PC & ELEKTRONIK

Anne und Manfred König

Handbuch PIC24/dsPIC-Mikrocontroller

Praxisbeispiele zur Anwendung der Module und Befehle

FRANZIS

PC & ELEKTRONIK

Anne und Manfred König

Handbuch PIC24/dsPIC-Mikrocontroller

Praxisbeispiele zur Anwendung der Module und Befehle

FRANZIS

Inhaltsverzeichnis

1 Die 16-Bit-PIC-Klasse

Die Bezeichnung »16-Bit-PIC« bezieht sich auf die Breite des Datenbusses. Diese Datenbreite ist in vielen Anwendungen sehr praktisch. Der hauptsächliche Gewinn an Komfort gegenüber den 8-Bit-PICs ist aber dem 24 Bit breiten Befehlsformat zu verdanken. Dadurch sind größere Adressräume möglich – sowohl bei den Daten als auch bei den Befehlen. Außerdem erlauben die breiteren Programmworte effizientere Adressierungsarten.

Die Klasse besteht aus den Familien

- PIC24FJ
- PIC24HJ
- PIC24EP

und den dsPICs, die zusätzlich die DSP-Einheit besitzen:

- dsPIC30FJ
- dsPIC33FJ
- dsPIC33EP

Mit den DSP-Befehlen können lineare mathematische Algorithmen, z.B. für Filter, Regelprozesse oder Fourier-Funktionen, effektiv unterstützt werden.

Die Familien unterscheiden sich physikalisch und in schaltungstechnischen Einzelheiten. In jeder Familie gibt es eine große Anzahl von Derivaten, die sich in Größe und Ausstattung beträchtlich unterscheiden.

Alle Derivate der 16-Bit-PIC-Klasse haben aber die gleiche Architektur. Die CPU arbeitet auf die gleiche Weise. Die Befehle sind die gleichen, bis auf die dsPIC-Befehle, die zusätzlich zu den gemeinsamen Befehlen implementiert sind. Weitgehend gleich ist auch das interne Management, z.B. die Organisation der Interrupts oder die Zusammenarbeit der CPU mit den Peripheriemodulen.

In diesem Buch werden vorwiegend die gemeinsamen Eigenschaften beschrieben. Wichtig ist aber auch, die relevanten Unterschiede zu kennen. Man muss unbedingt wissen, welche Details man im jeweiligen Datenbuch nachlesen muss und worauf man sich höchstwahrscheinlich bei allen Derivaten verlassen kann. Man weiß natürlich nie, was den Entwicklern der 16-Bit-Klasse in der Zukunft einfällt.

Die Unterschiede in den Derivaten betreffen hauptsächlich folgende Eigenschaften:

- Die Ausrüstung mit Peripheriemodulen, die in den Datenblättern als *Peripherals* bezeichnet werden. Dabei geht es nicht nur um die Frage, ob bestimmte Peripheriemodule vorhanden sind und ggf. in welcher Anzahl. Die Module unterscheiden sich auch oft in ihrer Komplexität und Leistungsfähigkeit.
- Die Ausstattung mit Programm- und Datenspeicher. Große Datenspeicher erfordern oft ein komplexeres Management des Adressraums.
- Die PPS(Peripheral Pin Select)-Eigenschaft. D.h., dass die Pins, die den Peripheriemodulen zugeordnet sind, in weiten Grenzen per Software selektiert werden können. Diese Eigenschaft ist oft ausschlaggebend für die Auswahl eines PIC-Derivats. Sie erleichtert nicht nur das Layout, sondern ermöglicht auch Schaltungstricks, die ohne PPS nicht möglich sind.

- Die komfortable DSP-Einheit, die dsPICs zusätzlich besitzen. Man kann sie aber ansonsten in schlichten Ausführungen erhalten.

Ein Schwerpunkt wird in diesem Buch auf die Eigenschaften und die Bedienung der Peripheriemodule gelegt. Es gibt viele Anwendungen, bei denen die Hauptarbeit in ihren Händen liegt und die CPU vorwiegend mit dem Management der Module beschäftigt ist. Die Peripheriemodule können im Wesentlichen mit dem Sprachumfang von C bedient werden.

Über die Beschreibung der inneren Organisation, z.B. die Adressierung der Speicher sowie den Aufbau der Assembler-Befehle, wird ebenfalls ausführlich berichtet.

