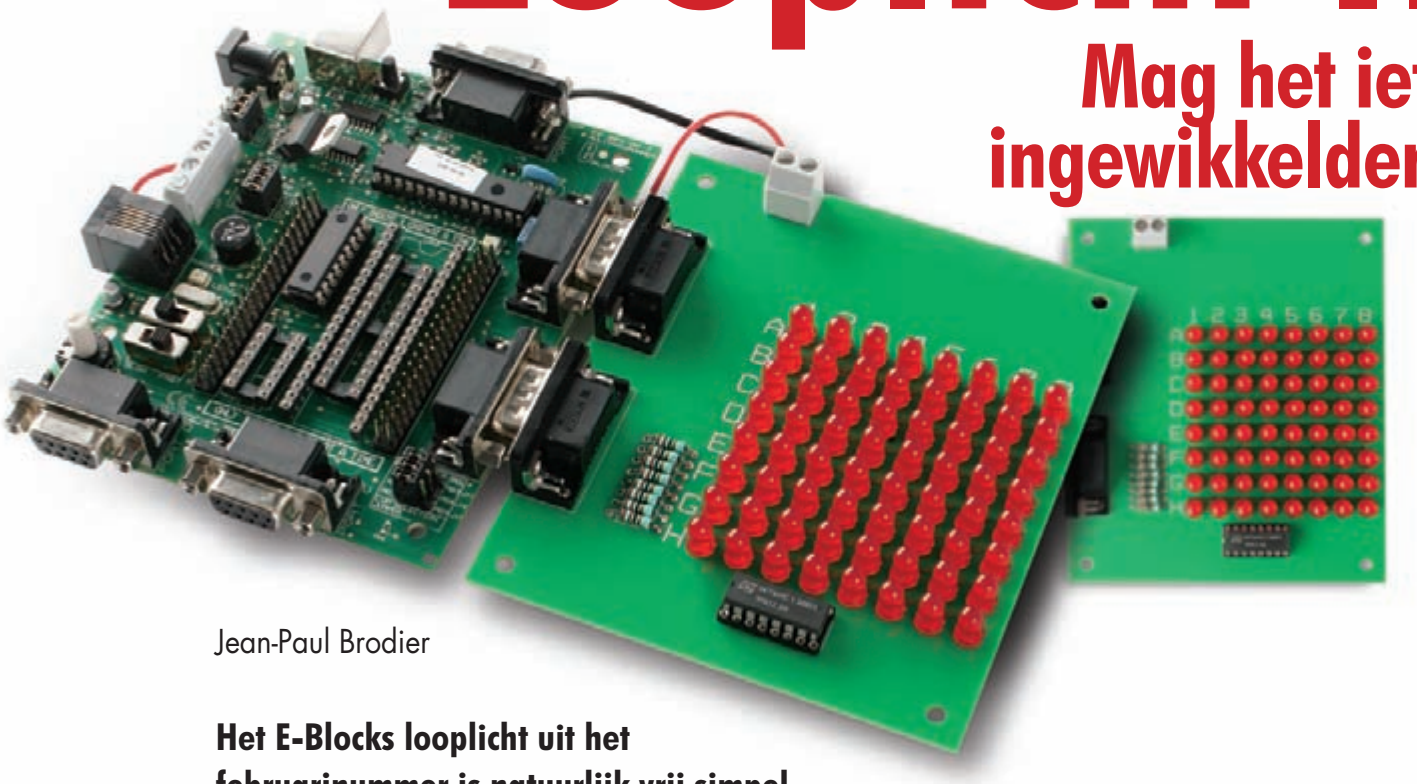
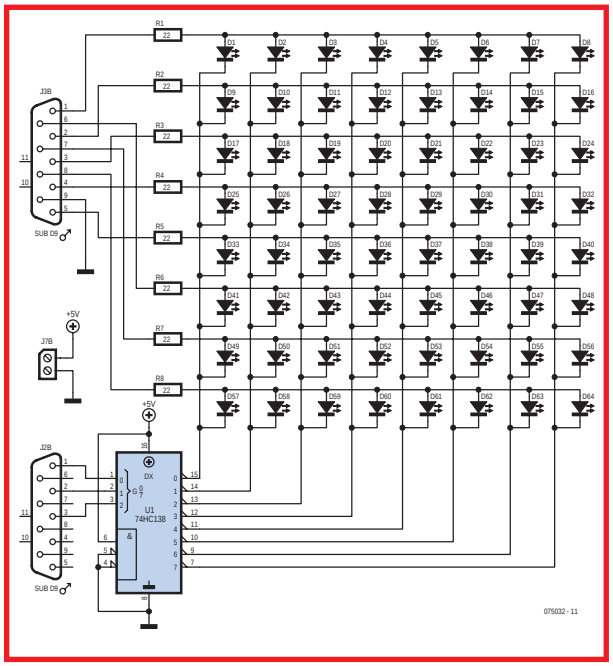
E-Blocks Looplicht II

