PC & ELEKTRONIK

Anne und Manfred König

Handbuch PIC24/dsPIC-Mikrocontroller

Praxisbeispiele zur Anwendung der Module und Befehle

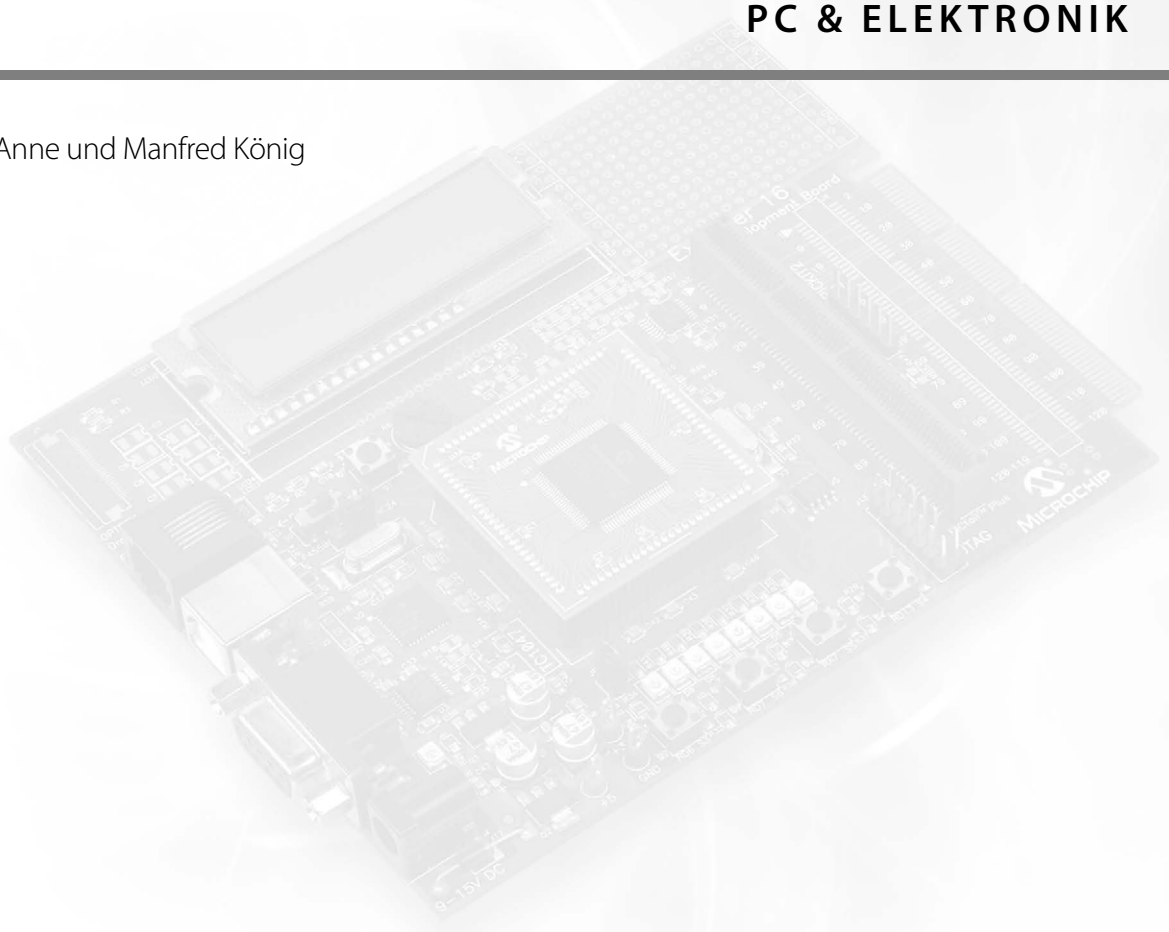
Inklusive CD-ROM

FRANZIS

Anne und Manfred König

Handbuch PIC24/dsPIC-Mikrocontroller

Anne und Manfred König



Handbuch PIC24/dsPIC-Mikrocontroller

Praxisbeispiele zur Anwendung der Module und Befehle

FRANZIS

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte Daten sind im Internet über http://dnb.ddb.de abrufbar.

Satz: DTP-Satz A. Kugge, München
art & design: www.ideehoch2.de
Druck: C.H. Beck, Nördlingen
Printed in Germany

ISBN 978-3-645-65273-5

Inhaltsverzeichnis

1 Die 16-Bit-PIC-Klasse ... 13
1.1 Oszillatoren ... 22
1.1.1 Auswahl des Systemtakts ... 23
1.1.2 Umschalten des Systemtakts ... 25
1.1.3 Fail Save Clock Monitor (FSCM) ... 26
1.2 Datenspeicher ... 27
1.2.1 Organisation des Datenspeichers ... 27
1.2.2 Besondere Datenregister ... 27
1.3 Programmspeicher ... 29
1.3.1 Adressierung des Programmspeichers ... 29
1.3.2 Aufteilung des Programmspeichers ... 30
1.3.3 Lesen des Programmspeichers ... 31
1.3.4 Löschen und Schreiben des Programmspeichers ... 32
1.4 Power Management ... 34
1.4.1 Taktfrequenz verringern ... 35
1.4.2 Peripheriemodule abschalten ... 36
1.4.3 Sleep-Modus ... 36
1.4.4 IDLE-Modus ... 38
1.4.5 DOZE-Modus ... 40

2 PIC24-Assemblersprache ... 41
2.1 Funktion der Befehle ... 42
2.2 Überblick über die Befehle ... 42
2.3 Form der Befehle ... 43
2.4 Argumente ... 44
2.4.1 Konstante Argumente ... 45
2.4.2 Direkte Adressierung eines Fileregisters ... 45
2.4.3 Direkte Adressierung eines W-Registers ... 45
2.4.4 Doppelregister ... 45
2.4.5 Indirekte Adressierung ... 45
2.4.6 Befehle mit mehreren Argumenten ... 47
2.4.7 Befehle mit direktem Fileregister ... 47
2.5 Verzweigungsbefehle ... 48
2.5.1 Argumente der Verzweigungsbefehle ... 49
2.5.2 CALL und RCALL ... 50

2.5.3 Bedingungsloser BRA-Befehl ... 50
2.5.4 Bedingte BRA-Befehle ... 50
2.6 Transferbefehle ... 51
2.6.1 MOV-Befehle ... 52
2.6.2 Austauschbefehle ... 52
2.6.3 PUSH und POP ... 52
2.6.4 TBLRD- und TBLWT-Befehle ... 53
2.7 Arithmetische und logische Befehle mit einem Operanden ... 55
2.8 Find-First-Befehle ... 57
2.9 Arithmetische und logische Befehle mit zwei Operanden ... 58
2.10 Befehle mit Bitargumenten ... 58
2.11 REPEAT-Befehl ... 60
2.12 Multiplikation ... 60
2.13 Division ... 61

