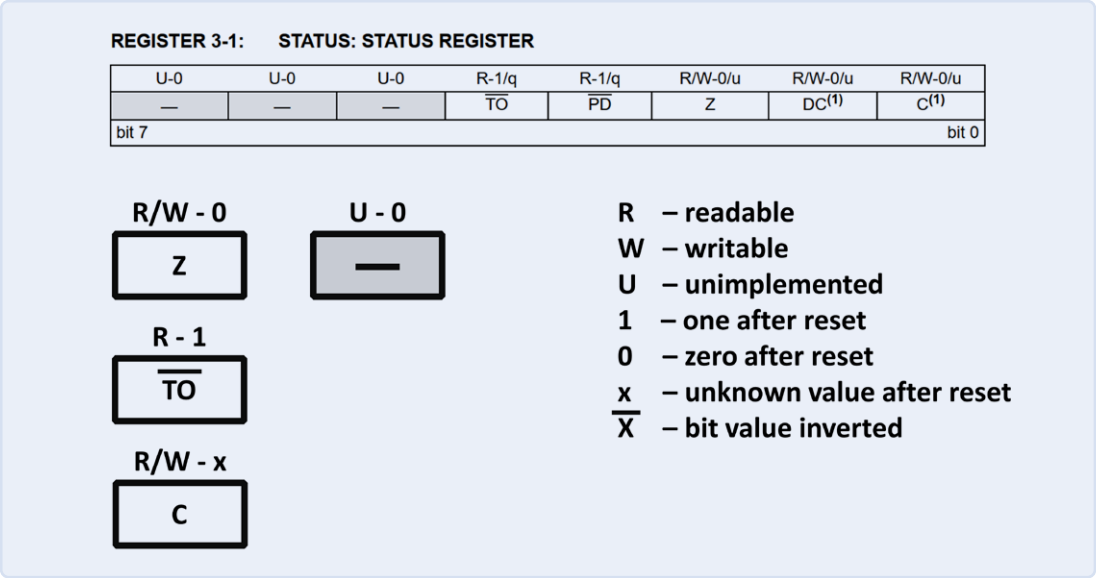
# Microcontroller-documentatie verklaard

## deel 2: registers en blokschema's




*Figuur 1. De beschrijving van de bits van een register kan er ingewikkeld uitzien (bron: Microchip Technology).*

**Stuart Cording (Elektor)**

Of we het nu leuk vinden of niet, we zullen met microcontroller-documentatie moeten werken. In het eerste deel van deze artikelreeks [1] hebben we gezien welke informatie er in de datasheet van een microcontroller te vinden is. In deze aflevering gaan we een stuk dieper en kijken we hoe gebruikers worden geïnformeerd over de functionaliteit van de chip, zoals de registers en de blokschema's.