Mag het iets ingewikkelder?



Jean-Paul Brodier

Het E-Blocks looplicht uit het februarinummer is natuurlijk vrij simpel. Deze maand gaan we het uitbreiden naar een groter aantal LED's. Helaas is het oorspronkelijke ontwerp niet voorbereid op deze uitbreiding.



Figuur 1: De 64 LED's worden opgesteld in een matrix van 8x8.

Om 64 LED's rechtstreeks aan te sturen, zouden 64 outputlijnen nodig zijn. Die heeft onze 18-pens PIC16F88 niet. Daarom gaan we de LED's gemultiplexed aansturen in een matrix van 8 rijen en 8 kolommen, zoals te zien is in het schema (**figuur 1**).

Zeeslag

We kunnen elke LED aanwijzen met een rij- en kolomnummer, net als in het spelletje 'Zeeslag'. Als we één rij selecteren (met een logisch '1' niveau) en één kolom (met een logische '0'), dan zal de LED op de kruising van die rij en kolom oplichten. (De anode is verbonden met de rij en de kathode met de kolom.)

Decoderen

Net als in het oorspronkelijke ontwerp worden de rijen aangestuurd met de outputlijnen van PORTB, maar dan komen we outputlijnen tekort om ook al de kolommen aan te sturen. Daarom maken we gebruik van een decoder-IC, de 74HC138. Dit is een 3-naar-8-decoder (BCD decoder). De binaire waarde op de drie ingangslijnen A, B en C bepaalt welke van de 8 uitgangslijnen geactiveerd

wordt (de geactiveerde lijn wordt logisch '0' gemaakt).
Om de acht kolommen één voor één aan te sturen, moeten we het kolomnummer variëren van 0...7 en binair aanbieden aan de ingangen van de decoder, die gekoppeld zijn aan drie uitgangen van PORTA. In de software wordt het kolomnummer telkens verhoogd met de programmaregel `KOLOM = KOLOM + 1`. De waarde kan nooit groter worden dan 7 dankzij de `Modulo` 8-operatie.
We maken in Flowcode een nieuw bestand aan met de naam Ch2D0.fcf ('2D' staat voor tweedimensionaal).
Om te beginnen moeten de variabele COLUMN en verschillende variabelen voor de rijen worden aangemaakt (zie **figuur 2**). Daartoe slepen we een Calculation-pictogram naar de lijn tussen BEGIN en END in de flowchart en passen de eigenschappen van het pictogram aan. (Gebruik de knop *Variabele toevoegen* in het venster *Variabelenbeheer*.)
Als de nieuwe variabelen zijn ingevoerd, kunnen ze in het programma gebruikt worden door de waarde van `KOLOM` telkens te incrementeren en naar PORTA te sturen, terwijl de bijbehorende `RIJ`-waarde naar PORTB gestuurd wordt.

