Experimenteren met de PIC

Dit artikel is een vervolg op de vorige Technische Notities van Marc. Destijds is de architectuur van de PIC16F84 doorgelicht. Nu is het moment aangebroken de ontwikkelomgeving MPLAB te installeren en daarna een programma te schrijven dat een LED laat knipperen.

7. Datasheets op internet

Ga naar *www.microchip.com* en download de datasheet van de PIC16F84. Diverse mensen hebben mij gevraagd waar ze de sheet van de PIC16F84 kunnen vinden. Kies linksboven 'PIC Micro MCU's'. Je komt dan in het gebied van de site waar alle PIC datasheets en application notes staan. Kies uit de lijst PIC16X8X. Scroll nu naar beneden, daar staat 'PIC16F8X Family Datasheets'. Klik hierop. Scroll naar beneden en selecteer PIC16F84 of PIC16F84A. Tenslotte laat de site een aantal servers zien waarmee je de sheet kunt downloaden. Kies een van de vier FTP-sites. De file 35007A.PDF wordt vervolgens gedownload. Zoals eerder vermeld: een papieren versie van deze datasheet is vereist. Microchip stelt zijn ontwikkelomgeving MPLAB gratis ter beschikking. Maak je geen zorgen om de kosten, of om een beperking in de juiste werking van de software. Geld verdienen doen ze echt wel: als je enkele PIC's koopt of hardware om ze te programmeren (PICSTART), stromen de dollars toch wel binnen. Ik raad aan een CD-ROM van Microchip aan te vragen bij AvNet in Breda. De MPLAB software kan direct van de CD-ROM geïnstalleerd worden. Alle CD-ROM's van Microchip die sinds 1998 zijn gemaakt zijn eigenlijk images van de Microchip website.

8. Installeren van MPLAB

De PIC16F84 is een oudere telg uit de Microchip-familie. Het maakt daarom niet uit welke versie van MPLAB je gaat installeren. Alles vanaf versie 4.XX en hoger is goed. De benodigde PC-configuratie is ook niet echt schokkend: een PC met Pentium 100, 16MB RAM, Windows9X en CD-ROM. Er is 15MB schijfruimte nodig. Zorg ervoor dat een browser is geïnstalleerd, bijvoorbeeld Netscape of Internet Explorer. Doe de CD-ROM in de PC en klik op INDEX.HTM. De browser wordt nu gestart en laat de startpagina zien, net als op internet. Je kunt nu alles downloaden, waaronder de datasheet. Voor MPLAB klik je op 'Development Tools', ergens links middenin. De browser springt nu naar de juiste webpagina. Hieronder staat 'MPLAB-IDE'. Met IDE wordt 'Integrated Development Enviroment' bedoeld oftewel 'geïntegreerde ontwikkelomgeving'. Klik hierop. Er wordt een nieuwe pagina geopend waarin verschillende versies worden weergegeven. Kies de meest recente versie, bijvoorbeeld MPLAB Version 4.99.07. De browser zal naar een volgend scherm springen waarin een download kan worden gestart vanaf CD-ROM naar PC. Meestal wordt dit aangegeven met 'MPLAB download disk 1 of 7'. De browser zal vanzelf melden dat een file opgeslagen moet worden op harddisk. Dump deze file voorlopig even op het bureaublad. De naam van de file kan varieren met de versie, maar het is in elk geval een *.EXE file. Als de file eenmaal op de harddisk staat, maak dan met de explorer/verkenner een directory aan: C:\MPLAB. Maak hieronder de volgende subdirectory aan: C:\MPLAB\SOURCE. Sleep de *.EXE file in de SOURCE-directory. Klik op deze file en er verschijnen tientallen files die de MPLAB software bevatten, inclusief een SETUP.EXE file. Klik op SETUP.EXE en MPLAB start zijn installatiescherm. MPLAB zal vragen allerlei dingen in te stellen.
Laat alles 'default' en kies overal 'ok'. Ik heb nooit meegemaakt dat het dan niet werkt: MPLAB is best een goed gedefinieerde en stabiele ontwikkelomgeving in het toch wankele Windows. MPLAB komt terecht onder WINDOWS\PROGRAM FILES. Van deze directory is niks nodig: we gaan werken in onze eigen werkdirectory C:\MPLAB. Herstart Windows en ga naar het startmenu. Er is een programmakeuze bijgekomen: Microchip MPLAB. Kies dit en MPLAB zal vervolgens starten. Wellicht zeurt MPLAB dat hij een emulator mist. Klik in dat geval op 'Editor Only'.

9. Aanmaken van een nieuw project onder MPLAB

Nu MPLAB is gestart is het volgende zichtbaar: een leeg grijs scherm met enkele buttons en de bekende keuzebalk. Allereerst maak je met explorer/verkenner onder MPLAB een project-directory aan: C\MPLAB\LEDPIC. In de LEDPIC-directory gaan we een programma schrijven om een LED te laten knipperen. De strategie van MPLAB is dat je moet werken met projects. Elk project heeft een set van files die bij dat project horen. Elke keer als iets nieuws wordt ontworpen, doe je dat in een nieuwe directory. Kies geen lange bestandsnamen: MPLAB draait in 16 bits mode en accepteert deze helaas niet. Sluit explorer/verkenner nu af.
Kies onder MPLAB in de keuzebalk voor 'project'. Selecteer vervolgens 'New Project'. MPLAB komt nu met een menu waarin de projectnaam moet worden opgegeven, *.PJT. Type LEDPIC.PJT en klik op ok. MPLAB opent nu een box waarin diverse dingen moeten worden gedefinieerd. Dit gedeelte is erg belangrijk. Het resultaat van een volledig uitgewerkt project zal een *.HEX-file zijn die je in de PIC kunt programmeren. Dit is het uitgangspunt in dit menu: MPLAB moet weten welke files hiervoor van belang zijn, welke files naar de uiteindelijke *.HEX leiden. In ons geval is dat dus LEDPIC.HEX. Deze zie je al staan, linksboven bij 'Target Filename'. Om het project verder te definiëren moet het volgende gebeuren:

a Processor kiezen
b Node kiezen: welke file de bron is om tot file *.HEX te komen
c Assembler van MPLAB (MPASM) instellen
d Nieuwe *.ASM openen

a. Processor kiezen
In deze box staat 'Development Mode' met daarachter 'Change'. Klik hierop. Er verschijnt een enorme lijst met PIC¸s. Wanneer je dacht dat Microchip alleen de PIC16F84 maakte, dan heb je het mis. Onder zendamateurs is dit gewoon een populair type. Kies altijd 'Editor Only' en kies uit de lijst de PIC16F84 of de PIC16F84A, afhankelijk van welke je hebt. Eigenlijk maakt het niets uit, de HEX-file zal er hetzelfde uitzien. Klik hierna op 'ok'. Je hebt nu vastgelegd dat je met de PIC16F84x wilt ontwikkelen, met alleen de editor van de MPLAB-ontwikkelomgeving. Waarom ¸editor only¸? Je kunt ook een emulator kopen of een zogenaamde ICD ontwikkel-kit voor bepaalde PIC's, maar dan zul je diep in de buidel moeten tasten. Daarom voorlopig 'editor only'.

b. Node kiezen
In de box staat 'Add Node'. Een node wordt hier letterlijk bedoeld als verbinding, welke file de bron is om straks om te zetten in een *.HEX-file. Klik op 'Add Node'. Zorg er altijd voor dat MPLAB de juiste directory aanwijst: C:\MPLAB\LEDPIC. Type nu LEDPIC.ASM rechtsboven als filename en klik op ¸ok¸. Nu is gedefinieerd dat LEDPIC.ASM de bron is waarmee MPLAB LEDPIC.HEX kan maken waarmee later de PIC geprogrammeerd kan worden. *.ASM staat voor een 'Assembly File'.Met de PIC schrijven we dus in Assembly. MPLAB zal deze *.ASM file omzetten in een *.HEX file m.b.v. de assembler.

c. Assembler van MPLAB (MPASM) instellen
Bij gebruik van de assembler MPASM moeten enkele dingen worden ingesteld om te zorgen voor een juiste uitvoer. Klik in het gedeelte ¸Project Files¸ op LEDPIC[.HEX]. De clickboxjes 'Build Node' en

'Node Properties' lichten nu op aan de rechterkant. Kies 'Node Properties'. Nu wordt een nieuwe box zichtbaar waarin de instellingen van de assembler kunnen worden veranderd. Laat alles zo staan, selecteer alleen 'Warn + Error' en klik op 'ok'. Bestudeer deze instellingen goed. Je ziet o.a. dat MPASM gevoelig is voor verschillen in kleine en grote letters. Je moet minstens eenmaal in dit menu komen anders voert MPLAB geen assemblage van LEDPIC.ASM uit. Klik nogmaals 'ok' om naar het hoofdscherm van MPLAB terug te gaan. De assembler MPASM is nu klaar voor gebruik.

d. Nieuwe *.ASM openen

Als laatste moet LEDPIC.ASM file worden geopend omdat deze simpelweg nog niet bestaat. Ga naar 'file' en kies 'new'. Klik daarna op 'ok'. MPLAB opent nu een editbox met de naam 'UNTITLED1'. Kies nu 'file' en 'save as'. Je kunt de file nu een naam geven. Geef hem de naam 'LEDPIC.ASM' en klik op 'ok'. Alles is nu correct ingesteld voor dit project.

10. Assembler-taal

Houd de instructie-set van de PIC16F84 gereed. Met de instructies die hierin staan kan het functionele gedeelte van het programma worden samengesteld. Verder kan de assembler van Microchip ook assembly directives verwerken. Deze assembly directives zijn speciale commando's om de assembler te sturen. Een van deze commando's is 'LIST'. Dit commando geeft aan welke processor we gekozen hebben. Het commando 'END' geeft aan dat de assembler moet stoppen met assembleren. Een andere belangrijke assembly directive is het commando '#INCLUDE' waarmee vreemde files aan een programma kunnen worden gehangen. De assembler pikt deze dan gewoon mee. We komen er enkele tegen bij het schrijven van kleine programma's. Probeer eventueel het boekje 'MPASM USERS GUIDE' te pakken te krijgen als je meer wilt weten van deze assembler directives. Voor wat we hier gaan doen is dit boekje niet nodig.

