熟悉的。本手册不是程序设计技术方面的指南，也不是某种语言的用户指南。

本手册的结构

本手册由八个章三个附录组成。

第1章是语言接口概述，给出适用于所有程序设计语言的一般信息。这一章还介绍了样品应用程序，它是Version 2 FMS软件的一部分。

第2章到第8章介绍FMS和七种语言之间的语言接口。每一章介绍如何用一种特定的语言来编写FMS应用程序。

- 第2章　VAX—11　BASIC
- 第3章　VAX—11　BLISS—32
- 第4章　VAX—11　C
- 第5章　VAX—11　COBOL
- 第6章　VAX—11　FORTRAN
- 第7章　VAX—11　PASCAL
- 第8章　VAX—11　PL/I

在每一章的末尾，出现用该语言写的样品应用程序以及该语言的定义文件。还提供了命令文件，用来运行每一种语言的样品应用程序。

附录A是“表格驱动程序调用”表。

附录B提供样品应用程序表格说明以及这些表格的屏幕影象。

附录C提供样品应用程序的数据文件。

本手册的用法

如果你要用本手册介绍的某种语言来编程序，去读该语言的那一章就行了。因为每一种语言的章都是自含的，为了易于访问，你可以把任何一章移开。如果你要用某种本手册不介绍的语言来写程序，那么请参阅第1章以及附录A。

如果你正在为你的应用程序挑选一种语言，那么你可能想要重新读一下你正在考虑中的那几种语言所在的章节以及各样品应用程序。本手册並没有对程序设计语言进行比较，但每一章确实都讨论了该语言所特有的程序设计考虑。

文档中的一些约定

方括号〔〕　表示该条目是可选的。

垂直省略号　表示在一个例子里没有把全部句子都给出来。

⋮

除非另行指出，否则你总是压下RETURN键来结束你的命令。

第一章　语言接口概述

FMS表格驱动程序按照VAX—11标准以及语言特有的规则来处理所有表格驱动程序调用。这些规则规定，如何传递自变量给表格驱动程序，后者又如何把值返往给你的程序。你的应用程序必须符合VAX—11 FMS接口的要求。每章中都介绍这几个专题：

- 表格驱动例程
  - 调用表格驱动例程作为过程
  - 存取表格驱动状态码作为函数
- FMS中自变量的传递
- 空自变量
- FMS数据类型
  - 字符串
  - 长字二进整数
  - 字二进整数
- 非FMS数据类型
- 一维数组
- 分配工作空间，终端控制区域和运行时刻存贮器驻留表格区域
- 使用FMS应注意的问题
- 数据转换
- 样品应用程序

本章只简单地介绍一下这些专题，而与特定语言有关的细节，则在后面各章给出。不打算使用本手册介绍的各语言的程序员，应阅读本章，以得到他们需要知道的信息。

FMS应用程序可以用任何VAX—11语言加上附录A给出的辅助工具来编写。附录A包含重要的语言接口信息：每个表格驱动程序调用的调用序列，数据存取代码，数据类型以及传递机构，都以与语言无关的表示法来提供（这种表示法由VAX—11过程调用和条件处理标准规定之）。

1.1　表格驱动例程

全部表格驱动例程都被当作过程或函数来调用。其语法遵照你的VAX—11语言的标准要求。

1.1.1　调用表格驱动例程作为过程

利用过程调用语句来调用一个FMS表格驱动例程。调用语句把控制转移到一个FMS例程的一个入口点，任选地还可以传递自变量给它，保存调用者程序的返回地址。调用语句必须遵守VAX—11过程调用和条件处理标准。至于细节，请参阅VAX—11运行时刻库参考手册的附录C

1.1.2　存取表格驱动状态码作为函数

在所有表格驱动调用完成时，一个FMS状态码被返送给调用者程序。可以用几种方法

来存取所返送的FMS状态码。（参见VAX—11 FMS表格驱动程序参考手册中关于状态返送方法的介绍。）

大多数语言都有办法使返送的值可用于程序。在许多语言中，如果你通过函数引用而不通过调用语句来指定例程的话，FMS状态码就可用于调用者程序。在这些情况下，你利用你的语言中调用一个函数的标准语法。一般来说，状态返送在寄存器零中。

