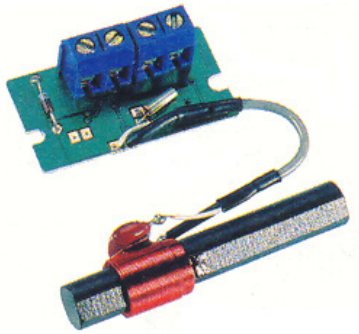
Tijdseinontvangst

Inleiding
Een klok die altijd heel precies op tijd is en zelf overschakelt naar zomertijd en wintertijd. Bij Frankfurt am Main staat de radiozender DCF77, die 24 uur per dag, jaar in jaar uit de juiste tijd uitzendt. Met een eenvoudige ontvanger is de tijd-informatie te ontvangen en in software om te zetten naar een heel precieze tijd en datum.

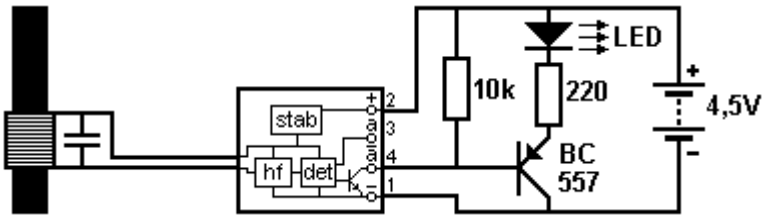
De ontvanger
Enige jaren geleden was het nog nodig zelf een ontvanger in elkaar te zetten, maar nu zijn er heel kleine ontvangerprintjes te koop die veel beter werken dan wat je zelf kunt maken. In de Conrad-gids van 2003 wordt zo'n printje aangeboden en bovendien hetzelfde printje in een weerbestendig kastje. Kijk ook eens op www.Conrad.nl en zoek met de eerste zes cijfers van het bestelnummer. Na het streepje staat het nummer van de catalogus.
DCF zendt uit op de wel erg lage frequentie van 77,5 kHz, hij maakt dus golven met een lengte van 3,9 km en dat zijn h é é l l a n g e golven. Deze golven worden in een gebouw sterk afgeschermd door de kooi van Farady van bouwstaal en betonijzer, want de gaten in de afscherming zijn heel klein ten opzichte van de golflengte. Voor een goede ontvangst zou "buiten" dus gewenst zijn, maar de ontvangertjes zijn zó gevoelig dat ze het binnen meestal toch wel doen.



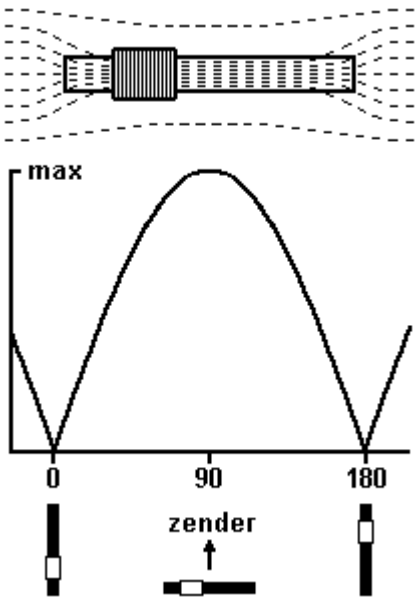
5 DCF-ontvangstprintplaat
Gebruiksklare DCF-ontvangstprintplaat met ferrietantenne · Voedingsspanning 1,2 - 15 V · Stroomverbruik 3 mA bij ingeschakelde ontvanger. Uitgangen Open collector NPN en Not inverted · Aansluiting via kroonsteen.
Bestnr. 64 11 38-44 **€ 10.50**

3 DCF-77 actieve antenne
Voor het aansluiten op alle starter- en applicatie boards. In een weerbestendige behuizing conform IP 54, overal te gebruiken, waar bijv. door gewapend beton, elektrische stoorvelden van machines en computers de DCF-ontvangst onmogelijk is. De ontvangstunit kan 360° worden gedraaid en kan hierdoor nauwkeurig worden gericht. Afm. 75 x 58 x 75 mm · Incl. wandhouder en PG7 schroefkoppeling · Aansluitschema.
Bestnr. 12 11 77-44 **€ 40.90**

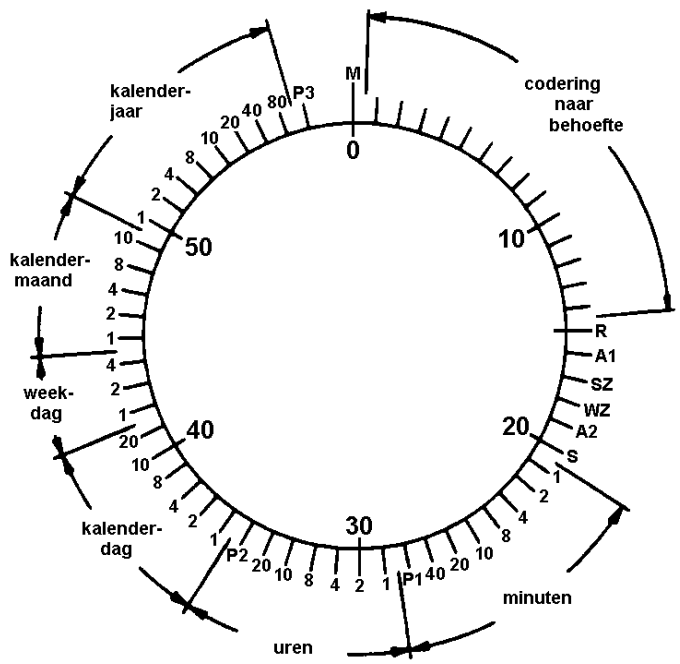
Eerst een experiment
De zender zendt secondenpulsen uit. Die ontvang je met het ontvangertje en je kunt ze zichtbaar maken met een LED. Maak eens een experimenteerschakeling waarmee je kunt rondlopen.



