EPPO-PIC-programmer

Eric Post, PE1MIX
pe1mix@amsat.org

Zoals aangekondigd in het maartnummer van *Electron*, is op de radiomarkt in Rosmalen het EPPO-project geïntroduceerd. Nu alle kopers even de tijd hebben gehad de schakeling in elkaar te zetten, gaan we hem gebruiken. In eerste instantie is dit project door initiator Dolf, PA3EGT, bedoeld als tool om hobbyisten een zetje te geven om PIC-processoren in C te programmeren maar wij gebruiken het met een gratis ontwikkelomgeving van Microchip: MPLAB. Het ontwerp is door Marc Simons, PE1RRT, hardwarematig uitgewerkt zodat we beschikken over een professionele print en kwaliteitsonderdelen. Dolf heeft de programmeersoftware geschreven. Op de foto is de compleet opgebouwde print te zien. De processor is door het flashgeheugen duizenden malen opnieuw te programmeren en kan elektronisch gewist worden. De instructieset bevat slechts 35 instructies. Na wat programmeerervaring hebben deze geen geheimen meer. Laat ik meteen de zaak nuanceren: helaas, het is niet supersimpel of intuïtief en je krijgt het echt niet cadeau. Je zult met deze processoren moeten stoeien om ervaring op doen. Heb je deze eenmaal opgedaan dan is het een ideaal hulpmiddel voor bijvoorbeeld schakel- en timingklusjes. Niet in de laatste plaats omdat functiewijzigingen doorgevoerd kunnen worden door de processor te programmeren met nieuwe software. Geen elco's meer bijplakken om tijden te verlengen maar gewoon een getal in de software veranderen. Met de 35 instructies kan veel gedaan worden met een hoge verwerkingssnelheid. Als je de instructieset eenmaal kent, gaat programmeren snel. In het begin gaat de meeste tijd zitten in het opzoeken van de instructies en hun effecten op de registers. Een praktisch hulpmiddel is dat deel van de datasheet waar de instructies staan. Het betreft negen pagina's. De beperkte instructieset heeft ook een nadeel: ingewikkelder berekeningen zoals deling van getallen is lastig en kost veel coderuimte. De 35 instructies zijn verzonnen om het ons gemakkelijk te maken. De PIC-processoren zelf snappen niets van die instructies en verwachten uiteindelijk gewoon enen en nullen. Om iets te kunnen doen is MPLAB noodzakelijk. Het is bij Microchip gratis te downloaden. Dit is een ontwikkelomgeving die we kunnen gebruiken om een programma te schrijven, op fouten te controleren en de HEX-file te genereren die we in de PIC-processor kunnen programmeren om vervolgens te laten uitvoeren.

Het bouwpakket

Het pakket wordt zeer compleet geleverd: een professioneel vervaardigde print, alle onderdelen, een cd-rom en zelfs de benodigde seriekabel. Een PIC-processor 12F675 ontbreekt niet. Door de opgedane ervaring en daardoor voortschrijdende ontwikkeling van de software is het aan te raden na het bouwen de laatste versie van internet te halen. Zorg dat de programmer tijdens gebruik altijd onder spanning blijft staan. Is de spanning weg geweest, start de software dan opnieuw. Op de cd-rom staat ook veel achtergrondinformatie en de documentatie van de hardware. In de directory EPPOMANAGER\MANUALS staan schema en layout. Minder ervaren bouwers kunnen de layout afdrukken en bijvoorbeeld met een markeerstift aangeven welke onderdelen al geplaatst zijn. Het assembleren van de print zal waarschijnlijk voor niemand problemen opleveren. Kijk goed naar de diodes want het verschil tussen de 6V8 zener en de 1N4148 is lastig te zien (eigenlijk te lezen). Amateurs die niet zo bedreven zijn met metaalfilmweerstanden kunnen deze beter voor montage even met een universeelmeter controleren. Een factor tien kun je er anders al snel naast zitten. Op het demoboardgedeelte zit helaas een kleine fout. De foute sporen zijn al doorgekrast maar met dun draad moeten er nog twee doorverbindingen worden gelegd. Op internet is dit beter te zien dan op de bijgevoegde layout. Bij de montage van LEDs onthoud ik zelf altijd de afkorting KNAP. Dat staat voor Kathode Negatief Anode Positief. Als je bedenkt dat de lange poot aan de LED altijd de plus is, hoef je nooit meer te twijfelen over hoe een LED gemonteerd moet worden. Indien gewenst kunnen programmer en demoboard van elkaar worden gescheiden en separaat worden gebruikt. Voor de programmeerspanning van 12,6 volt heeft het programmeergedeelte een voedingsspanning nodig van 15-20V. Hiervoor zijn verschillende oplossingen mogelijk. Ik heb zelf een standaard netadapter met een LM317 gemodificeerd zodat deze ongeveer 15 volt levert.