Vier unterschiedliche Beispiel-PICs

Wir haben unsere Beispiele jeweils an einem von vier typischen, aber deutlich unterschiedlichen Derivaten ausgearbeitet:

- PIC24FJ128GA010
- PIC24FJ256GB110
- PIC24EP512GU810
- dsPIC33FJ256GP710

Ausstattung der vier Beispiel-PICs

	PIC24FJ128GA010	*PIC24FJ256GB110*	*PIC24EP512GU810*	*dsPIC33FJ256GP710*
IO-Pins	85	84	83	85
Programm-speicher	128 kB	256 kB	512 kB	256 kB
SRAM	8 kB	16 kB	52 kB	30 kB
MIPS	16 MIPS	16 MIPS	70 MIPS	40 MIPS
PLL	4x	4x	Variabel bis zu 40-fach	4x
16-Bit-Timer	5	5	9	9
Input capture	5	9 (dedicated)	16 (dedicated)	8
Output compare	5	9 (dedicated)	16 (dedicated)	8
UART	2 + IrDA	4 + IrDA	4 + IrDA	2 + IrDA
SPI	2 std / enh	3 std / enh	4 std / enh	2 std
I2C	2	3	2	2
AD-Wandler	10 Bit/16 Kanäle	10 Bit/16 Kanäle	2x 10/12 Bit/32 Kanäle	2x 10/12 Bit/32 Kanäle
Komparator	Dual Comp.	Triple Comp.	Triple Comp.	–
PMP / PSP	Ja	Ja	Ja	–
PPS	–	32 RP, 12 RPI	30 RP, 47 RPI	–
CTMU	–	1	–	–
USBOTG	–	1	1	–
ECAN	–		2	2
CRC-Generator	–	–	1	–
DMA	–	–	15	8

Falls dem Explorerboard 16 der dsPIC...710A beigelegt ist, ist das kein Problem. Sie sind fast identisch, auf jeden Fall abwärtskompatibel. D.h., dass in den existierenden Projekten nur der richtige dsPIC angemeldet und der Name des entsprechenden Include-Files um das »A« erweitert werden muss.

Microchip Explorerboard16

Um eine schnelle Möglichkeit zum Nachvollziehen unserer Beispiele zu schaffen, haben wir alle mit dem Explorerboard16 durchgeführt. Selbst eine entsprechende Platine aufzubauen kostet mehr Geld und vor allem Zeit.