Nadat je de batterij hebt aangesloten zie je de LED in een secondenritme gaan knipperen. In het knipperen zit een zekere onregelmatigheid. Daarin zit informatie gecodeerd, verderop wordt dit verklaard. Leg de hele schakeling op een boek of in een koelkastbakje en loop er mee door het huis. Je krijgt daarmee gevoel voor de betrouwbaarheid van de ontvangst. De ontvanger heeft soms even tijd nodig om zijn versterking in te regelen.



Richten
Bij draaien van de ontvanger blijkt er steeds een stand te zijn waarin het niet of heel slecht werkt. Dat komt door de richtingsgevoeligheid van de antenne: het spoeltje met het staafje ferriet er in. De magnetische veldlijnen, die zich net als de golven in een vijver als cirkels om de zender heen uitbreiden, moeten door het spoeltje heen lopen. Het ferriet concentreert daarbij de veldlijnen. De ontvangststerkte van zo'n eenvoudige magnetische ferrietantenne verloopt sinusvormig als functie van de hoek met de richting waarin de zender zich bevindt. De ontvangst is minimaal als het staafje precies naar Frankfurt wijst.
De ontvanger heeft een enorme versterking en regelt die automatisch terug naar wat optimaal is, waardoor het bijna altijd wel werkt. Het nauwkeurige richten dat in de Conradgids wordt aangeprezen, is dan ook zelden nodig. Als hij ongeveer goed staat is het mooi genoeg. Bij een afwijking van 60° is de ontvangststerkte immers nog altijd 50%? Dat is ook wat je ervaart als je er mee rond loopt. Maar als het signaal heel zwak wordt en de versterking vèr wordt opgeregeld, gaat de storing uit de omgeving overheersen. Het nauwkeurig richten is dan ook alleen nodig wanneer er een sterke stoorbron in je directe omgeving is. Dan kun je door een precieze stand de storing op minimum draaien.



Tijd- en kalendercodering

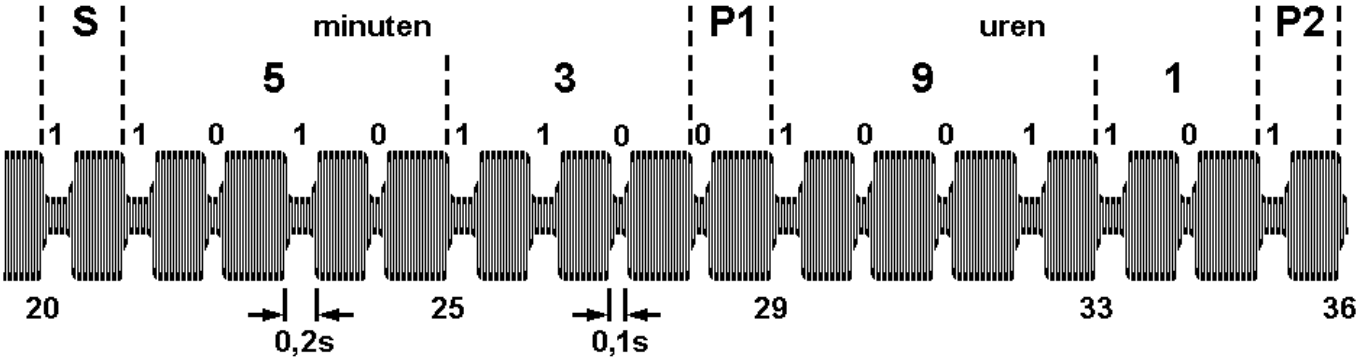
In iedere secondenpuls wordt één bit meegestuurd. Alle bits van één minuut samen coderen de tijd en de datum. Hiernaast zie je hoe de opeenvolgende bits in de minuut zijn toegewezen. Merk op dat het streepje tussen P3 en M ontbreekt. Dit is een secondenpuls die wordt overgeslagen op de 59e seconde. Bij de secondenpuls die daarop volgt begint er dus een nieuwe minuut.

Daarna wordt bit voor bit de tijd en datum doorgeseind van de daaropvolgende minuut, dus van na de over te - slaan secondenpuls.

Bitcodering

De zender maakt een wisselspanning van 77,5 kHz. Deze sinusvormige spanning wordt aan de zend-antenne toegevoerd. Daardoor ontstaat er een elektrisch en magnetisch veld dat wisselt met 77,5 kHz. Dat is het basisprincipe van elke radiozender.

De amplitude van de zendersinus is maximaal 100%. Elke seconde wordt de amplitude korte tijd verminderd tot 25%. Als je dat op een oscilloscoop bekijkt zie je elke seconde een korte insnoering van de sinustoon. Het begin van de insnoering geeft het beginmoment van die seconde aan. De tijdsduur van de insnoering is niet steeds hetzelfde. Een insnoering van 0,2 sec stelt een logische 1 voor en een insnoering van 0,1 sec een logische 0.



Hierboven zie je het zendersignaal gedurende 16 seconden, te beginnen om 19:34:20. De bitsgewijze code voor 19 uur 35 wordt van achter naar voor doorgeseind, dus eerst de eenheden minuten, dan de 10-tallen minuten, enz. De S is een start-bit, het is altijd 1. P1 is een pariteitsbit over de bitten van de minuten. Het totaal aantal enen in de minutenbits, plus bit P1, moet altijd even zijn. Evenzo is P2 een pariteitsbit over de urenbits en P3 is een pariteitsbit over de kalenderbits.

Zoals je ziet is de codering binair, maar dan wel per cijfer. Dit geldt voor de tijd en ook voor de kalender. Dit heet Binary Coded Decimal of BCD en is handig als je een klok wilt maken met cijferweergave. De codering voor de weekdag is in drie bits: 1 voor maandag t/m 7 voor zondag. Weekdag nul bestaat niet.

Voor wie verder het naadje van de kous wil weten:

M is de "Minutmarkierung", altijd een 0.

R = 1: de "Reserve Antenne" is in gebruik, bijv. bij onderhoud aan de "Haupt Antenne" → zwakker signaal.

A1 = 1: "Achtung", in het komende uur is er een sprong in de tijd van zomer- naar wintertijd of andersom.

SZ = 1 gedurende de "SommerZeit", GMT + 2 uur.