Programmeren in C

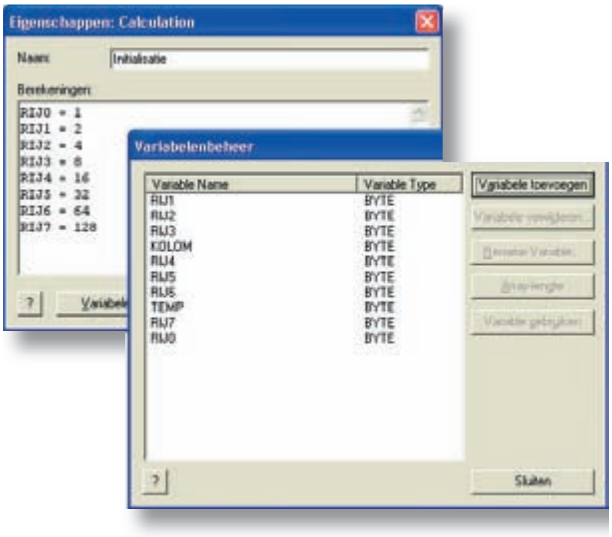
We maken gebruik van de programmeertaal C als dat handiger of sneller is dan het gebruik van de symbolen in de flowchart, bijvoorbeeld als er meer dan één rekenkundige operatie tegelijk uitgevoerd moet worden.
Het eerste 'C-Code'-pictogram in het programma heet Modulo 8 (zie **figuur 3**) en bevat slechts één regel code:

```
FCV_KOLOM = FCV_KOLOM % 8;
```

Het prefix `FCV_` voor de naam van de variabele `KOLOM` geeft voor de compiler aan dat deze variabele in Flowcode gedefinieerd is. Dat geldt voor alle Flowcode-variabelen die we in C gebruiken. Voor het gebruik van Flowcode-variabelen in assemblertaal wordt het prefix `_FCV_` gebruikt.
Het %-teken in de programmaregel geeft aan dat een modulo-operatie moet worden toegepast. Dat wil in dit geval zeggen dat het resultaat gelijk is aan de rest van de deling van het getal `KOLOM` door 8.
De variabele `KOLOM` wordt elke keer dat de programmalus doorlopen wordt met 1 opgehoogd. Zonder de modulo-operatie zou de waarde dus gewoon steeds groter worden, tot aan de maximale waarde van 255 die in een 8-bits register past en dan weer terugkomen op 0. Dat past niet in onze toepassing, omdat wij alleen maar de waarden 0...7 willen gebruiken om de drie outputlijnen van PORTA aan te sturen.
Zolang de waarde van `KOLOM` kleiner is dan 8 is de rest van de deling door 8 gelijk aan de waarde van `KOLOM` zelf. Pas als `KOLOM` de waarde 8 bereikt bewijst deze programmaregel zijn nut, want de rest van 8 gedeeld door 8 is 0, dus `KOLOM` wordt weer terug gezet op de beginwaarde.
We zouden hetzelfde effect kunnen bereiken door gebruik te maken van een bitmasker, een logische EN-operatie. We weten dat de toegestane waarden 0...7 overeenkomen met de binaire getallen 000...111. De regel C-code had er dus ook zo uit mogen zien:

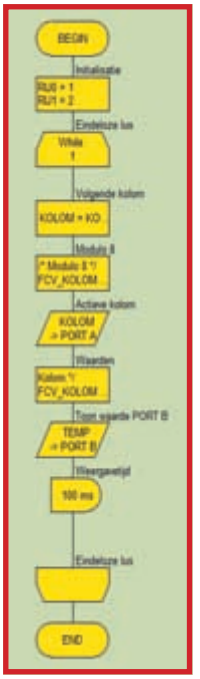
```
FCV_KOLOM = FCV_KOLOM & 7;
```

In C wordt de logische EN-operatie weergegeven met het symbool '&' (ampersand).

