Hardwarenahe Programmierung in C/C++

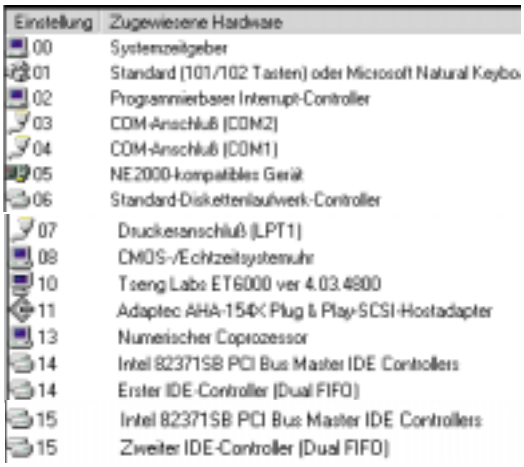
Hardware-Ressourcen, Assembler

Christian Zahler

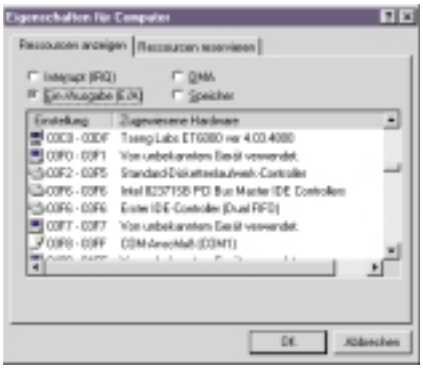
Übersicht Hardware-Ressourcen

Übersicht Interrupts

[Systemsteuerung]-[System], Karteikarte „Geräte-Manager“, Doppelklick auf „Computer“:



Übersicht IO-Ports



Beispielprogramm für Interrupt 0x05: PrintScreen

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>

void main()
{
   union REGS reg;
   char ch;
   printf("Programm zum Drucken des Bildschirminhalts\n");
   printf("Wollen Sie wirklich....\? [j/n]\n");
   ch=getche();
   if (ch=='j')
     int86(5,&reg,&reg);
   getch();
   putchar('\n');
}
```

Einige Assembler-Anweisungen im Überblick

zahl ... 2 Byte-Zahl (= 1 Word)

`MOV register, zahl`	move	verschiebt eine Zahl in ein Register (Laden einer Zahl aus dem Arbeitsspeicher)
`MOV zahl, register`	move	verschiebt Registerinhalt in Zahl (Schreiben einer Zahl in den Arbeitsspeicher)
`MOV register1, register2`	move	verschiebt Registerinhalt in anderes Register
`XCHG zahl1, register`	exchange	tauscht die Inhalte von zahl1 und register aus
`CMP register, zahl`	compare	vergleicht eine Zahl mit dem Registerinhalt
`JGE sprungmarke`	jump if greater or equa	springt zu einer Sprungmarke, falls Registerinhalt größer oder gleich einem bestimmten Wert war
`INC register`	increment	erhöht den Wert des Registers um 1

Direkter Einbau von Assembler-Anweisungen in C mit asm

Hinweis: Nach Assembler-Anweisungen kein Semikolon (;)!

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>

int max(int, int);

void main()
{
     printf("\nDie größere Zahl ist %d\n", max(3,5));
     getch();
}

int max(int zahl1, int zahl2)
{
     asm mov ax,zahl1
     asm cmp ax,zahl2
     asm jge ende
     asm mov ax,zahl2
     ende:
          return( AX);
}
```

Compilieren mit BCC -B ASM.C, damit Assembler-Befehle korrekt „übersetzt“ bzw. eingebunden werden.

Die Funktion __emit__

gestattet direkten Einbau von Bytefolgen in ein C-Programm

Ausschalten der Compiler-Warnungen

[Options]-[Compiler]-[Frequent Errors]- deaktivieren der Eintragung „Parameter ident is never used“

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>

int max(int, int);

void main()
{
     printf("\nDie größere Zahl ist %d\n", max(3,5));
     getch();
}

int max(int zahl1, int zahl2)
{
     __emit__
```

```
      (0x8B, 0x46, 0x04, /* MOV AX, [BP + 04] */
       0x3B, 0x46, 0x06, /* CMP AX, [BP + 06] */
       0x7D, 0x03,    /* JGE ip + 3 */
       0x8B, 0x46, 0x06); /* MOV AX, [BP + 06] */
      return( AX);
}
```

Beispiel zur Druckeransteuerung (INT 0x17)

