µC-Workshop (H39)

Aristoteles Tsiamitros

DD5FT

12. Mai 2012, 10:00-13:00 Uhr

Zahlensysteme

Polyadische Zahlensysteme

$$a_n\, a_{n-1}\, a_{n-2} \cdots a_1\, a_0\, a_{-1}\, a_{-2} a_{-m} = \sum_{k=-m}^{n} a_k \cdot B^k$$

Dezimalsystem

Wertevorrat: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Wertigkeit: 10^n, …, 10^2, 10^1 , 10^0 , 10^{-1} , 10^{-2} …

Beispiel:

$845 = 8 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$

Dualsystem

Wertevorrat: {0, 1}

Wertigkeit: 2^n, …, 2^2, 2^1 , 2^0 , 2^{-1} , 2^{-2} …

Beispiel:

$(101)_b = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (5)_d$

Oktalsystem

Wertevorrat: {0, 1, 2, 3, 4, 5, 6, 7}

Wertigkeit: 8^n, …, 8^2, 8^1 , 8^0 , 8^{-1} , 8^{-2} …

Beispiel:

$(453)_o = 4 \times 8^2 + 5 \times 8^1 + 3 \times 8^0 = (299)_d$

Hexadezimalsystem

Wertevorrat: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Wertigkeit: 16^n, …, 16^2, 16^1 , 16^0 , 16^{-1} , 16^{-2} …

Beispiel:

$(2F9A)_h = 2x16^3+15x16^2+9x16^1+10x16^0 = (12186)_d$

Umwandlung zwischen Zahlensystemen

Umwandlung Dezimal->Dual

Man wandelt eine Dezimalzahl in eine Dualzahl, indem man die Dezimalzahl in Zweierpotenzen zerlegt. Besetzte Zweierpotenzen haben den Koeffizienten 1, nicht besetzte Stellen den Koeffizienten 0.

Dies ist gleichwertig mit einer fortlaufenden Division durch 2 und Anordnen der Reste in umgekehrter Folge ihres Erscheinens nach folgendem Schema:

30	:	2	=	15	Rest	0
15	:	2	=	7	Rest	1
7	:	2	=	3	Rest	1
3	:	2	=	1	Rest	1
1	:	2	=	0	Rest	1

(30)d = (11110)b

Umwandlung Dual -> Dezimal

Horner-Schema

	1	1	0	1	1
+	0	2	6	12	26
2 *	1	3	6	13	27

$(11011)_b = 1*2^4+1*2^3+0*2^2+1*2^1+1*2^0 = (27)_d$

Dezimal -> Hexadezimal

Beispiel:	188 : 16	= 11	Rest	12	C
	11 : 16	= 0	Rest	11	B

(188)d = (BC)h

Hexadezimal -> Dezimal

$(3AF)h = 3 * 16^2 + 10 * 16^1 + 15 * 16^0 = (943)d$

Dual -> Hexadezimal

Eine Hexadezimalzahl benötigt vier Dualstellen. Diese Beziehung nutzt man für die direkte Umwandlung vom Dual- in das Hexadezimalsystem.

Beginnend bei den Dualstellen niedriger Wertigkeit, wird die Dualzahl in vierstellige Gruppen unterteilt. Jede dieser Gruppen kann direkt durch eine Hexadezimalzahl ausgedrückt werden.

Beispiel: (10110111001)b = (0101 1011 1001)b = (5B9)h

Hexadezimal -> Dual

Die Rückumwandlung einer Hexadezimalzahl in eine Dualzahl erfolgt in umgekehrter Weise, durch Umwandlung jeder einzelnen Hexadezimalziffer in eine vierstellige Dualzahl.

Beispiel: (5FC7)h = (0101 1111 1100 0111)b

Rechnen mit Dualzahlen

Addition

0	+ 0	= 0		
0	+ 1	= 1		
1	+ 0	= 1		
1	+ 1	= 0	Übertrag (Carry) 1	

Beispiel:

```
  0 0 1 1
+ 0 1 1 0
  C 1 1
= 1 0 0 1
```

Subtraktion

0	- 0	= 0	
0	- 1	= 1	Borger(Borrow) 1
1	- 0	= 1	
1	- 1	= 0	

Die Subtraktion wird auf die Addition zurückgeführt!

Einer- und Zweierkomplement

Einerkomplement = Negation der Dualzahl.

