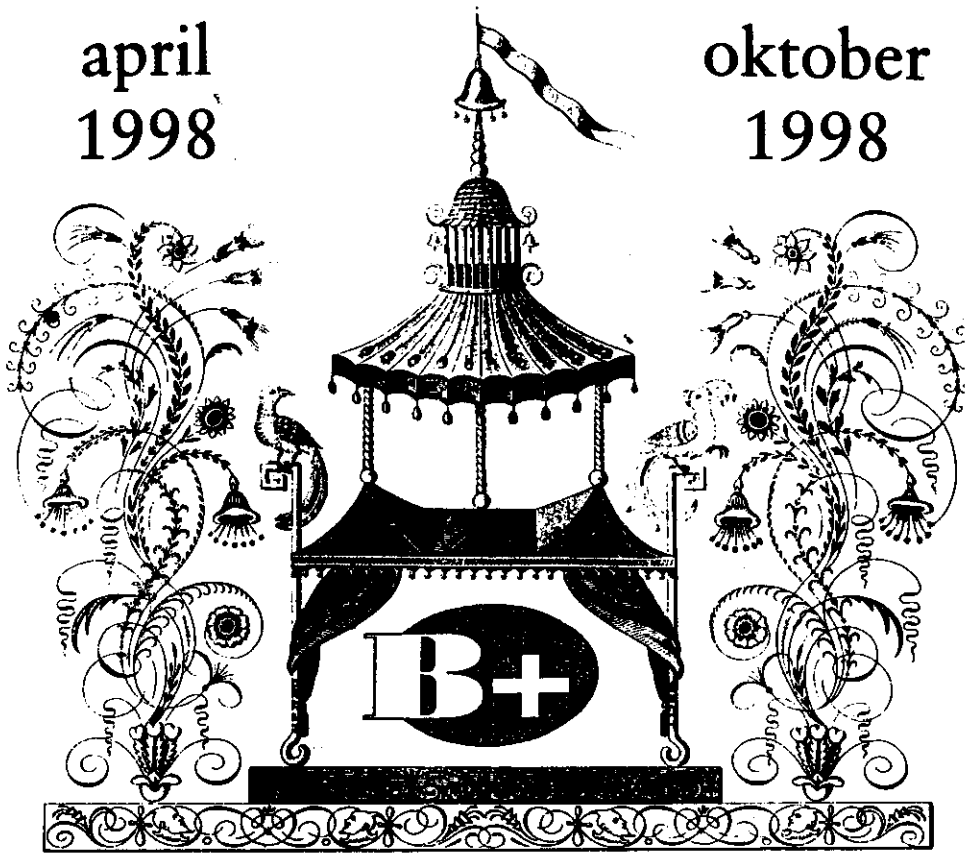
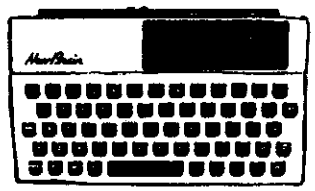
18
april
1998

17
oktober
1998



NewBrain dag

nonnenwater 8, gouda



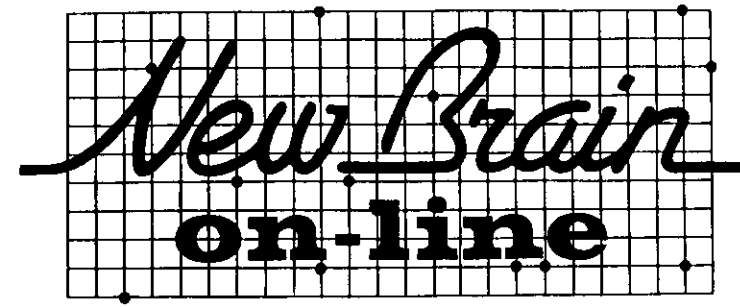
NewBrain-
gebruikersgroep
postbus 94494
1090 GL amsterdam

NewBrain on-line

uitgave van de NewBrain - gebruikersgroep

23

april *1998*



ten geleide

dit nieuwe nummer van newbrain on-line is geheel gewijd aan, hoe kan het ook anders: B+. u krijgt zowel achtergrondinformatie als praktische toepassingen voorgeschoteld

ton goossens behandelt in de vijfde aflevering van zijn serie 'de microcontroller' de aansluiting van een LCD-display, zodat het B+-bord u kan laten weten, wat het er nu zelf van denkt

van dré jansen krijgt u de opbouw van B+ en een inleiding op de digitale schakeltechniek. onder de titel 'looplamp' leert hij u met hexadecimale getallen omgaan. deze kennis komt u goed van pas in het volgende artikel, waarin abraham vreugdenhil zijn doolhofspel aan u voorstelt

we sluiten dit nummer af met een overzicht door jan wubben van de in de handel verkrijgbare interfacekaarten voor uw pc, en een artikel van dré jansen, waarin hij laat zien hoe je de timers in b+ gebruikt om een servomotor aan te sturen

menno stevens

NewBrain-
gebruikersgroep
postbus 94494
1090 GL amsterdam

voorzitter: **jan wubben,** (010) 4557698
secretaris: **maarten floor,** (020) 4964374
penningmeester: **menno stevens,** (020) 6924137
dré jansen, (0174) 414199
albert stuurman, (030) 2280163

postrekening 2505800 tnv hcc newbrain-gebruikersgroep

de newbrain-gebruikersgroep is een onderdeel van de
hcc hobby computer club
de molen 24, 3994 DB houten
inschrijvingsnummer kvk utrecht 82311

landelijke **newbraindagen**
zaterdag 18 april 1998 • zaterdag 17 oktober 1998
in het clubhuis van de afdeling gouda
nonnenwater 8, 2801 VA gouda

newbrain on-line
redactie: menno stevens, (020) 6924137
kopij voor nummer 23 graag naar het adres van de gebruikersgroep
of per e-mail aan mennostevens@hetnet.nl

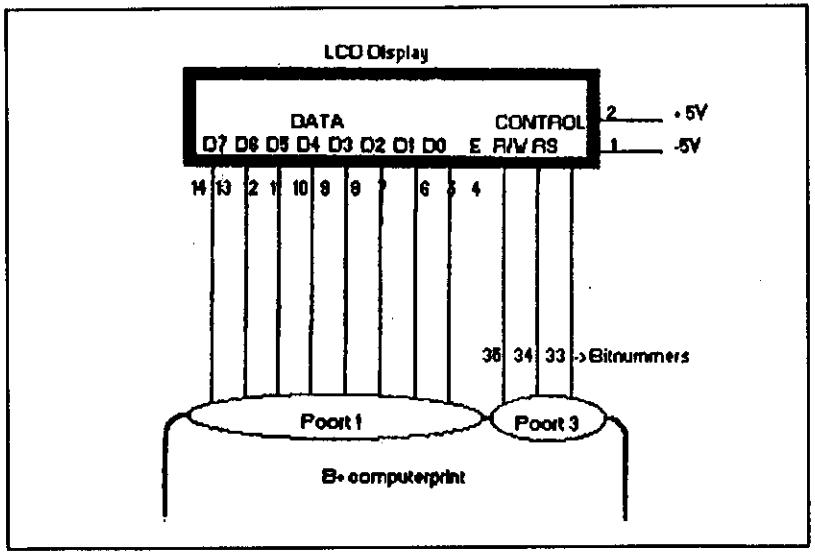
de micro-controller

een serieuze poging om de DOS-gebruikers enig inzicht in de werking van dit soort hardware te verschaffen
deel 5

