PIC-Grundlagen Teil 3

Der dritte Teil der Artikelserie beschäftigt sich mit der Installation des Assemblers sowie mit der ausführlichen Beschreibung der Assemblerbefehle für die PIC 16C5X-Familie.

Installation des Windows-Assemblers

Im Lieferumfang des PICStartPlus-Starterpaketes befinden sich aktuelle Softwareversionen des Editors, Assemblers und Simulators auf Diskette. Auch die Microchip CD-ROM enthält diese Programme.

Zur Installation wird aus Windows heraus das Setup-Programm „SETUP.EXE" auf der ersten Diskette mit der Aufschrift „MPLAP" gestartet.

Zu Beginn der Installation erfolgt dabei die Abfrage des Verzeichnisses, in das die Dateien kopiert werden sollen. Anschließend kann man zwischen der vollständigen und ausgewählten Installation wählen. Es empfiehlt sich, hier die vollständige Installation auszuwählen, damit alle Komponenten und die Hilfedateien installiert werden.

Am Ende der Installation erfolgt eine Abfrage, ob die Programme zum PICSTART-PLUS installiert werden sollen. Hierbei handelt es sich um das Ansteuerprogramm für das PIC-Programmiergerät. Danach schließt das Programm mit dem Anlegen einer neuen Programmgruppe die Installation ab.

In dieser Programmgruppe befindet sich der Assembler „MPASM", zum Assemblieren einzelner Dateien. Beim Programm „MPLAB" handelt es sich um den Editor, mit dem Programme erstellt und simuliert werden können. Ebenso kann von diesem Programm heraus der Assembler aufgerufen werden und auch die Ansteuerung des PIC-Programmiergerätes erfolgt von diesem Programm aus.

Diese Programme befinden sich, wie erwähnt, auch auf der Microchip-CD-ROM. Hierzu erfolgt der Aufruf des Programms „MPL31200.EXE" im Unterverzeichnis \SOFTWARE\. Die Zahlenfolge 31200 steht hierbei für die Version V3.12 und kann bei einer neueren Version abweichen. Die Installation verläuft identisch mit der zuvor beschriebenen Installation von der Diskette.

Die weitere Bedienung und Funktion des Windows-Assemblers wird später anhand eines Beispiels erklärt, nachdem wir die Assemblerbefehle ausführlich kennengelernt haben.

Datenbücher auf CD-ROM

Die Microchip-CD-ROM beinhaltet zusätzlich die Datenbücher der verschiedenen PIC-Controller sowie Hinweise und Beispiele zur Programmierung. Diese Daten sind mit Hilfe des „Acrobat Readers" darstellbar und bei Bedarf ausdruckbar.

Ist dieses Programm noch nicht auf Ihrem PC installiert, so kann es auch durch den Aufruf des Programms „SETUP.EXE" im Verzeichnis \INSTALL\WIN\DISK1\ der Microchip-CD-ROM installiert werden.

Ist der Reader fertig installiert, so erfolgt sein automatischer Start. Zunächst ist die Datei „MAIN.PDF" im Stammverzeichnis der CD zu öffnen. Daraufhin erscheint der in Abbildung 14 dargestellte Startbildschirm. Die Anwahl der einzelnen Unterpunkte erfolgt durch Doppelclick mit der Maus. Um z. B. die Datenblätter anzeigen zu lassen, ist der Menüpunkt „Data Sheets and Programming Specifications" auszuwählen.

Die Assemblerbefehle der PIC 16C5X-Familie

Bevor wir mit der Beschreibung der Assemblerbefehle beginnen, sei noch einmal kurz darauf hingewiesen, daß der PIC-Controller nur über ein Arbeitsregister (W-Register) verfügt und die Befehle auf dieses W-Register zugreifen, oder aber Operationen zwischen dem W-Register und einem anderen Register (File-Register) ausführen.

Die Assemblerbefehle können in drei Gruppen eingeteilt werden.

Die erste, folgend dargestellte Gruppe beinhaltet die Sprungbefehle und die Operationen mit Konstanten.

