Microcontrollers & Programmeertaal

Samenstelling: Peter Dams.

TECHNISCH INSTITUUT SINT-PAULUS
Kruisven 25
2400 Mol

6 TSO-Elektriciteit - Elektronica

Woord van dank.

Bij het uitwerken van een cursus als deze is het belangrijk dat je voor bepaalde zaken kan terugvallen op mensen die meer ervaring hebben dan jijzelf. Ik heb het geluk gehad, steeds op de hulp van volgende personen te kunnen rekenen:

- Marc Alberts van "MCS-electronics"; het is een voorrecht op de hulp te mogen rekenen van de 'maker' van BASCOM. Steeds een antwoord en altijd 'to the point'. Tevens is hij het die de vertaling van deze cursus naar het de engelse taal voor zijn rekening heeft genomen
- Ludwig Berckmans van "Elcom elektronica"; het zoeken naar de juiste hardware, connectors, … het verliep allemaal uiterst vlot. Het is duidelijk dat zijn kennis van de elektronicamarkt een doorbraak betekende in het ontwerp van het hardwarebord dat bij deze cursus hoort.
- Frans Heykants, verzorgde alle tekeningen die in deze cursus terug te vinden zijn. Met een enorme zin voor afwerking en perfectie maakte hij de tekeningen. Frans is ook een uiterst kritisch elektronicaliefhebber, iets wat bij het maken van deze cursus uiteraard van pas kwam.

Zonder de hulp van deze mensen zou de cursus niet zijn wat hij nu is.

Tenslotte nog een woord van dank aan alle collega's leerkrachten en gebruikers van deze cursus voor de opmerkingen en verbeteringen. Met de hulp van al deze mensen verdwijnen misschien ooit nog alle fouten uit deze cursus.

Bedankt voor de toffe (en hopelijk nog lange) samenwerking!

Peter Dams

1. PROGRAMMEEROMGEVING.

Als programmeeromgeving (integrated development enviroment = IDE) gebruiken we de Bascom-AVR van de firma MCS-electronics. Deze omgeving is op de bijgevoegde CD terug te vinden. Indien je zeker wil zijn dat je de meest recente versie hebt, dan kan je steeds een kijkje nemen op de website van deze firma (www.mcselec.com). De gebruikte software is een Basic-compiler voor de AVR-microcontrollers van de firma ATMEL. De datasheets van de gebruikte IC's zijn eveneens op de CD terug te vinden. Ook kan je ze terugvinden op de website van ATMEL (www.atmel.com).

De versie van de BASCOM die wij gebruiken is een demoversie. Deze versie is gratis te gebruiken en heeft de volledige functionaliteit van de commerciële versie. De enige beperking is dat de gegenereerde code maximaal 2kB is. Voor ons is dit echter geen beperking daar wij ons gaan toespitsen op de AT90S2313 en deze heeft slechts 2kB programmageheugen. Indien je echter meer code wil genereren, kan je steeds een (goedkope) commerciële versie bestellen via de website van MCS-electronics.

Daar de gebruikte versie van de BASCOM gratis is, is het niet realistisch om ondersteuning van dit programma vanuit MCS-electronics te verwachten. Voor vragen en opmerkingen die je hebt in verband met deze cursus, projecten die je bouwt en het gebruik van BASCOM kan je steeds terecht bij Peter Dams (dams.peter@advalvas.be). Bijna dagelijks wordt de mail gecontroleerd. Je kan dus snel een antwoord verwachten.

 Indien er wijzigingen aan de teksten of voorbeeldprogramma's worden doorgevoerd, dienen deze te worden doorgegeven aan de auteur. Alle opbouwende kritieken, ideeën en opmerkingen zijn welkom.

Om de programmeeromgeving onder de knie te krijgen gaan we de stappen die je moet doorlopen inoefenen aan de hand van een eenvoudig programma. Dit programma zal een aantal LED's die aan poort B van de microcontroller zijn aangesloten laten oplichten. Een uiterst prijsvriendelijke microcontrollerprint met alle noodzakelijke hardware, waar alle oefeningen op kunnen worden uitgetest is ook ter beschikking. In optie zijn tevens verschillende IC's verkrijgbaar die we in deze cursus gaan aansturen. Voor meer informatie verwijs ik naar de website www.elcom.be .

Tenslotte wil ik vermelden dat het niet de bedoeling is om een volledige cursus BASCOM te maken. Het is enkel de bedoeling beginners een eind op weg te zetten. Daarna is het aan u zelf om aan de hand van de "help-functie" en veel oefening nieuwe dingen uit te proberen. Tevens is het altijd mogelijk dat er tussen de recentste versie van BASCOM en deze cursus kleine verschillen zijn. Microcontrollertechniek is immers een uiterst "levende" materie.

1.1. Editeren.

Als eerste stap maken we gebruik van de editor van de programmeeromgeving. Een editor is een eenvoudige tekstverwerker die ons toelaat om het programma dat we willen uitvoeren in te geven. Als editor kan in principe gelijk welke tekstverwerker (MS-Write, Word, Notepad, …) gebruiken. De ontwerper van BASCOM heeft echter zelf een editor in de programmeeromgeving voorzien. Daar dit de makkelijkste manier van werken is gaan we gebruik maken van deze editor.
Via **FILE→ NEW** kan je een nieuw werkvlak krijgen om je programma in te geven. Tik onderstaande tekst in of open het bestand *VB01.BAS* via **FILE, OPEN** dat je op de CD terugvindt.

```
'Mijn eerste programma
Config Portb = Output
Portb = 170
End
```

Wanneer je dit programma hebt ingegeven zal je merken dat de editor bepaalde delen van je programma automatisch in verschillende kleuren zal weergeven. Dit is een teken dat de editor deze delen tekst herkent. De eerste lijn zal in het groen verschijnen. De editor zet automatisch alle commentaar in het groen. Commentaar is extra tekst, die in principe niets met het programma te maken heeft. Dit wil zeggen dat deze lijn later niet in code wordt omgezet. Commentaar dient om het programma te verduidelijken zodat je later weer weet wat je met bepaalde instructies wilde doen. Het plaatsen van commentaar is ook nuttig wanneer anderen met jouw programma aan de slag moeten. Ze zullen sneller de werking van jouw programma begrijpen. Het is van groot belang dat je je programma goed documenteert. Op het moment dat je een programma schrijft weet je nog wel wat de bedoeling is. Wanneer je echter later aanpassingen moet doorvoeren is het soms moeilijker om nog te weten wat je vroeger juist hebt gedaan en waarom. Met andere woorden: een programma zonder nuttige commentaar is een slecht programma. Om een lijn commentaar te schrijven begin je deze lijn met een enkel aanhalingsteken '.

Verder zal je zien dat er woorden blauw zijn geworden. Dit is het teken dat BASCOM deze woorden als instructies heeft herkend. In de uitgebreide 'help' die bij BASCOM wordt bijgeleverd kan je alle instructies, met een toelichting van hun werking, terugvinden. Om de uitleg van een instructie te krijgen ga je met de cursor op die instructie staan en druk je op functietoets **F1** (probeer dit uit).

1.2. Compileren en debuggen.

BASCOM is een Basic-compiler. Dit wil zeggen dat men heeft gekozen voor de programmeertaal 'Basic'. Dit is een vrij eenvoudige taal waarvan vele dialecten in omloop zijn. De basis van de taal is echter steeds hetzelfde. Dit is trouwens het voordeel van elke hogere programmeertaal. De taal blijft dezelfde, onafhankelijk van de gekozen processor. Dit in tegenstelling tot een lagere programmeertaal (assembler) die van processor tot processor totaal kan verschillen.

BASCOM is tevens een compiler. Dit betekent dat het ingegeven programma eerst in zijn geheel wordt omgezet naar machinetaal (de taal van de processor). Pas daarna wordt het uitgevoerd door een processor. De tegenhanger van een compiler is een interpreter. Deze zal de eerste lijn van jouw programma omzetten naar machinetaal. Daarna wordt de eerste lijn uitgevoerd. Vervolgens wordt de tweede lijn omgezet naar machinetaal. Deze wordt vervolgens uitgevoerd, enz… Zoals je wellicht aanvoelt zal een compiler sneller werken dan een interpreter.

Tijdens het compileren gaat de compiler controleren of je geen 'taalfouten' hebt geschreven. Indien je fouten maakt tegen de spellingsregels van de 'Basic' zal de compiler een 'syntax fout' genereren. Je moet eerst ervoor zorgen dat het programma wat je hebt ingegeven geen syntaxfouten meer bevat, pas dan zal de compiler zorgen voor een omzetting naar machinetaal.

Let wel: een programma zonder syntaxfouten betekent niet dat je programma juist is. Je kan immers ook een tekst schrijven die correct is wat spelling en spraakkunst betreft maar die qua inhoud klare onzin is. Zo is dat ook voor een programma. Na het compileren zal je nog moeten uittesten of je programma wel degelijk doet wat je er van verwacht. Het opsporen en oplossen van fouten noemen we debuggen.

Het starten van de compiler doe je op volgende manier: je drukt op functietoets **F7** ofwel klik je op het compiler-icoon. Dit is het zwarte IC-symbool in de taakbalk van je scherm.

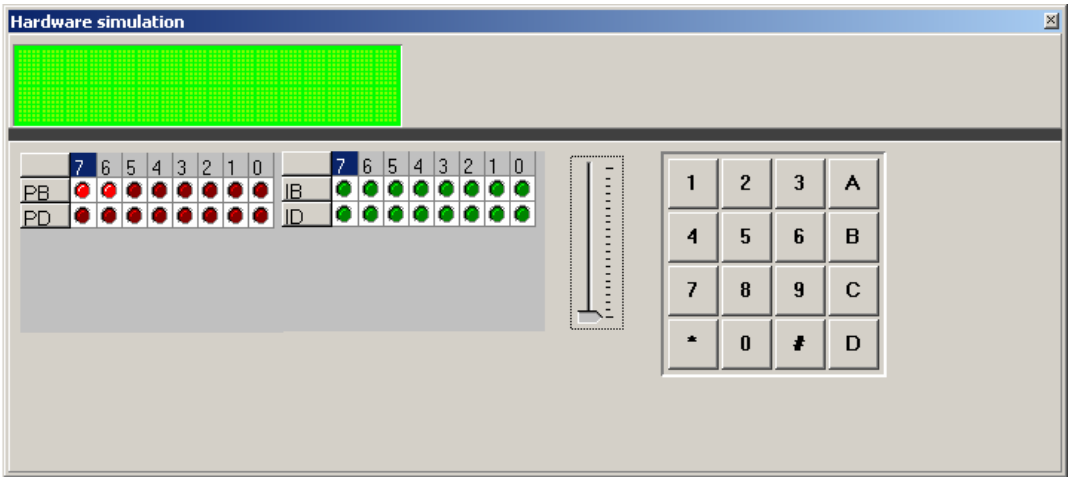
Indien er fouten worden gevonden wordt dit in het onderste deel van je scherm weergegeven. Het is de bedoeling dat je systematisch alle fouten wegwerkt.

1.3. Simuleren.

Wanneer je een programma hebt dat geen syntax-fouten meer bevat is het moment aangebroken om de functionaliteit van je programma uit te testen. Het makkelijkst is om dit eerst op een simulator te doen en nadien op de gekozen hardware. Een simulator is een softwarepakket dat de werking van de processor, samen met wat periferie, gaat nadoen. Bascom laat ons toe om met verschillende simulatoren te werken.

In de opties moet je enkel instellen met welke simulator je graag werkt en dan wordt de simulator van jouw keuze automatisch opgestart. Een uitstekende simulator is deze die in de AVR-studio van ATMEL zit. Deze kan je op je CD terugvinden. Wij kiezen echter voor de ingebouwde simulator van BASCOM zelf. Voor onze doeleinden voldoet deze immers perfect aan de noden. Het opstarten van de simulator doe je op volgende manier: je drukt op functietoets **F2** ofwel klik je op het simulator-icoon. Dit is het rode IC-symbool in de taakbalk van je scherm.