WZ = 1 gedurende de "WinterZeit", GMT + 1 uur. Met deze twee bits kan je klok zelf GMT uitrekenen.

A2 = 1: "Achtung", in het komende uur wordt er een schrikkelseconde ingevoegd.

Inbouwen

Het printje met losse ferrietantenne kun je inbouwen in een klein doosje, dat je op een geschikte plek neer legt. Bouw daar dan meteen de LED bij in, dat is gemakkelijk bij het uitrichten.

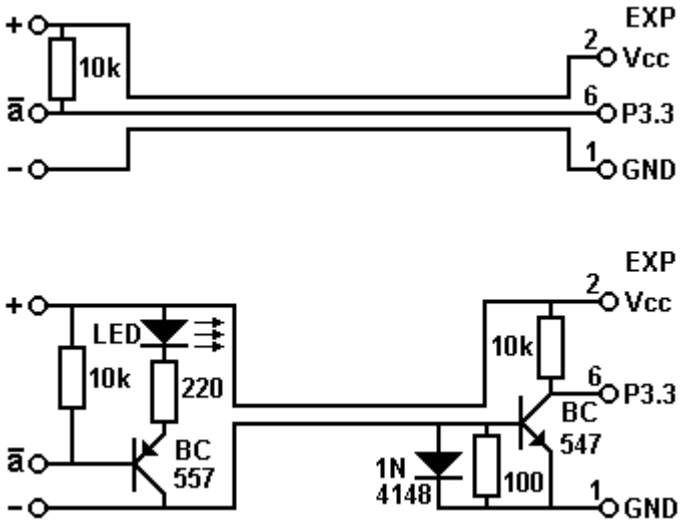
Let bij het inbouwen op een paar belangrijke zaken:

1. Geen gesloten metalen behuizing, want die schermt het magnetisch veld af.
2. Geen gesloten metalen ring om de ferrietstaaf, dat werkt als kortgesloten winding. Dus niet de staaf met een metaaldraadje ergens tegenaan binden. Geïsoleerd koperdraad (schellendraad) kan wel, als het met plastic-mantel en al in elkaar gedraaid wordt. Er blijft dan een elektrische onderbreking in de lus.
3. Het spoeltje zit expres asymmetrisch op de ferrietstaaf. De spoel is in resonantie met het condensatortje dat er tegenaan zit. Door schuiven op de staaf is de resonantie precies op 77,5 kHz gelegd. Lekker laten zitten.

Aansluiten op de µC

Je kunt een voedingsspanning van 5 volt (Vcc) betrekken van het processorprintje. Gebruik de geïnverteerde uitgang van de ontvanger. Deze uitgang kan direct op een poort-bit van de µC worden aangesloten, bijv. INT1 = bit 3 van poort 3 op de connector EXP. Tijdens de insnoering ziet de software hier een 0, daar tussen in ziet hij een 1. De LED is hier niet getekend.

Je hebt nu een driedraads verbinding. Zou het ook met twee draden kunnen? Voor een lange verbinding is dat wel handig. Bedenk dan dat de ontvanger zelf op 5 volt slechts 3 mA trekt, terwijl als je de LED toevoegt, deze 10 mA trekt zolang hij brandt. De opgenomen stroom varieert dus sterk in het secondenritme. De transistor bij de poort van de µC geleidt bij een stroom naar de ontvanger van meer dan 7 mA.



Software

Omdat het niet gegarandeerd is dat de zender altijd werkt moet onze klok uit zichzelf al redelijk goed lopen. We gebruiken dus de DCF-tijdcode alleen om hem gelijk te zetten en gelijk te houden. Commerciële DCF-klokjes zetten hun ontvangertje vaak maar één minuut per uur aan om te controleren of hij nog op tijd loopt en om de secondentik bij te trekken. Daar tussen in loopt hij op zijn eigen tijd. Dat spaart batterijen.

Autonome klok

Met behulp van een interne timer programmeren we een interrupt die precies 100 x per seconde de routine aanroept. Met een honderd-teller in software, de z.g seconden-timer, maak je dan een seconden-tik. Tel daarmee seconden, minuten, uren enz., waarmee er een klok is, die met de juiste snelheid loopt, maar nog niet op tijd.
Sla twee cijfers op in één register (byte) in twee maal 4 bits, in BCD dus. Dat is gemakkelijk voor het display, maar ook de tijdcode van DCF-77 is in BCD. Deze autonoom lopende klok loopt dus op het 12 MHz kristal van de 8051 en wijkt niet meer af dan een paar seconden per dag.

Synchroniseren

Het op tijd zetten doe je in stappen:
Secondentik: Kijk op het begin van de DCF-secondenpuls wat de stand van de seconden-timer is. Corrigeer dan de seconden-timer met stapjes van +1, 0 of –1, waardoor de seconden-timer synchroon gaat lopen met de secondenpuls van DCF-77. Je blijft dit voortdurend doen, waardoor hij synchroon blijft.
Seconden: Wacht op een ontbrekende secondenpuls. Lees dan gauw de stand van de autonoom lopende secondenteller uit en onthoud deze stand. Wacht dan opnieuw op een ontbrekende secondenpuls en vergelijk de stand van de secondenteller met de onthouden vorige stand. Als ze niet aan elkaar gelijk zijn doe je niets, dan was er duidelijk iets fout gegaan in de ontvangst. Zijn ze wel aan elkaar gelijk, dan was er precies één minuut tijdsverschil en je zet de secondenteller op 59. De seconden lopen nu op tijd.
Minuten, uren en kalender: Nu kunnen de nullen en enen gedecodeerd worden. Meet daartoe de lengte van de secondenpuls. Is deze:

- korter dan 0,05 sec dan keuren we de puls af,
- van 0,05 sec t/m 0,15 sec dan is het een 0,
- van 0,15 sec t/m 0,25 sec dan is het een 1,
- langer dan 0,25 sec dan keuren we de puls af.

