

#36: Drive Queue Elements

See also: The File Manager
 The Device Manager

Written by: Bryan Stearns June 12, 1985
Updated: March 1, 1988

This note expands on *Inside Macintosh*'s definition of the drive queue, which is given in the File Manager chapter.

As shown in *Inside Macintosh*, a drive queue element has the following structure:

```
DrvQEl = RECORD
    qLink:    QElemPtr; {next queue entry}
    qType:    INTEGER;  {queue type}
    dQDrive:  INTEGER;  {drive number}
    dQRefNum: INTEGER;  {driver reference number}
    dQFSID:   INTEGER;  {file-system identifier}
    dQDrvSz:  INTEGER;  {number of logical blocks on drive}
    dQDrvSz2: INTEGER;  {additional field to handle large drive size}
END;
```

Note that dQDrvSz2 is only used if qType is 1. In this case, dQDrvSz2 contains the high-order word of the size, and dQDrvSz contains the low-order word.

Inside Macintosh also mentions four bytes of flags that precede each drive queue entry. How are these flags accessed? The flags begin 4 bytes before the address pointed to by the DrvQElPtr. In assembly language, accessing this isn't a problem:

```
MOVE.L  -4(A0),D0    ;A0 = DrvQElPtr; get drive queue flags
```

If you're using Pascal, it's a little more complicated. You can get to the flags with this routine:

```
FUNCTION DriveFlags(aDQEPtr: DrvQElPtr): LONGINT;

VAR
    flagsPtr : ^LONGINT; {we'll point at drive queue flags with this}

BEGIN
    {subtract 4 from the DrvQElPtr, and get the LONGINT there}
    flagsPtr := POINTER(ORD4(aDQEPtr) - 4);
    DriveFlags := flagsPtr^;
END;
```

From MPW C, you can use:

```
long DriveFlags(aDQEPtr)
DrvQElPtr      aDQEPtr;

{ /* DriveFlags */
    return(*((long *)aDQEPtr - 1)); /* coerce flagsPtr to a (long *)
                                     so that subtracting 1 from it
                                     will back us up 4 bytes */
} /* DriveFlags */
```

Creating New Drives

To add a drive to the drive queue, assembly-language programmers can use the function defined below. It takes two parameters: the driver reference number of the driver which is to “own” this drive, and the size of the new drive in blocks. It returns the drive number created. It is vital that you **not** hard-code the drive number; if the user has installed other non-standard drives in the queue, the drive number you’re expecting may already be taken. (Note that the example function below arbitrates to find an unused drive number, taking care of this problem for you. Also, note that this function doesn’t mount the new volume; your code should take care of that, calling the Disk Initialization Package to reformat the volume if necessary).

```
AddMyDrive  PROC      EXPORT
;-----
;FUNCTION AddMyDrive(drvSize: LONGINT; drvRef: INTEGER): INTEGER;
;-----
;Add a drive to the drive queue. Returns the new drive number, or a negative
;error code (from trying to allocate the memory for the queue element).
;-----
DQESize      EQU        18          ;size of a drive queue element
;We use a constant here because the number in SysEqu.a doesn't take into
;account the flags LONGINT before the element, or the size word at the end.
;-----
StackFrame   RECORD      {link},DECR
result       DS.W         1          ;function result
params       EQU          *
drvSize      DS.L         1          ;drive size parameter
drvRef       DS.W         1          ;drive refNum parameter
paramSize    EQU          params-*
return       DS.L         1          ;return address
link         DS.L         1          ;saved value of A6 from LINK
block        DS.B         ioQElSize ;parameter block for call to MountVol
linkSize     EQU          *
            ENDR
;-----
            WITH      StackFrame      ;use the offsets declared above

            LINK      A6,#linkSize    ;create stack frame

;search existing drive queue for an unused number

            LEA        DrvQHdr,A0      ;get the drive queue header
            MOVEQ      #4,D0           ;start with drive number 4
```

```

CheckDrvNum
    MOVE.L    qHead(A0),A1    ;start with first drive
CheckDrv
    CMP.W     dqDrive(A1),D0  ;does this drive already have our number?
    BEQ.S     NextDrvNum      ;yep, bump the number and try again.
    CMP.L     A1,qTail(A0)    ;no, are we at the end of the queue?
    BEQ.S     GotDrvNum       ;if yes, our number's unique! Go use it.
    MOVE.L     qLink(A1),A1    ;point to next queue element
    BRA.S     CheckDrv        ;go check it.

```

```

NextDrvNum
    ;this drive number is taken, pick another

    ADDQ.W     #1,D0           ;bump to next possible drive number
    BRA.S     CheckDrvNum      ;try the new number

```

```

GotDrvNum
    ;we got a good number (in D0.W), set it aside

    MOVE.W     D0,result(A6)   ;return it to the user

    ;get room for the new DQE

    MOVEQ      #DQESize,D0     ;size of drive queue element, adjusted
    _NewPtr    sys             ;get memory for it
    BEQ.S      GotDQE          ;no error...continue
    MOVE.W     D0,result(A6)    ;couldn't get the memory! return error
    BRA.S      FinishUp        ;and exit

```

```

GotDQE
    ;fill out the DQE

    MOVE.L     #$80000,(A0)+    ;flags: non-ejectable; bump past flags

    MOVE.W     #1,qType(A0)     ;qType of 1 means we do use dqDrvSz2
    CLR.W      dqFSID(A0)       ;"local file system"
    MOVE.W     drvSize(A6),dqDrvSz2(A0) ;high word of number of blocks
    MOVE.W     drvSize+2(A6),dqDrvSz(A0) ;low word of number of blocks

    ;call AddDrive

    MOVE.W     result(A6),D0     ;get the drive number back
    SWAP       D0               ;put it in the high word
    MOVE.W     drvrRef(A6),D0    ;move the driver refNum in the low word
    _AddDrive                      ;add this drive to the drive queue

```

```

FinishUp
    UNLK       A6               ;get rid of stack frame
    MOVE.L     (SP)+,A0          ;get return address
    ADDQ       #paramSize,SP    ;get rid of parameters
    JMP        (A0)             ;back to caller

```

```

;-----
ENDPROC

```