Zoals je zal merken is het programma wat we hebben ingetikt reeds geladen in de simulator. Om te kunnen zien wat ons programma doet moeten we de hardware die we gebruiken zichtbaar maken. Dit doen we door op het blauwe LCD-icoon te klikken. We krijgen dan een extra scherm met daarin een vlak dat de LCD voorstelt (hierover later meer) en een aantal LED's. De LED's stemmen overeen met de twee poorten (B en D) van de AT90S2313. Er zijn rode LED's die de uitgang van de poort (PB en PD) voorstellen en er zijn groene LED's die de ingang van de poort (IB en ID) voorstellen. Een logische '1' wordt voorgesteld door een LED die oplicht. Een logische '0' door een LED die gedoofd blijft.



We kunnen ons programma nu stap voor stap laten uitvoeren. Dit doen we door het step-in-icoon aan te klikken.

Op de taakbalk vinden we de iconen die met de uitvoering van ons programma te maken hebben. Het linkse symbool is het run-icoon. Het start ons programma en daardoor worden de verschillende programmalijnen ononderbroken na mekaar uitgevoerd. Het tweede icoon is het pauze-icoon. Dit onderbreekt de run-mode. Druk je nadien op de run-knop dan ga je gewoon verder met datgene waar je mee bezig was toen je de simulator hebt onderbroken. De derde knop is de stop-knop. Deze onderbreekt de run-mode en reset de gehele processor. Zo kan je terug vanaf nul starten. De volgende drie iconen zijn step-knoppen. Zij laten je toe om stap voor stap ofwel regel per regel je programma te doorlopen. Op de onderlinge verschillen komen we later terug. Wij gebruiken nu echter de linkse van deze drie ofwel de step-into-code knop.
Je zal zien dat na vier maal op deze knop te drukken er een vier LED's op poort B oplichten en vier blijven er gedoofd. Dit stemt overeen met binair 10101010. Wanneer je deze code omzet naar het decimale talstelsel, dan bekom je 170.
Let wel: indien je in de simulator veranderingen doet aan je programma zal dit geen effect hebben op de uitvoering ervan. Wijzigingen moet je immers in de editor doorvoeren. Daarna moet je compileren en met het resultaat van de compilatie kan je in de simulator aan de slag.

1.4. Uittesten van het programma op hardware.

Nu ons programma werkt zoals wij het graag zouden willen, komen we bij de laatste stap in de ontwikkeling van een programma: de 'real-life test'. Om het programma door te sturen naar onze hardware-print maken we gebruik van de 'Sample Electronics programmer'. Dit is één van de vele programmers die BASCOM ondersteunt. Voor meer informatie verwijs ik naar de handleiding die bij het printje hoort. Om de programmer te starten druk je op het groene icoontje dat een ZIP-IC-voet voorstelt.

In het programmerscherm klik je weer op hetzelfde icoon en BASCOM doet de rest. Dit is: de processor wissen, daarna programmeren en vervolgens het programma controleren of het wel goed in de processor is aangekomen. Vervolgens wordt print gestart zodat het programma automatisch wordt uitgevoerd. Indien alles goed is verlopen zouden er nu vier van de acht LED's moeten branden.

OPGAVE.

Probeer eens om alle LED's van poort B te laten oplichten! De oplossing kan je terugvinden in VB02.BAS.

2. GESTRUCTUREERD PROGRAMMEREN.

Om een programma te schrijven gebruiken we verschillende structuren (herhalingen, testen, …). In een hogere programmeertaal zijn deze structuren als het ware ingebakken. Dit wil zeggen dat deze structuren ook in BASCOM zitten ingebakken. De structuren in andere programmeertalen zijn vergelijkbaar zodat het later dus makkelijk is om op een andere hogere programmeertaal over te stappen. In dit deel gaan we de verschillende structuren bestuderen en uitvoerig inoefenen.

2.1. Declareren van variabelen en constanten.

Een computerprogramma verwerkt gegevens die steeds veranderen. Om met deze gegevens te kunnen werken en rekenen moet de microcontroller deze gegevens kunnen bewaren. Dit gebeurt in het geheugen van de microcontroller. Als u nu gegevens wil gebruiken moet u aangeven waar deze in het geheugen staan. Om nu op een eenvoudige manier deze gegevens te benaderen geven wij aan, de op te slaan gegevens een naam. Deze naam noemen we variabele. Een constante gebruikt geen extra geheugen maar wordt op eenzelfde manier aangesproken. Een variabele is een gegeven dat kan veranderen (bv. Temperatuur, tijd, …), een constante is een gegeven dat niet zal veranderen (bv. Pi=3,1415….).
Variabelen worden in alle programmeertalen gebruikt om een naam te geven aan een specifieke plaats in het geheugen. Als je een variabele hebt gedefinieerd, blijft deze naar dezelfde geheugenplaats wijzen, totdat hij terug wordt vrijgegeven. Maak je echter geen zorgen: je hoeft niet de plaats in het geheugen op te geven waar de informatie moet worden opgeslagen (BASCOM doet dat voor jou); je moet de variabele alleen een naam geven, zodat je telkens dat nodig is naar de geheugenlocatie kan verwijzen.
Bij het benoemen van een variabele heb je enorme keuze. Je kan namen van variabelen en constanten eenvoudig houden of je laat ze de inhoud die ze bevatten nauwkeurig beschrijven. Je kan bijvoorbeeld de naam van een telvariabele gewoon I noemen, maar ook een meer beschrijvende naam gebruiken, bijvoorbeeld AantalStuks. Wij geven enkel betekenisvolle namen aan onze variabelen. Hoewel je een aanzienlijke vrijheid hebt bij het geven van namen, zijn er enkele beperkingen:

- Een naam voor een BASCOM variabele mag tot 32 karakters lang zijn.
- Geldige karakters zijn letters en cijfers.
- Het eerste karakter van een naam van een variabele moet een letter zijn.
- Een naam voor een variabele mag geen gereserveerd woord zijn. Samenstellingen met gereserveerde woorden zijn echter wel toegestaan.

Bijvoorbeeld, het volgende statement is niet toegestaan omdat AND een gereserveerd woord is.

AND = 8

Het volgende statement is echter wel toegestaan:

ToAND = 8

Gereserveerde woorden omvatten alle BASCOM commando's, statements, functienamen, interne registers en operatornamen. Een lijst van gereserveerde woorden kan u in bijlage terugvinden.

Je kan een hexadecimal of binair getal voorstellen met het prefix &H of &B.

a = **&H**A , a = **&B**1010 en a = 10 zijn allemaal gelijk.

Het aantal geheugenplaatsen dat de compiler reserveert voor een variabele is afhankelijk van het gekozen datatype. Als we bijvoorbeeld enkel moeten bijhouden of een klep open staat of toe hebben we voldoende aan één bit (0 of 1). Als we het aantal Colaflesjes in een bak moeten tellen dan zal de variabele AantalFlesjes maximaal 24 worden. Het heeft dus geen zin om hiervoor veel geheugenplaatsen te voorzien. Je kan aan de compiler laten weten hoeveel plaats hij moet voorzien door de variabele te dimensioneren. Dit is vertellen welk datatype overeenkomt met de gebruikte variabelen. In BASCOM ben je verplicht elke variabele te declareren (dit is trouwens een goed principe). Je hebt de keuze uit volgende datatypes:

- Bit (1/8 byte). Een bit kan de waarde 0 of 1 hebben.
- Byte (1 byte = 8 bits). Dit is een unsigned integer. Met integer bedoelen we een geheel getal (dus géén decimale) met unsigned bedoelen we dat we geen teken bijhouden (alle getallen zijn dus positief). Met een Byte kunnen we dus getallen bijhouden van 0 tot 255.
- Integer (twee bytes). Integers worden bewaard als signed 16-bit integers. We kunnen dus enkel gehele getallen bewaren. Het getallenbereik voor een Integer is van –32.768 tot +32.767.
- Word (twee bytes). Words worden bewaard als unsigned 16-bit integers. We kunnen dus enkel positief gehele getallen bewaren. Het getallenbereik voor een Word is van 0 tot 65.535.
- Long (Vier bytes). Longs worden bewaard als signed 32-bit gehele getallen met een getallenbereik van –2.147.483.648 tot 2.147.483.647.
- Single. Singles worden bewaard als signed 32-bit getallen met drijvende komma (floating point).
- String (tot 254 bytes). Strings worden bewaard als bytes en worden afgesloten met een 0-byte. Een string gedimensioneerd met een lengte van 10 bytes beslaat dus 11 bytes.

Variabelen kunnen zowel intern (standard of default), extern als in EEPROM worden opgeslagen (zie later). Verder spreken we af dat we de naam van de variabele laten beginnen met een letter die weergeeft van welk datatype de variabele is. Dit noemt men de 'hungarian notation' voor variabelen.

Een bit-variabele begint met bt
Een byte-variabele begint met b
Een integer-variabele begint met i
Een word-variabele begint met w
Een long-variabele begint met l
Een single-variabele begint met s
Een string-variabele begint met st

Om een variabele toe te kennen, moet je de compiler hiervan op de hoogte brengen met het DIM statement.

Dim btLed1 As Bit, iIngang as Integer, bKilo as Byte, stNaam As String * 10

Het STRING type heeft een bijkomende parameter nodig om de lengte op te geven.

Verder gaan we in deze cursus nog gebruik maken van ARRAY's. Een array is een verzameling van sequentieel geïndexeerde elementen die allemaal hetzelfde datatype hebben. Elk element van een array heeft een uniek indexnummer. Wanneer je één element uit een array verandert, veranderen de andere elementen niet. De index is een getal (byte, integer, word of long). Het maximaal aantal elementen in een array is 65535. Het eerste element in een array is steeds één. Dit betekent dat de elementen one-based zijn.

```
Dim a(10) as byte  'maak een array die a noemt met 10 elementen
```

Een element kunnen we bereiken door zijn index tussen haakjes mee te geven.

a(5)

a(teller)

2.2. Structuren.

2.2.1. Sequentie.

Een sequentie is een opeenvolging van commando's. Deze worden één na één uitgevoerd, eerst de eerste, dan de tweede, dan de derde, enz. Als alle stappen zijn uitgevoerd dan stopt de microcontroller met de uitvoering. Een voorbeeld van een sequentie kan je terugvinden in onderstaand programma *(VB03.BAS)*. Tik het in en ga het effect ervan na in de simulator. Indien je het programma op hardware uittest dan voeg je tussen elke lijn het wachtcommando tussen. Anders is de uitvoering te snel om waar te nemen. Het wachtcommando is: WAIT 1 (de 1 staat voor het aantal seconden dat de microcontroller 'wacht').

```
Config Portb = Output

Portb = 0
Portb = 1
Portb = 2
Portb = 4
Portb = 8
Portb = 16
Portb = 32
Portb = 64
Portb = 128

End
```

OPGAVE

Pas de sequentie zodanig aan dat de LED's nu in de andere richting lopen. (oplossing *VB04.BAS*)

2.2.2. Selectie.

In de loop van een programma kan het nodig zijn om bepaalde delen uit te voeren afhankelijk van een bepaalde voorwaarde. Bijvoorbeeld ALS het te koud is DAN moeten we verwarmen. Deze structuur wordt dan ook de IF – THEN structuur genoemd. De IF – THEN wordt in verschillende vormen gebruikt.

2.2.2.1. Enkelvoudige selectie.

Dit is de meest eenvoudige vorm. We bekijken eerst de code van *VB05.BAS*:

```
Dim bIngang As Byte

Config Portb = Output
Config Portd = Input

bIngang = Pind
If bIngang > 1 Then
   Portb = 255
End If
```

Eerst declareren we de byte-variabele 'bIngang'. Vervolgens definiëren we poort b als uitgang en poort d als ingang. We lezen de informatie die staat op poort d in en bewaren deze informatie in de variabele bIngang. Daarna komen we bij de selectiestructuur. Wanneer de informatie die we ingelezen hebben groter is dan één, dan sturen we op poort b de waarde 255 uit. Met andere woorden, op dat moment worden alle uitgangen van poort b hoog. Indien de waarde van bIngang gelijk is aan één of nul, dan gaan we onmiddellijk naar 'End If'. Dit statement moeten we gebruiken om de selectiestructuur af te sluiten.
Bij een enkelvoudige structuur gaan we de statements (programmacode) die staan tussen 'Then' en 'End If' uitvoeren indien de voorwaarde die na de 'If' staat waar is. Algemeen kunnen we de enkelvoudige selectiestructuur als volgt voorstellen:

```
If voorwaarde Then
    Statements
End If
```

OPGAVE:

Voer het programma uit en test het met verschillende waarden voor de ingang.

