Computer Engineering
WS 2012

Einleitung

HTM – SHF - SWR

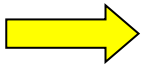
Themen:

- Ein- und Ausgabe, Peripherie
  - Memory Mapped, IO Mapped
  - Einfache Peripherie, General Purpose IO (GPIO)
  - Timer
  - Interruptverarbeitung
  - Serielle Übertragung
  - Speicher: RAM, Flash, EEPROM
- Softwareerstellung für Embedded Systems
  - Erstellung, Download, Debuggen
  - Startup-Code

Literatur:

- **LPC24XX User Manual**
  http://www.nxp.com/acrobat_download/usermanuals/UM10237.pdf
- **LPC2468 Data Sheet**
  http://www.nxp.com/acrobat_download/datasheets/LPC2468.pdf
- **The Insider's Guide To The NXP LPC2300/2400 Based Microcontrollers, Hitex**
  http://www.hitex.com/index.php?id=download-insiders-guides
- **ARMv5 Architecture Reference Manual, gilt u. a. für ARM7**
  http://www.arm.com/miscPDFs/14128.pdf
- **Sloss, Symes, Wright : ARM System Developer's Guide, Morgan Kaufmann Publishers, 2004.**
- **William Hohl: ARM Assembly Language, CRC Press, 2009.**
- **Helmut Bähring: Mikrorechner-Technik, Springer, 2002.**
- **Wayne Wolf: Computer as Components, Morgan Kaufmann,2008.**

Übersicht



- Einleitung
- Softwareerstellung

Was kennzeichnet ein eingebettetes System?

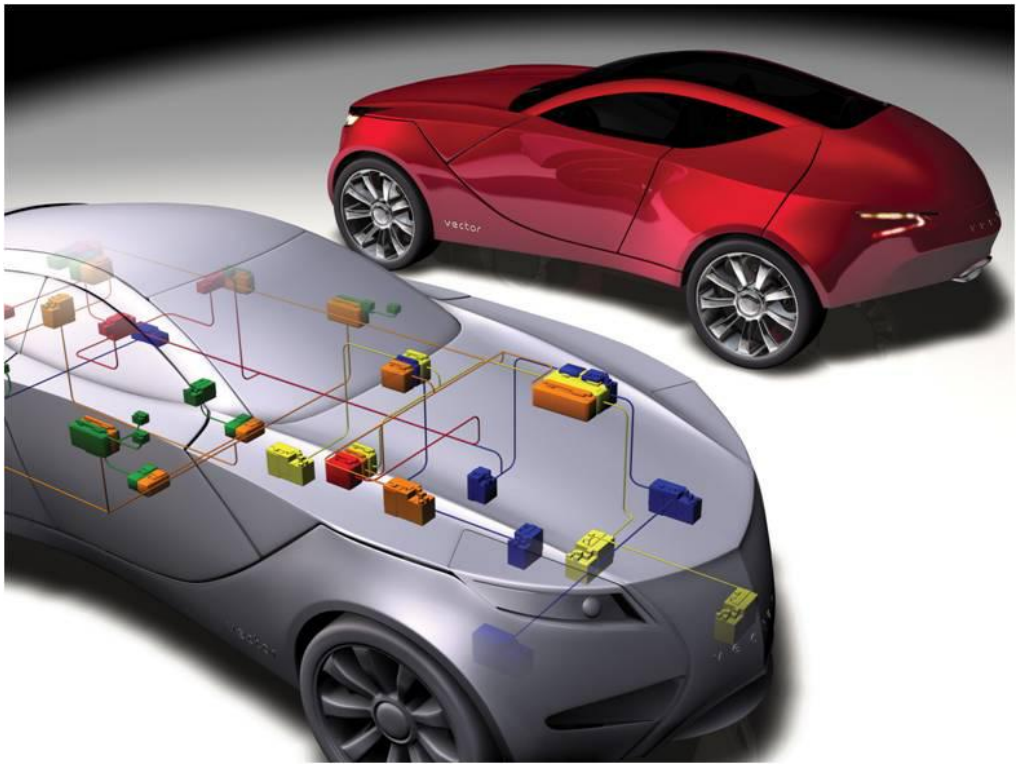
- Gerät, das ein programmierbares Prozessorsystem enthält
- Kein Allzweckrechner
  - PC selbst ist kein eingebettetes System
  - PC kann aber zum Aufbau eines eingebetteten Systems verwendet werden

Anwendungsbereiche

- Automatisierung
  - Steuern und Regeln von Prozessen, Fertigungs- und Produktionsanlagen, Überwachung
- Medizintechnik
  - Beatmungsgeräte, Narkosesysteme
  - Computer-Tomograph, Ultraschallgeräte
- Netzwerk
  - Router, Switches, WLAN, VoIP
- Haushalt
  - Waschmaschine, Mikrowelle, …
- Unterhaltung
  - Handys, Digitalkameras
  - Digitales Fernsehen, Video-on-demand, Hi-Fi, DVD, Spiele, ...

