Microcontroller

De PIC 16F & 18F voor beginners

Opdracht Elecktronica “Programmeren C in PIC`s

Document Versie : Ver 1.0
Plaats : Groningen

Ismaêl Drenth

2008

ISBN XX-XXX-XXXX-X

Eerste druk, eerste oplage 2008

Microcontroller

De PIC 16F & 18F voor beginners

Opdracht Elektronica “Programmeren C in PIC`s

Document Versie : Ver 1.0
Plaats : Groningen

Ismaêl Drenth

2008

Inhoud

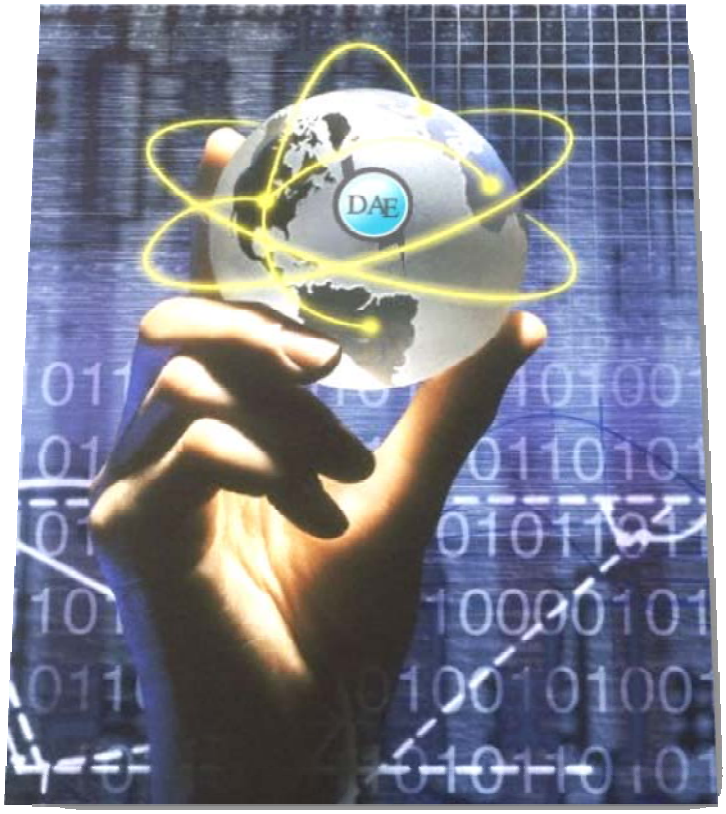
Voorwoord

Dit is een technische cursus van de firma Drenth Automatic Electronics.

Deze cursus is bedoeld voor de richting AEC (Automatisering Elektronica), AEN (Automatische Energie), Mechatronica, HT (Human Technology), bedrijven en eventuele geïnteresseerden.
U wordt begeleid op de leer manier: eerst theorie en daarna de praktijk.
De leer fase wordt u stapsgewijs uitgelegd: eerst de ondergrond, daarna op elektronica basis en vervolgens de praktijk fase.
Dit verslag is gebaseerd op het idee dat veel scholen en bedrijven geen idee hebben hoe belangrijk de basis is en hoe de opdracht geklaard moet worden.
Deze documentatie kan als naslagwerk of als lesmateriaal gebruikt worden.
Ik wens jullie een hele fijne les/hobby dagen toe, en ik zie eventuele vragen graag tegemoet op het forum www.D-A-E.eu (kies forum).

Met vriendelijke groet:

Ismaêl Drenth.



Figuur 1.1: DAE technologische connecties.

Samenvatting

Deze documentatie gaat over microcontrollers qua geschiedenis, toepassingen, onderzoek, programmeren en praktijkoefeningen. Deze documentatie kan gebruikt worden als naslagwerk voor scholen, bedrijven en hobbyisten. Het doeleinde van de documentatie is het aanleren van het gebruiken van microcontrollers. In het eerste deel wordt duidelijk verteld over de geschiedenis van microcontrollers , wat er eerst was en hoe het nu uitgebreid wordt. De toepassing van de microcontrollers binnen het bedrijfsleven en onder hobbyisten wordt duidelijk na voren gebracht. Op verschillende plekken komt onderzoek aan bod. Hier wordt onder meer beschreven hoe je problemen kan opzoeken en welke berekeningen bij bepaalde componenten toegepast moeten worden. Na een kort overzicht van de verschillende bestaande programmeer talen komt de krachtigste programmeertaal voor IC's uitgebreid aan bod. Binnen praktijkoefeningen wordt stapje voor stapje verteld over de toepassingen en het gebruik van elektronica. Daarna wordt alle theorie gebruikt voor een praktijk opdracht waarbinnen je de basis van programmeren en compileren aanleert.

1 Inleiding

1.1 Onderzoek DAE testboard

In eerste instantie wordt er gewerkt met het DAE-testboard, een lowcost ontwikkelbord van de firma Drenth Automatic Electronics dat te bestellen is op het internet. De 16F877A microcontroller wordt hier als kern gebruikt. Alle interessante mogelijkheden van deze microcontroller worden uitvoerig onderzocht met behulp van het DAE-testboard.

In dit deel wordt er dus geen overkoepelend eindproduct ontwikkeld of iets totaal nieuws gemaakt. De bedoeling is hier dat de basismogelijkheden van de combinatie van de 16F877A en het DAE-testboard onderzocht worden. Er wordt hier dus enkel software ontwikkeld, in de C of assembleertaal. De resultaten van deze opgave kunnen in volgende opleidingen of bij bedrijven als referentie of als naslagwerk gebruikt worden om verdere toepassingen te ontwikkelen. Om bepaalde routines en programmeerwijzen te verduidelijken wordt er gedocumenteerd in flowcharts. De uitwerking van deze programma's is later in deze documentatie opgenomen.

Concreet worden de volgende mogelijkheden bestudeerd:

Deel 1
- -Digitale I/O
  - o Inlezen van drukknoppen en aansturen van LED's

Deel 2
  - o Aansturen van 4 gemultiplexte 7segment displays
  - o Aansturen LCD display d.m.v. een 4 of 8 bits databus
- -Interrupts
- -A/Dconversie
- -Seriële communicatie opzetten (RS232)
- -Timers
- -Pulsbreedte Modulatie (PWM: Pulse Width Modulation)
- -EEPROM
- -Resetcondities

LET OP!! Woorden met * worden achterin het (Gebruikte symbolen en afkortingen) uitgelegd.

1.2 Vermogentoepassing

In het tweede deel van dit eindwerk wordt afgestapt van het ontwikkelbord en wordt er een concrete toepassing uitgewerkt voor het labo vermogenelektronica. Hier wordt er gebruik gemaakt van een andere microcontroller, namelijk de 16F690, die enkele verbeterde enhanced* functies bevat. Vooral de 'enhanced capture and compare PWMmodule' wordt van dichtbij bekeken.

Via pulsbreedte modulatie is het mogelijk om FET's gecontroleerd en precies te schakelen. Zo kun je vanuit een gelijkspanning (DC) een wisselspanning (AC) vormen, via de nodige PWM-signalen. Voor deze toepassing moet vanzelfsprekend een printplaat PCB* ontwikkeld worden met de nodige randcomponenten. De precieze werking en opbouw van deze schakeling en de ontwikkelde software wordt later in deze documentatie besproken.

2 De PICmicro

2.1 Wat is een microcontroller?

Een microcontroller IC* is een informatieverwerkend systeem in 1 enkele chip dat gebruikt wordt om elektronische apparatuur te besturen. In tegenstelling tot een microprocessor die je in een PC terugvindt zijn deze chips op enkele randcomponenten na volledig op zichzelf werkend. Je vindt deze componenten terug in: magnetrons, wasmachines, telefoons en zelfs speelgoed. Een typische IC is onderaan deze pagina afgebeeld.
De IC is ontworpen om (middels de CPU*) toegewijde, specifieke taken uit te voeren, die hij uit zijn niet volatiel (vluchtig) programmageheugen ROM* ophaalt. De nodige bewerkingen worden uitgevoerd (RAM*) om zodoende gepast te reageren via een interface* of via periferie* I/O*. Een IC is (her)programmeerbaar, waardoor het een flexibele component is. Het zijn relatief goedkope IC's, waardoor ze wijdverspreid zijn. De meerderheid van de computersystemen die tegenwoordig gebruikt worden zijn 'Embedded Systems'. De IC valt ook onder deze categorie en is waarschijnlijk het eenvoudigste voorbeeld van een ingebed of geïntegreerd systeem. Ingebedde systemen hebben meestal minimale eisen qua geheugen en programmalengte, wat hen compact maakt. Deze systemen ondersteunen simpele, maar op het eerste zicht ongewone I/O. Een monitor, harde schijf of een printer van een PC kan niet zomaar aangesloten worden. Voorbeelden van I/O waar deze componenten wel goed mee overweg kunnen zijn: een relais, een schakelaar, een LED of zelfs een spanning. Deze systemen vind je vanzelfsprekend terug in elektronische sturingen of regelkringen.

Figuur 1.2: een IC in DIP behuizing

PIC's zijn bijzonder populair bij ontwikkelaars en hobbyisten om de volgende redenen:

-Lage kostprijs (enkele euro's per component)
-Hoge beschikbaarheid en ruim varianten aanbod
-Makkelijk overstapbaar naar andere microcontrollers in C.
-Uitgebreide collectie van toepassingsnotities
-Lage kostprijs van de ontwikkeltools, veel software en compilers zijn gratis
-Gebruiksgemak omdat ze herprogrammeerbaar zijn, en dit in circuit

Een PIC heeft heel wat mogelijkheden aan boord. Dit uit zich al wanneer je de datasheet van een willekeurige PIC bekijkt. Deze bedraagt een paar honderd bladzijden aan informatie. Het merendeel van deze pagina's beschrijft de werking van de inwendige modules. Deze complexe en uitgebreide hardware is dan ook goed gedocumenteerd. De instructieset om deze modules aan te sturen is echter beperkt. We hebben dan ook te doen met een RISC processor*.

2.2 Microchip Technology en andere fabrikanten

Er zijn heel wat fabrikanten die IC`s vervaardigen. Bijna elke firma die halfgeleiders produceert, maakt ook IC's. In deze documentatie worden enkel de PIC's van Microchip bestudeerd. Voor de volledigheid is er hier een opsomming van de belangrijkste fabrikanten van IC's opgenomen:

-Microchip Technology: de PICmicro en dsPICreeks
-Intel: MCSreeks
-Ubicom: snelle PICklonen
-Atmel: AVR en 8051
-Dallas Maxim: lowpower 16 bits RISC
-Motorola: 68HCreeks
-Philips: 8051
-Parallax: BASIC stamps

Figuur 2: fabrikanten die C's produceren

Een hele lange tijd geleden produceerde de afdeling Microelectronics van de firma General Instruments een chip genaamd de PIC1650. Deze chip werd beschreven als een 'Programmable Intelligent Computer'. Deze component is de moeder van alle PICmicrochips. Zijn functionaliteit is vergelijkbaar met de huidige PIC16C54. Deze controller werd ingezet als randapparaat voor de CP1600microprocessoren. Daarom wordt de afkorting PIC vaak ingevuld als 'Peripheral Interface Controller'.
In het jaar 1989 werd Microchip Technology opgericht door een groepering van gedurfde kapitalisten die de afdeling verworven van General Instruments. Er is echter geen informatie te vinden hoe Microchip tegenwoordig de afkorting PIC invult.
Onlangs zijn ze gestart met een nieuwe benaming: 'PICmicro MCU', wat staat voor 'PICmicro MultiChip Unit'. De benamingen PIC en PICmicro worden zowat door elkaar gebruikt en meestal weggelaten. Overigens kondigde Microchip op 8 november 2006 de verzending van zijn 5 miljardste PIC Microcontroller aan.

2.3 Types & naamgeving

De naamgeving van een PIC is gekoppeld aan het type maar is niet altijd even eenduidig of eenvoudig om te achterhalen. Er bestaan dan ook heel wat uitzonderingen.

De volgende patronen zijn echter opmerkbaar:

hoofdtype	prefix	programma geheugen			type	revisie		temp.	freq.		behuizing
PIC	10 12 16 18	(L) (L)	F C	(R) (E)	xxx	(A) (B) …	-	(I) (E)	(04) (20)	/	PDIP SSOP …

Tabel 1: naamgeving bij PICs

Voor de 8bits MCU's wordt het hoofdtype aangeduid door de 3 letters: PIC. Er bestaan momenteel nog 2 hoofdtypes: de 16bits MCU's (ook aangeduid als PIC) en de 16bits DSC's (aangeduid als dsPIC). Deze letteraanduidingen worden echter vaak weggelaten.
De prefix van een PIC bestaat uit 2 cijfers en heeft betrekking op het volgende:

- Prefix 10: PIC's met een 12bits programmawoord en 6 of 8 pinnen
- Prefix 12: 12 en 14bits programmawoord met 8 pinnen
- Prefix 16: 12 en 14bits programmawoord met meer dan 8 pinnen
- Prefix 18: 16bits programmawoord

Vervolgens wordt het type programmageheugen aangeduid:

- Een L :duidt aan dat het om een lowvoltage variant gaat
- Type F: flash, elektronisch herprogrammeerbaar
- Type C : EPROM, slechts 1 maal programmeerbaar OTP* of met een window
- Type CR: in dit geval is de ROM voorgeprogrammeerd in de fabriek
- Type CE: in dit geval is er een aparte EEPROM aanwezig in de chip

Het type duidt alle varianten van de PIC's aan en daarmee wordt de aanwezige inwendige hardware bedoeld. Het bestaat steeds uit 3 of 4 cijfers, afhankelijk van het nieuwe IC type, maar er is geen logica in terug te vinden. Na het type volgt er eventueel een letter die de revisie van de chip aanduidt.