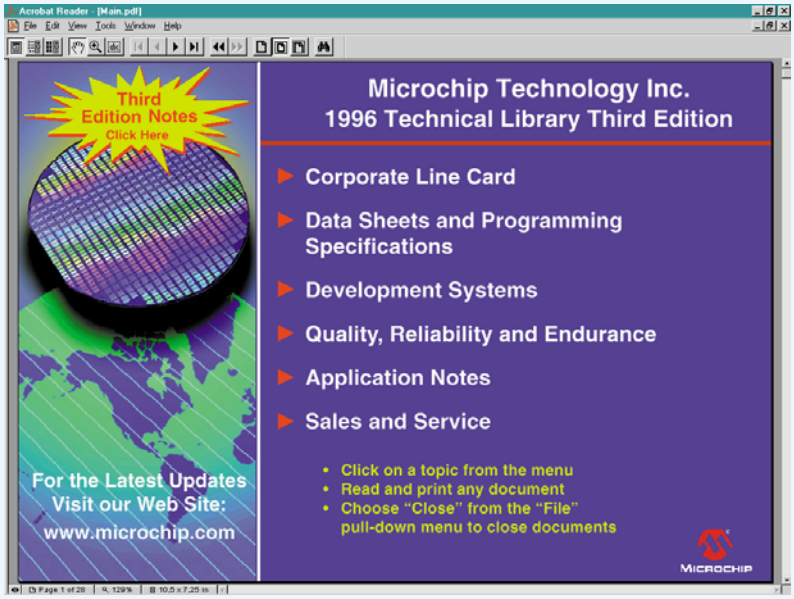

Bild 14: Ansicht des Startbildschirms der PIC-Datenbücher auf CD

1. Sprungbefehle und Operationen mit Konstanten

ANDLW

Syntax: ANDLW k
Funktion: UND-Verknüpfung einer Konstanten mit dem W-Register
Status-Flags: Z
Beschreibung: Der Befehl führt eine bitweise UND-Verknüpfung der Konstanten k und dem W-Register durch. Ist das Ergebnis der Verknüpfung Null, o wird das Zero-Bit gesetzt.
Beispiel:
```
; im W-Register steht C1h    (=11000001b)
ANDLW  B3h                  ; (=10110011b)
; im W-Register steht 81h    (=10000001b)
```

CALL

Syntax: CALL k
Funktion: Aufruf eines Unterprogramms
Status-Flags: keine
Beschreibung: Der Befehl beinhaltet den Aufruf eines Unterprogramms. Dabei erfolgt die Sicherung der aktuellen Adresse auf dem Stack, so daß an dieser Stelle nach Beendigung des Unterprogramms fortgefahren werden kann. Zu beachten ist, daß die Größe des Stacks nur zwei ineinander verschachtelte Unterprogrammaufrufe erlaubt und nur Unterprogramme in der ersten Hälfte einer Seite erreichbar sind. Die Seite wird dabei durch die Bits PA0 und PA1 des STATUS-Registers bestimmt (siehe PIC-Grundlagen Teil 2).
Beispiel:
```
; Programmablauf
CALL U_FKT                  ; Unterprogrammaufruf
; Rücksprung nach dem Unterprogramm
;                           ...

U_FKT                       ; Unterprogrammablauf
;                           ...
RETLW 00h                   ; Unterprogramm beenden
```

CLRWDT

Syntax: CLRWDT
Funktion: Löscht den Watch-Dog-Timer
Status-Flags: Timeout, Powerdown
Beschreibung: Der Watch-Dog-Timer ist bei der Programmierung des PICs aktivierbar und muß dann regelmäßig durch diesen Befehl zurückgesetzt werden. Erfolgt das Rücksetzen nicht rechtzeitig, so löst der WatchDog-Timer einen Reset aus und das Programm wird neu gestartet.

