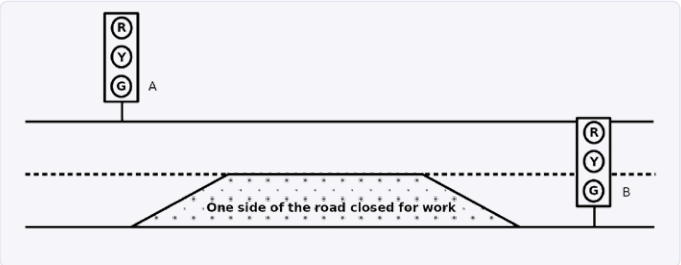
Verkeerslichten programmeren in PIC-assembleertaal

Andrew Pratt (Groot-Brittannië)

In dit kleine project programmeren we twee verkeerslichten met zes LED's die de beide sets rode, gele en groene lichten vertegenwoordigen (**figuur 1**). Deze bootsen het type verkeerslicht na dat bij wegwerkzaamheden wordt gebruikt om de verkeersstroom via één rijstrook te regelen. Het programmeren doen we in PIC-assembleertaal en dus niet in een 'hogere' taal.

Bij het opstarten zijn beide verkeerslichten rood, en dat blijven ze gedurende 10 s. Vervolgens toont verkeerslicht A gedurende 2 s rood en geel en daarna groen. Na nog eens 20 seconden toont verkeerslicht A gedurende 5 s geel en daarna weer rood. Beide verkeerslichten blijven dan 10 s rood voordat verkeerslicht B dezelfde volgorde doorloopt. Deze cyclus wordt herhaald tot de stroom wordt uitgeschakeld. Daarnaast is er ook een enable-signaal dat de kleurwisselingen mogelijk maakt – als dit signaal laag wordt, stopt de reeks zodra beide verkeerslichten rood zijn. Het enable-signaal moet hoog zijn gedurende de gehele periode van 10 s waarbij beide lichten rood zijn voordat de cyclus wordt gestart.

Uit deze tekstuele beschrijving moeten we de verschillende toestanden afleiden waarin het systeem zich kan bevinden:

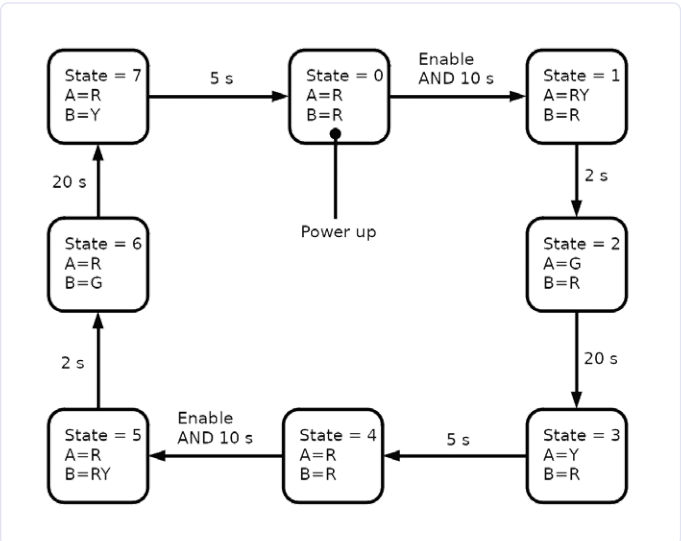


Figuur 1. Bij wegwerkzaamheden moet het verkeer worden geregeld. Opmerking: in Groot-Brittannië rijdt men aan de linkerkant!

Toestand	Verkeerslicht A	Verkeerslicht B
0	rood	rood
1	rood + geel	rood
2	groen	rood
3	geel	rood
4	rood	rood
5	rood	rood + geel
6	rood	groen
7	rood	geel

Nu we de acht verschillende toestanden hebben geïdentificeerd, moeten we de toegestane overgangen en de beginvoorwaarden uitwerken om deze toestanden te activeren. Dit is een heel eenvoudig voorbeeld in die zin dat de toestanden 'netjes' op elkaar volgen met steeds slechts één enkele toegestane overgang naar de volgende toestand. Het resulterende Moore-toestandsdiagram is in **figuur 2** te bewonderen. De uitgangen zijn alleen afhankelijk van de huidige toestand en zijn telkens in elk blok aangegeven.

De overgang naar de volgende toestand wordt bepaald door de verstreken tijd in een bepaalde toestand, met uitzondering van het verlaten van



Figuur 2. Toestandsdiagram wegwerkzaamheden.

