

**Befehlssatz der  
Mikrocontroller der  
51er -Familie**



# Mikrocontrollerfamilie 8051

## Befehlssatz

### Abkürzungen:

<i>A</i> :	Akkumulator
<i>Rn</i> :	Register R0..R7
<i>Ri</i> :	R0 oder R1
<i>dadr</i> :	direkte Byte-Adresse im int. Speicher (auch SF-Register)
<i>badr</i> :	direkte Bit-Adresse im int. Speicher (auch SF-Flags)
<i>adr16</i> :	absolute 16-bit Adresse
<i>adr11</i> :	11-bit Adresse relative zu 2k-Seite (0 .. 2047 bzw. 0 .. 7FFH)
<i>rel</i> :	8-bit Adresse relative zu Befehl (-128 .. +127 bzw. 80H .. 7FH)
<i>konst8</i> :	8-bit Konstante (0 .. 255 bzw. 0 .. 0FFH)
<i>konst16</i> :	16-bit Konstante (0.. 65535 bzw. 0 .. 0FFFFH)
<i>#B</i> :	Anzahl der Bytes, die der Befehl belegt
<i>#Z</i> :	Anzahl der Prozessorzyklen für die Befehlsausführung

### Adressierungsarten:

**Registeradressierung:** in Befehlen, die nur mit *A*, *B*, *Rn* und *DPTR* arbeiten  
Bsp.: `MOV A,R0`

**Direkte Adressierung:** alle Adressen im internen Speicher (0..7FH und SFR)  
Bsp.: `MOV dadr,80H`  
`MOV A,SCON`

**Indirekte Adressierung (int. Speicher):** Adresse (0..FF) steht in *Ri*  
Bsp.: `MOV A,@R0`

**Indirekte Adressierung (ext. Speicher):** Adresse (0..FFFF) steht in *DPTR* oder Adresse (0..0FFH) in *Ri*  
Bsp.: `MOVX A,@DPTR`

**Indirekte, indizierte Adressierung (Programm-Speicher):** Adresse berechnet sich aus  $A + DPTR$  oder  $A + PC$   
Bsp.: `MOVC A,@A+DPTR`

**Unmittelbare Adressierung (immediate):** Die Adresse ist der Befehl selbst (Laden von Konstanten)  
Bsp.: `MOV R7,#0FFH` (8-bit Wert)  
`MOV DPTR,#8000H` (16-bit Wert)

## Transfer-Befehle (Daten kopieren):

Kopieren eines Datenbytes aus dem und in den internen Speicher / SFR:

<b>MOV</b>	<i>A,Rn</i>	1 B	1 Z	<i>dadr,A</i>	2 B	1 Z
	<i>A,dadr</i>	2 B	1 Z	<i>dadr,Rn</i>	2 B	2 Z
	<i>A,@Ri</i>	1 B	1 Z	<i>dadr,dadr</i>	3 B	2 Z
	<i>A,#konst8</i>	2 B	1 Z	<i>dadr,@Ri</i>	2 B	2 Z
				<i>dadr,#konst8</i>	3 B	2 Z
	<i>Rn,A</i>	1 B	1 Z	<i>@Ri,A</i>	1 B	1 Z
	<i>Rn,dadr</i>	2 B	2 Z	<i>@Ri,dadr</i>	2 B	2 Z
<i>Rn,#konst8</i>	2 B	1 Z	<i>@Ri,#konst8</i>	2 B	1 Z	

Kopieren eines Datenbytes aus dem und in den externen Speicher:

<b>MOVX</b>	<i>A,@Ri</i>	1 B	2 Z	<i>@Ri,A</i>	1 B	2 Z
	<i>A,@DPTR</i>	1 B	2 Z	<i>@DPTR,A</i>	1 B	2 Z

Kopieren eines Datenbytes aus dem Programmspeicher:

<b>MOVC</b>	<i>A,@A+PC</i>	1 B	2 Z			
	<i>A,@A+DPTR</i>	1 B	2 Z			

Kopieren eines Datenbits von Carry in internen Datenspeicher und umgekehrt:

<b>MOV</b>	<i>C,badr</i>	2 B	1 Z			
	<i>badr,C</i>	2 B	2 Z			

Kopieren einer 16-bit Konstanten in das DPTR-Register:

<b>MOV</b>	<i>DPTR,#konst16</i>	3 B	2 Z			
------------	----------------------	-----	-----	--	--	--

## Transfer-Befehle (Daten austauschen):

Austausch von Datenbytes zwischen Akkumulator und internem Speicher:

<b>XCH</b>	<i>A,Rr</i>	1 B	1 Z	
	<i>A,dadr</i>	2 B	1 Z	
	<i>A,@Ri</i>	1 B	1 Z	

Austausch der Datenbits 0..3 zwischen Akkumulator und Register:

<b>XCHD</b>	<i>A,@Ri</i>	1 B	1 Z	
-------------	--------------	-----	-----	--

Austausch der Datenbits 0..3 und 4..7 im Akkumulator:

<b>SWAP</b>	<i>A</i>	1 B	1 Z	
-------------	----------	-----	-----	--

## Transfer-Befehle (Daten von/nach Stack kopieren):

Kopieren eines Datenbytes vom internen Datenspeicher / SFR in den Stack:

<b>PUSH</b>	<i>dadr</i>	2 B	2 Z	
-------------	-------------	-----	-----	--

Kopieren eines Datenbytes aus dem Stack in den internen Datenspeicher / SFR:

<b>POP</b>	<i>dadr</i>	2 B	2 Z	
------------	-------------	-----	-----	--

## Sonstige Befehle:

Leerbefehl (No operation):

<b>NOP</b>		1 B	1 Z	
------------	--	-----	-----	--

## Logik-Befehle mit Bytes:

Das Ergebnis der logischen Operation steht immer im linken Operanden\*:

<b>ANL</b>	<i>A, Rn</i>	1 B	1 Z	<i>dadr, A</i>	2 B	1 Z
<b>ORL</b>	<i>A, dadr</i>	2 B	1 Z	<i>dadr, #konst8</i>	3 B	2 Z
<b>XRL</b>	<i>A, @Ri</i>	1 B	1 Z			
	<i>A, #konst8</i>	2 B	1 Z			

Invertieren des Akkumulator-Inhalts:

<b>CPL</b>	<i>A</i>	1 B	1 Z	
------------	----------	-----	-----	--

Löschen des Akkumulator-Inhalts\*:

<b>CLR</b>	<i>A</i>	1 B	1 Z	
------------	----------	-----	-----	--

\*: Wenn das Ziel der Akkumulator ist, wird im PSW das *P*-Bit beeinflusst.

## Logik-Befehle mit Bits:

Das Ergebnis der logischen Operation steht immer Carry:

<b>ANL</b>	<i>C, badr</i>	2 B	2 Z	
<b>ORL</b>	<i>C, /badr</i>	2 B	2 Z	

Invertieren eines Bits:

<b>CPL</b>	<i>C</i>	1 B	1 Z	
	<i>badr</i>	2 B	1 Z	

Löschen eines Bits

<b>CLR</b>	<i>C</i>	1 B	1 Z	
	<i>badr</i>	2 B	1 Z	

Setzen eines Bits

<b>SETB</b>	<i>C</i>	1 B	1 Z	
	<i>badr</i>	2 B	1 Z	

## Arithmetik-Befehle:

Das Ergebnis der arithmetischen Operation steht immer im Akkumulator:

<b>ADD</b>	<i>A,Rn</i>	1 B	1 Z	
<b>ADDC</b>	<i>A,dadr</i>	2 B	1 Z	
<b>SUBB</b>	<i>A,@Ri</i>	1 B	1 Z	
	<i>A,#konst8</i>	2 B	1 Z	

Im PSW werden entsprechend dem Ergebnis die Bits *C,OV,AC* und *P* angepasst.

ADD(C): C = 1: Ergebnis >255

SUBB: C = 1: [Dest-Byte] < [Src-Byte] (z.B. [A] < konst8)

C = 0: [Dest-Byte] ≥ [Src-Byte] (z.B. [A] ≥ konst8)

Das jeweilig adressierte Datenbyte wird um 1 erhöht oder erniedrigt:

<b>INC</b>	<i>A</i>	1 B	1 Z	
<b>DEC</b>	<i>Rn</i>	1 B	1 Z	
	<i>@Ri</i>	1 B	1 Z	
	<i>dadr</i>	2 B	1 Z	

Das DPTR-Register (16 Bit) wird um 1 erhöht:

<b>INC</b>	<i>DPTR</i>	1 B	1 Z	
------------	-------------	-----	-----	--

Der Akkumulator *A* wird mit Register *B* multipliziert - Ergebnis in *B,A*:

<b>MUL</b>	<i>AB</i>	1 B	4 Z	
------------	-----------	-----	-----	--

Im PSW wird *OV* gesetzt, wenn das Ergebnis größer als 255 ist, *C* ist immer 0.

Der Akkumulator *A* wird durch Register *B* dividiert - Quotient in *A*, Rest in *B*:

<b>DIV</b>	<i>AB</i>	1 B	4 Z	
------------	-----------	-----	-----	--

Im PSW wird *OV* gesetzt, wenn *B = 0* ist, *C* ist immer 0.

