

IT: Hardwarenahes Programmieren

Datum: 06.10.2022 2_0_Programmiersprache_C.doc

2.0.1

Programmiersprache C im Vergleich

Abgrenzung verschiedener Programmiersprachen

Programmiersprachen können grob in folgende Klassen eingeteilt werden:

1) Maschinensprachen

Jeder Mikroprozessortyp besitzt seine eigene Maschinensprache, bei der jeder Maschinenbefehl als eindeutiger Binärcode codiert ist. Der Programmspeicher des Prozessors kann direkt mit diesen Codes beschrieben werden.

(Location: Speicheradressen; Object: Maschinencodes in Hexadezimalcode)

2) Assemblersprachen

Verwenden "lesbare" Befehlsworte (Mnemonics) die stellvertretend für die Maschinencodes verwendet werden. Der Assembler ist ein Programm, der die Mnemonics automatisiert in Maschinencodes übersetzt.

3) In *Hochsprachen*

können komplexe Ausdrücke (y = a*b+c-3) formuliert werden, die ein Compiler wieder in einzelne Maschinenbefehle zerlegen muss. Im

Wesentlichen werden heutzutage zwei Klassen dieser sogenannten Compiler-Sprachen unterschieden:

- Strukturierte Programmiersprachen (C, Pascal,...)
- Objektorientierte Sprachen (C++, Java, C#,..)

```
#include "controller.h"
int main(void)
  bit_init(_PORTD_,7,OUT);
  while (1)
     bit_write(_PORTD_,7,1);
     bit_write(_PORTD_,7,0);
}
```

```
Adresse
        Maschinencode
000000 e200
000001 b90a
000002 e200
000003 b90b
000004
        e000
       b90b
000005
000006 cffb
```

```
.include "m328pdef.inc"
start:
        ldi r16,0x20
        out DDRD,r16
        ldi r16,0x20
loop:
        out PORTD, r16
        ldi r16,0
        out PORTD,r16
        rjmp loop
```

Entwicklung der Programmiersprache C

Nach einer längeren Enstehungsphase als Programmiersprache für UNIX-Systeme in den 1970er Jahren erschien 1978 das Buch "The C-Programming Language" von den Sprachentwicklern Brian W. Kernighan und Dennis M. Ritchie. Seit 1989 ist die Sprache C vom Amerikanischen Normungskomitee ANSI standardisiert. In ANSI-C geschriebener Quellcode konnte seither auf allen Computersystemen, für die ein ANSI-C-Compiler existiert, nahezu unverändert zu einem lauffähigen Programm übersetzt werden. C zeichnet sich unter anderem durch folgende Eigenschaften aus:

- Unterstützung von Sprachmitteln der Strukturierten Programmierung
- Hardwarenahe Programmierung durch direkte Speicherzugriffe (Pointer) und Bitoperationen
- Hohe Laufzeitperformance

Diese Eigenschaften verhalfen C, bis zum Ende der 90er Jahre, zu einer starken Verbreitung bei der Programmierung aller erdenklichen Computersysteme. Betriebssysteme wie Unix/Linux oder Windows wurden ursprünglich in C codiert. Seither hat die Bedeutung von ANSI-C für die Programmierung von PC-Systemen stetig nachgelassen. Die Komplexität dieser Programme konnte mit dem funktionalen Ansatz von C nicht mehr angemessen umgesetzt werden, so dass in diesem Segment heute ausschließlich objektorientierte Sprachen (C++) verwendet werden.

Anders sieht es bei der Programmierung von "kleinen" Mikrocontrollern aus. Diese haben, gegenüber ihren großen Brüdern, eine sehr begrenzte Hardwareausstattung, wodurch C seine Stärken als hardwarenahe und ressourcenschonende Sprache voll ausspielen kann.

Um diese Stärken nutzen zu können, ist die Kenntnis der Hardware des Systems sowie der Grenzen eines Mikrocontrollers von großer Bedeutung.



IT: Hardwarenahes Programmieren

Datum: 06.10.2022 2_0_Programmiersprache_C.docx

Programmiersprache C im Vergleich

2.0.2

Workflow bei der Programmentwicklung in C

Um ein C-Programm zu erstellen werden mehrere Programme benötigt. Dies sind der Quellcode-Editor, der Compiler, der Linker, ein Debugger und ein Downloadprogramm.

In einer modernen Entwicklungsumgebung (IDE=Integrated Development Environment) sind diese Programme alle unter einer grafischen Programmoberfläche zusammengefasst. Grundsätzlich lässt sich aber jeder Quellcode auch in der Textkonsole compilieren, linken und downloaden.

Die Vorgehensweise zur Erstellung eines lauffähigen Maschinen-Programms nennt man Build- oder Make-Prozess.



Der Präprozessor bearbeitet Präprozessoranweisungen (Direktiven), die nicht selbst Maschinencode erzeugen, aber die Übersetzung des Quellcodes beeinflussen.

Mittels Präprozessordirektiven können:

- → Bibliotheksfunktionen eingebunden werden
- → Konstanten und Makros definiert werden
- → Bedingte Compilierungen gesteuert werden

Beispiele für Präprozessor-Direktiven:

#include <stdio.h> #include "controller.h"

#define Pi 3,14159265

#ifdef Pi

#else

#endif



Der Compiler übersetzt den Quellcode einer Source-Datei in den Maschinencode. Dabei können Programme aus einer oder mehreren Quelldateien (Sourcen) bestehen. Der Compiler durchläuft verschieden Bearbeitungsschritte:

- Lexikalische Analyse
- **Syntaxanalyse**
- Semantische Analyse
- Optimierungen

Eingangsdateien (Endungen):

Sourcecode: *.c Headerdatei: *.h Bibliotheken: *.lib

Ausgangsdateien Binärdatei: *.obj

Codeerzeugung

Der Linker bindet unterschiedliche Objektdateien und Bibliotheksdateien zu einem fertigen Programm zusammen. Das Ergebnis wird meist in eine Intel-Hex-Datei exportiert.

Intel-Hex-File: *.hex (*.elf)

- :1000000075812F120199120252
- :10001000027B90005B12028575
- :100020001200691203271200E3
- :100030000E12022DE59012037A

Download/

Debugging

Linker

Der Debugging-Prozess (Programmtest mit Fehlersuche) kann auf der Zielhardware oder in einem Simulator erfolgen. Er ist nicht mehr Teil des Make

Download der hex-Datei auf die Zielhardware. Z.B. über virtuellen COM-Port (USB)