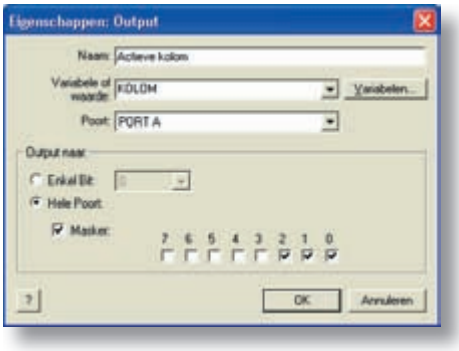


Figuur 2: Bij het aanmaken van nieuwe variabelen wordt ook een initiële waarde toegekend.

We kunnen de waarde van de `KOLOM`-variabele niet botweg naar PORTA schrijven. Om te voorkomen dat de waarde van de lijnen RA3...RA7 beïnvloed wordt, moeten we bij het schrijven naar het poortregister gebruik maken van een masker. We regelen dat in het venster *Eigenschappen* van het eerste OUTPUT-pictogram (zie **figuur 4**). Plaats vinkjes bij "Masker" en de bits die geschreven moeten worden. De andere bits worden niet beïnvloed.
Elke keer dat de lus doorlopen wordt, schrijven we een nieuwe `RIJ`-waarde naar PORTB, waardoor bepaald wordt welke LED's op deze rij moeten oplichten. Het is de bedoeling dat een lichtpuntje zich van links naar rechts of van rechts naar links verplaatst op de LED-matrix. Daarvoor wordt gebruik gemaakt van de variabele `TEMP`. Wat er in `TEMP` geschreven wordt, wordt bepaald aan de hand van de variabele `KOLOM` en vervolgens weggeschreven naar PORTB. De TEMP byte is nodig om tijdelijk een waarde in op te kunnen slaan die later gebruikt moet worden.
Als het programma zonder fouten gecompileerd kan worden, mag het naar de PIC geschreven worden met het commando *Chip -> Compileer naar Chip*, net zoals we dat in het vorige artikel beschreven hebben. Het resultaat is een lichtpuntje dat diagonaal over de 8x8 LED-matrix beweegt, maar we zien nog steeds slechts één lichtpuntje in beweging.
Om een duidelijke lijn te zien, zouden alle LED's van de diagonaal tegelijk moeten oplichten. Maar hoe laten we A1 en B2 tegelijk oplichten? Als we rij A en B tegelijk activeren en ook kolom 1 en 2, dan zouden er vier LED's

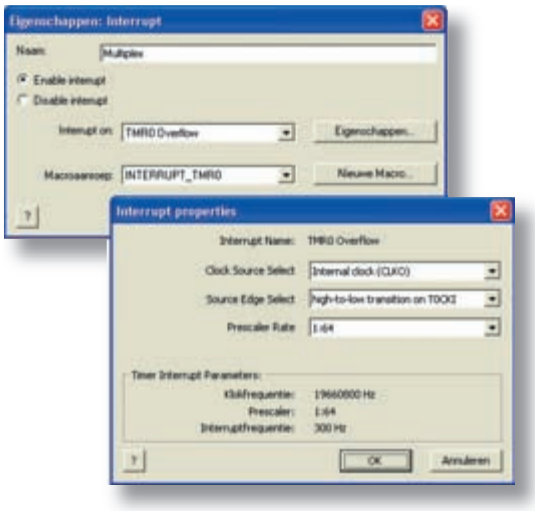


Figuur 3: Het testprogramma voor het matrixdisplay.

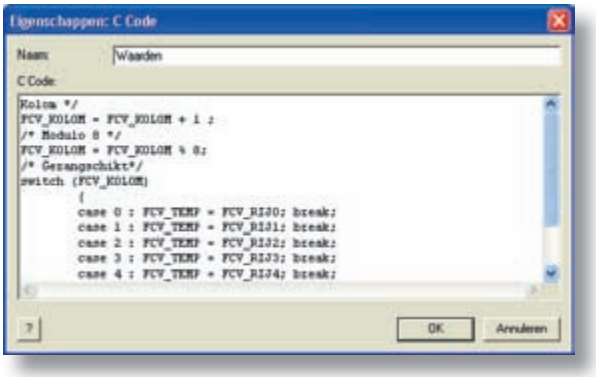


Figuur 4: Toekennen van de waarde van variabele KOLOM aan de uitgangslijnen van PORTA.