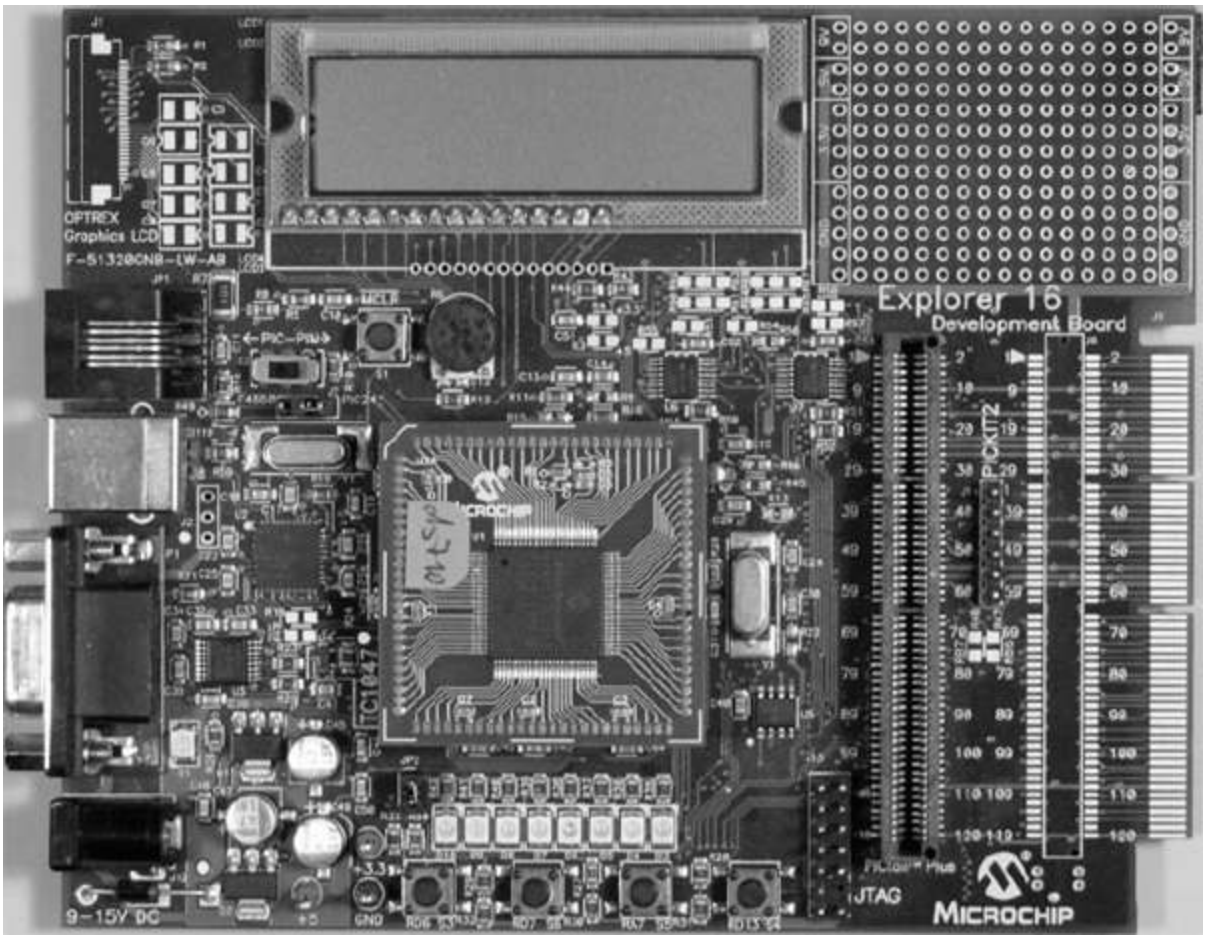

Bild 1.1: Explorerboard16

Einige Eigenschaften des Explorerboards

- Stiftleisten für die Aufnahme von PIMs (Plug-in-Module); damit können unterschiedlichste Controller schnell aufgesteckt werden. Zwei PIMs, PIC24FJ128GA010 und dsPIC33FJ256GP710, sind bereits inklusive.
- Stromversorgungsbuchse für 9–15 V ungeregelt zur verpolungssicheren Versorgung aller Komponenten

- RS232-Port inklusive V24-Treiber zum direkten Anschluss an den PC oder über einen V.24/USB-Wandler, um PIC-Ausgaben direkt in einem Terminalprogramm sehen zu können; Eingaben an den PIC sind natürlich auch möglich.
- Temperatursensor vom Typ TC1074 mit analogem Ausgang direkt mit Pin RB4 (AN4) verbunden
- USB-Anschluss
- ICD-Steckverbinder zum Programmieren und Debuggen; mögliche Debugger sind ICD3, PICKit3 oder auch Real-ICE
- Zweizeiliges LCD inklusive Controller zur Textausgabe
- Anschlussmöglichkeit für ein Grafik-LCD
- 4 Kurzhubtaster für den Benutzer auf den Pins RD6, RD13, RA7 und RD7
- 1 analoges Poti am Pin RB5 (AN5)
- 8 LEDs an den Pins RA0 bis RA7
- 2 Oszillatormöglichkeiten: ein 8-MHz-Quarz als XT-Osc und ein 32.768-kHz-Uhrenquarz für das PIC-interne RTCC-Modul
- 1 serielles EEPROM vom Typ 25LC256 mit 256 kBit Speicher; eine gute SPI-Bus-Demonstration.
- Passende Steckverbindung für den Programmer/Debugger PICkit3
- 3 Erweiterungsteckverbinder für EdgeCardModular und PICtail/Plus zur Erledigung fast aller Eventualitäten. Möglichkeit, eine Vielzahl von angebotenen Erweiterungsmodulen anzuschließen

Beispielprojekte

Zu jedem ausgearbeiteten Beispiel gibt es ein eigenständiges Projekt. Wir erstellen immer zu jedem

Projekt zwei Dateien, die die Eigenschaften aller Portpins festlegen. Diese Dateien sind zum Teil von dokumentarischer Bedeutung.

PPPiop.h-Datei

In der ersten Datei mit dem Namen PPPiop.h (PPP ist der Projektname) stehen alle Deklarationen und Definitionen, die für die Festlegung der (anfänglichen) Eigenschaften aller Pins benötigt werden. Dazu gehört die Festlegung, ob die Pins Ein- oder Ausgänge sind, ob sie analog oder digital sind und ggf. Open-drain-Eigenschaften haben. Die tatsächlich verwendeten Pins erhalten immer einen Namen. Bei PICs mit PPS (Port Pin Select) gibt es auch noch Deklarationen, die für die Zuordnung zur Funktionalität notwendig sind. Diese Deklarationen werden für jeden Port durchgeführt. Auch nicht verwendete Pins werden dabei berücksichtigt.

Auch die zugehörigen Prototypen der Funktionen, die die deklarierten Werte nach dem RESET in die zuständigen Register schreiben, befinden sich in der PPiop.h-Datei.

PPPiop.c-Datei

Die PPPiop.c-Datei enthält alle Funktionen, mit denen die Anfangszustände der Portpins nach dem RESET in die zuständigen Register geladen werden. Bei PICs mit PPS-Fähigkeit enthält sie auch die Funktionen, die den Pins, die wählbaren Peripheriemodulen zugeordnet werden können, die richtige Funktion zuordnet.