Hierna volgt er eventueel een temperatuursaanduiding: I staat voor Industrial en E staat voor Extended. Deze kan ook gevolgd worden door een aanduiding van maximum klokfrequentie. Deze 2 aanduidingen worden enkel vermeld wanneer er van dezelfde PIC varianten zijn qua temperatuur of klokfrequentie. Als laatste volgt er informatie over het type behuizing (package). PIC's komen in zowat alle types behuizingen voor.

Deze structuur geldt voor het overgrote deel van de PIC's maar zeker niet voor allemaal. Om maar een paar uitzonderingen op te noemen:

- PIC14000: dit is een mixedsignal PIC omdat het analoge functies aan boord heeft
- PIC16C85: heeft een EEPROM aan boord, vergelijkbaar met flash, toch staat er een C in zijn typenummer
- PIC16HV540: deze PIC heeft een ingebouwde spanningsregelaar
- De PIC17reeks waren de voorlopers van de PIC18reeks, ze zijn echter zonder succes uit productie gehaald

Natuurlijk kun je alle soorten PIC`s op veel manieren gaan catalogeren.
Hier werd geprobeerd om d.m.v. de naamgeving ze te catalogeren. Maar er zijn nog andere mogelijkheden:

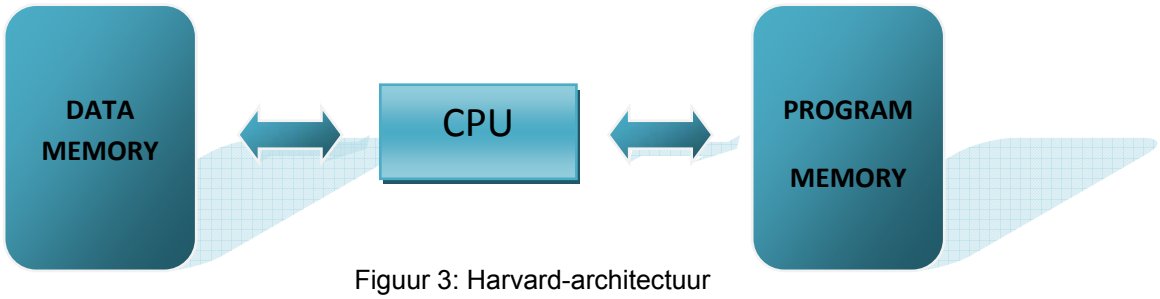
-Grootte van het programmageheugen
-Grootte van de aanwezige RAM/ROM
-Aantal I/O pinnen
-Interne mogelijkheden: ADC, timers, PWM, interne oscillator, max. klokfrequentie,…
-Interfaces: USART, USB, CAN, ethernet,…

Meestal zal een combinatie van parameters uitmaken welke PIC het meest geschikt is voor een bepaalde applicatie. In deze documentatie wordt met 2 types gewerkt. De 16F877A is een leuke PIC om mee te starten door het grote aantal pinaansluitingen. De 16F690 is gekozen om wat dieper in te gaan op de verbeterde mogelijkheden van de PWM-module. Zodra je het programmeren van 1 type PIC onder de knie hebt, is het makkelijk om over te stappen naar een ander type.

2.4 Inwendige opbouw

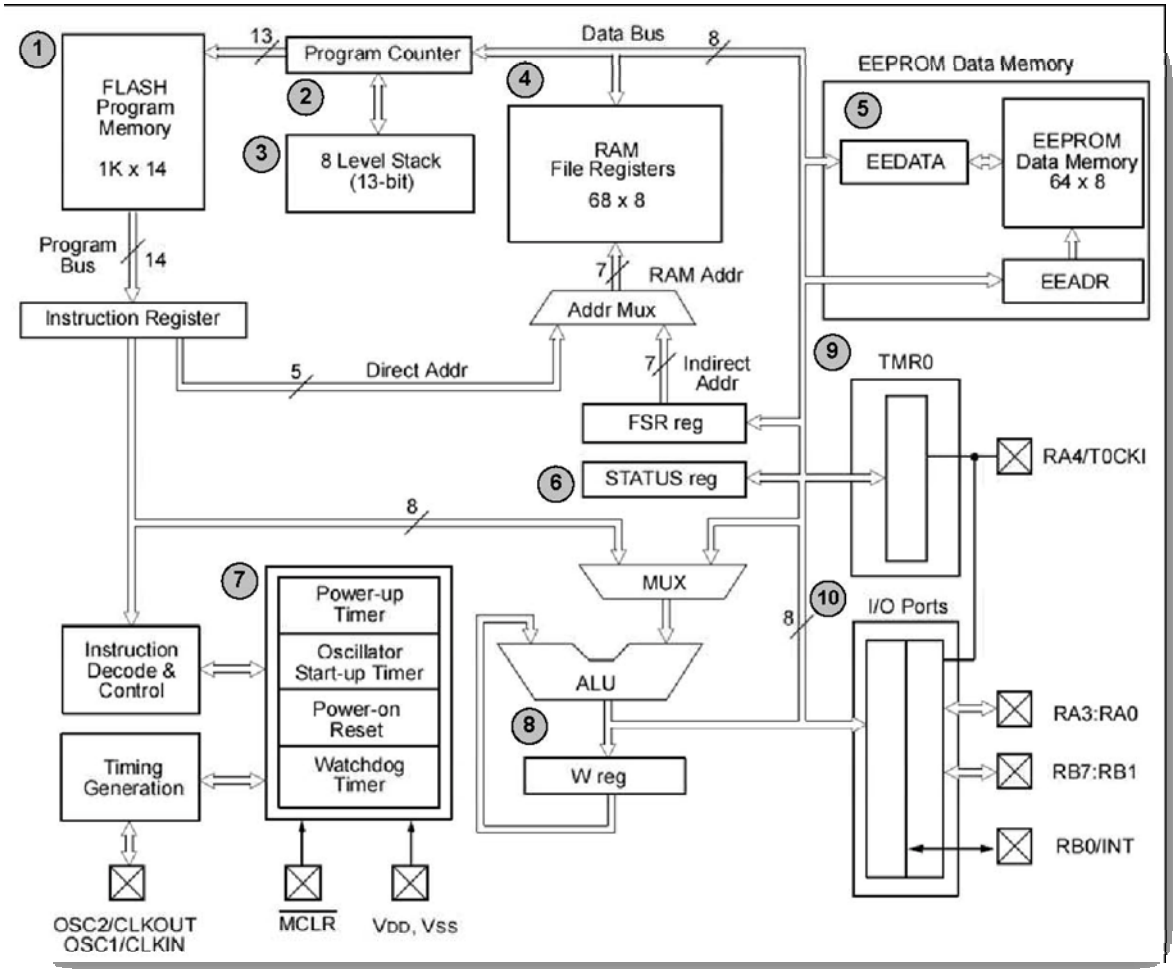
2.4.1 Architectuur

Alle PIC's volgen de Harvard-architectuur zoals in onderstaande figuur 3. Dit wil zeggen dat het geheugen voor de programma code en het geheugen voor de data gescheiden zijn. Er wordt ook geen gebruik gemaakt van pipelining, De meeste instructies zullen dus 1 klokcyclus (4 klokpulsen) duren. Er kan geen extern programmageheugen gebruikt worden bij een PIC want er is geen externe geheugenbus aanwezig. Er zijn hier echter een paar uitzonderingen op te vinden in de PIC18reeks.



Figuur 3: Harvard-architectuur

In volgende figuur is het blokdiagram van de inwendige opbouw van de PIC16F84A weergeven. Deze opbouw is voor alle PIC's gelijkaardig maar soms nog een groot stuk uitgebreid met modules of I/O-pinnen. De belangrijkste delen zijn aangeduid met cijfers en worden verder besproken.



Figuur 4: blokschema van de PIC16F84A

Op de figuur 4 is duidelijk zichtbaar dat programmacode (1) en datageheugen (4) gescheiden opgeslagen worden. We zien hier ook duidelijk dat de instructies 14-bits breed zijn. Het geheugen en de databussen zijn 8-bits breed. De instructies worden geselecteerd door middel van de programcounter (2). De programcounter is beïnvloedbaar door een aantal SFR's* die in het datageheugen aanwezig zijn. Er is tevens een hardware stack (3) van enkele niveaus diep die vanzelfsprekend dezelfde breedte heeft als de programcounter. De stack houdt terugkeeradressen bij wanneer er gesprongen wordt naar een subroutine.
De 2 belangrijkste onderdelen in dit blokdiagram zijn het W-register en de ALU (8). De data, waar bewerkingen op uitgevoerd moeten worden, worden opgeslagen in het W-register. De bewerkingen zelf worden uitgevoerd door de ALU. Het resultaat is daarna weer beschikbaar in het W-register.

2.4.2 Datageheugen: SFR & GPR

Het datageheugen (4) bevat 2 types registers:

-Special Function Registers (SFR): Deze registers bepalen de werking van de inwendige apparatuur of geven informatie over de werking weer. De configuratie van de PIC vindt dus in deze registers plaats.
-General Purpose Registers (GPR): Deze registers zijn vrije geheugenplaatsen en kunnen gebruikt worden om 8-bits variabelen op te slaan.

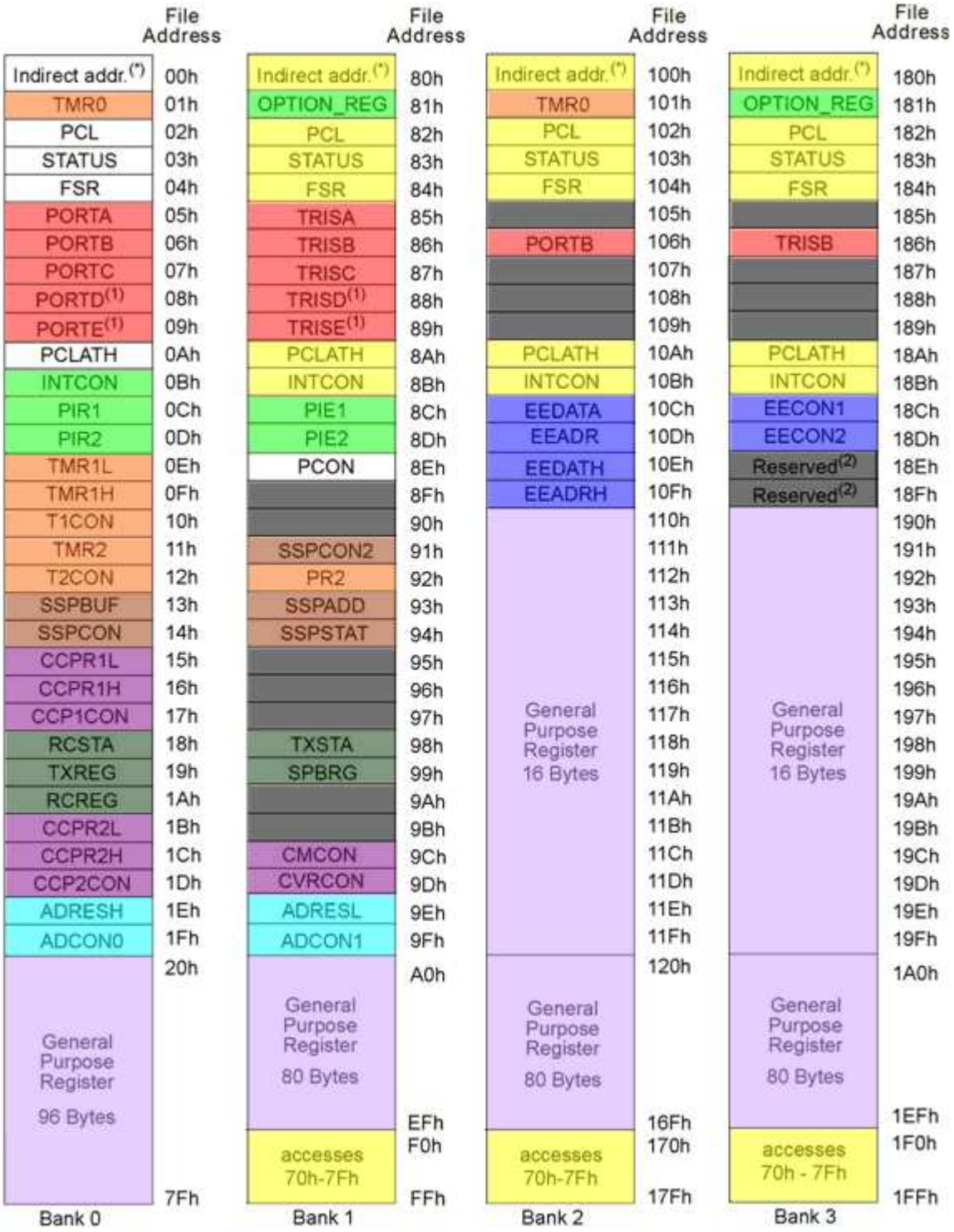
Het datageheugen van een PIC is onderverdeeld in banken. Dit wil zeggen dat eerst de juiste bank geselecteerd moet worden vooraleer een register toegankelijk is. Deze selectiebits zijn aanwezig in het STATUS-register (6), dat over alle banken toegankelijk is. Naast deze directe methode is er ook een indirecte adresseermethode, die echter nog steeds de huidige waarde van de bankselectbit(s) gebruikt. Deze methode plaatst het adres (een pointer) van het te selecteren register in het FSR (file select register). De data is dan beschikbaar in het INDF-register (indirect file register).