11. De hardware

Bouw de schakeling uit figuur 11.1, bijvoorbeeld op experimenteerboard, bij voorkeur compleet met de spanningsregelaar. Dan kan de PIC niet kapot gaan als een te hoge voedingsspanning wordt aangebracht. De condensatoren van 100 nF zijn er ook niet voor niets: plaats C7 en C8 dicht op de pootjes van de spanningsregelaar, C4 dicht bij pen 14 van de PIC. Maak ook goede verbindingen naar massa. De PIC heeft drie basiselementen nodig om goed te functioneren:

1. Een kristal of resonator met condensator van ongeveer 22 pF
2. Ordentelijk gefilterde 5 V dc voedingsspanning en massa
3 Een reset-circuit

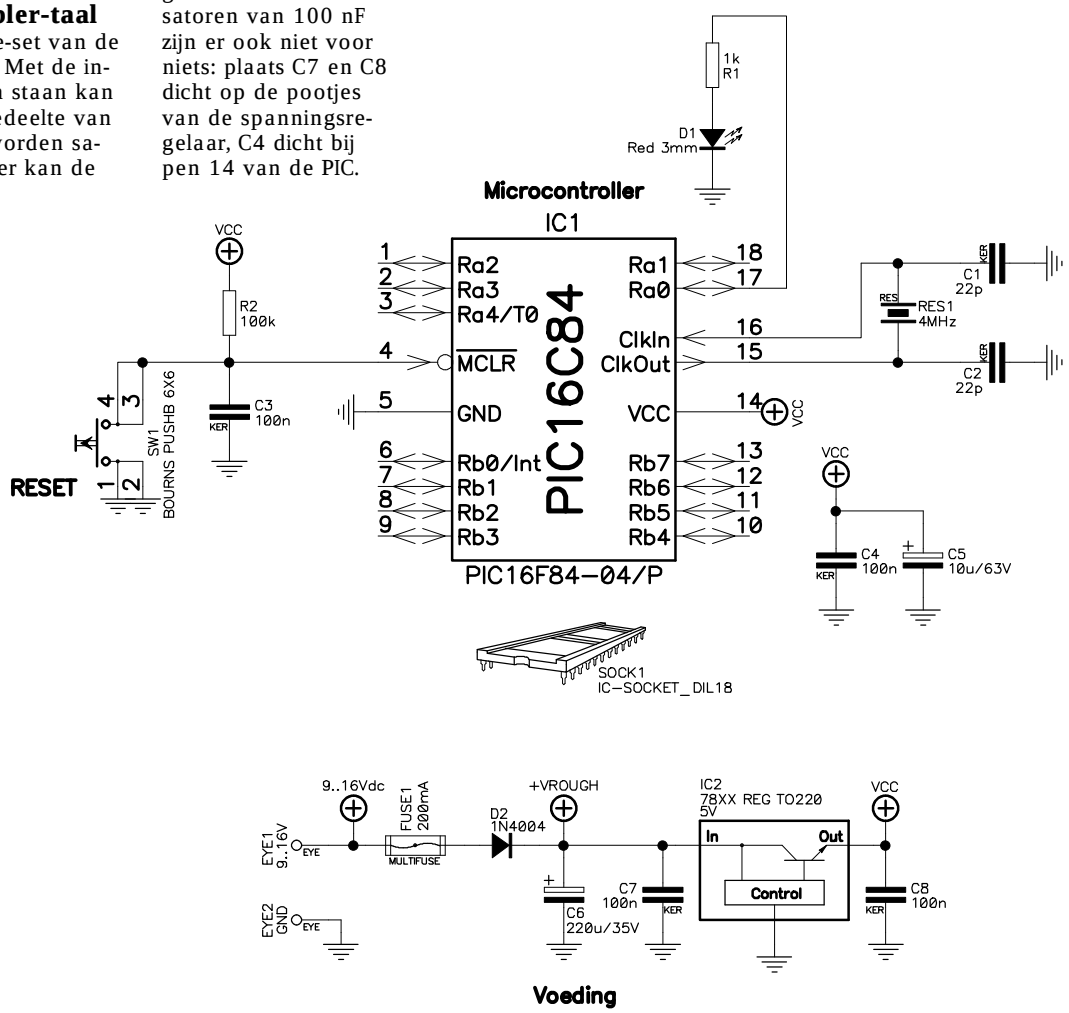
RS1 is een kristal of resonator van 4 MHz. Dit maakt het ons straks gemakkelijk: de Microchip executeert een instructie in 4 cycles. ¨ x 4 MHz = 1 MHz. Voor een instructie is dus 1 microseconde nodig. Anders gezegd: per seconde 1 miljoen instructies. Het reset-circuit is opgebouwd rond R2 en C3. Met de drukschakelaar SW1 kan een externe reset worden geven. LED D1 wordt via een weerstand aangesloten op pin RA0. Het is de bedoeling de LED te laten knipperen met een forse tijdvertraging die wordt geschreven in assembly-code. Figuur 11.2 toont de materialenlijst voor dit project. Brigatti in de HobbemaStraat te Eindhoven kan indien gewenst dit pakketje leveren, exclusief printje.

12. Programmeren van de PIC

De PIC moet worden voorzien van een programma. De volgende mogelijkheden bestaan:

1. Koop een PICSTART. Dit is een programmeer-apparaat van Microchip waarmee alle PIC's kunnen worden geprogrammeerd, Dit is de mooiste oplossing, omdat hij meteen onder MPLAB kan worden aangestuurd.
2. Kijk of een eventueel beschikbaar programmeerapparaat zoals een GALEP of een HILO de PIC16F84 kan programmeren. Vaak is dit het geval.
3. Zoek op internet.

Er zijn veel 'video-ontvangerverbeteraars' die een PIC16F84 gebruiken om signalen te emuleren die naar kaartjes gaan die in satellietontvangers worden gestopt. Meestal hangen deze aan de seriele poort en kan met een DOS programma de *.HEX file (dus ook de LedPic.HEX-file) naar de PIC worden gestuurd. Let op: vaak wordt de RS232-spanning gebruikt om de



Figuur 11.2

FLASH van de PIC te programmeren. In veel gevallen is deze RS232-spanning te laag waardoor de PIC niet goed wordt geprogrammeerd. Voorkom frustraties! Als u bij zoekmachine Yahoo zoekt op 'pic programmer' is het aantal hits meer dan 8000. Enkele interessante sites zijn:
http://www.covingtoninnovations.com/noppp/
http://www.nottingham.ac.uk/pic84/pic84.html
http://www.codepuppies.com/~ben/sens/pic/sx/
http://www.jdm.homepage.dk/newpic.htm

Velen hebben ook de datasheet in hun listing staan, erg handig. Toepassing van deze hints is uiteraard op eigen risico.