### De eerste pagina's van de datasheet

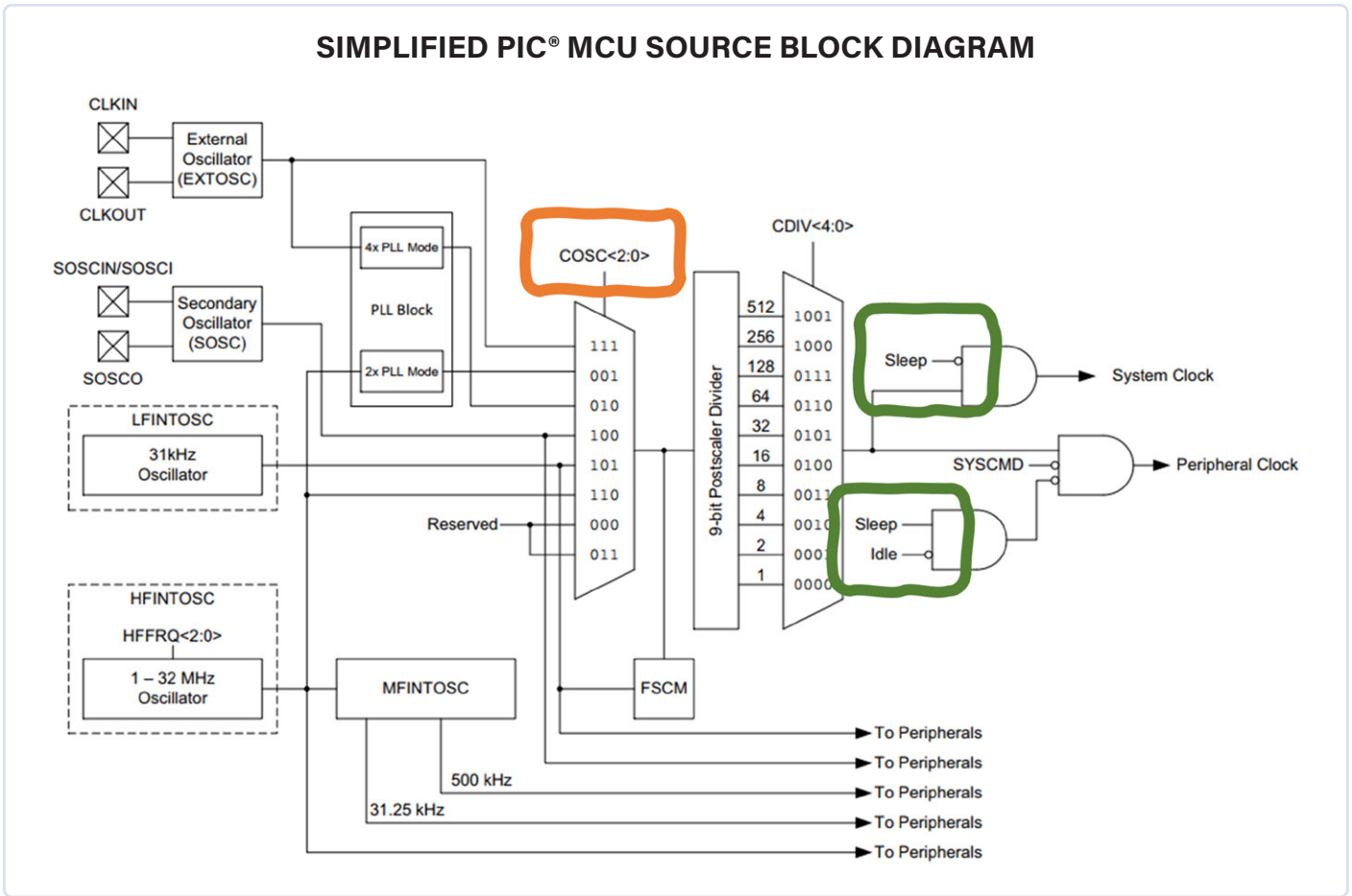
Nu we weten wat er zoal in een datasheet te vinden is, kunnen we naar de details gaan kijken. Datasheets beschrijven de zaken op een wat eigenaardige manier, en elke fabrikant doet dat weer op een eigen manier. We blijven ons concentreren op de 8-bits PIC16F18877-microcontroller van Microchip Technology [2], dus het is van belang om die datasheet te downloaden zodat je dit verhaal kunt volgen.

Na het downloaden en openen van de datasheet word je geconfronteerd met de eerste twee pagina's, die een korte beschrijving geven van de microcontroller, plus een overzicht van de belangrijkste eigenschappen en de periferie. De *Core Features* vormen geen onafhankelijke periferie; het gaat om functies die nauw verbonden zijn met de processorkern. Het gaat om functies zoals de brown-out schakeling die detecteert wanneer de voedingsspanning onder het vereiste niveau daalt, en watchdog-timers, die de microcontroller resetten en vaak worden gebruikt in veiligheidskritische systemen.

De derde pagina komt al meer ter zake en vertelt hoeveel en welke periferie we krijgen, en hoeveel RAM, flashgeheugen en EEPROM aanwezig is. De pagina's vier tot en met zeven geven basisinformatie over de behuizingen en de pinning van de diverse uitvoeringen. De *Pin Allocation Tables* zijn een essentieel onderdeel van elke microcontroller-datasheet. Met zoveel mogelijkheden op elke chip hebben microcontrollers tegenwoordig meer functionaliteit dan aansluitpinnen. Daarom kunnen verschillende mogelijkheden (en soms zelfs meer dan één) worden toegewezen aan een enkele pin. Bij deze controller zien we dat pen RA2 kan worden gebruikt als een ADC-ingang, een analoge spanningsreferentie, een comparator-ingang, een DAC-uitgang of als een interrupt-pin. Soms kan een bepaalde functie worden toegewezen aan verschillende pinnen, wat meer flexibiliteit biedt. De toewijzing van functionaliteit aan pinnen kan dus bijzonder ingewikkeld zijn.

