# Mikrocontrollerspezifische C-Erweiterungen

*Wilhelm Brezovits*

ANSI-C-Compiler für Mikrocontroller verfügen über Erweiterungen, um einen effizienten und vollständigen Zugriff auf die Architektur des Bausteins zu ermöglichen.

Diese Erweiterungen sind natürlich nicht (wie ANSI-C) genormt.

Das bedeutet, dass die Implementierung dieser Erweiterungen - also die entsprechende Syntax, von Compiler zu Compiler abweichen darf – und dies auch tut.

## Erweiterungen

Die Erweiterungen lassen sich generell in drei Gruppen zusammenfassen:

1. zusätzliche Datentypen
2. Steuerwörter
3. Intrinsic/Builtin - Funktionen

### 1. zusätzliche Datentypen

- **Bits**

Im Adressraum der Maschine befinden sich bitadressierbare Bereiche.

Der Befehlsvorrat vom Mikroprozessor im Mikrocontroller unterstützt mit einer eigenen Bitverarbeitungshardware die Bitmanipulation der Bits in den bitadressierbaren Bereichen (boolsche Operationen, Bit-Setzen/Löschen, Sprünge wenn Bit gesetzt/nicht gesetzt, ...).

Um diese Bitbefehle zu nutzen, verfügen Mikrocontroller Compiler über den Datentyp `bit`

`bit`/`bit` (Implementierung: KEIL/TASKING-Syntax).

- **SFR** (Special Function Register)

Die Architektur (Mikroprozessor, Betriebssystem on Silicon – Interruptsystem und die On-Chip-Peripherals) des Mikrocontrollers wird durch sogenannte SFR dargestellt.

Hinter SFR verstecken sich also die Register der CPU, des Interruptsystems aber auch der

On-Chip-Peripherals (wie z.B. Port-Zustände, Timer-Control-Register, AD-Wandler Konfigurations- oder Ergebnisregister, u.s.w.).

Aus Programmierersicht können SFR als „Schnittstelle“ zur „Hardware“ des Mikrocontrollers betrachtet werden.

Der Befehlsvorrat des Mikrocontrollers unterstützt den Zugriff auf die SFR. So kann z. B. mit einem Befehl ein SFR gelesen, manipuliert und zurückgeschrieben werden.

Aus Sicht von ANSI-C handelt es sich bei den SFR im Prinzip um den Datentyp „`unsigned int volatile`“, mit der Eigenschaft, eine „fixe Adresse“ im Adressraum zu haben (SFR-Bereich).

Zur Unterstützung der Architektur des Bausteins verfügen Mikrocontroller Compiler über den Datentyp `sfr` und `sbit` / `sfr`, `esfr` und `sfrbit`, `esfrbit` (Implementierung: KEIL/TASKING-Syntax).

### 2. Steuerwörter

Um für eine Variable eine bestimmte Adresse (einen bestimmten Platz im Zielspeicher) im internen oder externen Speicher (im Adressraum) festzulegen, ein bestimmtes Adressierungsschema - abweichend vom Speichermodell in dem übersetzt wird auszuwählen oder eine bestimmte Größe von Feldern definieren zu dürfen - gibt es Steuerwörter.

z.B.: `idata`/`iram`, `bdata`/`bitword`, `sdata`/`system`, `near`/`near`, `far`/`far`, `huge`/`shuge`, `xhuge`/`huge`, (Implementierung: KEIL/TASKING-Syntax)

Auch gibt es Steuerwörter für die Definition einer Interrupt Service Routine oder für Registerbankunterstützung (`interrupt`, `using`).

### 3. Intrinsic/Builtin - Funktionen

Der Mikrocontroller verfügt über Befehle wie z.B. `IDLE` (Enter_Idle_Mode) oder `SRVWDT` (Service_Watchdog_Timer).

Die Hochsprache (ANSI-C) gestattet es nicht, solche Befehle abzusetzen, da dafür keine Syntax vorgesehen ist – die Programmiersprache C ist ja hardwareunabhängig und kennt solche Befehle nicht.

Damit der Programmierer Zugriff auf solche bausteinspezifische Befehle hat, gibt es die intrinsic-/builtin - Funktionen:

einige Beispiele:

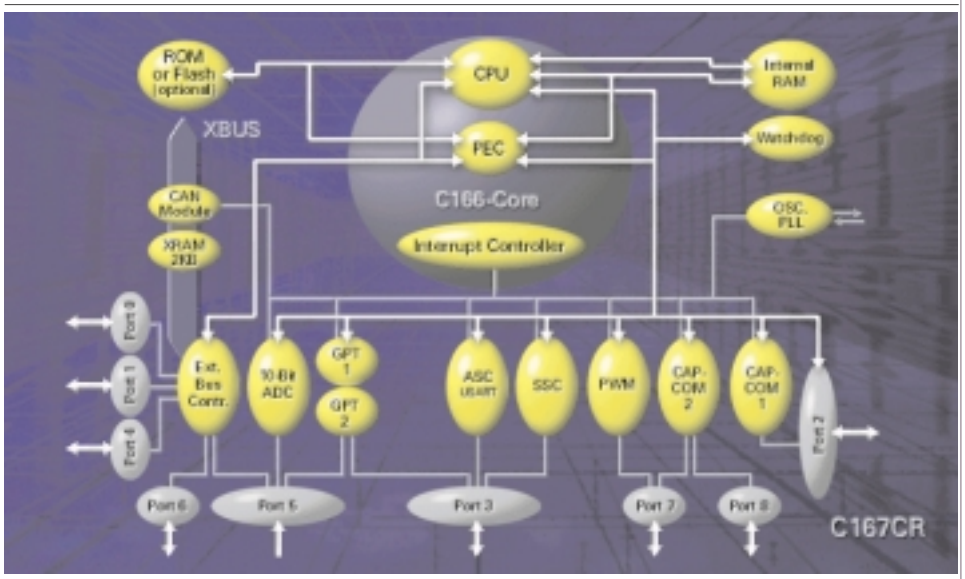
`_idle_()`/`_idle()`, `_srvwdt_()`/`_srvwdt()`, `_trap_(0)`/`_int166(0)`, `_nop_()`/`_nop()`, `_pwrdn_()`/`_pwrdn()`, (Implementierung: KEIL/TASKING-Syntax)

## Schlussbemerkung

Solange wir die mikrocontrollerspezifischen Erweiterungen des C Compilers nicht nutzen, programmieren wir portabel, haben aber auch keinen Zugriff auf die Architektur.

Wir werden also bewusst alle Erweiterungen nutzen, um effizienten und vollständigen Zugriff auf die Architektur des Mikrocontrollers zu haben, dadurch sind unsere Programme aber nicht mehr portabel. Genau das wollen wir aber, wir programmieren für Embedded Systems (anwendungsspezifische Software in anwendungsspezifischer Hardware).

*Architektur eines Mikrocontrollers*





Wilhelm Brezovits wilhelm.brezovits@siemens.at

# REKIRSCH - 2