Embedded Software Engineering 1

HS 2024 – Prof. Reto Bonderer

Autoren: Laurin Heitzer, Simone Stitz

https://github.com/P4ntomime/EmbSW1

V1.0.20241210

Inhaltsverzeichnis

1 Hardware Abstraction Layer (HAL)

2 Inline-Funktionen in C

2.1 Kosten einer Funktion

- Code einer Funktion ist nur einmal im Speicher vorhanden
  - Vorteil: spart Speicher
- Aufrug einer Funktion bewirkt zeitliche Einbusse im Vergleich zu direkter Befehlsausführung
  - Nachteil: Zeitverlust, Overhead

⇒ Bei sehr kleinen Funktionen, z.B. Einzeilern, (welche oft aufgerufen werden), lohnt sich der Overhead für den Funktionsaufruf oft nicht.

2.2 C-Makros

- **Reine Textersetzung ohne jegliche Typenprüfung**
- Bei Nebeneffekten (z.B. `++i`) verhalten sich Makros oft nicht wie beabsichtigt
  - Nebeneffekte sollten generell vermieden werden

⇒ Makros sollten **nicht** eingesetzt werden!
⇒ Makros lösen das Overhead-Problem

Beispiel: Maximum zweier int-Zahlen mit Makros

```c
// file: main.c
#define MAX(a,b) ((a)>(b) ? (a) : (b))

int main(void)
{
  int z1 = 4;
  int z2 = 6;
  int m = MAX((z1, z2));
  // expectation: m = 6, z1 = 4, z2 = 6
  // true values: m = 6, z1 = 4, z2 = 6

  int m = MAX((++z1, ++z2));
  // expectation: m = 7, z1 = 5, z2 = 7
  // true values: m = 8, z1 = 5, z2 = 8
  return 0;
}
```

Erklärung:

```c
m = MAX((++z1, ++z2)); // expanded to
m = ((++z1)>(++z2) ? (++z1) : (++z2));

// plug in values from example
// z1 = 4, z2 = 6
m = ((5) > (7) ? (++z1) : (8));
// z2 is incremented twice!

// -> m = 8, z1 = 5, z2 = 8
```

2.3 Inline-Funktionen

- Lösen Overhead-Problem ⇒ Code wird direkt eingefügt
- **Typenprüfung** findet statt
- Inline-Funktionen **müssen in Header-Files definiert sein**, damit der Compiler auch Inlning macht
  - Inlining wird nur gemacht, wenn dem Compiler auch eine **Optimierungsstufe** mitgegeben wird z.B. `clang -c -03 foo.c`
- Wenn Funktionen `static` deklariert werden, wird garantiert, dass Funktionen nicht auch noch im Objectfile als Funktion vorhanden sind

Achtung: Rekursive Funktionen und Funktionen, auf die mit einem Funktionspointer gezeigt wird, werden **nicht** inlined!

Beispiel: Maximum zweier int-Zahlen mit inline-Funktion

```c
// file: header.h
static inline int max(int a, int b)
{
    return a > b ? a : b;
}
```

```c
// file: main.c
#include "header.h"

int main(void)
{
  int z1 = 4;
  int z2 = 6;
  int m = max((z1, z2));
  // as expected: m = 6, z1 = 4, z2 = 6

  int m = max((++z1, ++z2));
  // as expected: m = 7, z1 = 5, z2 = 7
  return 0;
}
```