Beispiel: X=(1101)b = (13)d, -> E=(0010)b

Zweierkomplement=Einerkomplement + 1, Z=E+1

Beispiel:
X=(1101)b = (13)d
E=(0010)b
Z=(0010)b+(0001)b=(0011)b

Meistens wird das Zweierkomplement verwendet.

Subtraktion (x=123-11)

(123)d=(0111 1011)b
(11)d =(0000 1011)b
Zweierkomplement:
(-11)d =(1111 0100 + 0000 0001)b=(1111 0101)b

Addition (statt Subtraktion):
  0111 1011 =(+123)d
+ 1111 0101 =+(-11)d
 1 0111 0000
-1 0000 0000
 = 0111 0000 (+112)d

Subtraktion (x=11-123)

(+11)d =(0000 1011)b
(+123)d=(0111 1011)b
(-123)d =(1000 0100 + 0000 0001)b=(1000 0101)b

  0000 1011=(+11)d
+ 1000 0101=+(-123)d
= 1001 0000

2er Komplement :
  0110 1111
 + 0000 0001
 = 0111 0000 =(-112)

Multiplikation

- Einstellig

  0 x 0 = 0
  0 x 1 = 0
  1 x 0 = 0
  1 x 1 = 1

  Die einstellige Multiplikation kann mit einer UND-Verknüpfung realisiert werden.

- Mehrstellig

3 x 5 = 15
 0011 x 0101
  0011
 0000
0011
0000
0001111

Division

- 36 : 7 = 5 Rest 1

36			
- 7			
29	Q=1	08	
- 7		- 7	
22	Q=2	01	Q=5
- 7		- 7	
15	Q=3	-06	Rest negativ, ein
- 7			Schritt zurück
08	Q=4	Q=5 Rest 1	

Logische Grundverknüpfungen

Sämtliche Schaltungen der Digitaltechnik basieren auf drei **Grund-verknüpfungen**:

UND (AND):

Die Ausgangsvariable hat nur dann den Signalzustand ´1´, wenn alle Eingangsvariablen ebenfalls den Signal-zustand ´1´ haben.

ODER (OR):

Bei der ODER-Verknüpfung hat die Ausgangsvariable dann den Signalzustand ´1´, wenn mindestens eine Eingangsvariable den Signalzustand ´1´ hat.

NICHT (NOT):

Das Eingangssignal wird negiert. Der Signalzustand ´1´ am Eingang wird zu ´0´ am Ausgang und umgekehrt.

Abgeleitete Verknüpfungen

NAND: Die NAND-Verknüpfung entsteht durch Negierung des Ausgangs einer UND-Verknüpfung. Der Ausgang der NAND-Verknüpfung hat den Signalzustand ´1´ , wenn NICHT alle Eingänge den Signalzustand ´1´ haben.

NOR: Die NOR-Verknüpfung entsteht durch Negierung des Ausgangs einer ODER-Verknüpfung. Der Ausgang der NOR-Verknüpfung hat den Signalzustand ´1´ , wenn KEIN Eingang den Signalzustand ´1´ hat.

XOR (Exclusiv ODER): Es wird der Zusammenhang zwischen genau zwei Eingangsvariablen und einer Ausgangsvariablen hergestellt. Die Ausgangsvariable hat nur dann den Signalzustand ´1´, wenn die Eingangsvariablen verschiedene Signalzustände haben. Man bezeichnet diese Verknüpfung auch als Antivalenz.

Halbaddierer

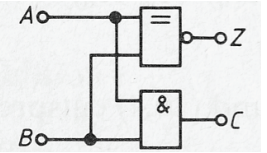
a b	a+b	c	s
0 0	00	0	0
0 1	01	0	1
1 0	01	0	1
1 1	10	1	0

a, b = zu addierende Zahlen
a+b = Duale Summe
c = Übertragsbit
s = Summenbit

Carry:

	a	$\bar{a}$
b	1	
$\bar{b}$		

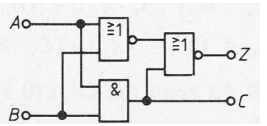
$c = a \cdot b$



Summe:

	a	$\bar{a}$
b		1
$\bar{b}$	1	

$s = a \cdot \bar{b} + \bar{a} \cdot b$



Volladdierer

a b c_1	a+b+c_1	c_2	s
0 0 0	00	0	0
0 0 1	01	0	1
0 1 0	01	0	1
0 1 1	10	1	0
1 0 0	01	0	1
1 0 1	10	1	0
1 1 0	10	1	0
1 1 1	11	1	1