Listing 1. Het complete verkeerslicht-programma.

```
;PROG_8_03.asm
LIST     P=16F1823
#INCLUDE <p16f1823.inc>
#INCLUDE <fsm_macros.inc>

    RADIX DEC              ; Default numbers are to base 10.
    BOOK_CONFIGURATION     ; See macro in fsm_macros.inc.

    CBLOCK 0x70
    TICKS                  ; Used to count 8.2ms ticks.
    SECONDS                ; Used to hold the seconds count.
    ENDC

    ORG 0X00
    GOTO START

    ORG 0X04               ; Interrupt vector.
    BCF INTCON, TMR0IF     ; Clear the tmr0 overflow interrupt flag.
    DECF TICKS, F          ; Decrement by one the 8.2ms ticks.
    BTFSS  STATUS, Z       ; Test to see if TICKS has reached zero.
    GOTO $+4               ; If TICKS has not reached zero jump to return from interrupt.
    INCF SECONDS, F        ; If TICKS has reached zero increment the seconds count.
    MOVLW 122
    MOVWF TICKS            ; Restore TICKS to 122 (122 x 8.2 ms 1 second).
    RETFIE                 ; Return from interrupt.

START
    MOVLB 1                ; Bank 1 required for the following macros, and TRISC.
    SET_FREQ_32MHZ         ; See file fsm_macros.inc.
    SET_TMR0_CASE1         ; Gives 8.2 ms ticks. See fsm_macros.inc.
    CLRF TRISC             ; Set all PORTC as outputs.
    MOVLB 0                ; Select bank 0 for PORTA and PORTC.
;=============================================================================================
S0
    MOVLW b'00100100'
    MOVWF PORTC                  ; Turn Red A (RC5) and Red B (RC2) on others off.
SS0_0
    CLRF SECONDS                 ; Clear the seconds counter.
SS0_1
    BTFSS PORTA, 5
    GOTO SS0_0                   ; If the enable signal is low return to sub-state 0.
    MOVLW 10
    IF_REG_LESS_THAN_W SECONDS
    GOTO SS0_1                   ; If less than 10 s have elapsed stay in substate 1.
;=============================================================================================
S1
    MOVLW b'00110100'
    MOVWF PORTC             ; Turn Red A (RC5) and Yellow A (RC4) and Red B (RC2) on others off.
SS1_0
    CLRF SECONDS            ; Clear the seconds counter.
SS1_1
    MOVLW 2
    IF_REG_LESS_THAN_W SECONDS
    GOTO SS1_1
S2
    MOVLW b'00001100'
    MOVWF PORTC               ; Turn Green A (RC3) and Red B (RC2) on others off.
SS2_0
    CLRF SECONDS
SS2_1
    MOVLW 20
    IF_REG_LESS_THAN_W SECONDS
    GOTO SS2_1                ; If less than 20 s have elapsed stay in substate 1.
;=============================================================================================
S3
```

verder op volgende pagina...

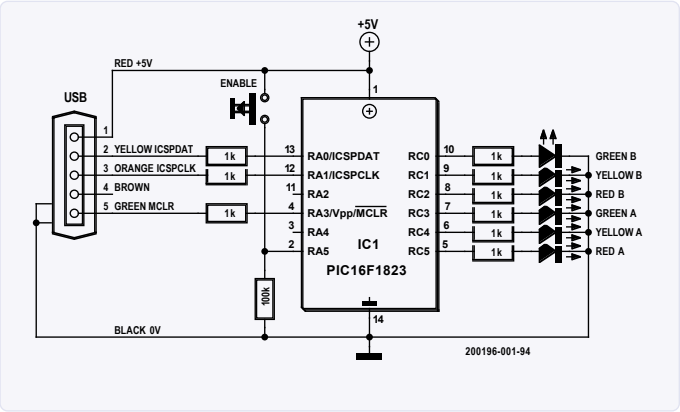
```
    MOVLW b'00010100'
    MOVWF PORTC                     ; Turn Yellow A (RC4) and Red B (RC2) on others off.
SS3_0
    CLRF SECONDS                    ; Clear the seconds counter.
SS3_1
    MOVLW 5
    IF_REG_LESS_THAN_W SECONDS
    GOTO SS3_1                      ; If less than 5 s have elapsed stay in substate 1.
;=================================================================================================
S4
    MOVLW b'00100100'
    MOVWF PORTC                     ; Turn Red A (RC5) and Red B (RC2) on others off.
SS4_0
    CLRF SECONDS                    ; Clear the seconds counter.
SS4_1
    BTFSS PORTA, 5
    GOTO SS4_0                      ; If the enable signal is low return to sub-state 0.
    MOVLW 10
    IF_REG_LESS_THAN_W SECONDS
    GOTO SS4_1                      ; If less than 10 s have elapsed stay in substate 1
;=================================================================================================
S5
    MOVLW b'00100110'
    MOVWF PORTC              ; Turn Red A (RC5) and Red B (RC2) and Yellow B (RC1)on others off.
SS5_0
    CLRF SECONDS                ; Clear the seconds counter.
SS5_1
    MOVLW 2
    IF_REG_LESS_THAN_W SECONDS
    GOTO SS5_1                  ; If less than 2 s have elapsed stay in substate 1 .
;=================================================================================================
S6
    MOVLW b'00100001'
    MOVWF PORTC                   ; Turn Red A (RC5) and and Green B (RC0) on others off.
SS6_0
    CLRF SECONDS                  ; Clear the seconds counter.
SS6_1
    MOVLW 20
    IF_REG_LESS_THAN_W SECONDS
    GOTO SS6_1                    ; If less than 20 s have elapsed stay in substate 1.
;=================================================================================================
S7
    MOVLW b'00100010'
    MOVWF PORTC                 ; Turn Red A (RC5) Yellow B (RC1) on others off.
SS7_0
    CLRF SECONDS                ; Clear the seconds counter.
SS7_1
    MOVLW 5
    IF_REG_LESS_THAN_W SECONDS
    GOTO SS7_1                  ; If less than 5 s have elapsed stay in substate 1.
    GOTO S0

    END
```