## Registerbeschrijvingen begrijpen

Registerbeschrijvingen bestaan uit twee delen: de naam van het register en de namen van de bits (of bitgroepen) in het register. Pagina 38 van deze datasheet beschrijft de functionaliteit van het STATUS-register in de processor. Dit register bestaat uit bits die zijn geïmplementeerd en bits die niet zijn geïmplementeerd. De geïmplementeerde bits zijn gemarkeerd met een combinatie van R en/of W die aangeeft of we ze kunnen lezen en/of schrijven. Na het streepje komt meestal een cijfer: "0" of "1". Dit geeft de waarde van dit bit aan na een reset. Bij sommige bits staat een *x*, wat aangeeft dat de waarde na een reset onbekend is. De waarde van sommige bits is afhankelijk van welke gebeurtenis de microcontroller uit zijn reset-toestand heeft gehaald. Als de microcontroller gewoon is ingeschakeld, is hij gewekt door een power-on reset (POR). Als de voeding een brown-out reset (BOR) heeft veroorzaakt, kan er een andere waarde in een specifiek bit staan. In dit voorbeeld zijn zulke "reset-afhankelijke" bits gemarkeerd met een *q*.
Niet-geïmplementeerde bits kunnen soms gevaarlijk zijn. Volgens deze datasheet zouden de hoogste drie een "0" moeten bevatten. Maar het is niet duidelijk wat er gebeurt als je naar deze bits probeert te schrijven. De datasheet zou een algemene aanbeveling kunnen geven. Als die er niet is, is het waarschijnlijk verstandig om ervoor




*Figuur 2. Blokschema van de klokperiferie, waarin de bron van de kloksignalen kan worden geselecteerd en de kloksignalen kunnen worden in- en uitgeschakeld (bron: Microchip Technology).*

### Niet-geïmplementeerde bits nul maken

```
myValue &= 0b00011111;     // clear 3 most-significant bits
                           // (unimplemented bits)
STATUS = myValue;          // write 'myValue' into STATUS register
```

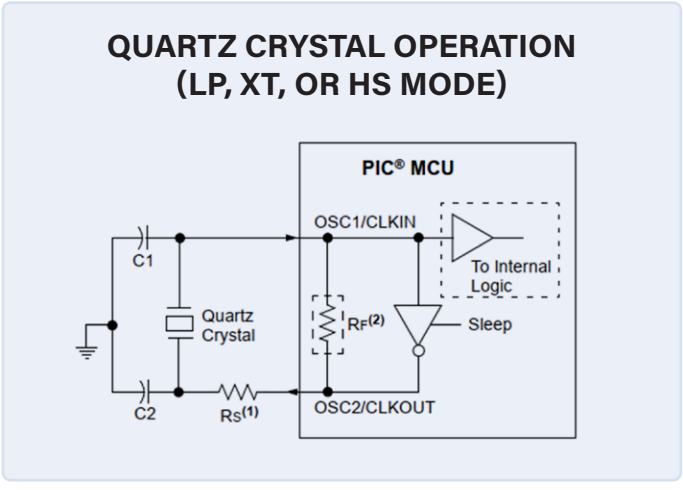
te zorgen dat wanneer je iets naar die registers schrijft, de niet-geïmplementeerde bit nul worden gemaakt. In C/C++ kan dat als volgt:

```
myValue &= 0b00011111; // clear 3 most-significant bits
                       // = (unimplemented bits)
STATUS = myValue;      // write 'myValue' into
                       // STATUS register
```

Opmerking: het is wel aan te raden om duidelijk commentaar bij deze code te zetten, om te voorkomen dat in de toekomst iemand je veiligheidsmaatregel "wegoptimaliseert".
Een streepje boven de naam van een bit geeft aan dat de waarde geïnverteerd is. Dat geldt bijvoorbeeld voor het bit TO ("Time Out"). Als een time-out optreedt, wordt de waarde dus niet "1", zoals je zou verwachten, maar "0". Voor de programmeur is het vaak onduidelijk waarom zulke bits geïnverteerd zijn; je moet ermee leren leven en de bits in je code correct behandelen.
Soms is een groep bits nodig om iets te configureren, zoals de prescaler voor een klokbron, of de baudrate voor een seriële interface. Zulke bitgroepen worden aangeduid als `BIT_GROUP<X:Y>`, waarbij `X` het meest significante bit van de groep is en `Y` het minst significante bit. En opgepast: soms zijn de bits die bij een groep horen geen opeenvolgende bits in het register, soms zijn ze zelfs verdeeld over meer dan één register!

