HR.C
5.31.85

```
/*
 * Commodore Z8000-HR console
 */
#include        <coherent.h>
#include        <con.h>
#include        <io.h>
#include        <uproc.h>
#include        <errno.h>
#include        <sched.h>
#include        <sgtty.h>
#include        <signal.h>
#include        <stat.h>
#include        <rico.h>
#include        "hr.h"


/*
** ZCIO1 base address
*/
#define ZCIO1   0x0000
#define PKBD    0x0205  /* keyboard enable port         */
#define KBDEN   0x02    /* keyboard enable bit          */
#define PC2     0x04
#define PC3     0x08

/*
 * Control register addresses
 */
#define MICR    0x01    /* master interrupt             */
#define MCCR    0x03    /* master configuration         */

/*
 * Register addresses for Port A
 */
#define PACAS   0x11    /* port a command and status    */
#define PAMS    0x41    /* port a mode specification    */
#define PAHS    0x43    /* port a handshake spec        */
#define PADPP   0x45    /* port a data path polarity    */
#define PADD    0x47    /* port a data direction        */
#define PASIOC  0x49    /* port a special io control    */
#define PAPP    0x4b    /* port a pattern palarity      */
#define PAPT    0x4d    /* port a pattern transition    */
#define PAPM    0x4f    /* port a patterm mask          */
#define PAIV    0x05    /* port a interrupt vector      */
#define PADATA  0x1b    /* port a data                  */

/*
 * Register addresses for Port C
 */
#define PCDD    0x0d    /* port c data direction        */
#define PCSIOC  0x0f    /* port c special io control    */
#define PCDPP   0x0b    /* port c data path polarity    */
#define PCDATA  0x1f    /* port c data                  */

/*
 * Misc hardware flags and values
 */
#define MIE     0x80    /* master interrupt enable      */
#define PAE     0x04    /* port a enable                */
#define C_IPIUS 0x20    /* clear IP&IUS command         */
```

```
#define HRVECTOR 8      /* port a interrupt vector      */
#define LASTKEY 0xfe


hrload( )
{
        register struct mbuf    *p;
        register struct client  *cp;
        register struct evmgr   *ev;
        int     s;
        void    hrmouse( ),
                hrkey();

        s = sphi();
        setivec(HRVECTOR, hrkey);
        outb(ZCIO1+PAMS,  0x1a);
        outb(ZCIO1+PAHS,  0x00);
        outb(ZCIO1+PADPP, 0x00);
        outb(ZCIO1+PADD,  0xff);
        outb(ZCIO1+PASIOC,0x00);
        outb(ZCIO1+PAPP,  0x80);
        outb(ZCIO1+PAPT,  0x00);
        outb(ZCIO1+PAPM,  0x80);
        outb(ZCIO1+PAIV,  HRVECTOR);
        outb(ZCIO1+PACAS, 0xc0);
        outb(ZCIO1+MICR, inb(ZCIO1+MICR)|MIE);
        outb(ZCIO1+MCCR, inb(ZCIO1+MCCR)|PAE);
        outb(PKBD, KBDEN);
        outb(ZCIO1+PCDATA, 0);          /* toggle pc3   */
        outb(ZCIO1+PCDATA, PC3);
        spl(s);
}


hruload()
{
        printf("BYE FROM HRdrv\n");
        outb(ZCIO1+MCCR, inb(ZCIO1+MCCR) & ~PAE);
        clrivec(HRVECTOR);
}


void
hrkey()
{
        register struct mbuf    *p;
        register uint   key;
        int     i;

        /*
         * Clear interrupt condition in chip and read data from PA and PC3
         */
        outb(ZCIO1+PACAS, C_IPIUS);
        key = inb(ZCIO1+PADATA) & 0x7f;
        if ( inb(ZCIO1+PCDATA) & PC2 )
                key += 0x80;
/*
        if ( hrsmgr != NOEVMGR && (p = mbufp) )
        {
                mbufp = p->mb_next;
                hrlink(hrsmgr, p);
                wakeup(&client[hrsmgr].head);   /* should catch all weird cas
```

```
                p->mb_m.m_msg[0] = SM_KKEY;
                p->mb_m.m_msg[1] = hrticks;
                p->mb_m.m_msg[2] = key;
        }
*/
        outb(ZCIO1+PCDATA, 0);
        outb(ZCIO1+PCDATA, PC3);
}
```