Stel je voor dat je B+ gaat gebruiken voor een of andere (slimme) toepassing. Wat je ook doet, er zal een moment komen dat B+ iets terug wil melden: 'het lukt wel' of 'het lukt niet' of 'de motor heeft geen spanning'. Je bedenkt het maar, iedere besturing moet af en toe iets terugmelden. Dat kan natuurlijk met lampjes of piepers, maar het mooiste is natuurlijk, als je het gewoon kunt aflezen. En daarvoor is dan een LCD het meest eenvoudige en goedkoopste middel. Bij alle dumpzaken kun je voor een paar guldens wel een LCD kopen, bij Conrad in Rotterdam liggen voor minder dan veertig gulden splinternieuwe LCD's met 2 × 16 *characters*.

Het LCD gedraagt zich als een *device*, er zitten aansluitingen voor 8 databitjes op en voor drie controlesignalen. De databitjes zijn precies dezelfde als op de

B+-poort staan. Dat is eenvoudig, een 8-polig kabeltje en klaar. De controlesignalen zijn iets ingewikkelder. Er is een R/W-signaal. Dat staat voor *Read / Write*. Als R/W nul is, kan er naar het LCD worden geschreven; als R/W een is, kan er uit het LCD worden gelezen. Het tweede signaal is RS. Dat staat voor *register select*. Kennelijk kun je daarmee een register kiezen. Als RS nul is, kies je voor het besturingsregister; als het een is, kies je voor het dataregister. Het derde controlesignaal is E. Dat staat voor *enable*. De mensen die (vroeger) iets hebben gedaan aan Motorola processors zoals de 6800 of de beroemde 6809 weten precies, waar ik het over heb. Het E-signaal zorgt voor de dataoverdracht. Als je RS en R/W goed hebt gezet, kun je met het E-signaal de actie uitvoeren. Je zou kunnen zeggen dat het als een soort kloksignaal werkt.



Drie signalen die je met een programma kunt besturen, die kun je dus ook aan een poort vastmaken, nog een driepolig kabeltje erbij dus. Ook de voeding voor het LCD kan uit de B+-computer komen. Het LCD gebruikt net

iets meer dan niets, de belasting is niet te meten. Hieronder de tekening van de B+-computer met het LCD. Voor de poorten gebruiken we (deze keer) poort 1 en een paar draadjes van poort 3. In de tekening hebben we de andere aansluitingen van B+ voor de duidelijkheid weggelaten.

Poorten hebben nummers in B+, we hebben het daar al eens over gehad. De afzonderlijke pennen van de poorten hebben echter ook nummers, dat wil zeggen, ze zijn afzonderlijk te benaderen in het programma. Poorten heten *P* in B+, dus *P1* en *P2* zijn complete 8-bits poorten. *Q10* is het nulde bit van poort 1 en *Q17* is het zevende bit van die poort. Zo is *Q33* het derde bit van poort 3 enzovoort. De operaties die B+ kent met bits zijn (de n staat voor het nummer van het bit):

Qn +	daarmee wordt een bit 1 gemaakt
Qn -	daarmee wordt een bit 0 gemaakt
Qn!	daarmee wordt een bit 1 en direct daarna 0 gemaakt
Qn ,	test of een bit 0 of 1 is

Tenslotte hier een voorbeeldprogramma. Als u nooit iets heeft geprogrammeerd in B+, zal het enige aandacht vergen; voor de B+-programmeurs is het een simpel voorbeeld.

```
IF
X
                ;display data bit0,,bit7 = P1
                ;display control RS = bit 33, R/W = bit 34, E = bit 35
S1 = "LCD at P1, contr. at P3"
K1 = '38,01,0E,06,00'     ;set function, clear, set mode
C1      ,!C3
        Q34 + Q35!
        Q17 ,<$ Q34-  ,!C0

C2      ,!C3
        V1 - 1 ,#$  V2 - 8,#$  ,!C0
```

```
C3     Q35! ,!C0

B
Q33-   Q34- Q35-
       M12 = K1
       P1 = M12_ ,!C2 M12 :0,#$
Q33+
       M10 = S1
       P1 = M10_ ,!C1 M10 :0,#$
X
```

Het programma begint altijd met *IF*. Dat is een seintje voor de monitor: 'er komt een programma aan'. Het programma staat tussen twee grote *X*'en. Op de volgende regels staat enig commentaar, zodat je over enige tijd nog weet waar het over gaat.

De daarop volgende regel declareert een *string* (LCD at P1 contr. at P3). Het nummer van de string is willekeurig. Daarna worden er enige constanten vastgelegd met het *K*-commando. Deze constanten zijn eigenlijk commando's. Er moeten aan het LCD enige zaken worden uitgelegd, hoeveel regels, beginnen op de nulde positie, naar rechts schuiven enzovoort. Eigenlijk is het juist vastleggen van deze getallen het moeilijke deel van het hele programma. Je moet daarvoor informatie over het LCD hebben en die informatie op de juiste manier kunnen vertalen in getallen. Gelukkig hebben practisch alle LCD's dezelfde commando's. Ze gebruiken in het LCD ook vrijwel allemaal dezelfde chips, vandaar.

Als u toch een LCD op de kop tikt dat *per se* niet naar deze commando's wil luisteren, bel dan even; via het Internet zijn de juiste gegevens in een wip gevonden (bovenstaande gegevens kwamen op zondagmorgen als een file van 1,1 Mb uit Japan in minder als 8 minuten!), dat terzijde.

Daarop volgt *C1*, waarmee een subroutine wordt beschreven. In deze sub-

routine wordt een andere subroutine aangeroepen (*C3*) met het commando: *,!C3* Dat staat voor 'ga (onvoorwaardelijk) naar subroutine *C3*'.
In de volgende regel wordt bit *Q34* nul gemaakt. *Q34* is het R/W-bit van het LCD, en om te schrijven zouden we dat 0 maken, dat hebben we eerder al afgesproken. Daarna wordt bit *Q35* 1 gemaakt en 1 μs later weer 0 gemaakt.

De volgende regel test of bit *Q17* (dat is dus het zevende bit van poort1) al 0 is. Als dat niet zo is, dan wordt de test herhaald. Dat is de instructie *,<$* spring naar het begin van de regel ($) als het bit 1 is. Daarna wordt bit *Q34* 0 gemaakt en het programma keert terug naar de plaats waar het is aangeroepen.

De volgende subroutine is eenvoudiger. Na het aanroepen van *C3* volgt er een wachtlus van plm 8 μs. En terug naar het programmadeel dat de subroutine heeft aangeroepen.

De derde subroutine is nog eenvoudiger: zet bit *Q35* op 1 en direct daarop weer naar 0 en keer terug.