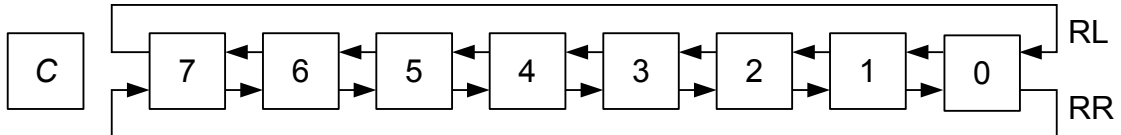
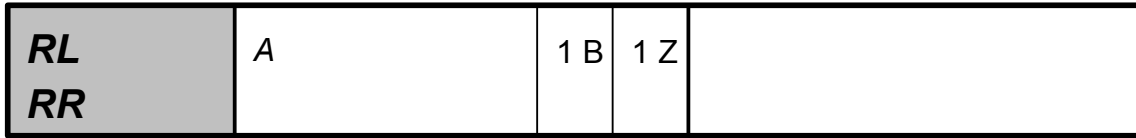
Dezimalkorrektur bei einer Addition von BCD-Zahlen in *A*:

<b>DA</b>	<i>A</i>	1 B	1 Z	
-----------	----------	-----	-----	--

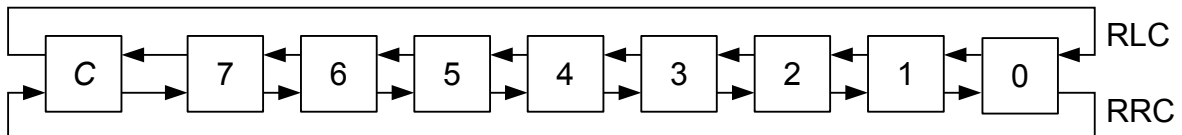
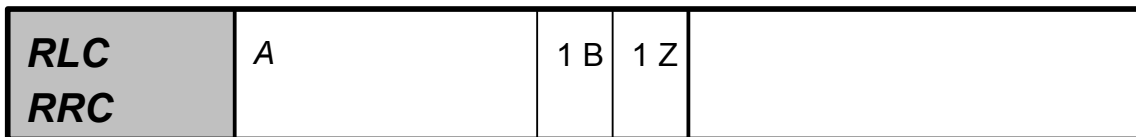
Im PSW wird *C*-Bit (Carry) angepasst.

## Schiebe-Befehle

Rotiert alle Bits des Akkumulators um eins nach links oder rechts (ohne C):



Rotiert alle Bits des Akkumulators mit Carry um eins nach links oder rechts:





## Unbedingte Sprung-Befehle:

Sprung an eine absolute 16-bit Adresse:

<b>LJMP</b>	<i>adr16</i>	3 B	2 Z	
-------------	--------------	-----	-----	--

Sprung an eine relative 11-bit Adresse innerhalb einer 2k-Seite:

<b>AJMP</b>	<i>adr11</i>	2 B	2 Z	
-------------	--------------	-----	-----	--

Sprung an eine zum Befehl relative Adresse (-128 .. +127):

<b>SJMP</b>	<i>rel</i>	2 B	2 Z	
-------------	------------	-----	-----	--

Sprung an eine indirekte und indizierte Adresse:

<b>JMP</b>	<i>@A+DPTR</i>	1 B	2 Z	
------------	----------------	-----	-----	--

## Unbedingter Sprung in eine Unterroutine:

Sprung an eine absolute 16-bit Adresse:

<b>LCALL</b>	<i>adr16</i>	3 B	2 Z	
--------------	--------------	-----	-----	--

Sprung an eine relative 11-bit Adresse innerhalb einer 2k-Seite:

<b>ACALL</b>	<i>adr11</i>	2 B	2 Z	
--------------	--------------	-----	-----	--

## Rücksprung aus einer Unterroutine:

Rücksprung aus einer normalen Unterroutine:

<b>RET</b>		1 B	2 Z	
------------	--	-----	-----	--

Rücksprung aus einer Interruptroutine:

<b>RETI</b>		1 B	2 Z	
-------------	--	-----	-----	--

## Bedingte Sprung-Befehle:

Bedingter relativer Sprung in Abhängigkeit des Carry-Bits:

<b>JC</b> <b>JNC</b>	<i>rel</i>	2 B	2 Z	
-------------------------	------------	-----	-----	--

Bedingter relativer Sprung in Abhängigkeit vom Akkumulatorinhalt (=0 oder ≠0):

<b>JZ</b> <b>JNZ</b>	<i>rel</i>	2 B	2 Z	
-------------------------	------------	-----	-----	--

Bedingter relativer Sprung in Abhängigkeit von einem Bit:

<b>JB</b> <b>JNB</b> <b>JBC</b>	<i>badr,rel</i>	3 B	2 Z	
---------------------------------------	-----------------	-----	-----	--

Bedingter relativer Sprung, wenn Akkumulator ungleich Datenbyte oder Akkumulator/Register/Datenbyte ungleich Konstante ist:

<b>CJNE</b>	<i>A,dadr,rel</i>	3 B	2 Z	
	<i>A,#konst8,rel</i>	3 B	2 Z	
	<i>Rn,#konst8,rel</i>	3 B	2 Z	
	<i>@Ri,#konst8,rel</i>	3 B	2 Z	

C = 1: [Dest-Byte] < [Src-Byte] (z.B. [A] < konst8)

C = 0: [Dest-Byte] ≥ [Src-Byte] (z.B. [A] ≥ konst8)

Das Register/Datenbyte wird um 1 erniedrigt, bedingter relativer Sprung, wenn ≠0:

<b>DJNZ</b>	<i>Rn,rel</i>	2 B	2 Z	
	<i>dadr,rel</i>	3 B	2 Z	