# VAX/VMS操作系统

# 怎样写设备驱动器

国防科技大学研究所六〇二教研室

一九八五年六月

# 前　言

如何为非 DEC 设备，将其安装到 VAX-11/780 系统上，并正确写出该设备的驱动程序，也是许多同志很关心的一个问题，也是一个较复杂的问题。涉及的面较广，不仅要了解 I/O 数据基（即 I/O 数据结构），了解中断的分派方法，了解系统提供的大量宏指令，了解系统提供的许多例行程序的功能及调用方法，了解有关 VAX 系统的全局符号，而且要了解具体设备的特性，诸如操作功能，有关寄存器，以及设备与 VAX 系统的接口情况。

本材料首先介绍了设备驱动器的结构、功能，I/O 数据结构，以及驱动器中常用的例行程序，并重点介绍了如何写一个设备驱动器，并以宽行打印机驱动器为例，分析了怎样写一个设备驱动程序。许多工作必须按规定去做，不能想当然，否则就会出错。

我们介绍与分析的驱动器是 VAX/VMS 版本 2.0 上实用的，版本 3.0 可能稍有变化，但其思想以及各种工作是类似的，或者是一样的。

由于成文时间仓促，错误不当之处在所难免，请批评指正。

编　者

1984 年 9 月

# 目　　录

# 第 一 章

# 设备驱动器介绍

## 1. 概 述

设备驱动器是一组表格和例行程序的集合。操作系统用它来实现用户的I/O请求，通常一类设备有一个驱动器。

驱动器的功能是：予置控制器和设备；检查用户的I/O请求是否合法；启动设备进行I/O传输；处理设备中断；处理设备超时；报告并登记设备错误；撤消I/O操作；完成I/O请求等。

用户程序，驱动器和设备间的关系如下图：

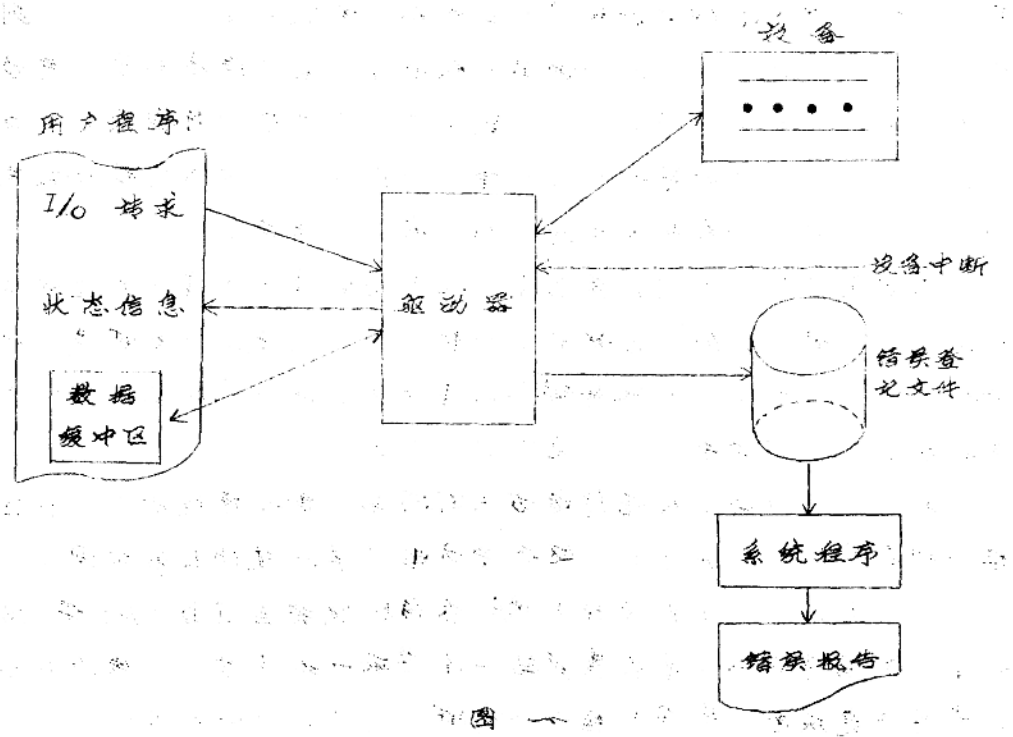
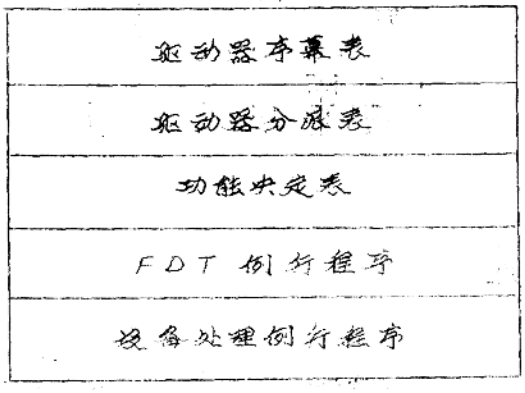



