Die wichtigsten Elemente von C im Überblick
Fett geschriebene Wörter sind reservierte Wörter, *kursiv* geschriebene Wörter sind Platzhalter

1. Datentypen

1. a) Schlüsselwörter zur Vereinbarung von Bezeichnern für Hardware-Komponenten

Schlüsselwort	**adressierte Hardware**	**Beispiele**
at		siehe nächste Zeilen
sbit	1 Bit (I/O-Bit ≡ Port-Bit oder Konfigurations-Bit oder HW-Melde-Bit = "Flag")	**at** 0x80 **sbit** P0_0; **at** 0xB7 **sbit** P3_7; **at** 0xB7 **sbit** SCL;
sfr	8 Bit (Register)	**at** 0x80 **sfr** P0;

Adressbereich 128 .. 255 = 0x80 .. 0xFF

1. b) Schlüsselwörter zur Vereinbarung von Variablen im Arbeitsspeicher

Schlüsselwort	**Speicherbedarf /Bit**	**Wertebereich**	**Bemerkungen**	**Beispiele**
bit	1	0, 1	kein Array möglich	**bit** gefunden = 0;
[unsigned] char	8	0 .. 255	interpretierbar als Zahl oder ASCII-Code	**char** Zaehler = 12; **char** Zeichen = "x"; **char** Wort[5] = "Wort"; /* Zeichenkette (String) aus 4 (!) Zeichen */
signed char	8	-128 .. 127		**signed char** x = −100;
unsigned int	16	0 .. 65535		**unsigned int** Anzahl = 7;
[signed] int	16	−32768..+32767		**int** Index = −8;
unsigned long	32	0 .. $\approx 4 \cdot 10^9$		**unsigned long** X = 4000000000;
[signed] long	32	$\approx -2 \cdot 10^9$ .. $\approx 2 \cdot 10^9$		**long** Schulden = −1000000000;
float	32	$\approx -10^{38}$ .. -10^{-38}, 0, 10^{-38} .. 10^{38}		**float** Epsilon0 = 8.854E−12;
double	48	$\approx -10^{308}$ .. -10^{-308}, 0, 10^{-308} .. 10^{308}		**double** psi = 2E−137;

[...] ≡ optionale Angabe
Der Typ mit der höchsten Komplexität innerhalb eines Ausdrucks bestimmt den Typ des Ergebnisses. Beispiel: Index / 3 hat den Typ int; Epsilon0 / 3 hat den Typ float.
Zahlen mit Nachkommastellen (float und double) weichen im Allgemeinen geringfügig vom exakten Wert ab; dadurch kann zum Beispiel eine Bereichsprüfung statt der Prüfung auf exakte Gleichheit notwendig sein.

2. Zeichen

2a. Operatoren, nach absteigender Priorität geordnet (1 = höchste, 15 = niedrigste)

Zeichen	Bedeutung	Priorität	Beispiel
( )	Klammerung; Übergabe der Argumente an Funktionen	1	x = y * (4 + 7); z = sin(0.25 * pi);
[]	Indizierung eines Array-Elementes	1	a = Sudoku[2, 7];
!	logisches NICHT	2	P1_1 = !P1_1;
++	Inkrement; als Präfix oder als Postfix	2	x = b++; // x = b; b = b + 1;
––	Dekrement; als Präfix oder als Postfix	2	x = --b; // x = b − 1; b = b − 1;
+	einstelliges Plus (Vorzeichen)	2	x = +a;
–	einstelliges Minus (Vorzeichen)	2	x = −a;
*	Multiplikation	3	x = a * b;

/	Division	3	x = 25 / 4; //x == 6
			x = 25E0 / 4; //x == 6.25E0

%	Restbildung (modulo)	3	x = 25 % 4; //x == 1
+	Addition	4	
–	Subtraktion	4	
<<	Linksschieben; nicht auf float anwendbar	5	a =0xE3; a = a << 2; //a == 0x8C
>>	Rechtsschieben; nicht auf float anwendbar	5	a =0xE3; a = a >> 4; //a == 0x0E
<	kleiner als; Ergebnis true oder false	6	5 < 7 // == true
<=	kleiner gleich	6	9 <= 7 // == false
>	größer als	6	"C" > "A" // == true
>=	größer gleich	6	5 >= 5 // == true
==	gleich	7	5 == 7 // == false
!=	ungleich	7	5 != 7 // == true
~	bitweises NICHT	?	~0xB6 // == 0x49
&	bitweises UND	8	0xF3 & 0x3E // == 0x32
^	bitweise Antivalenz (exklusives ODER)	9	0xF3 ^ 0x3E // == 0xCD
|	bitweises ODER	10	0xF3 | 0x3E // == 0xFF
&&	logisches UND	11	1 < 3 && 5 > 7 // == false
||	logisches ODER	12	1 < 3 || 5 > 7 // == true
=	Zuweisung	14	x = 3;

2b. Sonstige Zeichen

Zeichen	Bedeutung	Beispiel
{ }	Kennzeichnung eines Anweisungsblocks	
;	Abschluss einer Anweisung (nicht hinter Funktionsköpfen und Anweisungsblöcken)	x = 73; aber: **void main**()
#	Präfix von Präprozessor-Anweisungen	**#include** <reg52m.h>
"	Markierung von Zeichen und Zeichenketten	"A", "Hallo"
/*	Markierung des Beginns eines Kommentars	/* Hier beginnt ein Kommentar
*/	Markierung des Endes eines Kommentars	und hier endet er */
//	Markierung des Rests der Zeile als Kommentar	// Ende-Mark. nicht erforderlich