## Tijd voor de klok

De klok-periferie is het belangrijkste deel van de microcontroller. Hier wordt bepaald wat er wordt gebruikt als kloksignaal voor de processor en voor de periferie in de chip. Het is nuttig om dit grondig te bestuderen. Je vindt dit op pagina 110, Normaal gesproken wordt deze periferie eenmalig geconfigureerd bij het starten van het programma en daarna niet meer veranderd. Als de configuratie wordt veranderd, kan dat gevolgen hebben voor alle periferie, van de bemonsteringsfrequentie van de ADC tot de baudrate van de seriële UART en de CAN-interface.
In dit voorbeeld hebben we enkele externe bronnen (kristaloscillatoren of keramische resonatoren) en een paar interne bronnen. Die laatste zijn misschien niet nauwkeurig genoeg (vooral bij temperatuurveranderingen) als betrouwbare klok voor sommige interfaces, zoals de UART. Waarschijnlijk wordt dat in de datasheet vermeld. Anders moet je de nauwkeurigheid checken in het hoofdstuk *Electrical Characteristics*.
In het schema zie je dat een groep van drie bits, `COSC<2:0>`, in een register dat te maken heeft met de oscillator, wordt gebruikt om te kiezen tussen de beschikbare klokbronnen met behulp van een multiplexer. Om één of andere reden wordt alleen de groep bits genoemd in zulke schema's en niet het register waar ze bij horen. De beste manier om te vinden in welk register die groep zit, is ernaar te zoeken in het PDF-bestand. Ze kunnen zomaar verstopt zitten in een register van ongerelateerde periferie! Multiplexers worden vaak gebruikt in blokschema's waar tussen signalen kan worden geschakeld.
De uitvoer van dit blok levert de *System Clock*, waarschijnlijk voor de processor, en de *Peripheral Clock* voor de periferie. Naar die begrippen kan worden verwezen in andere delen van de datasheet, vooral bij de bespreking van de *Low Power*- of *Sleep*-modi. Uit de groen gemarkeerde gedeelten kunnen we al concluderen dat er een *Idle*-modus is die alleen de klok voor de periferie uitschakelt, en een *Sleep*-modus die de klok voor zowel de periferie als voor de processor uitschakelt.
Als een kristal of keramische resonator moet worden gebruikt, worden er waarschijnlijk aanwijzingen gegeven over hoe die moeten worden aangesloten en over de beste print-layout. Dat is ook hier het geval (pagina 112): er zijn twee condensatoren nodig en eventueel een serieweerstand. Het feit dat dit blokschema vergezeld gaat van vier application notes over het ontwerpen van oscillatoren, geeft al aan dat dit nog niet zo eenvoudig is.




*Figuur 3. Aanbevolen schema van een kristaloscillator (bron: Microchip Technology).*

## Voorbij de registers en blokschema's

We hebben nu de basis besproken van hoe registers worden beschreven en hoe we het blokschema van de klokperiferie moeten interpreteren. We hebben zelfs een blik geworpen op één van de oscillatoren en het voorbeeldschema voor het gebruik van een kristal. In het volgende deel van deze serie gaan we kijken naar een ander belangrijk periferie-element: het resetblok. En we gaan ook op zoek naar alles wat het datasheet ons niet vertelt.

Als je belangstelling hebt voor informatie die je instap in de wereld van de microcontrollers gemakkelijker kan maken, werp dan eens een blik op onze inleidende artikelen [3] en boeken.

*vertaling: Evelien Snel — 230101-03*

### Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een email naar de auteur via stuart.cording@elektor.com.

### Gerelateerde producten

- **T. Hanna, *Microcontroller Basics with PIC*, Elektor 2020 (SKU 19188)**
  www.elektor.nl/19188
- **A. Pratt, *Programming the Finite State Machine*, Elektor 2020 (SKU 19327)**
  www.elektor.nl/19327

### WEBLINKS


[1] Stuart Cording, "Microcontroller-documentation verklaard (deel 1)": www.elektormagazine.nl/magazine/elektor-295/61525
[2] 8-bit PIC16F18877-microcontroller: https://microchip.com/wwwproducts/en/PIC16F18877
[3] Tam Hanna, "PIC's programmeren: van de grond af": https://www.elektormagazine.nl/magazine/elektor-157/59016