图 一

驱动器的结构：驱动器的结构如图二所示：



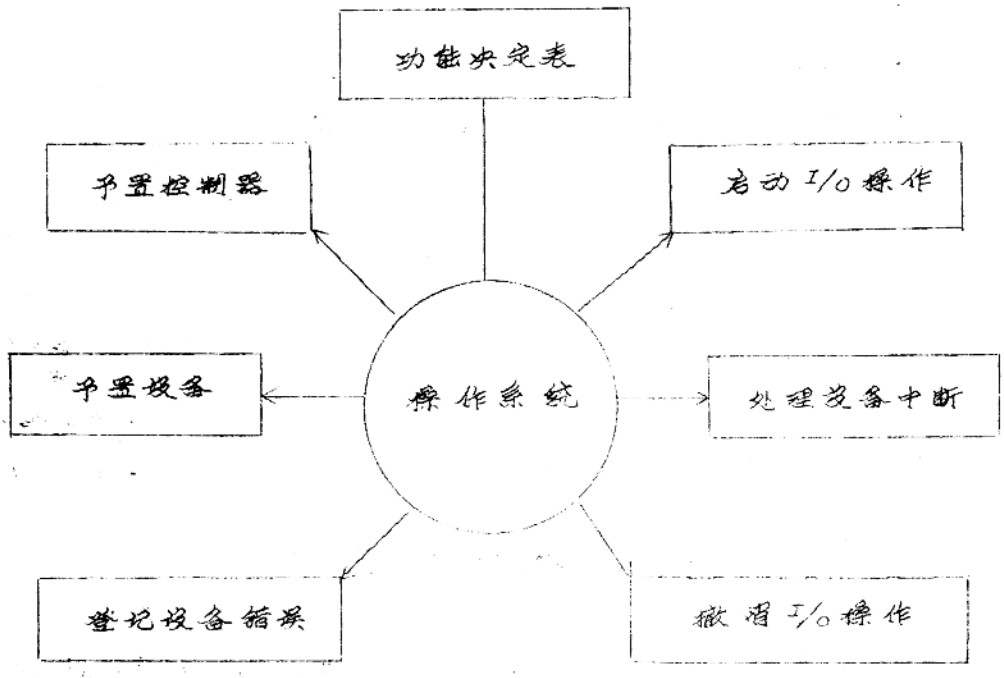

图二． 设备驱动器的一般结构

其中驱动器本幕表（DDT）为驱动器装入过程（SYSGEN）描述了驱动器（名字、尺寸等）和设备类型，驱动器分派表（DDT）列出了驱动器各个例行程序的入口地址，功能决定表（FDT）列出了所有合法的$QIO$功能码，指出了哪些是缓冲功能，并为每个$QIO$功能码指出了对它进行I/O予处理例行程序的入口地址。FDT例行程序和设备处理例行程序就是在驱动器分派表和功能决定表中所列的入口地址的那些例行程序。