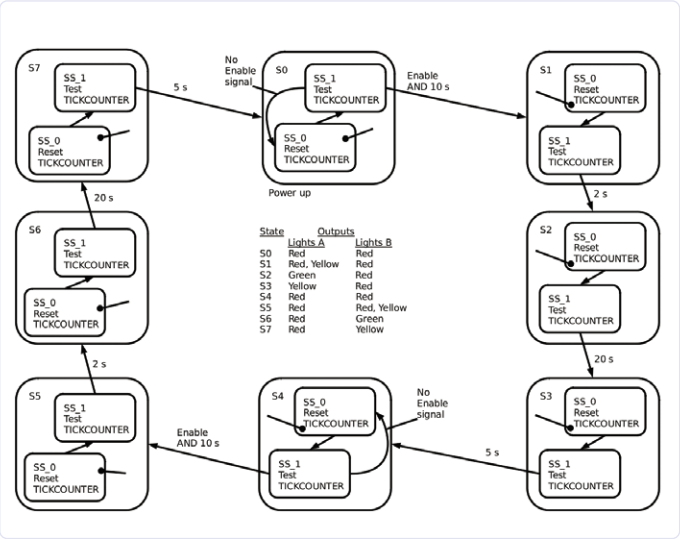
de toestanden 0 en 4 waarin beide verkeerslichten rood zijn: in deze twee toestanden moet het enable-signaal gedurende 10 s aanwezig zijn. Hiermee wordt voldaan aan de voorwaarde dat de cyclus doorloopt naar de volgende dubbel rood-toestand als het enable-signaal laag wordt. In het schema van **figuur 3** zijn er zes LED's aangesloten via serieweerstanden. De waarde van deze weerstanden is niet kritisch, 1 kΩ resulteert in een stroom van ongeveer 3 mA. Er wordt een PIC van het type 16F823 gebruikt; deze wordt geprogrammeerd via een FTDI USB/serieel-adapter.

Voorafgaand aan het produceren van de code die de bovenstaande toestandsmachine uitvoert, moet de PIC worden geconfigureerd, zoals de configuratiebestand-instellingen, de oscillatorfrequentie en de geconfigureerde poorten. Het is het beste om telkens kleine stapjes te doen en al doende te testen. Dat gebeurt door het toevoegen van debug-code die moet bewijzen dat elk bit dat u toevoegt echt werkt en vertrouwd kan worden – wanneer u een compleet programma schrijft en pas dan probeert het aan da praat te krijgen, dan bent u niet goed bezig.

De top-level toestandsmachine van het assembly-programma is te zien in **figuur 4**, terwijl de daadwerkelijke PIC-assemblycode in **listing 1** is afgedrukt. De programmalistings en bijbehorende besprekingen van de delen van het programma die de timing en input-/output-controle



Figuur 3. Schema van het (virtuele) verkeerslicht-systeem.



Figuur 4. Het volledige toestandsdiagram voor de verkeerslichten.

regelen, vallen helaas buiten het bestek van dit artikel; hetzelfde geldt voor een speciaal *test as you go*-programma en een mini-debugger voor de interne secondeteller. Die twee laatste zijn nuttig om het definitieve programma stapsgewijs op leerzame wijze op te bouwen. Alle (sub)routines en de bijbehorende toestandsdiagrammen kunnen echter gratis worden gedownload van [1].

200196-03

IN DE STORE

› **(Engelstalig) boek: "Programming Eight Bit PICs in Assembly as State Machines".**
Komt binnenkort beschikbaar: www.elektor.nl/books

WEBLINK

[1] **Projectpagina bij dit artikel**:
www.elektormagazine.nl/200196-03