HANDLEIDING LABVIEW

(1e jaar Practicum Natuurkunde 1)

juni 2012

Radboud Universiteit Nijmegen
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
M.A.H. Asselbergs
I.V. Grigorieva
P.F. Klok

Inleiding

Deze handleiding hoort bij het onderderdeel LabVIEW van het Practicum Natuurkunde 1. De eerste hoofdstukken zullen behandeld worden tijdens een college/practicumsessie van een uur. Tijdens een daarop volgende practicumdag zal eerst LabVIEW verder behandeld worden, waarna er de rest van de dag gelegenheid is om de opdrachten voor LabVIEW onder begeleiding te maken.

Deze handleiding bestaat uit:

1. Wat is LabVIEW?
2. LabVIEW opstarten
   Opslaan van programma's
3. LabVIEW programmeeromgeving
   Front Panel en Block Diagram
   Hiërarchie van VI's en sub-VI's
   Front Panel knoppen
   Block Diagram knoppen
   Tools Palette knoppen
   Het Icon pictogram
   Het verloop van een programma
4. Werken met LabVIEW
   In- en uitvoer
5. Opdrachten

Conventies

NB: *LabVIEW termen zijn meestal in het Engels en worden de eerste keer, bij de definitie of de verklaring ervan,* **vet** *weergegeven.*
NB: *Bij uitklapbare menu's wordt de volgorde van selecties* **vet** *weergegeven door de achtereenvolgende namen met een pijl (→) ertussen, bijvoorbeeld:* **Functions→Arith & Compar→Numeric→Reciprocal**
NB: *Er zijn verschillende versies van LabVIEW beschikbaar, die verschillen in uitklapbare menu's kunnen hebben.*

1. Wat is LabVIEW?

LabVIEW[1][2] is een grafische ontwikkelomgeving gericht op het aansturen en uitlezen van meetinstrumenten en op de presentatie en verwerking van meetgegevens. De naam LabVIEW staat voor **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench. In LabVIEW kunnen "virtuele" apparaten worden gebouwd, zoals bijvoorbeeld een oscilloscoop of een functiegenerator.

Een fysiek apparaat dat door LabVIEW als virtueel apparaat (**VI**) nagemaakt wordt, bestaat uit een buitenkant voor invoer (bijvoorbeeld instelknoppen) en uitvoer (bijvoorbeeld een scherm om signalen te tonen). Daarnaast is er een binnenkant waarin de schakeling zit om invoer te lezen, signalen te verwerken en uitvoer te maken. Die buiten- en binnenkant van een VI worden in LabVIEW weergegeven door twee vensters. Het **Front Panel**, de buitenkant, geeft de invoer- en uitvoermogelijkheden weer en het **Block Diagram**, de binnenkant, het programma met de bewerkingen die "intern" uitgevoerd worden. De verbinding tussen de twee wordt door LabVIEW verzorgd.

In LabVIEW wordt gebruik gemaakt van de grafische programmeertaal **G**. Grafisch wil hier zeggen dat de programmeur geen regels code hoeft te schrijven, maar dat een programma op het Block Diagram wordt gemaakt door functiesymbolen (**Icons**) met draden (**Wires**) te verbinden. De verbinding tussen Front Panel en Block Diagram wordt gemaakt door in- en uitvoerelementen (**Terminals**) op het Front Panel te plaatsen, die door LabVIEW meteen ook op het Block Diagram geplaatst worden en zo in het "programma" op het Block Diagram geïntegreerd kunnen worden.

Een groot voordeel van LabVIEW ten opzichte van klassieke programmeertalen is dat de programmeur dus geen aandacht hoeft te besteden aan het programmeren van een gebruikersinterface. Vrijwel alle bedieningselementen die op apparaten kunnen voorkomen zijn al in LabVIEW aanwezig en kunnen met een paar muisklikken op het Front Panel worden gezet. Hierbij valt te denken aan bijvoorbeeld (draai)knoppen, schakelaars, meters, oscilloscoopschermen.

Verder heeft de programmeur bij het maken van het Block Diagram de beschikking over een zeer uitgebreide bibliotheek functies, van een simpele vermenigvuldiging tot bijvoorbeeld een Fouriertransformatie of matrixvermenigvuldiging. Ook is het mogelijk om zelfgemaakte VI's als functies in andere VI's te gebruiken en meerdere VI's tegelijk te laten lopen.