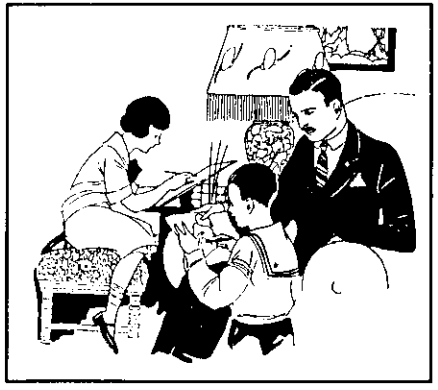
Het volgende commando is de *B*, die aangeeft dat het programma daar moet beginnen. Het zou erg mis gaan, als het programma zou beginnen met de subroutines, die door niemand zijn aangeroepen!

Het programma begint met de bits *Q33*, *Q34* en *Q35* op 0 te zetten, zoals we dat hiervoor hebben afgesproken.
Daarna wordt een memorypointer *M12* naar de constanten in *K1* gezet.

Op de volgende regel wordt *P1* geladen met de inhoud van de pointer *M12*. Het eerste commando (38) uit de constanten wordt dus in de poort

geladen. De waarde van de pointer wordt verhoogd met 1 door het commando '_' (de *underscore*). Daarna wordt de subroutine *C2* onvoorwaardelijk (!) aangeroepen, die ervoor zorgt dat het E-signaal gemaakt wordt. De informatie (38) wordt nu in het controleregister van het LCD geschreven. Tevens wordt er een paar milliseconden gewacht om te zorgen dat het LCD de informatie kan verwerken. Als het programma terugkomt uit de subroutine, wordt de inhoud van de pointer (die al is opgehoogd) getest op '00'. Als er geen '00' staat, springt het programma naar het begin van de regel en herhaalt de hele boel, totdat het laatste getal uit *K1* (de '00') wordt opgehaald. Het programma gaat dan verder op de volgende regel.

Nu wordt *Q33* '1' gemaakt. We hebben hiervoor al gezien dat *Q33* het RS-signaal is, we kiezen nu dus voor het dataregister. We zetten pointer *M10* naar de string *S1*.



Daarna laten we de inhoud van de pointer naar *P1* gaan. *P1* is zoals we eerder zagen, de datapoort voor het LCD (u mag natuurlijk ook best een andere poort kiezen). Het eerste karakter uit *S1* (de L) komt dus in het LCD. De pointer wordt verhoogd met de *underscore* en het programma gaat naar subroutine *C1*. In deze subroutine staat geen wachtlus, maar een routine die bit 7 van poort 1 leest om te zien of het LCD de letter al heeft verwerkt. Zolang bit 7 een 1 is, is het LCD *busy*. Je zou de routine *C1* natuurlijk ook kunnen vervangen door een wachtlus, maar nu is het netter: het LCD zegt zelf, wanneer we verder mogen.

Op het einde van de regel wordt getest, of de inhoud van de pointer al '00'

is. Als het niet zo is (de *,#$*) dan springt het programma weer naar het begin van de regel en herhaald alles tot de '00' wordt gevonden. Die '00' ziet u niet in de string staan, die wordt er door de compiler automatisch aan toegevoegd, tijdens het inlezen.

De daarop volgende *X* is het seintje voor de compiler om te starten, de invoer is compleet.

Mocht het zo zijn dat u bovenstaande direct volledig begrijpt, dan zou u de volgende regels op de plaats van de twee laatste regels kunnen zetten.

```
Q98-
Q98,>$     Q98 - P1 = P7 ,!C1 ,!$
```

Q98 is een zogenaamd *machinebit*. In de (huidige) versie van de compiler is hiervoor nog geen vertaling gemaakt. Het bit *Q98* is het bit dat aangeeft dat er een teken via de seriële poort is ontvangen.

Eerst wordt *Q98* 0 gemaakt. Daarna wordt getest of bit *Q98* 1 is, als dat niet zo is, opnieuw. Als het wel zo is, wordt eerst bit *Q98* weer 0 gemaakt en daarna wordt de inhoud van poort 7 in poort 1 geschreven. Poort 7 is de seriële poort. Dus als u op uw PC een toets intikt en verbonden bent met B+ (bijvoorbeeld met *Telix*), dan verschijnt het ingetikte teken op het LCD. De subroutine *C1* zorgt weer voor de E-puls, waardoor het teken echt wordt weggeschreven en er wordt gewacht tot het LCD zegt (bit *Q17*), dat het gelukt is.

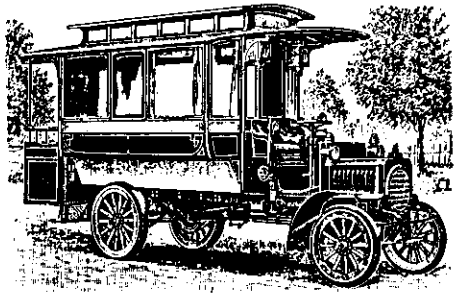
Ton Goossens
(079-3310893 a.u.b. na 18.00 uur)

de vorige delen van ton goossens' artikelenreeks over de microcontroller zijn afgedrukt in newbrain on-line 20 en volgende. de serie is eerder verschenen in de softwarebus van de hcc dos-gg

opbouw

de opbouw van B+

In een B+-controller zit de 'denker' ook wel CPU of processor genoemd. Er zitten twee typen geheugen in ROM, en RAM. Dit alles moet *per stuk* worden benaderd, hetgeen de nodige organisatie en management kost. Een foutje en . . . daaaag!

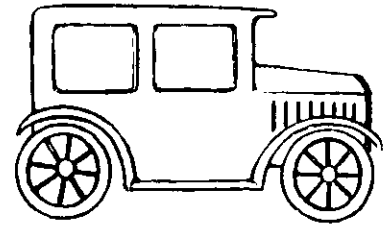


Wanneer iemand een brief aan mij (Dré Jansen) wil sturen, dan moet dat naar Dre.jansen@net.hcc.nl worden gestuurd. In de controller spreekt men ook van adressen, en op elk adres 'woont' iets of iemand. Wanneer je naar een adres gaat, dan neem je het openbaar vervoer, de bus. Ook in computerland heet dat de *bus*.

Wanneer je bij mij een pakketje wilt brengen, dan moet je dat in je handen meenemen. In computerland worden er geen fysieke zaken meegenomen, dat zou maar een rommeltje worden, toch gaat er 'iets' van het ene adres naar het andere. Dat 'iets' noemen we *data*. Dit 'data' heeft niets te maken met de datum op de verjaardagskalender, 28 mei 1994. Maar 28 mei 1994 en 18 april 1998 zijn zelf *wel* data, data zijn namelijk *gegevens*.

Aangezien er geen mannetjes zijn die de data van het ene adres met de bus naar het andere adres vervoeren, gaat dat in computerland op een wat andere manier, het principe is echter gelijk. Een *adresbus* is een bundel draadjes, die op boven geschreven manier op binaire wijze een adres aanwijzen. De *databus* is eveneens een bundel draden, waarop ook een binaire waarde staat, die wordt dan ook van het ene adres naar het andere adres gebracht. Dus de ene bundel geeft de woonplaats aan, de andere bundel het pakketje dat naar die woonplaats moet.