Figuur 7: De interruptroutine is erg eenvoudig.



Figuur 5: Interrupteigenschappen.

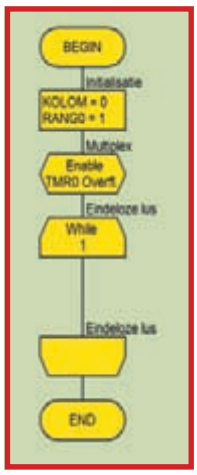


Figuur 6: De C-code in de interruptroutine.

tegelijk gaan branden in plaats van twee. De oplossing ligt in het gemultiplexed aansturen van de LED-matrix. Achtereenvolgens worden eerst rij A en kolom 1 tegelijk geactiveerd en daarna rij B en kolom 2 enzovoort. Als we dat snel genoeg doen, lijkt het voor het oog of de LED's tegelijk branden. We zien dan een diagonale lijn die bestaat uit de LED's A1, B2 enzovoort.
Voordat we het programma daarvoor gaan aanpassen, schrijven we het weg onder een andere naam, bijvoorbeeld Ch2D1.fcf (met Bestand->Opslaan als...)

Interrupts

Het scannen van het display moet erg snel gebeuren, maar het is simpel werk waarbij geen berekeningen en dergelijke nodig zijn. Daarom is de meest geschikte manier om zoiets te doen met behulp van een interruptroutine die wordt aangestuurd met één van de ingebouwde timers van de microcontroller.
We slepen een zeshoekig INT-pictogram naar een plaats onder de initialisatie van de variabelen en stellen de interrupteigenschappen in zoals in **figuur 5**.
Elke keer dat teller TMR0 door 0 gaat, dat is 300 keer per seconde, wordt de routine INTERRUPT_TMR0 aangeroepen. (In Flowcode noemt men deze routine een MACRO.) We hoeven nu alleen maar de instructies die eerst in het hoofdprogramma zaten over te brengen naar deze macro. Klik op Macro->Bewerken en daarna op INTERRUPT_TMR0. Sleep dan een C-Code-pictogram naar het begin van de flowchart en kopieer de instructies uit het vorige programma er naartoe.
Om werk te besparen en fouten te voorkomen is het verstandig om de file Ch2D0.c te openen in Kladblok of een andere eenvoudige tekstverwerker. Sleep met de muis over het te kopiëren gedeelte van de programmacode en druk op Ctrl-C (kopiëren). Ga dan terug naar Flowcode



Figuur 8: Demonstratie van multiplexing. Hier wordt een statische diagonale lijn weergegeven.

De ANSI-standaard en C in MPLAB

De programmeertaal C zoals die wordt ondersteund door de MPLAB-compiler komt niet geheel overeen met de ANSI-standaard.
De maskeroperatie kan geschreven worden als:

```
FCV_KOLOM &= 7;
```

C-programmeurs zijn dol op deze compacte schrijfwijze, omdat die moeilijk te lezen is (ook voor henzelf) en ontmoedigend werkt voor nieuwelingen, waardoor die een groter ontzag voor meer ervaren programmeurs krijgen. Vreemd genoeg wordt de uitdrukking:

```
FCV_KOLOM %= 8;
```

die volgens de ANSI-standaard geheel correct is, door de compiler juist weer niet geaccepteerd.

Het gebruik van moderne computers met knip- en plakmogelijkheden maakt het besparen van een paar toetsaanslagen met dit soort syntaxconstructies vrijwel overbodig. Of een uitdrukking nu in de lange of in de verkorte vorm geschreven wordt, het resulterende programma in assemblertaal is in beide gevallen gelijk, zoals in de .LST-files van de assembler te zien is.