3 DSP-Einheit (nur dsPIC) ... 63
3.1 DO-Befehl ... 64
3.2 Akkumulatoren ACCA und ACCB ... 64
3.3 Sättigung der Akkumulatoren ... 65
3.4 Überläufe ... 66
3.5 DSP-Zahlenformate ... 67
3.6 X- und Y-Datenbereiche ... 70
3.7 DSP-Befehle mit Prefetch ... 70
3.7.1 Prefetch-Operanden ... 71
3.7.2 Prefetch-Sonderfall für euklidische Abstände ... 72
3.7.3 Beschreibung der DSP-Befehle mit Prefetch ... 72
3.7.4 Modulo-Adressierung ... 73
3.8 DSP-Befehle ohne Prefetch (Akkumulatorbefehle) ... 74
3.9 DSP-Anwendung ... 75

4 Assemblercode und C ... 79
4.1 Assemblercode in MPLAB C30/XC16 Files ... 79
4.1.1 Built-in-Funktionen ... 79
4.1.2 Erstellen von eigenen Assemblerunterprogrammen ... 80
4.1.3 Inline-Assemblerzeilen im C30/XC16-Compiler ... 84
4.2 Der C30/XC16-Compiler und sein ASM-Code ... 85
4.2.1 Compileroptimierungen ... 85
4.2.2 Beobachten des erzeugten Assemblercodes ... 86
4.2.3 Erstes kleines Beispiel ... 87
4.2.4 Datentransfer ... 88
4.2.5 Multiplikation: 16 Bit • 16 Bit mit 32-Bit-Ergebnis ... 92

4.2.6 Division: 16 Bit / 16 Bit ... 94
4.2.7 Division: 32 Bit / 16 Bit ... 97
4.2.8 Anwendungsbeispiel GPS-Positionsdaten ... 99

5 Umgang mit den Peripheriemodulen ... 101
5.1 Ausschalten und Abschalten der Peripheriemodule ... 102
5.2 Initialisierung von Peripheriemodulen ... 103
5.3 Programmtechnische Verfahrensweisen in C ... 104

6 Interrupts ... 107
6.1 Interrupt-Status-Flags ... 107
6.2 Zulassen von Unterbrechungen ... 108
6.3 Ablauf von Unterbrechungen ... 108
6.4 Schachteln von Interrupts ... 109
6.5 Interrupt-Prioritäten ... 109
6.6 CPU-Priorität ... 110
6.7 Interrupt-Vektoren ... 110
6.8 Alternative Interrupt-Vektoren ... 111
6.9 Traps ... 111
6.10 Register des Interrupt-Systems ... 112
6.10.1 INTCON1-Register ... 113
6.10.2 INTCON2-Register ... 113
6.10.3 IFS- und IEC-Register ... 113
6.10.4 Prioritätsregister ... 114
6.11 Initialisieren der Interrupts ... 115
6.12 Schreiben der Interrupt-Routine ... 116
6.13 Ablauf von Interrupt-Routinen ... 119

7 DMA ... 121
7.1 DMA-Speicher ... 122
7.2 DMA-Transfer ... 122
7.3 Adressierungsarten ... 124
7.4 Register des DMA-Controllers ... 125
7.4.1 DMAxCON-Register ... 126
7.4.2 DMAxREQ-Register ... 126
7.4.3 DMAxPAD-Register ... 126
7.4.4 DMAxSTA- und DMAxSTB-Register ... 126
7.4.5 DMAxCNT-Register ... 127

8 I/O-Ports ... 129
8.1 Funktionen der Portpins ... 129
8.2 Eingänge und Ausgänge ... 130
8.3 Register der I/O-Ports ... 134

8.3.1 Zustandsregister PORTx und Latch-Register LATx 134
8.3.2 Richtungsregister TRISx..... 135
8.3.3 Open-drain-Register ODCx 136
8.3.4 AD1PCFG- und ANSx-Register..... 137
8.4 Interrupts durch Flanken an Portpins..... 138
8.5 Nutzung der Portpins durch Peripheriemodule 140
8.6 Peripheral Pin Select 140
8.7 Zuordnungsprozedere für Ausgangsfunktionen 142
8.8 Zuordnungsprozedere für Eingangspins 144

9 Timermodule 149
9.1 Register der Timermodule 149
9.1.1 TxCON-Register..... 149
9.2 Zählweise..... 150
9.2.1 Beispiel TMRint010: Beobachtung von Timer-Interrupt und Timerreset..... 151
9.3 Vorteiler 152
9.4 Timertypen..... 153
9.4.1 Typ A..... 153
9.4.2 Typ B 154
9.4.3 Typ C..... 155
9.5 Funktion als Zähler..... 155
9.6 Gated Timer..... 156
9.6.1 Beispiel T2GATE010: Demonstration des Gated-Timer-Modus..... 157
9.7 Lesen und Schreiben der Timerregister 158
9.8 32-Bit-Timerpaar..... 159
9.9 Schreiben und Lesen von 32-Bit-Timern 159

10 Watchdog Timer..... 161
10.1 Einschalten des Watchdogs..... 162
10.2 Überlaufperiode 162
10.3 Rücksetzen des Watchdog Timers 163
10.4 Watchdog-Fenster 163

11 AD-Wandler 165
11.1 Register der einfachen 10-Bit-AD-Wandler 167
11.1.1 ADxCON1-Register 167
11.1.2 ADxCON2-Register 168
11.1.3 ADxCON3-Register 168
11.1.4 AD1CHS-Register 169
11.1.5 AD1CSSL-Register..... 169
11.2 Konfiguration des AD-Wandlers..... 170

11.2.1 Konfiguration der Pins als analoge Eingänge ... 170
11.2.2 Festlegen der Referenzspannung ... 170
11.2.3 Festlegen des Wandlungstakts TAD ... 171
11.2.4 Festlegen der Abtastzeit ... 171
11.3 Einzelne Messungen ... 172
11.3.1 Beispiel ADsingle010: Einzelmessungen mit wechselnden Kanälen ... 174
11.4 Automatische Messfolgen ... 175
11.4.1 Starten der Wandlung ... 176
11.4.2 Ergebnisbuffer ... 177
11.4.3 Beispiel ADint010: Wiederholte Messung eines Kanals ... 177
11.5 12-Bit-AD-Wandler mit mehreren parallelen 10-Bit-Kanälen ... 181
11.5.1 ADxCON1-Register ... 181
11.5.2 ADxCON2-Register ... 181
11.6 Beispiel ADdma710: wiederholte Messung eines Kanals ... 183