3. Reservierte Wörter, die nicht unter 1. und 4. vorkommen

Zeichen	Bedeutung	Beispiel
break	beendet die Ausführung der Konstrukte mit den Anweisungen **for**, **while**, **switch** und **do** ... **while**	**while** (a < 8) {i = i + a++; if (i>99) **break**;}
#include	Einbinden von Textdateien vor dem Compilieren	**#include** <reg52m.h>
main	Name des Hauptprogramms	**void main**() {*Anweisungen*}
return()	Rückkehr zum aufrufenden Progr. mit Rückgabewert	**if** (Nenner == 0) **return** (–1);
static	lokale Variable wird beim ersten Aufruf "ihrer" Funktion initialisiert und behält zwischen den Aufrufen ihren jeweiligen Wert	**static int** i = 0;
void	statt Typangabe bei Funktionen ohne Rückgabewert	**void** auslesen()

4. Konstrukte

4a. Datenstrukturen

Syntax	Bedeutung	Beispiel
Typ Bezeichner[*Index*];	Vereinbarung eines Arrays; Indexbereich beginnt immer mit 0	**int** Zahlen[10]; Indexbereich = 0..9 (!)
Bezeichner[*Index*]	Zugriff auf ein Element eines Arrays	z = Zahlen[5];

struct *Bezeichner1* {
Typ Bezeichner2;
... (usw.)
};
	Vereinbarung eines Strukturtyps von Variablen (im Allgemeinen verschiedenen Typs)
	und Deklaration einer Variablen
		struct Person {
		char Name[30];
		int Alter; }
		struct Person Angest;

Bezeichner1.Bezeichner2	Zugriff auf ein Element der Struktur	Angest.Name = "Meier" Angest.Alter = 35;

4b. Ablaufsteuerungs-Konstrukte

Syntax	Bedeutung	Beispiel

if (*Bedingung*) *Anweisung*(*sblock*)
else *Anweisung*(*sblock*)
	bedingte Ausführung von Anweisungen; **else** und der folgende Anweisungsblock sind optional	**if** (N != 0)
		b = Z / N //N != 0
		else b = 1E32; //N == 0

switch (*Ausdruck*) {
case *Wert_1*: *Anweisung*(*sblock*)
case *Wert_2*: *Anweisung*(*sblock*)
...
default: *Anweisung*(*sblock*) }
	Fallunterscheidung; *Ausdruck*, *Wert_1* und *Wert_2* müssen ganzzahlig sein; die Anweisungen werden ab der Zeile ausgeführt, in der *Wert_i* == *Ausdruck* gilt; daher sind im Allgemeinen **break**-Anweisungen notwendig; **default** entspricht "**else**"	**switch** (x – y) {
		case 0: {
		x = 2 * x; //x–y==0
		break; }
		case 1: {
		x = 0.5 * x; //x–y==1
		break; }
		default: x = y; } //sonst

for (*Startwert*; *Ausführungsbedingung*; *Inkrement*)
Anweisung(*sblock*)
	Zählschleife; die Angaben in Klammern beziehen sich auf eine Zählvariable	x = 1;
		for (i = 0; i < a; i++)
		x = x * i;

4b. Ablaufsteuerungs-Konstrukte (Fortsetzung)

Syntax	**Bedeutung**	**Beispiel**

while (*Ausführungsbedingung*)
Anweisung(*sblock*)
	Schleife; wird ausgeführt, solange *Ausführungsbedingung* true ist (möglicherweise 0 Mal)	t = 0;
		while (P1_0 = 0)
		  t = t + P1_2;

do *Anweisungs*(*sblock*) **while** (*Ausführungsbedingung*);	Schleife; wird ausgeführt, solange *Ausführungsbedingung* true ist (mindestens ein Mal)	**do** x = x / 2; **while** (x >= 2);

Typ Bezeichner(*Typ Parameter*)
Anweisung(*sblock*)
	Vereinbarung einer Funktion; das erste *Typ* bezieht sich auf den Rückgabewert; *Parameter* kann auch eine Liste mehrerer Parameter sein (mit Kommata getrennt)	**int** Quadrat (**int** x)
		  return (x * x);
		
		void RSTNull()
		  P1_0 = 0;

void *Bezeichner*() **interrupt** *N*
Anweisung(*sblock*)
	Vereinbarung einer Interrupt-Routine; N muss im Bereich {0..5} liegen und gibt an, welchem Interrupt die Routine zugeordnet ist:
	0 : externer Interrupt 0
	1 : Überlauf des Timers T0
	2 : externer Interrupt 1
	3 : Überlauf des Timers T1
	4 : serielle Schnittstelle
	5 : Überlauf des Timers T2 und externer Interrupt 2
		void T2_IR() **interrupt** 5
		{
		  TF2 = 0;
		  P0_5 = !P0_5;
		}

5. Weitere reservierte Wörter

asm, auto, code, data, enum, goto, register, short, typedef, union, volatile