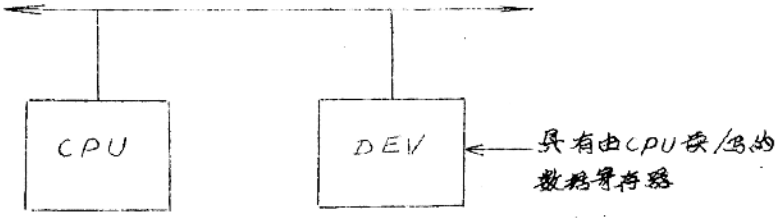
当进行I/O处理时，操作系统和驱动器通过共享的数据结构协同操作，对用户的I/O请求，由操作系统决定在什么时候执行何种功能，进入驱动器的哪个例行程序。操作系统与驱动器之间的关系如下图所示。 （见下页图三）

数据传送方法：数据传送方法有两种，即程序设计I/O和直接存储器访问（DMA），这两种方法主要指硬件上的不同。

程序设计I/O（见图四）的设备传输数据必须由CPU读/写设备的数据寄存器，每次读/写一个字或一个字节，在传输每个字或字节完成时，设备产生一个中断，已通知中央处理机。




图三．操作系统和驱动器例行程序

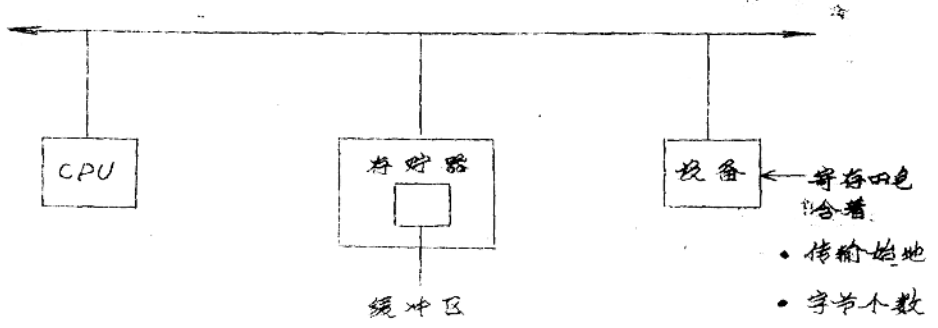



图四．程序设计的 I/O 设备

直接存贮器访问（DMA）的设备传输信息不要求频繁地占用处理机，一旦驱动器激活了设备之后，设备就能传输大号信息，这样：

- CPU告诉设备有多少字节或字要传输；
- CPU告诉设备用户缓冲区在存贮器中的位置；

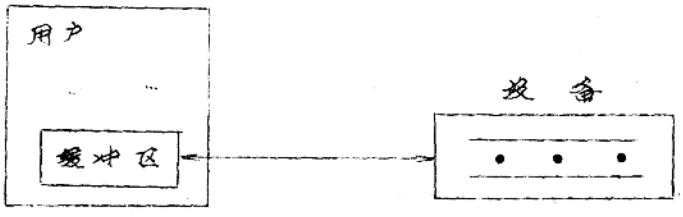
- 设备传输一个信息块；
- 设备直与存贮器通讯，而不经过中央处理机；
- 在该块传输结束时，设备产生中断。




图五. 直接存贮器访问设备

I/O 的类型：I/O 的类型有两种，即直接 I/O 和缓冲的 I/O，这两种类型主要指软件的不同。

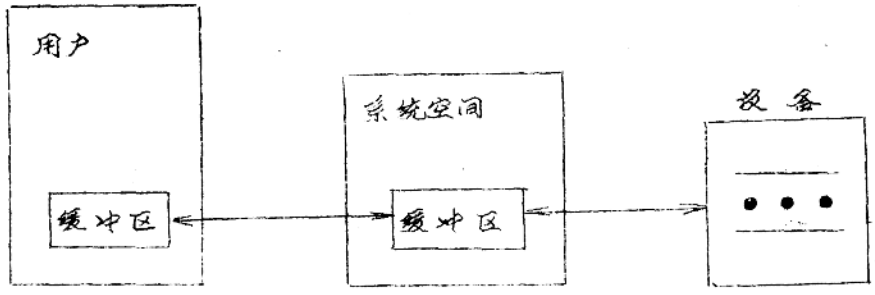
直接 I/O（见图六）允许信息在用户缓冲区和设备之间直接传输，在传输过程中要封锁缓冲区中的物理页，直到 I/O 完成。这种类型仅对 DMA 设备有用。