In hulpregisters stellen we de nullen en enen samen tot bytes. Als er een bit was afgekeurd dan is het byte ongeldig, evenals wanneer het pariteitsbit niet klopt. Je kunt vaak het hoogstwaardige bit van een byte gebruiken om aan te geven of het byte geldig (0) of ongeldig (1) is. Dit bit wordt niet gebruikt bij een klok in BCD, omdat er slechts 59 seconden in een minuut zitten, 59 minuten in een uur en 24 uren in een dag.
Nadat de secondenteller van 59 naar 00 gegaan is kunnen de geldige bytes uit de hulpregisters worden overgenomen in de registers van de autonome klok. Het duurt dus een paar minuten na het aanzetten voordat de klok op de juiste tijd springt. Dat is juist leuk.

Evaluatie

Het synchronisatiemechanisme zoals hier beschreven is behoorlijk robuust. Zodra de klok eenmaal op de juiste tijd gesprongen is hoeft eigenlijk alleen de seconden-timer een beetje in de pas gehouden te worden met de ontvangen secondenpuls. Verder loopt de klok autonoom zijn seconden, minuten en uren, en zelfs de kalender als dat moet, wel af. Behalve twee maal per jaar. Uiteindelijk moet er een goed compromis gezocht worden om de klok na inschakelen snel genoeg op tijd te laten lopen maar hem daarna niet door radiostoring (biksem) zijn tijd onterecht te laten wijzigen. De overgang van winter- op zomertijd en andersom is hierbij een vervelende spelbreker.

Assembler code voor de 8051

Begin een programma altijd met een informatieve kop. We programmeren nu een interrupt op 1/100 sec., die een teller (R4 van bank 2) tot 100 laat tellen.

```
;*******************************************************
;***                  DCF-77 decoder                 ***
;***    Poort P3: Input - Output                     ***
;***            b3: DCF-77 ontvanger input           ***
;***                                                 ***
;***                     Software                    ***
;***                                                 ***
;***    Het programma loopt hoofdzakelijk            ***
;***    op de interrupt van timer 0.                 ***
;***    Deze loopt 100 maal per seconde af.          ***
;***                                                 ***
;***    Registerbank 2 = DCF-ontvanger:              ***
;***            R0 = puls-moment 0...99              ***
;***            R1 = positie van de missende puls    ***
;***            R2 = teller van de pulslengte        ***
;***            R3 = verzamelregister decoded bits   ***
;***            R4 = honderd deler binair            ***
;***            R5 = seconden in BCD                 ***
;***            R6 = minuten in BCD                  ***
;***            R7 = uren in BCD (24-uurs klok)      ***
;***            22 = bit-register DCF-decoder        ***
;***               b0 = 0: Sec-tik niet synchroon,   ***
;***                    1: Sec-tik gesynchroniseerd. ***
;***               b1 = 0: Seconden niet synchroon,  ***
;***                    1: Seconden gesynchroniseerd.***
;***               b2 = 0: Minuten niet synchroon,   ***
;***                    1: Minuten gesynchroniseerd. ***
;***               b3 = 0: Uren niet synchroon,      ***
;***                    1: Uren gesynchroniseerd.    ***
;***               b4 = 0: Niet in DCF-puls          ***
;***                    1: Wel in DCF-puls           ***
;***               b5 = 0: Te lang/kort DCF-bit      ***
;***                    1: Geen lengtefout           ***
;***            24 = -1,0,+1 voor sync 100-deler     ***
;***            25 = Teller aantal énen (pariteit)   ***
;***            26 = Gedecodeerde minuten            ***
;***            27 = Gedecodeerde uren               ***
;***                                                 ***
;*******************************************************

        ORG 0

TEMP    EQU 20H         ;Kladregister in de interrupt
SCALER  EQU 21H         ;Extra deler voor snellere timers
DCFB    EQU 22H         ;DCF statusbits
SCHUIF  EQU 23H         ;+1, 0 of -1 voor 100-teller
PARI    EQU 24H         ;Pariteits teller voor DCF
SMIN    EQU 25H         ;Store Minuten
SUUR    EQU 26H         ;Store Uren
SP      EQU 81H         ;Stack Pointer
TCON    EQU 88H         ;Timer Control
TR0     EQU 4           ;Timer 0 Run
TMOD    EQU 89H         ;Timer Mode
TH0     EQU 8CH         ;Timer 0 hoge byte
TL0     EQU 8AH         ;Timer 0 lage byte
P1      EQU 90H         ;Poort 1 (display)
IE      EQU 0A8H        ;Interrupt Enable register
ET0     EQU 1           ;Enable Timer 0
EA      EQU 7           ;Enable Alle interrups
P3      EQU 0B0H        ;Poort 3 (input-output)
PSW     EQU 0D0H        ;Programma Status Woord
ACC     EQU 0E0H        ;Accumulator
B       EQU 0F0H        ;B-register

;*** Sprongtabel ***

start   LJMP init       ;Spring naar initialisatie
        DB 0,0,0        ;Vector external interupt 0
        DB 0,0,0,0,0    ;Opvulling tot adres 0B hex
        SJMP int100H    ;Vector timer 0 overflow
        DB 0,0,0        ;Opvulling tot adres 10 hex

;*******************************************************
;***                                                 ***
;***   Interrupt routine van timer 0 loopt op 100 Hz, ***
;***   iedere 10 msec wordt dit programma afgewerkt.  ***
;***                                                 ***
;*******************************************************

int100H PUSH ACC        ;Bewaar accumulator-inhoud
        PUSH PSW        ;en het programma status woord
        CLR IE.EA       ;zet alle interrupts uit
        CLR TCON.TR0    ;en zet timer 0 even stil.
        MOV A,#0F7H
        ADD A,TL0       ;Tel 65536 - 10000 + 7 = D8F7
        MOV TL0,A       ;op bij de stand van timer 0
        MOV A,#0D8H     ;en plaats het resultaat weer
        ADDC A,TH0      ;in timer-registers TL0
        MOV TH0,A       ;en TH0.
        SETB TCON.TR0   ;Zet nu timer 0 weer aan
        SETB IE.EA      ;en zet de interrupts aan.

;Het even stilzetten van Timer 0 duurt 7 instuctiecycles,
;dit is reeds verdisconteerd in het optelgetal.

;En nou gaan we echt iets doen:

        DEC SCALER      ;Maak SCALER één lager
        MOV A,SCALER    ;en kijk wat hij geworden is.
        JNZ int100r     ;Als hij niet nul is: klaar.
        MOV SCALER,#06  ;Preset SCALER op 6, 2 of 1
        ACALL DCFdec    ;en werk de DCF-decoder af.
int100r POP PSW         ;Herstel oude registerbank
        POP ACC         ;en accumulator-inhoud.
        RETI            ;Einde interrupt-routine

;*******************************************************
;***                                                 ***
;***                  DCF-77 decoder                 ***
;***                                                 ***
;*******************************************************

DCFdec  MOV PSW,#10H    ;Schakel registerbank 2 in.
        ACALL dcfklok   ;Hoog de DCF-klok op met 1/100 s.
;       ACALL dcfpuls   ;Pulsdetectie.
;       ACALL dcftik    ;Synchroniseer de tik.
;       ACALL dcfbit    ;Evalueer bit en verwerk.
;       ACALL dcfdec    ;Decodeer minuten en uren.
;       ACALL dcfum     ;Neem uren/min over in DCF-klok.
        RET

;*** Laat de doorlopende klok lopen ***
dcfklok INC R4          ;Hoog de honderd-teller op
        CJNE R4,#100,dkret ;Nog geen honderd, return
        MOV R4,#00      ;Wel honderd, maak deler nul en
;       ACALL klokBCD   ;hoog de klok in BCD 1 sec. op.
dkret   RET

;*******************************************************
;***                                                 ***
;***                  INITIALISATIE                  ***
;***                                                 ***
;*******************************************************

init    MOV R0,#7FH     ;Zet het RAM van adres 7F
        MOV A,#00       ;tot adres 00 op nul, zodat
init1   MOV @R0,A       ;de klok na reset steeds weer
        DJNZ R0,init1   ;op nul begint.

        MOV SP,#2FH     ;Stackpointer van 30 omhoog.
        MOV PSW,#00H    ;Kies registerbank 0
        MOV TMOD,#01H   ;Timer 0 in mode 1
        MOV IE,#82H     ;Enable timer 0 interrupt
        SETB TCON.TR0   ;Zet timer 0 aan

;*******************************************************
;***                                                 ***
;***                 HOOFDPROGRAMMA                  ***
;***                                                 ***
;*******************************************************

begin   MOV A,14H       ;Pak 100-deler R4 van DCF-klok
        MOV P1,A        ;en stuur dat naar het display.
        SJMP begin      ;en opnieuw naar begin.

        END
```