Anwendungsbereiche

- **Automobiltechnik**
  - **Oberklasse:**
    Anzahl Steuergeräte >80
  - **Diverse Bussysteme:**
    - **CAN**
    - **LIN**
    - **Flexray**
    - **MOST**
  - **Anwendungen:**
    - **Motorsteuerung**
    - **x-by-wire:**
      Bremsen,
      Lenkung
    - **Komfort**
    - **Unterhaltung**



Typische Anforderungen an eingebettete Systeme

- **Hohe Ausfallsicherheit**
- **kurze Reaktionszeiten**
- **vorgegebene technische Randbedingungen und Schnittstellen**
- **Geringer Leistungsverbrauch**
- **Niedrige Entwicklungs-/Produktionskosten**

Eingebettetes Prozessorsystem: Typische Komponenten

- CPU
- Speicher
  - Programmspeicher (Flash), Datenspeicher (RAM), Parameterspeicher (EEPROM)
- Zeitgeber
  - Steuerung von periodischen Abläufen
  - PWM (Puls-Weiten-Modulation)
- Ein- und Ausgabe
  - Digitale Schnittstellen
  - parallel, seriell (RS232, USB, I2C, CAN, ...)
  - Analoge Schnittstellen
    - Analoge Ein- und Ausgabe
- Systemmanagement
  - Speicherverwaltung
  - Interruptsteuerung
  - DMA-Steuerung
  - Powermanagement, Ausfallsicherheit (Watchdog)

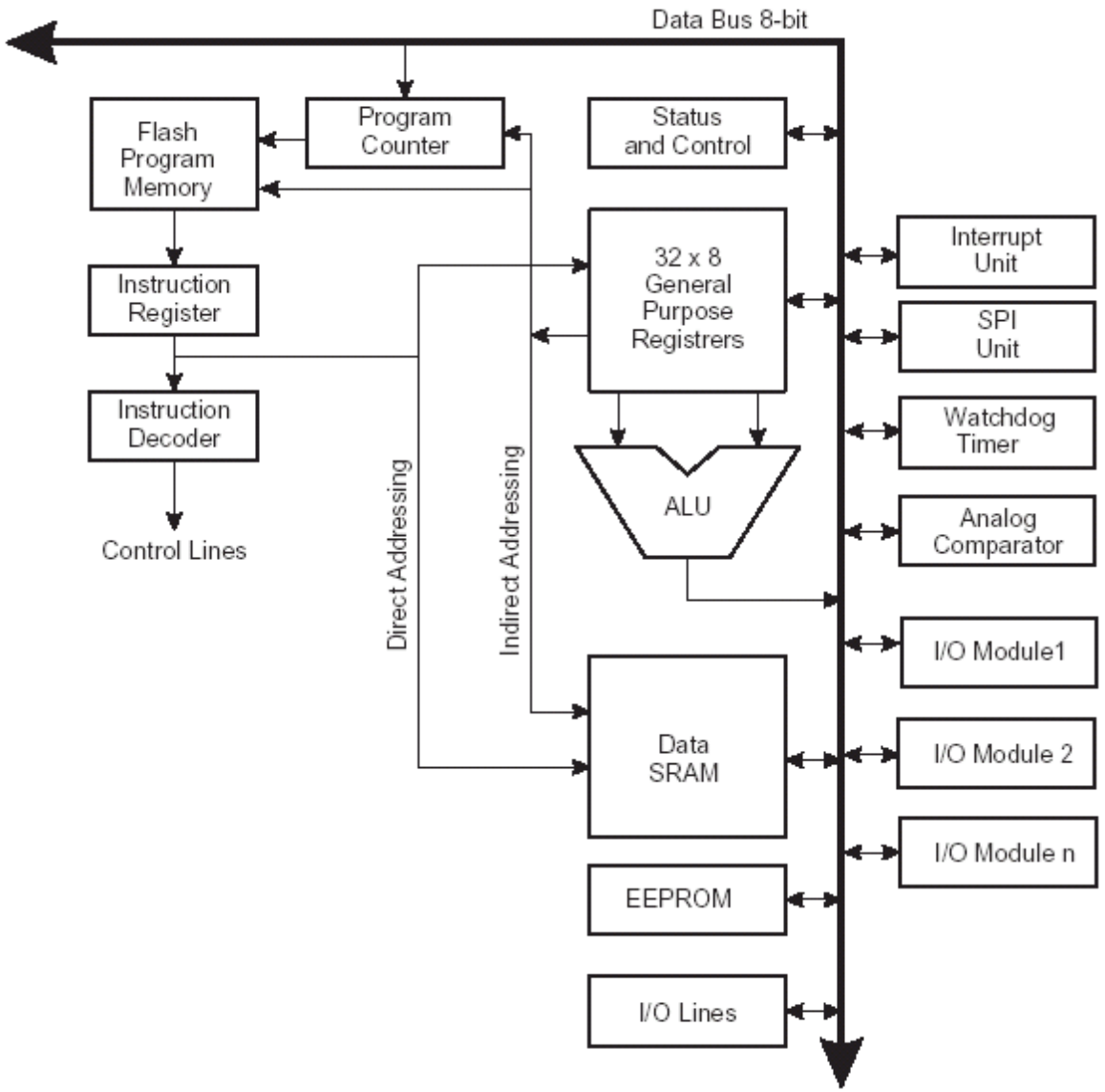
Mikrocontroller

- Enthält viele Komponenten auf einem Chip, z. B.:
  - CPU
  - Speicher
  - RAM
  - Flash
  - Zeitgeber
  - I/O

- Unterscheiden sich in
  - Geschwindigkeit (Taktfrequenz)
  - Breite Datenbus (z. B.: 4, 8, 16, 32 Bit)
  - Speichergrößen
  - I/O Komponenten