图六. 直接 I/O

缓冲的 I/O（见图七）是通过系统地址空间的一个间接缓冲区（系统缓冲区）与设备进行信息传输的，也就是用户缓冲区不能直接与设备传输信息，必须通过系统缓冲区来进行。这时要求用户进程完全是可交换的和可分页的。这种类型对 DMA 和程序

设计设备都是有用的。




图七 缓冲的 I/O

变换寄存器：由 UNIBUS 适配器和 MASSBUS 适配器使用变换寄存器，每个适配器具有自己的一组变换寄存器，用这些寄存器将 UNIBUS/MASSBUS 的地址（这些地址是设备理解的）翻译成系统理解的地址。

数据路径：单总线适配器对实现直接存对访问（DMA）传送的 UNIBUS 设备，通过两种数据路径实现信息传输。（注意：MASSBUS 没有数据路径的概念），即直接数据路径和缓冲数据路径。

在任何时刻能够由任何设备使用直接数据路径。

在一个设备能够使用之前必须为该设备分配缓冲的数据路径。注意数据路径仅对 UNIBUS 的 DMA 操作有意义。

直接数据路径 ——每个 UNIBUS 适配器仅一个，在一个时刻一次仅允许 8 位或 16 位传输到系统存贮器中。

缓冲的数据路径 —— 每个 UNIBUS 适配器 15 个，在一个时刻，数据的 32 位或 64 位传输到系统存贮器中。

一般地，缓冲的数据路径比直接数据路径更有效，并且只要有可能就应当使用缓冲的数据路径。

驱动器的分支处理：

驱动器应当在尽可能低的 IPL 级上运行。这样驱动器有时要求降低它的 IPL 级，这个过程称为分叉。VAX/VMS 操作系统为处理这个过程，采用了分叉进程的概念。所谓驱动器分叉进程是动态地立的，并且有最小的关联。驱动器分叉进程的关联是：寄存器 $R_3$、$R_4$；部件控制块 JCB 的一个数据结构，这个数据结构的始地由 $R_5$ 指出；驱动器中将执行的下一条指令地址(PC)；分叉进程代码是在 S0 空间中。像共文进程一样，驱动器分叉进程也能够挂起和被中断。

这样，驱动器分叉进程是动态地立具有最小关联的执行仪表，以设备部件的名义执行驱动器代码，当驱动器分叉进程请求一个不可用资源时，将驱动器分叉进程放入到等待队列中，等待队列的元素就是 UCB，而 UCB 中存入了分叉进程关联，当驱动器分叉进程由于资源可用重新执行时，仍在原来的 IPL 级上执行，放入等待队列是不忘预的活动，当 I/O 请求排队到忙的设备部件时，即进入到驱动器的例行程序时，为了降低 IPL 把驱动器分叉进程放入到分叉队列中，并且不包括资源的分配，分叉队列的元素也是 UCB，而 UCB 中存入了分叉进程关联，放入分叉队列是忘预的活动。

分叉队列是先进先出队列，每个分叉 IPL 一个分叉队列，分是 IPL 为 6，9，10 和 11，它是等待由分叉分派程序执行的驱动器分叉进程的队列，为降低 IPL 而放入到分叉队列中。