Let op! Op sommige plaatsen, o.a. in het blok dat heet `DCF-77 decoder` begint een regel met **;** ten teken dat deze instructie er nog niet bij hoort. Als je die vergeet krijg je een foutmelding bij het assembleren.
Het programma tot zo ver bestaat achtereenvolgend uit:

1. Een sprongtabel naar `init` en naar `int100H`, de interruptroutine, die met 100Hz loopt.
2. De interruptroutine, die Timer 0 weer "opwindt".
3. Een extra deler voor snellere microcomputers. Bij `MOV SCALER,#06` moet het juiste getal staan.

4. Een aanroep van de DCF-decoder elke keer dat de extra deler op nul gestaan heeft.
5. De DCF-decoder bestaat uit een aantal stappen, voorlopig doen we alleen de eerste.
6. De eerste stap, die R4 tot 100 laat tellen.
7. De initialisatie van de interruptroutine op Timer0, die meteen door loopt in:
8. Het hoofdprogramma dat enkel de inhoud van R4 binair naar het display-tje stuurt.

Assembleer dit programma en programmeer de 8051 er mee. Als het loopt knipperen de segmentjes van het display, het middenstreepje precies 1 x per sec. Als dit zo werkt ben je een heel eind, want nu kunnen we in kleine stapjes uitbreiden.
Het is mogelijk dat het middenstreepje langzamer knippert, bijv. eens per 3 seconden of zelfs eens per 6 seconden. Dat hangt af van het type 8051. Verander dan in `MOV SCALER,#06` het getal in `#02` resp. `#01`.

De doorlopende klok
Haal de **;** bij `ACALL klokBCD` weg. Op dit punt gaan we verder:

```
        ACALL klokBCD   ;hoog de klok in BCD 1 sec. op.
dkret   RET

;*** Klok in BCD in R5 (sec), R6 (min), R7 (uur) ***
klokBCD MOV A,R5        ;Seconden uit R5 in A
        ADD A,#01       ;Hoog op door 1 erbij te tellen
        DA A            ;Decimal adjust i.v.m. BCD
        MOV R5,A        ;Plaats in seconden terug
        CJNE A,#60H,ktr ;Return als seconden geen 60 is
        MOV R5,#00      ;Seconden op nul, minuten ophogen

        MOV A,R6        ;Minuten uit R6 in A
        ADD A,#01       ;Hoog op door 1 erbij te tellen
        DA A            ;Decimal adjust i.v.m. BCD
        MOV R6,A        ;Plaats in minuten terug
        CJNE A,#60H,ktr ;Return als minuten geen 60 is
        MOV R6,#00      ;Minuten op nul, uren ophogen

        MOV A,R7        ;Uren uit R7 in A
        ADD A,#01       ;Hoog op door 1 erbij te tellen
        DA A            ;Decimal adjust i.v.m. BCD
        MOV R7,A        ;Plaats in uren terug
        CJNE A,#24H,ktr ;Return als uren geen 24 is
        MOV R7,#00      ;24 uur is 0 uur.
ktr     RET
```

In R5, R6 en R7 loopt dan een 24-uurs klok in BCD. We kunnen de seconden zien als we het hoofdprogramma even aanpassen:

```
begin   MOV A,15H       ;Pak de seconden van de klok,
        CPL A           ;inverteer alle bits
        MOV P1,A        ;en stuur dat naar het display.
        SJMP begin      ;en opnieuw naar begin.
```

Je ziet dan de segmentjes binair tellen, in a b c d de eenheden seconden, in e f g de tientallen. Merk op hoe hij na 5 9 terug gaat naar 0 0.