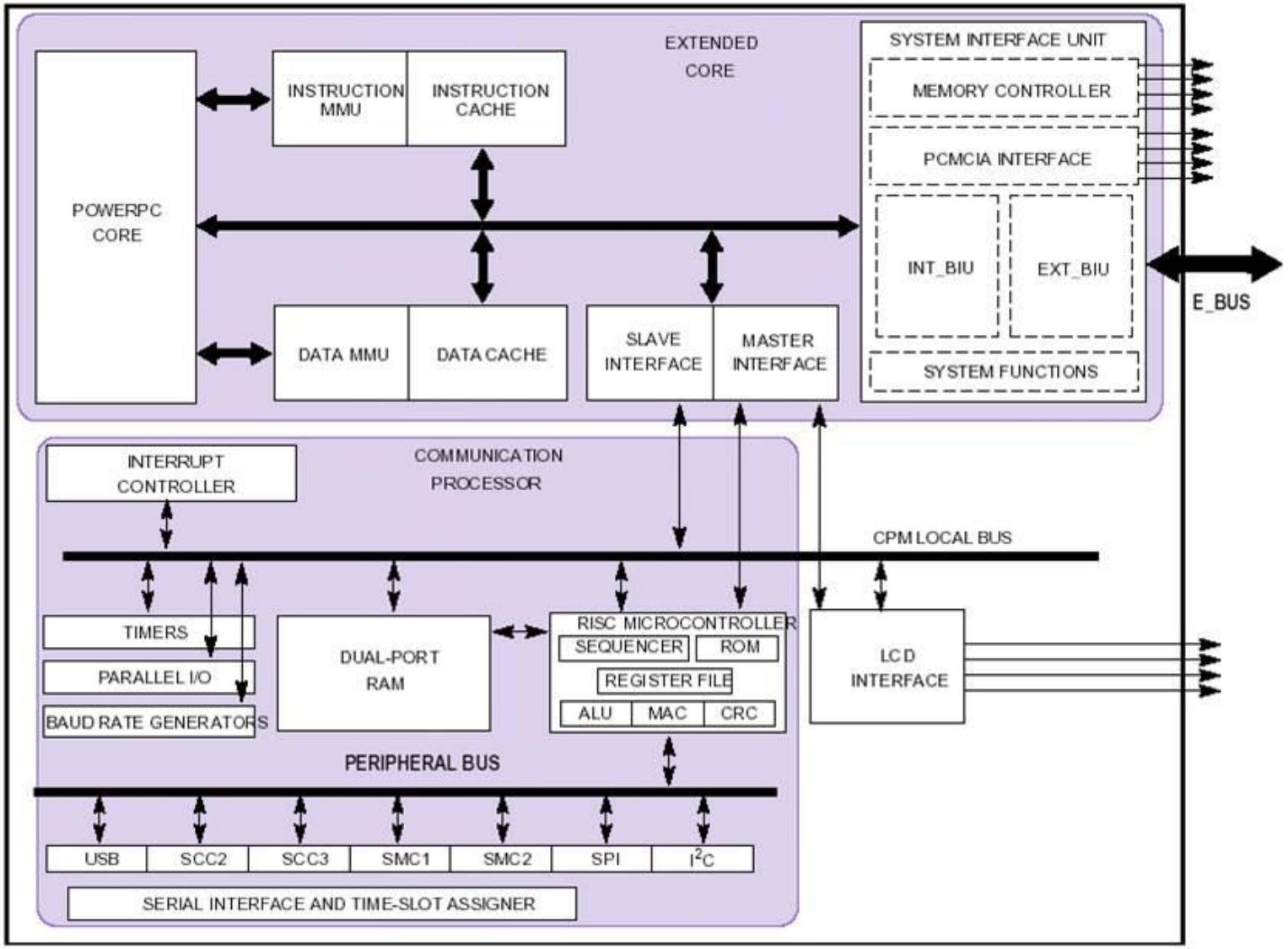
Mikrocontroller ATMEL, AVR

- **Harvard-Architektur**
- **8-Bit Datenbus**
- **I/O Module:**
  - **53 I/O Lines**
  - **4 Timer**
  - **A/D Converter**
  - **2 RS232**
  - **1 SPI**
  - **CAN Feldbus**



CE WS12

Mikrocontroller Freescale, MPC823e (Power PC)