Fig. 1 Compleet opgebouwde print

Installatie van MPLAB

Aangezien de meeste amateurs wel enige computerervaring hebben, ga ik snel door de installatie heen. Ga naar http://www.microchip.com en dan naar 'design MPLAB IDE'. Daar is de laatste versie als zipfile te downloaden. Ik gebruik versie 7.50 onder Windows XP. Pak de zipfile uit en start 'MPLAB V7.50 Install' zonder dat er verdere applicaties openstaan. Doe de installatie met de standaard settings: klik next, next, 'complete' aangevinkt, next, next (standaard directory), next, next, klik 'MPLAB IDE Document select' weg, sluit installatie af. Nu staat er op de desktop een icoon voor MPLAB V7.50. Maak onder de Microchip-directory een directory aan met de naam 'projects'. Het volledige pad zal dan zijn c:\Program Files\Michrochip\projects. Download van de museumsite http://www.jancorver.org/bouw/eppo/ de file 'knipper.asm' en sla deze op in de projects-directory.

Aan de slag met MPLAB

We starten MPLAB en maken een project aan. Weer in vogelvlucht: klik onder 'project' de Project Wizzard aan, next, kies de PIC12F675, next, next, gebruik als project name 'knipper', browse naar de aangemaakte projects-directory en selecteer deze, next, selecteer in het linkerschermpje 'knipper.asm' en toets 'add', next, finish. Dubbelklik in het 'knipper.mcw'-venster op 'knipper.asm'. Bovenstaande wijst zichzelf in de praktijk. We zien nu het programma 'knipper.asm' als tekst op ons scherm staan. Dit programma lijkt een grote brij. Het valt in de praktijk gelukkig nogal mee. Ik heb het door Microchip geschreven programma 'f675temp.asm' gebruikt. Hierin zijn al veel zaken geregeld die we dus zelf niet meer hoeven te doen. Het zelfgeschreven stuk staat tussen 'remaining code goes here' en 'initialize eeprom locations'. Dit is nog een hele lap tekst maar hopelijk goed te volgen door het commentaar. Uiteraard is het niet nodig het wiel telkens opnieuw uit te vinden. Ik grijp daarom vaak terug op programma's of delen daarvan die ik al eerder heb gemaakt. Deze gebruik ik opnieuw, eventueel na aanpassing.

