

## 9. Die Adressierungsarten des MSP 430

- 9.1 Übersicht über die Adressierungsarten
- 9.2 Register-Operanden
- 9.3 Indexregister mit Distanz
- 9.4 Symbolische Adresse (relativ zum PC)
- 9.5 Absolute Adresse
- 9.6 Indirekte Adresse
- 9.7 Indirekte Adressierung mit Postinkrement
- 9.8 Konstante als Operand
- 9.9 Beispiele für die Programmierung mit dem Stackpointer

### 9.1 Übersicht über die Adressierungsarten

Unterschiedliche Adressierungsarten erlauben es, den Speicher in unterschiedlicher Art und Weise mit Hardwareunterstützung zu lesen und zu schreiben. **Einadress-Maschinen** haben oft nur zwei explizite Adressierungsbefehle, LOAD und STORE, die den Speicher benutzen; alle anderen Operationen können nur auf die Register zugreifen. **Zweiadress-Maschinen** erlauben in der Regel sowohl Operanden in Registern als auch Operanden im Speicher. Mit *einem* Befehl können *zwei* Operanden geladen, miteinander verknüpft und an einer der beiden Adressen wieder *abgespeichert* werden (Beispiel: ADD src,dst).

Der MSP430 kennt **sieben Adressierungsarten**. Alle sieben können für den Quelloperanden in Zweiadressbefehlen und für den Operanden in Einadressbefehlen verwendet werden, aber nur vier davon für den Zieloperanden in Zweiadressbefehlen.

## Tabelle der Adressierungsarten

As/Ad	Adressierungsart	Syntax	Beschreibung
00/0	Register	Rn	der Operand steht im Register
01/1	Indexregister mit Distanz (indexed)	X(Rn)	(Rn + X) zeigt auf den Operanden
01/1	symbolisch	ADDR	(PC + X) zeigt auf den Operanden (Adressierung relativ zum Befehlszähler)
01/1	absolut	&ADDR	Das Speicherwort nach dem Befehl enthält die absolute Adresse des Op.
10/-	indirekt	@Rn	Rn wird als Zeiger auf den Operanden benutzt (Adressregister)
11/-	indirekt mit Postinkrement	@Rn+	Rn wird als Zeiger auf den Operanden benutzt, danach wird Rn inkrementiert
11/-	immediate	#N	Das Wort nach dem Befehl enthält die Konstante N.

## 9.2 Register-Operanden

### Assembler-Code

```
MOV R10, R11
```

### Beschreibung

Kopiere den Inhalt von R10 nach R11; R10 bleibt unverändert

	Vorher	Nachher
R10	0A023h	0A023h
R11	0FA15h	0A023h
PC	PC <sub>old</sub>	PC <sub>old</sub> +2

## 9.3 Indexregister mit Distanz (1)

### Assembler-Code

```
MOV 2(R5), 6(R6)
```

### Beschreibung

Kopiere den Inhalt von der Quelladresse (Inhalt von R5+2) zur Zieladresse (Inhalt von R6+6). Die Quell- und Zielregister (R5 und R6) werden nicht verändert.

## Indexregister mit Distanz: Beispiel

### Vorher:

Adresse	Register
0FF16h   00006h	R5   01080h
0FF14h   00002h	R6   0108Ch
0FF12h   04596h	PC
01094h   0xxxxh	0108Ch
01092h   05555h	+00006h
01090h   0xxxxh	01092h
01084h   0xxxxh	01080h
01082h   01234h	+00002h
01080h   0xxxxh	01082h

### Nachher:

Adresse	Register
0xxxxh   PC	R5   01080h
0FF16h   00006h	R6   0108Ch
0FF14h   00002h	
0FF12h   04596h	
01094h   0xxxxh	
01092h   01234h	
01090h   0xxxxh	
01084h	
01082h	
01080h	

## 9.4 Symbolische Adresse (relativ zum PC)

### Assembler-Code

MOV EDE, TONI

entspricht:

MOV x(PC), y(PC)

### Beschreibung

Kopiere den Inhalt von der Quelladresse EDE (Inhalt von PC+x) an die Zieladresse TONI (Inhalt von PC+y). Die Speicherwörter nach der Instruktion enthalten die gewünschte *Distanz* (Offset, relative Adresse) zwischen dem PC und dem Quell- bzw. Zieloperanden. Der Assembler berechnet beim Assemblieren die Offsets x und y automatisch und fügt sie in den Befehl ein.