12 Analoge Komparatoren ... 185
12.1 Komparatorereignis ... 186
12.2 Register der analogen Komparatoren ... 186
12.2.1 CMCON-Register für Dual-Comparator-Module ... 187
12.2.2 CMSTAT-Register für Triple-Comparator-Module ... 188
12.2.3 CMxCON-Register für Triple-Comparator-Module ... 189
12.3 Komparatorreferenzspannung ... 190
12.3.1 CVRCON-Register ... 191
12.4 Beispiel ADCMP010: Komparator und AD-Wandler an einem gemeinsamen Pin ... 192

13 Output-Compare-Module ... 195
13.1 Gemeinsame Grundfunktionalität ... 196
13.2 Output-Compare-Modi ... 196
13.2.1 Modi 0 bis 5 ... 197
13.2.2 Modi 6 und 7 ... 199
13.3 Dedizierte Timer der OC-Module ... 201
13.4 Controlregister der einfachen OC-Module ... 203
13.4.1 OCxCON-Register ... 203
13.5 Controlregister der komplexen OC-Module ... 204
13.5.1 OCxCON1-Register ... 204
13.5.2 OCxCON2-Register ... 205
13.6 Initialisierung der OC-Module ... 206
13.6.1 Beispiel OCpulse010: Erzeugen von Pulsen im 50-Hz-Takt ... 207
13.6.2 Beispiel OCppulse: Pulse mit vorgegebenem Offset ... 209
13.6.3 Beispiel OC1FLT110: Prüfen des OCx-Ausgangs im Fehlerfall ... 211

14 Input Capture ... 215
14.1 Grundfunktionalität ... 215
14.2 Modi der Input-Capture-Module ... 216
14.3 Dedizierte Timer der IC-Module ... 217
14.4 Controlregister der einfachen IC-Module ... 218
14.4.1 ICxCON-Register ... 218
14.5 Controlregister der komplexen IC-Module ... 219
14.5.1 ICxCON1-Register ... 219
14.5.2 ICxCON2-Register ... 220
14.6 Initialisierung der IC-Module ... 221

15 UART-Module ... 223
15.1 Register der UART-Module ... 224
15.1.1 Status- und Controlregister UxSTA ... 224
15.1.2 UxMOD-Register ... 225
15.2 Baudrate ... 225
15.3 Senden eines Bytes ... 227
15.4 Die Status-Flags TRMT und UTXBF ... 228
15.5 Die Bits UTXISEL [1:0] ... 228
15.6 Senden von Zeichenketten ... 229
15.7 Sporadisches Senden von Daten ... 230
15.8 Empfangen von Bytes ... 231
15.9 Das Status-Flag URXDA ... 232
15.10 Die Bits URXISEL [1:0] ... 232
15.11 RX-Interrupt-Routinen ... 233
15.12 Empfang von Strings mit vereinbartem Endzeichen ... 233

16 SPI-Module ... 237
16.1 Register der SPI-Module ... 238
16.1.1 SPIxSTAT-Register ... 238
16.1.2 SPIxCON1-Register ... 239
16.1.3 SPIxCON2-Register ... 239
16.2 Übertragungsgeschwindigkeit ... 240
16.3 Lesen und Schreiben von Daten ... 240
16.4 SPI-Status ... 241
16.5 Umgang mit dem SPI-EEPROM 25AA512 von Microchip ... 242
16.5.1 Lesen von Datenblocks ... 243
16.5.2 Schreiben von Datenblocks ... 244
16.6 Beispiel SPIEE010: Bedienen des serielles EEPROM 25AA512 ... 246

17 I²C-Bus ... 249
17.1 Geschwindigkeit der I²C-Übertragung ... 249
17.2 Register der I²C-Module ... 250
17.2.1 I2CxCON-Register ... 250
17.2.2 I2CxSTAT-Register ... 251

18 Parallel Master Port (PMP) ... 253
18.1 Verwendung der Steuerleitungen (PMRD, PMWR oder PMRD, PMENB) ... 253
18.2 Register des PMP-Moduls ... 254
18.2.1 PMCON-Register ... 254
18.2.2 PMMODE-Register ... 255
18.2.3 PMADDR-Register ... 255
18.2.4 PMAEN-Register ... 256
18.3 Beispiel PMP010: SRAM IS61LV256AL ... 256
18.3.1 Schreiben ins SRAM ... 257
18.3.2 Lesen aus dem SRAM ... 258
18.3.3 Geschwindigkeit der SRAM-Routinen ... 258

8 I/O-Ports

Als Port bezeichnet man eine zusammenhängende Menge von jeweils bis zu 16 Ein-/Ausgängen. Die Ports werden mit den Namen PORTA, PORTB, PORTC usw. benannt. Diese Namen werden auch für die zugehörigen Zustandsregister benutzt. In den Dokumentationen wird ein beliebiger Vertreter der Ports mit PORTx bezeichnet. Die einzelnen Portpins werden in der Assemblersprache mit PORTx, #0 bis PORTx, #15 benannt. In C können sie über eine Struktur mit PORTxbits.Bitname bezeichnet werden. In den Listen und in den Pinout-Diagrammen werden die Pins mit RA0 bis RA15, RB0 bis RB15 usw. bezeichnet.

8.1 Funktionen der Portpins

Die Portpins können entweder nur als einfache I/O-Pins fungieren oder wahlweise eine alternative Funktion erhalten.

Die Anzahl von Ports und deren Eigenschaften sind für jedes Derivat im Datenblatt zu ermitteln. Dort findet man als wichtigste Informationen im Kapitel »Device Overview« das Pindiagramm und unter dem Absatz »Pinout Description« die Listen mit den Eigenschaften und Funktionen der Pins.

Bei manchen Derivaten gibt es zu jedem Pin eine feste Menge von möglichen Funktionen, die im Pindiagramm zu sehen sind. Falls sich unterschiedliche Funktionen gegenseitig ausschließen, hat immer die Funktion Priorität, die am weitesten links in der Liste steht.

Bei vielen Derivaten ist die Funktion der Pins per Software wählbar. Diese Fähigkeit wird als *Peripheral Pin Select* (PPS) bezeichnet. Bei diesen Derivaten findet man im Pindiagramm nur die Bezeichnung RP oder RPI mit je einer Nummer, denen man per Software bestimmte Funktionen zuordnen kann.