2.2.2.2. Samengestelde selectie.

We kunnen nu de voorwaarde die we opgeven gaan uitbreiden en afhankelijk maken van een aantal voorwaarden. We gaan voor de voorwaarde een samengestelde voorwaarde gebruiken. Deze bestaat uit uitdrukkingen (expressions) en operatoren. We kennen een aantal verschillende operatoren:

- Rekenkundige operatoren, deze voeren berekeningen uit.
- Relationele operatoren, deze gebruiken we om getallen en strings te vergelijken.
- Logische operatoren, deze gebruiken we om toestanden te testen en om individuele bits aan te passen.
- Functionele operatoren, deze gebruiken we om de eenvoudige operatoren uit te breiden.

Uitdrukkingen en operatoren

Een uitdrukking kan een numerieke constante zijn, een variabele of een waarde die we bekomen door verschillende constanten, variabelen en andere uitdrukkingen te combineren door middel van operatoren. Deze laatste soort gebruiken we in een samengestelde selectie.

Rekenkundige operatoren (Arithmetic operators)

Als rekenkundige operatoren kent BASCOM +, - , * , \, / en ^.

- Integer

  Deling met gehele getallen wordt voorgesteld door de backslash (\).
  Voorbeeld: Z = X \ Y

- Modulus

  De modulus wordt berekend met de modulus operator **MOD**.

  De modulus geeft als uitkomst de rest van een deling van gehele getallen in plaats van het quotiënt.

  Voorbeeld: X = 10 \ 3 ; rest = 10 MOD 3

  Na uitvoering van deze code zal X=3 en rest =1

- Overflow en delen door nul

  Delen door nul geeft een error.

  Een overflow treedt op wanneer je de boven- of ondergrens van een datatype overschrijdt. Stel dat je een byte-variabele hebt die de waarde 200 bevat. Wanneer je daar 100 bijtelt zou je 300 moeten uitkomen. Een byte kan echter maximaal 8 bits bevatten en 300 bevat er 9. Dit zal tot

gevolg hebben dat de hoogst bit komt te vervallen. De uitkomst van 200 + 100 zal dus 44 worden. Dit verschijnsel noemen we overflow. (Zie Basiselektronica in 4EE).

Op overflow vindt momenteel nog geen controle plaats in BASCOM. Je zal dus zelf moeten uitkijken wat je doet!

Relationele Operatoren

Relationele operatoren worden gebruikt om twee waarden met elkaar te vergelijken. Hoe dit gebeurt kan je in onderstaande tabel terugvinden.

Operator	**Geteste Relatie**	**Uitdrukking**
=	Gelijkheid	X = Y
<>	Ongelijkheid	X <> Y
<	Kleiner dan	X < Y
>	Groter dan	X > Y
<=	Kleiner dan of gelijk aan	X <= Y
>=	Groter dan of gelijk aan	X >= Y

Logische operatoren

BASCOM kent vier logische operatoren.

Operator	**Betekenis**
NOT	Logisch complement
AND	Conjunctie
OR	Disjunctie
XOR	Exclusive or

We kunnen de logische operatoren gebruiken om bytes te testen voor een bepaald bitpatroon. De AND kunnen we gebruiken om bepaalde bits te maskeren met een ‘nul’ en de OR kunnen we gebruiken om bepaalde bits te maskeren met een ‘één’. De XOR kunnen we dan weer gebruiken om selectief bits te complementeren.

Als voorbeeld van het gebruik van deze operatoren in een selectiestructuur nemen we de code van *VB06.BAS*.

OPGAVE

Probeer de toestand van poort b te voorspellen voor verschillende toestanden van poort d.

In dit deel van de cursus gaan we ook de combinatie IF-THEN-ELSE bekijken. Deze werkt als volgt:

```
If voorwaarde Then
        Statements1
Else
        Statements2
End If
```

In deze code wordt het programmadeel 'Statements1' uitgevoerd als de uitkomst van de voorwaarde 'True' is, indien de uitkomst van de voorwaarde 'False' is, wordt het programmadeel 'Statements2' uitgevoerd. Een voorbeeld van deze code kan je terugvinden in *VB07.BAS*.

OPGAVE

Bekijk de code en test uit!

2.2.2.3. Genestelde selectie.

In plaats meerdere selecties en samengestelde voorwaarden te gebruiken kunnen we ook een genestelde selectie toepassen. Met een genestelde selectie bedoelen we een selectie uitvoeren in een selectie. De keuze tussen beide manieren van werken is vrij te kiezen, maar kies steeds die manier die de best leesbare code oplevert. Vergelijk het eerste deel van de code in *VB08.BAS* met het tweede deel. Beide delen hebben dezelfde uitwerking. Toch is er nog een verschil tussen beide codes. Vergelijk de tijd die het programma nodig heeft om door het eerste deel van de code te lopen met de tijd nodig om het tweede deel te doorlopen. In het eerste deel hebben we altijd drie testen te doorlopen. In het tweede deel van de code is dit maximaal twee testen. Het tweede deel van de code is in dit geval te verkiezen boven het eerste deel.

Let eveneens op de insprongen die je in de code terug kan vinden. Zij vergroten de leesbaarheid van de code, daarom programmeren wij enkel op deze manier.

```
Config Portb = Output
Config Portd = Input


'deel 1 twee selecties na mekaar
If Pind.0 = 1 And Pind.1 = 1 Then
  Portb = 1
End If

If Pind.0 = 1 And Pind.1 = 0 Then
  Portb = 2
End If

If Pind.0 = 0 Then
  Portb = 3
End If


'deel 2 genestelde selectie
If Pind.0 = 1 Then
  If Pind.1 = 1 Then
    Portb = 1
  Else
    Portb = 2
  End If
Else
  Portb = 3
End If


End
```

OPGAVE

Test de code uit!

OPGAVE

Ga uit van de schakeling in bijlage 1. Schrijf een programma dat de LED's doet lopen indien schakelaar 2 is ingedrukt. Indien schakelaar 2 niet is ingedrukt moeten alle LED's oplichten. De looprichting is links indien schakelaar 3 is ingedrukt. Indien schakelaar 3 niet is ingedrukt moeten de LED's rechts lopen. De oplossing kan je terugvinden in *VB09.BAS*.

2.2.2.4. Meervoudige selectie.

Wanneer we met behulp van de IF-THEN structuur een selectie moeten maken uit zeer veel mogelijkheden dan wordt de leesbaarheid van het programma zeer slecht. Tevens moeten telkens veel testen worden doorlopen. Om dit op te vangen bestaat de meervoudige selectie of de SELECT-CASE-structuur.

De uitwerking van deze structuur is als volgt:

SELECT CASE var
CASE test1 : *statements*
[CASE test2 : *statements*]
CASE ELSE : *statements*
END SELECT

Opmerkingen

Var	Variabele die moet getest worden
Test1	Testvoorwaarde 1.
Test2	Testvoorwaarde 2.

Je kan testen op voorwaarden zoals:
CASE IS > 2 :
Een andere mogelijkheid is te testen op een bereik:
CASE 2 TO 5 :

Nota:
De SELECT – CASE structuur voert de statements uit die horen bij de eerste test die als uitkomst “True” geeft.

Een voorbeeld van de SELECT-CASE structuur kan je terugvinden in *VB10.BAS*. In dit programma komen we nog twee andere statements tegen namelijk INPUT en PRINT.

INPUT en PRINT zijn twee commando’s die gebruik maken van de RS-232 verbinding van de microcontroller met een ander toestel (in ons geval onze PC). Met INPUT kunnen we via de RS-232 een aantal toetsaanslagen van ons toetsenbord binnenlezen en deze toekennen aan een variabele. Met het PRINT-statement werken we andersom. Hier gaan we de inhoud van een variabele via de RS-232 verbinding versturen. Dit wordt dan in ons geval op het scherm weergegeven. Wanneer je gebruik maakt van de simulator komt de ontvangen tekst in het blauwe scherm.

```
Dim bIngave As Byte


Input "Enter value (0-255) " , bIngave
Select Case bIngave
   Case 1 : Print "Een"
   Case 2 : Print "Twee"
   Case 3 To 5 : Print "Drie, Vier of Vijf"
   Case 6 : Print "Zes"
   Case 7 : Print "Zeven"
   Case Is >= 10 : Print ">= 10"
                   Print "Groter dan tien"
   Case Else : Print " Niet in Case statement, dus Acht of Negen"
End Select


End
```

OPGAVE

Maak een programma dat vraagt hoeveel dagen na zondag we zijn. De ingave gebeurt via het toetsenbord. Je programma moet dan via het scherm weergeven welke weekdag je op dat moment bent. Indien we '2' opgeven moet je dus 'dinsdag' krijgen, als je '10' geeft moet je 'woensdag' krijgen, enz..

Een oplossing kan je terugvinden in *VB11.BAS*.

2.2.3. Iteratie of herhaling

In de loop van een programma kan het nodig zijn om bepaalde delen een aantal maal uit te voeren. Denk bijvoorbeeld hierbij aan ons looplichtje. BASCOM heeft hiervoor een aantal statements ingebakken.

2.2.3.1. WHILE - WEND

Deze herhalingsstructuur voert het aantal statements tot 'WEND' uit, zolang een bepaalde voorwaarde "true" is.

WHILE voorwaarde

    statements

WEND

Opmerking:

Wanneer de voorwaarde waar is, dan worden alle volgende statements uitgevoerd tot het WEND-statement wordt tegengekomen. BASCOM keert dan terug naar het WHILE-statement en controleert terug de voorwaarde. Wanneer deze nog steeds waar is, dan herhaalt het proces zich. Wanneer de voorwaarde niet meer waar is, dan wordt er verder gegaan met het eerstvolgende statement na het WEND-statement. Dit wil zeggen dat wanneer de test van de voorwaarde, de eerste keer reeds 'false' geeft, dat de statements tot WEND nooit worden uitgevoerd.

Een voorbeeld van een WHILE-WEND structuur kan je terugvinden in onderstaand voorbeeld *VB12.BAS*.

```
Dim bTeller As Byte

bTeller = 1
While bTeller < 10
  Print bTeller
  Incr bTeller
Wend
End
```

Merk in bovenstaand voorbeeld op dat we de statements tussen de WHILE en de WEND laten inspringen. Dit bevordert de leesbaarheid van het programma en dit doen we dus steeds op deze manier. Verder maken we gebruik van het INCR statement om de variabele bTeller met één te verhogen. We zouden dit ook als volgt kunnen doen: bTeller = bTeller +1. Toch verkiezen we het increment statement te gebruiken. De reden hiervoor is dat BASCOM voor het uitvoeren van de increment minder code genereert dan voor het uitvoeren van een optelling.

OPGAVE

Herschrijf het looplicht uit *VB03.BAS* door nu gebruik te maken van de WHILE-WEND structuur. Een oplossing kan je terugvinden in *VB13.BAS*.

2.2.3.2. DO - LOOP

Deze herhalingsstructuur voert het aantal statements tot 'LOOP' uit, totdat een bepaalde voorwaarde "true" is. In sommige programmeertalen staat deze structuur ook gekend als REPEAT-UNTIL.

DO

statements

LOOP [UNTIL uitdrukking **]**

Opmerking

Je kan een DO-LOOP verlaten met het EXIT DO-statement. In gestructureerd programmeren vermijden we deze mogelijkheid zoveel mogelijk. Het is de kunst om je test zodanig te schrijven dat de EXIT DO niet nodig is. Het heeft natuurlijk geen zin om je programma extra te overladen met code, simpelweg om de EXIT DO te vermijden. De leesbaarheid van je programma moet blijven primeren.

Verder is de structuur van de DO-LOOP zodanig opgebouwd, dat de code die staat tussen DO en LOOP altijd minstens één keer wordt uitgevoerd. Daarna wordt de uitdrukking aan een test onderworpen.