```
INT 17 - PRINTER - OUTPUT CHARACTER
     AH = 00h
     AL = character
     DX = printer port (0-3)
LPT1 .... 0
LPT2 .... 1
LPT3 .... 2
LPT4 .... 3
Return: AH = status bits
         0 = time out
         1 = unused
         2 = unused
         3 = I/O error
         4 = selected
         5 = out of paper
         6 = acknowledge
         7 = not busy
-------------------------------
INT 17 - PRINTER - INITIALIZE
     AH = 01h
     DX = printer port (0-3)
Return: AH = status (see AH = 00h above)
-------------------------------
INT 17 - PRINTER - GET STATUS
     AH = 02h
     DX = printer port (0-3)
Return: AH = status (see AH = 00h above)
```

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>
/* Prototypen-Deklaration */
int p ready(void);
void printstr(char *);
/* Registervariable definieren */
union REGS rg;

void main()
{
     if (p ready())
     {
       printstr("\16Drucker OK KF!\n\r");
       /* Die Zahl 16 gibt im 1. Byte des Strings die
       Stringlänge an. \16, weil nur ein Byte belegt werden soll. */
     }
     else
     {
       printf("\n Drucker nicht bereit KF!\007\n");
     }
     getch();
} /* Ende Main */

/* Abfrage ob Drucker bereit */
int p_ready()
{
     rg.x.ax = 0x0200; /* AX für Druckerstatus initialisieren */
     rg.x.dx = 1;  /* LPT 2 einstellen */
     int86(0x17, &rg, &rg); /* Drucker-Interrupt auslösen */
     if ((rg.x.ax & 0x6900)==0)
       return 1;
     else
       return 0;
}

void printstr(char *str)
{
     int i;
     for (i=1; i <= str[0]; i++)
     {
          rg.x.ax = str[i];
          rg.x.dx = 1;
          int86(0x17, &rg, &rg);
     }
}
```

Selbstprogrammierte Interruptserviceroutine

Beispielprogramm „Uhr“

INT 1C - CLOCK TICK
This interrupt is called (in the IBM) at the end of each time-update operation by the time-of-day routines. It normally points to an IRET.

Dieser Interrupt wird 18,2mal pro Sekunde vom System aufgerufen. Er kann daher mit dem Ticken einer Uhr verglichen werden.

Wenn ein Programm einen Interrupt installiert, dann muss am Ende des Programms unbedingt der alte Zustand wiederhergestellt werden. Die Variablen `alt_vec_1c` bzw. `alt_vec_9` speichern die Adressen der Interruptservice-Routinen für die Interrupts **0x1c** und **0x09**.

```
#include <stdio.h>
#include <dos.h>

#define BILD  0xb800   /*Bildschirmspeicherbeginn für VGA-Karte*/
#define POS   140
#define BLINK 0x0
#define FARBE ((0x40 < 8) + '0')

int display = 1;

void interrupt (*alt_vec_1c) ();
void interrupt (*alt vec 9) ();

void interrupt zeit_func()
{
long far *zeit_pt;
long zeit, h, m, s;
alt_vec_1c ();

if (display)
{
  zeit_pt = (long far *)MK_FP(0,0x46C); // MK_FP erzeugt far-Zeiger
  zeit = *zeit pt * 10 / 182;
  h = zeit / 3600;
  m = zeit / 60 - h * 60;
  s = zeit - h * 3600 - m * 60;
  poke (BILD, 0 + POS, FARBE - '0');
/* poke speichert integer-Wert an der Adresse */
/* Syntax:    poke (segment, offset, value);  */
  poke (BILD, 2 + POS, FARBE | h / 10);
  poke (BILD, 4 + POS, FARBE | h % 10);
  poke (BILD, 6 + POS, FARBE | BLINK | ':' - '0');
  poke (BILD, 8 + POS, FARBE | m / 10);
  poke (BILD, 10 + POS, FARBE | m % 10);
  poke (BILD, 12 + POS, FARBE | BLINK | ':' - '0');
  poke (BILD, 14 + POS, FARBE | s / 10);
  poke (BILD, 16 + POS, FARBE | s % 10);
  poke (BILD, 18 + POS, FARBE - '0');
}
}

void interrupt key func()
{
  int count;
  alt vec 9();
  if ((inportb(0x60) == 22)&&((peekb(0,0x417) & 12) == 12))
  // 0x60 ... Kenncode der Tastatur (Port-ID)
  // 22 ..... U-Taste
  {
    if (display)
    {
      for (count = 0; count < 10; count++)
          poke(BILD, count * 2 + POS, 32 + peekb(BILD, POS-1) < 8);
      display=0;
    }  // if (display)
    else display = 1;
  } // if inportb
}

void main(void)
{
  printf("UHR - Ein/Ausschalten mit Strg-Alt-U\n");
  alt_vec_1c = getvect(0x1C); /* Original-Timer-ISR sichern */
  setvect(0x1C, zeit func);   /* Neue ISR installieren */
  alt_vec_9 = getvect(0x9);
  setvect(0x9, key func);
  keep(0, 8192/16);
}
```