Aangezien een LabVIEW programma niet bestaat uit regels code die na elkaar uitgevoerd worden, wordt de uitvoeringsvolgorde van een VI op een andere manier bepaald. Functies op het Block Diagram worden pas uitgevoerd als alle gegevens op de draden aan de ingang aanwezig zijn. Dit systeem wordt "data-dependent execution" genoemd en talen die op deze manier werken heten "data-flow languages". Omdat het in sommige gevallen toch nodig is om expliciet een volgorde op te leggen, zijn er in LabVIEW speciale functies aanwezig om dat te kunnen doen.

LabVIEW is ontwikkeld door National Instruments. Dit bedrijf levert een scala aan apparatuur die op de computer aangesloten kan worden om bijvoorbeeld spanningen te meten, signalen op te wekken en andere apparaten aan te sturen. Bij geavanceerdere wetenschappelijke meetapparatuur wordt vrijwel altijd een stuurprogramma voor LabVIEW meegeleverd. Dit maakt het zeer eenvoudig een apparaat via de computer te bedienen en het verrichten van metingen te automatiseren. LabVIEW wordt dan ook op zeer veel plaatsen in de wereld toegepast bij fysisch onderzoek.

---

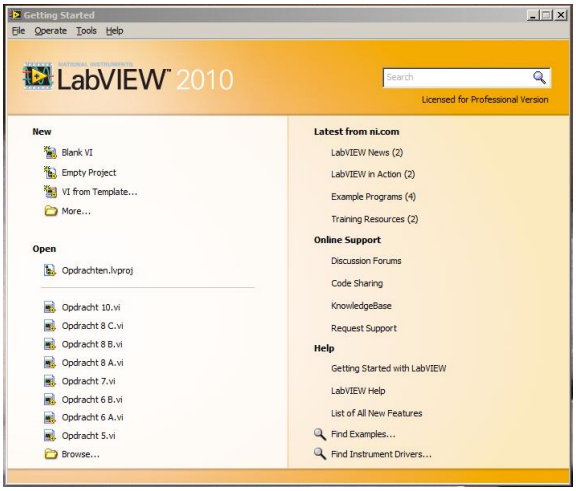
[1] LabVIEW is een trademark van National Instruments Corporation.

[2] Dit hoofdstuk is gedeeltelijk gebaseerd op de handleiding bij de cursus "Data-acquisitie met behulp van de microcomputer en LabVIEW" van de Universiteit Utrecht (J.J. Keijser, J. Kuperus, G. Bardelmeijer en P. de Wit, 1995).

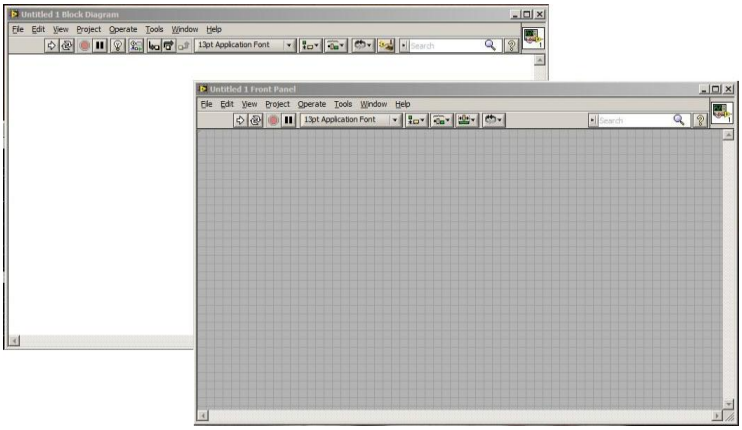
2. LabVIEW opstarten

Mogelijk is LabVIEW nog niet volledig geïnstalleerd op de pc, zie de website (www.hef.ru.nl/~pfk/education/ic) voor instructies.

Na het starten van LabVIEW via de het Start Menu of de snelkoppeling op het bureaublad verschijnt het **Getting Started** venster. Door op **New→Blank VI** te klikken wordt er een nieuwe VI aangemaakt en verschijnt er een leeg Front Panel en een corresponderend Block Diagram.



Het Getting Started scherm van LabVIEW versie 2010.