## Symbolische Adresse: Beispiel

### Vorher:

Adresse	Register
0FF16h	011FEh
0FF14h	0F102h
0FF12h	04090h
	PC
0F018h	0xxxxh
0F016h	0A123h
0F014h	0xxxxh
01116h	0xxxxh
01114h	01234h
01112h	0xxxxh

### Nachher:

Adresse	Register
	0xxxxh
0FF16h	011FEh
0FF14h	0F102h
0FF12h	04090h
	PC
0F018h	0xxxxh
0F016h	0A123h
0F014h	0xxxxh
01116h	0xxxxh
01114h	0A123h
01112h	0xxxxh

## 9.5 Absolute Adresse

### Assembler-Code

MOV &EDE, &TONI

entspricht:

MOV x(0), y(0)

### Beschreibung

Kopiere den Inhalt von der Quelladresse EDE zur Zieladresse TONI. Die Speicherwörter nach der Instruktion enthalten die *absolute* Adresse des Quell- bzw. Zielooperanden.

## Absolute Adresse: Beispiel

### Vorher:

Adresse	Register
0FF16h	01114h
0FF14h	0F016h
0FF12h	04292h
	PC
0F018h	0xxxxh
0F016h	0A123h
0F014h	0xxxxh
01116h	0xxxxh
01114h	01234h
01112h	0xxxxh

### Nachher:

Adresse	Register
	0xxxxh
0FF16h	01114h
0FF14h	0F016h
0FF12h	04292h
	PC
0F018h	0xxxxh
0F016h	0A123h
0F014h	0xxxxh
01116h	0xxxxh
01114h	0A123h
01112h	0xxxxh

## 9.6 Indirekte Adresse

### Assembler Code

```
MOV.B @R10, 0(R11)
```

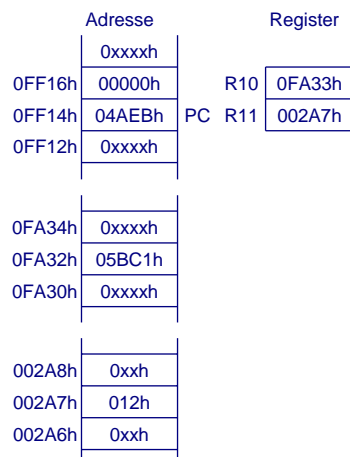
### Beschreibung

Kopiere den Inhalt von der Quelladresse (R10 als Adressregister) zur Zieladresse (R11 als Adressregister). Die Register werden nicht verändert.

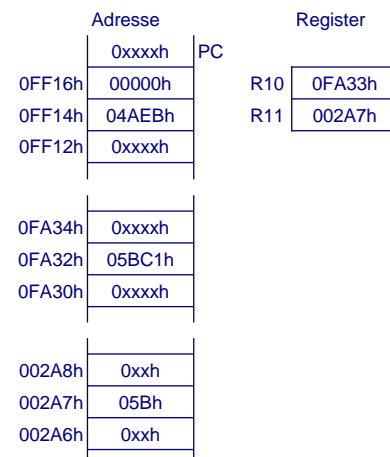
Bei 2-Operanden-Befehlen nur für den Quelloperanden möglich.

## Indirekte Adresse: Beispiel

### Vorher:



### Nachher:



## Register (indirekt)

@Rn Register n enthält die *Adresse* des Operanden

**Beispiel:** `mov #0FFFEh, R15 // lädt Register 15 mit Adresse FFFEh`  
`mov @R15, R5 // schreibt den Wert an der`  
`// Speicheradresse FFFEh in das`  
`// Register 5`

### Register (indexiert)

`offset(Rn)` die Adresse des Operanden ist die Summe aus dem Inhalt von Rn und dem Offset

**Beispiel:** `mov #0FFFFh, R15 // lädt Reg. 15 mit Adresse`  
`// (bzw. Zahl) FFFFh`  
`mov #00123h, -1(R15) // schreibt 123h ab Adresse FFFEh`