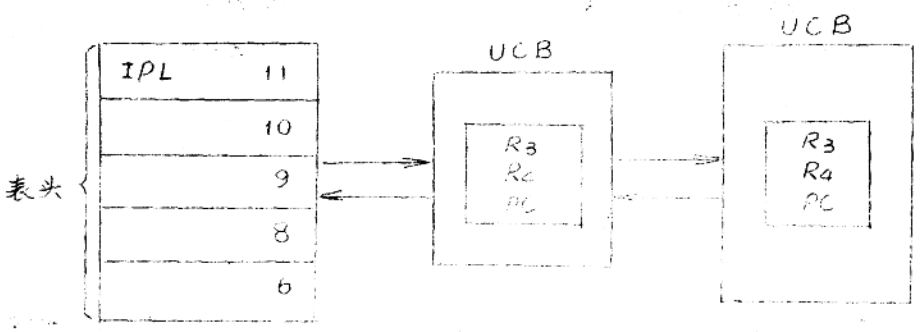
等待队列也是先进先出队列，它是等待一个特殊资源的驱动器分叉进程的队列，当资源变成可用时，重新启动队列中驱动器分叉进程（不包括分叉分派程序）。

分叉的分派：

当驱动器分叉时，分叉进程降低 IPL，$R_3$，$R_4$ 和 PC 存入到部件控制块 UCB 中，将 UCB 排队到合适的 IPL 级的分叉队列中，并用该 IPL 级请求一个软件中断，当 CPU 的 IPL

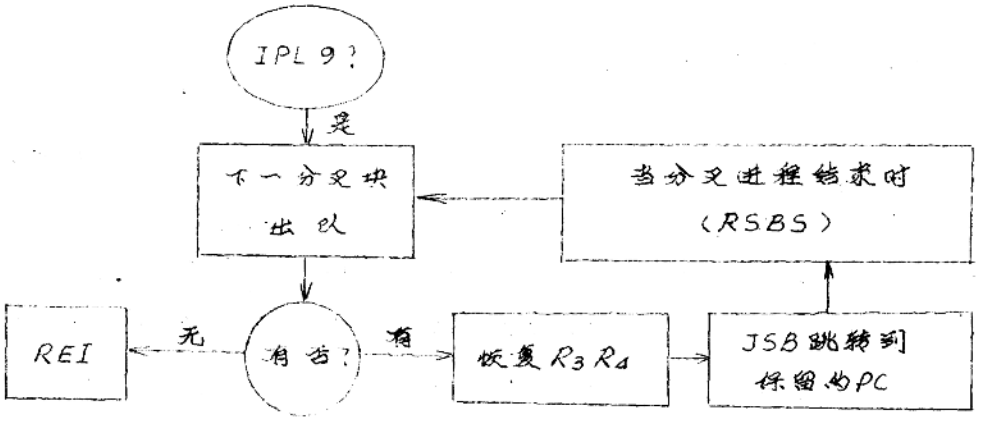
降低到所请求的软件中断IPL级以下时，响应该软件中断，此时的软件中断服务程序就是分叉分派例行程序，分叉分派例行程序恢复驱动器分叉进程关联，使得驱动器分叉进程在降低的IPL级上运行。

