ROBO-BITS-54

Jaargang14, nummer 3, september 2011

Afz.hcc Robotica gg, p.a. Henk de Gans,Koelmanhof 2 3861GG Nijkerk..

hcc robotica

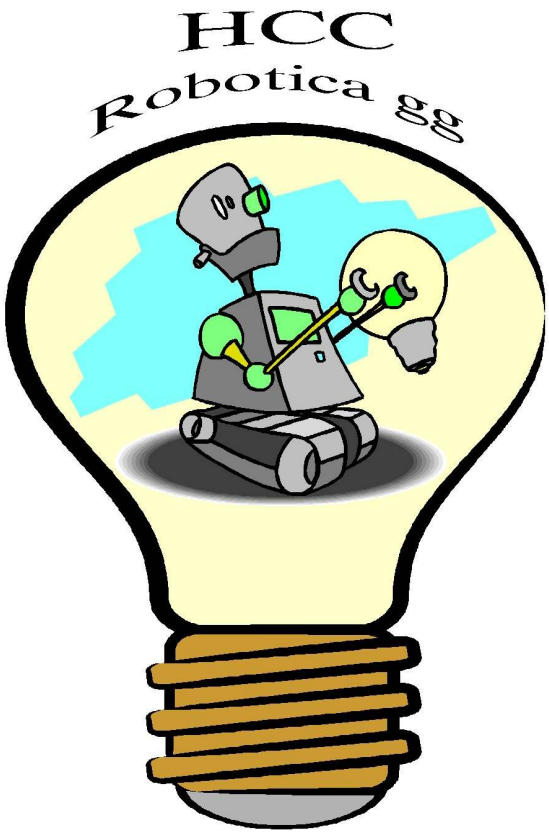
De Robobits is een uitgave van de hcc!robotica gebruikers groep, en wordt vier keer per jaar als PDF beschikbaar gesteld aan de leden. hcc!robotica is een onderdeel van de hcc! (hobby computer club), een vereniging van bijna 150.000 leden.

==========

Redactie adres: H.J. de Gans, Koelmanhof 2, 3816GG Nijkerk. hj.de.gans@gmail.com
Tekst aanleveren in WORD of platte tekst in ASCII. Afbeeldingen los er bij in JPG, GIF of BMP formaat.

==========

inhouds opgave:



REDACTIE

Voor u ligt dan al weer RoboBits 54. Wel een beetje laat voor een september nummer, maar we wilden even wachten tot na de open dag en roborama wedstrijd.

Deze dag was bijzonder geslaagd! Veel belangstelling en ook veel deelnemers uit diverse groeperingen. Op onze website vind u een uitgebreid foto verslag van deze dag. Ook verschijnt binnen kort de nieuwsbrief met daarin een verslag. Vandaar dat in deze robobits er verder niks over te zien en lezen is!

Deze keer ook niet veel kopij, maar de kopij die aangeboden is, is wel zeker de moeite van het lezen waard! Mijn dank aan Joep en Dré voor hun inbreng! En op de valreep nog een update van de robobits i.v.m. een ingezonden stuk van Iwan, ook hij : hartelijk dank!

Ik wens u veel leesplezier.

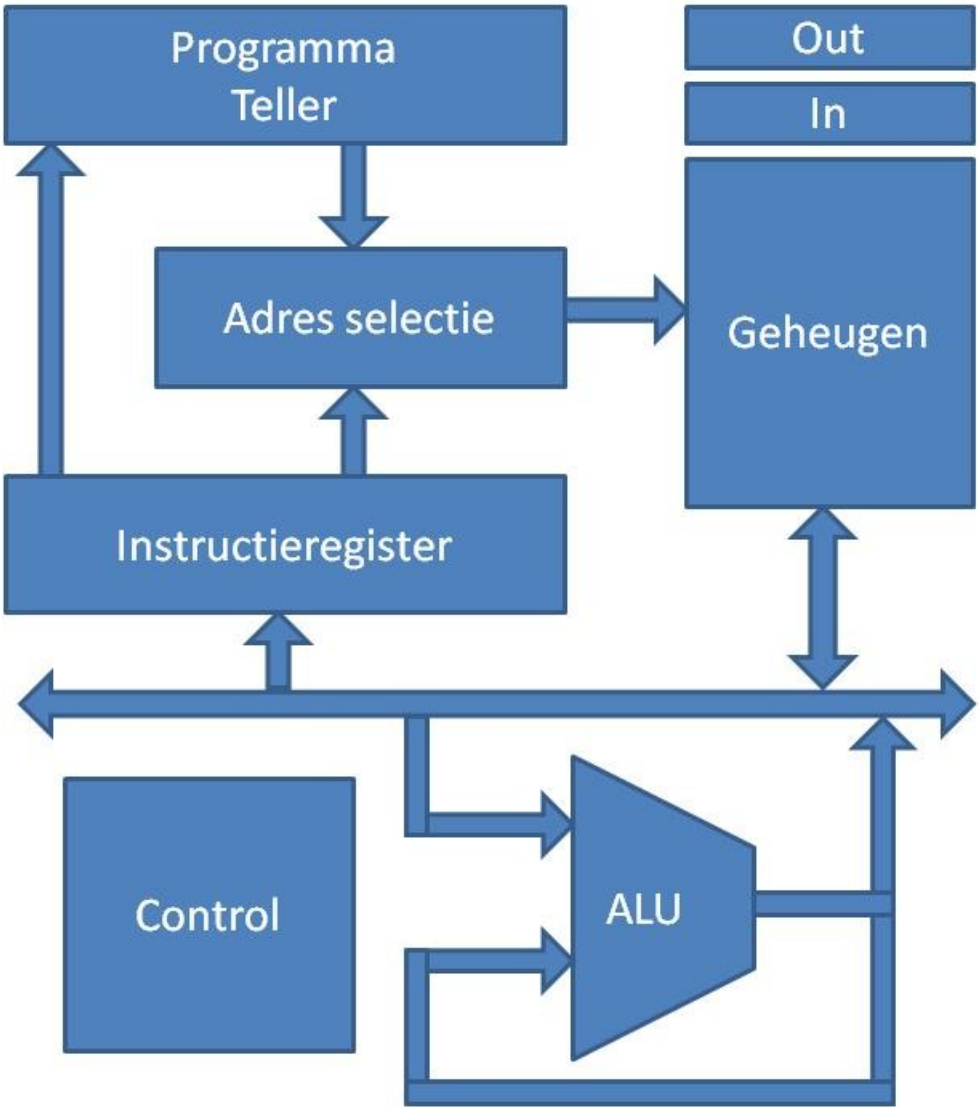
Uw redacteur
Henk de Gans

PS kopij voor het decembernummer graag voor half december inleveren!!

bouw je eigen computer deel 2

Bouw je eigen computer – deel 2

Beste Roboteer,
In het vorige deel van dit artikel hebben we gekeken naar de 'little man computer' – een model waarmee de interne werking van de computer wordt uitgelegd aan de hand van de taken die een denkbeeldig mannetje diep in de computer uitvoert.
Uiteindelijk hebben we dit vertaald naar een model zoals hieronder staat:

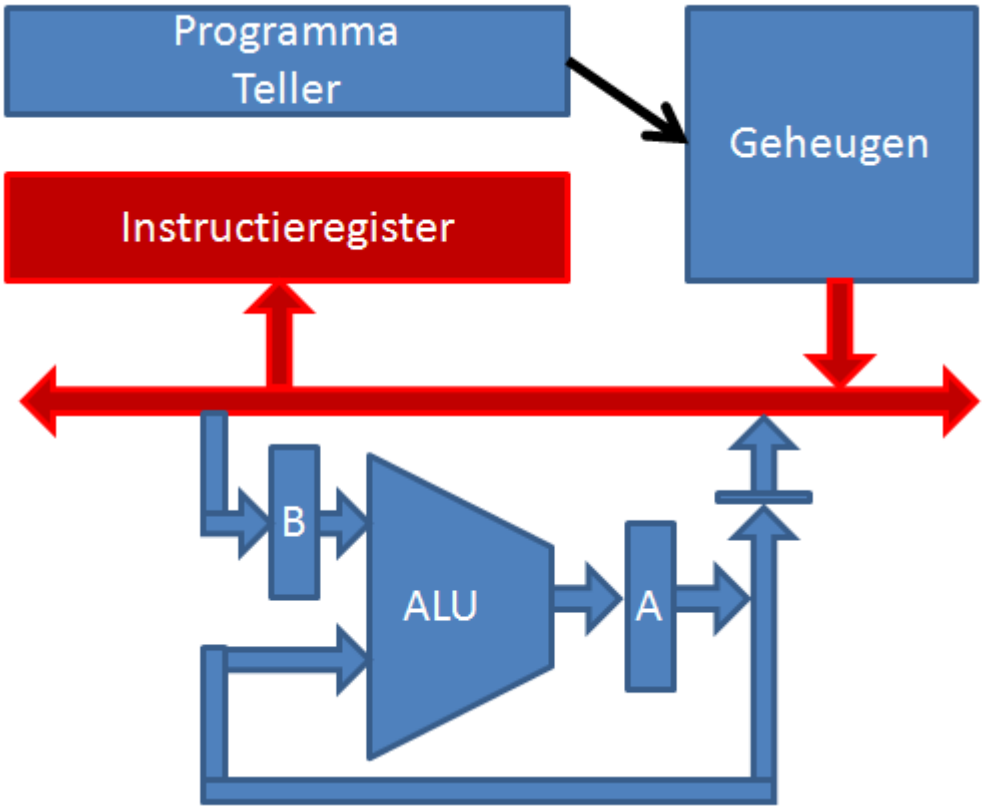


We hebben tot slot afgesloten met de vraag:
Zou het mogelijk zijn om een dergelijke computer te maken en bijvoorbeeld een variant van het eerste programma van de SSEM te draaien?
Om van bovenstaand model een werkende computer te maken en daarop een programma te draaien hebben we de volgende stappen uitgevoerd:

- Ontwerp van een instructieset.
- Bouw van het eerste deel van de hardware (alles behalve de 'control ' module).
- Testen van de hardware met een moderne computer.
- Testen van de instructieset.
- Bouw van de control module (met daarin de instructieset).
- Schrijven van een assembler.
- Schrijven, debuggen en uitvoeren van het programma.