Liste der Beispiele auf der beigefügten CD-ROM

Modul	*Projektname*	*PIC*	*Out-Pins*	*In-Pins*	*Ergebnis*
ADC + CMP	ADCMP010	GA010	Led3	Poti	LCD-Modul
ADC	ADdma710	GP710		TEMP	Debug
ADC	ADint010	GA010		TEMP	Debug
ADC	ADsingle010	GA010		Poti	LCD-Modul
FLASH	Flash010	GA010	–	–	Debug
FLASH	Flash810	GU110	–	–	Debug
LCD	LCD_GA010	GA010	–	–	LCD
LCD	LCD_GB110	GB110	–	–	LCD
LCD	LCD_GP710	GP710	–	–	LCD
LCD	LCD_GU810	GU810	–	–	LCD
OC	OCPULSE	GA010	OC1		Oszilloskop
OC	OCPPULSE	GA010	OC1, diag1		Oszilloskop
PMP	PMP010	GA010		PMP	Oszilloskop
OCded	PWMFLT110	GB110		SW4	Oszilloskop
SPI	SPIEE010	GA010			Debug
TMR2	T2GATE010	GA010	diag1, diag2	T2CK	Oszilloskop
TMR1	TMRINT010	GA010	diag1, diag2		Oszilloskop
UART	UART010	GA010	U2TX	U2RX	PC, Terminal

Entwicklungsumgebung

Die Beispielprojekte in diesem Buch wurden unter Windows XP mit MPLAB 8.88 erstellt. Der verwendete C-Compiler ist der C30/XC16 V3.31 von Microchip. Die Projekte sind auch im MPLABX mit dem XC16 V1.21 getestet worden. Die Compiler C30/XC16 verwenden die gleiche Form des Quellcodes und ergeben den gleichen Assemblercode.

Nach dem Importieren ins MPLABX muss im Projekt der verwendete Compiler ausgewählt werden. Der C30/XC16 ist übrigens im MPLABX problemlos verwendbar. Einziger Nachteil: Er wird nicht mehr weiterentwickelt.

Als Programmer bzw. Debugger haben wir den RealICE verwendet. Die billigere Variante, der PICKit3, kann genauso

gut eingesetzt werden, sowohl als Programmer als auch als Debugger.