Übersicht

- Einleitung
- Softwareerstellung

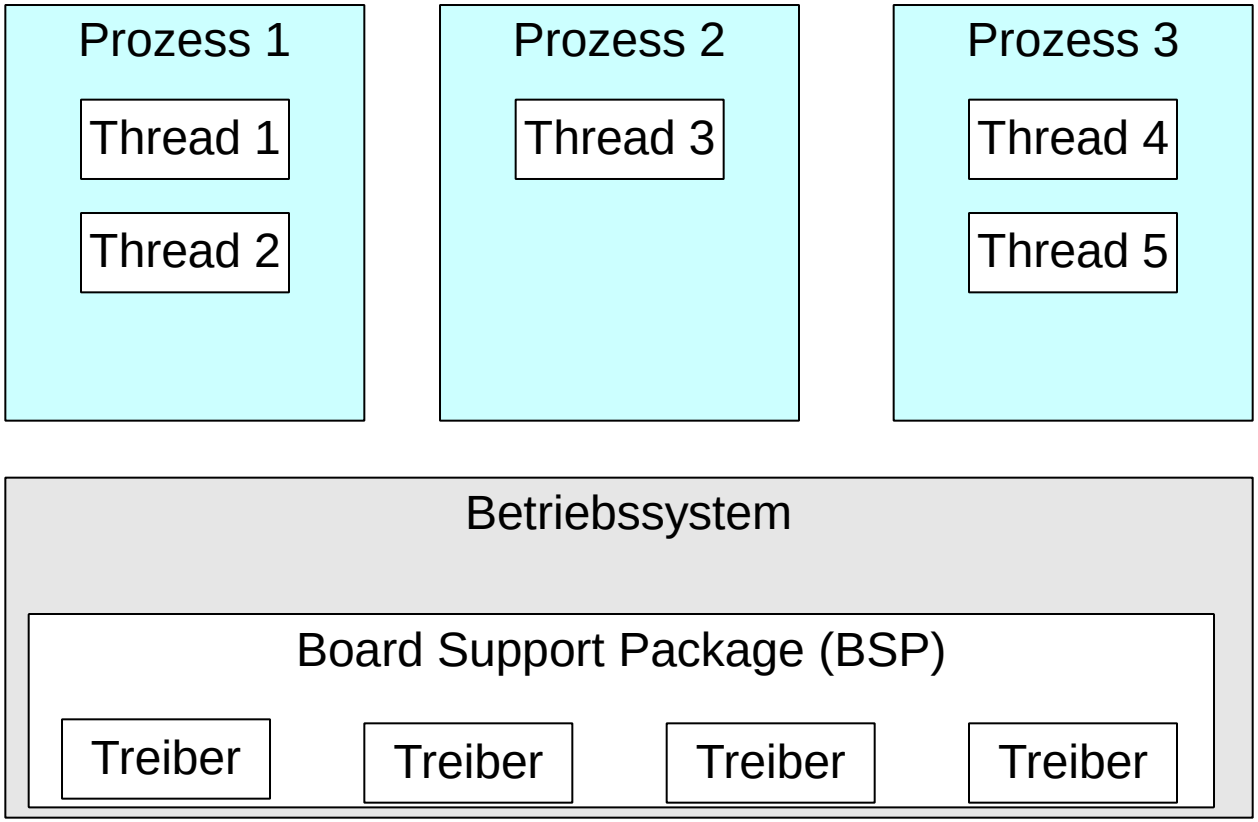
Besonderheiten Softwareentwicklung für Eingebettete Systeme

- Aufteilung der Implementierung in
  - Hardware-basiert
  - Software-basiert
- Betriebssysteme
  - häufig kein Betriebssystem
  - wenn Betriebssystem, dann meist
    - kein Speicherschutz
    - echtzeitfähig
    - erfordern meist BSPs (Board Support Package)
- Massenspeicher
  - häufig kein Massenspeicher
    - Programme werden im ROM und
    - nichtflüchtige Daten im EEPROM gespeichert
- Softwareerstellung
  - Verwendung von Crosscompilern
    - erfordert „Download des erzeugten Codes“

Besonderheiten Softwareentwicklung für Eingebettete Systeme

- Debugging, Fehlersuche
  - erfordert zusätzliche Werkzeuge
    - Simulator
    - Crossdebugger, Anbindung über
      - serielle Schnittstelle
      - Netzwerk
      - JTAG-Interface
    - Logikanalysator
      - Fehlersuche ohne Beeinflussung des Zeitverhaltens
    - Emulator
- Testen
  - erfordert
    - Einbindung in die reale Umwelt oder
    - Verwendung von „Hardware in the Loop-Technik“
- Softwarepflege
  - erfordert Möglichkeit zum Firmware-Update