Een voorbeeld van een DO-LOOP structuur kan je terugvinden in *VB14.BAS*.

```
Dim bTeller As Byte

DO
  INCR bTeller
  PRINT bTeller
LOOP UNTIL bTeller = 10
Print bTeller

End
```

Merk ook hier op dat we de statements tussen DO en LOOP laten inspringen. Dit doen we ook hier om de leesbaarheid te vergroten. Zoals reeds eerder gezegd, proberen wij geen gebruik te maken van het EXIT DO-statement. Dit wil zeggen dat wij steeds een uitdrukking gaan schrijven die we telkens gaan testen. Indien we geen uitdrukking zouden schrijven en gebruik maken van een EXIT DO, dan mag de UNTIL achter het LOOP statement verdwijnen. Je moet dan wel zelf voor een test zorgen (zie *VB15.BAS*).

DO

Statements

IF voorwaarde **THEN**

EXIT DO

END IF

LOOP

Indien je dit niet doet, dan blijft je programma zich oneindig herhalen.

DO

statements

LOOP

OPGAVE

Herschrijf het looplicht uit *VB04.BAS* door nu gebruik te maken van de DO-LOOP structuur. Een oplossing kan je terugvinden in *VB16.BAS*.

2.2.3.3. FOR – NEXT.

Deze herhalingsstructuur voert het aantal statements tussen FOR en NEXT een aantal keer uit.

FOR var = start **TO** end [**STEP** value]

Statements

NEXT var

var	De variabele die als teller wordt gebruikt
start	De startwaarde van de variabele var
end	De eindwaarde van de variabele var
value	De waarde van de variable var wordt vermeerderd/vermindert met de waarde value telkens de overeenkomstige NEXT wordt tegengekomen.

- Voor toenemende lussen, moet je TO gebruiken.
- Voor afnemende lussen moet je een negatieve stap opgeven.
- Een FOR-structuur moet steeds met het NEXT-statement worden afgesloten.
- Het gebruik van STEP is optioneel. De defaultwaarde is 1.
- Je kan een FOR-NEXT structuur vroegtijdig afsluiten door het EXIT FOR-statement.
- Het plaatsen van de naam van de variabele bij NEXT is optioneel. Toch maken we er de gewoonte van om dit steeds te doen. Dit bevordert immers de leesbaarheid van de code.

Volgende code (*VB17.BAS*) is een voorbeeld van het gebruik van de FOR-NEXT structuur:

```
Dim bTeller As Byte
Dim iTeller As Integer

For bTeller = 1 To 10 Step 2
        Print "De waarde van bTeller = " ; bTeller
Next bTeller

Print "Nu met een negatieve step"

For iTeller = 10 To -5 STEP -1
        Print "De waarde van iTeller = " ; iTeller
Next iTeller

End
```

OPGAVE

Herschrijf het looplicht uit *VB03.BAS* door nu gebruik te maken van de FOR-NEXT structuur. Een oplossing kan je terugvinden in *VB18.BAS*.

Zoals je kan merken, zijn voor sommige problemen, meerdere structuren bruikbaar als oplossing. De kunst is om de meest efficiënte structuur te kiezen voor jouw specifiek probleem.

2.2.3.4. Genestelde iteratie

De verschillende herhalingsstructuren die we gezien hebben zijn ook te 'nesten'. Als voorbeeld van een geneste herhalingsstructuur bekijken we de code in *VB19.BAS*.

```
Dim bTeller1 As Byte
Dim bTeller2 As Byte


Print "Voorbeeld van geneste FOR..NEXT statements."
For bTeller1 = 1 To 10
        For bTeller2 = 1 To 10
                Print "bTeller1 = " ; bTeller1 ; " bTeller2 = " ; bTeller2
        Next bTeller2
Next bTeller1


End
```

OPGAVE

Maak een looplicht dat 10 keer van links naar rechts en terug loopt. Een mogelijke oplossing is te vinden in de code van *VB20.BAS*.

OPGAVE

Ga uit van de schakeling in bijlage 1. Schrijf een programma dat de LED's doet lopen indien schakelaar 2 is ingedrukt. Indien schakelaar 2 niet is ingedrukt moeten alle LED's oplichten. De looprichting is links indien schakelaar 3 is ingedrukt. Indien schakelaar 3 niet is ingedrukt moeten de LED's rechts lopen. Het programma moet continu werken. Een oplossing kan je terugvinden in *VB21.BAS*.

2.2.3.5. Iteratie als timer of tijdfunctie.

Het is mogelijk om een herhalingsstructuur te gebruiken om een bepaalde wachttijd te creëren. Dit is, in dit geval, enkel zinvol als de tijd niet exact moet zijn. Indien je programmeert in assembler kan je uiteraard exact uitrekenen hoelang een bepaald deel code duurt. Indien je een tijdfunctie nodig hebt, kan je best gebruik maken van de in BASCOM voorziene tijdfuncties. Hiervoor zijn een aantal functies bruikbaar. Je kan ze terugvinden in de handleiding van BASCOM onder:

DELAY

WAIT

OPDRACHT

Bestudeer deze functies en hun afgeleiden

2.3. Subroutines.

Indien er in je programma delen zitten die op verschillende plaatsen regelmatig terugkomen, dan kan je uiteraard deze delen op die verschillende plaatsen telkens herhalen. Het is echter ook mogelijk om deze herhalende delen één keer te schrijven en op de momenten dat je ze nodig hebt, oproepen. Nadat het deelprogramma is uitgevoerd, keert de controller automatisch terug naar de plaats in het hoofdprogramma vanwaar hij is vertrokken. Deze manier van werken noemen we het werken met subroutines.

Het is ook mogelijk een sprong te maken waaraan geen voorwaarde is gekoppeld en waarna het niet meer mogelijk is naar de originele plaats terug te keren. Dit statement noemt het GOTO-statement. Deze manier van werken is echter in een gestructureerde programmeertaal uit den boze en deze gaan wij dan ook niet gebruiken!!!! Het gebruik van de GOTO nodigt immers uit tot het creëren van onleesbare programma's (ook wel eens spaghetti genoemd).

BASCOM laat verschillende manieren van werken toe. Het maakt onderscheid in subroutines, die al dan niet een waarde van het hoofdprogramma verwachten. Het maakt tevens onderscheid tussen subroutines die al dan niet een waarde aan het hoofdprogramma moeten teruggeven. Ook de manier van uitwisselen van gegevens tussen het hoofdprogramma en de subroutine kan op verschillende manier. Laten we al deze verschillende manieren van werken een van naderbij bekijken.

2.3.1. GOSUB.

Met GOSUB springen we naar een subroutine met het gespecificeerde label en voeren de code uit die daar staat. Wanneer we het RETURN-statement tegenkomen, zal de uitvoering van het programma verdergaan met het eerstvolgende statement dat na het GOSUB-statement staat

GOSUB label

Label	De naam van het label waarnaar we springen.

Een voorbeeld kan je terugvinden in de onderstaande code van voorbeeld *VB22.BAS*.

```
Dim bVariabele As Byte


GOSUB Routine
Print "Dit was een eerste oproep"
GOSUB Routine
Print "Dit was de tweede oproep"
END


Routine:
        bVariabele = bVariabele + 2
        PRINT bVariabele
        RETURN
```

Specifiek aan een GOSUB is dat er bij het oproepen van de subroutine geen parameters worden meegegeven. De subroutine geeft ook geen waarde terug. De variabelen die in de subroutine worden gebruikt zijn overal in het programma bruikbaar. We zeggen dat het publieke variabelen zijn.

In een subroutine is het mogelijk om weer een nieuwe subroutine op te roepen.

OPDRACHT

Schrijf een programma dat gebruik maakt van een subroutine dat een looplicht 10 keer van links naar rechts doet lopen. Een mogelijke oplossing kan je terugvinden in *VB23.BAS*.

2.3.2. CALL – subroutine.

Met CALL roep je een subroutine op. De code die in de subroutine staat wordt uitgevoerd en daarna wordt er teruggekeerd naar de eerstvolgende lijn die in het hoofdprogramma volgt na de lijn waarin de CALL stond.

CALL Test [(var1, var-n) **]**

Var1	Gelijk welke BASCOM variabele of constante.
Var-n	Gelijk welke BASCOM variabele of constante
Test	De naam van de subroutine. In dit geval 'Test'

Het is mogelijk om bij het oproepen van subroutines met 'CALL', al dan niet parameters mee te geven.

Het is van belang dat de SUBroutine wordt gedeclareerd (DECLARE) voordat je kan gebruik maken van een CALL naar deze subroutine. Het is tevens van belang dat het aantal gedeclareerde parameters overeenkomt met het aantal parameters dat van het hoofdprogramma wordt doorgegeven naar de subroutine. Met deze parameters kan je waarden doorgeven aan variabelen die alleen in de subroutine gekend zijn. We spreken van locale variabelen of LOCALS.

Het is tevens belangrijk dat wanneer je constanten wil doorgeven aan een subroutine, je deze parameters moet declareren met het BYVAL argument.

Met het CALL statement kan je een zowel een procedure als een subroutine aanroepen.

Bv: **Call Test2(x,y,z)**

Een andere geldige schrijfwijze is: **Test2 x,y,z**

Merk op dat in de tweede schrijfwijze de haakjes zijn weggelaten. Op deze manier kan je zelf je eigen statements aanmaken.

Als voorbeeld bekijken we de code van *VB24.BAS*.

```
Dim bHoofd1 As Byte , bHoofd2 as Byte
Declare Sub Test(bSub1 As Byte , BYVAL bSub2 As Byte)

bHoofd1 = 65
bHoofd2 = 5
Call test(bHoofd1 , bHoofd2)
Print bHoofd1 ; "  " ; bHoofd2

bHoofd1 = 65
bHoofd2 = 5
test bHoofd1 , bHoofd2
Print bHoofd1 ; "  " ; bHoofd2

End



SUB Test(bSub1 as byte , BYVAL bSUb2 as byte)
  Print bSub1 ; "  " ; bSub2
  bSub1 = 10
  bSub2 = 15
  Print bSub1 ; "  " ; bSub2
End SUB
```

Wanneer je dit programma uitvoert, dan genereert het volgende output:

```
65  5

10  15

10  5

65  5

10  15

10  5
```

Zoals je wellicht opmerkt, wordt de verandering van bSub2 niet gezien in het hoofdprogramma. Variabele bHoofd2 blijft dus ongewijzigd. Wat variabele bHoofd1 echter betreft, de wijziging van bSub1 wordt hier wel doorgegeven aan het hoofdprogramma. In het geval van de eerste parameter geven we aan de subroutine door waar deze variabele in het geheugen staat. Zowel hoofdprogramma als subprogramma werken nu met dezelfde geheugenplaats. We werken met eenzelfde referentie. Dit wil zeggen

dat wanneer de subroutine de waarde van deze variabele wijzigt, deze wijziging ook in het hoofdprogramma zichtbaar zal zijn. We noemen dit een ‘call by reference’ (BYREF).

In het geval van de tweede parameter wordt er geen referentie doorgegeven naar de subroutine, maar enkel de waarde van de variabele op dat moment. Omdat de subroutine niet weet waar het hoofdprogramma de variabele bewaart, is het onmogelijk om de wijzigingen die de subroutine aan deze variabele doet door te geven naar het hoofdprogramma. Het doorgeven van de waarde van een variabele noemen we een ‘call by value’ (BYVAL).

Indien je een ‘call by value’ wil doen moet je dit expliciet opgeven, anders beschouwt BASCOM de call als ‘by reference’. Merk tevens op dat het doorgeven van een constante steeds ‘by reference’ moet gebeuren.

Het is tevens mogelijk om een variabele in de subroutine te declareren met behulp van het statement LOCAL. Op dat moment is deze variabele enkel in de subroutine gekend, ‘lokaal’ dus.

2.3.3. CALL – Function.

We hebben zojuist gezien dat een call van een subroutine een waarde van een variabele kan veranderen. Een functie geeft echter direct een waarde terug. Het gebruik is vergelijkbaar met de subroutines, het verschil zit hem hoofdzakelijk in de declaratie.

DECLARE FUNCTION TEST[([BYREF/BYVAL] var as type)] As type

test	Naam van de functie.
Var	Naam van de variabele(n).
Type	Datatype van de variabele(n) en van het resultaat. Byte,Word/Integer, Long of String.