Wenn ein Pin einem Peripheriemodul zugeordnet wird, übernimmt das Modul die Herrschaft über diesen Pin, sobald das Modul eingeschaltet wird und sofern kein anderes Modul höherer Priorität eingeschaltet ist.

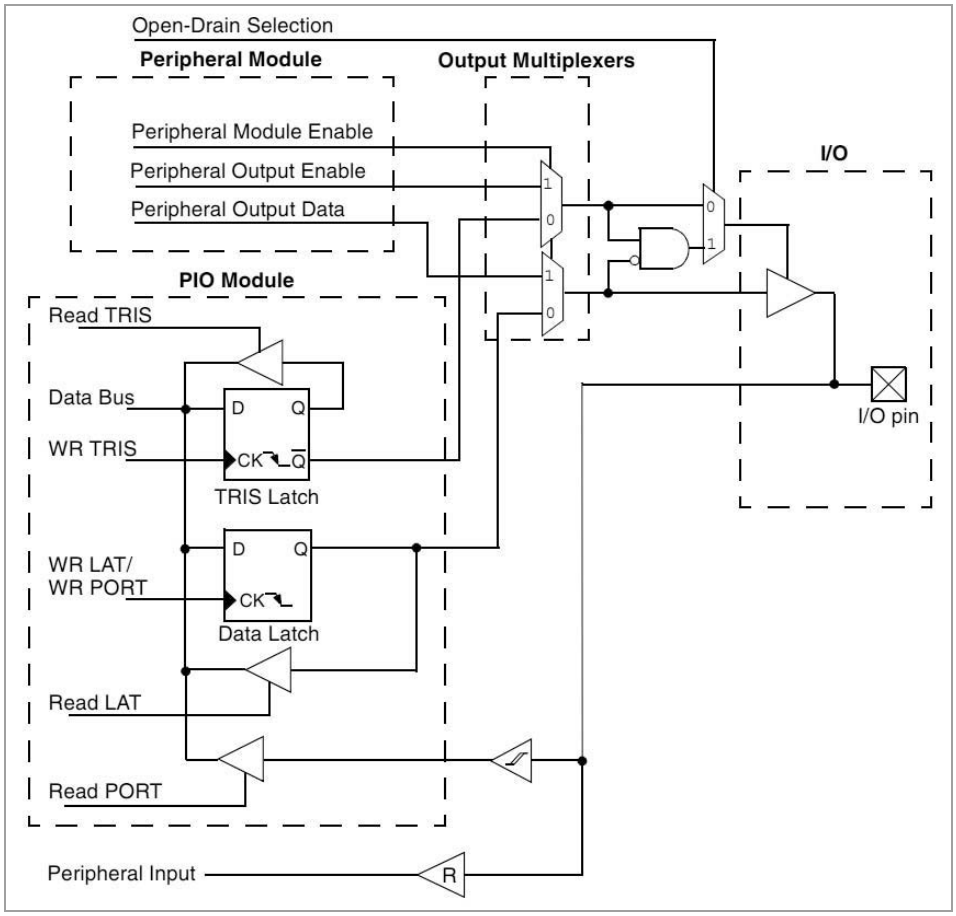


Bild 8.1: Typische Portstruktur

8.2 Eingänge und Ausgänge

Um als Ausgang dienen zu können, besitzt jeder Portpin einen Ausgangstreiber. Dieser besitzt einen Output-Enable, welcher per Software ein- und ausgeschaltet werden kann. Falls der Portpin aber von einem Peripheriemodul verwendet wird, das einen eigenen Output-Enable besitzt, wird der Output-Enable des Portpins ausgeschaltet.

Eingänge für die AD-Wandler oder die analogen Komparatoren sind beim RESET immer als analoge Eingänge konfiguriert und die Ausgangstreiber sind abgeschaltet. Falls ein Eingang als analoger Eingang konfiguriert wird, wird die digitale Eingangsstufe abgeschaltet und es wird bei der digitalen Abfrage des Pins immer eine »0« gelesen.

Ansonsten kann der Zustand eines Eingangs immer gelesen werden, egal, welchem digitalen Peripheriemodul er zugeordnet ist. Peripheriemodule, die einen Portpin als digitalen Eingang benutzen, z. B. die Input-Capture-Module, beeinträchtigen die Eingangsfunktion des Pins nicht. Ein normaler Lesezugriff auf den Portpin ist also problemlos möglich. Darüber hinaus übernimmt das Input-Capture-Modul auch nicht die Kontrolle über die Ausgangstreiber. Man kann also durch Einschalten des Ausgangstreibers und Ausgeben einer geeigneten Flanke ein Capture-Ereignis auslösen.

Wenn ein Portpin als analoger Eingang geschaltet ist, sollte man daran denken, den Ausgangstreiber auszuschalten, weil sonst das Messen in der Regel keinen Sinn ergibt. Dabei gibt es aber durchaus sinnvolle Ausnahmen wie z. B. das Messen von Lade- und Entladezeiten von Kondensatoren.

Die Portpins können also wahlweise als digitale Ausgänge oder als analoge oder digitale Eingänge konfiguriert werden.

Damit ein Portpin als Ausgangspin funktioniert, muss der zugehörige Output-Enable eingeschaltet werden, was über ein zugehöriges Bit in dem zugeordneten TRIS-Register geschieht. Peripheriemodule, die einen Pin als Ausgang benutzen, übernehmen meist über einen Multiplexer die Herrschaft über die Datenleitungen der Pins und über die Output-Enables. Das entsprechende Bit des TRIS-Registers wird dann außer Kraft gesetzt.

Bei der Verwendung von Peripheriemodulen sollte man immer im entsprechenden Datenblatt nachsehen, wer die Kontrolle über den Zustand des Portpins hat.

Bei vielen Ports sind die Ausgänge per Software entweder als Push-Pull- oder als Open-drain-Ausgänge über das ODCx-Register konfigurierbar. Ob es zu einem Port ein solches ODC-Register gibt, ist im Kapitel »Memory-Organisation« des jeweiligen Derivats nachzulesen. Im Pindiagramm der Derivate sind die Pins dunkel markiert, für die Open-drain möglich ist. Diese Pins sind damit 5-V-kompatibel.