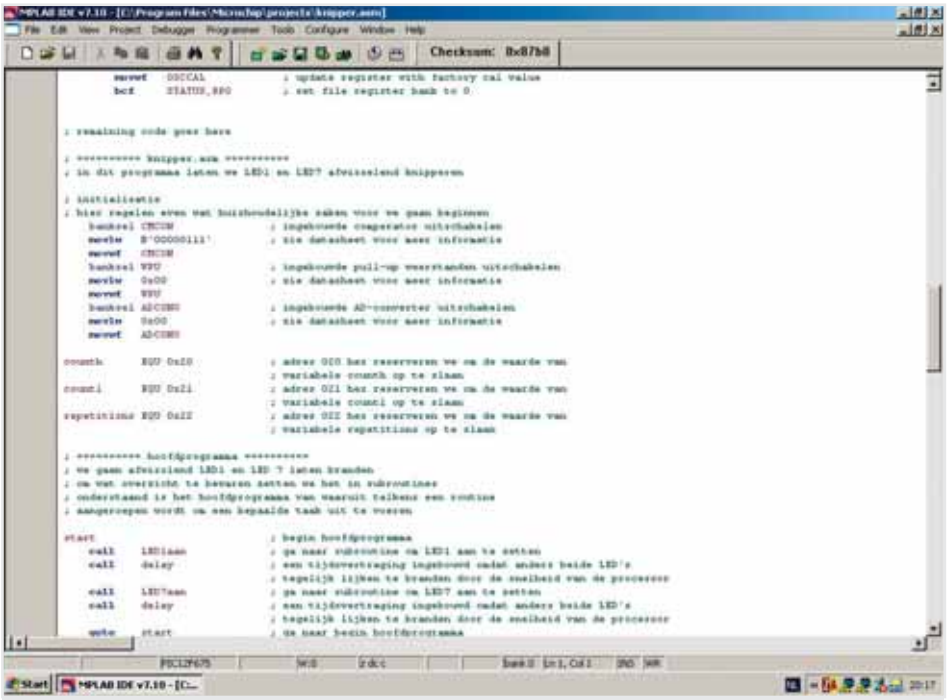

Fig. 2 Screenshot MPLAB met bestand knipper.asm

Het demoboard

Er zijn zes IO-lijnen, we gaan twaalf LEDs aansturen. Dit is mogelijk doordat elke IO-lijn als ingang of uitgang te definiëren is. Dit kan zelfs tijdens het uitvoeren van het programma. Als we een IO-lijn als ingang definiëren, wordt hij hoogohmig en kan daar geen stroom door lopen. Om LED1 te laten branden moet GP5 als uitgang hoog worden en GP4 als uitgang laag. LED1 zal nu branden. Om te voorkomen dat LED3 en LED5 gaan branden zullen we GP1 en GP2 als ingang moeten schakelen. Op deze manier zijn er met vier IO-lijnen twaalf LEDs te schakelen. GP0 en 3 hebben we niet nodig om de LEDs aan te sturen. Om stroom te besparen worden ze als ingang geschakeld. In de tabel is te zien welke uitgangen hoog en laag moeten zijn om een bepaalde LED te laten branden. IO-lijnen waarbij niets is aangegeven worden als ingang geschakeld. Bedenk bij het ontwerpen van schakelingen en het schrijven van software dat door de interne structuur van de PIC-processor GP3 alleen als ingang en niet als uitgang gebruikt kan worden (zie pagina 25 van de datasheet).
Op pagina 10 van de datasheet van de 12F675 staat dat het geheugen van de PIC-processor is opgedeeld in twee verschillende banken.

Tijdens uitvoer van het programma kan het nodig zijn van bank te wisselen om in het juiste register te schrijven. Dit kan met het MPLAB-commando 'banksel'. We zullen in ons testprogramma vaak de registers TRISIO en GPIO gebruiken. Met TRISIO stel je de poorten in. Bij een 0 is het betreffende bit als uitgang geschakeld, bij een 1 als ingang. GPIO dient om een uitgang daadwerkelijk hoog of laag te maken. Omdat het niet mogelijk is om direct een waarde naar een register of geheugenlocatie te schrijven, wordt dit via het W-register gedaan. Eerst laden we dit met de gewenste waarde en dan schrijven we dit naar bijvoorbeeld TRISIO. Het W-register is dus het middelpunt van veel handelingen. Een heel register 'clearen', dat wil zeggen alle bits 0 maken, is wel mogelijk zonder gebruik van het W-register. Met bovenstaande kennis in ons achterhoofd moet het volgende programma te begrijpen zijn. Alles achter puntkomma's is commentaar en wordt dus niet als programmadata gezien.
Het programma zoals we het zien in de listing is te verdelen in een aantal stappen:

- Initialisatie
- Begin
- Doe LED1 aan
- Tijdvertraging
- Doe LED7 aan
- Tijdvertraging
- Ga terug naar het begin

Als we de vertraging niet zouden programmeren, zou het in- en uitschakelen van de LEDs zo snel gaan dat ze allebei tegelijk

	LED											
	1	2	3	4	5	6	7	8	9	10	11	12
GP1					0	1			0	1	0	1
GP2			0	1			0	1			1	0
GP4	0	1					1	0	1	0		
GP5	1	0	1	0	1	0						