GOTO

Syntax: GOTO k
Funktion: Springe zur angegeben Position im Programm
Status-Flags: keine
Beschreibung: Mit diesem Befehl kann der Programmablauf ab jeder Position k im Programmspeicher fortgesetzt werden. Ein Rücksprung wie bei der CALL-Anweisung ist möglich. Mit dem Befehl kann jede Adresse einer Seite erreicht werden, wobei die Seite durch die Bits PA0 und PA1 des STATUS-Registers bestimmt wird (siehe PIC-Grundlagen Teil 2).
Beispiel:
```
START                       ; Programmschleife
;                           ...
GOTO START                  ; Sprung zum Anfang der
                            ; Programmschleife
```

IORLW

Syntax: IORLW k
Funktion: ODER-Verküpfung einer Konstanten mit dem W-Register
Status-Flags: Z
Beschreibung: Der Befehl führt eine bitweise ODER-Verknüpfung der Konstanten k und dem W-Register durch. Ist das Ergebnis der Verknüpfung Null, so wird das Zero-Bit gesetzt.
Beispiel:
```
; im W-Register steht C1h    (=11000001b)
IORLW  B3h                  ; (=10110011b)
; im W-Register steht F3h    (=11110011b)
```

MOVLW

Syntax: MOVLW k
Funktion: Konstante in das W-Register laden
Status-Flags: keine
Beschreibung: Der Befehl kopiert die Konstante k in das W-Register. Hierbei werden keine Status-Flags gesetzt.
Beispiel:
```
; im W-Register steht ein beliebiger Wert
MOVLW  B3h
; im W-Register steht B3h
```

OPTION

Syntax: OPTION
Funktion: Laden des OPTION-Registers mit dem W-Register
Status Flags: keine
Beschreibung: Der Befehl kopiert den Inhalt des W-Registers in das OPTION-Register zur Konfiguration des Timers und Vorteilers. Der Befehl findet nur in der PIC 16C5X-Familie Anwendung, da bei diesen Controllern nicht direkt auf das OPTION-Register zugegriffen werden kann.
Beispiel:
```
MOVLW  05h                  ; im W-Register steht 05h
OPTION                      ; 05h in das OPTION-
                            ; Register schreiben
```

RETLW

Syntax: RETLW k
Funktion: Beenden eines Unterprogramms mit Übergabe einer Konstanten im W-Register
Status-Flags: keine
Beschreibung: Der Befehl beendet ein Unterprogramm, das durch eine CALL-Anweisung aufgerufen wurde. Der Programmablauf wird direkt nach dem CALL-Aufruf fortgesetzt. Das Unterprogramm wird dabei mit einer Konstanten k, die im W-Register übergeben wird, beendet.
Beispiel:
```
; Programmablauf
CALL U_FKT                  ; Unterprogrammaufruf
; Rücksprung zum Unterprogramm
; im W-Register steht 23h
;                           ...

U_FKT                       ; Unterprogrammablauf
;                           ...
RETLW 23h                   ; Unterprogramm beenden
```

SLEEP

Syntax: SLEEP
Funktion: Schaltet den PIC in den Stand-By-Mode
Status-Flags: Timeout, Power-Down
Beschreibung: Der Befehl setzt den PIC in den Power-Down-Mode, wobei er die Befehlsabarbeitung unterbricht und alle I/O-Pins ihren Zustand beibehalten. Das Power-Down-Bit wird dabei gelöscht und das Time-Out-Bit wird gesetzt.
Der PIC kann diesen Mode nur durch einen externen Reset am /MCLR-Pin, oder über den internen Reset vom Watchdog wieder verlassen.
Beispiel:
```
; Programmablauf
SLEEP                       ; in Power-Down-Mode
                            ; wechseln
; dieser Programmcode wird nicht mehr ausgeführt
```

TRIS

Syntax: TRIS f
Funktion: Laden des TRIS-Registers mit dem W-Register
Status-Flags: keine
Beschreibung: Der Befehl kopiert den Inhalt des W-Registers in das TRIS-Register zur Konfiguration der I/O-Ports. Der Parameter f bestimmt, bei welcher Port konfiguriert werden soll.
Für den Port A ist f=5, für Port B ist f=6 und für Port C ist f=7. Der Befehl wird nur in der PIC 16C5X-Familie verwendet, da bei diesen Controllern nicht direkt auf die TRIS-Register zugegriffen werden kann.
Beispiel:
```
MOVLW  0Fh                  ; im W-Register steht 0Fh
                            (=00001111b)
TRIS  6                     ; 0Fh in das TRIS-Register
                            ; für Port B schreiben
; RB0 bis RB3 sind als Eingänge und RB4 bis RB7 sind als
  Ausgänge geschaltet
```