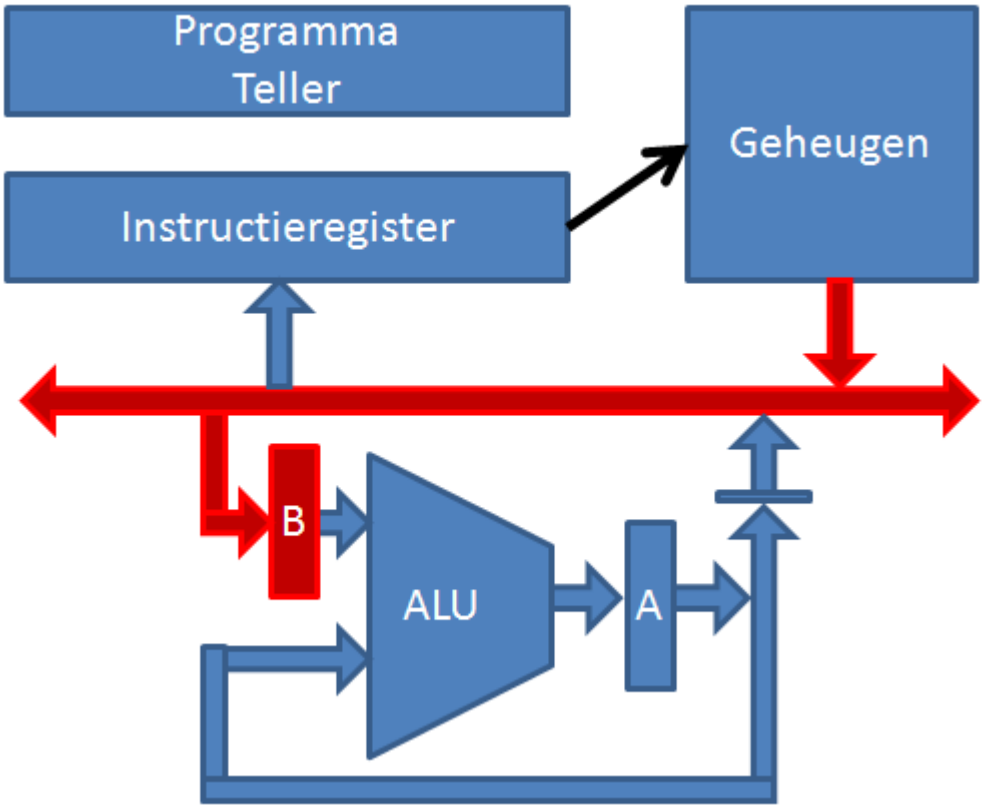
Ontwerp van de instructieset.

Bij de little man machine hadden we al gezien welke elementaire instructies een computer nodig heeft, namelijk **load**, **store** en **jump** (al dan niet onder een bepaalde voorwaarde). En daarnaast is het natuurlijk nodig om ook een bewerking uit te voeren, zoals **add** (optellen).
In het bovenstaande model zijn de paden waarover de data loopt met pijlen aangegeven. Bij de vertaling naar hardware moet je er rekening mee houden dat op een dergelijk pad maar 1 signaal tegelijk kan lopen. Je kunt nu eenmaal niet een waarde uit je geheugen lezen en tegelijk, via dezelfde pinnen, een andere waarde naar datzelfde geheugen wegschrijven. Het uitvoeren van de instructies moet dus in stapjes gebeuren. Om dit mogelijk te maken worden de data tussendoor opgeslagen. Hiervoor gebruiken we in ons geval een tweetal registers, A en B. Dit is weergegeven in het onderstaande (iets vereenvoudigde) model:

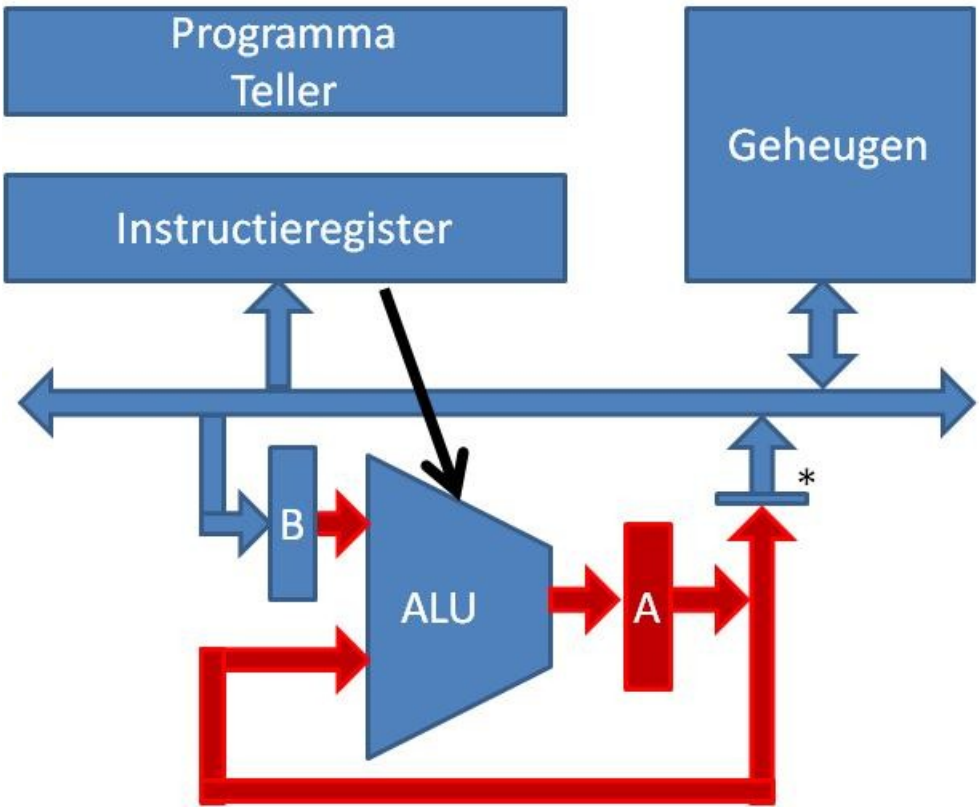


Zoals we gezien hebben bij ons kleine mannetje is de eerste stap altijd het ophalen van de instructie. De programmateller geeft aan op welke plaats de instructie in het geheugen staat (zwarte pijl) en de instructie wordt van het geheugen via de databus doorgegeven aan het instructieregister. Dit pad is in het bovenstaande schema rood gekleurd.

