```
#include <stdio.h>
#include <tos.h>
#include <string.h>
#include <aes.h>
#include "kbd.h"

/*****************************************************************************/

#define SS_DOLLAR_NORMAL 1                      -1
                                                                          10
#define INVALID_PTR     -1L"

#define KBD_TYPE        "US"        /* Or "GY"                  */

#define RS232_VEC_ID    76          /* $130/4                   */

#define KBD_OS_JUMP     (0xfc3d5eL)  /* Entry KBD-interrupthandler      */   KBD_OS_JUMP
#define RS232_OS_JUMP   (0xfc3426L)  /* Entry RS232-output              */   RS232_OS_JUMP

#define SYSHDR_PTR      (0x4f2L)     /* Ptr to Sysheaderstructure       */   SYSHDR_PTR
#define RWABS_VEC       (0X476L)     /* Rwabs-vector                    */   RWABS_VEC

#define RCV_STAT_REG    (0xfffffa2bL) /* Receiver-Status-register       */   RCV_STAT_REG
#define DATA_REG        (0xfffffa2fL) /* USART-Data-register            */   DATA_REG

/*****************************************************************************/

#define INSTALL_DIALOG    \
\
"[0][ |KBD V1.4a by Bert Oberholz '91 |" \                                30
\
"     |       suosws::oberholz         |" \
  " oberholz@suosws.enet.dec.com   | ]"\
\
    "[Install|Quit]"

#define INSTALL_WARN_DIALOG    \
\
"[1][ XBRA-chain interrupted         | " \
 "  | KBD V1.4a by Bert Oberholz '91 " \                                   40
\
 "|       suosws::oberholz         |" \
 " oberholz@suosws.enet.dec.com   ]"\
\
    "[Install|Quit]"

#define REMOVE_DIALOG    \
\
"[0][ |KBD V1.4a by Bert Oberholz '91 |" \
\                                                                         50
"     |       suosws::oberholz         |" \
  " oberholz@suosws.enet.dec.com   | ]"\
\
    "[Remove|Keep]"

#define REMOVE_BUTTON  1
#define INSTALL_BUTTON 1
```

```
#define QUIT_BUTTON    2
#define NO_AES         0*
                                                                                60
/*************************************************************************/

#define STR4EQU(a,b)     (strncmp(a,b,4)==0) /* spec. comparision for XBRA */      strncmp

#define BEGIN_ROM        (0xfc0000L)                                          BEGIN_ROM
#define END_ROM          (0xefffffL)                                            END_ROM

#define IN_ROM(a)        (((a) >= BEGIN_ROM && (a) <= END_ROM) ? 1 : 0)         IN_ROM

/*************************************************************************/                70

#define BOOT_KEY         DEC_F_20*

/*************************************************************************/

#define LEARN_KEY        DEC_DO
#define LEARN_BUF_SIZE   10                    /* do not increase  */
#define LEARN_IN_PROG    1
#define LEARNED          2
#define NO_LEARN         3*                                                     80

/*************************************************************************/

#define LED_ON           0x13
#define LED_OFF          0x11

#define WAIT_LED         0x81
#define ALT_LED          0x82
#define CAPS_LED         0x84
#define HOLD_LED         0x88                                                   90

#define SOUND_BELL       0xA7*

/*************************************************************************/
/* Global pointer declarations                                           */
/*************************************************************************/

char  *gl_ptr_rs;                /* Ptr to Receiver-Status-reg.   */
char  *gl_ptr_ud;                /* Ptr to USART-Data-reg.        */
                                                                               100
char  *gl_kbshift;               /* Ptr to Shiftregister          */
IOREC *gl_iorec;                 /* Ptr to Keyboard-IOREC-struct. */

/*************************************************************************/
/* Declaration of XBRA structure                                         */
/*************************************************************************/

typedef struct
{
   char xb_magic[4];                                                           110
   char xb_id[4];
   long xb_oldvec;
} XBRA;
```

```
/*****************************************************************************/
/* Forward-declarations of subroutines (ANSI-C)                              */
/*****************************************************************************/

void install_kbd      (char *txt, int def_button, char* dialog);                    install_kbd
void remove_kbd       (XBRA *old_xbra_ptr, XBRA *xbra_ptr,                          remove_kbd
                  int def_button, char* dialog);
long init_ptr         (void);                                                       init_ptr
char convert_scan_code (char dec_scan);                                             convert_scan_code
