Software Engineering 2

Praktikum zum Thema

Entwicklung verteilter Echtzeitsysteme

Kurzeinführung IO Mapped IO

Einführung

Fragestellung:

- Wie kann ein Programm / die CPU auf Devices zugreifen?
- Devices sind HW Bausteine, die Zusatzfunktionen zur Verfügung stellen (z.B. Timer, Schnittstellen zur Umwelt - s. Kaffeemaschine aus der Einführung).
- Ohne diese Devices kann ein Rechner nicht mit seiner Umwelt kommunizieren (z.B. keine Steuerungsaufgaben durchführen, keine Daten übertragen, keine Ausgabe erzeugen, keine Daten einlesen, …).

Low Level Zugriffe via I/O Mapped :

- Die „Register der Devices“ werden über einen separaten Adressraum angesprochen.
- Alternative: Memory Mapped: s. Vorlesung

I/O Protokolle und Konsequenzen

- Ein Device hat Daten-, Status- und Kontrollregister.
- Über diese Register wird oftmals ein Device - spezifisches Protokoll gefahren.
- Typisches Beispiel:
  - Eine Device hat ein Status- und ein Datenregister.
  - Die CPU schreibt Daten in das Datenregister.
  - Über ein Status Bit wird die CPU informiert, ob die Daten fehlerfrei bearbeitet wurden.
  - Die CPU muss auf die „Antwort“ im Statusregister warten.
- Im allgemeinen ist die CPU um ein Vielfaches schneller als das Device. Wie wartet die CPU auf die „Antwort“ im Statusregister (s. Interrupt Behandlung)?
  - **Polling** (Problem: CPU kann nicht für andere Aufgaben genutzt werden)
  - **Interrupt** (Problem: ggf. Overhead für Interrupt, Probleme mit Realtime Anforderungen)
  - **Timer & Polling**: Es wird ein zyklischer Timer Interrupt erzeugt. Bei jedem Timer Interrupt werden die entsprechenden Statusregister der Devices abgefragt.

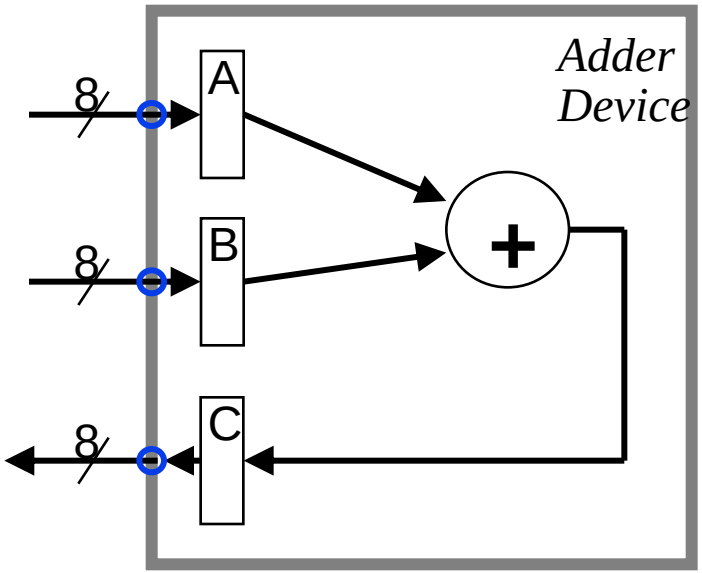
I/O Mapped Devices (I/O-adressierbare Bausteine)

- Die Devices haben einen eigenen Adressraum (I/O Adressraum) – sie werden nicht über Hauptspeicheradressen angesprochen.
- Der Programmierer greift auf das I/O Mapped Device über spezielle Befehle (Funktionen) zu (in8, in16, in32, out8, out16, out32). Diese werden letztlich auf entsprechende Maschinenbefehle abgebildet.
- Das Device wirkt wie eine "intelligente Speicherstelle" aus dem I/O Adressraum und führt eine HW-Funktion aus.

Beispiel: Programmierung von I/O Mapped Devices

HW-Funktion sei C=A+B

- Der Baustein hat die Basisadresse 0x0200
- Die Register der Bausteins
- Offset 0x00 : Parameter A
- Offset 0x01 : Parameter B
- Offset 0x02 : Ergebnis der Addition (Register C)



Beispiel: Programmierung von I/O Mapped Devices (Fortsetzung)

```
#include <inttypes.h>
#include <sys/neutrino.h>
#include <hw/inout.h>

#define adder_base_addr (0x0200)
#define adder_reg_a     (adder_base_addr + 0x00)
#define adder_reg_b     (adder_base_addr + 0x01)
#define adder_reg_c     (adder_base_addr + 0x02)

uint8_t do_add( uint8_t op1, uint8_t op2 ){
   uint8_t resu;

   ThreadCtl(_NTO_TCTL_IO,0);
   out8(adder_reg_a, op1);
   out8(adder_reg_b, op2);
   resu = in8(adder_reg_c);
   return(resu);
}//do_add

printf("3+7 = %d\n", do_add(3,7));
```

Dieser Funktionsaufruf liefert die Zugriffsrechte für die in und out Befehle

Programmierung der I/O Karten im Praktikum

Situation

- Sie müssen die Analoge und Digitale I/O Karte programmieren.
- Datenblätter gibt es bei Herrn Lohmann, auf der HomePage zur SY Vorlesung oder im Netz unter
  http://www.accesio.com/manuals/104-dio-48s.pdf
  http://www.accesio.com/manuals/104-aio12-8.pdf
- Damit Sie auf I/O Ports zugreifen können, brauchen Sie die Berechtigung vom OS. Diese erhalten Sie über den Funktionsaufruf
  ThreadCtl(_NTO_TCTL_IO, 0)
- Lesen Sie die Datenblätter, das soll im Praktikum ebenfalls geübt werden.

Zugriff Digital I/0

Beispiel:

```
#define D_IOBASE                                0x300
    #define DIGITAL_CARD_CONTROL                (D_IOBASE + 0x03)
    #define DIGITAL_CARD_CROUP0_PORTA           (D_IOBASE + 0x00)

// Initialisierung: Group 0: port A as output, port B as input 0x82
    out8( DIGITAL_CARD_CONTROL, 0x82);

// Schreibe 0 auf den Digitalen Ausgang des Triggers
    out8(DIGITAL_CARD_CROUP0_PORTA, 0x00);
```