

WD.L 5.31.85
(ROM)

```
/*  
 * Driver for Western Digital hard disk controller for the Z8000HR.  
 *  
 * Uses Commodore's SASI-like command block structure.  
 */
```

```
extern char tryaboot; /* <> 0 => auto-boot in progress */
```

```
#ifndef CMDBLKPADDR  
#define CMDBLKPADDR (0x02000000L) /* WD command blk phys addr */  
#endif  
#ifndef BUFPADDR  
#define BUFPADDR (0x02000400L) /* CMDBLKPADDR + 1K */  
#endif
```

```
#include <coherent.h>  
#include <buf.h>  
#include <con.h>  
#include <stat.h>  
#include <uproc.h>  
#include <errno.h>
```

```
int wdlload();  
int wdread();
```

```
/*  
 * Western Digital Controller port addresses  
 */
```

```
#define WDIO 0x0500 /* Data Register (word mode) */  
  
#define WDTDR 0x00 /* test drive ready */  
#define WDREST 0x01 /* restore to cyl 0 */  
#define WDRSS 0x03 /* Request status */  
#define WDCTF 0x05 /* check track format */  
#define WDFMTT 0x06 /* format track */  
#define WDREAD 0x08 /* Read */  
#define WDWRITE 0x0A /* Write */  
#define WSDP 0x0C /* set drive parameters */  
#define WCCBA 0x0F /* change command block address */  
#define WDDIAG 0xE3 /* run drive diagnostics */  
#define WDCDIAG 0xE4 /* run controller diagnostics */  
  
#define NSEC 17 /* number of sectors per track */  
#define NTRK 4 /* number of tracks per cylinder */  
#define NCYL 306 /* number of cylinders */  
#define SIXTEENUS 0xF /* sixteen microsecond step rate */
```

```
typedef struct wdcmd { /* command block layout */  
    unsigned char c_opcode; /* command class & opcode */  
    unsigned char c_lunhiaddr; /* lun [7:5] & sector addr [4:0] */  
    unsigned char c_midaddr; /* middle sector address */  
    unsigned char c_lowaddr; /* low part of sector address */  
    unsigned char c_blockcnt; /* number of blocks in I/O */  
    unsigned char c_control; /* reserved control byte */  
    unsigned char c_highdma; /* high DMA addr (phys segment) */  
    unsigned char c_middma; /* middle DMA address */  
    unsigned char c_lowdma; /* low DMA address */  
    unsigned char c_rsvd1; /* reserved */  
    unsigned char c_rsvd2; /* reserved */  
    unsigned char c_rsvd3; /* reserved */  
    unsigned char c_errorbits; /* error information */  
    unsigned char c_lunladd2; /* error lun and high sector addr */  
    unsigned char c_ladd1; /* error middle address */
```

```

} WDCMD;

typedef struct wd_idc_params {           /* drive initialization params */
    unsigned char    p_optstep;         /* options & step rate */
    unsigned char    p_headhicyl;      /* heads <6:4>, hi cyl count <3:0> */
    unsigned char    p_cyl;            /* low byte of cyl count */
    unsigned char    p_precomp;        /* precomp cyl / 16 */
    unsigned char    p_reduce;         /* reduced write current / 16 */
    unsigned char    p_sectors;        /* sectors per track */
} WDINFO;

WDINFO wdinfo[] = {                    /* add entries as needed */
    { SIXTEENuS, (2<<4)|(612/256), 612%256, 128/16, 128/16, NSEC },
    { SIXTEENuS, (4<<4)|(306/256), 306%256, 128/16, 128/16, NSEC },
    { SIXTEENuS, (4<<4)|(612/256), 612%256, 128/16, 128/16, NSEC }
};

int      typeno = 2;                    /* drive type selected from table */
int      errors;                       /* total # of errors */

/*
 * set up the controller and drive as needed.
 */
wdload()
{
    errors = 0;
    wdsetparam();
}

wdwait()
{
    register long l = 1000000L;
    register WDCMD *cbp;
    register unsigned i;

    out(WDIO, 1);                       /* strobe I/O line to WD */

    cbp = (WDCMD *)CMDBLKPADDR;
    do {
        i = cbp->c_errorbits;
        --l;
    } while ((i&0xFF) == 0xFF && l);
    if (!l) {
        if (tryaboot == 0) puts("\nTIMEOUT ERROR\n");
        ++errors;
    }

    out(WDIO, 0);                       /* reset IEO on DMA chip */
}

/*
 * read a block off of the WD disk device specified.
 * return the number of blocks read.
 */
wdread(lun, bn, addr)
int      lun;                          /* unit number */
long     bn;                            /* block # */
char     *addr;                         /* destination address */
{
    register int i;
    register WDCMD *cbp;
    register unsigned char *cp;

```

```

for (cp = cbp, i = 0; i < sizeof(WDCMD); ++i)
    cp[i] = 0;
cbp->c_opcode = WDREAD; /* read a sector */
cbp->c_blockcnt = 1;
cbp->c_highdma = (BUFPADDR >> 24);
cbp->c_middma = (BUFPADDR >> 8);
cbp->c_lowdma = 0x00;
cbp->c_lunhiaddr = (lun << 5) | (bn >> 16);
cbp->c_midaddr = bn >> 8;
cbp->c_lowaddr = bn;
cbp->c_errorbits = 0xff; /* make ready */
wdwait(); /* wait for i/o complete */

if (cbp->c_errorbits & 0x7F) {
    if (tryaboot == 0) puts("read error\n");
    ++errors;
}
ppcopy(BUFPADDR, addr, 512);
return (errors == 0);
}

```

```

/*
 * set the drive and controller parameters for the WD hard disk.
 * this uses the the values in the parameter table in this file.
 * these have to match the order and type of those found in rom.c ...
 */

```

```

wdsetparam()
{
    register WDCMD *cbp;
    register unsigned char *cp;
    register int i;
    register WDINFO *ip;

    cbp = (WDCMD *) CMDBLKPADDR;
    for (cp = cbp, i = 0; i < sizeof(WDCMD); ++i)
        cp[i] = 0;
    cbp->c_opcode = WSDSP; /* set drive params */
    cbp->c_blockcnt = 0;
    cbp->c_highdma = (BUFPADDR >> 24);
    cbp->c_middma = (BUFPADDR >> 8);
    cbp->c_lowdma = 0x00;
    cbp->c_lunhiaddr = 0;
    cbp->c_midaddr = 0;
    cbp->c_lowaddr = 0;
    cbp->c_errorbits = 0xff; /* make ready */
    ip = (WDINFO *)BUFPADDR;
    *ip = wdinfo[typeno];
    wdwait(); /* wait for i/o complete */

    if (cbp->c_errorbits & 0x7F) {
        if (tryaboot == 0)
            puts("controller/drive initialization failed\n");
        ++errors;
    }
}

```