Merk op dat toegang tot de interne EEPROM (5) van een PIC ook via een aantal SFR's verloopt. Hier moet er dus ook indirect geadresseerd worden. De routine hiervoor is verder in deze documentatie uitgewerkt.

Een voorbeeld van een geheugenmap is op de volgende pagina opgenomen. Het gaat hier om de geheugenmap van de 16F877A. Deze heeft 4 geheugenbanken (0 t.e.m. 3). Bepaalde geheugenlocaties zijn niet geïmplementeerd of gereserveerd door het systeem (zwart).

Sommige registers overlappen alle banken (geel). Voorbeelden van overlappingen zijn: STATUS, INTCON en het laatste deel van de GPR's. Registers kunnen overlappend zijn omdat ze steeds toegankelijk moeten zijn (zoals STATUS waar de bank gewijzigd kan worden) of omdat ze snel toegankelijk moeten zijn (zoals INTCON waar de interruptwerking geregeld wordt). Een deel van het GPR-geheugen is ook overlappend. Deze variabelen zijn dus ook snel toegankelijk. Deze locaties zullen meestal gebruikt worden tijdens een ISR*.

Waarvoor elke bit uit elk register precies dient kan vanzelfsprekend teruggevonden worden in de datasheet van de PIC. Het heeft dan ook geen zin om dit over te nemen. De registers kunnen wel ingedeeld worden volgens hun functie. Veel van deze functies hebben rechtstreeks te maken met de modules die de PIC inwendig aan boord heeft. Dit is te zien op de volgende pagina.



Figuur 5: geheugenmap van de 16F877A

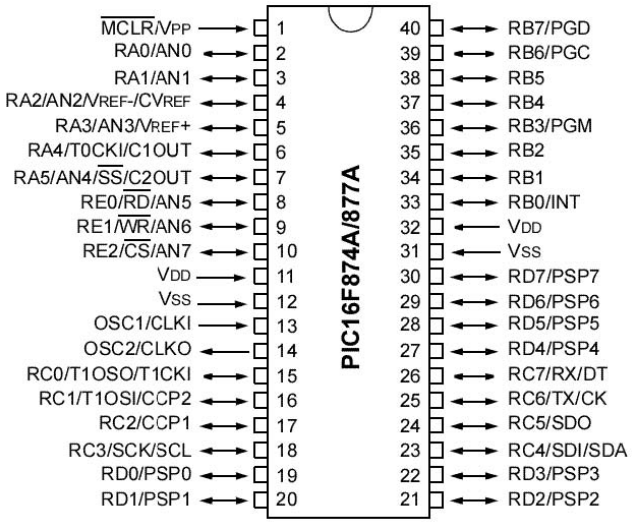
- Algemeen: wit
- I/O Poorten: rood
- Timer0/1/2: oranje
- Interrupts: lichtgroen
- CCP1/2: donkerpaars
- USART: donkergroen
- MSSP: bruin
- EEPROM: donkerblauw
- ADC: lichtblauw
- GPR: lichtpaars

2.4.3 Modules en I/O

Een PIC heeft altijd hardwaremodules aanwezig die eventueel verbonden zijn met bepaalde I/Opinnen. De 16F877 heeft er maar 2: een timer (9) en een digitale I/Oport (10). Er kunnen er echter veel meer zijn. Hieronder een lijst met wat een PIC zo allemaal aan boord kan hebben:

-Digitale I/O pinnen
-Interne klokoscillatoren
-8/16-bits timers
-Interne EEPROM
-USART
-MSSP (I²C en SPI)
-PSP (Parallel Slave Port)
-CCP: Capture, Compare en PWM
-A/D converters
-USB
-CAN
-Ethernet

Deze lijst wordt in de toekomst zeker nog uitgebreid. Door het grote aantal modules die soms aanwezig zijn in één chip worden I/Opinnen vaak gemultiplext. Tijdens de initialisatie van de PIC wordt dan daadwerkelijk bepaald wat de functie van een bepaalde pin is. Een voorbeeld daarvan zie je in de pinaansluitingen van de 16F877A in volgende Fig. 6



Figuur 6: Pinaansluiting van de 16F877A

2.4.4 Speciale functies

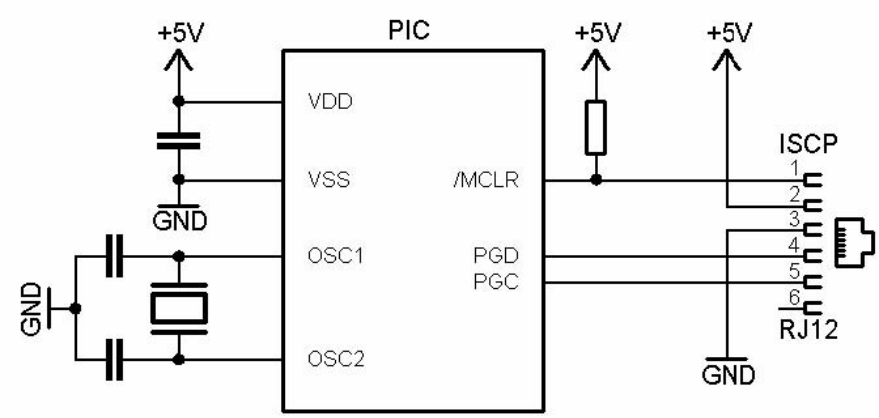
Een PIC heeft enkele speciale functies aan boord (7). Een groot deel ervan dient ingesteld te worden via het configuratiewoord van de PIC. Een opsomming van mogelijke speciale functies:

-Oscillator selectie: intern, extern, high speed of RC-kring
-Resetcondities: power-on POR*, brown-out BOR*, oscillator start up timer OST* en power-up timer PWRT*. De oorzaak van een reset kan ook gedetecteerd worden.
-Meerdere interrupts
-Watchdogtimer WDT*: controlemechanisme die de PIC kan reseten na een bepaald tijdsinterval

-Sleepmode: de PIC in een energiezuinige toestand brengen
-Codeprotectie
-Incircuit serial programming ICSP*: De PIC programmeren terwijl hij nog aanwezig is in de schakeling
-Lowvoltage programming LVP*: de PIC programmeren d.m.v. een lage spanning (een eenvoudigere programmer kan gebruikt worden)
-Incircuit debugger: de PIC kan gedebugged worden in de schakeling.

2.5 Minimale randcomponenten

De volgende figuur toont een aantal componenten rondom de PIC die noodzakelijk zijn.



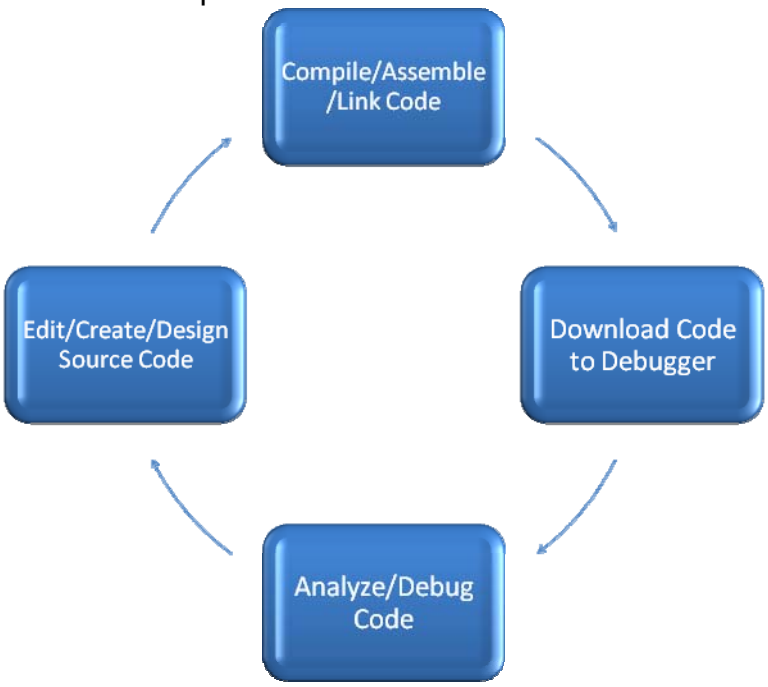
Figuur 7: randcomponenten van een PIC

-Een voeding: Zonder spanning kan een PIC vanzelfsprekend niet werken. De meeste PIC's werken op 5V, sommige op 3,3V. Een ontkoppelcondensator is noodzakelijk.
-Een klok: meestal wordt een kristal gebruikt maar een RC-kring is ook mogelijk. Sommige PIC's hebben intern een RC-kring aanwezig, deze zijn echter temperatuursafhankelijk
-Resetcircuit: een pull-up weerstand volstaat om de PIC te resetten bij het opkomen van de voedingsspanning. Er kan ook een schakelaar naar massa bijgeplaatst worden.
-Een RJ12-connector die toelaat om de PIC in zijn schakeling te programmeren is ook aangewezen. Deze aansluiting is gestandaardiseerd door Microchip en neemt slechts 5 draden in. Via de /MCLR* pin wordt de programmeerspanning aangebracht. PGD* en PGC* zijn respectievelijk de data en de klok. De voedingsspanning dient ook aangesloten te worden en kan eventueel geleverd worden door de programmer.

2.6 Het programmeerproces

Het programmeerproces start bij het kiezen van een programmeertaal. Er kan gekozen worden voor een highlevel language zoals: C, BASIC, PASCAL of JAL. Kies je echter voor de assembleertaal, dan sta je een stuk dichter bij de hardware. Assembler wordt dikwijls ook machinecode genoemd. Alle PIC's hebben een gereduceerde instructieset RISC*. Concreet varieert het aantal instructies tussen de 35 à 70. In een korte tijdspanne zul je een uitgebreider programma met meer functionaliteit kunnen maken met een hogere programmeertaal. Een assembleerprogramma zal, indien voldoende tijd is genomen om het te programmeren, altijd sneller en kleiner in omvang zijn en efficiënter werken.

De volgende stap is natuurlijk het programmeren zelf. De datasheet van de PIC met de beschrijving van de hardware en de instructieset zijn onmisbaar in deze stap. Een programma zal nooit in 1 keer van begin tot einde geschreven worden. Het is veeleer een opbouwend proces: een routine of een deel van de code schrijven, controleren, indien nodig wijzigen en implementeren in het geheel. Dit wordt mooi geïllustreerd in de 'design circle' van Microchip.



Figuur 8: the Microchip design circle

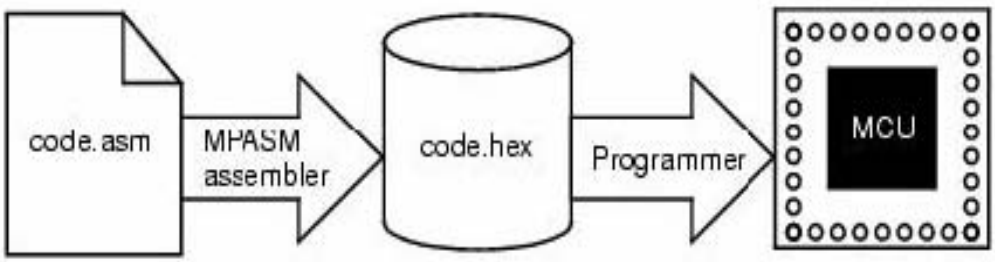
Vervolgens wordt het programma gecompileerd naar machinecode waarna het gesimuleerd kan worden of geprogrammeerd in de PIC. Het volledige programmeerproces kan plaats vinden in de MPLAB IDE. Deze 'Integraded Development Enviroment' bevat zowel: een editor, een assembler, een simulator en heel wat ondersteuning voor debuggers, emulators en programmers. Voor een handleiding omtrent MPLAB verwijs ik naar Microchip MPLAB IDE user's guide.