Het Block Diagram en het Front Panel.

Opslaan van programma's

VI's worden op de gebruikelijke manier met **File→Save** opgeslagen. Front Panel en Block Diagram zijn gekoppeld en worden dus automatisch beide opgeslagen. Het gebruik van **File→Save as** werkt iets anders dan normaal. Er verschijnt een venster met de vraag welke VI in het geheugen moet blijven, het origineel of de kopie of beide. De eerste (en standaard al geselecteerde) optie "Substitute copy for original" komt overeen met hoe "Save as" in de meeste andere programma's werkt, het wordt aanbevolen deze manier van opslaan te kiezen.

***NB**: Let erop dat VI's altijd op de "home disk" (op FNWI is dat* **U:** *onder MS-Windows) opgeslagen worden en niet op een lokale schijf, die laatste kan namelijk gewist worden als je uitgelogd hebt! Je kunt VI's ook op je bureaublad zetten om te bewaren.*

3. LabVIEW programmeeromgeving

De LabVIEW programmeeromgeving bestaat uit de volgende componenten:

- De **Front Panel** en **Block Diagram** vensters met verschillende functies op de menubalken en het pictogram van de VI in de rechter bovenhoek.
- Het **Controls** pop-up venster of **Controls Palette**, gekoppeld aan het Front Panel.
- Het **Functions** pop-up venster of **Functions Palette**, gekoppeld aan het Block Diagram.
- Het **Tools** pop-up venster of **Tools Palette**.

Het **Front Panel** dient ervoor om tijdens de werking van de VI invoer te geven of te wijzigen (instellingen zoals gevoeligheden, standen van schakelaars) en om uitvoer zichtbaar te maken (in numerieke of grafische vorm).

Het **Block Diagram** bevat het programma dat gegevens van de invoer leest, deze gegevens bewerkt en de resultaten toont via de uitvoer. Tijdens het bouwen van de VI programmeert men door draadverbindingen aan te brengen tussen elementaire functieblokjes (functiesymbolen uit het Functions venster) en elementen van het Front Panel, die allemaal ook op het Block Diagram getoond worden, de zogenaamde **Terminals**. Een element van het Front Panel kan als instelling (**Control**, hiervan kan de waarde tijdens uitvoering van de VI veranderd worden) of als uitvoerelement (**Indicator**, hiervan kan de waarde tijdens uitvoering van de VI niet veranderd worden, er worden alleen maar waarden weergegeven die uit het programma komen) dienen. In de functieblokjes kunnen allerlei bewerkingen plaatsvinden, zoals optellen, vermenigvuldigen, lineaire kleinste-kwadraten aanpassingen (LKK's), Fast Fourier Transforms (FFT's), enz.

Het **Controls Palette** verschijnt als tijdelijk venster als er met de rechtermuisknop in het Front Panel venster geklikt wordt. Zodra het Front Panel verlaten wordt en dus niet meer actief is, verdwijnt het Controls Palette weer. Door de pin in de linkerbovenhoek van het tijdelijke venster aan te klikken, wordt het venster “permanent” zichtbaar, dat wil zeggen, als het Controls Palette actief is. Het is een menu met submenu’s die de elementen bevatten, waaruit geput kan worden om op het Front Panel te gebruiken.

Het **Functions Palette** verschijnt als tijdelijk venster als er met de rechtermuisknop in het Block Diagram venster geklikt wordt. Het werkt analoog aan het Controls Palette, maar dan in combinatie met het Block Diagram. Het is een menu met submenu’s die de functies bevatten, waaruit geput kan worden om op het Block Diagram te gebruiken.

Het **Tools Palette** is een menu om een tool voor een bepaalde bewerking te selecteren (bijvoorbeeld Wiring Tool, Positioning Tool; deze worden later behandeld). Het venster wordt permanent geopend door in de menubalk van Front Panel of van Block Diagram **View→Tools Palette** te kiezen. Analoog kunnen zo ook het Controls Palette vanuit het Front Panel en het Functions Palette vanuit het Block Diagram permanent geopend worden. De tools van het Tools Palette kunnen statisch gebruikt worden, dat wil zeggen dat een tool in het Tools Palette geselecteerd moet worden om het te gebruiken. Maar dynamisch gebruik is ook mogelijk, dan wordt het tool dat het meest voor de hand ligt om gebruikt te worden, gekozen, afhankelijk van de positie van de cursor ten opzichten van elementen en bedrading.