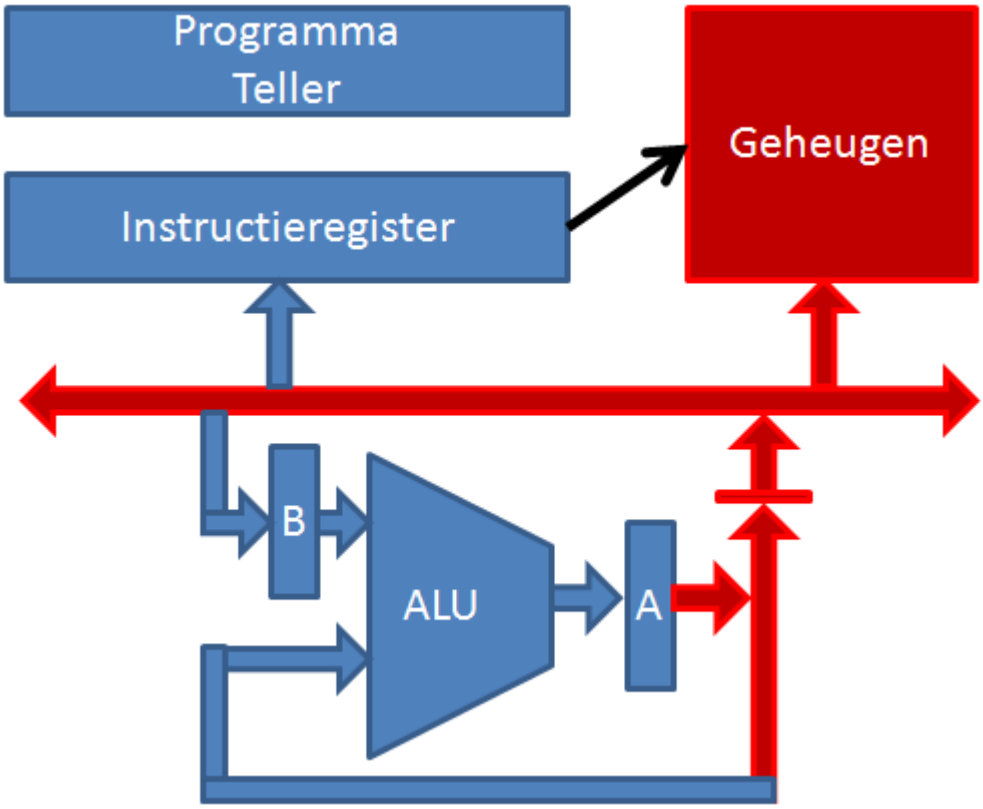
Nu de instructie in het instructieregister staat, kan de controlemodule bepalen wat de volgende stap moet zijn. Hiervoor hebben we onderscheid gemaakt tussen de instructie zelf (in ons geval de hoogste 4 bits) en bijbehorende data (laagste 12 bits). De bovenste 4 bits geven bijvoorbeeld aan het B-register geladen moet worden (Load-B of LDB) en de laagste 12 bits geven het adres (postvakje) aan waar de waarde opgehaald moet worden. De controlemodule zorgt er dan voor dat de laagste 12 bits als adres wordt gebruikt, dat het geheugen het resultaat op de bus zet en dat het B-register dit inleest. Schematisch ziet dit er als volgt uit:



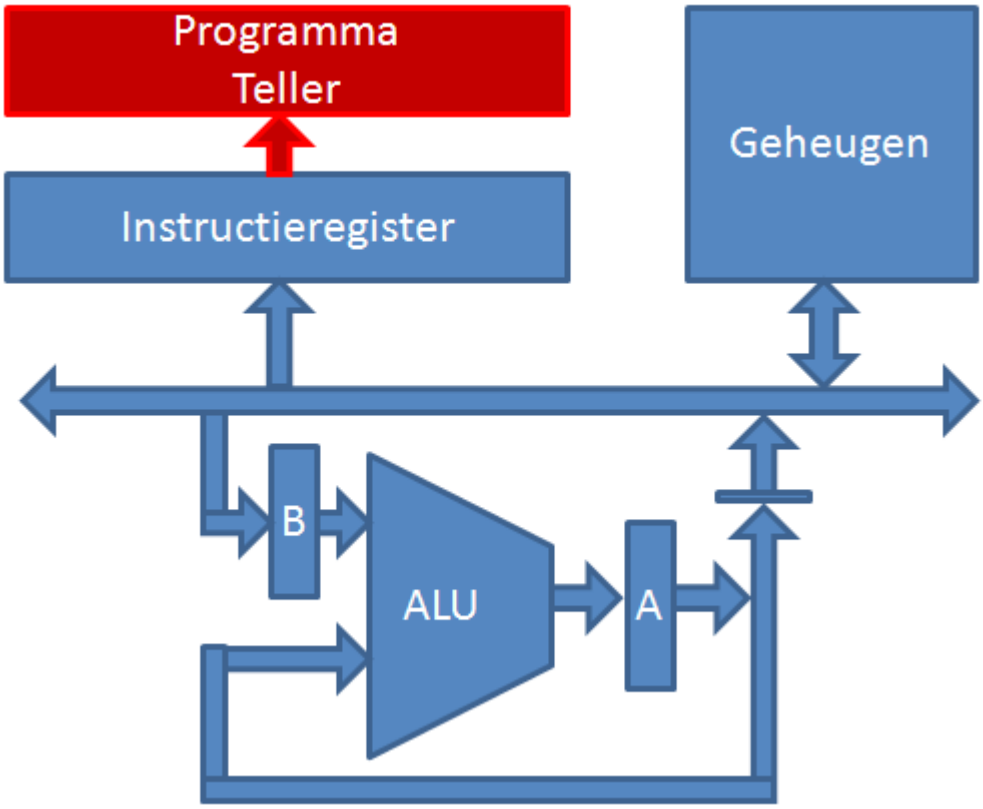
Nadat dit is uitgevoerd, wordt de volgende instructie uit het geheugen opgehaald en in het instructieregister opgeslagen, bijvoorbeeld een OPR $2AC - de ADD instructie. De hoogste 4 bits van de instructie geven aan dat de ALU een bewerking moet uitvoeren (OPR) en de uitkomst in het A-register geladen moet worden. De laagste 12 bits ($2AC) geven aan welke bewerking dit precies is. Alle signalen zijn al goed aangesloten (zie de rode pijlen in onderstaand schema) en het enige wat de controlemodule hoeft te doen is zorgen dat A opnieuw wordt geladen.



*Merk op dat tussen de output van A en de databus een extra buffer zit, die ervoor zorgt dat de uitvoer van A niet standaard op de bus wordt gezet, maar wel beschikbaar als ingang voor de ALU.
Deze buffer wordt geactiveerd als we het resultaat van de berekening met de instructie Store-A (STA) opslaan in het geheugen. In onderstaande figuur is de datastroom van STA weergegeven.



Om een werkend programma te bouwen hebben we naast de besproken bewerkingen ook een Jump (JMP) instructie nodig. Deze instructie laad een nieuw adres in de programmateller, zodat de volgende instructie vanaf dat adres verder wordt uitgevoerd (zie onderstaande figuur).



Het laden van de programmateller wordt, net als voor alle andere registers, bestuurd door de controlemodule. Met een enkele extra digitale poort kan dit laad-signaal geblokkeerd worden en dan doet deze instructie ineens niets (NOP, No-Operation). En met een paar poortjes extra wordt geregeld dat het laad-signaal wordt doorlaten als:

- het A-register 0 is (JZ – Jump Zero),
- het A-register niet 0 is (JNZ – Jump Not Zero),
- de waarde van de Carry 1 is (JCS – Jump Carry Set)
- de waarde van de Carry 0 is(JCC – Jump Carry Clear)
- het hoogste bit van het A-register 1 is – wat aangeeft dat het getal in A negatief is (JMI – Jump Minus)
- het hoogste bit van het A-register 0 is (JNM – Jump Not Minus).

Samen met NOP en onvoorwaardelijk springen (JMP) hebben we 8 verschillende instructies die het programmaverloop kunnen beïnvloeden, afhankelijk van een bepaalde conditie.