分叉队列的结构如图八。




图八　分叉队列

分叉分派程序的控制流图如图九。




图九．分叉分派程序的通常逻辑流图

## §2. I/O 数据结构（数据基）

设备驱动器中要用到的主要的数据结构有如下几个：

I/O 请求包 IRP，　　适配器控制块 ADP

部件控制块 UCB，　　驱动器调度表 DDT

通道请求块 CRB，　　功能决定表 FDT

通道控制块 CCB，　　驱动器序幕表 DPT

中断数据块 IDB，

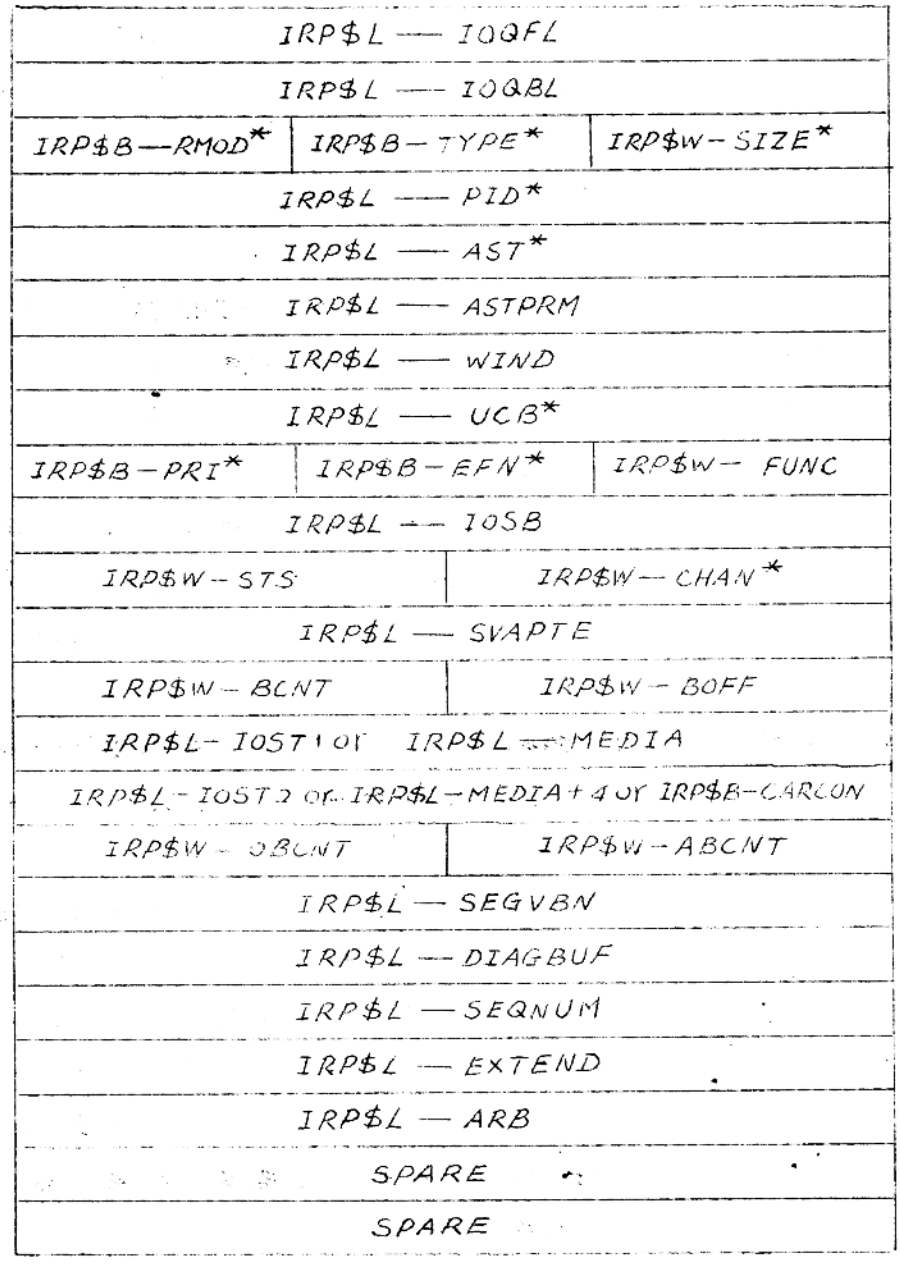
设备数据块 DDB

下面我们描述上述的数据结构，给出它们的某些域，并介绍它们之间的逻辑关系，驱动器对称有"*"号的域只能进行读操作，而标有"Spare"或"Unused"的域除被说明外，一般认为是留给 DIGITAL 公司将来使用的。

### 1. I/O请求包（IRP）

当用户进程使用 $ QIO 系统服务，以排队一个 I/O 请求时，由 $QIO 系统服务在非分页动态系统空间中建立 IRP，它包含了描述当前 I/O 请求的所有信息，并在 I/O 请求完成后充当异步系统自陷（AST）控制块。它由两部分组成，功能/设备相关段和功能/设备无关段。它依照 I/O 请求进程的基本优先级到该设备的等待 I/O 请求队列中排队，在 I/O 请求完成时，状态信息也保存在 IRP，以返回给发 I/O 请求的进程的 I/O 状态块 IOSB。