Het pictogram van de VI (**Icon pictogram**, **VI Icon** of simpelweg **Icon**), dat rechtsboven in het venster van Front Panel en Block Diagram getoond wordt, dient om een gebouwde VI in een andere VI te kunnen gebruiken. Een op deze manier gebruikte VI wordt **sub-VI** genoemd.

Front Panel en Block Diagram

Het Front Panel bepaalt hoe de VI eruit gaat zien. De metafoor (en daarmee de naamgeving) is die van het knoppenpaneel van een instrument. Dit is het “user interface” van de VI. Gewoonlijk wordt dit een eigen ontwerp, maar het is ook mogelijk om het

bedieningspaneel van een bestaand instrument nauwkeurig na te bootsen. Door ergens op het werkblad van het venster met de rechtermuisknop te klikken, komt het Controls Palette te voorschijn. Hieruit kan vervolgens het gewenste object gekozen worden. In het Block Diagram verschijnt dan automatisch de bij het object behorende Terminal.

In het Block Diagram worden de verbindingen tussen de invoer en de uitvoer gelegd, het code-gedeelte van de VI. Door ergens op het werkblad van het venster met de rechtermuisknop te klikken, komt het Functions Palette te voorschijn.

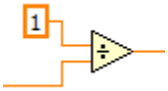
Om Front Panel en Block Diagram naast elkaar te krijgen, kan één van de **Tile** opties van het Window menu uit de menubalk gekozen worden. Dit Window menu is zowel in het Front Panel als in het Block Diagram venster aanwezig. Om te wisselen tussen de beide vensters wordt de optie **Show Block Diagram** van het Window menu in het Front Panel venster gekozen of de optie **Show Front Panel** van het Window menu in het Block Diagram venster. Men kan ook eenvoudigweg met de linkermuisknop in het gewenste venster klikken.

Hiërarchie van VI's en sub-VI's

In principe is het mogelijk om iedere VI weer binnen een andere VI te gebruiken. We spreken dan van een **sub-VI**. Een sub-VI is te vergelijken met een functie in een klassieke programmeertaal. Net zoals bij een functie kan men parameters meegeven en komt er eventueel een functiewaarde (of -waarden) terug. Hierdoor ontstaat de hiërarchische structuur van VI's.

In feite zijn alle functies uit het Functions Palette in LabVIEW ook te beschouwen als sub-VI's. Zo zou bijvoorbeeld de ingebouwde functie Functions→Arith & Compar→Numeric→Reciprocal

voorgesteld kunnen worden door de volgende sub-VI:



waarbij de niet aangesloten draden de aansluitpunten (“Terminals”) voorstellen.

Om de code waarin een sub-VI voorkomt begrijpelijker te maken kun je ook een **Icon** maken voor een sub-VI en kan er een beschrijving aan worden toegevoegd. Als de sub-VI dan wordt opgenomen in een programma wordt de hele sub-VI gerepresenteerd door het Icon. Bij ieder project is het van groot belang om de code goed te documenteren. Vooral als er samengewerkt moet worden is het belangrijk om goed (=duidelijk) commentaar te schrijven in de (sub-)VI's en om duidelijke Icons te maken.

Een sub-VI maken gaat als volgt: maak eerst een kopie van de oorspronkelijke VI, selecteer dan in het Block Diagram alle elementen die tot de sub-VI behoren, kies uit de menubalk **Edit→Create sub-VI**. Het pictogram kan gewijzigd worden door er met rechts op te klikken en te kiezen voor **Edit Icon**. Sla de nieuwe sub-VI altijd op onder een nieuwe naam!
Een sub-VI opnemen in een Block Diagram gebeurt door de optie **Select a VI...** van het Functions Palette (onderaan) van het Block Diagram te kiezen.