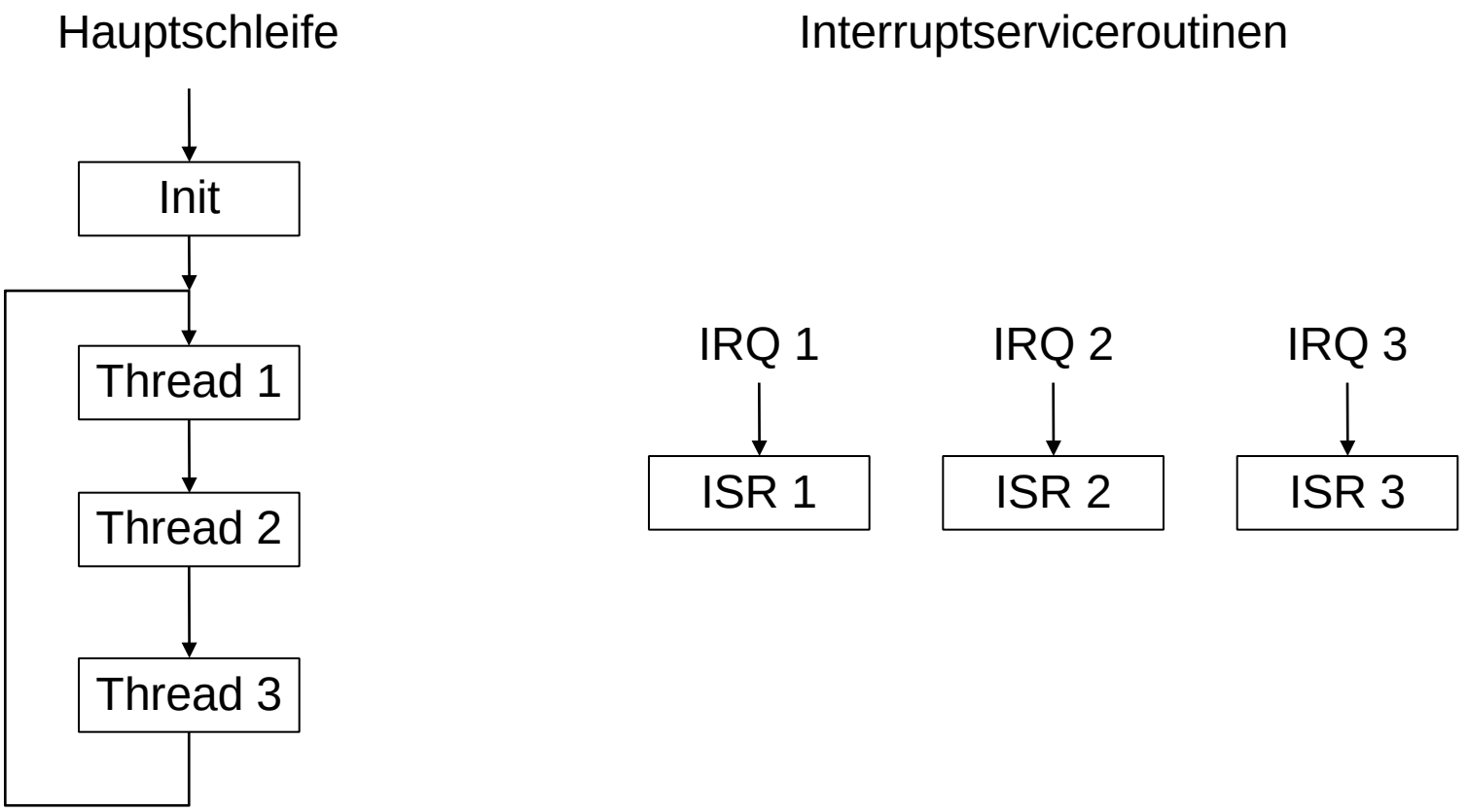
Embedded System mit Betriebssystem



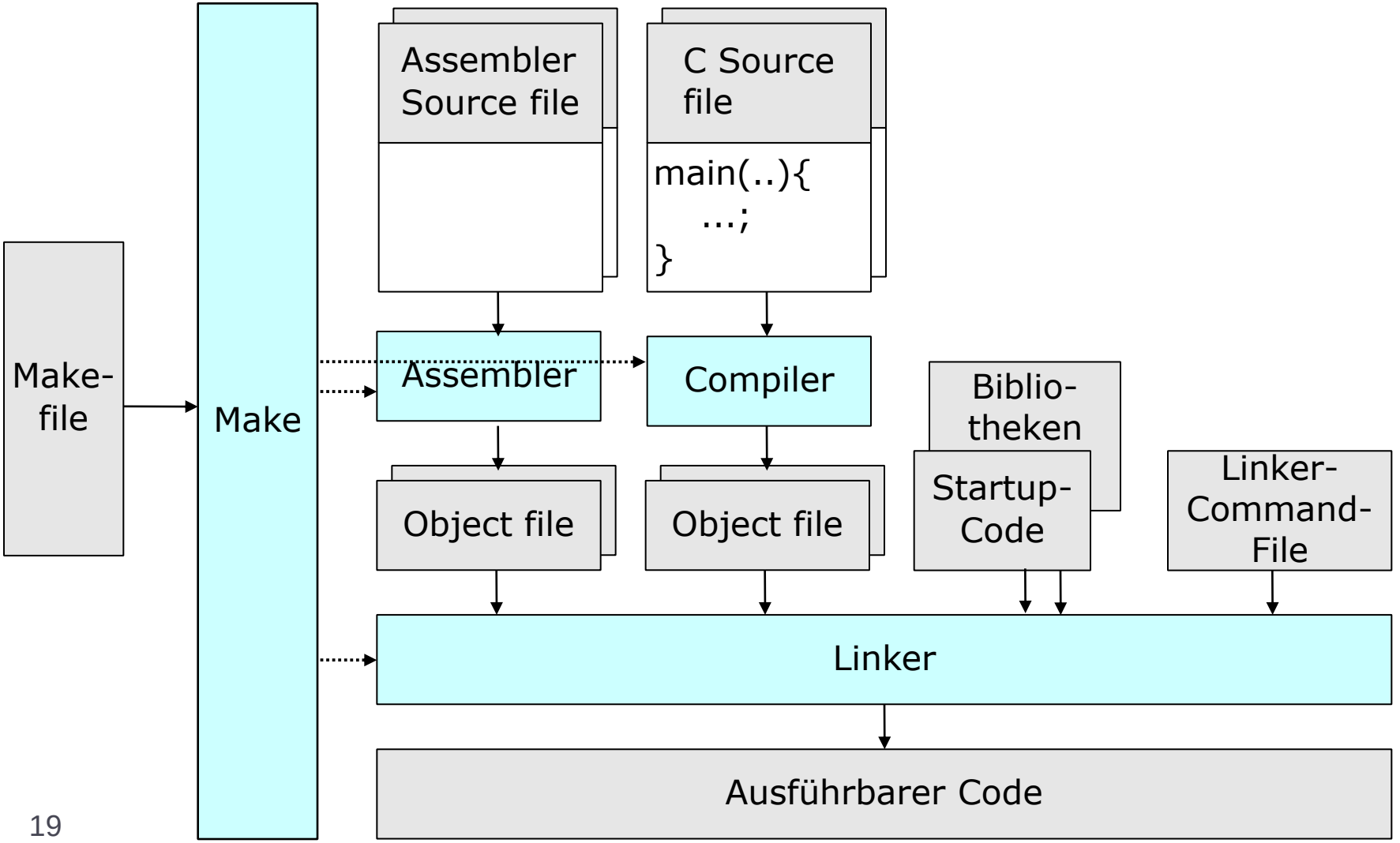
Typischer Aufbau einer Thread

```
int variable1;

double variable2 = 3.14;


void *PrintHello(void *threadarg){

  int variable3;

  initialisierung();

  while( 1 ) {

    warte_auf_Ereignisse();

    bearbeitung();

    ausgabe();
  }
}
```