Een ander klein verschil tussen ANSI en MPLAB: In MPLAB moeten alle variabelenamen in hoofdletters gespeld worden, terwijl volgens de ANSI-standaard hoofd- en kleine letters door elkaar gebruikt mogen worden en ook herkend worden als van elkaar verschillend.

en plak de regels in het C-Code-pictogram met Ctrl-V (plakken) (**zie figuur 6**).
Als de C-code is toegevoegd, hoeven alleen nog de definities van PORTA en PORTB te worden toegevoegd (zie **figuur 7**). Het multiplex demonstratieprogramma (**figuur 8**) maakt gebruik van de interruptroutine.

Beweging!

We wilden een lichtanimatie en die kunnen we nu gaan maken. We maken een diagonaal looplicht en nog een tweede op een afstand van 4 LED's van de eerste. Dit wordt weer een nieuw programma, nu met de naam Ch2D2.fcf.
Om te beginnen wordt de initiële waarde van `RIJ` aangepast, zodat er niet één, maar twee lichtpuntjes op de regel komen. De hoofdlus van het programma zorgt er daarna voor dat deze waarde in de registers rondgeschoven wordt.
De opeenvolgende hexadecimale waardes voor `RIJ0...RIJ3` zijn 0x11, 0x22, 0x44 en 0x88. Diezelfde waarden worden herhaald voor `RIJ4...RIJ7`. Omdat Flowcode geen hexadecimale getallen begrijpt, moeten we de waarden invoeren als decimale getallen, dus 17, 34, 68 en 136.
Tijdens het draaien van het programma worden de waarden opgeschoven door het getal telkens met 2 te vermenigvuldigen. Als het meest linkse bit '1' geworden is (bij 136, binair 10001000), wordt bij de volgende stap weer de startwaarde 17 (00010001) gebruikt.
Afhankelijk van het moment krijgen we nu drie of vier lijnen te zien in plaats van twee (zie **figuur 9**).
Door de initiële waarden te veranderen kunnen we nu verschillende patronen laten weergeven, zoals een diagonale lijn (1, 2, 4, 8, 16, 32, 64, 128), sergeantsstrepen (1, 2, 4, 8, 4, 2, 1, 2) of een onderbroken lijn. Het programma houdt bij hoeveel keer de cyclus doorlopen is en kan na een gegeven aantal herhalingen een nieuw patroon inladen om weer te geven met behulp van de constanten die van te voren in het programmageheugen geladen zijn.

Onderdelenlijst

Weerstanden
R1...R8 = 22 Ω

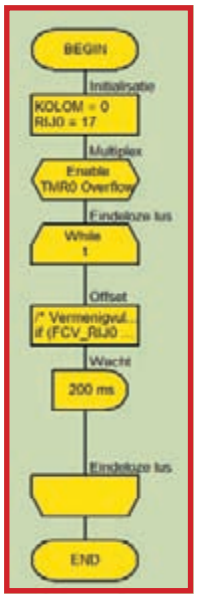
Halfgeleiders
D1...D64 = LED, diameter 3 of 5 mm
IC1 = 74HC138

Diversen
J2B, J3B = 9-polige sub-D-connector (male) voor printmontage
J7B = 2-polige schroefaansluiting voor printmontage

Hoewel dit één van de kleinste microcontrollers is die momenteel op de markt zijn, gebruikt zelfs dit laatste programma nog lang niet alle beschikbare capaciteit. Het neemt slechts 391 van de 4096 geheugenwoorden in beslag dat het flashgeheugen groot is.