Front Panel knoppen

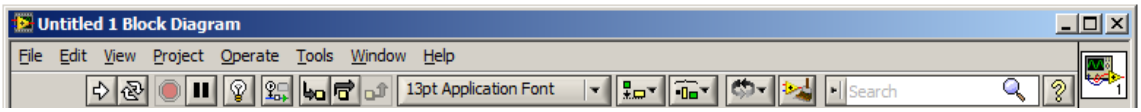
Het Front Panel venster bevat behalve titel en menubalk nog twee onderdelen: een knoppenbalk en rechtsboven het Icon (zie figuur 3.1). Het laatste is alleen van belang bij het maken van sub-VI's. De knoppenbalk en de Tools optie uit de menubalk bevatten een aantal knoppen waarmee de belangrijkste handelingen van LabVIEW geselecteerd kunnen worden. De belangrijkste knoppen worden hieronder beschreven, waarbij in de linker kantlijn steeds een afbeelding van de knop is opgenomen.

Figuur 3.1: De kop van het Front Panel.

	**Run**. Deze knop wordt gebruikt om een VI eenmalig uit te voeren.
	**Broken Run**. Soms ziet men in plaats van de Run knop de Broken Run knop. In dat geval zit er een fout in de VI, waardoor deze niet kan worden uitgevoerd. Door op de knop te drukken verschijnt een venster met foutmeldingen. Deze foutmeldingen worden gegenereerd bij het compileren van de VI.
	**Continuous Run**. Met deze knop kan een VI doorlopend uitgevoerd worden. Vooral voor kleine test VI's is dit handig. Dit zal zelden voorkomen bij grotere VI's, omdat die zelf meestal code bevatten in de trend van “Doe-dit-totdat-op-Stop-wordt-gedrukt”. Als men op de Continuous Run knop heeft gedrukt, kan de VI gestopt worden door nogmaals hierop te drukken. Men kan ook op de Stop knop drukken, maar dit is niet aan te raden (zie de beschrijving van deze knop).
	**Stop**. Deze knop verschijnt alleen als de VI loopt. Hiermee kan de VI gestopt worden, maar pas op: hiermee wordt de VI direct gestopt, hetgeen soms nadelige gevolgen kan hebben. Zo gaan bijvoorbeeld gegevens, die nog niet naar disk weggeschreven zijn, verloren.

Block Diagram knoppen

Dit venster heeft wat meer knoppen dan het Front Panel venster. Deze worden gebruikt voor het opsporen van fouten in een programma. Hieronder worden de belangrijkste van de nieuwe knoppen beschreven.



Figuur 3.2: De kop van het Block Diagram venster.

	**Highlight Execution**. Een zeer handige knop voor het opsporen van fouten. Hiermee kan men de waarden zien die tijdens het uitvoeren van een VI over de draden gaan. De uitvoering van de VI wordt hierdoor wel een beetje vertraagd. Door nog een keer te klikken zet men de highlighting uit en wordt de VI weer op normale snelheid uitgevoerd.
	**Pause**. Deze knop dient om de instructies van het programma stap voor stap uit te laten voeren (“single step mode”). Hoe groot de “single steps” zijn, wordt bepaald door de volgende twee knoppen in deze tabel, Step Into en Step Over.
	**Step Into**. Als men hierop klikt, wordt de eerste instructie van een samengestelde instructie (een sub-VI, een loop, een sequence structuur) uitgevoerd, waarna de volgende instructie binnen die samengestelde instructie uitgevoerd kan worden.
	**Step Over**. Als men hierop klikt, wordt een samengestelde instructie (een sub-VI, een loop, een sequence structuur) helemaal uitgevoerd tot de volgende (samengestelde) instructie.

	**Step Out**. Deze knop wordt ingedrukt om de single step mode te verlaten en het uitvoeren van het Block Diagram af te ronden.

Tools Palette knoppen

Het Tools pop-up scherm dat hieronder getoond wordt, bevat een aantal knoppen om verschillende gereedschappen of “tools” te selecteren voor gebruik. Als de keuze gemaakt wordt voor automatische detectie van het te gebruiken tool, hoeft dit pop-up scherm niet gebruikt te worden.

De belangrijkste tools van het Tools Palette worden in onderstaande tabel beschreven.

