

-- DiskDefs.Mesa Edited by Sandman on August 24, 1977 9:23 AM

## DIRECTORY

AltoDefs: FROM "altodefs",  
AltoFileDefs: FROM "altofiledefs";

DEFINITIONS FROM AltoFileDefs, AltoDefs;

DiskDefs: DEFINITIONS = BEGIN

-- standard disk

nDisks: CARDINAL = 1;  
nHeads: CARDINAL = 2;  
nTracks: CARDINAL = 203;  
nSectors: CARDINAL = 12;

-- physical disk address

DA: PRIVATE TYPE = MACHINE DEPENDENT RECORD [  
sector: [0..17B],  
track: [0..777B],  
head, disk: [0..1],  
restore: [0..1]];

-- DAs with special meaning

InvalidDA: DA = DA[17B,777B,1,1,1];

-- disk header

DH: TYPE = MACHINE DEPENDENT RECORD [  
packID: CARDINAL,  
diskAddress: DA];

-- file identifier

FID: TYPE = MACHINE DEPENDENT RECORD [  
version: CARDINAL,  
serial: SN];

-- disk label

DL: TYPE = MACHINE DEPENDENT RECORD [  
next, prev: DA,  
blank: UNSPECIFIED,  
bytes: CARDINAL,  
page: CARDINAL,  
fileID: FID];

-- disk final status

DFS: PRIVATE TYPE = {  
CommandComplete, HardwareError,  
CheckError, IllegalSector};

-- disk status word

DS: PRIVATE TYPE = MACHINE DEPENDENT RECORD [  
sector: [0..17B],  
done: [0..17B],  
seekFailed: [0..1],  
seekInProgress: [0..1],  
notReady: [0..1],  
dataLate: [0..1],  
noTransfer: [0..1],  
checksumError: [0..1],  
finalStatus: DFS];

-- useful status configurations

DSfree: CARDINAL = 1; DSfake: CARDINAL = 3; DSDone: CARDINAL = 17B;  
DSmaskStatus: DS = DS[0,DSDone,1,0,1,1,0,1,LAST[DFS]];  
DSgoodStatus: DS = DS[0,DSDone,0,0,0,0,0,0,CommandComplete];  
DSfakeStatus: DS = DS[0,DSfake,0,0,0,0,0,0,CommandComplete];  
DSfreeStatus: DS = DS[0,DSfree,0,0,0,0,0,0,CommandComplete];

-- disk subcommands

DSC: PRIVATE TYPE = {DiskRead, DiskCheck, DiskWrite};

-- hardware disk command

DC: PRIVATE TYPE = MACHINE DEPENDENT RECORD [  
seal: BYTE,  
header, label, data: DSC,  
seek, exchange: [0..1]];

```
CBptr: TYPE = POINTER TO CB;
```

```
-- disk command block (label, page, and zone added)
```

```
CB: TYPE = PRIVATE MACHINE DEPENDENT RECORD [
  nextCB: POINTER TO CB,
  status: DS,
  command: DC,
  headerAddress: PUBLIC POINTER TO DH,
  labelAddress: PUBLIC POINTER TO DL,
  dataAddress: PUBLIC POINTER,
  normalWakeups: WORD,
  errorWakeups: WORD,
  header: PUBLIC DH,
  label: PUBLIC DL,
  page: PUBLIC CARDINAL,
  zone: PUBLIC POINTER TO CBZ];
```

```
nCB: CARDINAL = 3; -- minimum for full disk speed
ICBZ: CARDINAL = SIZE[CBZ]+nCB*(SIZE[CB]+SIZE[CBptr]);
```

```
-- Note: if there are n CBs, there are n+1 entries in the
-- cbQueue (an extra one contains a NIL to mark the end).
-- The extra one is represented by queueVec: ARRAY [0..1)
-- and thus is included in SIZE[CBZ].
```

```
CBZptr: TYPE = POINTER TO CBZ;
```

```
CBZ: TYPE = PRIVATE MACHINE DEPENDENT RECORD [
  checkError: PUBLIC BOOLEAN,
  errorCount: PUBLIC [0..77777B],
  info: PUBLIC POINTER,
  cleanup: PUBLIC PROCEDURE[CBptr],
  errorDA: PUBLIC vDA,
  currentPage: PUBLIC CARDINAL,
  currentBytes: PUBLIC CARDINAL,
  normalWakeups: WORD,
  errorWakeups: WORD,
  cbQueue: DESCRIPTOR FOR ARRAY OF CBptr,
  qHead, qTail: CARDINAL,
  queueVec: ARRAY [0..1) OF CBptr;
  -- the queue vector starts at queueVec.
  -- after the queue vector there follows
  -- ARRAY OF CB, the CBs for the zone.
```

```
-- Procedures in DiskIO
```

```
RealDA: PROCEDURE [v:vDA] RETURNS [DA];
VirtualDA: PROCEDURE [da:DA] RETURNS [vDA];
```

```
SetDisk: PROCEDURE [POINTER TO DISK];
GetDisk: PROCEDURE RETURNS [POINTER TO DISK];
ResetDisk: PROCEDURE RETURNS [POINTER TO DISK];
```

```
ResetWaitCell: PROCEDURE;
SetWaitCell: PROCEDURE [POINTER TO WORD] RETURNS [POINTER TO WORD];
```

```
DDC: TYPE = RECORD [
  cb: CBptr,
  ca: POINTER,
  da: vDA,
  page: PageNumber,
  fp: POINTER TO FP,
  restore: BOOLEAN,
  action: vDC];
```

```
DoDiskCommand: PROCEDURE [arg:POINTER TO DDC];
```

```
RetryCount: CARDINAL = 8;
RetryableDiskError: SIGNAL [cb:CBptr];
UnrecoverableDiskError: SIGNAL [cb:CBptr];
```

```
CBinit: TYPE = {clear,dontClear};
```

```
InitializeCBstorage: PROCEDURE [
```

```
    zone:CBZptr, nCBs:CARDINAL, page:PageNumber, init:CBinit];
GetCB: PROCEDURE [zone:CBZptr, init:CBinit] RETURNS [cb:CBptr];
CleanupCBqueue: PROCEDURE [zone:CBZptr];
DiskCheckError: SIGNAL [page:PageNumber];
DiskRequestOption: TYPE = {swap, update, extend};
DiskRequest: TYPE = RECORD [
    ca: POINTER,
    da: POINTER TO vDA,
    firstPage: PageNumber,
    lastPage: PageNumber,
    fp: POINTER TO FP,
    fixedCA: BOOLEAN,
    action, lastAction: vDC,
    signalCheckError: BOOLEAN,
    option: SELECT COMPUTED DiskRequestOption FROM
        swap => [desc: POINTER TO DiskPageDesc],
        update => [cleanup: PROCEDURE[CBptr]],
        extend => [lastBytes: CARDINAL],
        ENDCASE];
DiskPageDesc: TYPE = RECORD [
    prev, this, next: vDA,
    page: PageNumber,
    bytes: CARDINAL];
SwapPages: PROCEDURE [arg:POINTER TO swap DiskRequest]
    RETURNS [page:PageNumber, byte:CARDINAL];
END.
```