Embedded System ohne Betriebssystem



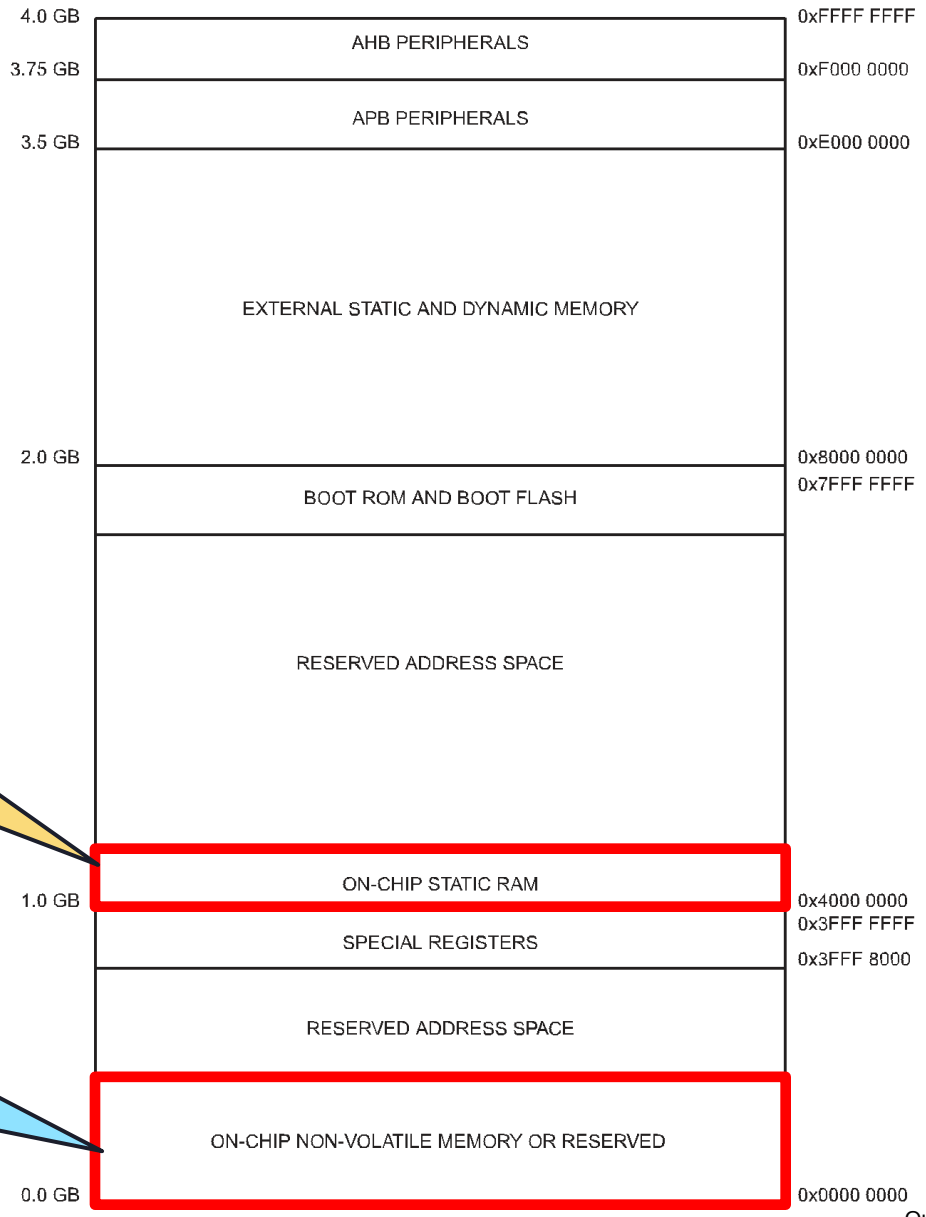
Softwareentwicklung



Speicherarten

ROM (Flash)	• Persistente Speicherung • Löschbar (blockweise) • N-mal wiederholbar (typ. 1000) • Langsam • Preiswert	• Programmspeicher
RAM	• Flüchtige Speicherung • Schnell • Teuer	• Arbeitsspeicher
EEPROM	• Persistente Speicherung • Byteweise löschbar • N-mal wiederholbar (typ. 10^6) • Langsam • Teuer	• Speicherung von Konfigurationsdaten