Architektur

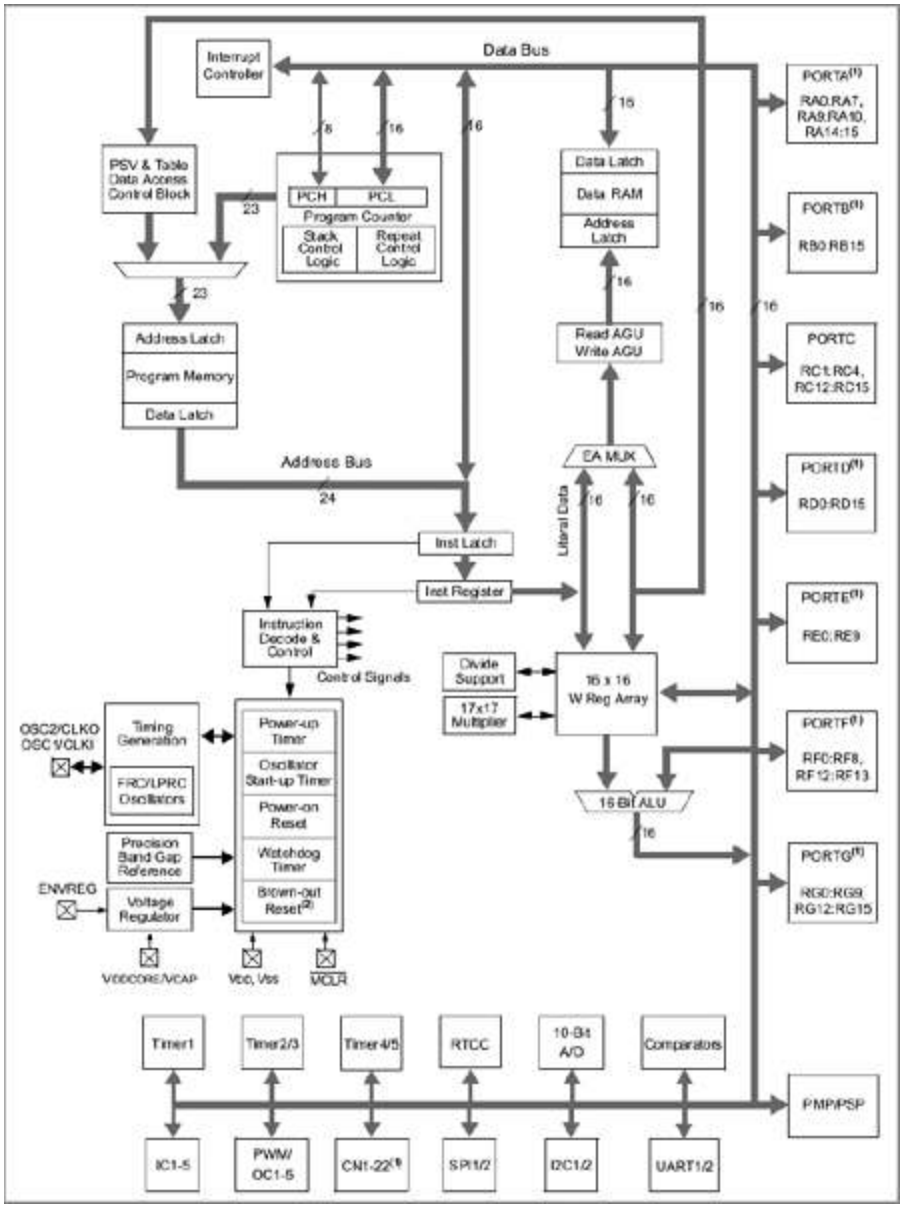

Bild 1.2: Blockdiagramm des PIC24FJ128GA010

Wer mit 8-Bit-PICs vertraut ist, wird erkennen, dass die typische PIC-Philosophie erhalten wurde:

- Die Busse zu den Daten-und Programmspeichern sind voneinander getrennt.

- Die Befehle sind (mit einigen Ausnahmen) komplett in einem Programmwort enthalten, d.h. sowohl die Befehlscodes als auch die Argumente.
- Die Befehle werden (mit einigen Ausnahmen) in einem einzigen Befehlszyklus abgearbeitet.
- Eine Fülle von Peripheriemodulen führen ohne Beteiligung der CPU eine Menge nützlicher Funktionen aus.
- Neben den normalen Datenregistern gibt es einen Bereich von *Special Function Register* (SFR), über die die Software mit den Peripheriemodulen kommuniziert.

Durch den 16 Bit breiten Datenbus und vor allem durch die 24 Bit breiten Programmworte ist aber ein kleiner Quantensprung gegenüber den 8-Bit-Vorgängern gelungen.

- In dem 24 Bit breiten Programmwort lässt sich eine Fülle von Adressierungsarten realisieren. Der direkt adressierbare Datenraum beträgt 8 kBytes. Bis zu 64 kBytes (32 K Datenworte) können indirekt über Zeigerregister adressiert werden.
- Obwohl der Datenbus 16 Bit breit ist, können fast alle Befehle, die Datenargumente besitzen, auch mit Byte-Argumenten ausgeführt werden.
- Im Bereich der SFR befinden sich 16 Arbeitsregister (W0 bis W15), die als bevorzugte Datenregister oder als komplexe Zeigerregister dienen und die auch sonstige Spezialaufgaben wahrnehmen.
- Es gibt bis zu 126 Interrupt-Quellen. Jede Interrupt-Quelle hat einen eigenen Interrupt-Vektor. Jeder Interrupt-Quelle ist ein Prioritätslevel zugeordnet. Unter den Interrupt-Quellen befinden sich 8 Fehler-Interrupts (Traps), die nicht maskierbar sind.

- Daten, die im Programmspeicher liegen, können wahlweise mit den gleichen Befehlen gelesen werden, als würden sie im Datenspeicher liegen. Zu diesem Zweck kann ein Teil der Datenadressen auf den Programmbus umgeleitet werden. Diese Fähigkeit nennt man PSV (Program Space Visibility).

dsPICs

Die dsPICs besitzen zusätzlich die DSP-Einheit, mit der lineare mathematische Algorithmen z.B. für Filter, Regelprozesse oder Fourier-Funktionen effektiv unterstützt werden können.

Die DSP-Einheit ist nahtlos in die Architektur der 16-Bit-PIC-Familie integriert. Sie besteht aus zwei 40 Bit breiten Akkumulatoren, die in Zusammenwirken mit einigen sehr mächtigen Befehlen die Zwischenergebnisse der Algorithmen speichern und verwalten.

Der Datenspeicher der dsPICs ist in einen X-Bereich und einen Y-Bereich aufgeteilt, sodass mit einem einzigen Befehl auf zwei Register gleichzeitig zugegriffen werden kann. Mit einem Befehl, der nur einen Befehlszyklus dauert, kann beispielsweise gleichzeitig die Multiplikation zweier Register ausgeführt und in einem der beiden Akkumulatoren aufsummiert werden, und zusätzlich werden die Werte für die nächste Multiplikation aus Tabellen im X- und Y-Bereich in die Multiplikatorregister geladen.

Peripheriemodule

Peripheriemodule sind selbstständige elektronische Schaltungen innerhalb des Mikrocontrollers, die ihre Arbeit ohne direktes Zutun der CPU durchführen. Die CPU hat dabei nur die Aufgabe, diese Module zu konfigurieren und eventuell ihre Ergebnisse zu bearbeiten und sie zu steuern. Die Module können wahlweise auch dann arbeiten, wenn die CPU zu Stromsparzwecken ausgeschaltet ist.