void my_c_hdl         (void);                                                       my_c_hdl
void learn            (char dec_scan);                                              learn
void send_rs232       (char code1, char code2);                                     send_rs232

/*****************************************************************************/
/* The OS-routine process_it expects *iorec in A0, st_scan in D0             */
/* The OS-routine put_rs232  expects code in D0                              */   130

void (*process_it)    (IOREC *iorec, char st_scan);
void (*put_rs232)     (char code);

/* Keep this in mind! This is by chance the way my compiler passes Params  */
/*****************************************************************************/

/*****************************************************************************/
/* external declarations of assembler routines                               */
/*****************************************************************************/   140

extern void my_asm_hdl (void);                                                      my_asm_hdl
extern void my_rwabs   (void);                                                      my_rwabs

/*****************************************************************************/
/*                                                                           */
/* main procedure                                                            */
/* - decide whether to install or remove kbd                                 */
/*                                                                           */
/*****************************************************************************/   150

void main(void)                                                                     main
{
XBRA *xbra_ptr, *old_xbra_ptr;
void (*chkvec)();
int  found = 0;

   (long) chkvec      = INVALID_PTR;
   (long) old_xbra_ptr = INVALID_PTR;
                                                                                    160
   /* get current vector */

   xbra_ptr = (XBRA *) Setexc (RS232_VEC_ID, chkvec);

   /* follow the chain as long as it is XBRA and KBD not found */

   while (     STR4EQU ((--xbra_ptr)->xb_magic,"XBRA") &&
      (found = STR4EQU (   xbra_ptr ->xb_id,   "KB14")) == 0)
   {
      old_xbra_ptr    = xbra_ptr;                                                   170
      (long) xbra_ptr = xbra_ptr->xb_oldvec;
```

```
  }

  if (found)

    /* KBD already installed */

    remove_kbd (old_xbra_ptr, xbra_ptr,
            REMOVE_BUTTON, REMOVE_DIALOG);

  else if (IN_ROM ((long) xbra_ptr + sizeof(XBRA)))

    /* XBRA chain ok or no chain at all */

    install_kbd("   \n  ", INSTALL_BUTTON, INSTALL_DIALOG);

  else

    /* XBRA chain interrupted */

    install_kbd(" \n XBRA-chain interrupted!                    ",
            QUIT_BUTTON, INSTALL_WARN_DIALOG);

}

/*************************************************************************/
/*                                                                       */
/* remove_kbd                                                            */
/* - do dialog                                                           */
/* - unchain KBD or restore old vector                                   */
/*                                                                       */
/*************************************************************************/

void remove_kbd (XBRA *old_xbra_ptr, XBRA *xbra_ptr,
            int def_button, char* dialog)
{

void (*oldvec)();
int button;


  if ((button=form_alert(def_button,dialog)) != QUIT_BUTTON)
  {
    /* remove */

    if ((long) old_xbra_ptr == INVALID_PTR) /* no more chain memb. ? */
    {
      /* restore vector */

      (long) oldvec = xbra_ptr->xb_oldvec;

      Setexc (RS232_VEC_ID, oldvec);
    }
    else

      /* unchain */
```

remove_kbd

```
        old_xbra_ptr->xb_oldvec = xbra_ptr->xb_oldvec;
    }

    /* run from autofolder ? */

    if (button == NO_AES)

        puts("\n\x1bp DEC-LK201 Keyboard-driver V1.4a deactivated! \x1bq\n");

    /* Process terminate with success */

    Pterm (SS_DOLLAR_NORMAL);
}

/*************************************************************************/
/*                                                                       */
/* install_kbd                                                           */
/* - do dialog                                                           */
/* - setup pointers                                                      */
/* - configure RS232-port                                                */
/* - setup US-keytable (conditionally)                                   */
/* - modify RS232-rcv-buf-full vector to point to my handler             */
/*                                                                       */