De laatste stap is de PIC programmeren. Deze stap wordt ook dikwijls ‘burnen’ genoemd omdat de code als het ware in de PIC gebrand wordt. Hiervoor is een hardware tool nodig die we ook een programmer noemen. Er bestaan heel wat varianten op het vlak van programmers en hun mogelijkheden (en kostprijs). Sommige kunnen naast programmen ook debuggen zoals de ICD2 van Microchip die hieronder afgebeeld is. Zie Figuur 9.

Voor meer informatie omtrent deze programmer verwijs ik naar de Microchip website: MPLAB PICKIT2 of ICD2 InCircuit Debugger User’s Guide. Er zijn heel wat lowcost varianten en klonen beschikbaar maar de meeste programmers zijn voorzien om incircuit te programmeren.

Figuur 9: MPLAB pickit2 & ICD2 programmer

Deze stappen worden mooi weergeven in de volgende figuur 10. Namelijk het omzetten van een assembler programma naar machinecode, om het vervolgens in de PIC te programmeren. Ik ga in deze documentatie niet in op locatieonafhankelijke code en linkers om het niet nodeloos ingewikkeld te maken. Deze mogelijkheden worden voornamelijk gebruikt bij hogere programmeertalen.



Figuur 10: Het programmeerproces

2.7 Assembleerprogramma

Een programma geschreven in assembler heeft zowat altijd dezelfde opbouw. Daarom gebruiken we een sjabloon om te starten aan een nieuw programma. Het volgende stuk code is de sjabloon die ook gebruikt wordt in de andere programma's in deze scriptie. Hierin komen voornamelijk 'directives' voor en bijna geen instructies. Directives geven instructies door aan de compiler. Op de volgende pagina worden deze wat nader uitgelegd.

```
      TITLE      "template"
      PROCESSOR  16F877

      #include <p16f877a.inc>

      __CONFIG _HS_OSC & _DEBUG_OFF & WRT_OFF & _CPD_OFF & _LVP_OFF &
      _BODEN_OFF & _PWRT_OFF & _WDT_OFF

      ;Disable message: Register in operand nog in bank0
      ERRORLEVEL -302

;RAM-variablen
      CBLOCK 0x20
            var1
            var2
      ENDC

;vectors
      ORG   0x0000     ;reset
      GOTO  init
      ORG   0x0004     ;isr
      GOTO  isr

;includes subroutines
      #include <subroutines.inc>

;initialiseren
Init


;start programma
Start

      GOTO start


;interrupt routine
Isr

      RETFIE           ;return from interrupt

      END
```

Code 1: assembler template

1 TITLE
De titel van het programma.

2 PROCESSOR
Deze directive geeft aan welke PIC er gebruikt wordt.

3 #include <p16f877a.inc>
De include-directive zorgt ervoor dat de code aanwezig in een ander bestand hier ingevoegd wordt. Dit heeft hetzelfde effect al zou deze code echt op deze plaats staan. Op deze manier kunnen ook subroutines uit een ander bestand beschikbaar gesteld worden.

4 __CONFIG
Hiermee worden de configuratiebits van de PIC ingesteld. Dit is een combinatie van verschillende instellingen. Onder andere de oscillator wordt hier gekozen.

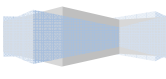
5 ERRORLEVEL 302
Dit is geen noodzakelijke directive maar heeft als doel om de berichten die de assembler genereert omtrent bankswitching te onderdrukken. Deze berichten zijn geen errors of warnings maar messages.

6 CBLOCK
Door middel van CBLOCK kunnen we een naam geven aan bepaalde locaties in het RAM geheugen. Dit gebruiken we dus om GPR's een naam te geven en gemakkelijk te kunnen gebruiken in ons programma. Dit kan ook verwezenlijkt worden d.m.v. de directive*: EQU.

7 commentaar
Er kan steeds commentaar in het programma gevoegd worden zolang er maar een puntkomma voor staat.

8 ORG
Via ORG kan code op een bepaalde geheugenlocatie geplaatst worden. De code die volgt na deze instructie start dan op dit adres. De locaties die zeker ingevuld moeten worden, zijn de resetvector op aders 0x0000 en de resetvector op locatie 0x0004.

9 label
Een label wordt steeds in de eerste kolom geplaatst. Deze labels kunnen dan aangeroepen worden d.m.v. een GOTO of een CALL. Instructies worden steeds in de 2e kolom geplaatst. Operanden hoeven niet in de 3e kolom geplaatst te worden, het zorgt echter voor een grotere leesbaarheid van de code. Labels zijn hoofdlettergevoelig.



10 END
Hiermee wordt het einde van het programma aangeduid.

3 Projectopdracht

3.1 “Knight rider”

Hieronder wordt een project beschreven dat kan worden gebruikt om de basis van het programmeren van het IC te oefenen
Hieronder is een lijst met componenten en de daar bijhorende waardes te zien. Ook zijn er schema's van het project getekend om het project makkelijker te maken. Het project kan eerst opgebouwd worden op een breadboard. Na de lijst van componenten volgt een korte beschrijving van de eigenschappen van de componenten en de belangrijkste aandachtspunten. Daarna
worden er beschreven hoe je een externe programmer op de IC kan aansluiten.

Componentenlijst.

Component	Aantal	Waarde	Bestel code
BreadBoard	1	NVT	DAE-001-0001
Draadbrug	11	NVT	DAE-001-0001
LED’s	8	NVT	DAE-002-0001 DAE-002-0002 DAE-002-0003 DAE-002-0004 DAE-002-0005 DAE-002-0006 DAE-002-0007 DAE-002-0010
Weerstanden	8	1KΩ	DAE-003-0010
Condensator	1	0,33uF of 10uF	DAE-004-XXXX
IC 16F628/16f648	1	NVT	DAE-005-XXXX
Condensator	1	0,1uF	DAE-004-XXXX
LM7805	1	5V	DAE-006-XXXX
Pickit2	1	NVT	DAE-011-0001

Tabel 2 Componentenlijst voor het opbouwen van project “Knight rider”.

3.2 Aansluiting

3.2.1 Theorie voeding.

Regulator of ook wel stabilisator genoemd.
We kunnen een regulator vergelijken met een cruise control in een auto of de automatische piloot in een vliegtuig. De cruise control houdt de snelheid constant, onafhankelijk van de rijomstandigheden.

We onderscheiden:

- Regulators met een vaste uitgangsspanning;
- Regulators waarbij we de uitgangsspanning met weerstanden kunnen instellen.

We zullen de vaste uitgangsspannings regulator nader toelichten, omdat deze op deze microcontrole voeding van toepassing is.

Deze regulator heeft drie aansluitingen:

- Een ingang;
- Een uitgang;
- Een massa-aansluiting. (ground)

Een bekende serie regulators is de 78XX-familie. Deze regulators hebben een vaste uitgangsspanning. De grootte wordt aangegeven door de laatste twee cijfers van de typeaanduiding.
De typeaanduiding kan ook nog een letter hebben. Deze letter geeft informatie over de maximale uitgangsstroom.
De 78XX-familie zorgt voor een positieve uitgangsspanning, de 79XX-familie voor een negatieve uitgangsspanning. In tabel Fig.3 zien we een overzicht van een (positieve) 5 V-regulator-serie. In tabel Fig.4 zien we een overzicht van een (negatieve) 5 V-regulator-serie

Tabel Fig. 3 Positieve 5 V regulator				
78L05	78M05	7805	78H05	78P05
5 V	5 V	5 V	5 V	5 V
100 mA	500 mA	1,5 A of 1 A	5 A	10 A

Tabel Fig. 4 Negatieve 5 V regulator				
79L05	79M05	7905	79H15	79P05
-5 V	-5 V	-5 V	-5 V	-5 V
100 mA	500 mA	1,5 A of 1 A	5 A	10 A

Tabel 3 en 4 uitgangspanning en uitgangstroom.

Thermische beveiliging

Regulators zijn intern thermisch beveiligd tegen overbelasting. Als de vermogensdissipatie te groot wordt, verlaagt de regulator de uitgangsstroom.
We kunnen een beperkt aantal regulators krijgen met een vaste uitgangsspanning.
Zie tabel Fig. 5.

Tabel Fig. 5 Regulators met vaste uitgangsspanning

Positieve

Regulator	7805	7806	7809	7812	7815	7818	7824
(V)	2	2	2	2	2	2	2
(V)	35	35	35	35	35	35	35

Negatieve

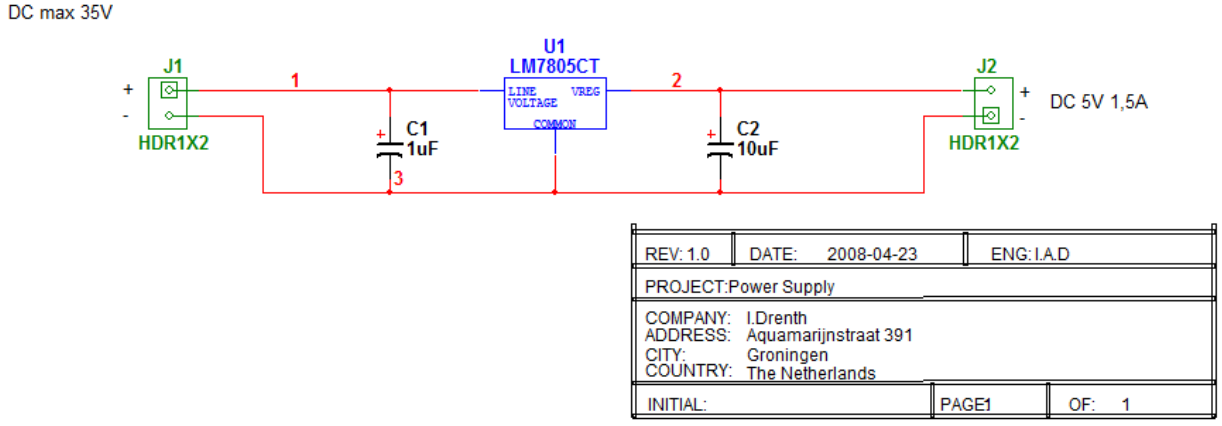
Regulator	7905	7906	7909	7912	7915	7918	7924
(V)	-2	-2	-2	-2	-2	-2	-2
(V)	-35	-35	-35	-35	-35	-35	-35

Tabel 5 regulator met vaste uitgangsspanning.

We bouwen eerst een gestabiliseerde voeding op.
Je heb het volgende nodig. Zie tabel 6.

Component	Aantal	Waarde	Bestel code
Condensator C2	1	0,33uF of 10uF	DAE-004-XXXX
Condensator C1	1	0,1uF	DAE-004-XXXX
LM7805	1	5V	DAE-006-XXXX

Tabel 6 Benodigde componenten.



Figuur 11: Het gestabiliseerde voeding.

Maak het volgende schema op het breadboard of gaatjes print. Zie Fig. 11
Sluit op de ingang maximaal 35V gelijkspanning aan. Controleer via een voltmeter op de uitgangspanning dat er 5V uitkomt. Dit is eigenlijk een heel belangrijke punt.

3.2.2 Theorie led.

LED (Light Emitting Diode)
Een led is een elektronische component, een diode die licht uitzendt als er een stroom in doorlaatrichting doorheen wordt gestuurd.

De kleur van het opgewekte licht is afhankelijk van de aard van de materialen waaruit de led is opgebouwd. Dit is ook de reden dat een led met een lange golflengte een lagere doorlaatspanning heeft dan een met een korte golflengte, bijvoorbeeld rood 1,5 V en blauw 3,5 V. Omdat de spanning over de led ook een beetje stijgt bij een grotere stroom zal de kleur iets naar een kortere golflengte opschuiven, een blauwe led zal bij lage stroom meer groenig schijnen en een rode led wordt (heel even) geel bij zoveel stroom dat hij stuk gaat.

De ontwikkeling van de blauwe led heeft lang op zich laten wachten. Pas in de jaren '90 zijn er betaalbare uitvoeringen met een redelijke helderheid op de markt. Inmiddels doet de blauwe led nauwelijks onder voor de groene. Met het beschikbaar komen van de blauwe led is volledige RGB kleurmenging mogelijk geworden.

Witte leds werden oorspronkelijk gemaakt door met behulp van een UV-led een fluorescerende laag ('fosfor') te belichten. Inmiddels zijn er ook varianten die direct wit licht uitstralen. Maar ook RGB-leds kunnen wit licht uitzenden.