Summe s:

b \ a	0	1
0	0	1
1	1	0
0	1	0
1	0	1

c_1

$$s = \bar{a}\, b\, \bar{c_1} + a\, b\, c_1 + \bar{a}\, \bar{b}\, c + a\, \bar{b}\, \bar{c_1}$$

Volladdierer

a b c_1	a+b+c_1	c_2	s
0 0 0	00	0	0
0 0 1	01	0	1
0 1 0	01	0	1
0 1 1	10	1	0
1 0 0	01	0	1
1 0 1	10	1	0
1 1 0	10	1	0
1 1 1	11	1	1

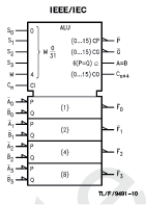
Carry c_2:

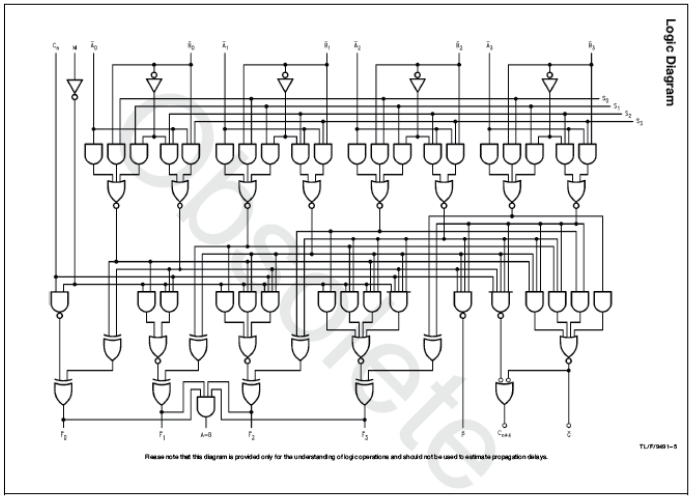
b \ a	0	1
0	0	1
1	1	0
1	1	0
0	1	0

c_1

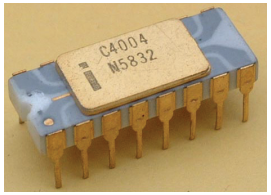
$$c_2 = a\, b\, +\, b\, c_1 + a\, c_1$$

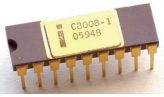
Arithmetic Logic Unit (ALU)