/*************************************************************************/

void install_kbd (char *txt, int def_button, char* dialog)                install_kbd
{
extern XBRA my_xbra1;
int button;

char unshift_tab [] = US_UNSHIFT_TAB;
char shift_tab   [] = US_SHIFT_TAB;
char capslock_tab[] = US_CAPSLOCK_TAB;

    if ((button=form_alert(def_button,dialog)) != QUIT_BUTTON)
    {
        /* setup pointers and RWABS-vector in supervisor mode */

        Supexec (init_ptr);

        /* setup RS232 port */

        Rsconf (2, 0, -1, -1, -1, -1);

        /* install new interrupthandler, save the old one */

        my_xbra1.xb_oldvec = (long) Setexc (RS232_VEC_ID, my_asm_hdl);

        /* Set keytable according to macro KBD_TYPE */

        if (strcmp (KBD_TYPE, "US") == 0)
            Keytbl (unshift_tab, shift_tab, capslock_tab);
    }

    /* run from autofolder ? */

    if (button == NO_AES)
```

```
   {
      if (txt[4] != ' ')
         puts (txt);

      puts("\x1bp DEC-LK201 Keyboard-driver V1.4a " KBD_TYPE " installed! \x1bq\n");
   }

   /* process terminate and stay resident, _PgmSize is compilerspecific! */

   Ptermres (_PgmSize, SS_DOLLAR_NORMAL);
}

/*****************************************************************************/
/*                                                                           */
/* init_ptr                                                                  */
/* - setup some global pointers referred by the interrupthandler             */
/*   including function-pointers pointing to ROM                             */
/* - modify RWABS-vector                                                     */
/*                                                                           */
/*****************************************************************************/

long init_ptr(void)                                                    init_ptr
{
extern XBRA my_xbra2;
long *ptr;

   /* setup ptr to kbshift-register */

   (long)   ptr       = SYSHDR_PTR;
   (long)   ptr       = *ptr;
   (long *) gl_kbshift = ((SYSHDR *) ptr)->kbshift;

   /* initialize kbshift-register */

   *gl_kbshift &= 0xf0;

   /* setup ptr to kbd-iorec-register */

   gl_iorec = Iorec(1);

   /* save and bend RWABS-vector */

   (long) ptr        = RWABS_VEC;
   my_xbra2.xb_oldvec = *ptr;
   *ptr              = (long) my_rwabs;

   /* init ptr to MFP */

   (long) gl_ptr_rs  = RCV_STAT_REG;
   (long) gl_ptr_ud  = DATA_REG;

   /* init function-ptr to ROM */

   (long) process_it = KBD_OS_JUMP;
   (long) put_rs232  = RS232_OS_JUMP;

   return (SS_DOLLAR_NORMAL);
```

```
}

/*****************************************************************************/
/*                                                                           */
/* my_c_hdl - this is the 'real' interrupt-handler called by my_asm_hdl()  */
/*                                                                           */
/*****************************************************************************/
                                                                          350
void my_c_hdl(void)                                                  my_c_hdl

{
char st_scan;
char dec_scan;

   /* is it rcv-buf-full interrupt ? */

   if (*gl_ptr_rs & 0x80)
                                                                          360
      /* is converted content of USART-dataregister a processable key ? */

      if ((st_scan = convert_scan_code (dec_scan=*gl_ptr_ud)) != 0)

         /* let the OS process it */

         process_it (gl_iorec,st_scan);

   /* every key is send to learn function */
                                                                          370
   learn (dec_scan);
}

/*****************************************************************************/
/*                                                                           */
/* learn - handle learn-mechanism                                        */
/*                                                                           */
/* If you see the reason why I had to limit it to 9-keystrokes.          */
/* Let me know.                                                          */
/*                                                                           */                                                          380
/*****************************************************************************/

void learn (char dec_scan)                                           learn
{
int        do_it_ind;
static int  learn_stat = NO_LEARN;
static char learn_buf [LEARN_BUF_SIZE];
static int  cur_ind;
char       st_scan;
                                                                          390
   /* this should by self-explaining */

   if (dec_scan == LEARN_KEY)

      if (*gl_kbshift & 0x03) /* shift active? */
      {