XORLW

Syntax: XORLW k
Funktion: Exklusiv-ODER-Verküpfung einer Konstanten mit dem W-Register
Status-Flags: Z
Beschreibung: Der Befehl führt eine bitweise Exklusiv-ODER-Verknüpfung der Konstanten k und dem W-Register durch.
Beispiel:
```
; im W-Register steht C1h    (=11000001b)
XORLW  B3h                  ; (=10110011b)
; im W-Register steht 72h    (=01110010b)
```

Die zweite Gruppe der Befehle beinhaltet alle byteorientierten Befehle, die sich auf das Arbeitsregister und die File-Register beziehen.

Das Ergebnis der Operation kann wahlweise in das W-Register oder in das angesprochene File-Register geschrieben werden.

Das Ziel ist durch den Parameter d bestimmt. Ist d=1, so wird das Ergebnis im File-Register, ist d=0, so wird das Ergebnis im W-Register abgelegt.

2. Byteorientierte Befehle, bezogen auf das Arbeitsregister und die File-Register

ADDWF
Syntax: ADDWF f, d
Funktion: Addiere ein Register zum W-Register
Status-Flags: Z, DC, C
Beschreibung: Der Befehl addiert den Inhalt des Registers zum W-Register und legt das Ergebnis im W-Register (d=0) oder im angegebenen Register (d=1) ab.
Beispiel:
```
;im W-Register steht 17h
; im FSR-Register steht C2h
ANDWF   FSR, 1
; im W-Register steht noch 17h
; im FSR-Register steht das Ergebnis 02h
```

ANDWF
Syntax: ANDWF f, d
Funktion: UND-Verknüpfung zwischen einem Register mit dem W-Register
Status-Flags: Z
Beschreibung: Der Befehl UND-verknüpft den Inhalt des Registers mit dem W-Register und legt das Ergebnis im W-Regsiter (d=0) oder im angegebenen Register (d=1) ab.
Beispiel:
```
; im W-Register steht 17h
; im FSR-Register steht C2h
ANDWF   FSR, 1
; im W-Register steht 17h
; im FSR-Register steht das Ergebnis 02h
```

CLRF
Syntax: CLRF f
Funktion: Register löschen
Status Flags: Z
Beschreibung: Der Befehl löscht das angegebene Register und beschreibt es mit Null. Dabei wird ebenfalls das Zero-Flag gesetzt.
Beispiel:
```
; in REG1 steht ein beliebiger Wert
CLRF    REG1
; REG1 ist gelöscht und enthält 00h
```

CLRW
Syntax: CLRW
Funktion: W-Register löschen
Status-Flags: Z
Beschreibung: Der Befehl löscht Arbeitsregister und setzt das Zero-Flag.
Beispiel:
```
; im W-Register steht ein beliebiger Wert
CLRW
; das W-Register ist gelöscht und enthält 00h
```

COMF
Syntax: COMF f, d
Funktion: Komplement eines Registers bilden
Status-Flags: Z
Beschreibung: Der Befehl bildet das Komplement des Inhalts des Registers und legt das Ergebnis im W-Register (d=0) oder im angegebenen Register (d=1) ab.
Beispiel:
```
; in  REG1 steht C1h          (=11000001b)
COMF   REG1, 0
; in REG1 steht C1h           (=11000001b)
; im W-Register steht 3Eh     (=00111110b)
```

DECF
Syntax: DECF f, d
Funktion: Dekrementiere Register
Status-Flags: Z
Beschreibung: Der Befehl verringert den Inhalt des angegebenen Registers und legt das Ergebnis im W-Register (d=0) oder im angegebenen Register (d=1) ab.
Beispiel:
```
; in  REG1 steht C1h          (=11000001b)
COMF    REG1, 1
; in REG1 steht C0h           (=11000000b)
```