	**Operate Value / Operating Tool**. Door op Operating Tool te klikken kunnen de instellingen van de Controls van de VI worden gewijzigd. Deze nieuwe waarden worden dan de eerstvolgende keer dat de VI uitgevoerd wordt, gebruikt. Dit is een handig tool om de beginwaarden van de VI mee te veranderen. Als een VI op disk wordt bewaard, worden ook de laatste instellingen van de Controls bewaard.
	**Position Size/Select / Positioning Tool**. Dit wordt veel gebruikt bij het maken van een VI. Hiermee kunnen voorwerpen verplaatst en vergroot of verkleind worden. Om een voorwerp te verplaatsen klikt men op het voorwerp totdat er een gestippelde rechthoek om het voorwerp heen verschijnt. Klik nu binnen de gestippelde rechthoek en sleep het voorwerp met de muis naar de gewenste locatie. Als men een voorwerp in het Front Panel venster verplaatst, heeft dat geen effect op het Block Diagram venster en omgekeerd.
	**Edit Text / Labeling Tool**. Hiermee kunnen teksten en getallen op het Front Panel venster worden veranderd of toegevoegd. Men kan vrije teksten toevoegen aan het Front Panel venster of teksten die gebonden zijn aan specifieke objecten. Dit is vooral handig voor commentaar. Het gaat niet altijd even gemakkelijk om teksten te wijzigen. Er moet nauwkeurig gemikt worden met de muis om ervoor te zorgen dat de bestaande tekst gewijzigd wordt en niet een nieuwe wordt toegevoegd.
	**Connect Wire (klosje) / Wiring Tool**. Het klosje heeft men vooral nodig in het Block Diagram venster om verbindingen te leggen. In het Front Panel venster wordt het gebruikt bij het maken van een sub-VI om de terminals mee aan te geven. In het Block Diagram venster verbindt men hiermee verschillende onderdelen van de VI met elkaar. Om twee objecten met elkaar te verbinden doe je het volgende klik op Connect Wire. De cursor verandert nu in een klein klosje. Schuif de muis naar het eerste object totdat dit oplicht. Klik met de muis en ga naar het tweede object. Om de draad, die automatisch getrokken wordt, te knikken drukt men met de muis op de punten waar de knik moet komen. Als men met de muis over het tweede object gaat, zal ook dit object oplichten. Druk nu wederom op de muisknop. Als de draad niet goed getrokken is of de objecten niet met elkaar verbonden mogen worden, dan wordt de draad gestippeld en zwart van kleur. De kleur en de dikte van een goede draad hangt af van het type signaal dat erover loopt. De codering van de draden is te vinden in de handboeken van LabVIEW, of eenvoudig door het Context Help venster te openen (Ctrl-H) en met het klosje de betreffende draad aan te wijzen. ***NB**: Vooral bij het knikken van draden ontstaan vaak losse stukjes “draad”, die nergens aan verbonden zijn. Als*

	dit gebeurd is, zal de Run knop gebroken zijn. Gebruik dan de optie "Remove Bad Wires" van het Edit menu om de losse draden op te ruimen (of de combinatie control-toets met B, Ctrl-B).
	**Set Color / Coloring Tool**. Dit gebruikt men alleen om een VI mee te verfraaien. Hiermee kan de kleur van een object of text worden veranderd. Een handige kleur is "T" (transparant); deze kan bijvoorbeeld worden gebruikt om de randen rond een stukje text of een grafiek te verwijderen.
	**Automatic Tool**. Deze knop kan groen of zwart gekleurd zijn. Als groen geselecteerd wordt, kiest LabVIEW, afhankelijk van waar de muis staat, de meest waarschijnlijk benodigde Tool. Het vergt wat handigheid, vooral wat betreft het positioneren van de muis, maar eenmaal aangeleerd is deze manier van Tool selecteren een stuk sneller dan elke keer zelf de benodigde Tool kiezen uit het Tools Palette.
	**Breakpoint, Set/Clear**. Hiermee kun je een breakpoint zetten in een VI. Dit is alleen handig bij grotere projecten met een hiërarchische opbouw. Als in een sub-VI een breakpoint wordt gezet zal het hoofdprogramma uitgevoerd worden totdat de sub-VI wordt aangeroepen. Op dat moment wordt de controle aan de gebruiker teruggegeven.
	**Probe Tool**. Deze Tool wordt gebruikt om tussenwaarden van de VI te controleren. Een ingewikkeld Block Diagram kan bijvoorbeeld een reeks operaties bevatten, waarbij een of meer van de operaties onjuiste gegevens produceren. De Probe Tool wordt op de gewenste plaats gezet, door eerst op deze knop te klikken en vervolgens op de draad waardoor de gegevens lopen. Er verschijnt dan een venstertje, dat zowel op het Diagram als op het Panel te zien is. Hierin wordt continu de waarde die over de draad loopt weergegeven.