Dit is een gigantisch werk, omdat er veel data van de ene plek naar de andere moeten. Dit gaat met grote snelheid en er kan wel eens wat fout gaan, da's nou het menselijke trekje van B+.



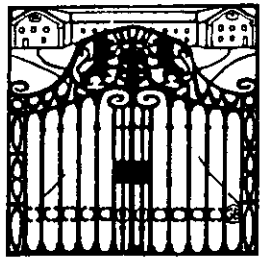
Nu heb ik een adres en data, maar wat moet ik ermee? Moet ik dit brengen, of halen, of even wachten, of weggooien? Dus stap ik naar m'n manager die op alles een antwoord weet. Deze manager is een vriendelijke, geduldige en vooral begripsvolle man, die mij begeleidt met deze moeilijke taak. Hij schudt mij de hand en vertelt wat ik moet doen. Ook hier weer door middel van een bundel draadjes, de *handshake*, waarmee wordt aangegeven, wat ik met de data moet doen. Bij een *read*-opdracht moet ik lezen en bij *write* moet ik de data afleveren. Een adres kan zoals gezegd een geheugenplaats zijn, maar ook een poort of een timer.

Dré Jansen

digitaal

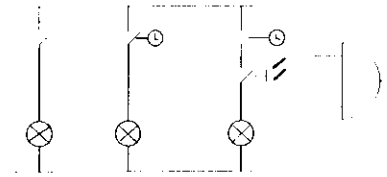
een klein stukje digitale schakeltechniek

Digitale techniek is niets meer, maar ook niets minder dan schakeltechniek. Een schakelaar kan aan staan of juist uit. Dan kun je dat weergeven met cijfers. Voor de hand liggen dan de waarden 0 en 1. Dan valt er nog af te spreken, welk cijfer we gebruiken voor *aan* en met welk cijfer we de *uit*-situatie aanduiden. Daarover zijn de meningen verdeeld, zodat beide worden toegepast. Wanneer we kiezen voor 1 = aan, dan spreken we van *positieve logica* en als juist voor 0 = aan wordt gekozen dan spreekt men van *negatieve logica*. In principe maakt het niet zo veel uit, maar je moet wel bij een eenmaal gemaakte keuze blijven, anders gaat het fout. Toch worden die beide logicavormen soms door elkaar gebruikt, maar dan wordt het wel degelijk goed vermeld.



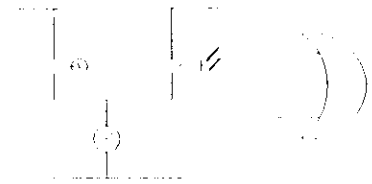
Nu de schakelaartjes. Één schakelaartje is voldoende om een lamp aan en uit te schakelen, maar stel je eens voor, dat die lamp alleen mag branden na 16:00 uur en om 24:00 weer uit moet. Een schakelklok is wat je nodig hebt. Nu wil je ook nog, dat de lamp alleen aan is, als het donker is, dus 's winters om 16:00 uur, 's zomers om 21:00 uur. Regelmatig de schakeltijden veranderen wil je niet, dus een licht gevoelige schakelaar in serie met de schakelklok zetten. Hiermee gaat de lamp alleen aan als de tijd tussen

16:00 uur en 24:00 uur licht *en* als het donker is.

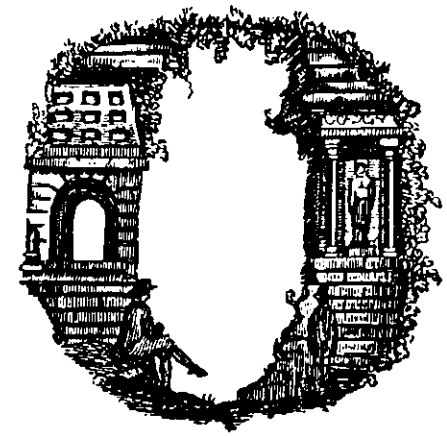


Zo'n serieschakeling van twee schakelaars heet in computerjargon een *en*-poort. De stroom kan door 'de poort' als schakelaar 1 *en* schakelaar 2 gesloten zijn.

Zo kunnen deze schakelaars ook naast elkaar staan, wanneer we weer dezelfde componenten gebruiken, met dezelfde instellingen.



De lamp zal altijd om 16:00 uur aangaan, de lamp zal om 24:00 uur *niet* uitgaan, want het is nog steeds donker. De tweede schakelaar staat nog aan. De lamp zal pas uitgaan wanneer de zon weer op komt.



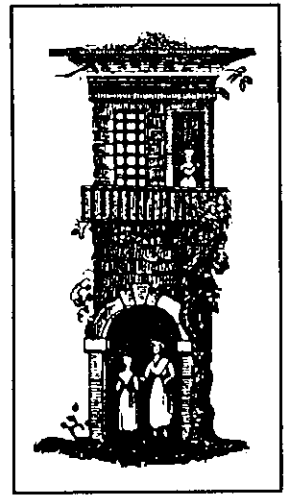
Een parallel schakeling van twee schakelaars. In computerland spreken we dan van een

of-poort, omdat de lamp aan is, als schakelaar 1 *of* schakelaar 2 aan is.

Wanneer we deze getallen in een tabel zetten, dan krijgen we de volgende tabellen:

de seriële ofwel *en*-tabel	de parallelle ofwel *of*-tabel
0 × 0 = 0	0 + 0 = 0
1 × 0 = 0	1 + 0 = 1
0 × 1 = 0	0 + 1 = 1
1 × 1 = 1	1 + 1 = 1

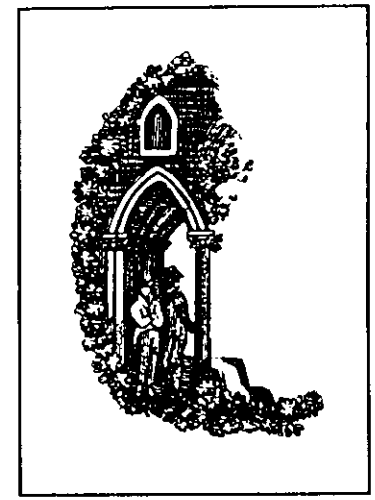
Dit soort tabellen noemt men ook wel *waarheidstabellen*. Er staan rekenkundige tekens in, de *en* ofwel *and* wordt vergeleken met vermenigvuldigen, en als je dat dan doet, dan lijkt het daar ook op. De *of* ofwel *or* lijkt op optellen.



Er is aan het einde een klein optelfoutje gemaakt, want 1 + 1 = 2 en geen 1. Dat komt doordat een schakelaar niet meer dan aan kan zijn. Hij is aan of uit, niet meer, maar ook niet minder. Zaken als 'dubbel aan' zijn niet mogelijk. Een computer doet niet anders dan dit soort tabellen nalopen.

Oplettende lezertjes, hebben gemerkt dat de nullen in de *and*-tabel zich als een *or* gedragen, en de nullen van de *or*-tabel gedragen zich als een *and*. De negatieve logica!