DECFSC
Syntax: DECFSC f, d
Funktion: Dekrementiere Register und springe bei Null
Status Flags: keine
Beschreibung: Der Befehl verringert den Inhalt des angegebenen Registers und überspringt den nachfolgenden Befehl, wenn dabei Null erreicht wird. Bei dem Befehl muß der Parameter d immer auf 1 gesetzt werder, da das Ergebnis sonst nicht in das Register zurückgeschrieben wird und so nie Null erreichen kann. Der Befehl eignet sich um Programmschleifen zu realisieren, die x mal wiederholt werden müssen.
Beispiel:
```
; REG1 beinhaltet 03h
LOOP
; Programmschleife, die 3 mal ausgeführt werden soll
DECFSZ  REG1, 1
GOTO    LOOP
```

INCF
Syntax: INCF f, d
Funktion: Incrementiere Register
Status-Flags: Z
Beschreibung: Der Befehl erhöht den Inhalt des angegebenen Registers und legt das Ergebnis im W-Register (d=0) oder im angegebenen Register (d=1) ab.
Beispiel:
```
; in  REG1 steht C1h          (=11000001b)
INCF    REG1, 1
; in REG1 steht C2h           (=11000010b)
```

INCFSC
Syntax: INCFSC f, d
Funktion: Dekrementiere Register und springe bei Null
Status-Flags: keine
Beschreibung: Der Befehl verringert den Inhalt des angegebenen Registers und überspringt den nachfolgenden Befehl,wenn dabei Null erreicht wird. Wie beim DECFSC-Befehl der Parameter d immer auf 1 gesetzt sein, da sonst das Ergebnis nicht in das Register zurückgeschrieben wird und so nie Null erreichen kann.
Beispiel:
```
; REG1 beinhaltet FDh (100h - Schleifendurchläufe)
LOOP
; Programmschleife, die 3 mal ausgeführt werden soll
INCFSZ  REG1, 1
GOTO    LOOP
```

IORWF
Syntax: IORWF f, d
Funktion: Äquivalenz-Verknüpfung eines Registers mit W-Register
Status-Flags: Z, C, DC
Beschreibung: Der Befehl bildet die Äquivalenz-Verknüpfung zwischen dem angegeben Register und dem W-Register. Dabei wird beim Ergebnis ein Bit nur gesetzt, wenn die entsprechenden Bits in beiden Registern übereinstimmen. Das Ergebnis wird im angegebenen Register (d=1) oder im W-Register (d=0) abgelegt.
Beispiel:
```
; im W-Register steht 17h          (=00010111b)
; in REG1 steht C2h                (=11000010b)
IORWF   REG1, 1
; im W-Register steht 17h          (=00010111b)
; in REG1 steht das Ergebnis 2Ah   (=00101010b)
```

MOVF
Syntax: MOVF f, d
Funktion: Registerinhalt in das W-Register laden
Status-Flags: Z, C, DC
Beschreibung: Mit diesem Befehl kann der Inhalt eines Register in das W-Register kopiert werden, wobei der Parameter d=0 sein muß, um als Ziel das W-Register zu erhalten. Wird der Parameter d auf 1 gesetzt, so wird das Register in sich selbst kopiert und nur die Status-Flags entsprechend gesetzt.
Beispiel:
```
; in REG1 steht C1h
MOVF REG1, 0
; in REG1 und im W-Register steht C1h
```

MOVWF
Syntax: MOVWF f, d
Funktion: W-Register-Inhalt in das angegebene Register kopieren
Status-Flags: keine
Beschreibung: Mit diesem Befehl kann der Inhalt des W-Register in das angegebene Register kopiert werden, wobei der Parameter d=1 sein muß, um als Ziel das Register zu erhalten.
Beispiel:
```
; im  W-Register steht C1h
MOVWF REG1, 1
; in REG1 und dem W-Register steht C1h
```

Die letzte Befehlsgruppe bilden die Bit-Befehle, mit denen jedes Bit der Register einzeln verändert und getestet werden kann. Ebenso können alle I/O-Leitungen mit den Bitbefehlen angesprochen werden, da diese direkt über die Register erreichbar sind.