13. Knipperen van de LED, kop en staart

De hardware is gereed, een programmer aanwezig, MPLAB heeft een lege editbox genaamd LEDPIC.ASM. Wat nu? Allereerst gaan we het programma 'kop en staart' geven. Type het volgende in, zie listing 13.1.
Dit is dus 'kop en staart'. Allereerst geven we met de LIST assembly directive aan dat sprake is van een PIC16F84, dat we 75 tekstkarakter-kolommen hebben en dat LEDPIC.HEX file het formaat INHX8M zal hebben. Het __CONFIG-commando is ook een assembly-directive en wel een speciale. Elke PIC heeft een 'configuration word' oftewel een configuratie-programmeerlocatie waarin staat hoe de PIC opstart. In dit geval heb ik gezorgd dat met de waarde 005h de PIC opstart met de XT-oscillator en de watchdog ingeschakeld. De watchdog is de waakhond die we de vorige keer al hebben

```
LIST      P=16F84,C=75,F=INHX8M  ;Definieer type processor
                                 ;
__CONFIG  005h                   ;Definieer hoe de PIC straks
                                 ;geprogrammeerd moet worden,
                                 ;het ¸programming
                                 ;configuration word¸: Code
                                 ;Protect = ON, Watchdog = ON,
                                 ;Powerup Timer = ON,
                                 ;Oscillator = XT
                                 ;
                                 ;(HIER KOMT HET PROGRAMMA)
                                 ;
          END                    ; Einde van het programma
```

Listing 13.1

```
LIST         P=16F84,C=75,F=INHX8M  ;Definieer type processor
__CONFIG     005h                   ;'programming configuration
                                    ;word' voor PIC16F84
                                    ;**********************
             ORG       0000h        ;Origin op 0000h (0000h =
                                    ;hexadecimaal 0000' enz.)
             goto      knipperstart ;ga naar knipperstart
                                    ;**********************
             ORG       0100h        ;Origin op 0100h, assembler
                                    ;gaat vanaf hier verder
knipperstart clrwdt                 ;Waakhond een botje geven
             goto      knipperstart ;blijf in de lus hangen, ga
                                    naar knipperstart
                                    ;**********************
             END                    ;Einde van het programma
```

Listing 13.2

```
LIST         P=16F84,C=75,F=INHX8M  ;Definieer type processor
__CONFIG     005h                   ;'programming configuration
                                    ;word' voor PIC16F84
                                    ;**********************
status       EQU       003h         ;status-register in RAM op
                                    ;positie 003h
ze           EQU       2h           ;zero-vlag in het
                                    ;status-register, is bit 2
rp0          EQU       5h           ;rp0-vlag in het
                                    ;status-register, is bit 5
optreg       EQU       081h         ;option-register in RAM op
                                    ;positie 081h, genaamd
                                    ;optreg!
trisa        EQU       085h         ;poort A I/O
                                    ;definitie-register in RAM
                                    ;op 085h
porta        EQU       005h         ;poort A in RAM op positie
                                    ;005h
                                    ;**********************
             ORG       0000h        ;Origin op 0000h (0000h is
                                    ;hexadecimaal 0000' enz.)
             goto      knipperstart ;ga naar knipperstart
                                    ;**********************
             ORG       0100h        ;Origin op 0100h, assembler
                                    ;gaat vanaf hier verder
knipperstart clrwdt                 ;Waakhond een botje geven
             goto      knipperstart ;blijf in de lus hangen,
                                    ;ga naar knipperstart
                                    ;**********************
             END                    ;Einde van het programma
```

Listing 14.1

gezien. Assembly-directive END is wellicht de belangrijkste: MPLAB moet immers weten wanneer de assembler moet stoppen. Alles achter een ';' wordt niet meegenomen door MPASM. Dit is alleen voor commentaar.
Druk nu op het icoon met de zwarte diskette om LEDPIC.ASM op de harde schijf op te slaan. Druk ook op de groene diskette: nu wordt LEDPIC.PJT opgeslagen, de project definitie-file die in het begin is gekozen. Klik nu op de groene zandloper (uiterst rechts) en MPASM zal de file assembleren naar LEDPIC.HEX.
Er verschijnt nu een nieuw window waarin staat 'Build Results'. Mochten er fouten zijn opgetreden dan staan ze in dit scherm vermeld en weet je waarom de assembler gestopt is. Probeer dit: type iets onzinnigs in LEDPIC.ASM, bijvoorbeeld voor de ';' waarachter '(HIER KOMT ONS PROGRAMMA)' staat en druk op de zandloper. Je ziet dat de assembler met een foutmelding komt. Dubbelklik op die foutmelding in de box van 'Build Results' en je ziet dat de cursor naar de positie springt waar deze fout gemaakt is.
Na het assembleren is LEDPIC.HEX beschikbaar. Probeer deze file te openen onder 'file'. Kies 'all files' en dubbelklik op LEDPIC.HEX. Hierin staat het volgende:
:02400E00ED00C3
:00000001FF

Wat je hier ziet is de 'setting' voor de programmeer-fuses van de PIC. Er is eigenlijk niet eens een programma maar MPLAB accepteert de assembly directives in elk geval wel. 'Kop en staart' is klaar, we gaan verder met het programma. Type in LEDPIC.ASM nu verder, zie listing 13.2.