Cijferweergave
We kunnen ook de nibbles (halve bytes) als cijfers op het display weergeven. Eerst maar eens de eenheden seconden. Verander het hoofdprogramma:

```
begin   MOV A,15H       ;Pak de seconden van de klok,
        ACALL segl      ;zet LSN om in 7-segmentbeeld
        MOV P1,A        ;en stuur dat naar het display.
        SJMP begin      ;en opnieuw naar begin.

;*******************************************************
;***                                                 ***
;***           Conversie van nibble in A             ***
;***            naar segmentbeeld in A.              ***
;***                                                 ***
;*******************************************************

segh    SWAP A          ;Verwissel nibbles
segl    ANL A,#0FH      ;Least significant nibble
        INC A           ;skip de RET,
        MOVC A,@A+PC    ;pak segment image in A
        RET             ;en return.
        DB 0C0H,0F9H    ;0,1
        DB 0A4H,0B0H    ;2,3
        DB 099H,092H    ;4,5
        DB 082H,0F8H    ;6,7
        DB 080H,090H    ;8,9
        DB 088H,083H    ;A,b
        DB 0C6H,0A1H    ;C,d
        DB 086H,08EH    ;E,F

        END
```

Je ziet nu de eenheden seconden keurig lopen van 0 tot en met 9.

Alle cijfers
We willen de andere cijfers ook wel eens zien. Daarom breiden we het hoofdprogramma weer wat uit.

```
begin   MOV A,15H       ;Pak de seconden van de klok,
        ACALL segl      ;zet LSN om in 7-segmentbeeld
        MOV P1,A        ;en stuur dat naar het display.
        JB P3.2,begin   ;Geen knopje, dan begin opnieuw.
        ACALL ums       ;wel knopje, dan uren, min, sec.
        SJMP begin      ;en opnieuw naar begin.

uummss  MOV A,17H       ;Pak de uren van de klok,
        ACALL displ     ;en stuur naar het display.
        ACALL leeg      ;Leeg display
        MOV A,16H       ;Pak de minuten van de klok,
        ACALL displ     ;en stuur naar het display.
        ACALL leeg      ;Leeg display
        MOV A,15H       ;Pak de seconden van de klok,
        ACALL displ     ;en stuur naar het display.
        ACALL leeg      ;Leeg display

        RET

displ   MOV R1,A        ;even opslaan in R1,
        ACALL segh      ;converteer tientallen
        MOV P1,A        ;stuur naar het display
        ACALL wait1     ;en wacht 0,5 sec.
        MOV A,R1        ;herstel het BCD-getal,
        ACALL segl      ;converteer de eenheden,
        ANL A,#7FH      ;zet de punt aan,
        MOV P1,A        ;stuur naar het display
        ACALL wait1     ;en wacht 0,5 sec.
        RET

leeg    MOV A,#255      ;Leeg display
        MOV P1,A        ;stuur naar het display
        ACALL wait1     ;en wacht 0,5 sec.
        RET

;*******************************************************
;***                                                 ***
;***       Wachtlussen van 0,25 en 0,5 seconde       ***
;***                                                 ***
;*******************************************************

wait1   ACALL wait2     ;Wachtlus van 0,5 sec
wait2   MOV R0,#250     ;Wachtlus van 250 msec.
wait3   MOV A,#250      ;Wachtlus van 1 msec
wait4   DEC A           ;Wachtlus van 4 usec.
        NOP             ;vulling om hem 4 usec te maken
        JNZ wait4       ;Terugsprong elke 4 usec.
        DJNZ R0,wait3   ;Terugsprong elke 1 msec.
        RET             ;Einde na 0,5 seconde.
```

Hierna staat nog steeds de Conversie van nibble in A naar segmentbeeld in A.
Als je nu op het toetsje drukt zie je na elkaar de tientallen en eenheden uren, dan is het even donker, dan volgen de minuten en net zo de seconden. Bij de eenheden brandt steeds de punt.
Daarna springt het display weer op de eenheden van de seconden.

DCF decodering
Nu loopt de klok wel, maar nog niet op tijd. Daartoe moeten we de pulsjes van DCF decoderen. We beginnen met het op tijd laten lopen van de secondentik. Verwijder daartoe de **;** vóór `ACALL dcfpuls` in het stukje `DCF-decoder` en voeg de volgende routine toe direct boven `INITIALISATIE`, dus na:

```
ktr     RET

;*** Ontvang en verwerk de DCF-puls ***
dcfpuls JNB P3.3,dcfp1  ;Zitten we in een DCF-puls?
;*** Niet in de DCF-puls ***
        CLR DCFB.4      ;Markeer geen-DCF-puls in status.
        RET

;*** Wel in de DCF-puls ***
dcfp1   INC R2          ;Hoog pulslengte op.
        JB DCFB.4,dcfpr ;Zaten we al in de DCF-puls?
                        ;Ja, dan zijn we klaar.
        SETB DCFB.4     ;Nee, markeer DCF-puls in status.
;*** Voorflank van de DCF-puls !!
        MOV TEMP,R0     ;Vorige pulsmoment even onthouden
        MOV A,R4        ;Huidige pulsmoment
        MOV R0,A        ;opslaan in R0.
        CJNE A,TEMP,dcfpr ;Is dit gelijk aan de vorige?
        JZ dcfpr        ;Doe niets als A nul was.
        ADD A,#206      ;Kleiner of groter dan 50.
        JC dcfp0        ;Als hij kleiner was dan 50:
        DEC SCHUIF      ;Zet SCHUIF op -1
        RET
dcfp0   INC SCHUIF      ;Zet SCHUIF op +1
dcfpr   RET
```