         learn_stat = LEARN_IN_PROG;
         cur_ind = 0;
         send_rs232 (LED_ON, HOLD_LED);
```

```
      }                                                                    400
      else if (learn_stat == LEARN_IN_PROG)
      {
         learn_stat = LEARNED;
         send_rs232 (LED_OFF, HOLD_LED);
      }
      else if (learn_stat == LEARNED)

         for (do_it_ind = 0; do_it_ind < cur_ind;)
         {
            st_scan = convert_scan_code(learn_buf[do_it_ind++]);           410
            if (st_scan) process_it (gl_iorec, st_scan);
         }
      else;

   else if (learn_stat == LEARN_IN_PROG)

      if (cur_ind < LEARN_BUF_SIZE)

         learn_buf [cur_ind++] = dec_scan;
                                                                           420
      else

         put_rs232 (SOUND_BELL);
}

/*************************************************************************/
/*                                                       */
/* send_rs232 - useful because switching the lamps always requires 2 codes */
/* - expects codes in D0 and D1! (called from C and assembler)           */
/*                                                       */                430
/*************************************************************************/

void send_rs232 (char code1, char code2)                                send_rs232

{
   put_rs232 (code1);
   put_rs232 (code2);
}

/*************************************************************************/  440
/*                                                       */
/* convert_scan_code                                        */
/* - manipulates the keyboard-shift-register                        */
/* - converts scancodes (how did you know ?)                           */
/*                                                       */
/*************************************************************************/

char convert_scan_code (char dec_scan)                                  convert_scan_code

{                                                                         450
char st_scan;
static long old_st_scan = 0;

   st_scan = 0;

   /* if last key was processable and alt is active */
```

```
if (old_st_scan && (*gl_kbshift & 0x08))

   /* reset alt bit */                                                    460

   *gl_kbshift &= 0xf7;

/* main dispatcher to convert scancodes, should be self-explaining */

switch (dec_scan)
{
   case DEC_REPEAT          : st_scan = old_st_scan;  break;

   case DEC_ALT             : *gl_kbshift |= 0x08;                       470
                              send_rs232 (LED_ON, ALT_LED);
                              break;

   case DEC_CTRL            : *gl_kbshift |= 0x04;    break;

   case DEC_SHIFT           : *gl_kbshift |= 0x03;    break;

   case DEC_CAPS_LOCK       : *gl_kbshift ^= 0x10;

                              if (*gl_kbshift & 0x10)                   480
                                 send_rs232 (LED_ON,  CAPS_LED);
                              else
                                 send_rs232 (LED_OFF, CAPS_LED);
                              break;

   case DEC_ALL_UP          : *gl_kbshift &= 0xf0;    break;

   case BOOT_KEY            : *gl_kbshift |= 0x0d;
                              st_scan = ST_DELETE;   break;
                                                                         490
   case DEC_HASH            : if (*gl_kbshift & 0x03)
                                 st_scan = ST_SCHLEIFE;
                              else
                                 st_scan = ST_HASH;
                              break;

   case DEC_SCHLEIFE        : if (*gl_kbshift & 0x03)
                                 st_scan = ST_HASH;
                              else
                                 st_scan = ST_SCHLEIFE;                 500
                              break;

   case DEC_RETURN          : st_scan = ST_RETURN;     break;
   case DEC_BACKSPACE       : st_scan = ST_BACKSPACE;  break;
   case DEC_TAB             : st_scan = ST_TAB;        break;
   case DEC_F_11            : st_scan = ST_ESC;        break;

   case DEC_SPACE           : st_scan = ST_SPACE;      break;
   case DEC_A               : st_scan = ST_A;          break;
   case DEC_B               : st_scan = ST_B;          break;          510
   case DEC_C               : st_scan = ST_C;          break;
   case DEC_D               : st_scan = ST_D;          break;
   case DEC_E               : st_scan = ST_E;          break;
   case DEC_F               : st_scan = ST_F;          break;
```

```
case DEC_G              : st_scan = ST_G;       break;
case DEC_H              : st_scan = ST_H;       break;
case DEC_I              : st_scan = ST_I;       break;
case DEC_J              : st_scan = ST_J;       break;
case DEC_K              : st_scan = ST_K;       break;