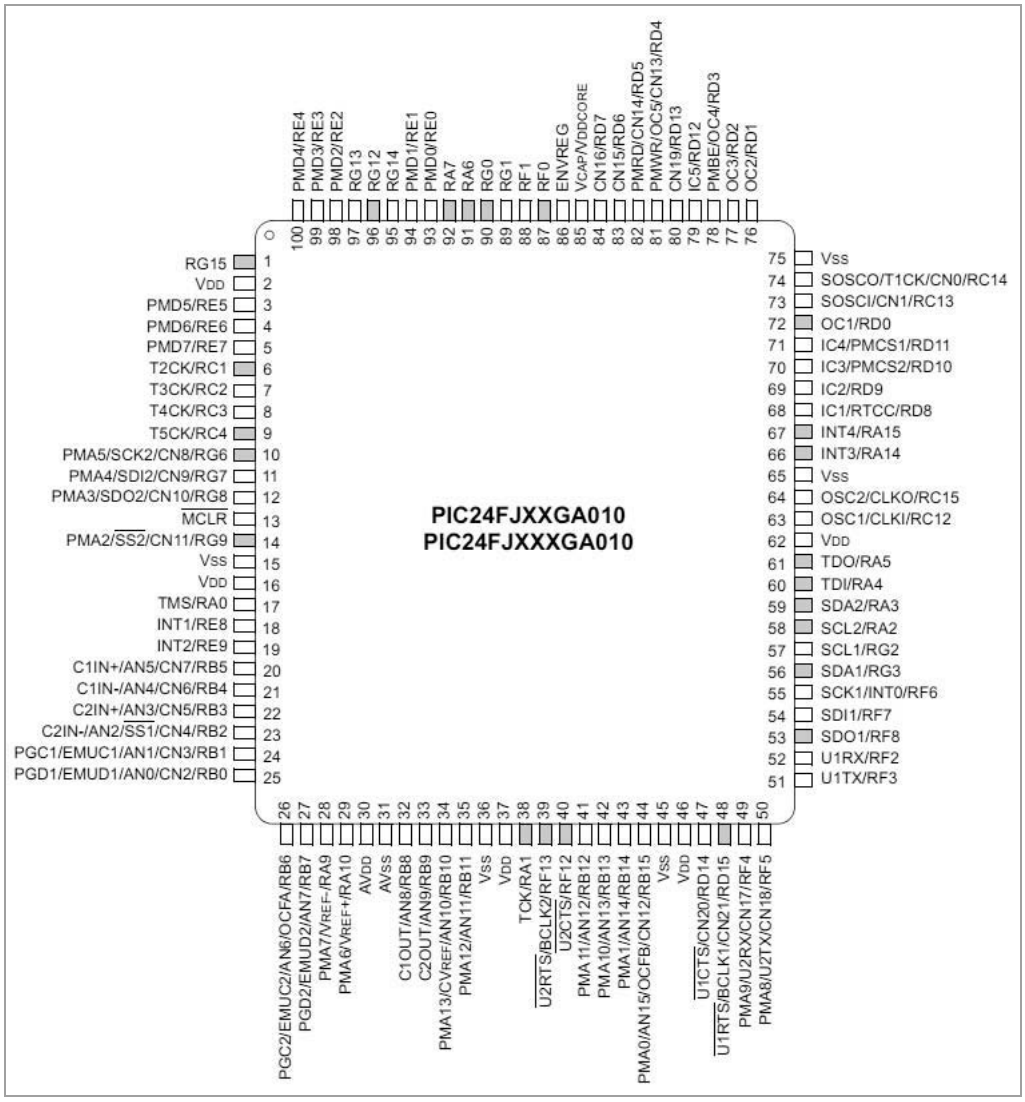


Bild 8.2: Pindiagramm des PIC24FJ128GA010

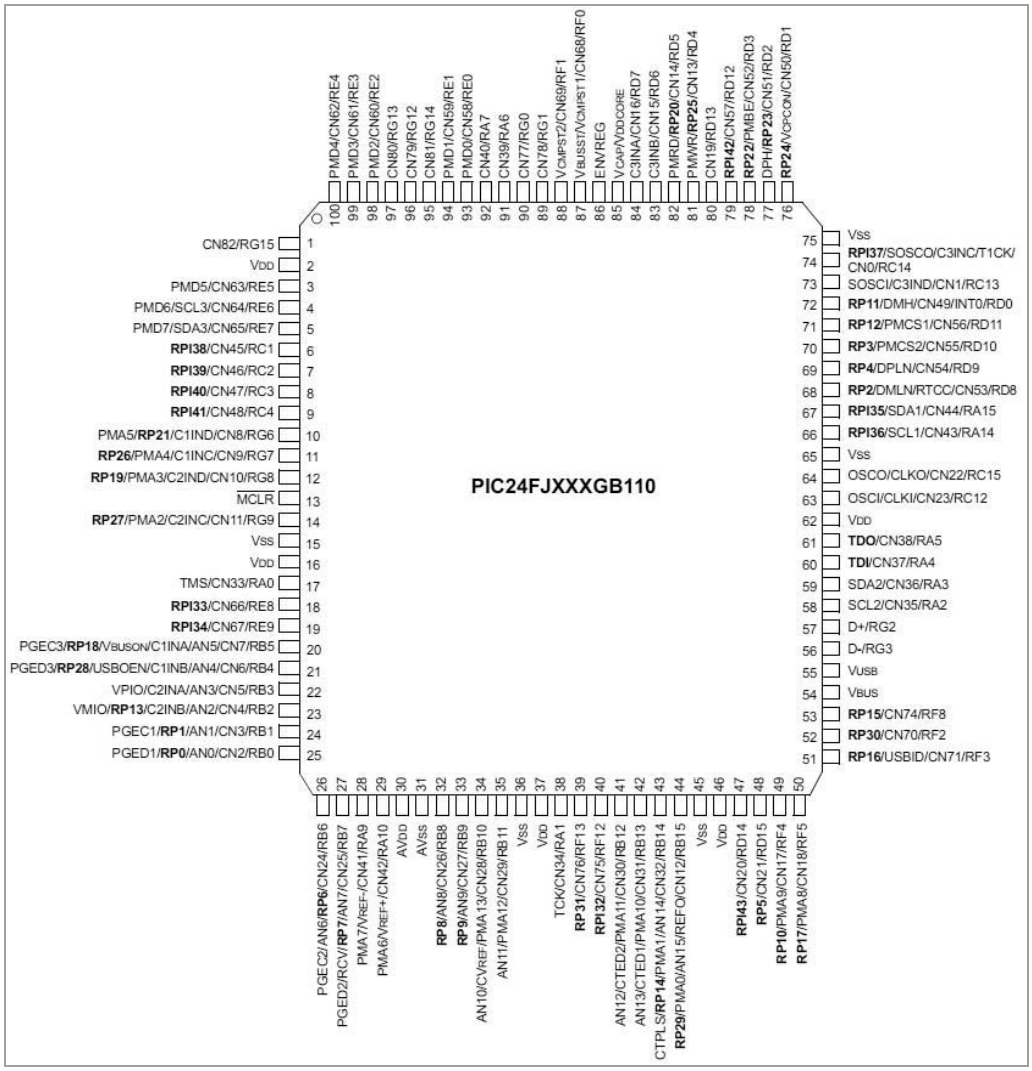


Bild 8.3: Pindiagramm des PIC24FJ256GB110

Achtung
Alle Pins, die über eine analoge Eingangsfunktionalität verfügen, sind nach dem RESET als analoge Eingänge konfiguriert, denn analoge Spannungen können an der digitalen Eingangsstufe unerwünschte Schaltvorgänge erzeugen. Man darf nicht vergessen, diese Konfiguration auszuschalten, wenn man sie als digitale Eingänge verwenden will.