Speicherbelegung LPC2468



RAM (64kByte)
Start: 0x4000 0000
Ende: 0x4000 FFFF

Flash(512kByte)
Start: 0x0000 0000
Ende: 0x0007 FFFF

Quelle: LPC24XX User Manual

Sektionen

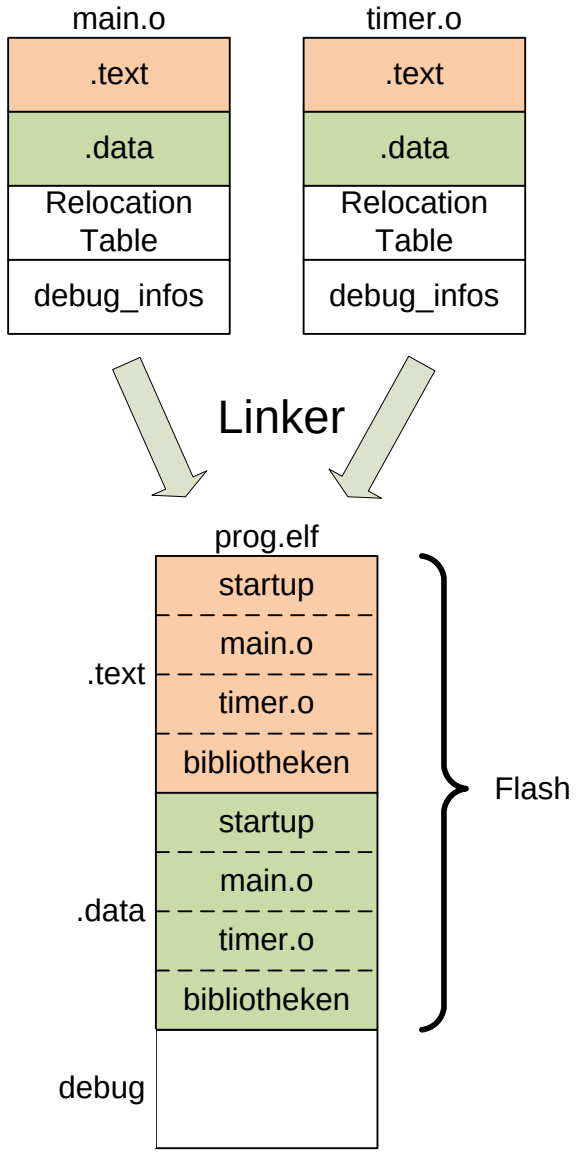
- Ausführbarer Code ist in Sektionen aufgeteilt
- Ermöglicht gezielte Platzierung des erzeugten Codes im Speicher
- Skript für Linker (Loader) gibt die Adressen der Sektionen vor

Wichtige Sektionen:

Name	Inhalt	Speicher-ort	Bemerkung
.text	Programmcode	FLASH	
.data	Initialisierte globale Variable	RAM	Bei Programmstart wird Inhalt der Variablen vom FLASH in das RAM kopiert
.bss	Nicht initialisierte globale Variable	RAM	Bei Programmstart wird Inhalt der Variablen gelöscht

Linker (loader, ld)

- Zusammenfügen der Sektionen
- Zuweisung der Adressen
  - **Jede Sektion hat zwei Adressen:**
    - **Virtual Memory Address (VMA): Adresse der Sektion, wenn das Programm ausgeführt wird.**
    - **Load Memory Address (LMA): Adresse, unter der die Daten abgelegt werden.**
- Anwendung der Relocation Table
  - **Platzhalter für Adressen werden durch die tatsächlichen Adressen ersetzt.**



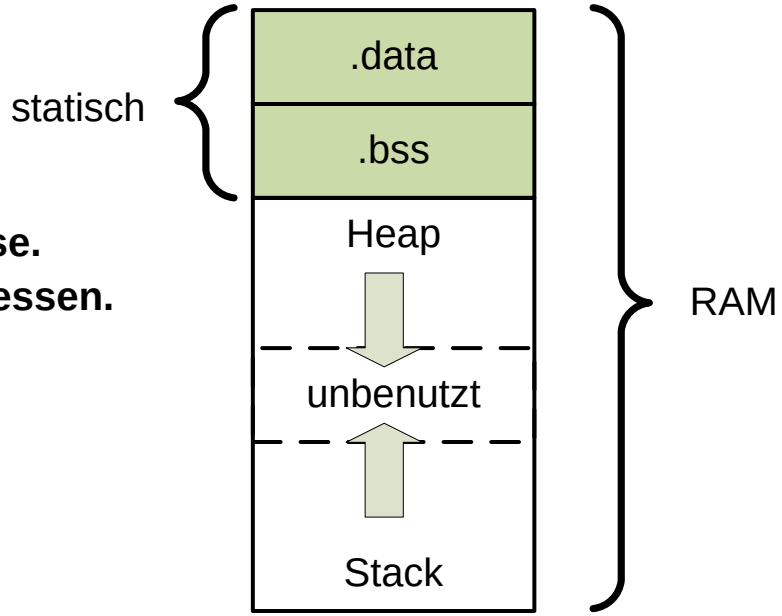
Programmiersprache C: Speicheraufteilung

- **Verwendung des Speichers:**
  - **Konstante (Verwendung des Schlüsselworts const):**
    → Spezielle Sektion .rodata.
  - **Globale und statische Variable:**
    → .data oder .bss (Abhängig von Initialisierung)
  - **Lokale Variable:**
    → Stack
  - **Dynamische Variable:**
    → Heap

Dynamische Verwendung
Speicherbedarf schwer vorhersagbar.
Kritisch: Rekursive Funktionsaufrufe.