Vrijwel alle afstandbedieningen voor elektronische apparatuur zenden hun commando over met behulp van IR-leds. Deze kunnen een relatief hoog vermogen verwerken. Infrarood leds worden ook veel toegepast als geïntegreerde zender in optokoppelaars (Eng. optocoupler), veiligheidscomponenten waarbij de zendende zijde en de ontvangende zijde optisch vast verbonden zijn maar elektrisch onderling deugdelijk geïsoleerd zijn. IR-leds kunnen ook toegepast worden als hulpverlichting voor analoge en digitale video-camera's met "nachtopname" aangezien de CCD-sensor ook gevoelig is voor de golflengte van een IR-led.

3.2.3 Led weerstand berekenen.

In elektronisch opzicht zijn leds en andere halfgeleiderdiodes interessante componenten omdat er een nagenoeg constante spanningsval over de aansluitingen optreedt, anders dan bij ohmse weerstanden.

Een led mag daarom nooit zonder meer op een spanningsbron worden aangesloten. Er dient altijd een stroombegrenzer aanwezig te zijn, zoals een transistor of een eenvoudige weerstand, omdat een led in feite een diode is. Over de led zal een spanning vallen, afhankelijk van het type led zo'n 1,1 V voor infrarode led's tot wel 3,5 V bij witte en blauwe led's.

De standaardstroom door een led is 20 mA continu, maar de meeste leds kunnen 10-30 mA verwerken. Het is overigens heel goed mogelijk om pulsvormige stromen tot wel 1 A te gebruiken als de gemiddelde stroom maar binnen de veilige grenzen blijft. In het geval van constante gelijkstroom laat de grootte van de stroombegrenzende weerstand zich als volgt berekenen:

I: stroomsterkte in amperes
Us: voedingsspanning
Ui: werkspanning van de led
R: voorschakelweerstand in ohm

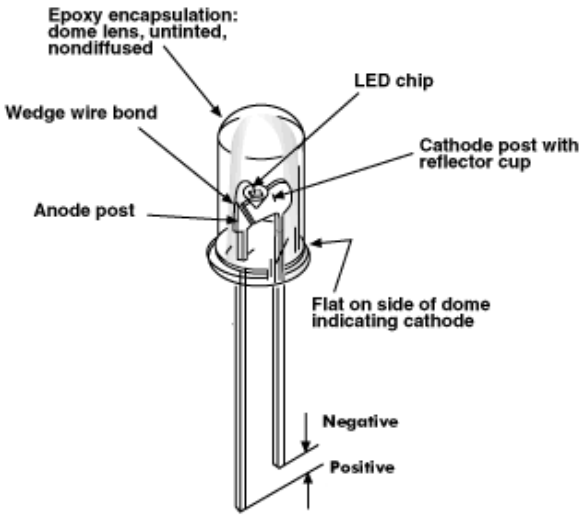
Het invullen van de waarden voor een blauwe power led: 20 mA, 1,6 V werkspanning en 5 V voedingsspanning levert op:

$$\frac{Us - Ui}{I}$$

$$\frac{5V - 1.6V}{0.02A} = \frac{3.4V}{0.02A} = 170\Omega$$

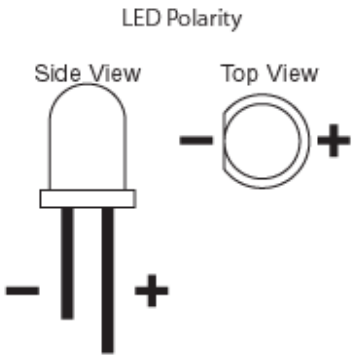
Hierbij is 180 Ω uit de E12-reeks de dichtstbijzijnde waarde.

Diode is polair gevoelig qua stroom doorvoer. De led heeft een Anode en een Kathode in de behuizing.



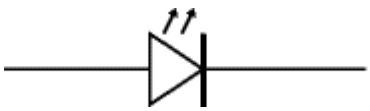
Figuur 12: Led zijkant view.

De Anode kant is altijd + en wordt bij licht leds met een langer pootje aangegeven. De Kathode is altijd de – en wordt met een korter pootje aangeduid in Figuur 13. Waarschuwing!!! Bij IR-diode is dit net anders om qua + en -.



Figuur 13: Led zijkant view.

In elektronische schema`s word er een speciale LED symbool gebruikt zoals in Figuur 14.1 De Anode en Kathode kant kan je herkennen aan de letter K van Kathode. Figuur 14.2.



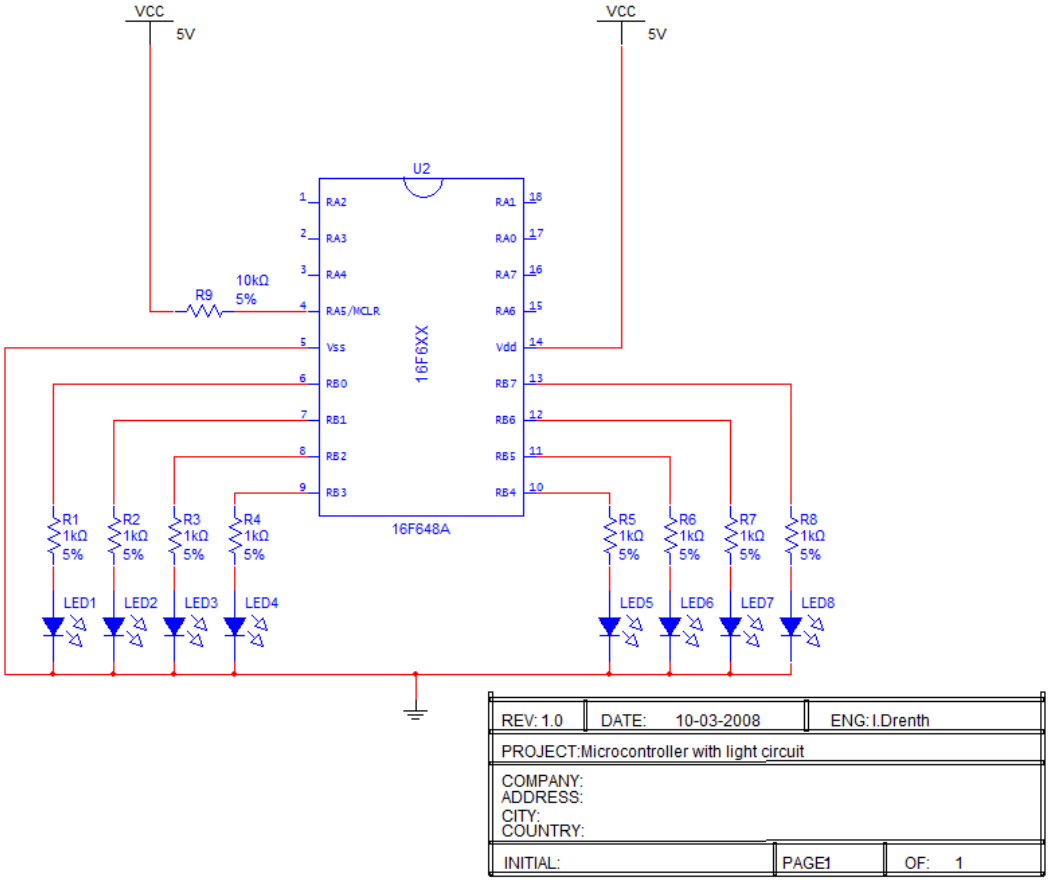
Figuur 14.1: Led symbool.

Figuur 14.2: Led symbool herkenen met K.

De pijlen die van de driehoek af wijzen geven aan dat het een lichtgevende led is. Als de pijltjes naar de driehoek wijzen, hebben we te maken met een licht opnemer, ook wel LDR genoemd.

3.3 Hoofdschema Knight rider.

Bouw nu het schema op wat in figuur 15 is weergegeven.



Figuur 15: Microcontroller met licht schema.

Bij het opbouwen moet men controleren of alles goed is aangesloten.
Neem anders het schema over op papier en streep per lijntje een streepje door het lijntje dat je hebt aangesloten. Controleer eventueel met een multimeter met geleiding stand of de aansluiting ook werkelijk aangesloten is.

3.4 Programmeren in C

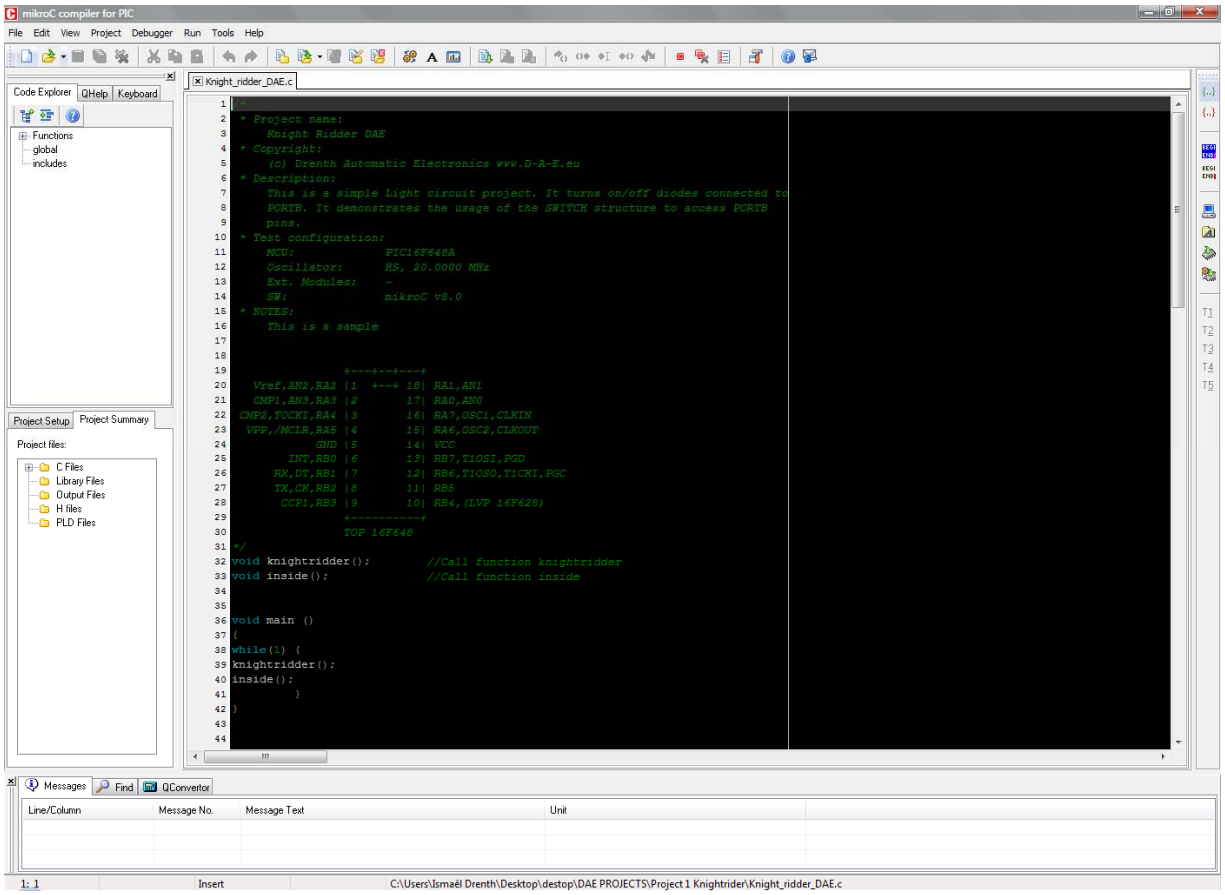
De software is geschreven in de C variant Mikro C, er is hiervoor gekozen omdat C bij grotere programma's overzichtelijker en krachtiger dan Basic. Verder is er voor C gekozen omdat je hierover veel informatie kan vinden op internet. Er is voor Mikro C gekozen omdat dit een C variant voor PIC microcontrollers is en deze een aantal handige bibliotheken heeft die duidelijke in de help file staan omschreven.

3.4.1 Installeren Mikro C

Mikro C in verschillende versies te downloaden op internet dat is verdeeld in 3 progamma talen.