Dan is er nog een derde poort, en wel de *exor*. Dit is heel bijzondere uitvoering van de *or*-poort. *Exor* staat voor *exclusive or*. Het bijzondere van deze poort is, dat in tegenstelling tot de gewone *or*-poort, dat wanneer er twee ingangen hoog zijn, de uitgang juist laag is. De waarheidstabel is gelijk aan die van de gewone *or*, maar de laatste regel is: 1 + 1 = 0. Jawel, nul!

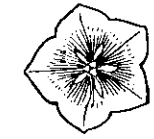


Waar dit element goed voor te gebruiken is, komt vanzelf wel, als je wat langer bezig bent in deze materie. De *exor* is een handige bouwsteen.

Als laatste is er de *not*-poort, eigenlijk een heel simpele. Als de ingang hoog is, is de uitgang laag. Is de ingang juist laag, dan is de uitgang hoog. Een nogal tegendraadse bouwsteen. Juist deze *not-gate* komt veel in combinatie voor met andere bouwstenen. Vaak zit hij al gekoppeld aan de uitgang van andere bouwstenen. Een *and* met *not* heet *nand*. Een *or* met *not* heet *nor*; zo is er ook de *exnor*.

Als je de uitgang van de waarheidstabel van bijvoorbeeld een *and* bekijkt, dan is die van een *nand* juist tegengesteld. Alleen als beide ingangen hoog zijn, is de uitgang laag. De reden van deze combinatie is, dat interne elektronica makkelijker te maken is en sneller werkt. Het hoe en waarom valt net buiten het kader van dit verhaal.

Dré Jansen



looplamp

Het was zaterdag 1 november 1997. Ik mocht een dagje van de *Roboticanen* bijwonen. Daar bleek dat het hexadecimale gebeuren verwarring schiep. Vreemd, want als je kunt lezen, dan beschik je over het onderscheidsvermogen tussen vele karakters: hoofdletters, kleine letters, cijfers, leestekens, etcetera — dan het hexadecimale wereldje, dat bestaat uit vier bitjes.

Als je daar lampjes aan hangt, dan kun je die bitjes zichtbaar maken, meer

niet. Met vier lampjes kun je 16 verschillende combinaties maken hexadecimaal (*hex* = 6 en *deci* = 10). Ik ben er van overtuigd dat dit zeer gemakkelijk te onderscheiden is, temeer daar er gebruik gemaakt wordt van karakters die reeds lange tijd geleden geleerd zijn.

Omdat er 16 karakters zijn, had men naast de 10 cijfers kunnen kiezen voor: & # @ en [, maar daar is geen vaste volgorde in aan te geven. *Dus* zouden we die moeten leren, en van leren krijg je hoofdpijn! Het alfabet heeft al een bekende volgorde. Natuurlijk kunnen we de cijfers 10, 11, 12 gebruiken, maar dan komt het probleem: is dat een 1 en 2 of staat hier 12? Daarom is gekozen voor de eerste zes letters van het alfabet.

Eerst wat over getallen: wanneer wij een getal schrijven, dan is dat in het decimale stelsel, omdat onze verre voorvaderen tien vingers hadden en om deze reden tot dit systeem zijn gekomen. Een cijfer in een getal heeft een dubbele functie: *grootte* en *plaats*.

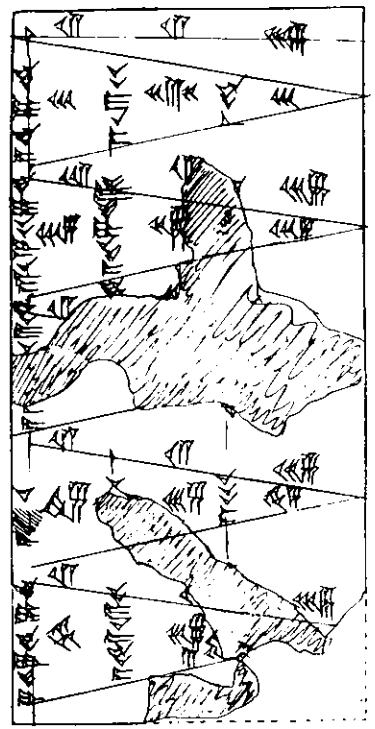
De grootte is duidelijk, een 7 betekent een volle hand, en twee vingers van de andere hand. De plaats is die van deze 7 in het getal 372. De twee betekent twee vingers. De 7 betekent 70 vingers, en wel om de reden dat dit cijfer op de tweede plaats van rechts staat. Zo ook de 3, die eigenlijk 300 betekent. De totale waarde van dit getal is dan ook 300 + 70 + 2 = 372. Eerste klas lagere school, zou je zeggen, maar let op, er is meer in het getal.

In een decimaal stelsel (*deci* = 10) kennen wij 10 karakters lopend van 0 tot en met 9, en dus *niet* van 1 tot 0. Een telefoonschijf is dus eigenlijk verkeerd, de nul geeft 10 pulsjes, en is daarmee dan ook geen nul.

De positie geeft in een decimaal stelsel de macht van 10 weer. Op de meest rechtse plaats, die LSB (*least significant bit* = minst belangrijke bit) wordt genoemd, is dit 10^0, hetgeen de waarde 1 vertegenwoordigt. Elk getal 'tot de nulde macht' is namelijk 1. Het tweede cijfer van rechts is 10^1 en geeft

de waarde 10 aan. Elk getal 'tot de eerste macht' is dit getal zelf. Op de derde plaats komt het kwadraat, op de vierde plaats de derde macht enzovoort (probeer dat maar eens uit op je zakjapanner). In het voorbeeld van hierboven 372 is dit de som van

$$3 \times 10^2 + 7 \times 10^1 + 2 \times 10^0 =$$
$$300 + 70 + 2 = 372$$



Waar in het decimale stelsel met slechts 3 posities een hoeveelheid van 372 kan worden weergegeven, zijn er in het binaire stelsel wel 9 posities nodig. De binaire weergave van 372 is: 101110100. Hier worden, in tegenstelling tot het decimale stelsel, de machten van het basiscijfer 2 genomen (*bi* = 2, denk maar eens aan het Engelse woord *bicycle* = tweewieler).

De meest linkse 1 is het MSB (*most significant bit* = meest belangrijke bit) en komt overeen met 2^8 omdat dit op de negende plaats staat. Wanneer we zo door gaan ontstaat het volgende sommetje:

$$1 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 +$$
$$0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$

$$256 + 0 + 64 + 32 + 16 + 0 + 4 + 0 + 0 = 372$$

Moeilijk? Nee toch! Ik weet dat je normaal al kunt werken met getallen stelsels die veel moeilijker zijn, stel eens voor, een mix van getallenstelsels,

een 60-tallig stelsel, uitgedrukt in decimale getallen, vermengd met een 12- of 24-tallig stelsel, ook weer uitgedrukt in het tientallig stelsel. Da's pas moeilijk! Nog moeilijker wordt het, als je deze waarden ook vectorisch gaat weergeven. Waarbij deze vectoren ook nog verschillende rotatiesnelheden kennen . . . Vraag het maar aan een kind, dat leert klokkijken.