ComponentName	*Pattern*	*Value*	*RefDes*	*CompMake*	*Description*
1N4004	DO41		D2	MultiComp	1A General Purpose Rectifier Diode
78XX REG TO220	TO220-3	5V	IC2	Motorola	Regulator 1.5A Positive fixed voltage
BOURNS PUSHB 6X6	BOURNS 6X6MM		SW1	Bourns	Bourns pushbutton PCB mount
CKER 63V 1E	CKER_1E	22p	C1	AVX	Multilayer Ceramic Capacitor X7R 63V
CKER 63V 1E	CKER_1E	22p	C2	AVX	Multilayer Ceramic Capacitor X7R 63V
CKER 100N/63V 1E	CKER_1E	100n	C3	AVX	Multilayer Ceramic Capacitor X7R 63V
CKER 100N/63V 1E	CKER_1E	100n	C4	AVX	Multilayer Ceramic Capacitor X7R 63V
CKER 100N/63V 1E	CKER_1E	100n	C7	AVX	Multilayer Ceramic Capacitor X7R 63V
CKER 100N/63V 1E	CKER_1E	100n	C8	AVX	Multilayer Ceramic Capacitor X7R 63V
ELCO 10U/63V	ELCO_037_11	10u/63V	C5	Philips	037-series 85-degree Radial Capacitor
ELCO 220U/35VLI	ELCO_136_10.5MM	220u/35V	C6	Philips	136-series Low ESR 105C Radial Capacitor
IC-SOCKET_DIL18	IC-SOCKET_DIL18		SOCK1	Augat	IC Insertion Socket machined contacts
LED3MMB_HLMP_RED	LED_3MM_BENDED	Red 3mm	D1	HP	LED low current 3mm red with legs bended
MULTIFUSE 200MA	MULTIFUSE 2E #1	200mA	FUSE1	Bourns	Multi Fuse MF-R series 60V
PIC16F84-04/P	DIL18		IC1		
RESISTOR SFR25	R SFR25 5%	1k	R1	Philips	Resistor Axial 250mW 5%
RESISTOR SFR25	R SFR25 5%	100k	R2	Philips	Resistor Axial 250mW 5%
RESONATOR 2PIN	RESONATOR_2PIN	4MHz	RES1	Murata	Ceramic Resonator
TESTEYE	CERAMICEYE1	9..16V	EYE1	MicroVar	Ceramic Test Eye
TESTEYE	CERAMICEYE1	GND	EYE2	MicroVar	Ceramic Test Eye

Tabel 1 Benodigde materialen

De directive ORG is belangrijk. Deze bepaalt vanaf welke plek de assembler instructies moet assembleren. De eerste ORG die de assembler tegenkomt staat op 0000h. Dus de 'goto'-instructie staat op deze 0000h. Deze 'goto' verwijst meteen naar knipperstart. Het springen naar 0100h is noodzakelijk omdat de vector van de interrupt op 0004h staat. Daar mogen we niet in de buurt komen (zie de datasheet van de PIC!).
Bij knipperstart aangekomen krijgt de waakhond iets te eten. Direct daarna komt weer een goto-instructie die wederom de waakhond iets te eten geeft, enz. Er is een lus opgezet.

14. Knipperen van de LED, instellen van belangrijke registers

Er moeten nog enkele dingen gebeuren alvorens het programma verder kan worden uitgewerkt:

a. De waakhond instellen
b. Zorgen dat pin 17 van de PIC bereikbaar is

Hiervoor worden allereerst enkele belangrijke registers gedefinieerd. Breid het programma uit zodat het volgende in LEDPIC.ASM komt te staan. Vergeet niet af en toe op te slaan. Zie listing 14.1.