case DEC_L              : st_scan = ST_L;       break;                  520
case DEC_M              : st_scan = ST_M;       break;
case DEC_N              : st_scan = ST_N;       break;
case DEC_O              : st_scan = ST_O;       break;
case DEC_P              : st_scan = ST_P;       break;
case DEC_Q              : st_scan = ST_Q;       break;
case DEC_R              : st_scan = ST_R;       break;
case DEC_S              : st_scan = ST_S;       break;
case DEC_T              : st_scan = ST_T;       break;
case DEC_U              : st_scan = ST_U;       break;
case DEC_V              : st_scan = ST_V;       break;                  530
case DEC_W              : st_scan = ST_W;       break;
case DEC_X              : st_scan = ST_X;       break;
case DEC_Y              : st_scan = ST_Y;       break;
case DEC_Z              : st_scan = ST_Z;       break;

case DEC_1              : st_scan = ST_1;       break;
case DEC_2              : st_scan = ST_2;       break;
case DEC_3              : st_scan = ST_3;       break;
case DEC_4              : st_scan = ST_4;       break;
case DEC_5              : st_scan = ST_5;       break;                  540
case DEC_6              : st_scan = ST_6;       break;
case DEC_7              : st_scan = ST_7;       break;
case DEC_8              : st_scan = ST_8;       break;
case DEC_9              : st_scan = ST_9;       break;
case DEC_0              : st_scan = ST_0;       break;

case DEC_SUCHEN         : st_scan = ST_HELP;    break;
case DEC_EINFUEGEN      : st_scan = ST_INSERT;  break;
case DEC_LOESCHEN       : st_scan = ST_DELETE;  break;
case DEC_SELEKTIEREN    : st_scan = ST_KP_PUNKT; break;                 550
case DEC_BILD_AUF       : st_scan = ST_UNDO;    break;
case DEC_BILD_AB        : st_scan = ST_CLR_HOME; break;
case DEC_ARR_LINKS      : st_scan = ST_ARR_LINKS; break;
case DEC_ARR_RECHTS     : st_scan = ST_ARR_RECHTS; break;
case DEC_ARR_AUF        : st_scan = ST_ARR_AUF; break;
case DEC_ARR_AB         : st_scan = ST_ARR_AB;  break;

case DEC_KP_PF1         : st_scan = ST_KP_OPAR; break;
case DEC_KP_PF2         : st_scan = ST_KP_CPAR; break;
case DEC_KP_PF3         : st_scan = ST_KP_SLASH; break;                 560
case DEC_KP_PF4         : st_scan = ST_KP_STERN; break;
case DEC_KP_MINUS       : st_scan = ST_KP_MINUS; break;
case DEC_KP_KOMMA       : st_scan = ST_KP_PLUS; break;
case DEC_KP_EINGABE     : st_scan = ST_KP_ENTER; break;
case DEC_KP_PUNKT       : st_scan = ST_KP_PUNKT; break;
case DEC_KP_1           : st_scan = ST_KP_1;    break;
case DEC_KP_2           : st_scan = ST_KP_2;    break;
case DEC_KP_3           : st_scan = ST_KP_3;    break;
case DEC_KP_4           : st_scan = ST_KP_4;    break;
case DEC_KP_5           : st_scan = ST_KP_5;    break;                  570
case DEC_KP_6           : st_scan = ST_KP_6;    break;
```

```
        case DEC_KP_7           : st_scan = ST_KP_7;      break;
        case DEC_KP_8           : st_scan = ST_KP_8;      break;
        case DEC_KP_9           : st_scan = ST_KP_9;      break;
        case DEC_KP_0           : st_scan = ST_KP_0;      break;

        case DEC_F_1            : st_scan = ST_F_1;       break;
        case DEC_F_2            : st_scan = ST_F_2;       break;
        case DEC_F_3            : st_scan = ST_F_3;       break;
        case DEC_F_4            : st_scan = ST_F_4;       break;
        case DEC_F_5            : st_scan = ST_F_5;       break;
        case DEC_F_6            : st_scan = ST_F_6;       break;
        case DEC_F_7            : st_scan = ST_F_7;       break;
        case DEC_F_8            : st_scan = ST_F_8;       break;
        case DEC_F_9            : st_scan = ST_F_9;       break;
        case DEC_F_10           : st_scan = ST_F_10;      break;
        case DEC_HELP           : st_scan = ST_HELP;      break;

        case DEC_OE             : st_scan = ST_OE;        break;
        case DEC_AE             : st_scan = ST_AE;        break;
        case DEC_UE             : st_scan = ST_UE;        break;
        case DEC_SS             : st_scan = ST_SS;        break;
        case DEC_APO            : st_scan = ST_APO;       break;
        case DEC_KOMMA          : st_scan = ST_KOMMA;     break;
        case DEC_PUNKT          : st_scan = ST_PUNKT;     break;
        case DEC_MINUS          : st_scan = ST_MINUS;     break;
        case DEC_PLUS           : st_scan = ST_PLUS;      break;
        case DEC_KLEINER        : st_scan = ST_KLEINER;   break;

/* useful for debugging purposes */

/* default  : st_scan = 0; printf("\n%x",dec_scan);  */

    }

    /* if new key is processable and alt is active */

    if (st_scan && (*gl_kbshift & 0x08))

        /* turn off alt indicator */

        send_rs232 (LED_OFF, ALT_LED);

    /* buffer new code */

    old_st_scan = st_scan;

    /* function value is new code */

    return st_scan;
}
```