Om wat duidelijker te zijn, worden dit soort getallen vaak in groepjes van vier neergeschreven. 101110100 wordt 1 0111 0100. Hier kan men dan het hexadecimale getal al herkennen, want als men dan die groepjes van vier afzonderlijk bekijkt dan zijn dit hexadecimale getallen. Met al die enen en nullen is een vergissing snel gemaakt. Door het toepassen van hex wordt die foutkans verkleind (hexadecimaal wordt vaak afgekort tot het woord hex, maar is *geen* 6-tallig stelsel).

looplamp

Een eenvoudige decoder kan op een 7-segments *display* eenvoudig deze hex-waarde weergeven. Wanneer je de getallen 1-2-4-8 achtereenvolgens presenteert, dan is hier nauwelijks een looplampje in te herkennen. Gebruik je gewone LED's, dan zie je onmiddellijk een looplampje. Het uitlezen is dan mogelijk wat minder, maar dat went snel.

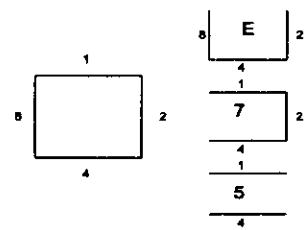
Ook het aansturen van diverse zaken is duidelijker met LED'jes dan met getallen. Natuurlijk moet je in B+ getallen blijven aansturen. Of . . . nee, met het Q-commando kun je per bitje schakelen. Dit droge verhaal over getallen is nodig om te weten hoe een computer 'denkt'.

Dré Jansen

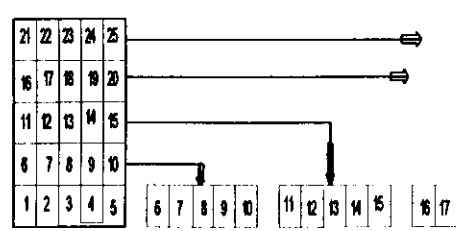
doolhof

Wanneer je je in een doolhof bevindt, dan sta je als het ware in een klein kamertje. Vanuit dat kamertje kan je natuurlijk zien vanwaar je gekomen bent, en waar je wel of niet naar toe kan gaan, afhankelijk van de aanwezigheid van een of meerdere deuren. Verder kun je *niets* zien. Zo ook in dit doolhof. De kamers zijn simpel, vierkant, in elke muur *kan* een deur zitten. *Een* deur is er altijd, namelijk jouw toegangsdeur.

De muren hebben een nummer: noord = 1, oost = 2, zuid = 4 en west = 8. Wanneer je die waarden wat nauwkeuriger bekijkt, dan zijn dit de vier laagste bits van een byte (*nibble*).



Zoals je ziet, een geheel geloten vierkantje heeft de waarde *F*. Een vierkantje dat aan de bovenzijde open is, heeft geen 1 in zich, dus waarde *E*. Wanneer de linkerzijde open is, ontbreekt de 8 en zal de waarde van de ruimte 7 zijn. Ook kunnen er twee deuren in de kamer zitten, zelfs géén deuren: 0.



Vele van die kamertjes geven een reeks getallen. Op deze manier staan de getallen achter elkaar. Een positie naar boven is een verschuiving van 5 posities naar rechts. Een stap naar links is -1, naar rechts +1.

Wanneer er geen deur is, brandt het lampje; alleen een donkere lamp geeft een deur aan.

Hieronder staat het volledige programma, met tussendoor de uitleg. Het begint met de 5 doolhoven:

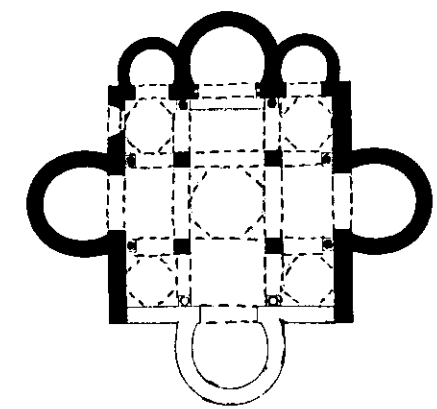
```
K1="1,E,C,6,D,6,9,3,A,C,3,C,4,1,1,6,A,9,6,C,2,9,7,B,1A,B"
K2="4,C,5,5,7,E,8,5,5,5,2,A,D,5,6,A,9,5,5,3,A,1C,5,5,5,3"
K3="3,C,6,E,C,6,A,B,A,A,A,8,5,2,A,A,A,E,9,3,A,9,3,1C,5,3"
K4="5,C,7,C,5,7,8,5,2,C,7,B,E,9,1,6,C,0,5,7,A,1A,9,5,5,3"
K5="4,E,C,5,7,E,A,A,C,5,2,9,1,3,C,3,D,4,5,1,6,D,1,5,16,B"

S1="\n Doolhof \V6"
L7 P1=10 Q44,>$ V6=5
```

wachtlus wacht op indrukken van de vuurknop van de joystick.

```
V7-1,#$ V8-20,#$ ;wachtlus ter compensatie van contactdender.
Q44,>@ V6-1,#$ V6=5 ,!$ ;doolhofkeuze, zolang de vuurknop in is.
,!S1 V6:1,=L8 V6:2,=L9 V6:3,=LA V6:4,=LB ,!LC
```

Melding van de doolhofkeuze naar de PC. De gekozen doolhof (de string K in memory) wordt in het geheugendeel geplaatst, dat later uitgelezen wordt voor het eigenlijke spel.

```
L8 M1=K1 ,!LD ;doolhof 1
L9 M1=K2 ,!LD ;doolhof 2
LA M1=K3 ,!LD ;doolhof 3
LB M1=K4 ,!LD ;doolhof 4
LC M1=K5 ,!LD ;doolhof 5
LD V2+M1
```



Van de gewenste doolhof wordt het starthokje, de ingang, aangewezen. Dit hokje wordt aangewezen door het eerste cijfer in de K-string. Voor doolhof 2 hokje 4.

```
L1
P1=M1 V5=M1 V5&F0 V5:10,=LE
```

Het eigenlijke programma start hier. Controle of de 'uitgang' al bereikt is. Dit doet men door te controleren, of er een 1 staat in het hoogste nibble. Vooralsnog is dit de enige functie van dit nibble. In de toekomst kan dat gebruikt worden voor een grotere doolhof, sleutel ophalen en meenemen, etcetera.

```
P4:FF,=$ V3=P4 ;wachten op richtingkeuze van de joystick.
V7-1,#$ V8-4,#$ ;wederom een contactdenderwachtlusje.
P4:FF,#$ V3:FE,=L3 V3:FD,=L4 V3:FB,=L5 V3:F7,=L6 ,!L1
```

Hierboven wordt de gekozen richting van de joystick gedecodeerd. Dat dit een beetje vreemd aandoet, komt door de waarden die de joystick uitstuurt bij gekozen richtingen. Nu het eigenlijke gebeuren:

```
L3 V4=M1 V4&1 V4:1,=L1 V2+5 ,!L1 ;omhoog (+5)
L4 V4=M1 V4>1 V4&1 V4:1,=L1 V2+1 ,!L1 ;naar rechts (+1)
L5 V4=M1 V4>2 V4&1 V4:1,=L1 V2-5 ,!L1 ;omlaag (-5)
L6 V4=M1 V4>3 V4&1 V4:1,=L1 V2-1 ,!L1 ;naar links (-1)
```

Deze vier bewegingen moeten als volgt worden gelezen: wanneer men in het starthokje, of een ander hokje staat, dan mag men alleen maar naar een kamer via een deur, een 0 dus. Als de richting naar rechts gaat, dan wordt er een controle op het tweede bitje gedaan, immers dat tweede bitje bepaalt de aanwezigheid van een deur in wand 2. Er wordt een verschuiving van 1 positie naar rechts gedaan (*V4>1*). Dan wordt er gecontroleerd, of dit een 1 is (*V4:1*). Zo ja, dan er is tegen een muur gelopen en gaan we naar *L1*, de plaats waar gewacht wordt op een joystickbeweging. Zo niet, dan wordt *V2* met 1 verhoogd, en bevindt men zich in de naastgelegen kamer.