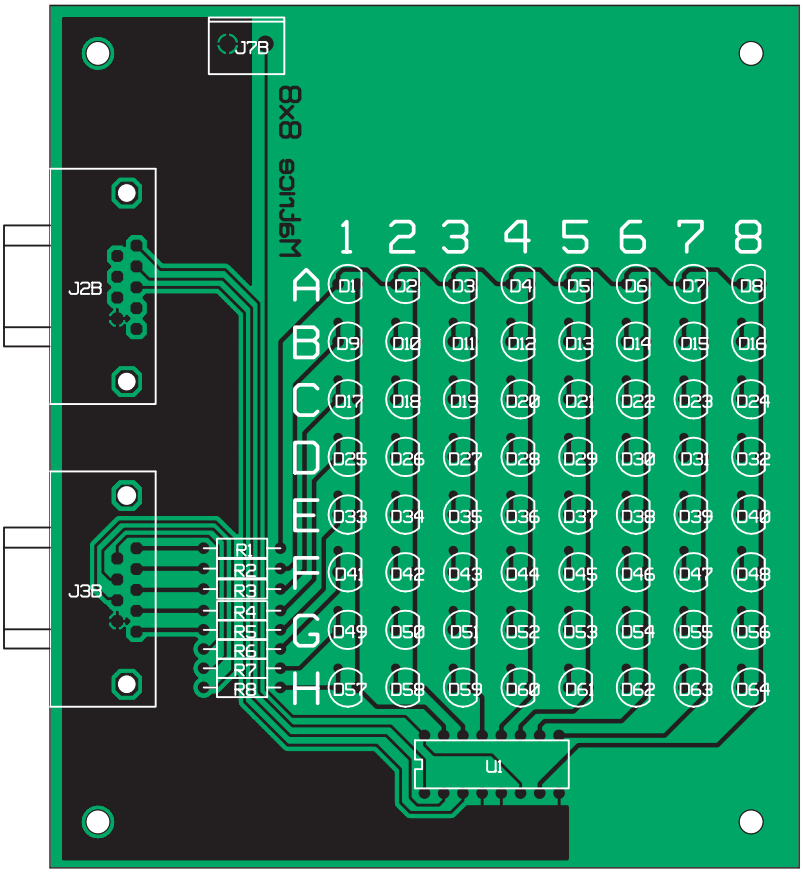
Opbouw

De schakeling kan worden opgebouwd op een stuk gaatjesprint of op de printplaat in **figuur 10**. De print is voorzien van twee male sub-D-connectors (J2B en J3B). Die zijn zo genoemd omdat ze corresponderen met de vrouwelijke DB9-connectors J2 (PORTA) en J3 (PORTB) op de E-Blocks multiprogrammer-kaart. De +5 V voedingsspanning wordt betrokken van de schroefconnector J7; de massalijn is via de sub-D-connectors (pen 9) al doorverbonden.
De weerstanden van 22 Ω zitten er eigenlijk meer voor de show, want de stroom door de LED's wordt in feite begrensd door (ketterij, dat mag toch niet?) de maximale uitgangsstroom van de microcontroller. De totale uitgangsstroom door de pennen van PORTB is circa 100 mA. Voor voldoende lichtopbrengst kunnen het beste high-efficiency-LED's gebruikt worden.
Een dubbelzijdige print is voor deze eenvoudige schakeling niet nodig. De matrixverbindingen kunnen gemakkelijk gemaakt worden met wire-wrap draad. Begin met het vertinnen van de kopervlakjes bij de anodes van de LED's. Duw daarna het draad met de punt van de soldeerbout in de soldeerklodder. Het isolatiemateriaal smelt nu van de draad af en er komt een geleidende verbinding met het soldeereiland tot stand. Trek voorzichtig aan het draad om te zien of het goed vast zit en ga door naar het volgende soldeervlakje. Als alle punten van een rij met elkaar verbonden zijn, kan de draad worden afgeknipt.



Figuur 9:
De eenvoud van de flowchart verbergt een groot aantal statements in C.

(075032-I)



Figuur 10:
De schakeling kan worden opgebouwd op een enkelzijdige printplaat met draadbruggen voor het doorverbinden van de 8 LED's op elke rij. Let goed op de polariteit van de LED's! De PCB-layout kan van de Elektuur-website worden gedownload.