1.2 在FMS中自变量的传递

自变量传递机构，涉及把数据传递给被调用例程的方法。VAX—11过程调用标准有三种传递自变量的方法：

- 通过引用
- 通过描述符
- 通过值

不过FMS希望你只通过引用或通过描述符来传递自变量。

通过引用指的是，自变量的存贮单元被传递给例程。FMS希望整数要通过引用来传递。

通过描述符指的是，一个描述符数据结构的地址被传递给例程。FMS希望字符串和数组要通过描述符来传递。

1.3 空自变量

当调用语法包含可选的自变量，而你又不想指定全部信息时，你可以使用空的自变量。为了简化你的程序，可以省略任何可选的自变量。在某些程序设计语言中，对每一个空自变量，用一个逗号来占据其位置。而在另一些语言中，则为每一个空自变量指定一个零地址。在最后一个必需的自变量的右边的那些可选自变量，可以从调用中简单地略去了事。

1.4 FMS数据类型

1.4.1 字符串

字符串是FMS使用的一般数据类型之一。你必须保证，你的串最初被说明时其长度足以容纳你的FMS数据。虽然FMS既接受动态串作为自变量，也接受定长串作为自变量，但它处理串时，却把所有的串都当成长定的。换句话说，当表格驱动程序返送一个值给输出自变量时，FMS并不更改一个动态串描述子的长度。

可以用两种方法来满足FMS的定长串约束条件。一个是，把你的定长串说明成将被返送的FMS数据的真正长度。你可以使用FMS/DESCRIPTION/BRIEF命令来确定串的长度。

另一个是，把一个单一的串，使用于几个不同的FMS调用中，用来从多个表格和字段中传送数据，或传送数据到多个表格和字段。你在说明这个串的长度时，必须不短于将返送给你的程序的最长字段值串。你同样可以使用FMS/DESCRIPTION/BRIEF命令来存取这个信息。利用FDV$RETLE调用来返送串变量中字段值有效部分的长度。然后，当访问已被打入该字段的数据时，可以利用这个长度。FDV$RETLE调用在通用用户动作例程中得到了应用。

1.4.2 长字二进整数

长字二进整数是FMS使用的另一个一般数据类型。数值自变量必须是长字二进整数。

如果你企图传递别的数值类型给表格驱动程序，该调用不能正确地工作。一个例外是FDV$DFKBD调用（见下一节）。

1.4.3 字二进整数

defkbd自变量是个字整数数组，当FDV$DFKBD例程被调用时传递这个自变量。FMS希望通过描述符来传递一个字整数数组。

1.5 非FMS数据类型

FMS所不认识的数据类型，只要它们不被传递给表格驱动程序，在你的应用程序中就尽管使用好了。

1.6 一维数组

一维数组是结构，在FMS中可用于如下自变量：

- tca（终端控制区）
- wksp（工作空间）
- mloc（存贮单元）
- defkbd（定义键盘）

你必须为FMS提供这些自变量的存贮空间。为此目的，你可以把它们定义成：

- 长字整数数组或字符串——对tca，wksp，mloc
- 字整数数组——对defkbd

1.7 分配工作空间，终端控制区域和运行时刻存贮驻留表格区域

FMS工作空间变量，终端控制区变量，和运行时刻存贮器驻留表格区变量，是字符串或长字整数数组。这些变量须被放在你的程序的一个公共存贮区中。

表格驱动程序用12个字节来关连有关每个工作空间和终端控制区的用户信息。一个运行时刻存贮器驻留表格区的分配量必须是表格的尺寸。所有这些变量的空间，均由你的应用程序分配之。注意，FMS只使用12个字节的空间于工作空间和终端控制区分配。多于12个字节的分配是浪费的。工作空间，终端控制区，和运行时刻存贮器驻留表格区，都只准说明一次。当最初把它们的地址传递给表格驱动程序之后，FMS便记住了这些地址。

对每个工作空间，表格驱动程序还根据你对存贮量的估计（为存放你的最大表格所需要的存贮量），分配一个附加的存贮空间量。如果你的估计偏小，表格驱动程序会自动地多分配些空间，但这影响性能。恰当的估计导致表格驱动程序更有效地操作。你可以利用FMS/DIRECTORY/FULL命令来发现分配了多少空间。

1.8 使用FMS应注意的问题

1.8.1 由FMS专用的存贮区域

终端控制区、工作空间、运行时刻存贮器驻留表格区的单元，是FMS专用的。终端控制区和工作空间通过FDV$ATERM调用和FDV$AWKSP调用来加接，一直保留到发出FDV$DTERM或FDV$DWKSP调用，或者到程序结束。运行时刻存贮器驻留表区被用于FDV$READ调用，一直保留到发出FDV$DEL调用，或者到程序结束。在你的程序