Die kamer wordt zichtbaar op het display, immers *V2* wordt nu maar *P1* gestuurd. *V2* is het laagste nibble dat door de inhoud van *M2* wordt aangewezen. De lampjes geven nu de nieuwe situatie weer. Je kunt terug, maar ook uitzien naar een nieuwe weg.

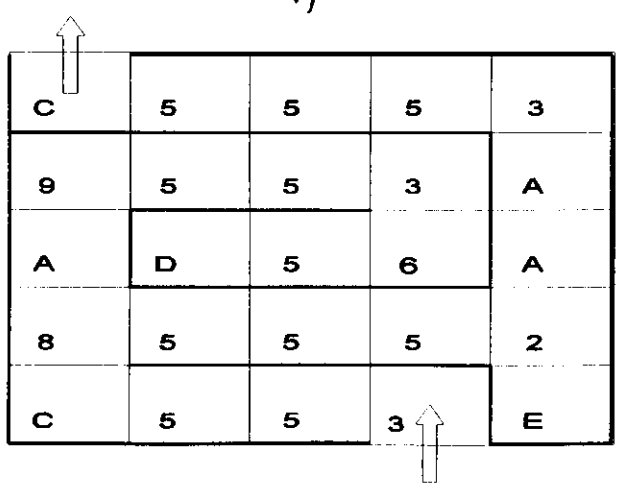
Bij een beweging naar links, wordt er eerst drie maal geschoven, immers dan pas is de betreffende wand op positie 1 aangekomen. Ook dan wordt er weer gekeken, of men erdoor kan, of niet; de verdere verwerking is gelijk aan een beweging naar rechts. Toch niet helemaal, immers ook de geheugenpointer *V2* wordt met 1 verminderd. Gevolg is, dat ook de informatie naar buiten 'in welke kamer sta ik?' wordt aangepast.

Nu gaan we naar boven: dat gaat op iets andere wijze. De bovenste wand was al op de gewenste positie, waar controles worden uitgevoerd. Dus nu niet schuiven, maar meteen controleren of de waarde 1 (geen doorgang) of 0 (doorgang) is. Dan moet de memorypointer met 5 omhoog worden gebracht, immers de kamerinformatie van boven elkaar gelegen ruimten liggen 5 memoryplaatsen van elkaar. Vandaar dat bij de opwaartse beweging *V2+5* vermeld staat, tenminste, als men een deur aantrof. Bij een neerwaartse beweging -5 bij doorgang.

```
LE V9=A ; tja, daar sta je dan, je bent eruit!
V7-1,#$ V8-1,#$ P1=F
LF V7-1,#$ V8-4,#$ P1=0 V9-1,=L7
V7-1,#$ V8-4,#$ P1=F ,!LF
```
Tijdlusje: de lampjes gaan even uit en weer aan, ten teken dat de taak volbracht is. Ze knipperen van vreugde.

```
; V1 en V2 en M1 doolhofdata
; V5 einddata
; K1 - K5 verschillende doolhoven
; P1 uitvoerLED's (bit 0-3 doolhofwand, bit 4 groene LED)
; P4 invoer joystick
X
```

Abraham Vreugdenhil



interface

verkrijgbare interfacekaarten

De volgende interface-kaarten zijn in de handel verkrijgbaar. Bij elk artikel staat waar het te koop is. De prijzen zijn inclusief btw.

Centronics 8-kanaals relaisinterface

Via de standaard printerpoort is deze kaart aan te sturen. Er is een externe voeding nodig, die 12 V - 500 mA levert (niet inbegrepen). Printerkabel nodig voor de verbinding. De relais zijn voorzien van schroefklemmen.
Bestelnummer 96 77 18-55 bij Conrad. Prijs *f* 99,95

Windows-software voor relaisinterface

Voor de Conrad 8-kanaals interface is software beschikbaar waarbij een 50-tal schakeltijden in te stellen zijn.
Bestelnummer 96 19 57-55 bij Conrad. Prijs *f* 19,95

PC-schakelinterface

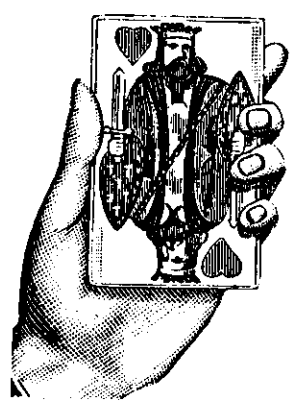
Interne interfacekaart voor 8-bits PC-AT-slot. Met deze kaart kunnen via de 25-polige *female* subD-connector 24 in- en uitgangen geschakeld worden, bijvoorbeeld voor een modelspoorbaan, of een huisalarminstallatie. Voorbeeldsoftware in bijgeleverde handleiding.
Bestelnummer 96 77 00-55 bij Conrad. Prijs *f* 69,95

PC-schakelinterface

Idem als boven; echter met 48 kanalen.
Bestelnummer 96 58 42-55 bij Conrad. Prijs *f* 99,95

Parallelle externe interfacekaart

Een *Velleman*-kit voorziet in een interfacekaart, die via een *flat cable* op de centronics-printerpoort kan worden aangesloten. De printer is weer op deze kaart aan te sluiten. De aansluiting naar de computer verloopt via opto-couplers. Hij kan geprogrammeerd worden in Turbo-pascal, Q-Basic en C++. Voorbeelden en testroutines worden op flop meegeleverd. De kaart heeft 16 optisch gescheiden digitale in- en uitgangen en 9 analoge uitgangen. Als dat niet genoeg is, kunnen nog tot drie *slave*-kaarten worden aangesloten.



Bestelnummer K8000 Computer Interfacekaart a *f* 215,- bij DIL electronica in Rotterdam, maar ongetwijfeld ook elders.

De volgende kaarten zijn als extensie op de K8000 leverbaar:

K6714, 16-kanaals relaiskaart
K6710 en K6711, 15-kanaals afstandsbediening
K2607, thermometeradapter
K6700 en K6701, tweedraads communicatiekanalen (maximaal 16)
K2633, 4-kanaals relaiskaart
K2634, 4-kanaals triackaart enzovoort

De prijzen en verkrijgbaarheid van de laatste set zijn niet eenduidig bekend.

Tenslotte: op aanvraag zijn er fotokopieën beschikbaar voor een RS232-interface (uit juli 1987) voor 8 parallelle digitale in- en uitgangen (beschrijving en schema; geen printlay-out!)