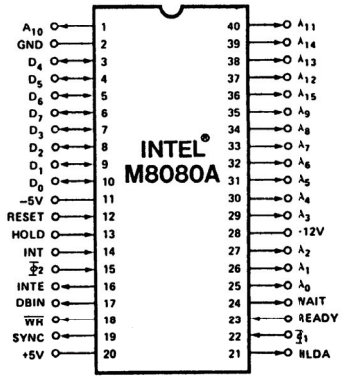




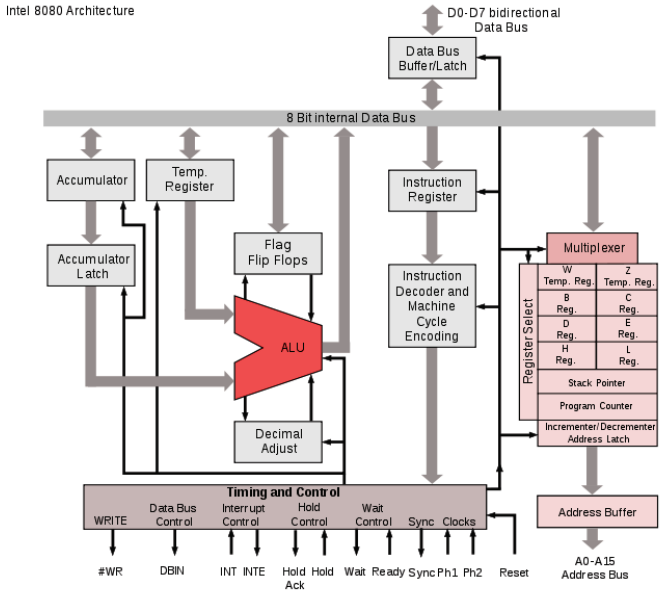
Mikroprozessoren



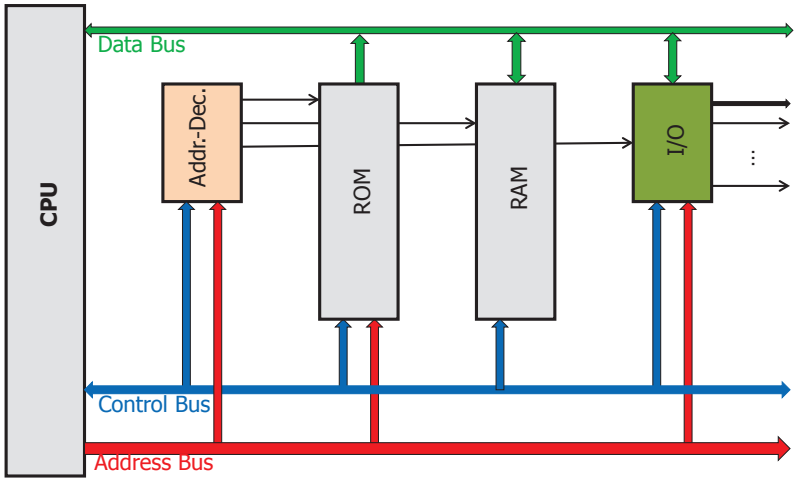




Intel 8080 Architektur

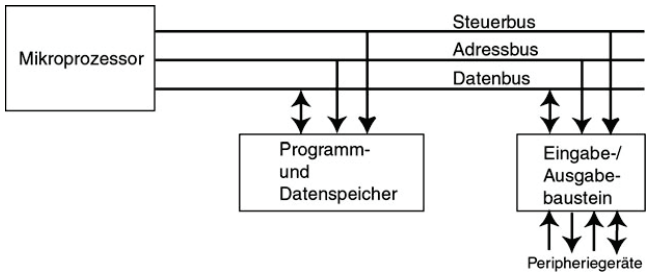


Rechnersystem



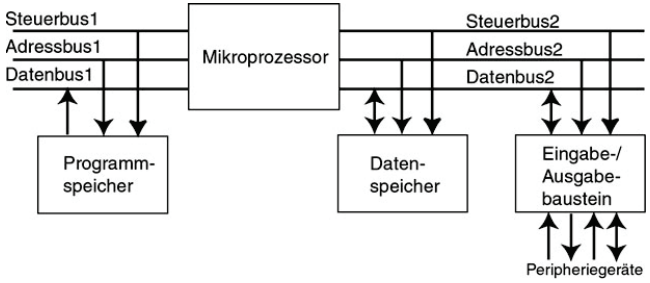
von Neumann-Architektur

Computer mit von Neumann-Architektur haben gemeinsamen Daten- und Programmspeicher.