Hier vindt in feite vertaling van datasheet naar assembly plaats. Als je in de datasheet het gedeelte van de 'register file map' opzoekt, zie je dat EQU een assembly-directive is die zorgt voor een verwijzing naar een RAM-positie in de 'register file map'. (Geheugensteuntje: bij de PIC is een 'file' een geheugenlocatie in RAM) Probeer deze regels te vergelijken en verifieer of de RAM-locaties in PICLED.ASM overeenkomen met die in de datasheet.
De 'ze' en de 'rp0' zijn uitzonderingen: dit zijn verwijzingen naar een bit in een 'register file locatie'. De 'ze' is de zero-flag en de 'rp0' is de 'register page select 0' van het status-register. De 'ze' is het 2e bit en de 'rp0' is het 5e bit in het status-register. (Kijk dit na in de datasheet) We moeten deze registers allemaal definieren omdat MPASM anders niet begrijpt wat we willen: we kunnen namen niet zomaar uit de lucht laten vallen. Vervolgens worden enkele belangrijke registers ingesteld: 'optreg' en 'trisa'. Zie hiervoor listing 14.2 en type deze aanvullingen in de PICLED.ASM box. De eerste instructie 'bsf' staat voor 'bit set file' en zal het gekozen bit in de 'file' hoog maken.
We maken 'rp0' in het status-register hoog. Reden? We moeten naar hoger gelegen RAM (RAM1) om bij 'optreg' en 'trisa' te kunnen komen. Dit zie je ook meteen: we laden binair 00001111 in het W-register en bewaren hem in 'optreg' d.m.v. de instructie 'movwf'. Als je weet dat met 'file' een RAM-locatie wordt bedoeld is het duidelijk: move W to file. Hiermee is 'optreg' geladen met binair 00001111.
Waarom met deze waarde? Bekijk de datasheet: de laatste 3 bits zetten de prescaler op maximale delay. Bit 3 is de toewijzing van de prescaler naar de waakhond. Dat is alles. Het resultaat is dat de watchdog ingesteld is op maximale tijd, en deze is ongeveer 2,3 seconden (ga dit na). Vervolgens gaan we het register 'trisa' laden om pin 17 van de PIC te kunnen gebruiken als uitgang voor de LED. De inhoud van de registers 'tris' bepaalt of de desbetreffende I/O-pin ingang of uitgang is. Als in zo'n 'tris'-register een bit 0 wordt gemaakt, komt dat overeen met een uitgang. Een bit 1 staat voor een ingang. Alleen bit 0 van het 'trisa'-register is uitgang: We laden 11111110 in W en bewaren hem in het 'trisa'-register. De instructie daarna is een 'bcf' (bit clear file) die zorgt dat weer in lager gelegen RAM (RAM0) kan worden gewerkt.
Wat je ook kan zien is dat de lus in stand gehouden wordt. Waarom ook niet? Alles wordt netjes opnieuw gedefinieerd, telkens als de lus wordt doorlopen. In de praktijk is deze redenering heel belangrijk. Teveel programmeurs denken: „och, als het eenmaal is ingesteld zal het wel loslopen".
De praktijk is helaas vaak anders: 'hangende' apparatuur moet dan even spanningsloos worden gemaakt voordat de werking weer correct is.

```
LIST      P=16F84,C=75,F=INHX8M ;Definieer type processor
__CONFIG  005h                  ;'programming configuration
                                ;word' voor PIC16F84
                                ;**********************
status    EQU     003h          ;status-register in RAM op
                                ;positie 003h
ze        EQU     2h            ;zero-vlag in het
                                ;status-register, is bit 2
rp0       EQU     5h            ;rp0-vlag in het
                                ;status-register, is bit 5
optreg    EQU     081h          ;option-register in RAM op
                                ;positie 081h, genaamd
                                ;optreg!
trisa     EQU     085h          ;poort A I/O
                                ;definitie-register in RAM
                                ;op 085h
porta     EQU     005h          ;poort A in RAM op positie
                                ;005h
                                ;**********************
          ORG     0000h         ;Origin op 0000h (0000h is
                                ;hexadecimaal 0000 enz.)
          goto    knipperstart  ;ga naar knipperstart
                                ;**********************
          ORG     0100h         ;Origin op 0100h, assembler
                                ;gaat vanaf hier verder
knipperstart clrwdt             ;Waakhond een botje geven
          bsf     status,rp0    ;Naar RAM bank 1, noemen we
                                ;ook wel RAM1
          movlw.  B'00001111'   ;Waakhond krijgt de
                                ;prescaler met max.
                                ;delaytijd
          movwf   optreg        ;bewaar in file ‚optreg‚
          movlw   B'11111110'   ;laatste bit is een 0,
                                ;wordt dus een uitgang
          movwf   trisa         ;want we bewaren hem in
                                ;‚trisa‚,is pin17, de
                                ;RA0-pin
          bcf     status,rp0    ;terug naar RAM0
          goto    knipperstart  ;blijf in de lus hangen, ga
                                ;naar knipperstart
                                ;**********************
          END                   ;Einde van het programma
```

Listing 14.2

```
LIST      P=16F84,C=75,F=INHX8M ;Definieer type processor
__CONFIG  005h                  ;'programming configuration
                                ;word' voor PIC16F84
                                ;**********************
status    EQU     003h          ;status-register in RAM op
                                ;positie 003h
ze        EQU     2h            ;zero-vlag in het
                                ;status-register, is bit 2
rp0       EQU     5h            ;rp0-vlag in het
                                ;status-register, is bit 5
optreg    EQU     081h          ;option-register in RAM op
                                ;positie 081h, genaamd
                                ;optreg!
trisa     EQU     085h          ;poort A I/O
                                ;definitie-register in RAM
                                ;op 085h
porta     EQU     005h          ;poort A in RAM op positie
                                ;005h
                                ;**********************
          ORG     0000h         ;Origin op 0000h (0000h is
                                ;hexadecimaal 0000' enz.)
          goto    knipperstart  ;ga naar knipperstart
                                ;**********************
          ORG     0100h         ;Origin op 0100h, assembler
                                ;gaat vanaf hier verder
knipperstart clrwdt             ;Waakhond een botje geven
          bsf     status,rp0    ;Naar RAM bank 1, noemen we
                                ;ook wel RAM1
          movlw   B'00001111'   ;Waakhond krijgt de
                                ;prescaler met max.
                                ;delaytijd
          movwf   optreg        ;bewaar in file ‚optreg‚
          movlw   B'11111110'   ;laatste bit is een 0,
                                ;wordt dus een uitgang
          movwf   trisa         ;want we bewaren hem in
                                ;trisa, is pin17, de
                                ;RA0-pin!
          bcf     status,rp0    ;terug naar RAM0
          bsf     porta,0       ;LED aan
          bcf     porta,0       ;LED uit
          goto    knipperstart  ;blijf in de lus hangen, ga
                                ;naar knipperstart
                                ;**********************
          END                   ;Einde van het programma
```