Tabel 1 Status van ingang of uitgang om een bepaalde LED te laten branden

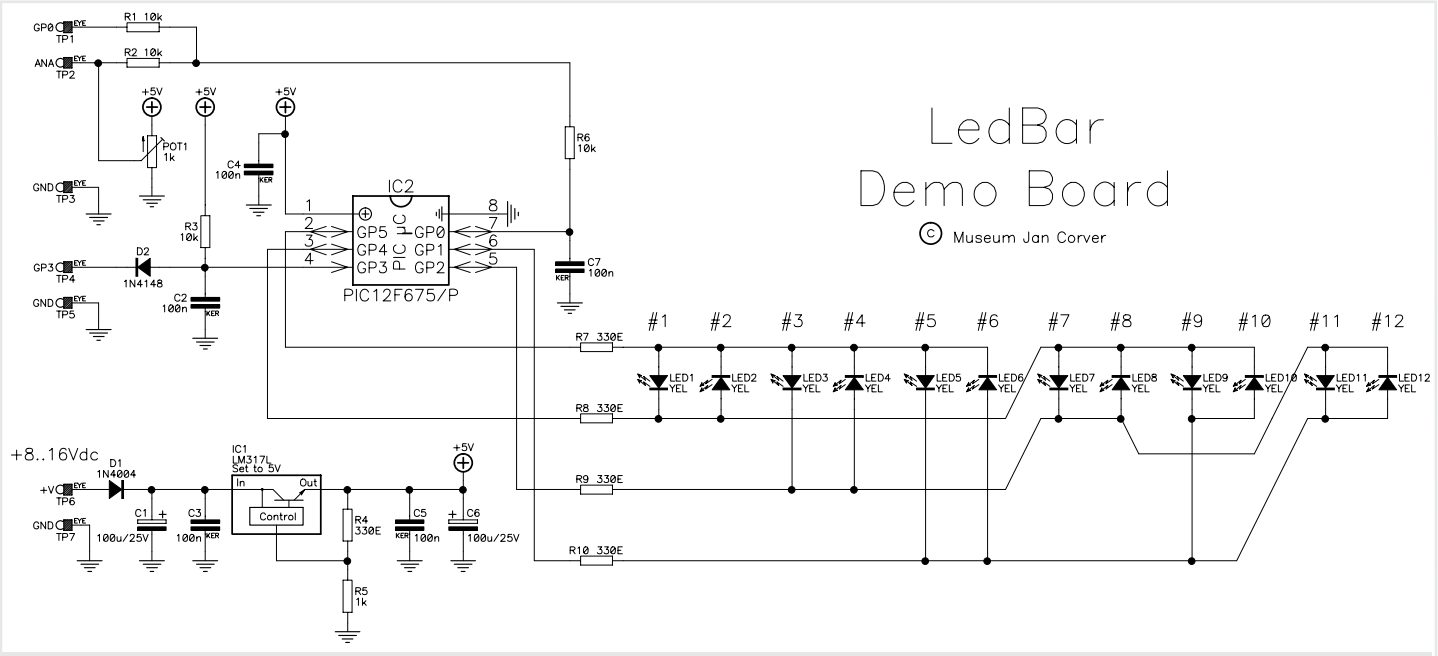

Fig. 3 Schema van demoboard

lijken te branden. Voor de duidelijkheid zijn de aansturing van de LEDs en de tijdvertraging als aparte routines geprogrammeerd. Een bijkomend voordeel is dat we de dezelfde vertragingslus twee keer aan kunnen roepen wat geheugenruimte en programmeertijd scheelt.

Enkele stappen in detail

Initialisatie

Hierin schakelen we features van de processor aan of uit, afhankelijk van wat we willen. In dit geval schakelen we de ingebouwde comparator, pull-up-weerstanden en analoge ingangen uit. Ook definiëren we geheugenplaatsen voor variabelen die we willen gebruiken. We hoeven geheugenplaatsen niet als getal te adresseren maar kunnen rechtstreeks naar een naam verwijzen. Dit komt de leesbaarheid van ons programma ten goede. Een geheugenplaats is te vergelijken met een laatje in een ladekast waar we in ons geval een getal in bewaren of uit kunnen halen.

LED1 aan

Eerst definiëren we de toestand van de IO-pinnen (TRISIO). Daarna maken we alle uitgangen laag. Mocht een van de uitgangen nog hoog zijn uit een andere subroutine, dan zal deze hoog blijven als we dit niet actief corrigeren. Hierna maken we bit 5 van GPIO hoog om LED1 te laten branden. Met deze kennis is de subroutine om LED7 te laten branden simpel te doorgronden.

Tijdvertraging

Elke instructie die de processor uitvoert kost tijd. Niet veel maar het kost tijd. Als we de processor veel instructies uit laten voeren, kunnen we op deze manier een vertraging programmeren. Kijk daarvoor in de listing. De vertraging bestaat uit drie lussen die een aantal malen herhaald worden. In dit geval is gekozen voor een standaardvertraging van 100 ms en deze te herhalen. De vertraging is dan uit te rekenen. Bekijk het commentaar in de listing, dan wordt het wel duidelijk. Een precieze beschrijving van alle instructies is in de datasheet van de PIC-processor te vinden. De MPLAB-commando's zijn niet alleen te vinden in de handleiding die op dezelfde plaats te downloaden is als de software maar ook via de helpfunctie van MPLAB. Als ontwikkelhulp heeft MPLAB nog veel meer potentie waar ik hier niet op in kan gaan: programma's kunnen zelfs softwarematig gesimuleerd worden, met breakpoints kunnen tijdvertragingen gemeten worden. Als we het programma in MPLAB hebben staan, gaan we naar 'project' en kiezen voor 'build all'. Met dit commando laten we MPLAB ons programma omzetten naar de HEX-code die nodig is om de PIC-processor te programmeren. Tijdens deze omzetting wordt meteen gecontroleerd op programmeerfouten. Als we bijvoorbeeld variabelen gebruiken die we niet eerst gedefinieerd hebben, zal het programma daar melding van maken. Ter illustratie heb ik eens in het begin van het programma een typefout gemaakt en i.p.v. 'repetitions EQU' 'repetition EQU' getypt. In de screendump is te zien dat we dan een melding krijgen dat we 'repetitions' niet eerst gedefinieerd hebben. Op deze manier