中，除了把终端控制区，工作空间，或运行时刻存贮器驻留表格区的地址传递给表格驱动程序之外，你绝对不要去碰这些区域。

1.8.2 为什么你应使用公共存贮区

传递给如下几个表格驱动例程的参数，应该谨慎地使用：

FDV$ATERM	加接终端
FDV$AWKSP	加接表格工作空间
FDV$READ	读表格到存贮器
FDV$SSRV	指定状态报告变量

例如，一旦发出FDV$SSRV调用，包含FMS状态和RMS状态的程序变量就变成易失的，且可以在任一调用点改变。作为一般规则，你应把状态报告变量放在静态存贮中。

在既需要FMS状态又需要RMS状态的情况下，可以使用FDV$STAT例程。注意，只有FDV$STAT调用和FDV$SSRV调用才提供RMS状态。有了FDV$STAT例程，你就不必担心易失性了。

终端控制区、工作空间、运行时刻存贮器驻留表格区，以及状态报告变量，所有这些单元，在表格驱动程序正在使用它们时，全都必须是一直存在的。它们必须一直保留到终端控制区和工作空间被断开（detached），存贮单元中的表格被删除，和状态报告变量不再被用为止。可以把变量放在一个公共存贮区中，这样可以保护变量，要不然的话，编译程序可能把它们放进动态存贮区。

1.9 数据转换

FMS只用ASCII字符串来显示数据。在终端屏幕上显示的所有信息，以及从终端操作员那儿接收来的所有信息，均是作为ASCII字段值来表示的。所以，要操纵数值数据的话，需要把ASCII字符串转换成数值数据，和把数值数据转换成ASCII字符串。你可以利用语言固有的转换操作功能，也可以建立你自已的转换功能。

1.10 样品应用程序

1.10.1 语言接口手册

每一个语言的章都有样品应用程序，是用该语言来写的。同样，还提供了一些有关的文件。

- 用来建造样品应用程序的命令文件：包含为了编译和连接你的程序所需要的全部信息。这个命令文件放在样品应用程序的源程序之前。安装FMS时，这个命令文件被放在目录FMS$EXAMPLES中。
- VAX—11 FMS样品应用程序：作为一个示范程序被纳入FMS配给工具包。安装FMS时，所介绍的每一种语言的样品程序被放在目录FMS$EXAMPLES中。样品应用程序出示了FMS提供的大多数功能。它被设计来作为一种学习工具。来自样品程序的例子，在本手册各处比比皆是。
- 样品程序的定义文件或其它引用文件（IncludeFites）：是样品应用程序包的组成部分。安装FMS时，这些文件被放在目录FMS$EXAMPLES中。这些文件包含样品应用程序用到的表格驱动例程的种种代码。虽然这些文件是为样品程序而建立的，但当你为自已

的应用程序建立定义时，它可为你提供一个有益的起点。

其它与样品应用程序有关的参考材料作为附录出现。附录B给出表格说明和屏幕影象。附录C给出数据文件

1.10.2 其它FMS文档

在FMS文档集里到处出现用BASIC语言写的样品应用程序（SAMP．BAS”）中的例子。“VAX—11 FMS介绍”这本资料的第2章，引导读者一步一步地走过样品应用程序，指出程序运行时FMS的能力。它的后面各章介绍执行诸如滚动和使用命名的数据这类具体操作所需要的代码。

第二章　用VAX—11 BASIC编写FMS应用程序

FMS表格驱动程序按照VAX—11标准以及语言特有的规则来处理所有表格驱动程序调用。这些规则规定，如何传递自变量给表格驱动程序，后者又如何把值返送给你的程序。本手册简略地介绍了语言特有的信息。至于细节，参见VAX—11 BASIC文档集。

你的VAX—11 BASIC应用程序必须符合VAX—11 BASIC FMS接口的要求。本章介绍的专题包括：

- 表格驱动例程
  调用表格驱动例程作为子程序
  存取表格驱动状态码作为函数
- FMS中自变量的传递
- 空自变量
- FMS数据类型
  字符串
  长字二进整数
  字二进整数
- 非FMS数据类型
- 一维数组
- 分配工作空间，终端控制区域和运行时刻存贮器驻留表格区域
- 使用FMS应注意的问题
- 数据转换
- 用VAX—11 BASIC写的样品应用程序

本章末尾有一个用BASIC语言写的样品程序（SAMP·BAS）。跟在SAMP·BAS代码后面的，是为SAMP·BAS建立的表格驱动程序定义文件。而建造样品应用程序所需要的命令文件信息，则放在2.10.2节。