Listing 15.1

15. Knipperen van de LED: LED aan en LED uit

Nu volgt het echte werk: het knipperen van de LED zelf. Dit zijn eigenlijk maar twee instructies. Voeg ze toe in LEDPIC.ASM, zie listing 15.1. De instructies 'bsf' en 'bcf' zagen we al eerder. Bit 0 van 'porta' is eerst hoog en daarna weer

laag. Pin 0 van 'porta' is pin 17 van de PIC. MPLAB kan dit assembleren tot een werkend programma.
Programmeer de PIC en plaats hem in de schakeling. Wat je dan ziet? Ach, de LED is gewoon continu aan. Maar dat kan toch niet? Inderdaad. Meet met een oscilloscoop op pin 17. Je zult zien dat de PIC de LED heel snel in- en uitschakelt. Zo snel dat je dat niet kunt zien: meer dan 100000 keer per seconde!
Hoe lossen we dit op? Er is een stukje programma nodig waarmee een vertraging gemaakt wordt, zodat we het knipperen van de LED met onze eigen ogen kunnen waarnemen. Dit stukje programma is bij voorkeur herbruikbaar en wordt een subroutine of gewoon routine genoemd. De PIC heeft een instructie om zo'n routine aan te roepen, de instructie 'call'. Kijk dit na in de datasheet. Tevens heeft de PIC een instructie om van de routine terug te keren, de instructie 'return'. Hoe werkt dit nu? Als de PIC een 'call' tegenkomt, neemt hij het huidige adres, de PROGRAM COUNTER dus, telt daar 1 bij op en bewaart dit op de STACK. De STACK is in de vorige Technische Noties besproken. Lees dit e.v.t. nog een keer na. De PIC doet nu een 'call' naar het label dat achter de 'call'-instructie staat: hij springt naar dat label. De PIC vervolgt zijn programma vanaf dat label. Een poosje later is het de bedoeling dat de instructie 'return' wordt gebruikt. De STACK wordt aangesproken en het originele adres verhoogd met 1, wordt uit de STACK gehaald en in de PROGRAM COUNTER gezet. Het programma vervolgt zijn weg net achter de plek waar eerder de instructie 'call' plaatsvond. Het is tijd voor een volgende listing. Voeg deze instructies ook toe aan PICLED.ASM, zie listing 15.2.
Direct nadat de LED is ingeschakeld met 'bsf porta,0' vindt een 'call' naar het label 'wachtmaar' plaats. De PIC springt hier dus heen. Ter illustratie heb ik een 'nop'-instructie gebruikt. Deze instructie doet niets. Meteen achter deze 'nop' volgt 'return' waardoor bij de volgende instructie de PIC zijn programma vervolgt bij 'bcf porta,0'.

16. Knipperen van de LED met routine 'wachtmaar'

De routine 'wachtmaar' moet nu worden ingevuld. De PIC heeft hier een handige instructie voor: decfsz. Deze instructie staat voor 'decrement file and skip if zero'. Dat 'skip if zero' wil niks anders zeggen dan dat de PIC over de eerstvolgende instructie springt als het register dat bij de instructie hoort gelijk aan 0 is. Bekijk het volgende stukje assembly, zie figuur 16.1.
Register 'wacht1' wordt met de instructie 'clrf' 0 gemaakt. Een poosje later komt de PIC de instructie 'dezfsz' tegen. Register 'wacht1' wordt verlaagd met 1 en de PIC kijkt of hij 0 geworden is. Helaas, register 'wacht1' is net van 00h naar FFh gegaan en is dus nog lang geen 0. De PIC zal dus in de loop 'blijfhier' blijven. Probeer dit mechanisme zelf te doorgronden. De PIC herhaalt deze 'decfsz' exact 256 maal voordat 'wacht1' 0 geworden is. De PIC springt over de instructie 'goto' waarna het programma wordt vervolgd. Dit is dus de basis voor een tijdvertraging: een wachtlus, een delay-routine. Om deze strategie te kunnen toepassen voor het programma LEDPIC moeten eerst enkele nieuwe registers worden gedefinieerd, zie figuur 16.2.

Zie figuur 16.3 voor de delay-routine.

Wat in de nieuwe delay-routine 'wachtmaar' gebeurt is een beetje gemeen: het is een lus binnen een lus geworden. Pas na 256 keer is 'wacht1' 0 geworden is en pas daarna wordt 'wacht2' aangesproken. Ga zelf na dat dit 256 x 256 x 3 instructies x1 ms per instructie duurt, oftewel bijna 200 ms. Dit is goed bruikbaar om de LED zichtbaar te laten knipperen.
Een aantal lezers zal zich verbazen over de loop 'knipperstart' die zo groot is en dat het wellicht een fout geeft in het knipperen van de LED. Deze fout is niet merkbaar omdat met de 196600 instructies van delay-routine 'wachtmaar' een zodanig groot beslag op de PIC wordt gelegd dat je hier niets van merkt. Het enige gevolg is dat na 100 jaar de PIC de LED nog steeds laat knipperen. Telkens wordt alles netjes opnieuw geïnitialiseerd en ingesteld. Bij deze geef ik de laatste listing (listing 16.1) waarin alle aanvullingen verwerkt zijn. Het is voor elkaar, de LED knippert! Nu voorzichtig zelf verder. De volgende keer pakken we iets heftigers.

```
LIST      P=16F84,C=75,F=INHX8M ;Definieer type processor
__CONFIG  005h                  ;‚programming configuration
                                ;word' voor PIC16F84
                                ;**********************
status    EQU     003h          ;status-register in RAM op
                                ;positie 003h
ze        EQU     2h            ;zero-vlag in het
                                ;status-register, is bit 2
rp0       EQU     5h            ;rp0-vlag in het
                                ;status-register, is bit 5
optreg    EQU     081h          ;option-register in RAM op
                                ;positie 081h, genaamd
                                ;optreg!
trisa     EQU     085h          ;poort A I/O
                                ;definitie-register in RAM
                                ;op 085h
porta     EQU     005h          ;poort A in RAM op positie
                                ;005h
                                ;**********************
          ORG     0000h         ;Origin op 0000h (0000h is
                                ;hexadecimaal 0000 enz.)
          goto    knipperstart  ;ga naar knipperstart
                                ;**********************
          ORG     0100h         ;Origin op 0100h, assembler
                                ;gaat vanaf hier verder
knipperstart clrwdt             ;Waakhond een botje geven
          bsf     status,rp0    ;Naar RAM bank 1, noemen we
                                ;ook wel RAM1
          movlw   B'00001111'   ;Waakhond krijgt de
                                ;prescaler met max.
                                ;delaytijd
          movwf   optreg        ;bewaar in file ‚optreg‚
          movlw   B'11111110'   ;laatste bit is een 0,
                                ;wordt dus een uitgang
          movwf   trisa         ;want we bewaren hem in
                                ;trisa, is pin17, de
                                ;RA0-pin!
          bcf     status,rp0    ;terug naar RAM0
          bsf     porta,0       ;LED aan
          call    wachtmaar
          bcf     porta,0       ;LED uit
          call    wachtmaar
          goto    knipperstart  ;blijf in de lus hangen, ga
                                ;naar knipperstart.
                                ;**********************
                                ;WACHTROUTINE ‚wachtmaar‚
wachtmaar nop                   ;doe een instucie niks, ter
                                ;illustratie. Later
                                ;invullen
          return                ;terug naar hoofdprogramma,
                                ;de ‚knipperstart‚-loop
                                ;**********************
          END                   ;Einde van het programma
```