Het programma in grote lijnen

```
; ========== knipper.asm ==========
; in dit programma laten we LED1 en LED7 afwisselend knipperen
; alle verwijzingen zijn naar datasheet DS41190D uit 2007
; in andere versies kan het op een andere pagina staan

; initialisatie
; hier regelen we huishoudelijke zaken voor we gaan beginnen
            banksel   GPIO            ; selecteer memorybank waarin GPIO zit
            clrf      GPIO            ; maak GPIO compleet laag
            movlw     B'00000111'     ; ingebouwde comperator uitschakelen
            movwf     CMCON           ; datasheet figuur 6.2 pagina 37
                                      ; CMCON zit in dezelfde bank als GPIO
            banksel   WPU             ; selecteer memorybank waarin WPU zit
            clrf      WPU             ; ingebouwde pull-up weerstanden uitschakelen
                                      ; datasheet register 3-3 pagina 20
            clrf      ANSEL           ; digitale IO i.p.v. AD-ingangen
                                      ; ANSEL zit in dezelfde bank als WPU
                                      ; meer info example 3-1 pagina 19

counth      EQU       0x20            ; adres 020 hex reserveren we om de waarde van
                                      ; variabele counth op te slaan
countl      EQU       0x21            ; adres 021 hex reserveren we om de waarde van
                                      ; variabele countl op te slaan
repetitions EQU       0x22            ; adres 022 hex reserveren we om de waarde van
                                      ; variabele repetitions op te slaan

; ========== hoofdprogramma ==========
; we gaan afwisslend LED1 en LED 7 laten branden
; om overzicht te bewaren zetten we het in subroutines
; onderstaand is het hoofdprogramma van waaruit telkens een routine
; aangeroepen wordt om een bepaalde taak uit te voeren

start                                 ; begin hoofdprogramma
            call      LED1aan         ; ga naar subroutine om LED1 aan te zetten
            call      delay           ; een tijdsvertraging ingebouwd omdat anders beide LEDs
                                      ; tegelijk lijken te branden door de snelheid van de processor
            call      LED7aan         ; ga naar subroutine om LED7 aan te zetten
            call      delay           ; een tijdsvertraging ingebouwd omdat anders beide LEDs
                                      ; tegelijk lijken te branden door de snelheid van de processor
            goto      start           ; ga naar begin hoofdprogramma

; ========== subroutines ==========

; ********** LED1 aan **********
; GP0, 1, 2, 3 als ingang schakelen
; GP4 en 5 als uitgang schakelen
LED1aan                               ; naam van subroutine
            banksel   TRISIO          ; ga naar BANK1 waarin TRISIO zich bevindt
            movlw     B'00001111'     ; definieer in en uitgangen
                                      ; tel van rechts naar links, 0 als uitgang, 1 als ingang etc.
            movwf     TRISIO          ; schrijf IO definitie naar TRISIO
            banksel   GPIO            ; selecteer BANK0 waarin GPIO zich bevindt
            movlw     B'00000000'     ; maak voor de zekerheid alle uitgangen 0
            movwf     GPIO            ; we weten niet met welke status GPIO hier aankomt
            BSF       GPIO,5          ; maak GP5 hoog
            return
; ********** einde LED1 aan **********


;  ********** LED7 aan **********
; GP0, 1, 3, 5 als ingang schakelen
; GP2 en 4 als uitgang schakelen
LED7aan                               ; naam van subroutine
            banksel TRISIO            ; ga naar BANK1 waarin TRISIO zich bevindt
            movlw     B'00101011'     ; definieer in en uitgangen
                                      ; tel van rechts naar links, 0 als ingang, 1 als ingang etc.
            movwf     TRISIO          ; schrijf IO definitie naar TRISIO
            banksel   GPIO            ; selecteer BANK0 waarin GPIO zich bevindt
            movlw     B'00000000'     ; maak voor de zekerheid alle uitgangen 0
            movwf     GPIO            ; we weten niet met welke status GPIO hier aankomt
            BSF       GPIO,4          ; maak GP4 hoog
            return
;  ********** einde LED7 aan **********


; ********** universele vertraging **********
;
; vul variable repetitions met D'01' (decimaal 01) voor elke 100 msec
; dus voor 1 seconde D'10'
delay                                 ; naam van subroutine
            movlw     D'05'           ; vul het W register met decimaal 5 voor 500 msec
            movwf     repetitions     ; schrijf dit naar de repetitieteller
                                      ; deze staat voor het aantal herhalingen van de 100 msec vertraging
outerloop 100ms                       ; start van buitenlus, wordt nu in dit geval 5 maal herhaald
            movlw     D'200'          ; vul het W register met decimaal 200
            movwf     counth          ; schrijf dit naar counth
middleloop 100ms                      ; deze buitenlus wordt dus 200 maal herhaald
            movlw     D'99'           ; vul het W register met decimaal 99
            movwf     countl          ; schrijf dit naar countl
innerloop 100ms                       ; deze binnenlus wordt dus 99 maal herhaald
            nop                       ; geen instructie maar kost wel processortijd
            nop                       ; geen instructie maar kost wel processortijd
            decfsz    countl,f        ; trek 1 van countl af en sla als deze 0 is de volgende
                                      ; instructie over
            goto      innerloop100ms  ; countl is nog geen 0 dus gaan we de binnen lus nog maar eens doen
            nop                       ; geen instructie maar kost wel processortijd
            decfsz    counth,f        ; trek 1 van counth af en sla als deze 0 is de volgende
                                      ; instructie over
            goto      middleloop100ms ; counth is nog geen 0 dus gaan we de middelste lus nog maar eens doen
            decfsz    repetitions,f   ; trek 1 van aantal herhalingen af en sla als deze 0 is de volgende
                                      ; instructie over
            goto      outerloop100ms  ; nog niet alle herhalingen gehad, dus het geheel nog een keer
      return                          ; het gewenste aantal herhalingen is doorlopen dus gaan we terug
; ********** einde universele vertraging **********
```