9 Timermodule

Timer sind Zählregister, die entweder durch den Systemclock oder von externen Pulsen getaktet werden. Im ersten Fall dient der Timer der Erfassung und Organisation von Zeitintervallen. In diesem Fall spricht man von einer *Timerfunktion*. Im zweiten Fall spricht man von einer *Zählerfunktion* (engl. counter).

Die Zählregister werden mit TMRx bezeichnet, wobei x eine Ziffer ist, mit der die Timer nummeriert werden. Die Nummerierung beginnt mit 1, nicht mit 0 wie von den 8-Bit-PICs gewohnt. Es gibt also keinen TMR0. Die Anzahl der Timer ist unterschiedlich bei den verschiedenen Derivaten. Wie viele und welche Art von Timern es bei einem bestimmten Derivat gibt, muss man in dem jeweiligen Datenblatt nachlesen. TMR1, TMR2, TMR3 gehören zur Mindestausstattung.

Alle Timer funktionieren im Wesentlichen gleich, es gibt jedoch drei Timertypen (A, B, C), die sich in bestimmten Fähigkeiten voneinander unterscheiden. Alle Timer sind 16 Bit breit, man kann aber gewisse Timer (die Typen B und C) zu 32-Bit-Timern zusammenschalten.

9.1 Register der Timermodule

Überblick über die Register der Timermodule

TMRx	Zählregister	TMRx zählt von 0 bis PRx
PRx	Periodenregister	Timerperiode ist PRx+1
TxCON	Timercontrolregister	Zum Einschalten und Auswählen von Vorteiler und Modus

9.1.1 TxCON-Register

TON	–	TSIDL	–	–	–	–	–
–	TGATE	TCKPS1	TCKPS0	T32	TSYNC	TCS	–

Bedeutung der TxCON-Bits

Bit	*Name*	*R/W*	*Bedeutung*
15	TON	R/W	1 = On
13	TSIDL	R/W	1 = Stop im Idle-mode
6	TGATE	R/W	1 = enable gate modus
5:4	TCKPS	R/W	00 = 1:1; 01 = 1:8; 10 = 1:64; 11 = 1:256
3	T32	R/W	1 = Bilde 32 Bit Timer mit zugeordnetem Typ C (nur Typ B, sonst U)
2	TSYNC	R/W	1 = Synchronisiere Zähler (nur Typ A, sonst U)
1	TCS	R/W	1 = Zählerfunktion, steigende Flanken am PIN TxCK

9.2 Zählweise

Wer mit dem TMR1 der 8-Bit-PICs vertraut ist, sollte wissen, dass bei den 8-Bit-Controllern ist der TMR1 freilaufend ist, d. h., das Timerregister wird hochgezählt, bis es überläuft. Mit Überlauf bezeichnet man den Übergang von 0xFFFF zu 0x0000. Im Augenblick des Überlaufs wird ein Interrupt ausgelöst.

Alle Timerregister der 16-Bit-PICs haben dagegen ein Periodenregister PRx, das bestimmt, bis zu welchem Wert das Timerregister hochgezählt wird. Nach dem letzten Timerzyklus wird es auf 0 zurückgesetzt. Wir sprechen hier deshalb nicht von einem Timerüberlauf, sondern von einem Timerreset. Als Timerperiode bezeichnet man die Zeit zwischen zwei Timerresets.

Beispiel

PRx = 3: → TMRx = 0, 1, 2, 3, 0, 1, 2, 3.

Eine Timerperiode besteht aus (PRx + 1) Timerzyklen!

Achtung
Der Interrupt wird erzeugt, sobald eine Übereinstimmung des TMRx mit dem PRx stattfindet, d. h. am Anfang des letzten Timerzyklus, d. h. einen Timerzyklus vor dem Timerreset.

Setzt man das Period-Register auf 0xFFFF, verhält sich der Timer praktisch wie ein freilaufender Timer, jedoch wird der Interrupt einen Timerzyklus vor dem Timerreset ausgelöst.

9.2.1 Beispiel TMRint010: Beobachtung von Timer-Interrupt und Timerreset

Verwendeter Controller	PIC24FJ128GA010 mit 8-MHz-Quarz ohne PLL (FCY = 4 MHz)
Verwendete Module	TMR1 mit Vorteiler 1:256
Verwendete Ausgangspins	DIAG1 Portpin RB14

Das folgende Programm zeigt, dass der Timer-Interrupt einen Timerzyklus vor dem Timerreset kommt. Der Timerreset ist der Moment, in dem der Timer wieder auf 0 zurückgesetzt wird. Der Pin DIAG1 wird in der Interrupt-Routine auf high und in der Hauptschleife im Moment des Resets auf low gesetzt.

Initialisierung von TMR1

Wir verwenden den TMR1. Den Vorteiler setzen wir auf 1:256. So haben wir 256 Befehlszyklen in jedem Timerzyklus, sodass die kleinen Ungenauigkeiten durch die Software nicht stören.

Das PRx-Register setzen wir auf 3. Eine Periode besteht also aus vier Timerzyklen. Der Timer zählt 0 1 2 3 0 1 2 3 ...

```
{
  _T1IF = 0;           // T1-Flag löschen
  _T1IE = 1;           // T1 Interrupt zulassen
  PR1 = 3;             // TMR1-Periode = 4 Timerzyklen
  T1CON = 0x8030;      // TMR1 mit Vorteiler 1:256 einschalten
}
```

Interrupt-Routine

```
void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void)
{
  DIAG1 = 1;
  _T1IF=0;
}
```

Hauptschleife

```
while(1)
{
    DIAG2=0; DIAG1=0; while (TMR1==0);
    DIAG2=1; while (TMR1==1);
    DIAG2=0; while (TMR1==2);
    DIAG2=1; while (TMR1==3);
}
```