Listing 15.2

```
          clrf    wacht1        ;0 => wacht1
                                ;
blijfhier decfsz  wacht1        ;verlaag wacht1 en spring
                                ;als 0
          goto    blijfhier     ;indien niet 0 naar
                                ;‚blijfhier'
                                ;
                                ;programma wordt vervolgd
```

Figuur 16.1

```
          clrf    wacht1        ;0 => wacht1
wacht1    EQU     020h          ;wachtroutine-teller 1
wacht2    EQU     021h          ;wachtroutine-teller 2
```

Figuur 16.2

```
                                ;**********************
                                ; WACHTROUTINE ‚wachtmaar'
wachtmaar clrf    wacht1        ;0 => wacht1
          clrf    wacht2        ;0 => wacht2
blijfhier decfsz  wacht1        ;verlaag wacht1 en spring
                                ;als 0
          goto    blijfhier     ;indien niet 0 naar
                                ;‚blijfhier'
          decfsz  wacht2        ;verlaag wacht2 en spring
                                ;als 0
          goto    blijfhier     ;indien niet 0 naar
                                ;‚blijfhier'
          return                ;terug naar hoofdprogramma,
                                ;de ‚knipperstart'-loop
                                ;**********************
```

Figuur 16.3

```
LIST    P=16F84, C=75, F=INHX8M ;Definieer type processor.
__CONFIG 005h                   ;'programming configuration
                                ;word' voor PIC16F84
                                ;**********************
status  EQU     003h            ;status register in RAM op
                                ;positie 003h
ze      EQU     2h              ;zero-vlag in het
                                ;status-register, is bit 2
rp0     EQU     5h              ;rp0-vlag in het
                                ;status-register, is bit 5
optreg  EQU     081h            ;option-register in RAM op
                                ;positie 081h, genaamd
                                ;optreg!
trisa   EQU     085h            ;poort A I/O
                                ;definitie-register in RAM
                                ;op 085h
porta   EQU     005h            ;poort A in RAM op positie
                                ;005h
wacht1  EQU     020h            ;wachtroutine-teller 1
wacht2  EQU     021h            ;wachtroutine-teller 2
                                ;**********************
        ORG     0000h           ;Origin op 0000h (0000h is
                                ;hexadecimaal 0000' enz.)
        goto    knipperstart    ;ga naar knipperstart
                                ;**********************
        ORG     0100h           ;Origin op 0100h, assembler
                                ;gaat vanaf hier verder
knipperstart clrwdt             ;Waakhond een botje geven
        bsf     status,rp0      ;Naar RAM bank 1, noemen we
                                ;ook wel RAM1
        movlw   B'00001111'     ;Waakhond krijgt de
                                ;prescaler met max.
                                ;delaytijd
        movwf   optreg          ;bewaar in file ‚optreg‚
        movlw   B'11111110'     ;laatste bit is een 0,
                                ;wordt dus een uitgang
        movwf   trisa           ;want we bewaren hem in
                                ;trisa, is pin17, de
                                ;RA0-pin!
        bcf     status,rp0      ;terug naar RAM0
        bsf     porta,0         ;LED aan
        call    wachtmaar
        bcf     porta,0         ;LED uit
        call    wachtmaar
        goto    knipperstart    ;blijf in de lus hangen, ga
                                ;naar knipperstart
                                ;**********************
                                ; WACHTROUTINE 'wachtmaar'
wachtmaar clrf  wacht1          ;0 => wacht1
        clrf    wacht2          ;0 => wacht2
blijfhier decfsz wacht1         ;verlaag wacht1 en spring
                                ;als 0
        goto    blijfhier       ;indien niet 0 naar
                                ;'blijfhier'
        decfsz  wacht2          ;verlaag wacht2 en spring
                                ;als 0
        goto    blijfhier       ;indien niet 0 naar
                                ;'blijfhier'
        return                  ;klaar, terug naar
                                ;hoofdprogramma, de
                                ;'knipperstart'-loop
                                ;**********************
        END                     ;Einde van het programma
```

Listing 16.1