Bouw

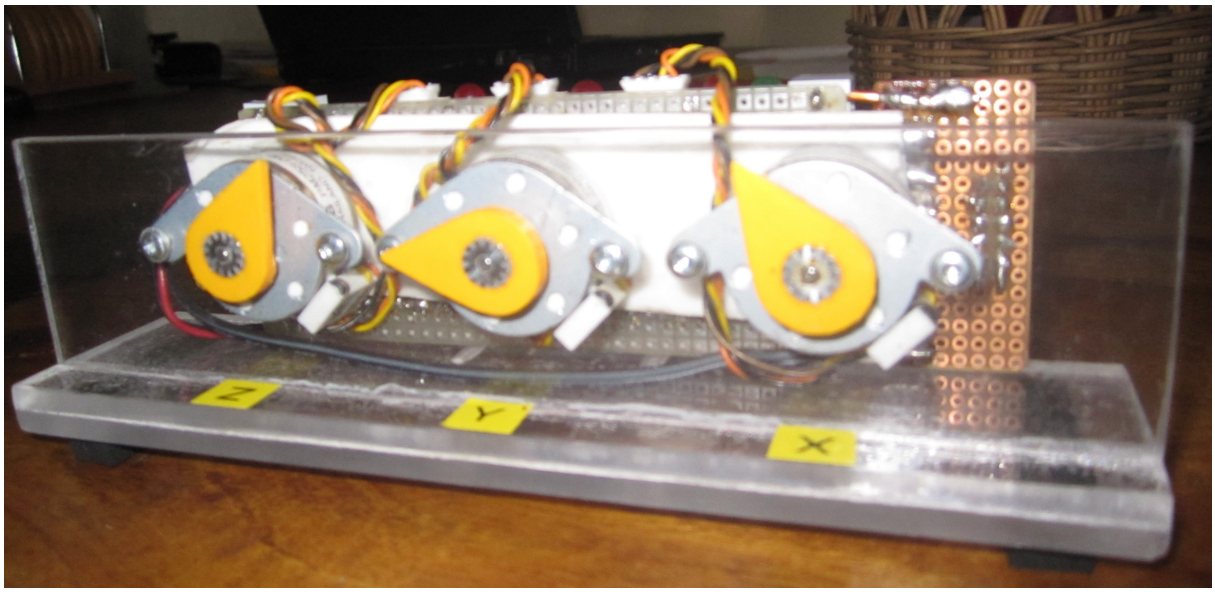
De computer is opgebouwd uit 'oude' onderdelen. Het meeste onderdelen waren al beschikbaar in de jaren 70 van de vorige eeuw en het meest moderne onderdeel is een 8kx8 statische geheugenchip uit begin jaren 80. Ook deze kan uit losse onderdelen worden opgebouwd. We hebben echter voor een complete chip gekozen omdat zelf bouwen veel tijd en geld zou kosten en het resultaat een enorme omvang zou hebben. Daarnaast is het weinig uitdagend: de opbouw per geheugenplaats is vergelijkbaar met A-registers met uitgangsbuffer. Dit wordt aangevuld met geheugendecoder en de 4096 geheugenplaatsen die nu beschikbaar zijn, zou zo'n 10.000 chips vragen!
De bouw is gestart met de geheugenmodule en de ALU en na de bouw hebben we moderne middelen ingezet om alles te testen: een PIC op Dwarf board van Voti, geprogrammeerd in JAL. De software is opgebouwd in stapjes: eerst zijn de verschillende onderdelen getest los en daarna zijn de onderdelen in samenhang getest, door pulsen te geven op de controlelijnen die later door controlemodule worden aangestuurd. Om eenvoudig de kunnen zien wat er gebeurt zijn op de print headers aangebracht met de belangrijkste signalen (Instructie-register, program counter, A-register en B-register). Door middel van probes (MCP23017- een 16 bits i2c IO chips van Microchip) kan het dwarf board na ieder stapje de status bepalen en uitprinten.
Het dwarf board is in eerste instantie ook gebruikt om de controlesignalen te genereren en daarmee is de werking van alle instructies stapje voor stapje getest. Vervolgens is de controle-module gemaakt, die de controlesignalen met standaard digitale poorten maakt.
Het resultaat.
Het bovenstaande beschrijft de ontwikkeling van een computer. Bij de ontwikkeling hebben we ook een kleine assembler geschreven om programma's om te zetten in hex code. Het genoemde dwarf board wordt gebruikt om het programma van de PC te laden in het geheugen van de computer. Nadat het programma is geladen is het dwarf board niet meer nodig en kan de computer zelfstandig het programma uitvoeren.
Het eerste programma voor deze computer bepaalt – geïnspireerd op de Manchester SSEM 'Baby' – de 'highest proper factor' – het hoogste deeltal van een 24-bits getal dat een geheel resultaat opleverd. Onze computer blijkt – ondanks de beperking van 8 bits rekenwerk, waardoor veel meer instructies nodig zijn – een stuk sneller dan deze pionier. In een seconde of 10 was het programma gereed, terwijl zijn vroege voorvader hier 52 minuten voor nodig had.
Al met al een leuk en geslaagd project wat ook meteen weer ideeën heeft opgeleverd voor een vervolg. Zo lijkt een 'stack machine' – een processor die sterk gericht is op forth-achtige taal – een kleine stap verder en bijzonder krachtig. Door deze niet met losse poortjes op te bouwen, maar met een FPGA (programmeerbare digitale hardware) kan op die manier prestaties worden gehaald die 5 tot 10 maal beter zijn dan een standaard PIC of Atmega processor.
Maar… dat zullen we voorlopig laten rusten, want we hebben gekozen om de andere kant op te gaan: we hebben een partij relais op de kop getikt en Aloys heeft een mooi frame gemaakt om deze op te monteren. Het doel: een computer opgebouwd uit zo'n 300 tot 500 relais. De oordopjes liggen klaar!

Karel Dupain & Joep Suijs

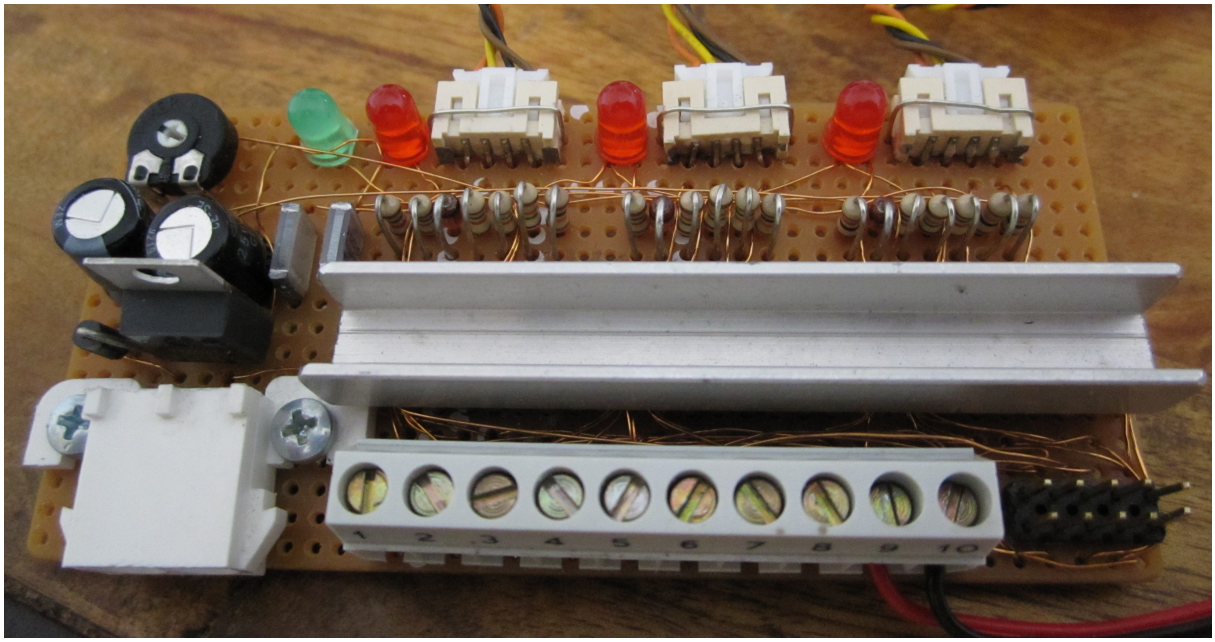
Stappenmotor stuurprintje

Stappenmotor stuurprintje. 13 jun 2011

Voor demonstratie heb ik een opstelling gemaakt met drie stappenmotoren.
Deze kunnen via de parallelle poort en/of USB worden aangestuurd.
Wat een stappenmotor is en hoe die moet worden aangesloten is al meerdere malen vermeld, maar op verzoek wil ik dat nogmaals beschrijven.
Om ervaring met besturen van stappenmotoren te krijgen, heb ik dit gebouwd.

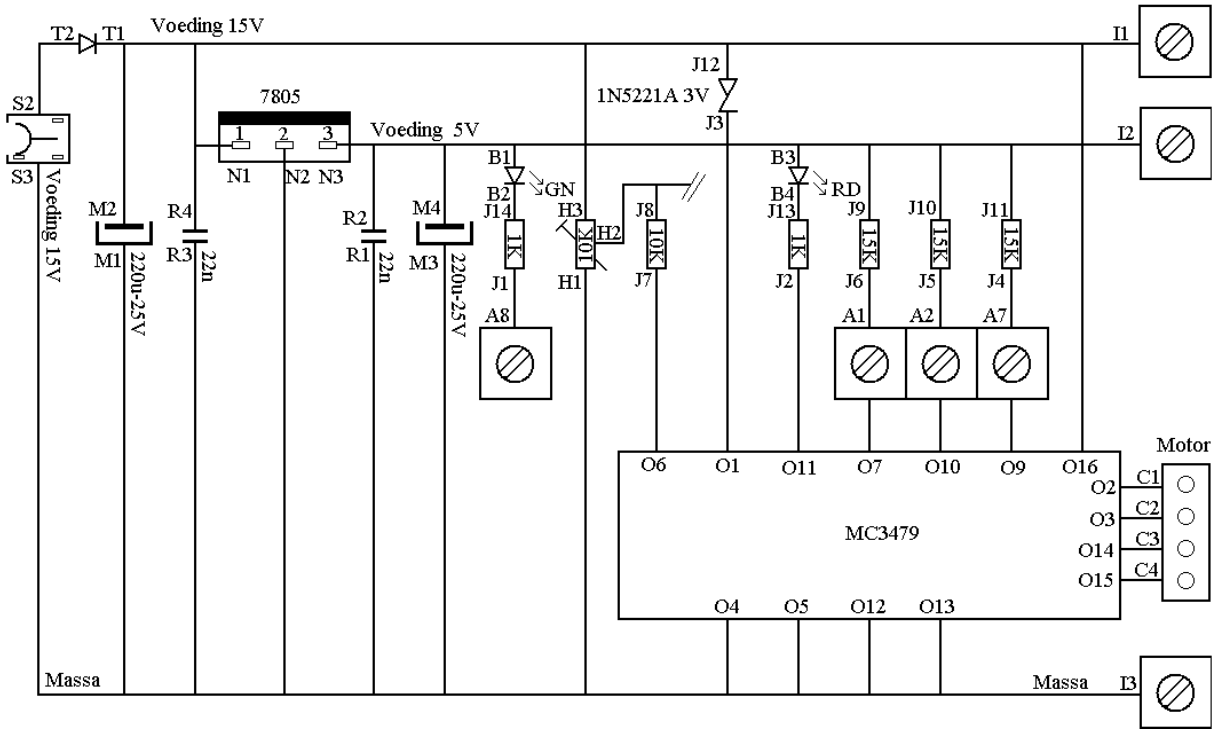


Drie bipolaire stappenmotoren, gemonteerd op een stukje plastic.
De sturing wordt verzorgd door een single chip de MC 3479.
In dit IC zit alles wat voor de aansturing nodig is.