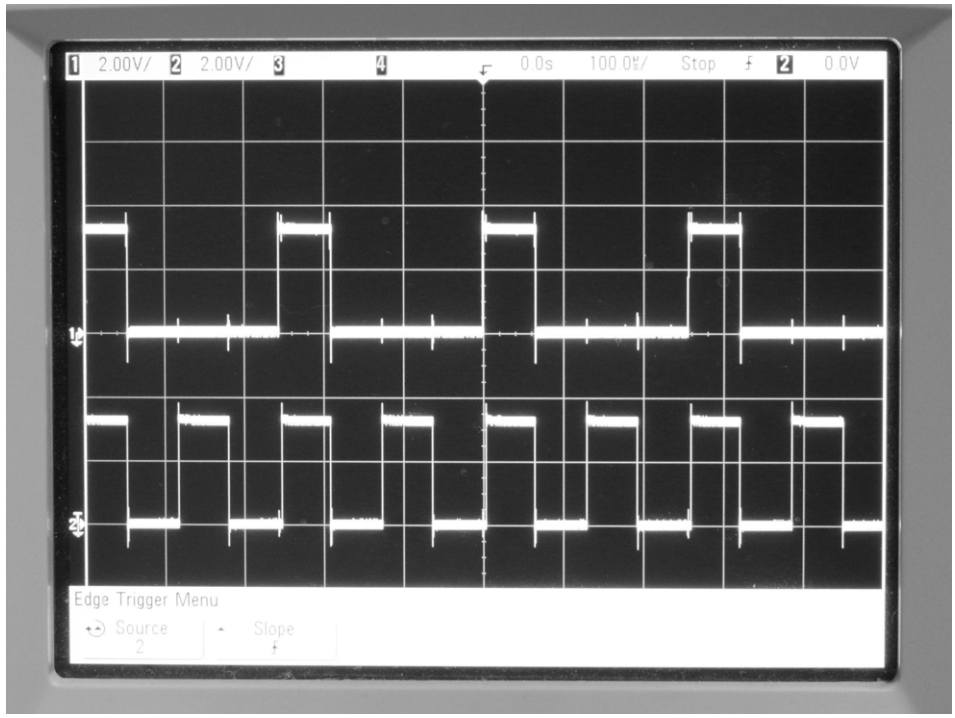


Bild 9.1: Timer-Interrupt und Timerreset

Das untere Signal des Oszilloskopbilds zeigt das Echo des niedrigsten Bits des Timers (DIAG2). Das obere Signal stellt den DIAG1-Pin dar, der im Interrupt gesetzt und im Hauptprogramm zurückgesetzt wird, sobald der Timer von 3 auf 0 springt.

9.3 Vorteiler

Alle Timermodule haben einen Vorzähler (engl. prescaler), der bewirkt, dass der Timer nicht bei jedem Befehlstakt, sondern nur bei jedem n-ten Takt (n = 1, 8, 64 oder 256) hochgezählt wird. Die Vorzählerregister können aber weder per Software gelesen noch geschrieben werden. Sie werden beim Beschreiben oder Löschen des TMRx-Registers gelöscht.

Die 4 Vorteilerwerte werden durch zwei Bits TCKPS1:TCKPS0 im Controlregister TxCON ausgewählt.

Beispiel

FOSC = 8 MHz, keine PLL, Vorteiler = 1:256

17 I²C-Bus

Der I²C-Bus dient der Anbindung des Microcontrollers an Peripheriebausteine, z. B. Speicherbausteine, LCD-Module, Sensoren oder Uhrenbausteine. Ebenso wie die SPI-Schnittstelle ist der I²C-Bus nicht für die Kommunikation zwischen Geräten geeignet.

Beim I²C-Bus handelt es sich um eine Zweidrahtleitung. Über eine der Leitungen (SDA) werden die Daten übertragen, die andere ist die Clock-Leitung (SCL). Das Management über den I²C-Bus hat ein Master, der die Initiative für die Übertragungen ergreift und den Clock erzeugt. Es gibt auch I²C-Anwendungen mit mehreren Mastern. In diesem Fall müssen sich die Master über die Prioritäten beim Zugriff über den Bus einigen. Das ist nicht immer so einfach. An einen I²C-Bus können mehrere Slaves angebunden sein. Sie werden über Adressen angesprochen, die zu Beginn einer Übertragung über die Datenleitung gesendet werden. In den typischen Anwendungen ist der PIC der Master, dennoch haben die I²C-Module viele Fähigkeiten, auch als I²C-Slaves zu fungieren.

Im Gegensatz zu anderen Schnittstellenmodulen wie UART, SPI oder PMP kann das I²C-Modul das Protokoll der Übertragung von Adressen und Daten nicht ohne Mitwirkung von Software durchführen. Die Software ist jeweils für die unterschiedlichen Gegebenheiten sehr verschiedenartig. Deshalb verweisen wir hier auf die Bibliotheksfunktionen.

17.1 Geschwindigkeit der I²C-Übertragung

Die Standardgeschwindigkeiten bei der Übertragung liegen bei 100 kHz, 400 kHz oder 1 MHz. Der Master bestimmt die Geschwindigkeit, der Slave hat aber auch die Möglichkeit, den Clock zu verzögern, falls nötig. Im Gegensatz zum UART, bei dem die Geschwindigkeit sehr gut eingehalten werden muss, wird beim I²C-Bus für einen bestimmten Baustein und eine bestimmte Versorgungsspannung eine maximale Clock-Frequenz spezifiziert, die man nicht überschreiten darf.

Die Geschwindigkeit wird durch den Wert BRG = (FCY / FSCL – FCY / 10.000.000 – 1) bestimmt. Dieser Wert muss ganzzahlig gerundet und in das Register I2CxBRG eingetragen werden. Bei der Rundung entsteht natürlich eine Ungenauigkeit, die gegebenenfalls einen gewünschten Wert von FSCL bei gegebenem FCY nicht realisierbar macht. Je größer der gewünschte Wert von FSCL ist, desto kleiner wird BRG, und umso größer wird in der Regel der relative Rundungsfehler.

Beispiel: Für FCY = 4 MHz und FSCL = 1 MHz ergibt sich BRG = 2,6. Der ganzzahlig gerundete Wert 3 ergibt einen tatsächlichen Wert von FSCL = 909 kHz. Bei FCY = 8 MHz erhält man BRG = 6,25. Der Wert von FSCL, den man durch den ganzzahlig gerundeten Wert 6 erhält, beträgt 1,026 MHz. Das sollte auch kein Problem sein.

Wer nicht gern rechnet, findet für die gängigen Werte von FCY Tabellen für die BRG-Werte in den Datenblättern.