zijn fouten er vooraf uit te halen. Omdat MPLAB niet weet wat de bedoeling is, kunnen niet alle fouten gedetecteerd worden. Fouten als 'naar TRISIO schrijven terwijl we in de bank met GPIO zitten' zijn snel gemaakt. Dit zijn lastig op te sporen zaken waar we zeer attent op moeten zijn. Ook kan het lastig zijn om het overzicht te bewaren als we bijvoorbeeld op zero of carry gaan testen. Ik kan het niet vaak genoeg herhalen: ervaring is alles. Op de site van Microchip en op internet in het algemeen is veel data te vinden waar we als beginner onze kennis mee op kunnen bouwen. Rondsnuffelen bij Microchip en zoeken met Google kost tijd maar levert veel op. Het is raadzaam om goed op te letten bij het overtypen van programma's of het knippen en plakken van code uit bijvoorbeeld tekstverwerkers. Ik heb gemerkt dat Wordpad en Word de draad wel eens kwijt willen raken omdat de 1 en de l er in MPLAB hetzelfde uitzien. Ik heb dat gehad met de variabele countl die in een aantal gevallen als count1 werd weergegeven.

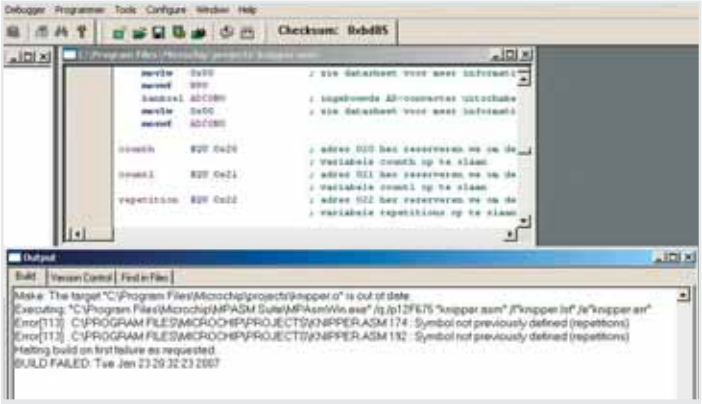

Fig. 4 Screendump van foutmeldingen

Installatie EPPO-manager

Het is nu tijd om de PIC-processor daadwerkelijk te programmeren. Download de laatste versie van de EPPO-manager van de EPPO-website. Deze is te bereiken via de museumpagina. Download de file met de laatste versie naar een subdirectory en pak deze uit. Er hoeft verder niets geïnstalleerd te worden. Enkele settings moeten worden aangepast. Sluit de programmer aan, start de EPPO-manager op en ga naar 'files'. Haal, mocht dit nog niet zo zijn, voorlopig het vinkje bij 'autoreload' weg. Kies bij 'files' voor 'settings' en stel daar de gebruikte COM-poort in en de 'script directory'. Eventueel kan nog een default editor ingesteld worden. Mocht de gebruikte COM-poort niet COM1 zijn, sluit dan na instellen de EPPO-manager af en start deze opnieuw. Hierna is alles klaar voor gebruik. De EPPO-manager maakt gebruik van scripts om de diverse processoren te kunnen programmeren. Het