Dit is een wat ingewikkelde routine, die kijkt of we in de DCF-puls zitten, of dat we er tussenin zitten. In de puls telt hij de lengte in R2. Dat gebruiken we later. Bovendien kijkt hij of de voorflank tussen 0 en 50, of tussen 49 en 100 van de honderddeler R4 ligt. In het ene geval maakt hij `SCHUIF` één hoger, in het andere geval één lager. Stond hij op 00, dan dus FF (kun je rekenen als –1) en aftellend. We kunnen dit proces volgen als we het hoofdprogramma even aanpassen:

```
begin   MOV A,SCHUIF    ;Pak het SCHUIF-getal,
        ACALL seg1      ;zet LSN om in 7-segmentbeeld
        JB P3.3,beg1    ;Kijk naar het DCF-puls bit,
        ANL A,#7FH      ;zet eventueel het puntje aan
beg1    MOV P1,A        ;en stuur dit naar het display.
        JB P3.2,begin   ;Geen knopje, dan begin opnieuw.
        ACALL ums       ;wel knopje, dan uren, min, sec.
        SJMP begin      ;en opnieuw naar begin.
```

Als we de 8051 programmeren met wat we tot hiertoe hebben gedaan zien we de punt van het display knipperen met de DCF-pulsjes. Als de punt niet knippert is de DCF-ontvanger niet (goed) aangesloten.
Bovendien zien we het cijfer vanaf 0 optellen of neertellen. Als het cijfer op nul blijft staan ben je vergeten de **;** bij `ACALL dcfpuls` weg te halen.

Secondetik bijtrekken
Haal de **;** weg vóór `ACALL dcftik` en voeg de onderstaande code toe na

```
dcfpr   RET

;*** Synchroniseer de secondentik ***
dcftik  MOV A,R4        ;Pak de stand van de 100-deler
        CJNE A,#50,dcftr ;Alleen als die op 50 staat
        ADD A,SCHUIF    ;Tel op +1, 0 of -1
        MOV R4,A        ;en zet terug in de teller
        MOV SCHUIF,#0   ;Schuiflengte op nul zetten.
        CLR DCFB.0      ;Reset de secondentik vlag.
        MOV A,R0        ;Pak het pulsmoment,
        ADD A,#161      ;95...99 wordt 0...4,
        JC dcft1        ;als R0 was 95...99,
        ADD A,#89       ;0...5 wordt 250...255,
        JC dcftr        ;als R0 was 0...5
dcft1   SETB DCFB.0     ;Ja, secondentik toch synchroon.
dcftr   RET
```

Hier wordt de 100-deler stapje voor stapje in de pas gebracht met de pulsjes van DCF. Bovendien wordt gekeken of de voorflank van de puls tussen stand 95 en stand 05 ligt. In dat geval vinden we dat de seconden voldoende op tijd lopen. Dit wordt gemeld met een 1 in `DCFB.0`. Het cijfer moet nu knipperen van 0 op 1 of op F. Na enig geknipper komt het cijfer tot rust op 0 en knippert nog heel af en toe op 1 of op F. De secondetik is dan helemaal op tijd. Als het cijfer nog steeds omhoog of naar beneden blijft tellen staat er nog een **;** bij `ACALL dcftik`.

Pulslengte bepalen
Nu gaan we de lengte van de puls, die in R2 staat bekijken en indelen in: ongeldig, nul of één.
Haal de **;** weg vóór `ACALL dcfbit`. We gaan dan verder na

```
dcftr   RET

;*** Bepaal de pulslengte en ontvang een bit ***
dcfbit  JB DCFB.0,dcfb1 ;Alleen als secondentik in sync.
dcfb0   CLR DCFB.5      ;Decodering ongeldig-bit.
        RET

dcfb1   CJNE R4,#90,dcfb2 ;Staat de 100-deler op 90?
        MOV R2,#00      ;Zet dan pulsduurteller op nul.
dcfb2   CJNE R4,#35,dcfbr ;Staat de 100-deler op 35?
        MOV A,R2        ;Evalueer de pulsduur,
;       JZ dcfb4        ;Ontbrekende puls!
        ADD A,#251      ;trek 0,05 sec af,
        JNC dcfb0       ;puls te kort, ongeldig!
        ADD A,#246      ;Trek 0,1 sec af,
        JNC dcfb3       ;het is een 0!
        SUBB A,#9       ;Trek weer 0,1 sec af,
        JNC dcfb0       ;puls te lang, ongeldig!
        INC PARI        ;het is een 1! Pariteitsteller+1
dcfb3   MOV A,R3        ;De 0 of 1 zit in de carry,
        RRC A           ;schuif hem vooraan in R3,
        MOV R3,A        ;het bitverzamelregister.
        RET

dcfbr   RET
```

Let op! Er staat weer een **;** voor de regel die springt op de ontbrekende puls. Dat komt hierna. Heb je die **;** er niet staan dan krijg je een foutmelding bij het assembleren.
Zoals je ziet wordt R2 op 0,9 sec op nul gezet en doen we de evaluatie op 0,35 sec. De 0 of de 1 wordt vooraan in R3 geschoven. Die willen we zien. Daarom wordt de eerste regel van het hoofdprogramma:

```
begin   MOV A,13H       ;Pak het DCF-register R3 en
        RLC A           ;schuif MSbit in carry.
        MOV A,#0        ;Maak A nul en schuif
        RLC A           ;de carry er van achteren in.
        ACALL seg1      ;zet LSN om in 7-segmentbeeld
```

De `ACALL seg1` stond er al. De rest blijft ook staan. Assembleer dit en zet het in de 8051. Je ziet dan na enige tijd (eerst moeten de seconden bijtrekken) even na een korte knippering van de punt een 0 en na een lange knippering een 1 op het display.