Die Bedienung der Peripheriemodule besteht immer aus zwei Teilen: den vorbereitenden Konfigurationen, die man häufig nur am Anfang eines Programms durchführt, und der Bedienung im Verlauf des weiteren Programms.

Der Trend geht zu immer mehr Komfort, was immer mehr Wahlmöglichkeiten zur Folge hat. Das ist einerseits zwar erfreulich, andererseits erfordert die Bedienung eines Peripheriemoduls oft ein längeres Studium der Datenblätter, die nicht immer zum Ziel haben, den Überblick zu erleichtern.

Standardperipheriemodule, die fast immer zu finden sind:

- Timer
- Output-Compare-Module
- Input-Capture-Module
- Watchdog Timer
- AD-Wandler
- Komparatoren
- UART-Module
- SPI-Module
- I^2C -Module

Das DMA-Modul, das bei manchen Derivaten eine wichtige Rolle spielt, wird nicht als Peripheriemodul bezeichnet.

1.1 Oszillatoren

Oszillatoren gehören zu den wichtigsten Elementen eines Controllers. Ohne den Takt eines Oszillators kann die CPU nicht arbeiten. Auch die meisten Peripheriemodule benötigen einen Clock.

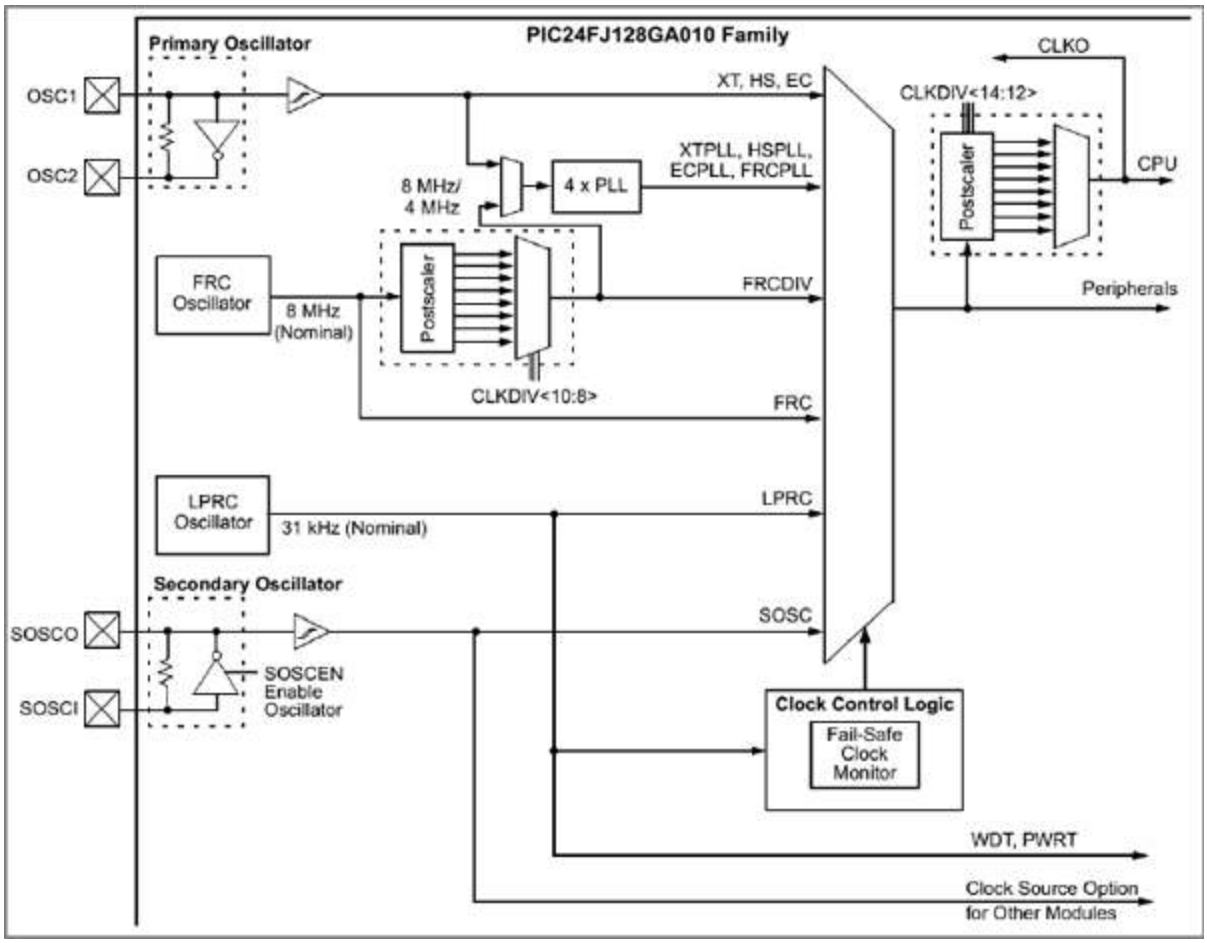

Bild 1.3: typischer Oszillatorblock

Die Ansprüche an den Oszillator können je nach Anwendung sehr unterschiedlich sein. Die hauptsächlichen Aspekte sind:

- Taktfrequenz
- Präzision
- Schaltungs- bzw. Bauteilaufwand

Bezüglich der Taktfrequenz sollte man bedenken, dass es die Schaltvorgänge sind, die viel Strom »verbrauchen«. Je höher die Taktfrequenz, desto höher der Stromverbrauch und die damit verbundene Wärmeentwicklung.

Die 16-Bit-PIC-Familie besitzt 4 verschiedene Taktquellen (engl. Clock Sources), die alle alternativ als Systemtakt

fungieren können. Sie können je nach Bedarf per Software umgeschaltet werden:

- POSC – primärer Oszillator an den Pins OSC1 und OSC2: Quarz oder Resonator von 3–32 MHz
- SOSC – sekundärer Oszillator an den Pins SOSCI und SOSCO: 31–33 kHz
- FRC – schneller interner Oszillator (Fast RC): 8 MHz, +-2%, schlimmstenfalls +-5%
- LPRC – langsamer interner Oszillator (Low Power RC): 31 kHz, +-20%

Die Frequenz der Taktquelle wird mit FOSC bezeichnet.

Aus dem Oszillatorblockschaltbild ist nicht ersichtlich, dass FOSC noch einmal durch 2 geteilt wird, bevor daraus die interne Frequenz (der Systemtakt) erzeugt wird. Der Systemtakt wird mit *FCY* bezeichnet. Jedoch kann man erkennen, dass die Frequenz FOSC im Fall des FRC noch einen wählbaren Postscaler besitzt. Im Fall des FRC und des POSC kann sie durch den PLL-Block laufen, wo sie verdoppelt bzw. vervierfacht wird, bevor daraus der Systemtakt wird. Bei den neuen, schnelleren PIC24EP/dsPIC33EP ist die PLL deutlich flexibler, und sie ist nicht auf eine Vervierfachung beschränkt.

Mit dem Systemtakt FCY wird normalerweise die CPU gespeist. Eine Ausnahme ist der wählbare DOZE-Modus, in dem ein Postscaler vor die CPU geschaltet wird, um Energie zu sparen. Der Systemtakt bleibt unverändert.

Mit dem Systemtakt FCY werden auch die Peripheriemodule gespeist. Eine Ausnahme ist das interne RTCC-Modul, das vom SOSC gespeist wird, und ein bestimmter TIMER-Typ, der direkt vom SOSC getaktet werden kann. Der DOZE-Modus betrifft nur die CPU und nicht die Peripheriemodule.

1.1.1 Auswahl des Systemtakts

Der Systemtakt kann per Software jederzeit verändert werden. Beim Umschalten des Systemtakts müssen ggf. größere Verzögerungszeiten in Kauf genommen werden. Um per Software den Systemtakt zu verändern, muss die Nummer des neuen Typs in die drei Bits NOSC im OSCCON-Register geschrieben werden. Diese Register sind jedoch gegen unbeabsichtigte Veränderungen geschützt. Das Schreiben von NOSC muss per Konfiguration zugelassen sein, und vor dem Schreiben muss das OSCCON-Register entriegelt werden.

NOSC	*Systemtakt*	*Kurzname*
000	Fast RC	FRC
001	Fast RC mit PLL	FRCPLL
010	Primary Oszillator	EC / XT / HS
011	Primary Oszillator mit PLL	ECPLL / XTPLL / HSPLL
100	Secondary Oszillator	SOSC
101	Low Power RC	LPRC
110	Reserviert	–
111	Fast RC mit Postscaler	FRCDIV

Nach dem RESET wird der Oszillator als Systemtakt gewählt, der im Konfigurationsregister CW2 in den drei Bits FNOSC angegeben wird. Diese Bits werden nach dem RESET in das OSCCON-Register übernommen.

Falls ein primärer Oszillator angeschlossen ist, muss der Typ dieses Oszillators im Konfigurationsregister des CW2 angegeben werden. Dazu dienen die zwei POSCMD-Bits im CW2-Register:

POSCMD	*Oszillator-Typ*	*Kurzname*
00	Externer Oszillator	EC
01	Quarz oder Resonator <= 10 MHz	XT
10	Quarz oder Resonator >= 10 MHz	HS
11	Kein primärer Oszillator	–

Wenn ein primärer Oszillator mit PLL für den Start vorprogrammiert ist, darf die Quarzfrequenz nicht viel höher als 4 MHz liegen. Wenn die PLL erst nach dem Start eingeschaltet wird, darf die Quarzfrequenz maximal 8 MHz betragen.