Het Icon pictogram

Het Icon pictogram (Icon) is steeds zichtbaar in de rechterbovenhoek van zowel het Front Panel als het Block Diagram venster. Het toont het pictogram voor de complete VI, zoals het verschijnt wanneer de VI gebruikt wordt als sub-VI in andere applicaties. In het Front Panel venster kan het pictogram geopend worden door erop te dubbelklikken. Het pictogram kan hier ontworpen worden. Door rechtsklikken op het pictogram in het Front Panel en **Show Connector** te kiezen, kunnen de in- en uitvoer terminals (connectors) worden gedefinieerd. Gebruik de Wiring Tool om een verbinding te leggen tussen een invoer terminal en een control of een uitvoer terminal en een indicator (klik eerst op de terminal en vervolgens op de Control/Indicator).

Het verloop van een programma

Zoals in het eerste hoofdstuk beschreven is, is LabVIEW een data-flow gestuurde taal. Een instructie wordt pas uitgevoerd als alle invoer voor die instructie beschikbaar is. Bij een optelling van twee getallen bijvoorbeeld, zullen de waarden van beide getallen beschikbaar moeten zijn om de optelling uit te kunnen voeren. Als gevolg hiervan kunnen delen van een VI onafhankelijk van elkaar worden uitgevoerd. Het deel dat als eerste alle invoerwaarden beschikbaar heeft, wordt ook als eerste uitgevoerd. Welk deel als eerste aan de beurt komt, is niet altijd van tevoren te voorspellen en hangt van de omgeving af. Men moet zich deze filosofie bewust zijn, en dit vraagt (zeker voor mensen die klassieke programmeertalen gewend zijn) enige omschakeling. In een aantal gevallen zullen we toch een vaste volgorde willen vastleggen voor de uitvoering van instructies. Hiervoor is een speciale functie beschikbaar, die later behandeld wordt.

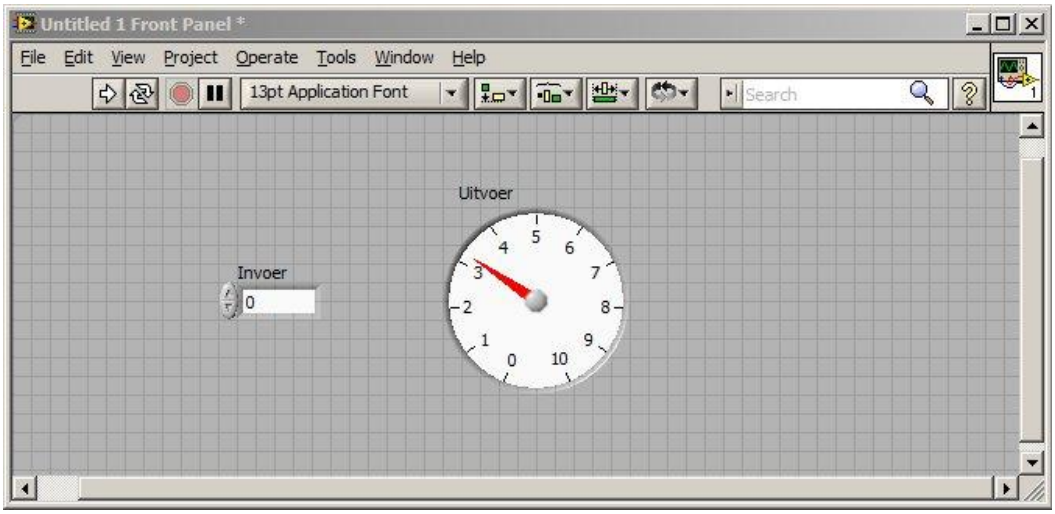
4. Werken met LabVIEW

In- en uitvoer

De invoer van de gebruiker voor de VI en de uitvoer op het scherm kunnen op een heleboel manieren geregeld worden. Zoals eerder vermeld, heten invoerelementen op het Front Panel **Controls** en uitvoer elementen **Indicators**. Deze worden op het Block Diagram verbonden tot VI's. De beste manier om dit principe te begrijpen is door zelf VI's te maken.