Onder het U-profiel, de koelstrip, zitten de drie stuur IC's. (in DIL behuizing)
Links de 5V voeding, boven de weerstanden, leds en motoraansluitingen.
De weerstanden zijn niet nodig, maar geven de IC ingangen een vast niveau. Onderaan stuuraansluitingen één maal kroonstrip, één maal 10 polige header.

Schema:

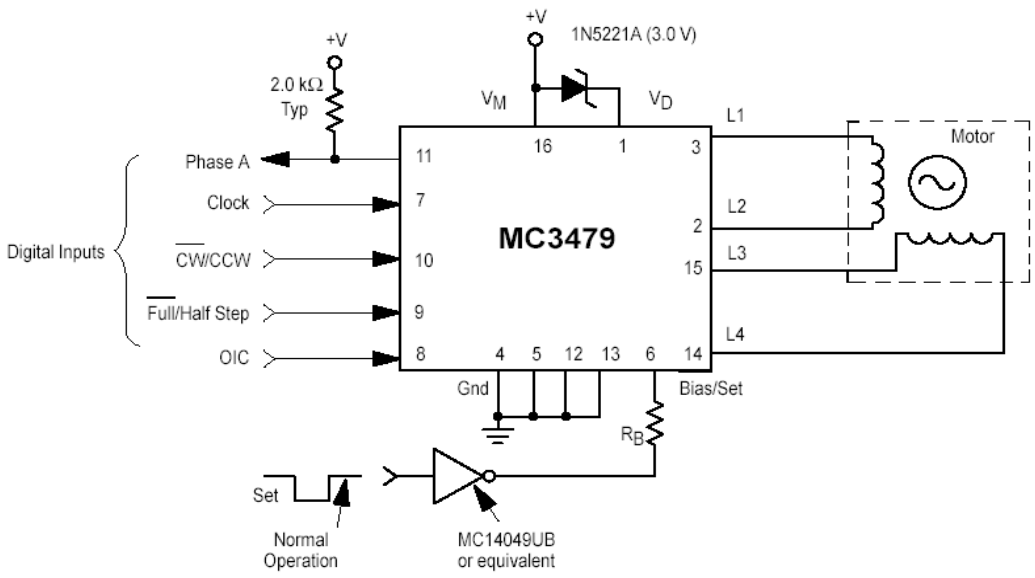


Beschrijving:

De voeding komt van een 'dikke stekker' voeding, een paar honderd milliampère volstaat. De spanning is 15V, aangesloten op de witte plug links onderaan (in het schema links bovenaan).
Gevolgd door een beschermingsdiode, verkeerdom aansluiten geeft geen schade.
De motoren worden direct op deze voedingsspanning aangesloten.
Een 7805 spanningsregelaar zorgt voor de voeding van de interne logica.
De nummering komt uit de componenten positie van mijn printje, niet op letten.

Het IC wordt, ondanks de lage stroom toch behoorlijk warm, daarom heb ik er een ventilatortje bij gezet. (rood-zwarte draadje rechts onderaan op de foto).
De ledjes, aangesloten op pootje -11- (phase -A-) geven aan, wanneer de motor weer in zijn uitgangspositie staat, dus na elke 4e full-stap, of na elke 8e half-stap.

Hieronder het applicatie schema zoals de fabrikant het graag ziet.
Er is één pootje dat vragen oproept, dat is pootje nummer -8-
Ik heb deze aansluiting niet gebruikt, dus los gelaten.
Het is bedoeld voor het aanpassen van het gedrag in half-stappen bedrijf.
Aangezien het voor de demo niet uitmaakt of je dit hoog of laag maakt, heb ik het verder niet aangesloten. In belaste toestand kan het wel uitmaken, maar dat is een kwestie van uitproberen.



Aansluiting:
De aansluitpunten, de nummers 1…..10 zijn twee maal uitgevoerd.
Één maal in de vorm van een printkroonstrip, één maal als 10-polige header.

-1- motor X stapje
-2- motor X draairichting
-3-4- motor Y stapje – cw/ccw
-5-6- motor Z stapje – cw/ccw
-7- alle motoren full/half stappen.
-8- stroombegrenzing bij stilstand.
-9- massa (-)
-10- + 5V

In onderstaand verhaal is van bovenstaande situatie uitgegaan.
Hierbij is stroombegrenzing in de demo opstelling uitgevoerd als (groen) ledje.
De motoren draaien al op minimale stroom, om oververhitting van de chip te voorkomen. (instelpotmeter, aangesloten op punt 6 van de chip).

De motoren zijn dmv de witte plugjes aangesloten.
Er zijn geen zekeringen of andere beveiligingen toegepast.

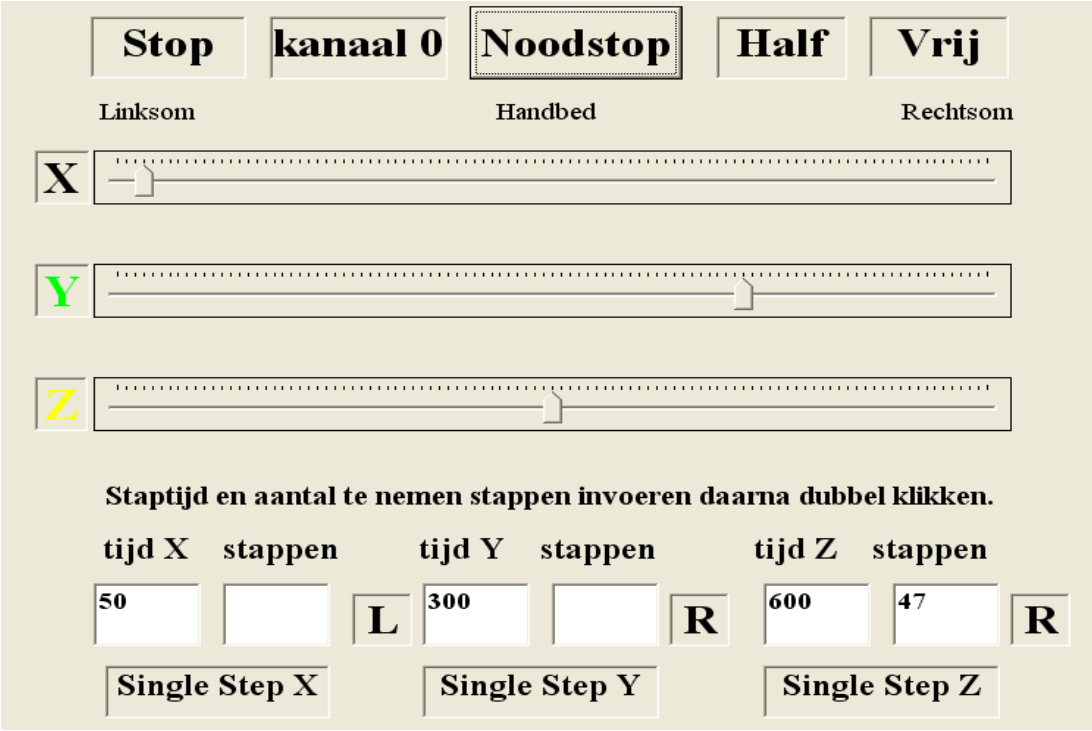
Bediening:
Voor zowel de USB als de parallelle poort uitvoering is er geen verschil in bediening. De software is uiteraard wel verschillend.
Alles is geprogrammeerd in Visual Basic 6.0
Op de bovenste rij staan een aantal knoppen:

Startknop, die na indrukken verbinding maakt met het device.
Tijdens het verbinden staat er 'wacht' dat staat er ook in noodstop situatie.
Bij USB staat het *USB kanaal* vermeld, hier is dat kanaal -0-.
Bij de parallelle poort moet je vóór het starten op deze plek het gebruikte *poortadres* invullen.

Noodstop hoeft geen uitleg, na elke handeling komt de fokus weer bij de noodstop te liggen. Dus een klap op de spatiebalk of return toets geeft noodstop.
Uiteraard staat alles ook stil, wanneer je op de noodstop klikt.

half is half-stap bedrijf, klik je hierop dan komen alle drie de motoren in full-stap bedrijf.

Zo ook de toggle switch *Vrij/Rem*.



Dan de drie schuifregelaars: *X Y* en *Z*
Zwart geeft aan dat de boel UIT staat.
Hier staat X uitgeschakeld, maar zodra je op de zwarte X klikt zal de motor linksom gaan draaien waarbij de stapjes elkaar elke 50 ms opvolgen.