该数据结构如图十。


IRP$L — IOQFL
IRP$L — IOQBL
IRP$B—RMOD*
IRP$B—TYPE*
IRP$W—SIZE*
IRP$L — PID*
IRP$L — AST*
IRP$L — ASTPRM
IRP$L — WIND
IRP$L — UCB*
IRP$B—PRI*
IRP$B—EFN*
IRP$W— FUNC
IRP$L — IOSB
IRP$W—STS
IRP$W—CHAN*
IRP$L — SVAPTE
IRP$W—BCNT
IRP$W—BOFF
IRP$L—IOST1 OR IRP$L—MEDIA
IRP$L—IOST2 OR IRP$L—MEDIA+4 OR IRP$B—CARCON
IRP$W—OBCNT
IRP$W—ABCNT
IRP$L — SEGVBN
IRP$L — DIAGBUF
IRP$L — SEQNUM
IRP$L — EXTEND
IRP$L — ARB
SPARE
SPARE

图十　I/O 请求包（IRP）

有关域的内容描述

IRP$W-SIZE* —— I/O 请求包的尺寸，当 EXE$QIO 分配并填写 I/O 包时把符号常号 IRP$C-LENGTH 写入该域。

IRP$B-TYPE* —— 控制块的类型，当 EXE$QIO 分配并填写 I/O 包时把符号常号 DYN$C-IRP 写入该域。

IRP$L-PID* —— 发出 I/O 请求的进程的说明，EXE$QIO 从 PCB 获得这个说明并写入该域。

IRP$L-AST* —— 在用户请求中说明的 AST 例程地址，由 EXE$QIO 把域地址写入该域，在 I/O 善后处理期间，如果该域包含了一个 AST 例程地址，该方式 AST 例程排队用户方式 AST 到请求的进程。

IRP$L-WIND —— 窗口控制块地址，如果 I/O 请求中说明是访问一个文件结构设备，EXE$QIO 写该域，ACP 读该域。

IRP$L-UCB* —— 设备单元进程 I/O 通道的 UCB 地址，EXE$QIO 从 CCB 拷贝该值。

IRP$W-FUNC —— 说明执行 I/O 请求的功能码。EXE$QIO 与驱动器 FDT 例程把交叉映射最基本的级（虚拟的 → 逻辑的 → 物理的）并把值写入该域，该功能码的 6 位描述了基本的功能，余下的 10 位修改功能。

IRP$B-PRI* —— 当发出 I/O 请求时进程的基本优先级，EXE$QIO 从 PCB 获得该域的值。

IRP$L-LOSB —— 进程 I/O 状态块的虚拟地址。

IRP$W-CHAN* —— 进程 I/O 通道的号码。

IRP$W-STS —— I/O 请求的状态，EXE$QIO 把该域予置为 0。

0，EXE$QIO，FDT 例程和驱动器分支进程根据 I/O 请求的当前状态修改该域，I/O 善后处理读该域以决定要作什么样的善后处理，该域的各位描述了 I/O 功能的类型。

IRP$L－SVAPTE——为直接 I/O 操作说明 I/O 传输缓冲区的第一个页表入口的虚存地址，FDT 例程为直接 I/O 传输在内存封锁页并写该域，在把控制传给启动 I/O 例程以前，IOC$INITIATE 把该域拷贝到 UCB 的域 UCB$－SVADTE 中。

IRP$W－BOFF ——一个直接 I/O 传输的第一页的字节偏移量。FDT 例程计算该值并写入该域。对缓冲的 I/O 操作，FDT 例程在该域中写入传输的字节数，IOC$INTTIATE 在调用启动 I/O 例程前把该域拷贝到 UCB 的域 UCB$W－BOFF 中。

IRP$W－BCNT ——I/O 传输的字节计数，FDT 例程计算该值并写入该域 IOC$LNITIATE 在调用启动 I/O 例程前把该域拷贝到 UCB 的域 UCB$W－BCNT 中，对缓冲 I/O 类功能，I/O 善后处理用该域来决定有多少字节要写入用户缓冲区。