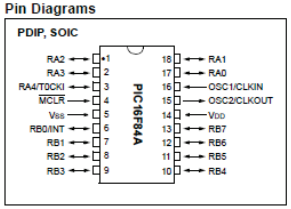
Harvard-Architektur

Computer mit Harvard-Architektur haben getrennten Daten- und Programmspeicher

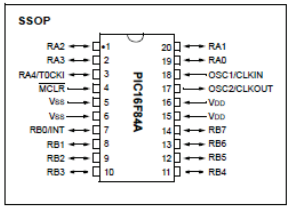


PIC16F84A

Pin Diagrams



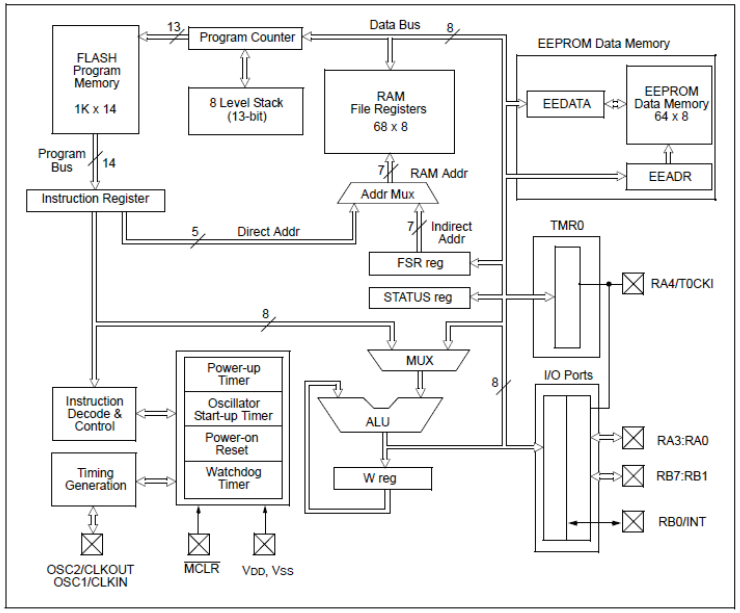
Dual In-Line-Package,
Small-Outline IC



Shrink Small Outline Package

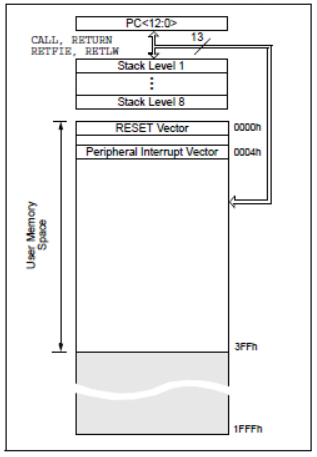
PIC16F84 Block-Diagramm

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM



Programmspeicher

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A



Datenspeicher

SFR (Special Function Reg.)

GPR (General Purpose Reg.)

FIGURE 2-2: REGISTER FILE MAP - PIC16F84A

File Address			File Address
00h	Indirect addr.(1)	Indirect addr.(1)	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2(1)	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	68 General Purpose Registers (SRAM)	Mapped (accesses) in Bank 0	8Ch
4Fh			CFh
50h			D0h
7Fh	Bank 0	Bank 1	FFh

Unimplemented data memory location, read as '0'.
Note 1: Not a physical register.

SFR

TABLE 2-1: SPECIAL FUNCTION REGISTER FILE SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

Status-Register

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7-6 **Unimplemented:** Maintain as '0'

bit 5 **RP0:** Register Bank Select bits (used for direct addressing)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

bit 4 **$\overline{TO}$:** Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$:** Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C:** Carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Option-Register

REGISTER 2-2: OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{RBPU}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG:** Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS:** TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE:** TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA:** Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0:** Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Interrupt-Control-Register

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

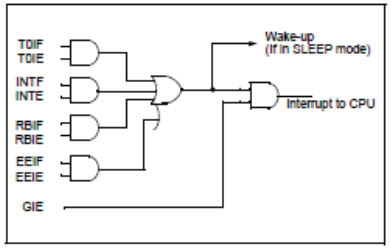
bit 6 **EEIE:** EE Write Complete Interrupt Enable bit
1 = Enables the EE Write Complete interrupts
0 = Disables the EE Write Complete interrupt