- Wanneer BYREF of BYVAL niet zijn opgegeven, wordt de parameter 'by reference' doorgegeven.
- Gebruik BYREF om de variabele 'by reference' door te geven met zijn adres.
- Gebruik BYVAL om een kopie van de variabele door te geven.
- Je moet een functie eerst declareren alvorens je de functie kan schrijven of aanroepen.

OPGAVE

Bestudeer als voorbeeld van een "declare function" het programma in *VB25.BAS*.

OPGAVE

Maak een programma dat de reeks van Fibonacci genereert. De microcontroller vraagt eerst om het eindgetal in te geven. Daarna genereert de microcontroller de reeks tot het Fibonacci-getal dat kleiner of gelijk is aan het eindgetal.

Voor elk getal uit de reeks van Fibonacci geldt dat $F_n = F_{n-1} + F_{n-2}$ met $F_0 = 0$ en $F_1 = 1$. Hiermee bekomt men de volgende rij van getallen

0,1,1,2,3,5,8,13,21,34,55,….

waarbij elk getal in de rij de som is van de twee vorige.

3. INTERFACING

3.1. LCD-Display.

BASCOM maakt het ons wel heel gemakkelijk wanneer we met een standaard LCD willen werken. In BASCOM zitten namelijk alle mogelijke voorzieningen om een aangesloten LCD aan te sturen. Eenmaal we BASCOM correct hebben geconfigureerd kunnen we gebruik maken van standaard commando's om tekst op de LCD te plaatsen. Zelfs het creëren van eigen karakters behoort tot de mogelijkheden. Hoe gaan we te werk?

3.1.1. Aansluiten van de LCD aan de microcontroller.

Het aansluiten van een LCD kan op twee verschillende manieren gebeuren.

- Door de pinnen van de LCD te verbinden met de pinnen van de microcontroller. Dit noemen we de "pin-mode". Het voordeel hiervan is dat je zelf de gebruikte pinnen vrij kan kiezen. Ze moeten niet tot dezelfde poort behoren of elkaar opvolgen. Je kan dus de pinnen nemen die voor je printontwerp het eenvoudigste resultaat geven. Het nadeel ervan is dat deze manier van werken meer code vraagt.
- Door de datapinnen van de LCD te verbinden met de databus. Dit noemen we de "bus-mode". Dit is de meest aangewezen manier van werken indien je over externe RAM beschikt. Op dat moment is er immers een databus en adresbus aanwezig in je ontwerp. Het gebruik van deze manier van werken resulteert in een ingewikkeldere print. Het voordeel is wel dat er minder code nodig is.

In de print die bij deze cursus hoort hebben we voor de "pin-mode" gekozen.

De LCD-display wordt aangestuurd in 4-bit mode. De reden hiervoor is de volgende. De mensen bij MCS-electronics hebben een manier uitgedacht om de grootte van de code die wordt gegenereed te beperken. Hiervoor maken zij gebruik van een speciale LCD-bibliotheek. De enigste vereiste bij deze manier van werken is dat de keuze van de gebruikte processorpinnen niet meer vrij is. Je bent verplicht van de pinnen te gebruiken die zij hebben ingebakken in hun bibliotheek. Op de door ons gebruikte print is de LCD aangesloten volgens deze configuratie. Daardoor kunnen wij de code die wordt gegenereerd beperken.

De aansluiting van onze LCD is als volgt gebeurt:

LCD DISPLAY	PORT	PIN
DB7	PORTB.7	14
DB6	PORTB.6	13
DB5	PORTB.5	12
DB4	PORTB.4	11
E	PORTB.2	6
RS	PORTB.0	4
R/W	Massa	5
Vss	Massa	1
Vdd	+5 Volt	2
Vo (contrastregeling)	0-5 Volt	3

Indien je een andere configuratie kiest moet je deze instellen in BASCOM. Deze instellingen kan je terugvinden in OPTIONS --> COMPILER --> LCD. Ook het type van de gebruikte LCD moet je hier ingeven. Wij werken met een 16*2. Dit wil zeggen dat onze display twee regels heeft, met op elke regel plaats voor 16 karakters. Het is ook mogelijk om je configuratie in de code van je programma zelf op te nemen. Daarvoor zijn CONFIG LCD, CONFIG LCDBUS, CONFIG LCDMODE en CONFIG LCDPIN voorzien. Hoe je hiervan gebruik moet maken kan je in het help-bestand van BASCOM terugvinden (telkens mét een voorbeeld).

Hoe we de speciaal ontwikkelde LCD-bibliotheek moeten gebruiken in onze programma's kan je terugvinden in *LCD01.BAS*.

```
$lib "lcd4.lbx"

Cls
Lcd "BASCOM-AVR"
Lowerline
Lcd "Tweede lijn"

End
```

Het oproepen van de bibliotheek gebeurt op de eerste lijn van het programmavoorbeeld. Deze bibliotheek is tevens terug te vinden op de CD die bij deze cursus hoort.

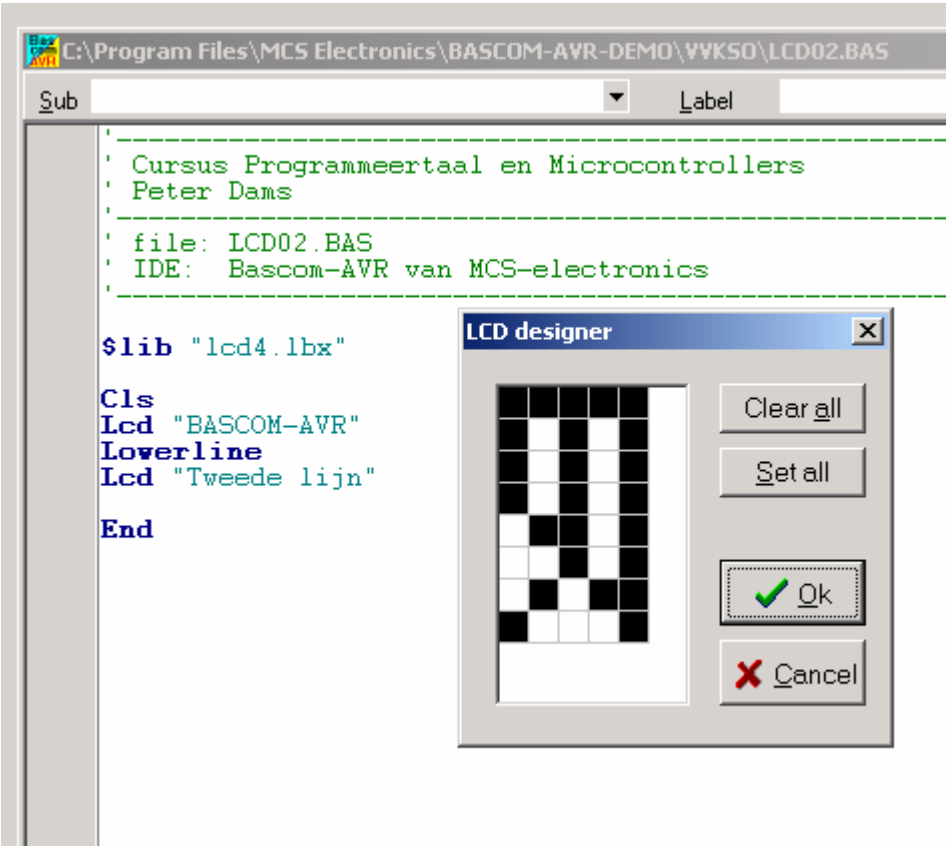
Op de tweede lijn gaan we de display leeg maken en plaatsen de cursor op de plaats voor het eerste karakter op de eerste lijn. Dit gebeurt met een "clear screen" of CLS. Indien je naar het eerste karakter op de eerste lijn wil gaan zonder de display leeg te maken, gebruik je het HOME-statement. Daarna sturen we tekst naar de display met het 'LCD'-statement. De laatste lijn op onze display, in ons geval de tweede, selecteren we met 'LowerLine'.

OPGAVE

Bestudeer het gebruik van $SERIALINPUT2LCD, SHIFTLCD en DISPLAY ON/OFF

3.1.2. Zelf karakters maken.

Ook dit wordt ons in BASCOM erg makkelijk gemaakt. We kunnen acht zelf getekende karakters maken en gebruiken. Voor het genereren van de juiste code maken we gebruik van de 'LCD-designer' die we terugvinden in het TOOLS-menu. We gaan als volgt te werk. Vertrek van de code uit het voorbeeld *LCD01.BAS*. Maak plaats voor een lijn code in te voegen vóór het statement CLS. Roep nu de LCD-designer uit het TOOLS-menu op.



Elk vakje stelt een puntje op onze display voor. Door op een vakje te klikken kan je het wisselen tussen ‘zwart’ of ‘wit’. Teken op deze manier het karakter dat je wenst. Indien je je creatie af hebt klik je op ‘OK’. De LCD-designer voegt automatisch de benodigde code in je programma in.

Het enige dat je nog moet wijzigen is het rode vraagteken in deze lijn. Dit moet je vervangen door een cijfer van 0 t.e.m. 7. Dit benoemt het karakter als één van de acht zelf te maken karakters. Nu kan je het zelf getekende karakter naar de display sturen. Dit doe je door gebruik te maken van het statement CHR().

Een voorbeeld kan je vinden in *LCD02.BAS*.

```
$lib "lcd4.lbx"

Deflcdchar 0 , 31 , 21 , 21 , 21 , 13 , 5 , 11 , 17
Cls
Lcd "BASCOM-AVR"
Lowerline
Lcd "Tweede lijn ";chr(0)

End
```

3.2. Polling – Interrupts.

Het is meestal de bedoeling dat ons programma bepaalde acties gaat ondernemen bij een verandering van één of andere ingang. De toestandsveranderingen aan de ingangen van onze microcontroller kunnen we op twee verschillende manieren gaan ondervangen.

Een eerste mogelijkheid is dat wij zelf in ons programma regelmatig de toestand van een ingang gaan opvragen en dat we dan kijken of er veranderingen zijn opgetreden. Deze manier van werken waarbij we de verschillende ingangen één voor één gaan afvragen noemen we "polling". Deze manier van werken is zeer eenvoudig te programmeren maar heeft een belangrijk nadeel. Namelijk we weten pas dat een ingang van toestand is veranderd op het moment dat wij er naar gaan kijken. Wanneer we met een uitgebreid programma te maken hebben kan de tijd tussen twee opeenvolgende vergelijkingen wel eens hoog oplopen. In gevallen waar snel moet gereageerd worden op een wijziging op een bepaalde ingang kan dit problemen opleveren. Om dit op te lossen maakt men in de computerwereld gebruik van zogenaamde "interrupts". Dit is een hardwarematige manier om een programma te onderbreken waar het op dat moment mee bezig is en het te dwingen eerst de noodzakelijke handelingen te ondernemen.

De controller die wij gebruiken (AT90S2313) heeft twee van dergelijke ingangen (INT0 en INT1). We spreken in zulk geval van externe interrupts. Daaruit kan je terecht besluiten dat er ook nog interne interrrupts zijn, maar daarover later meer. Wanneer er een toestandsverandering aan één van deze ingangen plaatsvindt zal de microcontroller een speciale subroutine uitvoeren die bij de interrupt hoort die er is opgetreden. Zulk een subroutine noemen we een interrupt-service-routine.

Om echter gebruik te kunnen maken van de interrupt mogelijkheden van een microcontroller moet je een aantal instellingen doen en moet je exact weten hoe de microcontroller reageert op een interrupt. BASCOM helpt ons echter hierbij zodat het werken met interrupts aanzienlijk wordt vereenvoudigd. Voor diegenen die graag dieper ingaan op deze materie verwijs ik naar de datasheet van ATMEL.

Welke acties moeten we nu in BASCOM ondernemen om met interrupts te werken?

1. We moeten de interrupts die we gebruiken activeren.
2. We moeten de interrupts die we hebben geactiveerd configureren
3. We moeten zeggen waar de interrupt-service-routine staat.
4. We moeten de interrupt-service-routine schrijven.

Laten we deze stappen één voor één toelichten.