vervaardigen van deze scripts is niet iets voor een beginner. De scripts voor de 12F675 en de 16F84 zijn compleet uitgewerkt en daarmee kunnen we voorlopig vooruit. De 16F84 is een veelgebruikte processor in bestaande amateurontwerpen en de 12F675 is ideaal om mee te experimenteren. Op termijn zullen er waarschijnlijk nieuwe scripts beschikbaar komen. Op de (Engelstalige) site van EPPO staan hardware en software uitgebreid beschreven. Veel vragen zullen daar beantwoord worden.

Fig. 5 PIC-processor in extra voetje om beschadiging te voorkomen

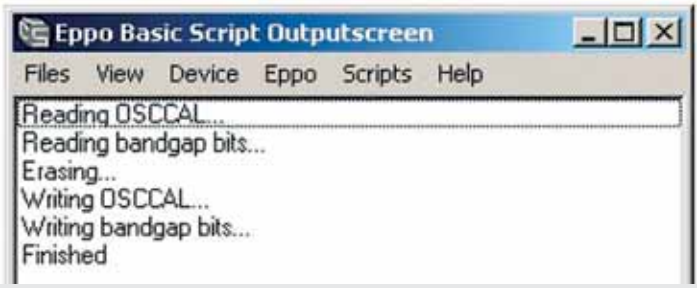

Fig. 6 Wissen van de PIC-processor

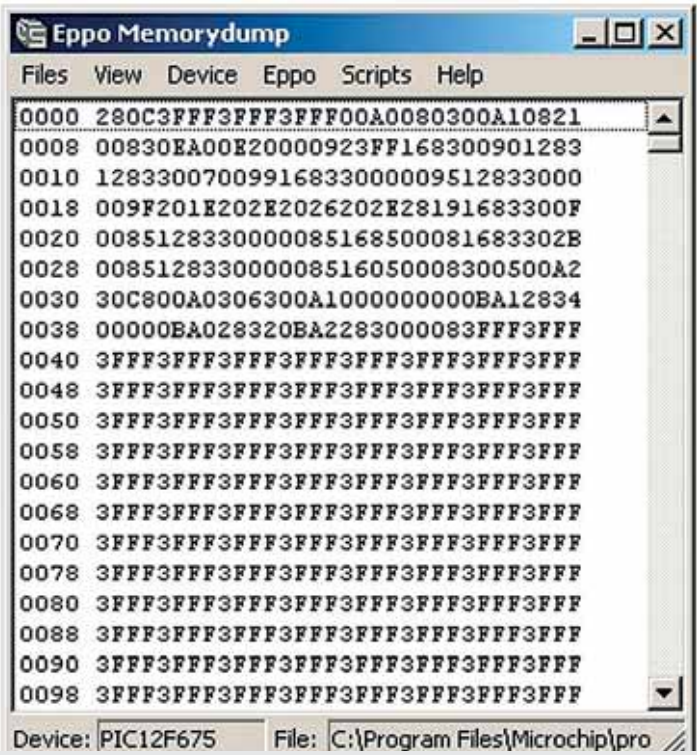

Fig. 7 Laden van het programma

Fig. 8 Programmeren van de PIC-processor

Programmeren van de PIC-processor

Selecteer via 'device' de door ons gebruikte 12F675. Doe de processor in de programmer. Op de print staat duidelijk aangegeven waar. Omdat het tijdens het ontwikkelen van software vaak voorkomt dat de processor in en uit het voetje gaat, kies ik er meestal voor deze in een extra IC-voetje te doen. Deze dient dan als bescherming van de IC-pootjes. Als het voetje kapot gaat, is het goedkoper dat te vervangen dan de processor zelf. Eerst wissen we de processor. Kies 'EPPO' en dan 'erase'. De software zal aangeven als dit gebeurd is. Via 'files' en 'open HEX-file' kiezen we 'knipper.hex' in de projects-directory. Dubbelklik hierop en de file wordt geladen. Een blok getallen is nu zichtbaar. Dit is de code zoals die in de PIC geprogrammeerd wordt. Duidelijk is te zien dat ons programma uit slechts acht regels code bestaat en dat de PIC-processor nog lang niet vol is. Om te programmeren kiezen we 'program' onder 'EPPO'. De code wordt nu naar de processor geschreven. Daarna kan de processor in het demoboard gestoken worden en na het aanbieden van de voedingsspanning zullen LED1 en LED7 afwisselend branden. Met de nu opgedane kennis moet het mogelijk zijn zelf andere LEDs aan te sturen en de vertragingstijd te veran-