- C
- Basic
- Pascal

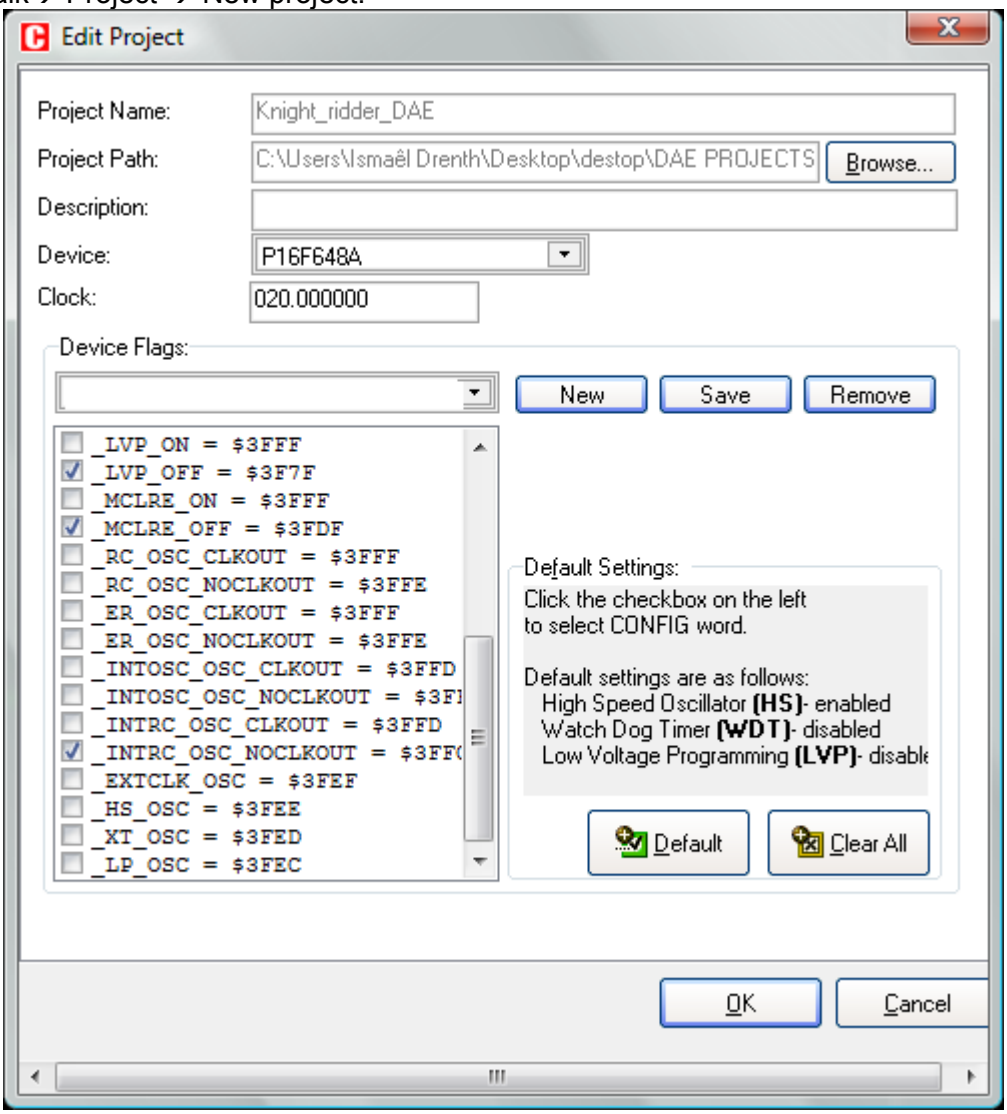
Wij gaan de C variant gebruiken van MicroE.
Download van www.D-A-E.eu → forum → Software.



Figuur 16: Mikro C programma.

3.4.2 Instellen van PIC in MikroC

Start een nieuw Project :
Takenbalk→ Project → New project.



Figuur 17: Mikro C PIC instellingen.

Stel het volgende in. (Figuur 17)
Project Name: Knight_rider_DAE
Project Path: X:\ (zorg dat je alles in 1 map houdt. Anders krijg je compileer problemen)
Description: Kan je text bij doen voor toepassing of versie.
Device: 16F648A (Dit geldt alleen bij dit project, de IC)
Clock: 20.000000 (Dit gaan we intern in de IC activeren via een RC netwerk)
Dit OSC heeft geen juiste waarde als de temperatuur varieert in de omgeving.
Device Flags: _LVP_OFF, MCLRE_OFF en _INTRC_OSC_NOCLKOUT (Vink deze combinatie)
Selecteer → OK

3.5 Programma in C geschreven

Hieronder staat de code die in de 16F648A zit, ik heb er voor gekozen om de code zo te laten als deze is geschreven en niet in volgorde van uit voeren. Hiervoor is gekozen omdat er geen vaste volgorde van uitvoeren is. In de uitleg wordt dan soms ook van de ene naar de andere plek in de code gesprongen.

Als eerste staan de declaraties van alle functies in de code, in het groen staat als commentaar achter elke declaratie wat de functie doet. In rood staat het hoofdprogramma en in blauw staan de functies.

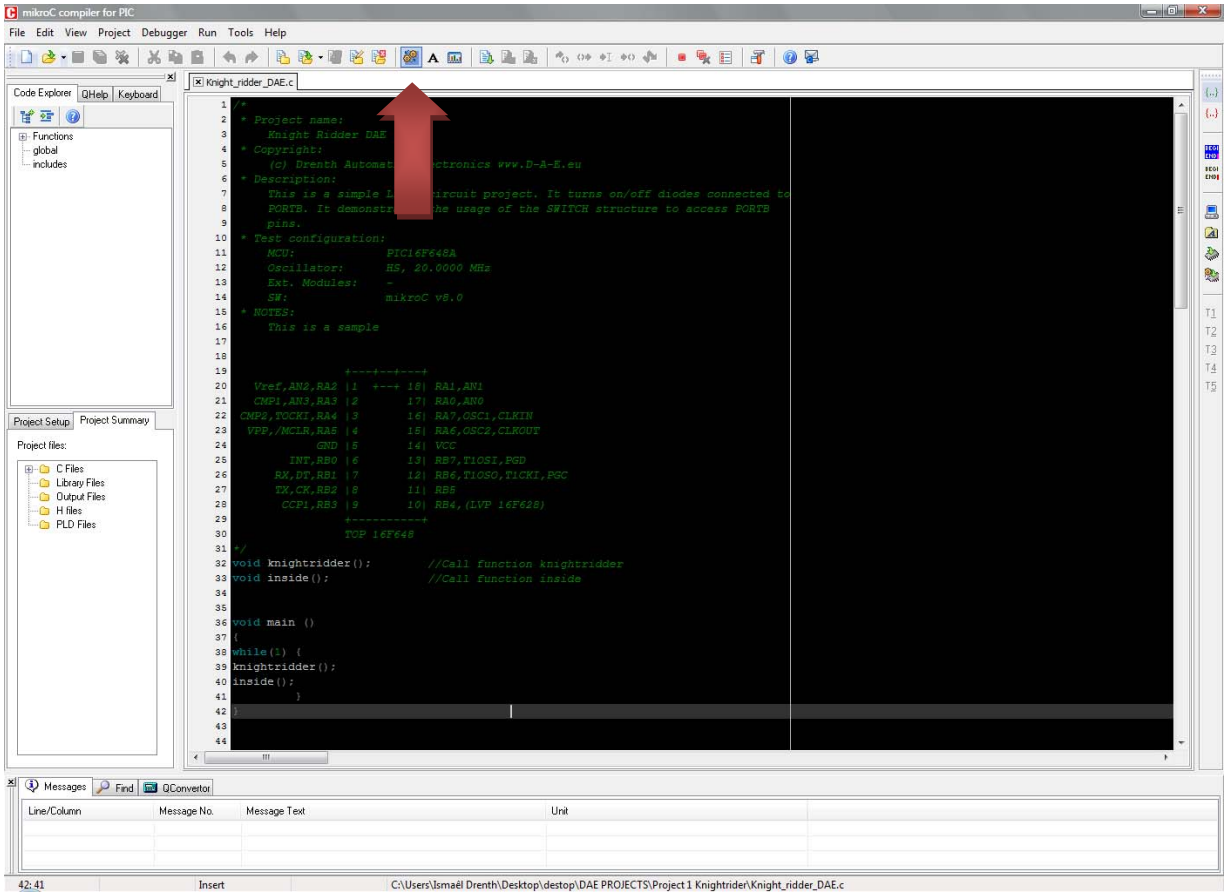
```
/*
 * Project name:
     Knight Rider DAE
 * Copyright:
     (c) Drenth Automatic Electronics www.D-A-E.eu
 * Description:
     This is a simple Light circuit project. It turns on/off diodes connected to
     PORTB. It demonstrates the usage of the SWITCH structure to access PORTB
     pins.
 * Test configuration:
     MCU:            PIC16F648A
     Oscillator:     HS, 20.0000 MHz (Dit is intern RC netwerk)
     Ext. Modules:    -
     SW:             mikroC v8.0
 * NOTES:
     This is a sample
                          +---+--+---+
        Vref,AN2,RA2 |1  +--+ 18| RA1,AN1
      CMP1,AN3,RA3 |2        17| RA0,AN0
   CMP2,TOCKI,RA4 |3         16| RA7,OSC1,CLKIN
     VPP,/MCLR,RA5 |4         15| RA6,OSC2,CLKOUT
                GND |5        14| VCC
            INT,RB0 |6         13| RB7,T1OSI,PGD
          RX,DT,RB1 |7         12| RB6,T1OSO,T1CKI,PGC
          TX,CK,RB2 |8         11| RB5
           CCP1,RB3 |9         10| RB4,(LVP 16F628)
                          +-------------+
                         TOP IC 16F648  */
void knightrider();              // Declareren van de functie knightrider
void inside();                   // Declareren van de functie inside

void main ()                     //Hoofdprogramma loop
{                                //Begin Hoofdprogramma loop
while(1) {                       //Begin van while loop
knightrider();                   //Oproepen functie knightrider
inside();                        //Oproepen functie inside
```

```
        }                       //Eind van while loop
}                               //Eind Hoofdprogramma loop
//#########################################################################
void knightrider()
{                               // Knight rider loop
TRISB = 0;                      // Instellen port Output  (Tri-State) registers
PORTB = 0;                      // Start alles op Port B laag
while(1){                       // While loop
PORTB.F0 = 1;                   // Port B0 is hoog (Led1 aan)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F1 = 1;                   // Port B1 is hoog (Led2 aan)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F2 = 1;                   // Port B2 is hoog (Led3 aan)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F3 = 1;                   // Port B3 is hoog (Led4 aan)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F4 = 1;                   // Port B4 is hoog (Led5 aan)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F5 = 1;                   // Port B5 is hoog (Led6 aan)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F6 = 1;                   //Port B6 is hoog (Led7 aan)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F7 = 1;                   //Port B7 is hoog (Led8 aan)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F0 = 0;                   //Port B0 is laag (Led1 uit)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F1 = 0;                   //Port B1 is laag (Led2 uit)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F2 = 0;                   //Port B2 is laag (Led3 uit)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F3 = 0;                   //Port B3 is laag (Led4 uit)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F4 = 0;                   //Port B4 is laag (Led5 uit)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F5 = 0;                   //Port B5 is laag (Led6 uit)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F6 = 0;                   //Port B6 is laag (Led7 uit)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
PORTB.F7 = 0;                   //Port B7 is laag (Led8 uit)
Delay_ms(100);                  // Tussen stop van 100ms voor de volgende stap
break;                          // Stop the While
    }                           // While loop stoppen
}                               // Knight ridder loop stoppen
//#########################################################################
```

```
//#########################################################################
void inside()
{
TRISB = 0;                  // Instellen port Output  (Tri-State) registers
PORTB = 0;                  // Start alles op Port B laag
while(1){                   // While loop
PORTB.F0 = 1;               //Port B0 is hoog (Led1 aan)
PORTB.F1 = 0;               //Port B1 is laag (Led2 uit)
PORTB.F2 = 0;               //Port B2 is laag (Led3 uit)
PORTB.F3 = 0;               //Port B3 is laag (Led4 uit)
PORTB.F4 = 0;               //Port B4 is laag (Led5 uit)
PORTB.F5 = 0;               //Port B5 is laag (Led6 uit)
PORTB.F6 = 0;               //Port B6 is laag (Led7 uit)
PORTB.F7 = 1;               //Port B7 is hoog (Led8 aan)
Delay_ms(100);
PORTB.F0 = 1;               //Port B0 is hoog (Led1 aan)
PORTB.F1 = 1;               //Port B1 is hoog (Led2 aan)
PORTB.F2 = 0;               //Port B2 is laag (Led3 uit)
PORTB.F3 = 0;               //Port B3 is laag (Led4 uit)
PORTB.F4 = 0;               //Port B4 is laag (Led5 uit)
PORTB.F5 = 0;               //Port B5 is laag (Led6 uit)
PORTB.F6 = 1;               //Port B6 is hoog (Led7 aan)
PORTB.F7 = 1;               //Port B7 is hoog (Led8 aan)
Delay_ms(100);
PORTB.F0 = 1;               //Port B0 is hoog (Led1 aan)
PORTB.F1 = 1;               //Port B1 is hoog (Led2 aan)
PORTB.F2 = 1;               //Port B2 is hoog (Led3 aan)
PORTB.F3 = 0;               //Port B3 is laag (Led4 uit)
PORTB.F4 = 0;               //Port B4 is laag (Led5 uit)
PORTB.F5 = 1;               //Port B5 is laag (Led6 uit)
PORTB.F6 = 1;               //Port B6 is hoog (Led7 aan)
PORTB.F7 = 1;               //Port B7 is hoog (Led8 aan)
Delay_ms(100);
PORTB.F0 = 1;               //Port B0 is hoog (Led1 aan)
PORTB.F1 = 1;               //Port B1 is hoog (Led2 aan)
PORTB.F2 = 1;               //Port B2 is hoog (Led3 aan)
PORTB.F3 = 1;               //Port B3 is hoog (Led4 aan)
PORTB.F4 = 1;               //Port B4 is hoog (Led5 aan)
PORTB.F5 = 1;               //Port B5 is hoog (Led6 aan)
PORTB.F6 = 1;               //Port B6 is hoog (Led7 aan)
PORTB.F7 = 1;               //Port B7 is hoog (Led8 aan)
Delay_ms(100);
PORTB.F0 = 0;               //Port B0 is laag (Led1 uit)
PORTB.F1 = 0;               //Port B1 is laag (Led2 uit)
PORTB.F2 = 0;               //Port B2 is laag (Led3 uit)
PORTB.F3 = 0;               //Port B3 is laag (Led4 uit)
PORTB.F4 = 0;               //Port B4 is laag (Led5 uit)
```

```
PORTB.F5 = 0;          //Port B5 is laag (Led6 uit)
PORTB.F6 = 0;          //Port B6 is laag (Led7 uit)
PORTB.F7 = 0;          //Port B7 is laag (Led8 uit)
break;                 // Stop the While
    }                  // While loop stoppen
}                      // Inside loop stoppen
```