Standaard is er bij onze microcontroller geen enkele interrupt geactiveerd. Dat wil dus zeggen dat je de interrupts die je wenst te gebruiken zelf zal moeten activeren. De interrupts kunnen we activeren door het ENABLE commando.

Vooreerst moet je kenbaar maken dat je de interrupts wil gebruiken. Dit gebeurt door:

ENABLE INTERRUPTS

Vanaf nu kan je individuele interrupts inschakelen of terug uitschakelen. Het inschakelen gebeurt als volgt:

ENABLE INT0

ENABLE INT1

Het terug uitschakelen van interrupts doe je als volgt:

DISABLE INT0

DISABLE INT1

Eenmaal een interrupt is geactiveerd kunnen we nog instellen hoe hij juist moet reageren. Dit gebeurt als volgt:

CONFIG INT0 = toestand

De toestand kan op drie verschillende manieren worden ingesteld.

LOW LEVEL	gaat een interrupt genereren als de ingang laag is. Wanneer de pin laag blijft, dan wordt er continu een interrupt gegenereerd
FALLING	genereert een interrupt om de dalende flank
RISING	genereert een interrupt om de stijgende flank

Om aan te duiden waar de microcontroller zijn interrupt-service-routine kan vinden, gebruiken we het "On interrupt" commando. Aangezien we twee externe interrupts hebben zijn er twee varianten, namelijk:

ON INT0 label

ON INT1 label

Het label bevat de naam van de serviceroutine. Bv ON INT1 Int1_int.

Ten slotte moet je de interrupt-service-routine nog schrijven. Dit doe je op dezelfde manier als je zou doen bij een gewone subroutine. Bv:

```
Int1_int:
  Incr iTeller
  Print "Int 1 occured" ; iTeller
Return
```

Je moet er wel met het volgende rekening houden. Wanneer een interrupt voorkomt dan zal de controller naar de service-routine springen. Bij het springen naar deze routine worden alle interrupts automatisch gedisabled. Verder zal de controller ook alle registers tijdelijk bewaren (op stack). Bij de eerste RETURN die wordt tegengekomen buiten een conditie worden alle registers terug in hun oorspronkelijke staat hersteld (van stack gehaald). Vervolgens wordt er teruggesprongen naar dat deel in het programma waar de controller mee bezig was op het moment dat de interrupt voorkwam. Een return die een interrupt-service-routine beëindigt noemen we een RETI (return from interrupt). Na een RETI worden de interrupts terug geactiveerd.

Je kan bij deze controller geen prioriteiten toekennen aan interrupts.

Je kan het opslaan van de registerinhouden op stack ook uitschakelen. Dit doe je door bij het on-interrrupt commando het nosave argument mee tegeven. Bv:

ON INT1 NaamServiceRoutine **NOSAVE**

OPGAVE: Tik het programma INT1.BAS in en test het uit.

```
Dim iTeller As Integer , bIngave As
Byte

Enable Interrupts
Enable Int1
Config INT1 = FALLING
On Int1 Int1_int

iTeller = 0
Print "Start"
Do
   Input "Geef een toets in" ,
bIngave
Loop Until bIngave = 27
End

Int1_int:
  Incr iTeller
  Print "Int 1 occured" ; iTeller
  waitms 500
  Gifr = 128
Return
```

Nog kort een woordje uitleg over de interrupt-service-routine in dit voorbeeldprogramma. In deze routine zie je een wachttijd van 500ms en een Gifr=128 staan. Dit is gedaan om op een softwarematige manier de contactdender (bouncing) van de schakelaar op te vangen. De wachtlus spreekt nu voor zich. We wachten gewoon tot de contactdender voorbij is. Ondertussen is echter wel door de controller opnieuw een interrupt gedetecteerd. Dit wordt bijgehouden in het "General Interrupt Flag Register" of kortweg "GIFR". We kunnen deze detectie ongedaan maken door naar de bitplaats die overeenkomt met de gewenste interrupt een "1" te schrijven (zie de uitleg bij GIFR in de ATMEL-datasheet). Aangezien de hoogste bit overeenkomt met INT1 schrijven we 128 weg naar het GIFR.

Deze twee lijnen code hebben enkel nut indien je het programma uitvoert op hardwareniveau. In de simulator zal je hiervan uiteraard niets merken.

OPGAVE:

Bestudeer de uitleg over interrupts die je kan terugvinden in de handleiding van BASCOM. Zoek in de datasheet van de AT90S2313 alle registers op die te maken hebben met de externe interrupts.

OPGAVE:

Diegenen die een print hebben kunnen aan een externe interrupt (bv. Int0) een blokgolf van 1Hz leggen. Schrijf nu een programma wat van je LCD-display een klok maakt. Bij het opstarten geeft je in je terminal (via RS-232) de starttijd in.

PROJECT

Je kan een DCF ontvanger gebruiken om aan de 1Hz pulsen te komen. Op dat moment heb je een klok met een precisie van een atoomklok. In een tweede fase kan je de codering van het DCF-signaal gebruiken om je klok automatisch juist te zetten! (tip: Conrad verkoopt kleine en goedkope DCF-ontvangers met een seriële uitgang die je dan aan je interruptingang koppelt). Met wat je op dit moment hebt geleerd zou je dit project moeten kunnen maken.

4. I²C.

Philips introduceerde begin jaren tachtig de 'Inter Integrated Circuit'-bus of kortweg de I²C-bus. De bedoeling was om in hun consumerproducten (televisies, videorecorders,…) de verbinding tussen de microcontroller en de rand-IC's te vereenvoudigen. In plaats van te werken met de klassieke busstructuren (databus, adresbus en controlebus) was het de bedoeling om de verbinding met zo weinig mogelijk "draden" te laten verlopen. De I²C-bus laat de communicatie verlopen over slechts twee draden. Philips heeft in de eerste jaren een groot aantal toepassings-IC's ontworpen zodat ook andere fabrikanten van dergelijke consumerproducten deze IC's gingen gebruiken. Op dit moment is de I²C-bus, algemeen gebruikt in een groot aantal verschillende toepassingen. Indien u graag meer te weten komt over de werking van deze bus dan verwijs ik u door naar de specificaties van de IC-bus. Deze kan u uiteraard terugvinden op de website van Philips.

Als uitgangspagina kan u best http://www.semiconductors.philips.com/buses/index.html gebruiken. Het document dat u moet hebben is: I2C_BUS_SPECIFICATION_3.pdf. Je kan uiteraard ook via de website www.elcom.be gaan. Hier zijn de nodige links naar alle noodzakelijke datasheets steeds terug te vinden. Dit document is ook terug te vinden op de CD-ROM die bij het controllerbordje hoort.

Hoe zit het nu met BASCOM en I²C? Welnu, het I²C-protocol zit als het ware in BASCOM ingebakken. Best zorg je ervoor dat je versie 1_11_6_8 van BASCOM-AVR hebt, of een recentere versie. Vanaf deze versie is het volledige I²C-gedeelte immers herschreven. De directives die je kan gebruiken in BASCOM zijn:

```
Config Sda = Portd.6
Config Scl = Portd.5
Config I2cdelay = 255
```

De instructies die te maken hebben met I²C zijn:

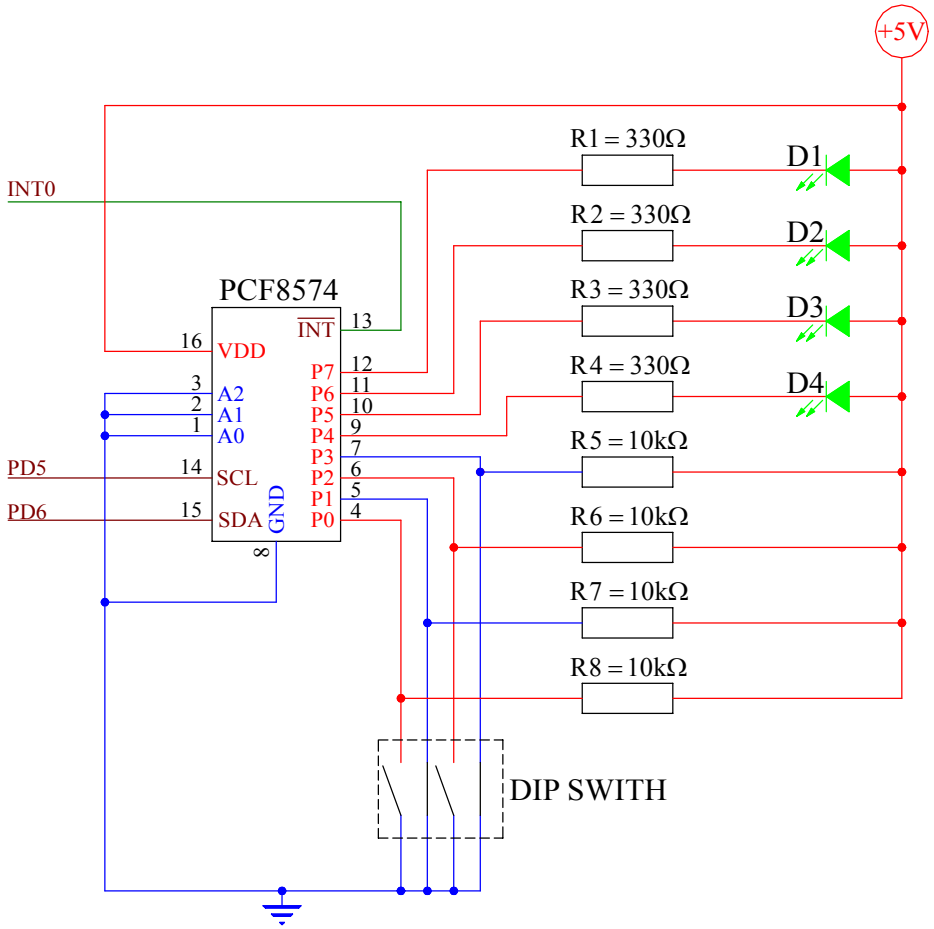
I2START : deze genereert een I²C start conditie.
I2CSTOP: deze genereert een I²C stop conditie.
I2CRBYTE: ontvangt een byte van een I²C-onderdeel.
I2CWBYTE: zend een byte naar een I²C-onderdeel.

Syntax:
I2CSTART
I2CSTOP
I2CRBYTE var, ack/nack
I2CWBYTE val

Opmerkingen

Var	Een variabele die de ontvangen informatie bevat.
ack/nack	Stuurt een ACK als er meerdere bytes moeten worden gelezen Stuurt een NACK als de laatste byte is gelezen.
Val	Dit kan een variablel of een constante zijn die moet weggeschreven worden.

Voor alle voorgaande programma's die we tot nu toenmaakten konden we beroep doen op de simulator om te controleren of ons programma wel degelijk werkte zoals wij dat hoopten. Voor het hoofdstuk I²C is dit echter niet mogelijk. Daarom gebruiken we best ons hardware-bordje. Maak de opstelling volgens onderstaand schema en plaats op het processorbordje jumpers JP1 en JP2 om de pull-up weerstanden voor de I²C-bus in te schakelen.



Een kort woordje over het schema. We maken voor deze oefening gebruik van de PCF8574, een 8-bit I/O expander van Philips. Met dit IC kunnen we via I²C een poort met acht pinnen gebruiken. Deze vormen een quasi-bidirectionele poort, wat betekent dat de pinnen zowel als ingang of als uitgang te configureren zijn. De uitgangen van de poort kunnen een stroom van 25mA sinken. Dit wil zeggen dat ze een stroom van 25mA kunnen leveren indien het uitgangsniveau van de poort een logische nul is.

Deze eigenschap laat ons toe om direct LED's aan te sturen zonder het gebruik van buffers. De PCF8574 is ook uitgerust met een interruptuitgang. Omdat dit IC niet als master kan fungeren in een I²C-bus maakt men gebruik van de interrupttechniek om toe te laten dat dit IC onmiddellijk een verandering aan één van zijn ingangen kan melden aan de microcontroller. Wanneer dus één van de ingangen verandert, zal de PCF8574 een logische nul op zijn interruptuitgang zetten. In de microcontroller moet een programma deze interrupt dan verwerken. Wanneer de controller dan van de PCF8574 gaat lezen of ernaar gaat schrijven, zal de interrupt automatisch gereset worden. Voor de datasheet van dit IC kan je terecht op de site van Philips of op de site van Elcom of op de CD-ROM die bij het controllerbordje hoort.