Programmiersprache C: Speicheraufteilung

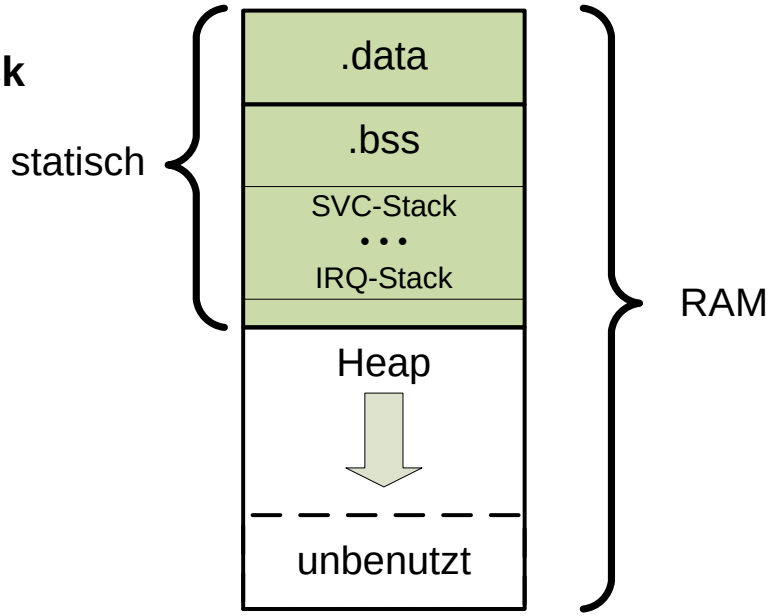
- Häufige Anordnung:
  - **Heap:**
    - **Beginnt am Ende der .bss-Sektion.**
    - **Wächst von kleinen nach großen Adressen.**
  - **Stack:**
    - **Beginnt an der höchsten RAM-Adresse.**
    - **Wächst von großen nach kleinen Adressen.**



- **Vorteil:** **Sehr flexibel.**
- **Nachteil:** **Kein Schutz vor Stack- oder Heapüberlauf.**
- **In Embedded Systems muss der Heap- und Stackbedarf vorab abgeschätzt werden.**
  Heap und Stack sollten statisch eingerichtet werden.

Programmiersprache C: Speicheraufteilung HiTOP

- **ARM 7 TDMI:**
  - **6 Stacks für 7 Betriebsarten**
    - **USR/SYS haben gemeinsamen Stack**
  - **Größen der Stacks werden statisch in der .bss-Sektion festgelegt**
    - **Stacks werden in *start.s* definiert**
  - **Heap nutzt den gesamten restlichen Speicher ab Ende der .bss-Sektion**
    - **Definiert durch die Laufzeitbibliothek**



Programmiersprache C: Laufzeitbibliothek

- Meist nicht Bestandteil des Compilers.
- Üblich:
  Verbindung der Anwendung mit dem Betriebssystem.
  - Betriebssystem wird über Software-Interrupt aufgerufen:
    - Umschaltung in System-Betriebsmode.
    - Höhere Privilegien.
  - Wesentliche Aufgaben:
    - Ein- und Ausgabe, Dateien und Geräte.
    - Prozessverwaltung, Starten und Stoppen von Prozessen.
    - Speicherverwaltung.

Programmiersprache C: Laufzeitbibliothek HiTOP

- Verwendung der **newlib**:
  - **Open-source Projekt**
  - **Standard C Bibliothek für Embedded Systems**
  - **Wird gepflegt von Red Hat Entwicklern**

  - **Kann einfach angepasst werden:**
    - **Unterstützung einer Vielzahl von CPU-Typen**
    - **Verwendung von insgesamt 17 Stub-Funktionen (System Calls).**
    - **Optimierung des Speicherbedarfs (printf versus iprintf).**
    - **Kann reentrant-fähig übersetzt werden.**

Programmiersprache C: Laufzeitbibliothek HiTOP

- newlib: System Calls, Beispiel Speicherallokierung

```
extern unsigned char end[];
extern unsigned char _end_of_heap_[];
static unsigned char *heap_ptr = NULL;

void* _sbrk_r( struct _reent *_s_r,  ptrdiff_t nbytes )  {

    unsigned char *prev_heap_end;

    if ( heap_ptr == NULL)
            heap_ptr = end;

    prev_heap_end = heap_ptr;

    if ((heap_ptr + nbytes) > _end_of_heap_){
      return NULL;
    }

    heap_ptr += nbytes;

    return (void*)prev_heap_end;
}
```

Programmiersprache C: Laufzeitbibliothek HiTOP

- newlib: System Calls, Beispiel Ausgabe

```
_ssize_t _write_r ( struct _reent *r, int file, const void *ptr,   size_t len) {
    int i;
    const unsigned char *p;
    int retval;

    p = (const unsigned char*) ptr;

    for (i = 0; i < len; i++) {
        if (*p == '\n' ){
           do{
              retval = uart0Putch('\r');
           }while(retval == -1);
        }

        do{
           retval = uart0Putch(*p);
        }while(retval == -1);
        p++;
    }

    return len;
}
```

Programmiersprache C: Laufzeitbibliothek HiTOP

- **newlib: System Calls, Beispiel Ausgabe**