Ook is er een schema met uitvoerige beschrijving en printlay-out beschikbaar voor een relaiskaart (8 relais) op een centronics-printerpoort, gedateerd juli 1994.

Jan Wubben

servomotor

servomotoren aansturen

In een B+ zitten meerdere timers, die het klusje kunnen klaren om een servomotor aan te sturen. De timer wordt aangestuurd vanuit een 16-bits register, dat maximaal de waarde FFFF kan bevatten. Een tel duurt een microseconde. Het complete tellen van deze teller duurt dan 65535 microseconden. Een servo wil een frequentie zien van 50 Hz. Een volledige periode duurt 20 milliseconde. Dat is dan 20.000 microseconde. Trek die twee getallen van elkaar af, en je krijgt het sommetje:

65535 - 20000 = 45535

Als je die waarde weer terugrekent naar hexadecimaal, dan kom je op 8BC0. Om een periode van 20 milliseconde te krijgen, moet je het timerregister vullen met de waarde 8BC0. Die teller telt tot aan FFFF en genereert dan een *interrupt*. Daarna wordt de teller weer met 8BC0 geladen en het hele spel start opnieuw.

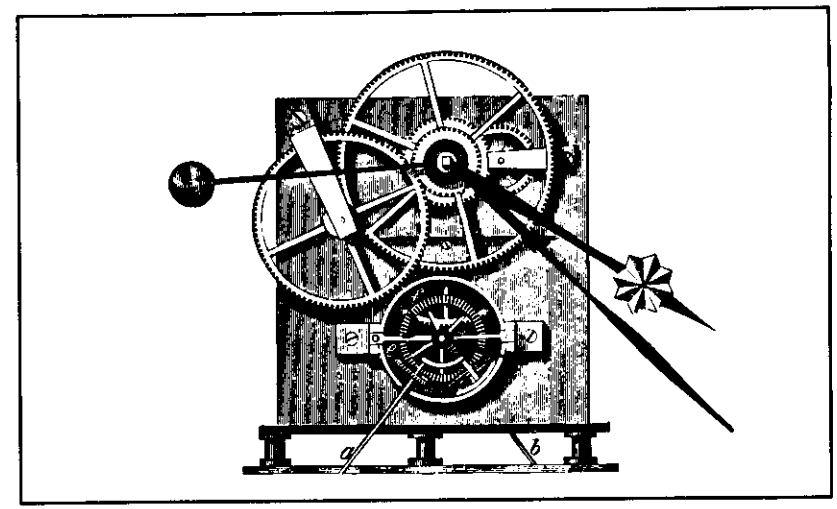
In *V1* plaats je het hoge byte, in *V2* plaats je het lage byte. Dan zet je de timer en het interrupt aan.

```
X
V1=8b
V2=C0
TC0=M1
```



```
Ti+
It2+
```

Nu wordt elke 20 milliseconde een *call* gedaan, de interrupt laat zich herkennen als *Ct2*. In deze call kun je enkele variabelen uitlezen, die de stand.



Dan kun je een bitje aanzetten, en na een zekere tijd zet je dit bitje weer uit. De tijd van het 'aan zijn' bepaalt de stand van het scheepsroer. Een *dutycycle* van 50 % geeft de middenstand aan.

```
Q10+ V1=10 V2=11
V10-1,#$ V11-1,#Q10-
```

In deze subroutine kun je meerdere servo's sturen, als je er maar voor zorgt binnen 20 ms terug te zijn, zodat de nieuwe interrupt ontvangen kan worden. Gedurende de rest van de tijd, kunnen er allerlei zaken geregeld worden.

Dit kan ook hardwarematig, de B+ blijft dan volledig beschikbaar voor andere zaken. Een dutycycle van 20 ms is namelijk ook met een 555 timertje te maken.

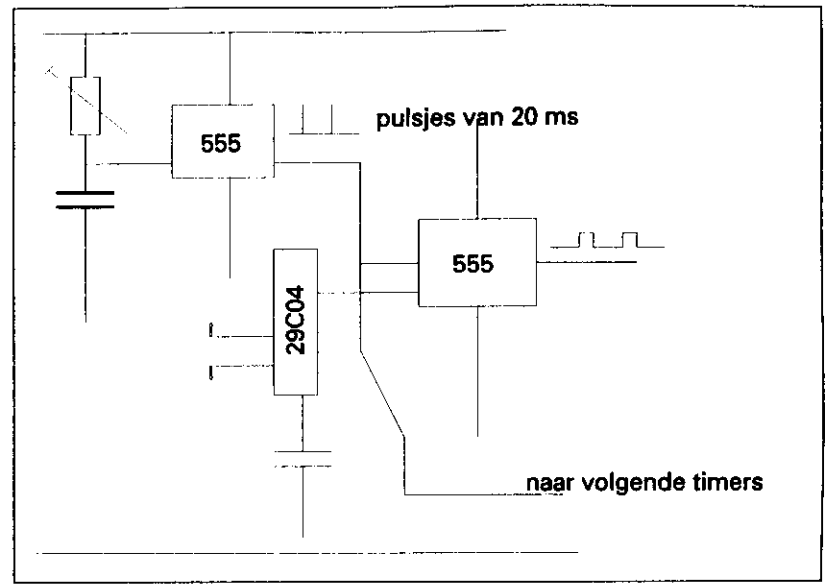
Laat een timer elke 20 ms een triggerpulsje maken. Dat triggert een tweede timer, de eigenlijke dutycyclemaker. Daarbij heb je natuurlijk een variabele weerstand nodig. Deze potmeter is dan een 29C04 chip. Hieraan zitten twee pootjes, die de omhoog- en omlaagbeweging van de potmeter regelen.

Deze potmeter komt in meerdere variaties, 100 stappen en 64 stappen. Een versie van 16 stappen wordt niet meer gemaakt. De waarden zijn er in 500 K, 100 K, 10 K en 1 K ohm. Die laatste heeft beperkingen in vermogen. Een 29C04 is net als een NE555 een 8-pins IC'tje. Kost ongeveer 4 gulden.

Met B+ is het nu heel simpel om voor het eigenlijke starten een reeks van 50 pulsjes te sturen naar de potmeter(s). De servo's staan dan in het midden, of een andere stand als u dat wilt.

Er zijn servo's die een afwijkende frequentie willen. Dat is eenvoudig te merken: als een dutycycle van 50 % niet de middenstand aangeeft, dan moet je de frequentie verhogen of verlagen.

De eerste 555 is ervoor om elke 20 ms een puls te genereren; servo's willen nu eenmaal een frequentie van ongeveer 50 Hz zien. Die kan dus vast worden ingesteld.



Een tweede 555 timer wordt met die eerste getriggerd, en is dus een *monostabiele vibrator*, waarvan met een potmetertje de pulsduur gevarieerd kan worden. Dat kan met een elektronische potmeter die middels twee pulsdraden gestuurd wordt, een voor de opwaartse en een voor de neerwaartse richting. In plaats van twee 555'jes kan je ook een 556'je nemen. Bij meerdere servo's is de gezamenlijke triggerpuls voor alle servo's te gebruiken.

Dré Jansen

inhoud on-line 23