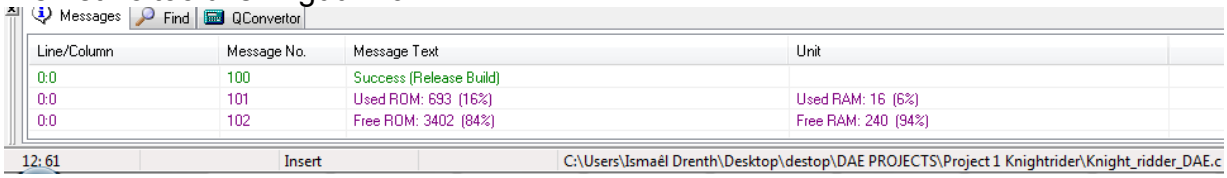
3.7 Compileren met Mikro C



Figuur 18: Compiler.

Als men klaar is met programmeren.
Geef dan het commando CTRL+F9 of de aangewezen rode pijl in Figuur 18.

Als het voltooid is. Figuur 19.

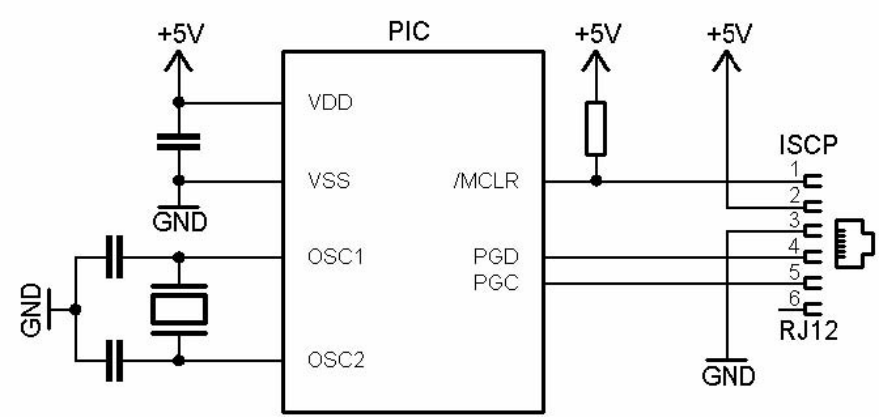


Figuur 19: Met succes gecompileerd.

Na een succes (Released build) wordt aangetoond dat er een Hex dump is gemaakt. (Figuur 19).
Zo niet, dan wordt er een foutmelding gegeven.
Dubbel klik op de fout en hij verwijst zich automatisch naar de fout.

3.8 Programmer aan de PIC aansluiten

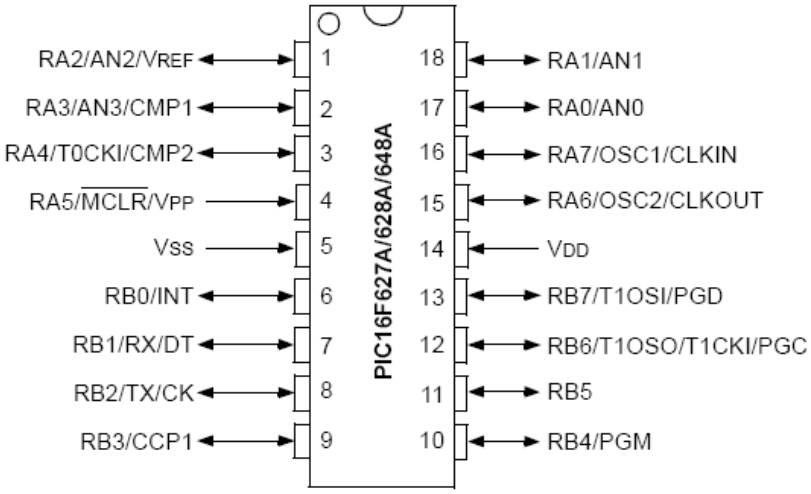
De aansluiting van de PICKIT2 of ICD2 .



Figuur 20 aansluiting voor de IC.

De weerstand op de MCLR is niet nodig voor het project Knight rider.

PDIP, SOIC

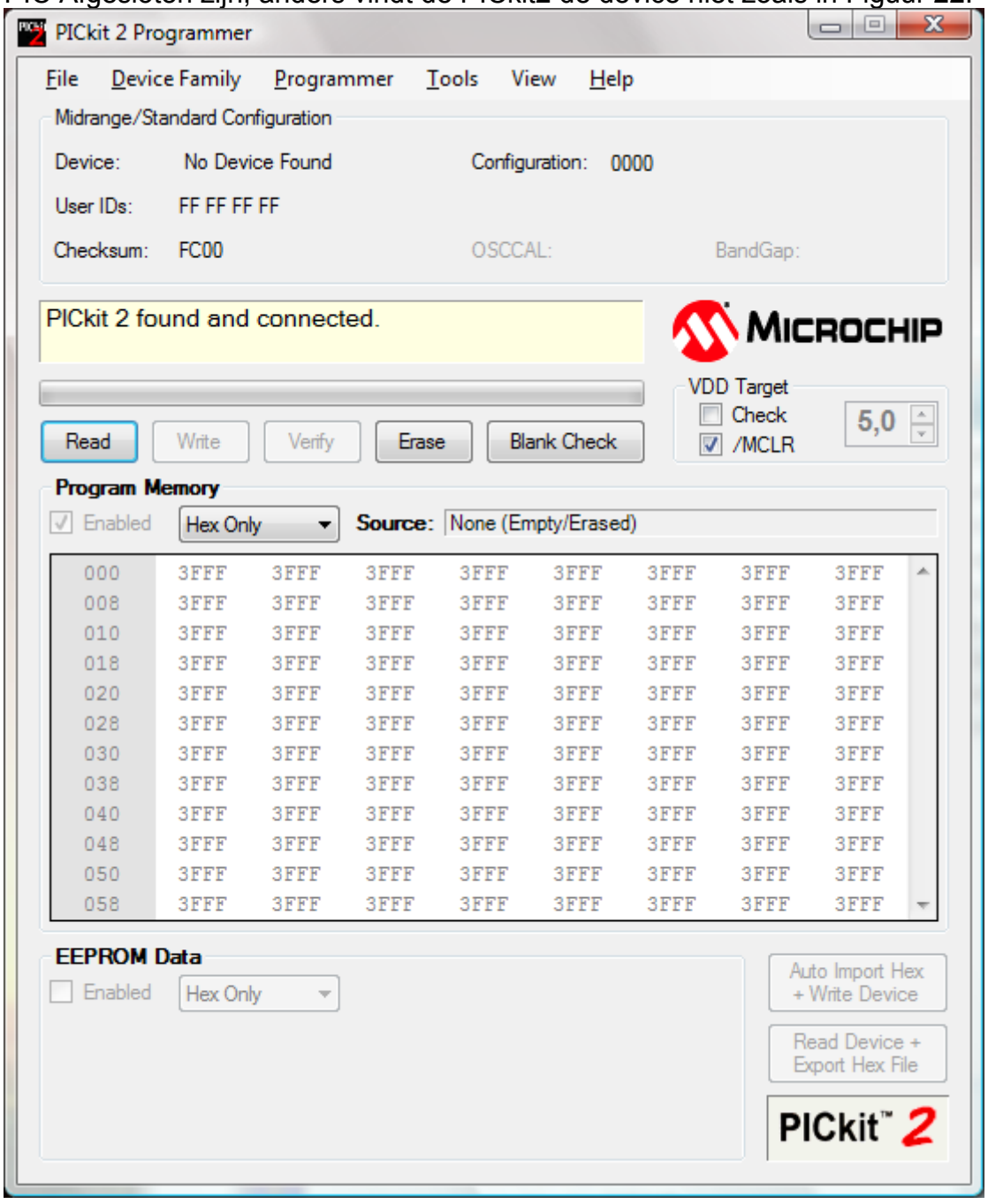


Figuur 21 PDIP uitvoering.

Elke PIC heeft zijn eigen dip aansluiting.
Voor het project gebruiken we de PIC 16F648A Figuur 21.
De aansluiting met de compiler is voor de 16F648A vanaf de ISCP*:
Pin 1 MCLR → Pin 4 PIC 16F648
Pin 2 5V → Pin 14 PIC16F648
Pin 3 GND → Pin 5 PIC16F648
Pin 4 PGD → Pin 13 PIC16F648
Pin 5 PGC → Pin 12 PIC16F648
Pin 6 NC → NOT CONNECTED

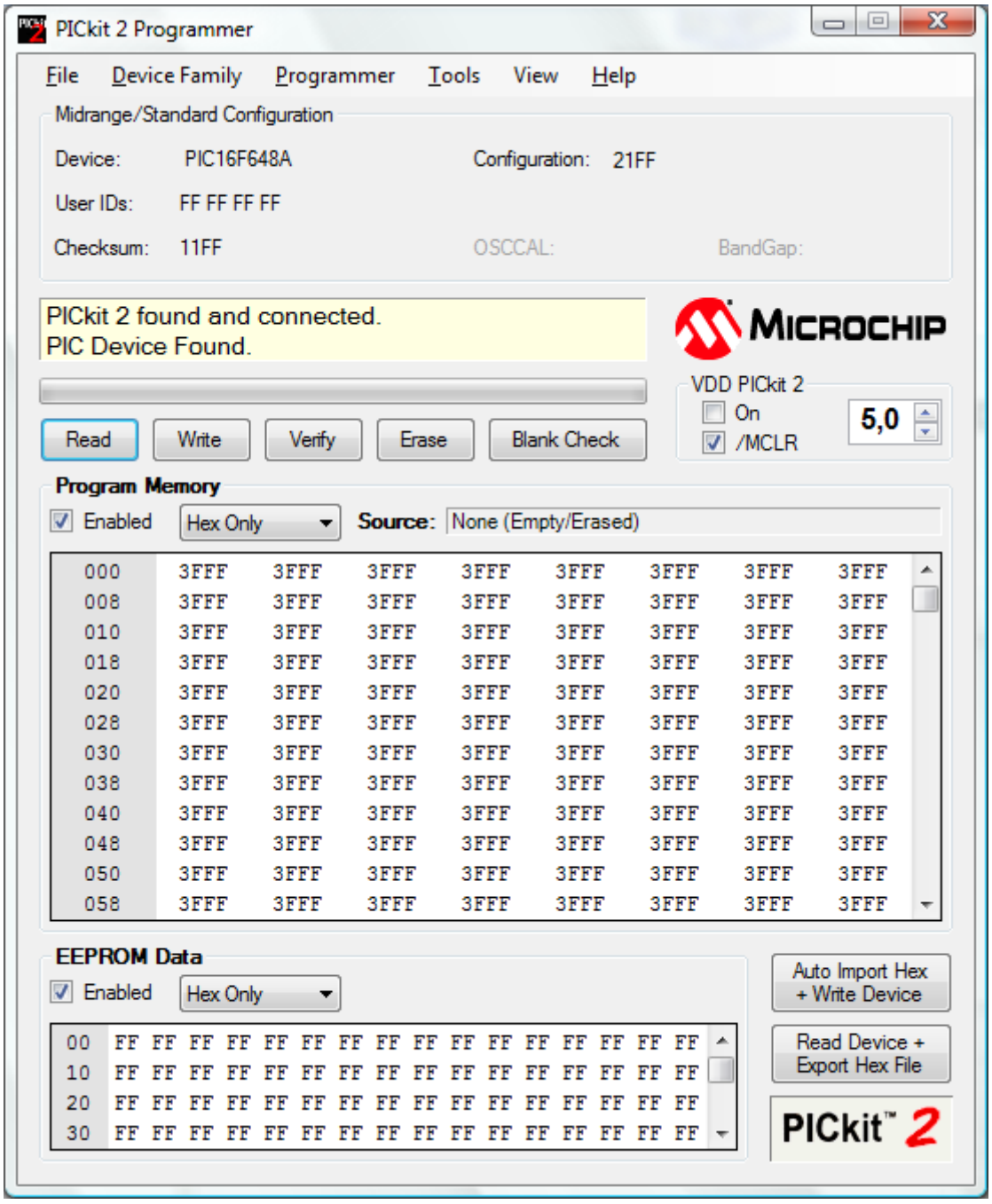
3.9 Programmeren met PICKIT2

Download PICKIT2 software van www.D-A-E.eu → forum → Software.
Installer PICKIT2 2 v2.50.02 setup A.
Sluit hem dan op de pc naar de breadboard aansluiting.
Voor dat je gaat programmeren in de PIC moet de spanning van het circuit van de PIC Afgesloten zijn, anders vindt de PICkit2 de device niet zoals in Figuur 22.