Y staat in bedrijf, en wel met stapjes van 300 ms rechtsom.
Elke verstelling van de schuifregelaar geeft een aanpassing van het uitleesvenster onderaan. Elke verandering in het uitleesvenster onderaan, geeft een verstelling van de instelschuif bovenaan.
Klik je op - R- dan zal deze toggleswitch naar -L- omschakelen.
Ook de schuifregelaar zal aan de linkerstand plaatsnemen.

De schuif is instelbaar van 20 ms tot maximaal 500 ms
Voer je een kleinere waarde dan 20 ms in, dan zal de waarde 20 ms blijven.
Elk USB commando duurt 10 ms, voor één stap puls twee maal (op en neer).
Hierdoor is de kortste pulstijd dus 20 ms. Het heeft geen zin sneller in te stellen.

Voer je een waarde groter dan 500 ms in, dan zal de schuif niet meer reageren en in de middenstand gaan staan.
De staptijd kan dan niet meer door de schuif worden aangepast.
Pas als er een waarde kleiner dan 500 wordt ingevuld, dan werkt de schuif weer.

Z is geel, dat wil zeggen: hij is bezig met uitvoeren van een vooraf ingevoerd aantal stappen. De teller laat zien hoeveel stappen er nog zijn te gaan.
Na het ingestelde aantal stappen stopt de motor, en zie je het aantal stappen in het venster staan. Voor de demo motoren is een hele omwenteling 48 stappen.
(in full-step stand natuurlijk).

Geheel onderaan kan je elke motor één stap tegelijk laten maken.
Hiervoor hoeft de motor niet te worden ingeschakeld (X-Y-Z blijven zwart).
In de praktijk is dit nodig om de beitel, frees, pen, etc. in de juiste stand te zetten
Uiteraard kan je met R en L wel de staprichting bepalen.

USB interface:
De print van Velleman, zie Conrad.nl kost nog geen drie tientjes.

http://www1.conrad.nl/scripts/wgate/zcop_nl3/~flN0YXRlPTM4OTMzNjQzNDc=?~template=PCAT_AREA_S_BROWSE&glb_user_js=Y&shop=NL2&zhmmh_lfo=&zhmmh_area_kz=&product_show_id=191003&gvlon=&p_init_ipc=X&p_page_to_display=fromoutside&~cookies=1&gclid=&scrwidth=1069

Het bouwpakket omvat:
5 digitale ingangen en 8 digitale uitgangen (bitjes)
2 analoge uitgangen, 2 analoge ingangen.
De analoge in/uitgangen zijn 8-bits breed, dus een resolutie 5/255= 20mV

Er wordt een duidelijke fool-proof bouwbeschrijving meegeleverd.
Alle benodigde drivers en uiteraard voorbeeld software wordt meegeleverd.
Programma voorbeelden voor C en VB

Software:
Zo, ga er maar eens lekker voor zitten, want hieronder volgt een lang verhaal.
Ik heb niet alles weergegeven, maar wie het hebben wil, is e-mailtje voldoende Voor motoren Y en Z geldt hetzelfde als voor motor X

```
'============START/STOP KNOP===========

Private Sub LBLstart_Click()
   If LBLstart.Caption = "Stop" Then       'stoppen
      WriteAllDigital (0)                  'motorgetal -0-
      CloseDevice                          'communicatie afsluiten
      End                                  'gestopt
   End If
   LBLstart.Caption = "Wacht"              'wacht op verbinding
   OpenDevice (0)                          'communicatie kanaal openen
   LBLstart.Caption = "Stop"                       'startknop wordt stopknop
   Herstel                                 'alle waarden naar beginstand
End Sub

'============FULL / HALF  STAPPEN===========

Private Sub LBLfh_Click()           'toggle switch full of half stappen
   If LBLfh.Caption = "Full" Then LBLfh.Caption = "Half" Else LBLfh.Caption = "Full"
   If LBLfh.Caption = "Full" Then SetDigitalChannel (7) Else ClearDigitalChannel (7)
   Herstel                          'alle knoppen en labels weer in startpositie
End Sub

'============REM===========

Werkt eender als full/half stappen, maar dan met bitje -8-
'============MOTOREN AAN/UIT==========

Private Sub LblX_Click()            'motor X   aan - uit zetten
   If timerX.Enabled = False Then
      timerX.Enabled = True                'continu timer aan
      TimerX1.Enabled = False      'aantal stappentimer uit
      LblX.ForeColor = vbGreen     'continu kleur groen
      Else
      timerX.Enabled = False               'continutimer uit
      LblX.ForeColor = vbBlack     'weergave uit melding
   End If
   Herstel                          'alle knoppen en labels weer in startpositie
End Sub

Zo ook bij de andere twee motoren (Y en Z)

'============DRAAIRICHTING===========

Private Sub LblXlr_Click()          'motor X draairichting
   If LblXlr.Caption = "L" Then LblXlr.Caption = "R" Else LblXlr.Caption = "L"
                                    'toggleswitch L/R
   If LblXlr.Caption = "L" Then SetDigitalChannel (2) Else ClearDigitalChannel (2)
                                    'draairichting aansturen
```

```
   If LblXlr.Caption = "L" And SLDX > 0 Then SLDX = SLDX * -1
                                    'slider X in juiste positie brengen
   If LblXlr.Caption = "R" And SLDX < 0 Then SLDX = SLDX * -1
                                    'slider X in juiste positie brengen
   Herstel                          'alle knoppen en labels weer in startpositie
End Sub
```

Zo ook bij de andere twee motoren (Y en Z)

```
'==========SCHUIFREGELAARS==========

Private Sub SLDX_Change()               'regelschuif motor X
  If Val(pulstijdX.Text) < 521 Then             'kleiner dan halve seconde geen slider gebruiken
    pulstijdX.Text = 520 - Abs(SLDX.Value)  'snelheid van 20 tot 500 ms
    If SLDX.Value >= 0 Then                     'draairichting rechtsom
      LblXlr.Caption = "R"                      'rechts weergeven
      ClearDigitalChannel (2)                   'rechts motorsturing
      Else
      LblXlr.Caption = "L"                      'linksom aangeven
      SetDigitalChannel (2)                     'motor linksom
    End If
    timerX.Interval = 520 - Abs(SLDX.Value)     'stappentimer instellen
  End If
  Herstel                                       'alle waarden na noodstop naar beginstand
End Sub
```

Zo ook bij de andere twee motoren (Y en Z)

```
'============STAP PAUZETIJD=========

Private Sub pulstijdX_dblclick()                    'pauzetijd tussen stappen  X
   If Val(pulstijdX.Text) < 20 Then pulstijdX.Text = 20'minimale staptijd 20 ms
   If pulstijdX.Text < 500 Then                     'bij kleinere pauze dan 500 ms slider
     If LblXlr.Caption = "L" Then                   'als linksom aan staat
       SLDX.Value = Val(pulstijdX.Text) – 520       'slider negatief instellen
       SetDigitalChannel (2)                        'linksom richting bitje aanzetten
       Else
       SLDX.Value = 520 - Val(pulstijdX.Text)       'slider rechtsom instellen
       ClearDigitalChannel (2)                      'richting bitje uitschakelen = rechtsom
     End If
     Else
     SLDX.Value = 0                 'bij grotere waarden dan 500 ms slider in middenstand
     timerX.Interval = Abs(Val(pulstijdX.Text))     'intervaltimer instellen
   End If
   Herstel                                          'alle knoppen en labels weer in startpositie
End Sub
```

Zo ook bij de andere twee motoren (Y en Z)

```
'===========AANTAL STAPPEN===========

Private Sub stappenX_dblClick()
  Call pulstijdX_dblclick               'controle op juiste waarde & slider aanpassen
   stapX = Val(stappenX.Text)           'het aantal stappen ophalen
   timerX.Enabled = False               'continutimer uitzetten
   TimerX1.Interval = Val(pulstijdX)            'het interval ophalen
   LblX.ForeColor = vbYellow            'kleur instellen
   TimerX1.Enabled = True               'het stappenproces starten
   Herstel                              'alle knoppen en labels weer in startpositie
End Sub
```

Zo ook bij de andere twee motoren (Y en Z)

```
'============SINGLE STEP ===========

Private Sub LBLssX_Click()          'stapje X
   SetDigitalChannel (1)            'bitje 1 motor X sturen
   ClearDigitalChannel (1)          'maakt bitje 1 weer 0
   Herstel                          'alle waarden naar beginstand
End Sub
```