Minutentik op tijd zetten
Na de ontbrekende secondentik komt de minutentik. Die gaan we op tijd zetten, maar alleen als er twee pulsen ontbreken precies één minuut na elkaar. Haal de **;** weg vóór `JZ dcfb4` en voeg deze regel toe tussen de `RET` en de `dcfbr RET`:

```
;*** Een ontbrekende puls ***
dcfb4   CLR DCFB.5      ;Dit is een ongeldig bit.
        MOV TEMP,R1     ;Vorige missende puls bewaren,
        MOV A,R5        ;huidige seconden
        MOV R1,A        ;opslaan in R1.
        CJNE A,TEMP,dcfb5  ;Is dit gelijk aan de vorige?
;*** Twee maal achtereen gemiste puls op dezelfde seconde
        MOV R5,#59H     ;Zet secondenteller op 59
        SETB DCFB.1     ;Seconden in sync.
dcfb5   CJNE R5,#59H,dcfb6 ;Missing puls in 59e sec?
        RET

dcfb6   CLR DCFB.1      ;Fout! Seconden niet in sync
dcfbr   RET
```

De `dcfbr RET` stond er al. Nu veranderen we het hoofdprogramma weer zo dat het begint met:

```
begin   MOV A,15H       ;Pak de seconden van de klok,
        ACALL seg1      ;zet LSN om in 7-segmentbeeld
        JB P3.3,beg1    ;Kijk naar het DCF-puls bit,
        ANL A,#7FH      ;zet eventueel het puntje aan
beg1    MOV P1,A        ;en stuur dit naar het display.
```

Alleen de eerste regel is anders, dan zijn er regels weg te halen, de rest stond er al. Assembleren we dit en zetten het in de 8051, dan zien we de eenheden seconden tellen. Drukken we op het knopje, dan zien we de uren, minuten en seconden achtereenvolgend. Na enige tijd zien we een 9 op de ontbrekende secondenpuls. Probeer dit een paar keer en zie de seconden ineens op tijd komen.

Uren en minuten decoderen

Er is nu al veel gedaan. De nullen en de enen komen al in R3. We moeten alleen op het juiste moment naar R3 kijken.
Haal de **;** weg bij `ACALL dcfdec` en voeg deze regels in na

```
dcfbr   RET

;*** Decodeer minuten en uren ***
dcfdec  JNB DCFB.1,dcfb0   ;Alleen als seconden in sync.
        CJNE R4,#60,dcret  ;Alleen op 0,6 sec.
        CJNE R5,#19H,dcfd1 ;en in de 19e seconde:
dcfd0   MOV R3,#0       ;Bitverzamelregister op nul,
        MOV PARI,#0     ;pariteitsteller op nul
        SETB DCFB.5     ;en nog geen fouten tegengekomen.
        RET

dcfd1   CJNE R5,#20H,dcfd2 ;In de 20e seconde
        CJNE R3,#80H,dcfb0 ;Check het ontvangen startbit.
        RET

dcfd2   CJNE R5,#28H,dcfd3 ;In de 28e seconde,
        JNB DCFB.5,dcfd0   ;als er niets fout is gegaan
        JNB PARI.0,dcfd0   ;en als de pariteit oneven is:
        MOV A,R3        ;Pak de verzamelde bits,
        ORL A,#80H      ;maak het msb één
        MOV SMIN,A      ;en sla het op in SMIN.
        SJMP dcfd0      ;Klaar voor ontvangst van uren.

dcfd3   CJNE R5,#35H,dcret ;In de 35e seconde,
        JNB DCFB.5,dcret   ;als er niets fout is gegaan
        JB PARI.0,dcret    ;en als de pariteit even is:
        MOV A,R3        ;Pak de verzamelde bits,
        RR A            ;schuif één bit naar rechts
        ORL A,#80H      ;maak het msb één
        MOV SUUR,A      ;en sla het op in SUUR.
dcret   RET             ;DCF decoder
```

Kijk regelmatig naar het cirkeldiagram op pagina 2 om te begrijpen waarom het zo moet. De minuten worden opgeslagen in `SMIN` en de uren in `SUUR`. Die moeten daar blijven zitten tot na de minutentik, bijv. op de 15e seconde, want in die minuut zijn ze geldig.
Haal nu ook de **;** weg bij `ACALL dcfdec` en voeg deze regels toe na

```
dcret   RET             ;DCF decoder

;*** Neem tijd over in de DCF-klok ***
dcfum   CJNE R5,#15H,dcur  ;In de 15e seconde,
        CJNE R4,#25,dcur   ;op een kwart seconde,
        CLR DCFB.2      ;stijk de minutenvlag
        CLR DCFB.3      ;en de urenvlag,
        JNB SMIN.7,dcfu1   ;als msb van SMIN = 1 dan:
        MOV A,SMIN      ;Pak SMIN,
        ANL A,#7FH      ;maak het msb nul
        MOV R6,A        ;en zet het in de klok.
        SETB DCFB.2     ;Hijs vlag minuten bijgewerkt.
dcfu1   MOV SMIN,#0     ;Zet SMIN op nul (=leeg)
        JNB SUUR.7,dcur ;Als msb van SUUR = 1 dan:
        MOV A,SUUR      ;Pak SUUR,
        ANL A,#3FH      ;maak de twee msb's nul,
        MOV R7,A        ;Plaats uren in de klok
        SETB DCFB.3     ;Hijs vlag uren bijgewerkt.
        MOV SUUR,#0     ;en zet SUUR op nul (=leeg).

dcur    RET
```

De minuten en de uren worden apart over genomen, want het kan voorkomen dat er een fout opgetreden is in één van beide. Nu kun je weer assembleren en kijken wat het doet.
Na enige minuten druk je op het TEST-knopje en zie je de echte tijd langs komen. Tijd om je horloge en de klokken in huis op tijd te zetten!

De kalender

Nu je hebt gezien hoe de uren en de minuten uit de brei van enen en nullen tevoorschijn komen, mag je zelf bedenken hoe je dat met de kalender moet doen.

Een display

Alles loopt nog via het enkele display-tje op de print. Als je eenmaal zo ver bent zul je een groter, compleet display willen hebben. Daarvoor zijn verschillende mogelijkheden, die niet in dit document thuis horen. Als je zoiets gaat maken denk er dan aan dat het verstandig is om de inhoud van het display alleen te vernieuwen als er iets veranderd is aan de stand van de klok, dus als de seconden veranderd zijn.

Naschrift

De methode om steeds een klein stukje toe te voegen en dat dan uit te proberen werkt goed, veel beter dan alles maar eens in te typen en dan proberen de fouten te vinden. Dat laatste is erg demotiverend.
Merk ook op dat het hele programma tot zover slechts zo'n 430 bytes lang is. De Assembler geeft dat weer. Waar zouden dan toch al die megabytes van PC-programma's voor nodig zijn?