Figuur 22 PIC device onbekend.

Daarna → Tools → check communication.

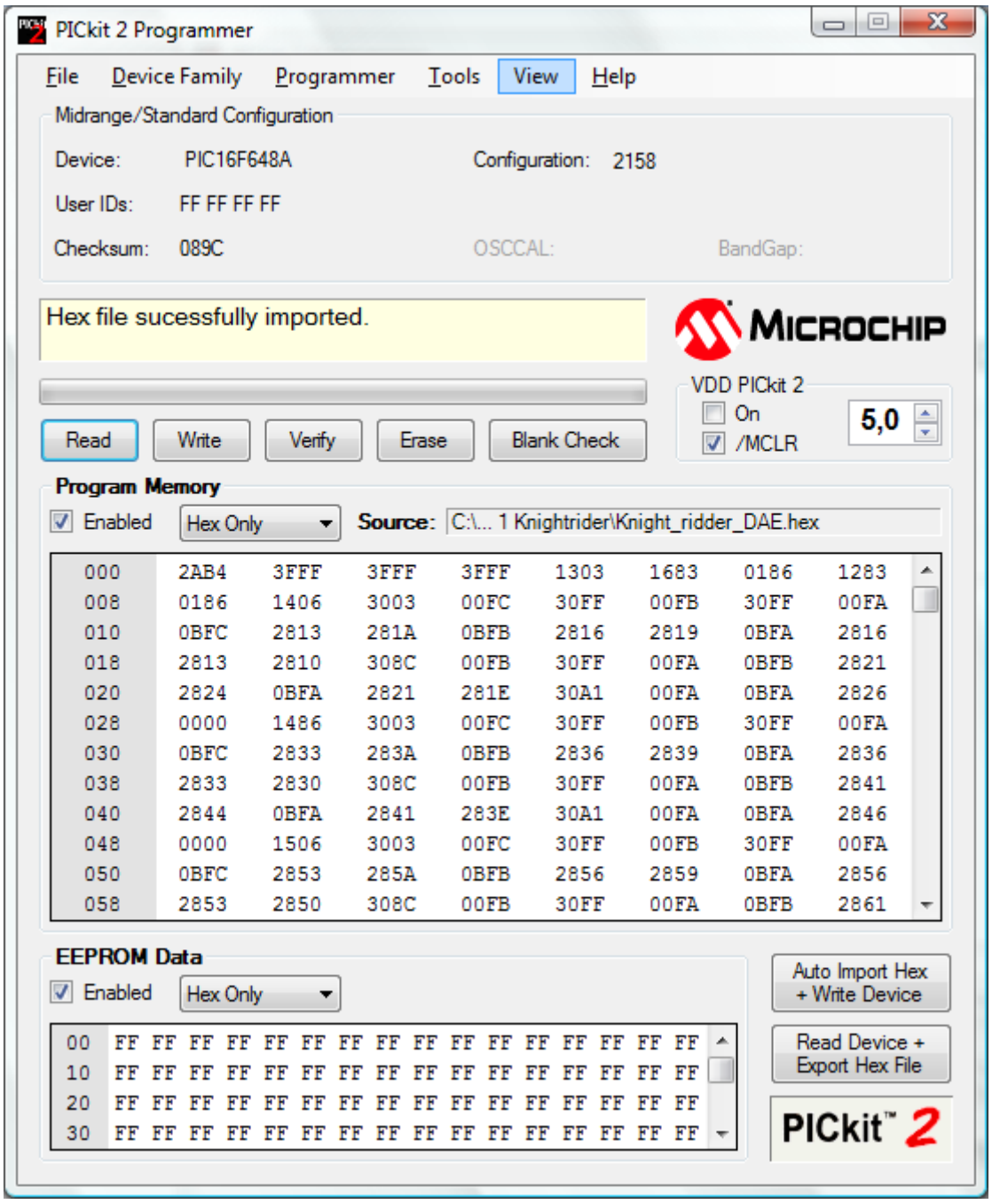


Figuur 23 PIC device PIC16F648A herkent.

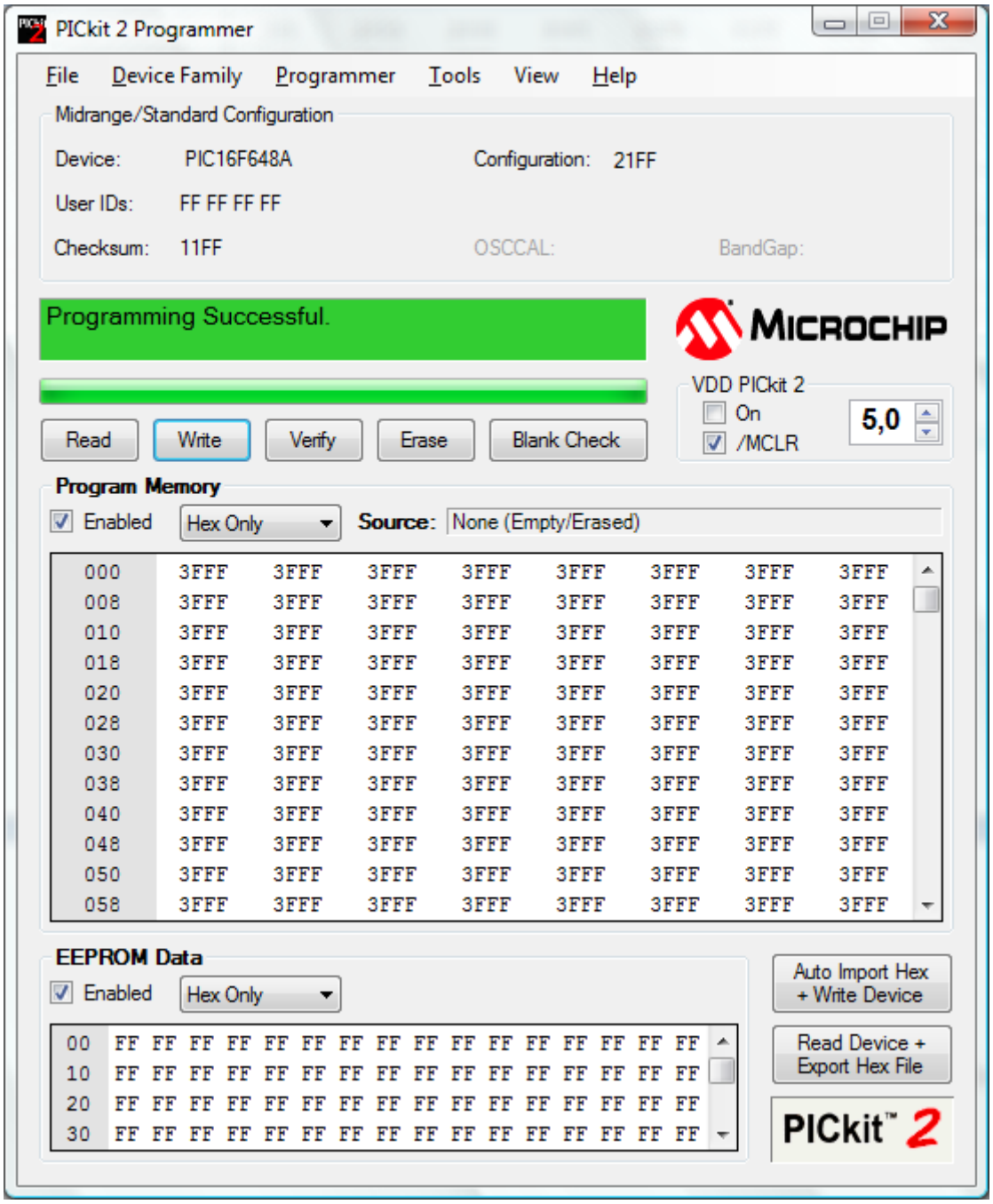
Nadat de PICKIT2 de Device herkent zoals in Figuur 23.
Daarna →Erase eerst de PIC
Daarna→ Blank check.
Daarna→ Verify.

Importeren van Hex dump wat gecompileerd is door MikroC of andere compileren kan via de sneltoets Ctrl+I of onder File → Importeren.

Als de Hex dump is geïmporteerd, laat hij Hex file sucessfully imported op display zien.



Wegschrijven naar het PIC kan via de sneltoets Ctrl+W of onder Programmeren → Write.



Daarna→ Verify.

Nu kan je de voeding weer inschakelen.

Het Programma start op.

Toepassingen

Microcontrollers zijn niet meer weg te denken uit het dagelijkse leven. Met microcontrollers worden thuis de wasmachine, magnetron, droger en auto bestuurd. De microcontrollers van
Microchip:
(http://www.microchip.com)
winnen door hun veelzijdigheid snel aan populariteit.

Deze microcontrollers hebben standaard timers, een USART, AD converters, PWM outputs etc. aan boord. Hierdoor kunnen met eenvoudige electronische schakelingen toch ingewikkelde functies worden uitgevoerd. Besturing van motoren, temperatuur meting, IR afstandsbediening en een looplicht zijn voorbeelden van schakelingen die met behulp van deze microcontrollers eenvoudig te maken zijn.

Door de massa gebruik van microcontrollers, komen er steeds meer source codes vrij voor programmeurs. De source code`s kan gebruikt worden door scholen, bedrijven en hobbyisten in verschillende toepassingen van automatiseringen.
We kunnen grote computers vervangen met een kleine microchip, maar ook hele printen met schakelingen.

Conclusie

Naar mijn mening gaan microcontrollers in de toekomst een grote rol spelen binnen scholen, bedrijven, en onder hobbyisten. De onkosten van de microcontroller zijn erg laag en dus snel terug te verdienen. Wie echt wil, kan de basiskennis vrij snel leren te beheersen. Voor programmeertalen zijn er genoeg compilers te krijgen voor microchip. De kracht en voordelen van het automatiseren van installaties en van Demotica besturingen zullen voor bedrijven grote veranderingen met zich mee brengen.

De mogelijkheden worden steeds groter omdat er steeds meer informatie beschikbaar komt van hobbyisten. Door de informatie en voorbeelden die hobbyisten verspreiden, raken steeds meer mensen geïnteresseerd in microcontrollers. Als deze mensen gemotiveerd bezig gaan met microcontrollers, zal dit de automatisering alleen maar ten goede komen.

Gebruikte symbolen en afkortingen

µC	MicroController
ADC	Analoog Digitaal Converter
ADCON	Instellen van poort ADC of Digital I/O
ALU	Arithmetic Logic Unit
ASCII	American Standard Code for Information Interchange
BOR	Brown-Out Reset
CISC	Complex Instruction Set Computer
CCP	Capture, Compare & PWM
CPU	Central Processing Unit
DIP	Dual Inline Package
DSC	Digital Signal Controller
EEPROM	Electrical Erasable Programmable Read Only Memory
Embedded Systems	Een elektronische systeem (hardware en software)
FET	Field Effect Transistor
GPR	General Purpose Register
HSB	Highest Significant Bit
I/O	Input/Output
IC	Integrated Circuit
ICD	In-Circuit Debugger
ICSP	In-Circuit Serial Programming
IDE	Integrated Development Environment
ISR	Interrupt Service Routine
LCD	Liquid Crystal Display
LSB	Least Significant Bit
LVP	Low-Voltage Programming
MCLR	Master Clear
MCU	Micro-Controller Unit
MSSP	Master Synchronous Serial Port
OST	Oscillator Start-up Timer
PC	Program Counter
PCB	Printed Circuit Board, printplaat
PGD	ProGram Data
PGC	ProGram Clock
PGM	LV programming mode
PIC	Microcontroller van Microchip
Pipelining	Instructie methode van decoderen, uitvoeren en wegschrijven
POR	Power-ON Reset
PSP	Parallel Slave Port
PWM	Puls Width Modulation
PWRT	PoWer-up Timer
RAM	Random Acces Memory
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
SFR	Special Function Register

TRIS	Instellen poort Input of Output (Tri-State) registers
TTL	Transistor Transistor Logic
USART	Universal Synchronous/Asynchronous Receiver Transmitter
WDT	Watch Dog Timer