Zo ook bij de andere twee motoren (Y en Z)

```
'============NOODSTOP==============

Private Sub CMDnoodstop_Click() 'alles stop zetten
   timerX = False                   'motor X stop
   timerY = False                   'motor Y stop
   timerZ = False                   'motor Z stop
   TimerX1 = False                  'aantal stappen timer uitzetten
   TimerY1 = False                  'aantal stappen timer uitzetten
   TimerZ1 = False                  'aantal stappen timer uitzetten
   WriteAllDigital (0)              'motorgetal -0-
   LBLfh.Caption = "Half"           'F/H naar H
   LBLrem.Caption = "Vrij"          'Rem los
   LblXlr.Caption = "R"             'Draairichting X startstand
   LblYlr.Caption = "R"             'Draairichting Y startstand
   LblZlr.Caption = "R"             'Draairichting Z startstand
   LBLstart.Caption = "Wacht"       'wachttijd melding in startknop
   TMRnoodstop = True               'laat de noodstop knipperen
   CMDnoodstop.Visible = False      'commandoknop even weg
End Sub
```

```
'============HERSTEL===========

Private Sub herstel()               'zet de instellingen in beginstand na noodstop
   If TMRnoodstop = True Then       'noodstop gemaakt
      TMRnoodstop = False           'noodsituatie opgeheven
      LBLnoodstop.Visible = False   'rode meldtekst uit
      CMDnoodstop.Visible = True    'noodstopknop zichtbaar
      LBLstart.Caption = "Stop"     'wachtmelding wordt stopknop
      If timerX.Enabled = False Then LblX.ForeColor = vbBlack 'bedrijfsmelding uit
      If timerY.Enabled = False Then LblY.ForeColor = vbBlack 'bedrijfsmelding uit
      If timerZ.Enabled = False Then LblZ.ForeColor = vbBlack 'bedrijfsmelding uit
      SLDX = 0                      'slider X in neutraalstand
      SLDY = 0                      'slider Y in neutraalstand
      SLDZ = 0                      'slider Z in neutraalstand
   End If
   CMDnoodstop.SetFocus             'fokus naar de noodstopknop
End Sub
```

```
'============ TIMERS==========

Private Sub timerX_Timer()              'bedrijfstimer X
   DoEvents
   SetDigitalChannel (1)                'bitje 1 motor X sturen
   ClearDigitalChannel (1)              'maakt bitje 1 weer 0
End Sub
```

Zo ook bij de andere twee timers (Y en Z)

```
Private Sub timerX1_Timer()                 'meer stappen timer X
   DoEvents
   SetDigitalChannel (1)                    'bitje 1 motor X sturen
   ClearDigitalChannel (1)                  'maakt bitje 1 weer 0
   stappenX.Text = Val(stappenX.Text) – 1       'terugtellen
   If Val(stappenX.Text) <= 0 Then                  'na gedane zaken, stoppen
     stappenX.Text = stapX                  'oude waarde terugzetten
     TimerX1.Enabled = False                'timer uitzetten
     LblX.ForeColor = vbBlack               'bedrijfsmelding uitzetten
   End If
End Sub
```

Zo ook bij de andere twee timers (Y1 en Z1)

```
Private Sub TMRnoodstop_Timer()         'knipperen van de rode noodstop melding
   If LBLnoodstop.Visible = False Then LBLnoodstop.Visible = True Else LBLnoodstop.Visible = False
End Sub
```

Voor wie geïnteresseerd is in de softwareversie voor de parallelle poort, is een e-mailtje voldoende om het te krijgen.

Groeten, Dré Jansen

Drejansen.1@gmail.com

ROBOTVOETBAL SOFTWARE

Sinds de vorige Robobits is er een hoop gebeurd binnen ons Robotvoetbal project met name wat betreft het ontwikkelen van de infrastructuur waaronder de scheidsrechter en het beeldherkennings systeem.
Vandaar dat het artikel over het programmeren van VPL even op zich laat wachten mede omdat er zonder afgeronde infrastructuur ook geen voetbalteams in VPL geprogrammeerd kunnen worden.

Het Wrox boek

Veel van de kennis om de infrastructuur te programmeren is te halen uit het standaard werk over Microsoft Robotics Developer Studio namelijk Professional MRDS uit 2008.
Hoe je gratis dit boek kunt downloaden, de updates van de voorbeelden aan de praat krijgt en overig commentaar bij de hoofdstukken houd ik bij in mijn: Blog WroxProfessionalMRDS.2008:
http://www.twintellect.com/DMS/robotvoetbal/Blog WroxProfessionalMRDS.2008.htm
De blog is in het engels omdat het boek zelf ook in het engels is. Dus als je het boek kunt lezen, kun je ook de blog lezen.
Om deel te kunnen nemen aan ons robotvoetbal project hoef je dit boek overigens helemaal niet te lezen. Voor het programmeren van een voetbalteam zullen we de Visual Programming Language gebruiken.
En de bespreking van hoe je die gebruikt voor robotvoetbal wordt in het volgende artikel in de reeks nader uitgelegd.

De applicatie

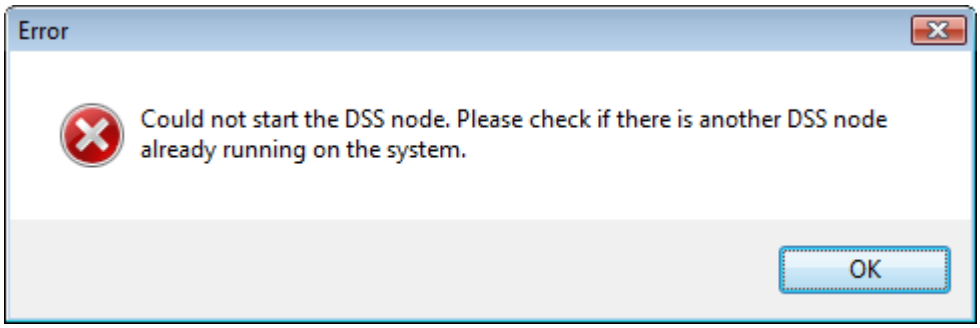
Bert Berrevoets heeft de basis van de applicatie gelegd welke bestaat uit de HCCRefereeModule, de HCCPlayerModule, de HCCGenericVisionModule en de HCCSimpleVisionModule.
In de volgende hoofdstukken worden ze kort beschreven.
De bestanden zijn gratis te downloaden net zoals de installatie handleiding vanaf onze Codeplex website. Zie het hoofdstuk “HCC Soccer Project op Codeplex”
Nadat de stappen voor het installeren van de applicatie gevolgd zijn kan deze opgestart worden.Het is mogelijk om alle onderdelen op 1 computer te laten lopen maar dat vereist wel een snelle computer en een speciale handelingen.

HCCRefreeModule

De HCCRefereeModule wordt opgestart vanaf een Dss Command Prompt door het commando “Referee” wat bovendien tot gevolg heeft dat de HCCSimpleVisionModule opstart.
De HCCRefereeModule houdt de tijd, score en straffen bij en kan natuurlijk het spel stilleggen onder andere voor timeouts.
Vanuit 2 andere computers op het netwerk starten team Yellow en team Blue hun VPL programma’s op. De tekst in de statusbalk van de RefereeBox verandert van “Waiting for teams” naar “Waiting for Blue” of “Waiting for Yellow” en als beide teams geregistreerd zijn in “Ready for Game”.
In de afbeelding hieronder is de RefereeBox te zien en waarvoor de knoppen gebruikt worden is wel uit hun titels te begrijpen. Alle knoppen leggen het spel stil en stoppen de klok. De teams krijgen deze informatie ook en moeten zelf in hun VPL code zorgen dat de robots stil blijven staan of bij het scoren van een goal naar hun positie bij de middenstip gaan.Deze goal wordt dus ook met de “Goal” button geregistreerd.
De HCCRefereeModule service runt op poort 60000 en 60001. Zie in de paragraaf “Player Module” hoe poorten worden vrijgegeven voor gebruik.

HCCPlayerModule

Na de installatie is er in VPL een HCCPlayerModule in de services lijst bijgekomen. In het installatie document staat kort beschreven hoe je hier mee begint maar in het volgende artikel van de reeks zal hier uitgebreid op ingegaan worden.
Het is mogelijk 2 teams op 1 computer te gebruiken door 2 VPL's op te starten en ieder team zijn eigen poorten te laten gebruiken. Die poorten stel je in onder het menu "Run\Port Settings". Ken aan het ene team poorten 50000 en 50001 toe en aan het andere 50002 en 50003. De eerste van iedere set poorten wordt gebruikt voor de webinterface en de tweede voor de communicatie tussen de services onderling. Sla je programma met duidelijke namen op zoals "HCCPlayer_TeamYellow50000.mvpl" en HCCPlayer_TeamBlue50002.mvpl".
Gegarandeerd dat je bij het runnen van het Blauwe team de volgende melding krijgt:



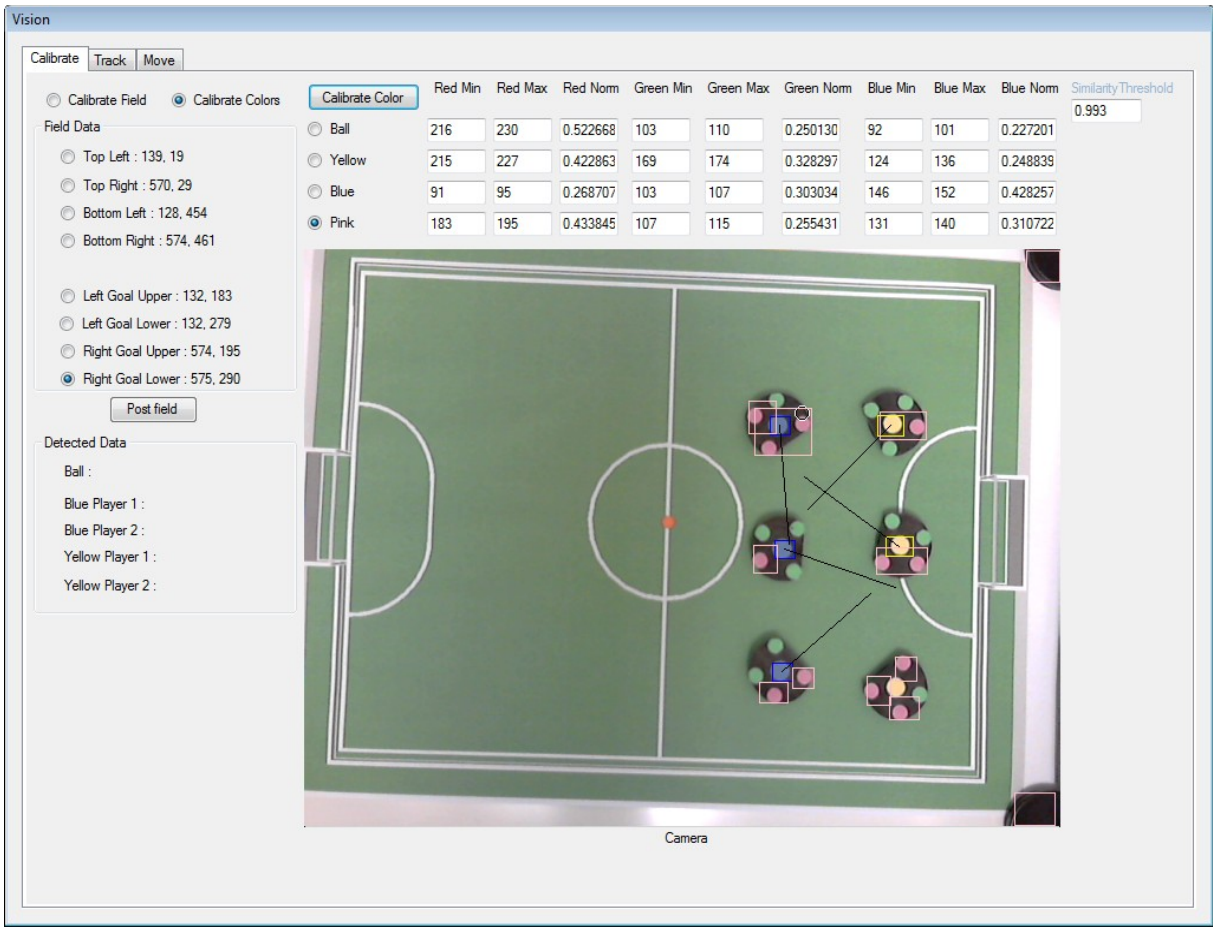
De oorzaak is dat deze poort niet vrijgegeven is om te gebruiken vanuit VPL.
Klik met rechts op de DSSCommandPrompt en kies starten als Administrator. Type vervolgens
`"httpreserve /p:50002 /u:je_host_naam\je_user_naam"`
Je_host_naam kun je vinden door in de command prompt "ipconfig /all" te typen en even te zoeken naar hostname. Kun je niet scrollen door de resultaten, verander dan in de properties waarmee je de DSSPrompt start de grootte van de buffer. Doe dit ook voor poort 50003.

De HCCGenericVisionModule

De HCCGenericVisionModule beschrijft het contract waarmee de communicatie tussen de verschillende onderdelen van het HCCSoccerProject plaatsvindt. Daarin staan de statische gegevens zoals de veldafmetingen, dynamische gegevens zoals de posities van de bal, posities van de spelers en spel controle gegevens van de scheidsrechter. Het verkeer van deze berichten gaat altijd van de HCCRefereeModule naar de beide applicaties van de HCCPlayerModule. De spelers kunnen deze gegevens alleen intepreteren en ermee de eigen spelers aansturen via BlueTooth.

De HCCSimpleVisionModule

De HCCSimpleVisionModule dient om eenmalig de afmetingen van het veld te registreren waaronder de positie van de doelpalen en het calibreren van de diverse kleuren voor de beeldherkenning. De beeldherkenning bepaalt vervolgens dynamisch de plaats van de bal, de plaats van de spelers en de richting waarin een speler staat. Al deze informatie stuurt hij volgens het in HCCGenericVisionModule beschreven contract naar de RefereeModule.
Deze module is een gestripte versie van het bij MRDS bijgeleverde SimpleVision project te vinden in de Microsoft Robotics Dev Studio 2008 R3 folder: \samples\Technologies\Vision\SimpleVision.
Deze SimpleVision kan naast kleurvlakken ook gezichten en handbewegingen herkennen om de robot te sturen. Voor het voetbalproject is vooral het herkennen van kleuren belangrijk, dus de rest is geschrapt. De functionaliteit voor het bepalen van welke kleurencombinatie een robot voorstelt en de richting waarin deze kijkt is toegevoegd aan het SimpleVision project.
Hieronder is de HCCSimpleVisionModule afgebeeld. De pijlen geven de richting van de spelers weer.



Het testveld voor beeldherkenning

Hoewel het er op bovenstaand plaatje al aardig op begint te lijken moet vermeld worden dat dit een A4 voetbalveld is met magneten op een WhiteBoard vastgehouden en met magneten als spelers.
De praktijk bleek namelijk een stuk weerbarstiger.

De praktijk

De infrastructure

In de praktijk hadden we al op eerdere HCCRobotica bijeenkomsten ervaren dat het laten communiceren van de HCCRefereeModule en de HCCPlayerModules vanaf verschillende computers nog niet zo makkelijk was. Hoewel op een enkele machines alles meestal goed draaide, zolang die machine het een beetje bij kon benen, wilde het op een network niet altijd lukken. Vage dingen zoals niet of pas erg laat registreren van teams en geen contact kunnen leggen vanuit VPL met de server die wel via Ping te benaderen is, zijn een paar van deze probleempjes.
Ook nog het noemen waard is dat we veel registratie, calibratie en controle functionaliteit hebben ingebouwd maar niet de mogelijkheid om aan te geven welk team op welke helft gaat spelen. Uh?

De camera ophanging

De camera moet ongeveer op een hoogte ter grootte van de diagonaal van het veld hangen.
Het veld moet dan wel helemaal zichtbaar zijn, de lijnen moeten liefst parallel lopen aan de zijkanten van het beeld want anders moet er in software gecompenseerd worden wat weer extra rekenkracht kost. Dit betekent dat de camera een soort richtmechanisme moet krijgen want eronder gaan staan en aanpassen met de hand is ondoenlijk. De arm moet ook goed stijf zijn zodat de camera niet op en neer danst en de bal onterecht “uit” gegeven wordt. Hiervoor moet dus nog een draad gespannen worden als bij een hangbrug.

De verlichting

Verlichting is wel de meest kritische factor bij beeld herkenning. Bij teveel licht worden alle kleuren flets en lijken te veel op elkaar terwijl bij te weinig licht er van kleur helemaal geen sprake is.
Nu was er op de open dag op 1 oktober 2011 eigenlijk teveel licht door de niet witte plafond verlichting. De ervaring leert overigens dat natuurlijk ambient licht het beste werkt.
Enkele mogelijke oplossingen zijn:

1. Voor iedere situatie de driver van de camera calibreren,
2. Calibreren met kleurenkaarten, een soort testbeeld, met verschillende tinten en groottes van een bepaalde kleur erop zodat meteen duidelijk is hoever je callibratie ernaast zit,
3. Altijd een eigen lamp meenemen die al het andere licht overstemt en daar op calibreren,
4. Calibreren ten opzichte van de som van alles wat de camera ziet binnen de statisch gekozen hoekpunten
5. De robots in het begin op vaste plekken en richting laten beginnen en hun paden volgen zodat het herkennen minder moeite kost en
6. Zwart/ wit labels gebruiken met vierkante patronen erop

Verder is de performance van de SimpleVisionModule door allerlei tekenacties zo laag dat er slechts ongeveer 5 maal per seconde een update verstuurt wordt wat weer veel te weinig is.

HCC Soccer Project op Codeplex

Al deze software is te downloaden van ons open source project http://hccsoccer.codeplex.com/
Je kunt de voortgang volgen door in te loggen met een Windows Live ID of een Codeplex ID en vervolgens op bovengenoemde pagina de “follow” link te klikkken.
Wil je mee ontwikkelen klik dan de “bberrevoets” link op de voorpagina en vervolgens “contact bberrevoets”.

Verdere ideeen en suggesties zijn welkom via CodePlex of mail naar iwan.tolboom@chello.nl.

En dan als uitsmijter een visioen van een hoop gepimpte Fez mini’s:
RoboCup 2011: Small Size League final: http://www.youtube.com/watch?v=Qj_YYWELS4o