文中到处利用样品应用程序中的例子来说明语言问题。在没有合适的SAMP. BAS例子的场合就提供其他例子。

2.1　表格驱动例程

你可以调用任一FMS例程作为一个子程序或一个函数。其语法要遵守标准的VAX—11 BASIC的要求。

2.1.1　调用表格驱动例程作为子程序

利用过程调用语句来调用一个FMS表格驱动例程。例如：

```
12235 CALL FDV$WAIT
```

调用表格驱动例程FDV$WAIT且不传递任何自变量。

```
5070 CALL FDV$GET(OPTION$, TERMINATOR%, , OPTION')
```

调用表格驱动例程FDV$GET且传递三个自变量。

请参阅附录A，那儿列出全部表格驱动程序调用。每一个表格驱动例程的调用序列，数据存取代码，数据类型，以及传递机构，都以与语言无关的表示法来提供（这种表示法由VAX—11过程调用和条件处理标准规定之）。关于VAX—11过程调用和条件处理标准的细节，参见VAX—11运行时刻库参考手册。

2.1.2 存取表格驱动状态码作为函数

每个表格驱动程序调用，当其完成时，都返送一个FMS状态码给调用者程序。为了从一个表格驱动例程那儿接收所返送的状态码，你应通过函数引用而不是通过调用语句来激活该例程。注意，这样做返送的是一个标准的VMS状态码。为了可移植性，也可以采用其它状态机构。（至于细节，参阅VAX—11 FMS表格驱动程序参考手册第2章。）

用EXTERNAL LONG FUNCTION语句来说明FMS 函数。下列语句说明並调用FDV$GET作为一个FMS函数。

```
EXTERNAL LONG FUNCTION FDV$GET
RETURN_STATUS=FDV$GET(OPTION$, TEMINATOR%, 'OPTION')
```

2.2 FMS中自变量的传递

自变量传递机构涉及向一个被调用例程传递数据的方法。VAX—11过程调用标准有三种传递自变量的方法：

- 通过引用
- 通过描述符
- 通过值

不过，FMS例程希望只通过引用和描述符来传递自变量。

通过引用，指的是自变量的存贮单元被传递给例程。FMS希望整数要通过引用来传递，这也是BASIC对整数的默认传递机构。

通过描述符，指的是描述符数据结构的地址被传递给被调用的例程。FMS希望字符串和数组通过描述符来传递，这也是BASIC对字符串和数组的默认传递机构。

2.3 空自变量

当调用语法包含可选的自变量，而你又不想指定全部信息时，你可以使用空的自变量。为了简化你的程序，可以省略任何可选的自变量。对每一个空自变量，用一个逗号来占据其位置。在最后一个必须的自变量的右边的那些可选自变量，可以从调用中简单地略去了事。下例中，FDV$GETAL调用只传递字段终止符值：

```
14058 CALL FDV$GETAL (, TERMINATOR%)。
```

2.4 FMS数据类型

2.4.1 字符串

字符串是FMS使用的一般数据类型之一。例如，FDV$GET调用传递的字符串是字段值（OPTION$）和字段名（'OPTION'）：

```
5070 CALL FDV$GET(OPTION$, TERMINATOR%, 'OPTION')
```

2.4.1.1 定长串的说明

你要保证你的串当初被说明时，其长度足以容纳你的FMS数据。虽然FMS既接受动态串作为自变量，也接受定长串作为自变量，但它处理串时，却把所有串都当成定长的。换句话说，当表格驱动程序返送一个值给输出自变量时，FMS并不更改一个动态串描述子的长度。

动态串的初始长度为零。因此，如果传递一个尚未赋值的动态串给表格驱动程序的话，所传递的串是一个长度为零的串。但表格驱动程序却要返送非零长度的数据给你。于是该数据放不下。解决这个问题的一个办法是，予先把动态串扩充到所要返送的FMS数据的实际长度。你可以利用FMS/DESCRIPTION/BRIEF命令来确定串的长度。一旦动态串被指定一个非零长度，它就可以供FMS使用了。

在样品应用程序中到处都在做动态串的予扩充工作。许多串都用空格予扩充到期望的串长度。下例中，通过在引号内安排三个空格，把串FIRSTL$和LASTL$予扩充到长度为3。

```
11937  FIRSTL$ = '   '
11938  LASTL$ = '   '
11940  CALL  FDV$RETDN( 'FIRST, FIRSTL$ )
11945  CALL  FDV$RETDN( 'LAST', LASTL$ )
```

你也可以利用BASIC函数SPACE$来予扩充一个串。SPACE$返往一个串，包含指定个数的空格。在下例中，SPACE$把串PASSWORD$予扩充成12个空格：

```
14060  PASSWORD$ = SPACE$ ( 12 )
14062  CALL FDV$RET ( PASSWORD$, 'SECRET' )
```