deren. Wellicht dat in we in een volgend artikel de AD-converter gebruiken en een schakelaar aansluiten. Om het helemaal gemakkelijk te maken is op de museumsite ook een programma te downloaden waarbij de routines voor elke LED al geprogrammeerd zijn. Via het museum hopen we met dit project een leuke, betaalbare en vooral complete mogelijkheid te bieden om kennis te maken met PIC-processoren en ze voor onze hobby in te zetten. Ondanks het feit dat ik andere programmeeralternatieven heb, heb ik met veel plezier deze kit gebouwd en gebruikt als voorbereiding op het schrijven van dit artikel. Veel informatie staat op de museumsite. Vragen kun je daar stellen. Ik heb geprobeerd diverse onderwerpen te behandelen. Het is helaas niet mogelijk om overal diep op in te gaan. Ik hoop dat ik genoeg informatie aangeleverd heb voor een tastbaar resultaat dat zelf kan worden uitgebouwd. Het is eenvoudiger iets te wijzigen dat al werkt dan wanneer je van nul af aan moet beginnen en niet weet waar het fout gaat. Hopelijk nodigen het bouwpakket en dit artikel mensen uit om te experimenteren met PIC-processoren. Ik wens iedereen veel succes en plezier met het knutselen.

advertentie

WWW-WLAN-SHOP-NL

Groot assortiment WIFI Producten uit voorraad leverbaar.
Voor 15.00 uur besteld - volgende dag in huis.
Afhalen op afspraak - 6 dagen per week.
Online bestellen - betalen met Ideal mogelijk.

HAWKING TECHNOLOGY — canopii

Antenne's
Routers
IP Camera's
Printservers

Antenne's
Connectoren
Verloop-
kabels

LINKSYS — SENAO

VoIP
Repeaters
Routers
AP's

Routers
AP's
Client
Bridges

Overige merken: Speedtouch - Tornado - OvisLink - Canyon - Pheenet - Devolo - Tranzeo.

www.wlan-shop.nl - Onderdeel van Professional Telecom Support te Hilversum
T: 035-7722123 ■ F: 035-6018064 ■ M: info@wlan-shop.nl

ANTENNE - BOUW
Bijzen

TEL. 0521 - 524410 FAX. 0521 - 524430
KORTE VENEN 2A 8331 TH STEENWIJK

voor info bezoek onze website
www.antennebouwbijzen.nl

Het beste voor u geïmporteerd:

SPE Linear Amplifier 1 kW nu leverbaar bij Webb Industries Europe

SPE LINEAR EXPERT 1K-FA.
The most technologically advanced in the world!

Two CPU's are used, one of which is dedicated to the C.A.T.'s. Pi-L output circuit. Over 13000 lines of SW for performances that cannot be found together in other amplifier.

Fully automatic!
Easy connection with all models "Icom, Yaesu, Kenwood" for immediate management of the bands, tuner and antennas. Same performance with all makes or homemade rigs. The operator has only to move the "Frequency Tuning Knob" on the transceiver!!!

The smallest in the world!
Built-in Power Supply and Automatic Antenna Tuner.
Dimensions: W 28 x H14 x D 32 cm. (connectors included).
Weight: approx. 20 kg.

General Spec's.
1.8 to 50MHz including WARC Bands.,Full Solid State. Large display. 1 KW PEP SSB Out; 900W CW out (typ). 700 Watt PEP out (typ) on 50MHz. Automatic Tuner built in.Two inputs and four outputsFully protected.RS232 Port for PC Control.
Full details on www.linear-amplifier.com. **3199,-**

Specifications are subject to change without notice.

Webb [industries] Europe
Im- & export of communication hardware, accessories and fun electronics

Elektronweg 12, 3542 AC Utrecht, The Netherlands
T : +31(0)30-789 00 40. F : +31(0)30-267 1 7 32 | info@webbindustries.nl

Webb [Industries] Europe B.V.
Exclusive Benelux distributor of Peiker, SPE, Hilberling, Weybrook and ImpactAudio.