bit 5 **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

FIGURE 6-10: INTERRUPT LOGIC



bit 2 **T0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

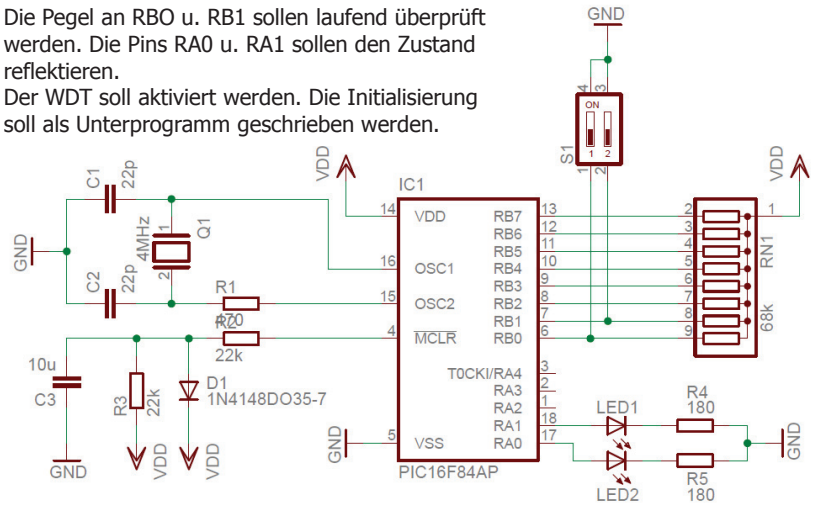
Instruction Set

TABLE 7-2: PIC16CXXX INSTRUCTION SET

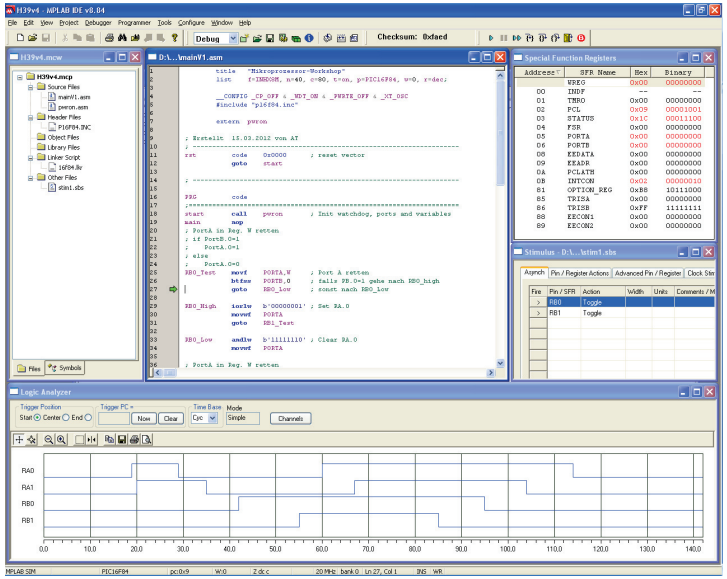
Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from interrupt	2	00 0000 0000 1001			
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	

Schaltungsbeispiel für uC-Workshop

Die Pegel an RBO u. RB1 sollen laufend überprüft werden. Die Pins RA0 u. RA1 sollen den Zustand reflektieren.
Der WDT soll aktiviert werden. Die Initialisierung soll als Unterprogramm geschrieben werden.



MPLAB



Werkzeuge

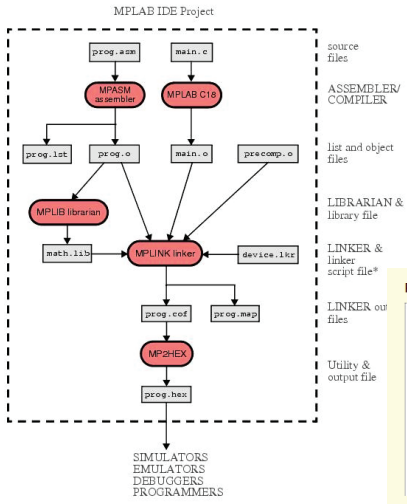


FIGURE: UTILITIES OPERATION

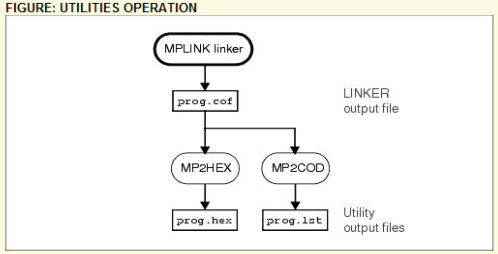
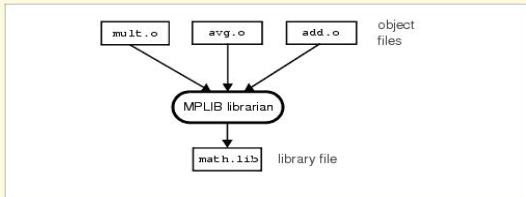
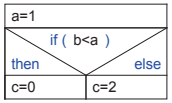


FIGURE: MPLIB LIBRARIAN OPERATION

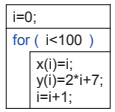


Programmfluss

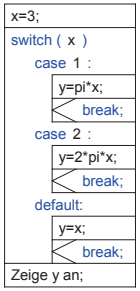
if-then-else-Anweisung



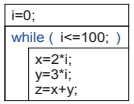
for-Schleife



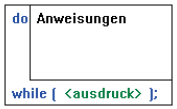
switch-Anweisung



While-Schleife



Do While (...)



MPLAB: Neues Projekt einrichten

Project -> Project Wizard -> Next

… danach den Programmanweisungen folgen.

Dateien hinzufügen

Vorhandene Dateien (aus älteren Projekten) übernehmen. Neue Dateien mit dem Editor erstellen.

Die Dateien dem Projekt hinzufügen.

Debugger wählen

Hier wird der MPLAB Simulator gewählt.

Debugger -> Select Tools->MPLAB SIM

MPLAB: Weitere Fenster

Weitere Fenster (nach Bedarf) öffnen, z. B.