NOP
Syntax: NOP
Funktion: Keine Operation kopieren
Status-Flags: keine
Beschreibung: Dieser Befehl hat keine Auswirkung, er benötigt lediglich einen Prozessorzyklus und kann so zur Verzögerung genutzt werden.
Beispiel:
```
NOP                    ; einen Prozessorzyklus warten
```

RLF
Syntax: RLF f, d
Funktion: Rotiere Register nach links durch das Carry-Flag
Status-Flags: C
Beschreibung: Mit diesem Schiebebefehl wird er Inhalt des Registers um ein Bit nach links geschoben. Dabei wird das höherwertige Bit 7 in das Carry-Flag und das Carry-Flag in das Bit 0 des Registers geschoben. Der Parameter d bestimmt wieder das ausgewählte oder W-Register als Ziel.
Beispiel:
```
; in REG1 steht 5Ch (=01011100b) und Carry-Flag gesetzt
RLF    REG1, 1
;in REG1 steht B9h (=10111001b) u. Carry-Flag gelöscht
```

RRF
Syntax: RRF f, d
Funktion: rotiere Register nach links durch das Carry-Flag
Status-Flags: C
Beschreibung: Mit diesem Schiebebefehl wird der Inhalt des Registers um ein Bit nach rechts geschoben. Dabei wird das Bit 0 in das Carry-Flag und dieses in das Bit 7 des Registers geschoben. Der Parameter d bestimmt wieder das Ziel.
Beispiel:
```
; in REG1 steht 5Ch (=01011100b) und Carry-Flag gesetzt
RRF   REG1, 1
; in REG1 steht AEh (=10101110b) und Carry-Flag
; gelöscht
```

SUBWF
Syntax: SUBWF f, d
Funktion: Subtrahiere W-Register vom Register
Status-Flags: Z, DC, C
Beschreibung: Der Befehl subtrahiert den Inhalt des W-Registers vom angegebenen Register und legt das Ergebnis im W-Register (d=0) oder im angegebenen Register (d=1) ab. Das Carry-Flag wird bei einem Überlauf gelöscht und gesetzt wenn bei der Rechnung kein Überlauf aufgetreten ist.
Beispiel:
```
;im W-Register steht 17h
; im REG1-Register steht C2h
SUBWF   REG1, 1
; im W-Register steht noch 17h
; im REG1-Register steht das Ergebnis ABh
; Carry-Flag ist gesetzt, da kein Überlauf aufgetreten ist
```

SWAPF
Syntax: SWAPF f, d
Funktion: Tausche Bytehälften vom Register
Status-Flags: keine
Beschreibung: Der Befehl tauscht die unteren (Bit 0 bis 3) und oberen (Bit 4 bis 7) Nibbles des Bytes aus und legt das Ergebnis im W-Register (d=0) oder im angegebenen Register (d=1) ab.
Beispiel:
```
; im REG1-Register steht C2h
SWAPF   REG1, 1
; im REG1-Register steht 2Ch
```

XORWF
Syntax: XORWF f, d
Funktion: Exklusiv-ODER-Verknüpfung zwischen dem angegebenen und W-Register
Status-Flags: Z
Beschreibung: Der Befehl führt eine Exklusiv-ODER-Verknüpfung zwischen dem W-Registers und Register und legt das Ergebnis im W-Register (d=0) oder im angegebenen Register (d=1) ab.
Beispiel:
```
; im W-Register steht AAh              (=10101010b)
; in REG1 steht C1h                    (=11000001b)
XORWF   REG1, 0
; im W-Register steht das Ergebnis 6Bh (=01101011b)
; in REG1 steht C1h                    (=11000001b)
```

3. Bit-Befehle für Veränderung und Test jedes Register-Bits

BCF
Syntax: BCF f, b
Funktion: Löscht ein Bit im Register
Status-Flags: keine
Beschreibung: Der Befehl löscht das Bit Nummer b in dem angegebene Register
Beispiel:
```
BCF  PORTB, 7          ;I/O-Pin RB7 auf LOW setzen
```