IRP$L－MEDIA ——第一个 I/O 状态长字，IOC$REQCOM 和 EXE$FINISHIO(C)把 R0 的内容写入该域，I/O 善后处理例程把该域的内容拷贝到用户 I/O 状态块，EXE$ZEROPARM 把 0 写入该域，而 EXE$ONEPARM 把 P1 写入该域。

IRP$L－MEDIA+4 ——第二个 I/O 状态长字，IOC$REQCOM 和 EXE$FINISHIO(C)把 R1 的内容写入该域，I/O 善后处理程序把该域的内容拷贝到用户

I/O 状态块。EXE$READ和EXE$WRITE把用户的I/O请求中的P4拷贝到该域。

IRP$L-DIAGBUF*——系统地址空间内的诊断缓冲区的地址。EXE$QIO在系统地址空间内分配诊断缓冲区。

IRP$L-SEQNUM*—— I/O处理序列号。

IRP$B-RMOD* ——在I/O请求时的进程访问方式，EXE$QIO由PSL获得处理机的访问方式，并将该值拷贝到这个域中。

IRP$L-ASTPRM ——由用户在I/O请求时所规定的AST例行程序所需的参数地址。如果该进程规定了AST例行程序和参数，EXE$QIO将参数地址写入该域中。如果，IRP$L-AST 域包括了地址，在I/O事后处理时，该方式AST例程排队一个用户方式AST，并且把IRP$L-ASTPRM的值作为到AST例行程序作为变元。

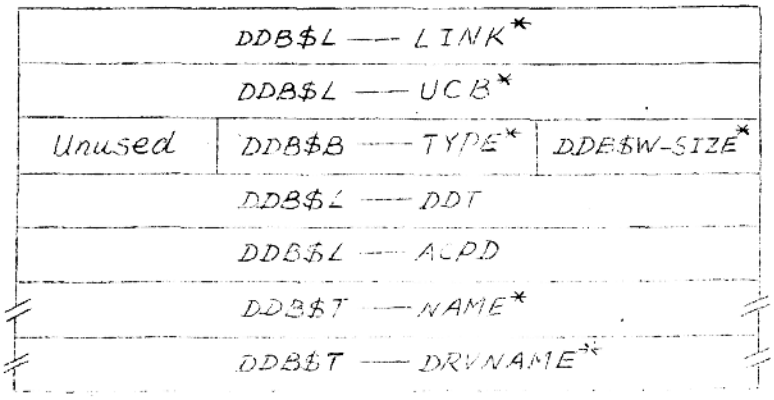
IRP$B-EFN* ——在I/O请求时规定的事件标志号和组号，如果I/O请求没有规定事件标志号EXE$QIO使用缺省值0。EXE$QIO写该域，当I/O操作完成时I/O事后处理例行程序调用SCH$POSTEF置该事件标志号。

还有几个域的内容，这里就不列出了。详细地描述见写一个设备驱动器指南的附录A。

## 2．设备数据块（DDB）

DDB是一个可变长度的块，它说明了一般的设备/控制器名和连种单个控制器的一组设备的驱动器名，该块由驱动器装入过程建立，每个控制器均有一个DDB，它包含连接于某个控制

器的相同类型的所有设备的信息及指向共它数据结构（UCB DDT）的指针。该DDB的所有域由由驱动器装入程序填写，VAX/VMS例程和驱动器访问该DDB。下图说明了DDB的域：




各个域的内容描述：

DDB$L—LINK*——下一个DDB的地址，如为0则说明该DDB是链表中最后的DDB。

DDB$L—UCB*——接于控制器的第一个部件的UCB地址。

DDB$B—TYPE*——控制块的类型，驱动器装入过程在建立该DDB时把常量DYN$L—DDB写入该域。

DDB$L—DDT——DDT的地址，每个设备驱动器的DPT必须为该域说明一个值。

DDB$L—ACPD——对控制器有缺省ACP名，如果设备是文件结构的，该域的主位字节包含结构式文件设备的类型，DPT必须为该域说明一个值。

DDB$T—NAME*——接于控制器的设备通用名

DDB$T—DRVNAME*—设备驱动器名，可以从DPT中获得。

图十一描述了DDB与共它数据结构的关系：