- *View -> Special Function Registers*
- *View -> Watch*
- *Debugger -> Settings -> Osc / Trace. Processor Frequency einstellen, z. B. Voreinstellung = 20 MHz bestätigen.*
- *Debugger -> Stimulus -> New WorkBook*
- *View -> Simulator Logic Analyzer*

MPLAB: Projekt assemblieren

Projekt assemblieren, binden und testen.

- Debug/Release wählen und dann <Build All>. Hoffentlich ist die Meldung BUILD SUCCEDED.
- Mit den Schaltflächen ***Step over***, ***Step Into***, ***Step out***, schrittweise durch das Programm gehen oder ***Run/Animate*** verwenden.

Einen Besuch Wert ist auch:

- Help -> Topics -> …

Hauptprogramm

```
1          title   "Mikroprozessor-Workshop, 12. Mai 2012"
2          list    f=INHX8M, n=40, c=80, t=on, w=0, r=dec;
3
4          __CONFIG _CP_OFF & _WDT_ON & _PWRTE_OFF & _XT_OSC
5          #include "p16f84.inc"
6
7          extern  pwron
8
9   ; Erstellt  15.03.2012 von AT
10  ; ------------------------------------------------------------------
11  rst         code    0x0000      ; reset vector
12              goto    start
13
14  ; ------------------------------------------------------------------
15
16  PRG         code
17  ;==================================================================
18  start       call    pwron       ; Init watchdog, ports and variables
19  main        nop
20  ; PortA in Reg. W retten
21  ; if (PortB.0=1) then
22  ;    PortA.0=1
23  ; else
24  ;    PortA.0=0
25  RB0_Test    movf    PORTA,W     ; Port A retten
26              btfss   PORTB,0     ; falls PB.0=1 gehe nach RB0 high
27              goto    RB0_Low     ; sonst nach RB0_Low
28
29  RB0_High    iorlw   b'00000001' ; Set RA.0
30              movwf   PORTA
31              goto    RB1 Test
32
33  RB0 Low     andlw   b'11111110' ; Clear RA.0
34              movwf   PORTA
35
```


Hauptprogramm

```
36  ; PortA in Reg. W retten
37  ; if (PortB.1=1) then
38  ;    PortA.1=1
39  ; else
40  ;    PortA.1=0
41  RB1_Test    movf    PORTA,W     ; siehe oben (RB.0)
42              btfss   PORTB,1
43              goto    RB1_Low
44
45  RB1_High    iorlw   b'00000010' ; Set RA.1
46              movwf   PORTA
47              goto    continue
48
49  RB1_Low     andlw   b'11111101' ; Clear RA.1
50              movwf   PORTA
51
52  continue    clrwdt
53              goto    main
54
55  ; ------------------------------------------------
56              end
```


Power On, Init.

```
1   ; Power On
2   ; Initialize PORTA and PORTB, Init Variables, etc.
3   ; ------------------------------------------------
4           #include "p16f84.inc"
5
6           code
7           global  pwron
8
9   pwron   ; WDT configuration
10          bsf     STATUS,RP0      ; Bank 1
11          movlw   b'10111000'
12          movwf   OPTION_REG
13          bcf     STATUS,RP0      ; Bank 0
14          clrwdt
15
16          ; PORTA configuration: Output
17          clrf    PORTA
18          bsf     STATUS,RP0      ; Bank 1
19          clrw
20          movwf   TRISA
21          bcf     STATUS,RP0      ; Bank 0
22
23          ; PORTB configuration: Input
24          clrf    PORTB
25          bsf     STATUS,RP0      ; Bank 1
26          movlw   0xFF
27          movwf   TRISB
28          bcf     STATUS,RP0      ; Bank 0
29
30          ; Init Port A
31          bsf     PORTA,0
32          bsf     PORTA,1         ; RA 0:1 = 1
33
34          retlw   0               ; w=0
35          end
```


Literatur

- *Roland Woitowitz, Klaus Urbanski, Winfried Gehrke*
  Digitaltechnik. Springer, 6. Auflage 2012
- *Klaus Wüst*
  Mikroprozessortechnik, Vieweg + Teubner, 4. Auflage 2011
- *Microchip*
  PIC16F84A, Data Sheet
  (2001 Microchip Technology Inc., DS35007B)
- *Microchip*
  PICmicro, Mid-Range Family, Reference Manual
  (December 1997 /DS33023A)