BSF
Syntax: BSF f, b
Funktion: Setzt ein Bit im Register
Status-Flags: keine
Beschreibung: Der Befehl setzt das Bit Nummer b in dem angegebene Register
Beispiel:
```
BSF REG1, 0            ; Bit 0 von REG1 setzen
```

BTFSC
Syntax: BTFSC f, b
Funktion: Teste Bit und springe wenn gelöscht
Status-Flags: keine
Beschreibung: Mit dem Befehl kann ein Bit eines Registers getestet werden. Ist das Bit gelöscht, so wird der nachfolgende Befehl übersprungen.
Beispiel:
```
BTFSC  PORTB, 1        ; I/O-Port RB1 testen
GOTO   GESETZT         ; wird ausgeführt, wenn Pin auf
                       ; High liegt
; weiter im Programm
```

BTFSS
Syntax: BTFSS f, b
Funktion: Teste Bit und springe wenn gesetzt
Status-Flags: keine
Beschreibung: Mit dem Befehl kann ein Bit eines Registers getestet werden. Ist das Bit gesetzt, so wird der nachfolgende Befehl übersprungen.
Beispiel:
```
BTFSC   REG1, 0        ; Bit 0  von REG1 testen
GOTO    GELOESCHT      ; wird ausgeführt, wenn Bit
                       ; gelöscht ist
; weiter im Programm
```

Die Befehle der PIC 16C5X-Familie sind in der Tabelle 3 noch einmal übersichtlich zusammengefaßt und erleichtern den Überblick bei der Programmierung.

Damit ist die Beschreibung der Assemblerbefehle für die PIC 16C5X-Familie abgeschlossen und im nächsten Teil der Artikelserie wird die Erstellung eines Beispielprogramms, bis hin zur Programmierung des PICs beschrieben. ELV

Tabelle 3: Die Assemblerbefehle für die PIC 16C5x-Familie

Sprungbefehle und die Operationen mit Konstanten.		
ANDLW	k	UND-Verknüpfung des W-Registers mit einer Konstanten
CALL	k	Unterprogrammaufruf
CLRWDT		Watch-Dog-Timer rücksetzen
GOTO	k	Sprung zur angegeben Position im Programm
IORLW	k	ODER-Verknüpfung des W-Registers mit einer Konstanten
MOVLW	k	W-Register mit einer Konstanten laden
OPTION		OPTION-Regsiter mit dem W-Registers laden
RETLW	k	Rücksprung aus Unterprogramm mit Konstantenübergabe im W-Register
SLEEP		Stand-By-Modus einschalten
TRIS	f	Tristate-Register mit dem W-Registers laden
XORLW	k	Exklusiv-ODER-Verknüpfung des W-Registers mit einer Konstanten
byteorientoierte Befehle		
ADDWF	f, d	Addiere W-Register zu Register
ANDWF	f, d	Undverknüpfung von W-Regsiter und Register
CLRF	f	Lösche Register
CLRW		Lösche W-Register
COMF	f, d	Bilde Komlement des Registers
DECF	f, d	Dekrementiere Register
DECFSZ	f, d	Dekremnetiere Register und springe bei Null
INCF	f, d	Incrementiere Register
INCFSZ	f, d	Incrementiere Register und springe bei Null
IORWF	f, d	Äquivalenz-Verknüpfung zwischen W-Register und Register
MOVF	f, d	Registerinhalt kopieren
MOVWF	f, d	W-Register nach Regsiter kopieren
NOP		Keine Operation
RLF	f, d	Rotiere Register nach links durch Carry-Flag
RRF	f, d	Rotiere Register nach rechts durch Carry-Flag
SUBWF	f, d	Subtrahiere W-Register von Register
SWAPF	f, d	Tausche Bytehälften vom Register
XORWF	f, d	Exklusiv-OODER-Verknüpfung von W-Register und Register
bitorientierte Befehle		
BCF	f, b	lösche Bit des Registers
BSF	f, b	setze Bit des Registers
BTFSC	f, b	teste Bit des Registers und springe wenn gelöscht
BTFSS	f, b	teste Bit des Registers und springe wenn gesetzt