## 9.7 Indirekte Adressierung mit Postinkrement

### Assembler-Code

```
MOV @R10+, 0(R11)
```

### Beschreibung

Kopiere den Inhalt von der Quelladresse (Inhalt von R10) zur Zieladresse (Inhalt von R11). Register R10 wird bei einer Byte-Operation um 1, bei einer Wort-Operation um 2 inkrementiert. Das Inkrementieren erfolgt nach der Operation; das Register (hier R10) zeigt auf die nächste Adresse im Speicher, ohne einen zusätzlichen Befehl zu erfordern. Dies ist zum Beispiel bei Verarbeitung von Arrays sehr nützlich.

Bei 2-Operanden-Befehlen nur für den Quelloperanden möglich.

## Indirekte Adressierung mit Postinkrement: Beispiel 1

Vorher:

Adresse	Register
0FF18h   0xxxxh	
0FF16h   00000h	R10   0FA32h
0FF14h   04ABBh	PC R11   010A8h
0FF12h   0xxxxh	
0FA34h   0xxxxh	
0FA32h   05BC1h	
0FA30h   0xxxxh	
010AAh   0xxxxh	
010A8h   01234h	
010A6h   0xxxxh	

Nachher:

Adresse	Register
0FF18h   0xxxxh	PC
0FF16h   00000h	R10   0FA34h
0FF14h   04ABBh	R11   010A8h
0FF12h   0xxxxh	
0FA34h   0xxxxh	
0FA32h   05BC1h	
0FA30h   0xxxxh	
010AAh   0xxxxh	
010A8h   05BC1h	
010A6h   0xxxxh	

Das Autoinkrement des Registerinhalts erfolgt **nach** dem Holen des Operanden.

Programmierkurs II

© Wolfgang Effelsberg

9. Adressierungsarten  
des MSP 430

9 - 15

## Indirekte Adressierung mit Postinkrement: Beispiel 2

```

mov #01234h, &00200h // schreibt 1234h ab Speicherzelle 200h
mov #00200h, R7      // lädt Register 7 mit Wert (Adresse) 200h
mov @R7+, R1         // lädt Register R1 mit 1234h
                    // danach steht 202h in R7
    
```

Was bewirkt folgender Ausdruck? `add @R15+, -2(R15)`

Der Ausdruck ist äquivalent zu den Anweisungen `add @R15, 0(R15)`  
`add #2, R15`

Programmierkurs II

© Wolfgang Effelsberg

9. Adressierungsarten  
des MSP 430

9 - 16



## 9.8 Konstante als Operand (immediate)

### Assembler-Code

MOV #45h, TONI

### Beschreibung

Kopiere die Konstante 45h, die sich im Wort nach der Instruktion befindet, an die Zieladresse TONI.

Bei 2-Operanden-Befehlen nur für den Quelloperanden möglich.

## Konstante als Operand: Beispiel

### Vorher:

Adresse	Register
0FF16h	01192h
0FF14h	00045h
0FF12h	040B0h
	PC
010AAh	0xxxxh
010A8h	01234h
010A6h	0xxxxh

### Nachher:

Adresse	Register
0FF18h	0xxxxh
0FF16h	01192h
0FF14h	00045h
0FF12h	040B0h
	PC
010AAh	0xxxxh
010A8h	00045h
010A6h	0xxxxh

## Die "Konstantenregister" R2 und R3

Register	Bits für Adressierungsart (As)	Konstante	Bemerkung
R2	00	-----	Registermodus für SR
R2	01	(0)	absolute Adressierung
R2	10	00004h	+4
R2	11	00008h	+8
R3	00	00000h	0
R3	01	00001h	+1
R3	10	00002h	+2
R3	11	0FFFFh	-1

### Vorteile dieser Art der Konstantengenerierung:

- keine speziellen Befehle erforderlich
- geringere Programmspeicheranforderungen, da kein zusätzliches Wort für die sechs meistbenutzten Konstanten erforderlich ist.
- geringere Ausführungszeit für die Befehle, da kein separater Speicherzugriff zum Holen der Konstanten nötig ist.

## 9.9 Beispiele für die Programmierung mit dem SP

### Merke: Der Stackpointer ist R1