17.2 Register der I²C-Module

Überblick über die Register der I²C-Module

I2CxCON	Controlregister
I2CxSTAT	Statusregister
I2CxBRG	Baudraten-Register
I2CxRCV	Empfangsregister
I2CxTRN	Senderegister
I2CxADD	Adresse des angesprochenen Slaves
I2CxMSK	Maskenregister der Adressen

17.2.1 I2CxCON-Register

I2CEN	–	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEM
GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN

Die Bits 4 bis 0 werden per Software gesetzt und abgefragt und per Hardware gelöscht (HC). Der I²C-Bus ist *idle*, wenn diese Bits alle gelöscht sind.

Bedeutung der Bits in I2CxCON

Bit	*Name*	*R/W*	*Bedeutung*
15	I2CEN	R/W	1: einschalten
13	I2CSIDL	R/W	1: Stop im Idle-Modus
12	SCLREL	R/W	1: SCLx freigeben, 0: SCLx halten (Slavemodus)
11	IPMIEN	R/W	1: IPMIzugelassen
10	A10M	R/W	1: 10-Bit-Adresse, 0: 7-Bit-Adresse
9	DISSLW	R/W	1: Slew rate Kontrolle ausgeschaltet
8	SMEM	R/W	1: SMBus Schwelle , 0: SMBis ausgeschaltet

Bit	Name	R/W	Bedeutung
7	GCEN	R/W	1:General call Interrupt zugelassen (Slavemodus)
6	STREN	R/W	1: Clock stretching zulassen (Slavemodus)
5	ACKDT	R/W	1: Sende Nack, 0: sende Ack
4	ACKEN	R/W, HC	Wird 1 gesetzt für Initialisierung der Ack-Sequenz, gelöscht per Hardware
3	RCEN	R/W, HC	Wird 1 gesetzt, um Receive-Modus zu starten, gelöscht per Hardware
2	PEN	R/W, HC	Wird 1 gesetzt für Initialisierung der Stop-Sequenz, gelöscht per Hardware
1	RSEN	R/W, HC	Wird 1 gesetzt für Initialisierung der Repeated-Startsequenz, gelöscht per HW
0	SEN	R/W, HC	Wird 1 gesetzt für Initialisierung der Startsequenz, gelöscht per Hardware

17.2.2 I2CxSTAT-Register

ACKSTAT	TRSTAT	–	–	–	BCL	GCSTAT	ADD10
IWCOL	I2COV	D/!A	P	S	R/!W	RBF	TBF

Bedeutung der Bits in I2CxSTAT

Bit	Name	R/W	Bedeutung
15	ACKSTAT	R-0,HSC	1: NAK vom Slave empfangen; 0: ACK empfangen
14	TRSTAT	R-0,HSC	1: der Master sendet gerade Daten
10	BCL	R/C-0,HS	1: eine Buscollision wurde erkannt
9	GCSTAT	R-0,HSC	1: general call address wurde vom Slave erkannt; Slaveaddress = 0000 0000
8	ADD10	R-0,HSC	1: die 10-Bit-Adresse hat übereingestimmt
7	IWCOL	R-0,HS	1: es wurde ins I2CxTRN geschrieben, bevor die letzte Aktion fertig war
6	I2COV	R-0,HS	1: Empfängerüberlauf
5	D_A	R-0,HSC	1: letzter Empfang waren Daten; 0: device address
4	P	R-0,HSC	1: Stop-Bit wurde erkannt
3	S	R-0,HSC	1: Startbit wurde erkannt

Bit	Name	R/W	Bedeutung
2	R_W	R-0,HSC	1: Slave erkennt einen Read-Befehl; 0: Write-Befehl
1	RBF	R-0,HSC	1: Empfang komplett, I2CxRCV voll
0	TBF	R-0,HSC	1: Sendung im Gange, I2CxTRN voll

Da sich auf dem Explorerboard 16 kein I^2C-Baustein befindet, möchten wir auf ein theoretisches Beispiel verzichten, weil es davon bei Microchip genug gibt. AN1079 ist ein solches für den dsPIC33FJ256GP710 und ein I^2C-EEPROM vom Typ 24XX256.

Anne und Manfred König

Handbuch PIC24/dsPIC-Mikrocontroller

Praxisbeispiele zur Anwendung der Module und Befehle

16-Bit-PICs sind die perfekte Lösung, wenn Sie einfache Handhabung und eine große Anwendungsbreite bei Controllern suchen.

Die Klasse der 16-Bit-PICs besitzt eine große Fülle von Typen, welche sich in ihrer Ausstattung und auch in einigen Punkten der Funktionalität unterscheiden. Der Schwerpunkt des Buches besteht darin, die gemeinsamen Eigenschaften verständlich zu machen.

Alle Typen der 16-Bit-Klasse besitzen im Wesentlichen die gleiche Architektur, die gleiche Organisation der Speicher und vor allem die gleiche Sprache bis auf die DSP-Befehle, welche nur den dsPIC-Typen vorbehalten sind.

Die dsPIC-Typen unterscheiden sich in ihren grundlegenden Eigenschaften nicht von den übrigen 16-Bit-Typen. Sie sind in ihrer Ausstattung auf schnelles Messen und auf schnelle Verarbeitung der Messergebnisse eingerichtet. Die DSP-Einheit ist nahtlos in die übrige Struktur eingebettet.

Bei den Peripheriemodulen, die zunehmend an Bedeutung gewinnen, gibt es Varianten, auf die besonders hingewiesen wird. Das Grundprinzip beim Umgang mit den Peripheriemodulen ist aber immer ähnlich. Für die Beispiele werden vier typische Vertreter der verschiedenen Varianten verwendet:

PIC24FJ128GA010
PIC24FJ256GB110
PIC24EP512GU810
dsPIC33FJ256GP710

Der praktische Umgang mit den Peripheriemodulen steht im Vordergrund der Beispiele. Hierbei sind oft feine Details zu beachten. Insbesondere die Verwendung der zugehörigen Interrupts erfordert einige Sorgfalt.

Dieses Buch ist zur Anwendung der 16-Bit- PICs und dsPICs geschrieben. Es setzt aber grundlegende Kenntnisse über die Funktion von Mikrocontrollern voraus.

Aus dem Inhalt:

- Die 16-Bit-PIC-Klasse
- PIC24-Assemblersprache
- DSP-Einheit (nur dsPIC)
- Assemblercode und C
- Umgang mit den Peripheriemodulen
- Interrupts
- DMA
- I/O-Ports
- Timermodule
- Watchdog Timer
- AD-Wandler
- Analoge Komparatoren
- Output-Compare-Module
- Input Capture
- UART-Module
- SPI-Module
- I2C-Bus
- Parallel Master Port (PMP)

9 783645 652735