De weerstanden in serie met de LED's hebben als doel de stroom door de LED's te beperken.

Geef nu het onderstaande programma in, of laad voorbeeld *I²C01.BAS*.

```
Config Sda = Portd.6
Config Scl = Portd.5
Config I2cdelay = 5

Const Badreswrite = &H40
Const Badresread = &H41

Dim Bpcfdata As Byte

Do
  I2cstart
  I2cwbyte Badresread
  I2crbyte Bpcfdata , Ack
  I2crbyte Bpcfdata , Nack
  I2cstop

  Bpcfdata = Bpcfdata And 15
  Shift Bpcfdata , Left , 4
  Bpcfdata = Bpcfdata Or 15

  I2cstart
  I2cwbyte Badreswrite
  I2cwbyte Bpcfdata
  I2cstop
Loop

End
```

Even het programma overlopen. Vooreerst configureren we BASCOM met de noodzakelijke parameters om de I²C-communicatie correct te kunnen initiëren. Dit kan via de menukeuze **OPTIONS → COMPILER → I2C** of direct in de programmacode. In dit voorbeeld is voor het laatste gekozen. De twee eerste lijnen dienen om BASCOM te melden welke pin van de microcontroller willen gebruiken om te

gebruiken als klok (SCL) en welke als data (SDA). Vervolgens kunnen we, met behulp van de delay, instellen met welke snelheid de communicatie gaat verlopen. BASCOM gebruikt dit getal om aan de hand van het gebruikte kristal de snelheid te bepalen. Het gebruikte bordje is uitgerust met een 4MHz kristal en met een I2Cdelay van 5 resulteerde dit in een kloksnelheid van SCL van ongeveer 56kHz. (volgens MCS zou dit rond de 100kHz moeten liggen en onafhankelijk zijn van het gebruikt kristal, hieraan wordt gewerkt). Dit is dus nog ruimschoots beneden de maximale kloksnelheid van 100kHz van de PCF8574.

Vervolgens worden er twee constanten gedeclareerd. Het zijn de adressen waarmee de PCF8574 te bereiken is. Het I²C adres van een chip wordt meestal bepaald door drie parameters. Vooreerst een vast gedeelte dat in principe voor elk soort chip anders is. In dit geval is dat 0100. Vervolgens volgt een door de gebruiker in te stellen deel. Dit wordt meestal hardwarematig ingesteld. Als je het schema bekijkt dan zie je in de voorstelling van de PCF8574 de adreslijnen A0…A2. In dit geval zijn ze met massa verbonden en dus alle drie 0. Deze manier van werken laat ons toe om in één I²C-systeem acht verschillende PCF8574 te adresseren. De waarden van de drie adresingangen komen achter het vast gedeelte. We bekomen nu 0100 000. De laatste bit van het adres bepaald of we gaan schrijven naar de chip (0) ofwel of we gaan lezen van de chip (1). Dit geeft ons dus twee verschillende adressen, één om te schrijven 0100 0000 (= 40 hexadecimaal) en één om te lezen 0100 0001 (= 41 hexadecimaal).

Vervolgens begint de code voor de communicatie. Eerst wordt de communicatie gestart. Dan gaan we het adres schrijven van de chip waarmee we willen werken. Dan gaan we lezen van de chip. In de datasheet zien we dat pas na het tweede leescommando de juiste informatie zullen lezen. De eerste keer sluiten we af met een ACK, waarmee we te kennen geven dat er nog een leescommando zal volgen. De tweede keer dat we gaan lezen sluiten we af met een NACK, waarmee we te kennen geven dat er geen leescommando's meer volgen. Vervolgens wordt de communicatie gestopt. In de volgende drie lijnen berekenen we de info die we gaan schrijven naar de PCF8574. De bedoeling is om de informatie die we van de vier schakelaars gaan lezen, gaan uitsturen naar de vier LED's. Als oefening kan je uitzoeken wat er juist gebeurt in deze drie lijnen.

In het laatste deel starten we terug de communicatie, sturen het adres van de chip waarmee we willen werken. We schrijven vervolgens de informatie er naar toe en sluiten de communicatie af. Nadien begint alles opnieuw. Dit gaat eindeloos door dank zij de do-loop zonder voorwaarden.

OPGAVE:

Programmeer de controller en controleer de werking van het programma. Om het elke keer terug te starten druk je op de RESET-knop van het bordje.

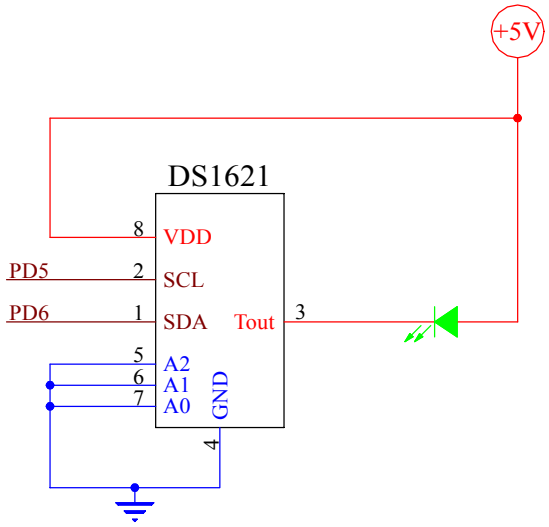
OPGAVE:

Herschrijf het programma zodanig dat je gebruik maakt van de interruptmogelijkheden van de PCF8574.

Er zijn momenteel zeer veel verschillende IC's te verkrijgen met I²C-mogelijkheden. Het loont zeker de moeite om eens op zoek te gaan naar dit soort chips. Hiervoor kan je onder andere terecht op de sites van Philips en Dallas Semiconductor. Om de mogelijkheden van dergelijke IC's te onderstrepen maken we nog een opstelling met een ander IC. Deze keer gebruiken we een DS1621 van Dallas Semiconductor. Het is een chip die de temperatuur meet, die regelmatig kan loggen en eventueel ook een uitgang sturen. We kunnen dus zeggen dat deze 8-pin IC een thermostaat is.

De datasheet van dit IC kan je downloaden van de site van Dallas Semiconductor. Het is interessant om voldoende tijd uit te trekken om de datasheet grondig door te lezen. Het geeft je een goed inzicht in de werking van het component en in zijn vele mogelijkheden.

Bouw onderstaand schema op en plaats op het processorbordje jumpers JP1 en JP2 om de pull-up weerstanden voor de I²C-bus in te schakelen.



De meeste pinnen van het IC zullen ondertussen al wel gekend zijn. SDA en SCL zijn de I²C-lijnen. A0,A1 en A2 vormen het deel van het I²C-adres van het IC dat door de gebruiker kan worden ingesteld (hier dus op 000). We herkenen ook de voeding (+5V en 0V). Als laatste lijn merken we Tout op. Dit is de "thermostaatslijn", waarover later meer. Als voorbeeldprogramma geef je onderstaand programma in of laad je *I²C02.BAS*.

```
'Begin configuraties en declaraties
Config Sda = Portd.6
Config Scl = Portd.5
Config I2cdelay = 5

Const Badreswrite = &H90
Const Badresread = &H91
Const Breadtempcommand = &HAA
Const Bstartconvert = &HEE
Const Bstopconvert = &H22
Const Bconfig = &HAC
Const Bcontinuousmode = 8

Dim Bmsb As Byte
Dim Blsb As Byte
Dim Btemp As Byte

$lib "lcd4.lbx"

'Begin programma
  Cls
  Upperline
  Lcd "temperatuur"

  I2cstart
  I2cwbyte Badreswrite
  I2cwbyte Bconfig
  I2cwbyte Bcontinuousmode
  I2cstop

  I2cstart
  I2cwbyte Badreswrite
  I2cwbyte Bstartconvert
  I2cstop

  Do
    I2cstart
    I2cwbyte Badreswrite
    I2cwbyte Breadtempcommand
    I2cstart
    I2cwbyte Badresread
    I2crbyte Bmsb , Ack
    I2crbyte Blsb , Nack
    I2cstop

    If Btemp <> Bmsb Then
      Lowerline
      Lcd Bmsb
      Lcd " graden"
      Btemp = Bmsb
     End If
  Loop
End
```

We overlopen weer de opbouw van het voorbeeldprogramma. Eerst configureren we de I²C-parameters die de compiler nodig heeft. Ze zijn in vorig voorbeeld reeds verklaard. Daarna volgt de declaratie van de constanten. Het vaste adresdeel van de DS1621 is 1001. Het instelbare hebben we op 000 gezet en de laatste adresbit bepaald of we lezen of schrijven. Dit geeft samen 90 hexadecimaal voor het schrijven naar de DS1621 en 91 hexadecimaal voor het lezen van de DS1621. Het uitlezen van de temperatuur start je door het commando AA hexadecimaal naar de chip te sturen. De temperatuur wordt pas omgezet in de DS1621 nadat je "Start Conversion"-commando is gegeven. Dit is EE hexadecimaal. Het stoppen van de conversie gebeurt door het commando 22 hexadecimaal te sturen. De temperatuursconversie kan op twee manier verlopen. Ofwel converteert het IC één keer (one shot) en wacht dan terug op een start-commando, ofwel converteert hij continu en stop hij pas na het stopcommando. De keuze tussen continu of one-shot kan je maken in het configuration register (AC hexadecimaal). Wij kiezen voor continue conversie. We moeten dus de laagste bit van het configuratieregister nul maken. Dit doen we door de waarde 8 naar het configuratieregister te sturen. Vervolgens dimensioneren we de variabelen. We voorzien twee byte-variabelen om de uitgelezen temperatuur in te bewaren (MSB en LSB) en een temp variabele (later meer hierover). Aangezien we gebruik maken van het LCD-scherm declareren we ook de display-bibliotheek.

Het programma start met het wissen van de LCD met het CLS-commando. Vervolgens plaatsen we de tekst "temperatuur" op de eerste lijn van de display. Daarna gaan we de DS1621 instellen in continu mode. We geven een I²C-start conditie en sturen het schrijfadres op de I²C-bus. Vervolgens het commando om het configuratieregister te schrijven, gevolgd door de waarde 8. Als tweede stap gaan we de temperatuursconversie starten. Dit gebeurt door weer een I²C-start conditie te geven, het schrijfadres te sturen en het commando om de conversie te starten (EE hexadecimaal). Vanaf nu zal de DS1621 continu zijn temperatuur uitlezen en omzetten in een binair getal.

De rest van het programma gaan we weer in een eindeloze lus uitvoeren (een do-loop zonder voorwaarden). Het uitlezen van de temperatuur moet volgens volgend patroon verlopen (zie datasheet pg 9). Eerst genereer je een startconditie. Vervolgens geef je het schrijfadres van de DS1621 en stuur je een "Read Temperature" commando. Daarna geef je terug een start en stuur je het leesadres van de DS1621 op de bus. Vervolgens lees je de temperatuur in twee bytes uit. Eerst de MSB (most significant byte) en nadien de LSB (least significant byte). Afsluiten doe je met een stopconditie. In de If-then lus gaan we de recent gelezen temperatuur vergelijken met de vorig gelezen temperatuur die we in de temp-variabele bewaren. We voeren deze lus alleen uit als de temperatuur verandert is. Dit geeft een stabielere uitlezing van de display. Als we dit niet doen, dan gaan we dikwijls naar dezelfde waarde naar de display sturen, wat voor een flikkering kan zorgen. Op de onderste lijn van onze display plaatsen we de gelezen temperatuur gevolgd door het woordje "graden". Tenslotte bewaren we de nieuw gelezen temperatuur in de

variabele bTemp om deze nieuwe waarde te kunnen gebruiken voor de volgende vergelijking in de if-then.

Je kan controleren of je opstelling werkt door je vinger op de chip te plaatsen om alzo de temperatuur te verhogen.

OPGAVE

De DS1621 meet tot op een halve graad nauwkeurig. Deze halve graad wordt doorgegeven in de LSB. Pas je programma aan zodat de temperatuur met een nauwkeurigheid van een halve graad op de display komt.