Wenn kein externer primärer Oszillator existiert, ist OSC1 ein normaler I/O-Pin. Der Pin OSC2 kann wahlweise als Clock-Ausgang dienen (Frequenz FOSC/2). Das Bit OSCIOFCN im CW2-Register muss dazu = 1 gesetzt werden. Wenn OSCIOFCN den Wert 0 hat, ist der Pin OSC2 ein normaler I/O-Pin.

1.1.2 Umschalten des Systemtakts

Während des Umschaltprozesses läuft die Software mit dem alten Takt weiter, und zwar so lange, bis der neue Takt stabil ist. Dann wird umgeschaltet und ggf. die alte Taktquelle abgeschaltet. Bei den internen Oszillatoren dauert es nur wenige Mikrosekunden, bis der Takt stabil ist, in der Regel beim FRC 15 µs und beim LPRC zwischen 40 und 70 µs. Das Zuschalten einer PLL allerdings dauert je nach PIC-Typ 1,5 bis 20 ms. Wenn diese Situation von Interesse ist, sollte unbedingt das zuständige Datenblatt nach den konkreten Werten angesehen werden. Wenn der neue Oszillator ein XT oder HS ist, bestimmt dessen Start-up-Zeit (1.000 Takte) die Dauer des Umschaltens.

Das Umschalten des Systemtakts per Software ist nur möglich, wenn es in der Konfiguration explizit erlaubt wird. Es geschieht durch Löschen des Bits FCKSM1. Je nach PIC-Typ ist es im Konfigurationsregister CW2 oder FOSC zu finden.

Softwareprozedere

Der Start des Umschaltprozesses auf einen neuen Oszillator geschieht in zwei Schritten:

- Schreiben der Nummer des neuen Oszillators in die NOSC-Bits von OSCCON
- Setzen des OSWEN-Bits im OSCCON-Register

Das OSWEN-Bit bleibt gesetzt, bis der Umschaltprozess vollendet ist. Danach wird es von der Hardware gelöscht. Fall ein Umschaltprozess aus irgendwelchen Gründen nicht durchführbar ist, bleibt das Bit ohne Time-out gesetzt. Es wird geraten, das OSWEN-Bit nach einem Umschalten des Oszillators zu überwachen.

Jedes Schreiben in das OSCCON-Register ist nur unmittelbar nach einer Entriegelungsprozedur möglich (d.h. innerhalb des nächsten Befehlszyklus). Die Entriegelung geschieht separat für das höhere und das niedrigere Byte von OSCCON. Alle Schreibbefehle müssen unmittelbar hintereinander erfolgen, sodass man diesen Prozess unbedingt in Assembler schreiben muss. Ein DISI-#n-Befehl zu Beginn sollte sicherstellen, dass kein Interrupt dazwischenkommt.

```
disi  #13     ; Interrupts für 13 Befehle sperren
mov   #0x15,w0    ; neue Oszillatorauswahl(NOSC=0b101) nach W0
 schreiben

mov   #OSCCONH, w1    ; Entriegelungsprozedur für OSCCONH durc
hführen
mov   #0x78, w2
mov   #0x9A, w3
mov.b w2, [w1]
mov.b w3, [w1]

mov.b w0, [w1]    ; neue Oszillatorauswahl nach OSCCONH laden

mov   #OSCCONL, w1    ; Entriegelungsprozedur für OSCCONL durc
hführen
mov   #0x46, w2
mov   #0x57, w3
mov.b w2, [w1]
mov.b w3, [w1]

bset  OSCCON, #0  ; Oszillatorumschaltung starten
```

Bei Verwendung der builtin-Funktionen sieht das so aus:

```
__builtin_write_OSCCONH(newosc);
__builtin_write_OSCCONL(0x1);
```

Es ist nicht möglich, von einem in einen anderen PLL-Modus umzuschalten. Von XT_PLL in FRC_PLL z.B. muss man über FRC gehen, also zwei Umschaltvorgänge durchführen.

1.1.3 Fail Save Clock Monitor (FSCM)

Der FSCM ist eine Hardwareeinrichtung, die automatisch auf den internen FRC umschaltet, sobald ein Fehler im aktuellen Takt entdeckt wird. Das kann auftreten, wenn ein externer Takt ausgewählt wurde und die Verbindung zu diesem unterbrochen wird, oder bei einem Quarzoszillator, wenn durch einen mechanischen Stoß der Quarz bricht.