作为上述过程的一个替代办法，你可以利用MAP语句和COMMON语句来把串说明成定长的。由于这两个语句被用来分配静态存贮，故所指定的串必是定长的。因此，下例中的所有串均是定长串：

```
215  MAP ( ACCOUNT ) ACCOUNT$ = 151
220  MAP ( ACCOUNT ) ACCTNO$ = 5,   ACCDATE$ = 7,   LAST$ = 20, &
                     FIRST$ = 15,   MIDDLE$ = 15,  STREET$ = 30, D&
                     CITY$ = 20,    STATE$ = 2,    ZIP$ = 5,
                     HOMEPH$ = 10,  WORKPH$ = 10,  OPW$ = 12
222  MAP ( TACCOUNT ) TEMPACCONT$ = 151
```

2.4.1.2 把一个串变量用于多个表格和字段

你还可以用另一种办法来满足FMS的定长要求。一个单一的串变量可以被用于不同的FMS调用中，以便传送数据到多个表格和字段，或从多个表格和字段中传走数据。你说明这个串变量时，其长度至少必须等于将返送给你的程序的最大字段值串之长度。你可以利用FMS/DESCRIPTION/BRIEF命令来得到这一信息。利用FDV$RETLE调用可以得到串变量中字段值的有效部分之长度。然后，当访问打入这个字段的数据时，便可以使用这一长度。

```
100          MAP ( ACCOUNT ) ACCOUNT$ = 100;

.
.
400          CALL    FDV$GET ( ACCOUNT$, TERMINATOR%, 'FEED' )
410          CALL    FDV$RETLE ( LENGTHFIELD%, 'FIELD' )
```

```
500     FLDVALUE$=SEG$=SEG$(ACCOUNT$,1%,LENGTHFIELD%)
```

在FDV$RETLE调用执行完后，LENGTHFIELD%等于名为'FIELD'的字段的长度。它也等于变量ACCOUNT$的有效部分。现在，可以使用LENGTHFIELD%作为长度来访问被打入字段'FIELD'的数据，而它现在是在变量ACCOUNT$中。若你访问ACCOUNT$时不使用BASIC SEG$函数的话，你访问的是整个变量，包括表格驱动程序充填这个串时所填入的所有空格。FDV$RETLE调用在通用用户动作例程中得到了应用。

2.4.2 长字二进整数

长字二进整数是FMS使用的另一个一般数据类型。例如，FDV$ATERM调用传递长字值，给出终端控制区尺寸（12）和逻辑I/O通道号（2）：

```
1040        CALL FDV$ATERM(TCA%(),12%,2%)
```

数值变量必须是长字二进整数。若你企图传递别的数值类型给表格驱动程序，那么调用将不能正确工作。一个例外是FDV$DFKBD调用（见下一节）。

2.4.3 字二进整数

defkbd自变量是个字整数数组，当FDV$DFKBD例程被调用时，传递这个自变量。FMS希望通过描述子来传递数组。

2.5 非FMS数据类型

FMS所不认识的BASIC数据类型，只要它们不被传递给表格驱动程序，在你的BASIC应用程序中就尽管使用好了。

2.6 一维数组

一维数组是结构，在FMS中可用于如下自变量：

- tca（终端控制区）
- wksp（工作空间）
- mloc（存贮单元）
- defkbd（定义键盘）

你必须为FMS提供这些自变量的存贮空间。为此目的，你可以把它们定义成：

- 长字整数数组或字符串——对tca，wksp，mloc
- 字整数数组——对defkbd

在样品应用程序中，tca，wksp，mloc自变量被传递给好几个表格驱动例程。这些自变量被定义成整数数组变量。你也可以把这些变量定义为字符串。(串可以是静态的也可以是动态的，但动态时它必须被扩充至正确的长度。)

下面的说明语句为整数数组变量WORKSPACE%，CHECKWKSP%，TCA%，和MENU_FORM%建立名字和存贮：

```
130   DIM   WORKSPACE%(3)          ! General   workspace
135   DIM   CHECKWKSP%(3)          ! Check     workspace
140   DIM   TCA%(3)                ! Terminal  Control Area
145   DIM   MENU-FORM%(500         ! Storage for memory-resident form
```

注意，在BASIC中，当访问一整个数组时，数组名后面必须跟着一对括号（），以便把数组区别于同名的标量。因此，当FDV$ATERM传递整个数组TCA%，数组名被写