OPGAVE

Je kan de DS1621 ook gebruiken als thermostaat. Je kan de DS1621 zo programmeren dat de Tout uitgang laag wordt als de temperatuur onder een bepaalde grens zakt. Zoek op in de datasheet hoe je de DS1621 moet instellen en schrijf vervolgens een programma dat van deze eigenschap gebruik maakt.

OPGAVE

De DS1621 heeft nog meer mogelijkheden. Zoek uit welke en test ze uit door er telkens een aangepast programma voor te schrijven.

OPGAVE

Tegenwoordig wordt I²C veel toegepast in TV's en videorecorders. Als je een tamelijk recent model binnen handbereik hebt dat (deels) defect is, dan kan je ons controllerbordje verbinden met de I²C-bus van dat toestel. Als je aan een schema van dat toestel kan geraken kan je proberen dit toestel te besturen vanuit het controllerbordje. Plezier gegarandeerd!!

5. RS-232.

BASCOM laat ons toe om op eenvoudige wijze informatie tussen de PC en ons controllerbordje uit te wisselen. Hiervoor maken we gebruik van één van de standaard poorten op de computer namelijk de seriële poort, ook wel COM-poort genoemd. De signalen die de seriële poort van de PC uitstuurt voldoen aan de RS-232 norm. Dit wil onder andere zeggen dat een logische één wordt voorgesteld door een negatieve spanning (op onze PC: -12V) en een logische nul wordt voorgesteld door een positieve spanning (op onze PC: +12V). Om de klassieke 0V/+5V signalen om te zetten naar RS-232 signalen kan je gebruik maken van een standaard interface IC, de MAX232. Dit IC wordt ook gebruikt op ons microcontrollerbordje.

Het enige wat je nog moet doen is een verbindingskabel maken tussen de PC en het bordje. Meestal wordt tegenwoordig voor de seriële poort van je PC een 9-pin sub-D connector gebruikt, net zoals op ons microcontrollerbordje. Aangezien de pin voor de uitgaande signalen en de pin voor de inkomende signalen dezelfde zijn op de PC en op het controllerbordje, moeten we de verbinding tussen deze pinnen kruisen. Zo wordt de "zendpin" (TXD) van de ene kant verbonden met de "ontvangstpin" (RXD) van de andere kant. Verder moeten we nog de massa (GND) van de PC verbinden met de massa van het controllerbordje. Dergelijke kabel noemen we een "null-modem". Indien je gebruik maakt van het controllerbordje dan zijn de te verbinden pinnen:

Pin 2 → Pin 3

Pin 3 → Pin 2

Pin 5 → Pin 5

Indien je PC voorzien is van een 25-polige sub-D dan maak je volgende verbindingen:

9-polig→ 25-polig

Pin 2 → Pin 2

Pin 3 → Pin 3

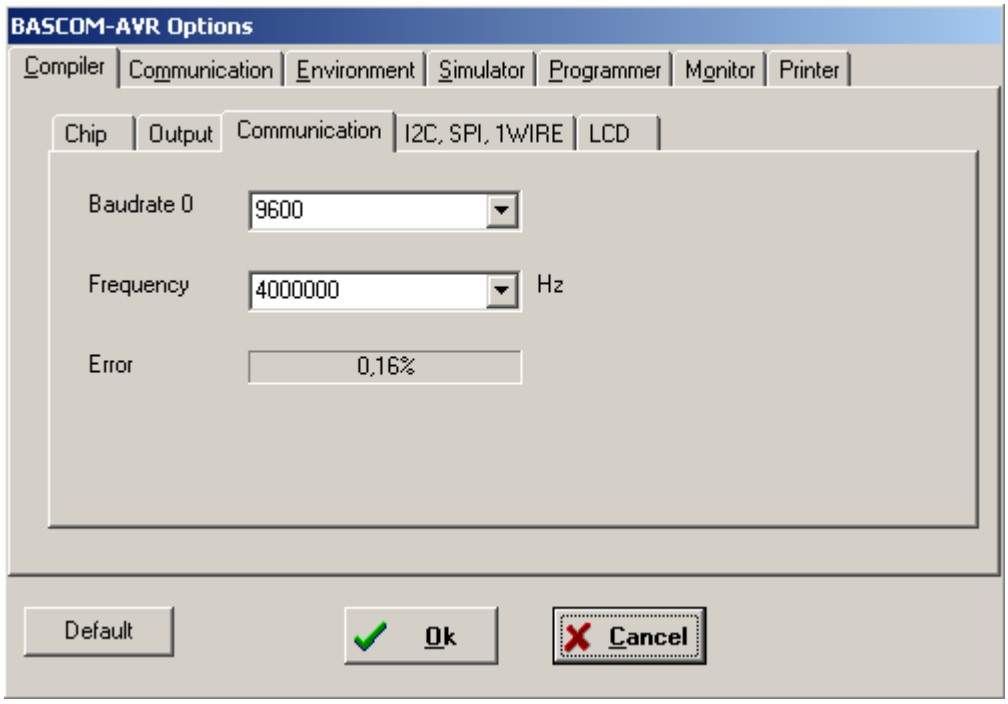
Pin 5 → Pin 7

Tot zover de hardwarekant van de zaak. We kunnen terug naar de softwarekant van de zaak.

In de vorige oefeningen hebben we soms gebruik gemaakt van de commando's "print" en "input". Wel, dit zijn twee belangrijke instructies voor de communicatie met de PC. Met het print-commando ga je een string versturen via de de RS-232 verbinding van je controllerbord naar de PC. Op de PC kan je dit dan zichtbaar maken met een terminalprogramma. Je kan gebruik maken van "Hyperterminal". Dit is een

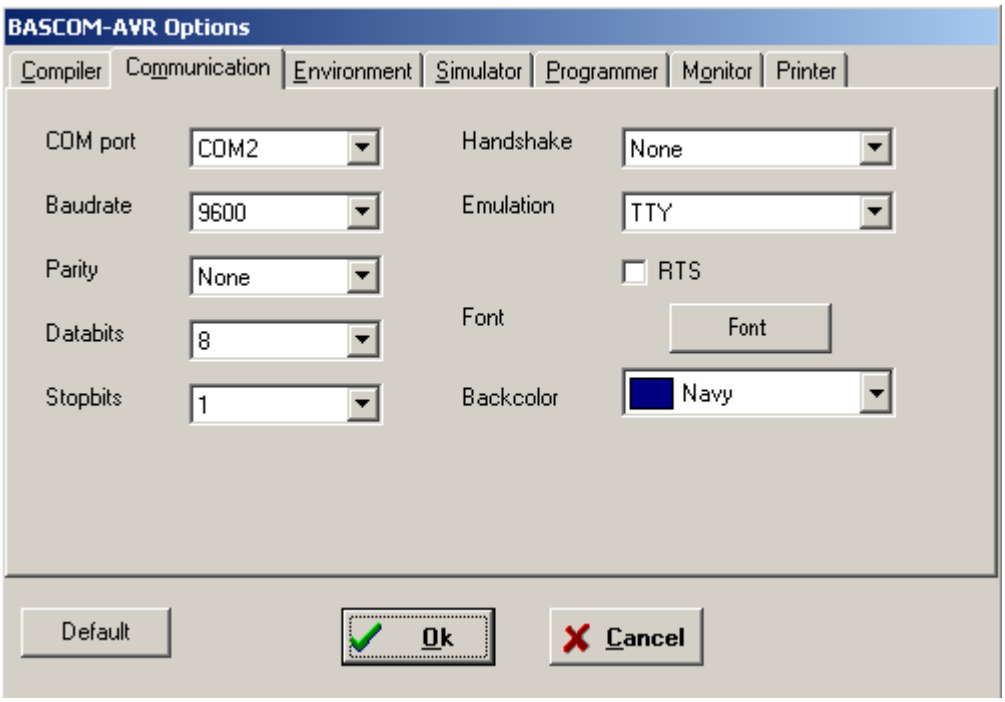
terminalprogramma wat standaard in Windows is voorzien. Wij gaan echter gebruik maken van het terminalprogramma dat bij in BASCOM zit.

Informatie in je controller binnenhalen kan je makkelijk doen met het input-commando. Verder zijn er ook hier weer een aantal instellingen die je moet doen. Vooreerst zijn er een aantal compilerdirectives die je moet instellen. Dit kan in via de menu structuur OPTIONS → COMPILER →COMMUNICATION. Hier moet je de transmissiesnelheid (Baudrate) instellen, en opgeven met welk kristal je werkt. Stel de snelheid in op 9600Bd en ons kristal is een 4MHz kristal.



Als we gebruik willen maken van de ingebouwde terminalemulator van BASCOM dan moeten we die ook instellen. Dit gebeurt eveneens via de menustructuur OPTIONS → COMMUNICATION

Natuurlijk moet de gekozen snelheid voor de microcontroller overeenstemmen met die van de PC. Kies tevens de juiste poort van je PC voor de communicatie.



We zijn nu klaar om een en ander uit te testen.

Geef onderstaand programma in of laad het programma *RS23201.BAS*.

```
'Begin configuraties en declaraties


Dim Bgetal1 As Byte
Dim Bgetal2 As Byte
Dim Bresultaat As Byte



'Begin programma
   Input "Geef het eerste getal: " , Bgetal1
   Input "Geef het tweede getal: " , Bgetal2
   Bresultaat = Bgetal1 + Bgetal2
   Print "De som is: " ; Bresultaat

End
```

De werking van het programma is vrij eenvoudig. Na de declaraties sturen we eerst de tekst “Geef het eerste getal: “ naar het scherm van de computer en wachten tot de gebruiker op de PC een getal ingeeft.

Daarna vragen we op identieke wijze om een tweede getal. Daarna berekent de controller de som van deze twee getallen en stuurt het resultaat naar de computer. Als je dit programma wil uittestten moet je

dus een terminalprogramma op de PC gebruiken. In BASCOM start je de terminalemulatie met de “modem-knop”.

OPGAVE

Geef als eerst getal het getal 200 in en als tweede getal 220. Wat kan je zeggen van het resultaat. Kan je narekenen waarvan deze waarde komt?

OPGAVE

Maak een uitbreiding van bovenstaand programma door na de ingave van de twee getallen te vragen welke bewerking we willen uitvoeren (+, -, x, of /). Test je programma grondig uit.

OPGAVE

Ga op zoek in de help-functie van BASCOM naar de andere commando’s die te maken hebben met onze RS-232 verbinding en probeer hun functie te doorgronden.

We gaan nu een groter programma maken, eigenlijk is het een samenvoeging van de I²C mogelijkheden en van de RS-232 capaciteiten van BASCOM. We gaan terug gebruik maken van de DS1621.

Voer volgend programma in of laad *RS23202.BAS* en test het programma grondig uit. De werking moet je nu kunnen begrijpen.

```
Config Sda = Portd.6
Config Scl = Portd.5
Config I2cdelay = 5

Const Badreswrite = &H90
Const Badresread = &H91
Const Breadtempcommand = &HAA
Const Bstartconvert = &HEE
Const Bstopconvert = &H22
Const Bconfig = &HAC
Const Bcontinuousmode = 8

Dim Bmsb As Byte
Dim Blsb As Byte
Dim Btemp As Byte

$lib "lcd4.lbx"

'Begin programma
  Cls
  Upperline
  Lcd "temperatuur"

  I2cstart
  I2cwbyte Badreswrite
  I2cwbyte Bconfig
  I2cwbyte Bcontinuousmode
  I2cstop

  I2cstart
  I2cwbyte Badreswrite
  I2cwbyte Bstartconvert
  I2cstop

  Do
    I2cstart
    I2cwbyte Badreswrite
    I2cwbyte Breadtempcommand
    I2cstart
    I2cwbyte Badresread
    I2crbyte Bmsb , Ack
    I2crbyte Blsb , Nack
    I2cstop

    If Btemp <> Bmsb Then
      Lowerline
      Lcd Bmsb
      Lcd " graden"
      Print "Temperatuur is " ; Bmsb ; " graden."
      Btemp = Bmsb
    End If
  Loop
End
```

Nog uit te werken:

6 ADC

7 ONE-Wire

8 Timers

9. RC5

Inhouds opgave.