Voorbeeld 1: In- en uitvoer

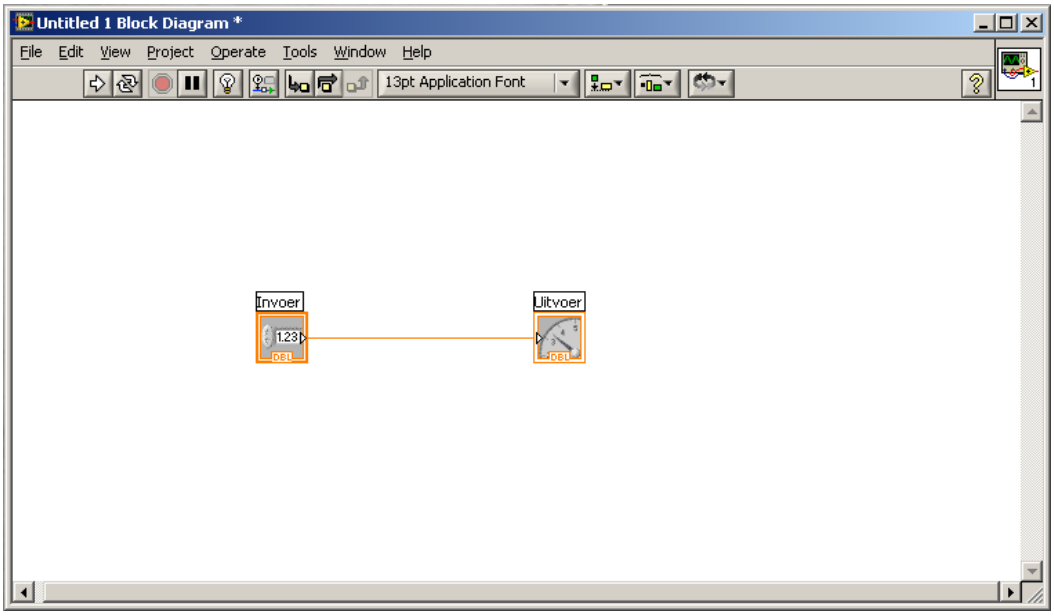
Klik met de rechtermuisknop in het Front Panel venster om het Controls Palette te laten verschijnen. Kies het **Num Ctrl** (numeriek invoer element) uit het **Controls→Num Ctrls** submenu (ofwel: kies **Controls→Num Ctrls→Num Ctrl**). Dit is een invoerelement (Control), het kan op het Front Panel gewijzigd worden als de VI eenmaal loopt. Een object verschijnt op het werkblad met een vlakje voor naamgeving (**label**). Tik bijvoorbeeld “Invoer” in om dit invoerelement een naam te geven. Naamgeving is niet strikt noodzakelijk, maar maakt de VI wel veel overzichtelijker. Als het object verplaatst wordt, gaat het label mee, omgekeerd niet. Merk op dat er op het Block Diagram een object verschijnt met het label “Invoer”.
Doe het bovenstaande nogmaals, maar kies nu een **Controls→Num Inds→Gauge**. Noem dit object “Uitvoer”. Dit is een uitvoerelement (Indicator), waar alleen waarden naartoe geschreven kunnen worden. Ook hier verschijnt er een object met label “Uitvoer” op het Block Diagram. Klik nu op het object “Invoer” en positioneer die op het werkblad op de gewenste plaats. Doe hetzelfde voor het object “Uitvoer”. Het Front Panel ziet er dan ongeveer als volgt uit:



Klik nu op het Block Diagram venster. Daar staan dus de twee symbolen met de labels “Invoer” en “Uitvoer” erboven. Kijk goed naar de verschillen tussen de twee symbolen, want het is erg gemakkelijk om ze te verwarren en het is belangrijk ze uit elkaar te kunnen houden: een Control is de bron van een signaal en een Indicator alleen maar de ontvanger.

Ga naar het Tools palet en kies hier de Wiring Tool. Verbind de twee objecten. Zie voor een uitvoerige uitleg paragraaf \ref{panel. Als de verbinding goed gelegd is, verschijnt er een oranje draad tussen de twee objecten.

Als de Run Knop een gebroken pijl vertoont, zijn de draden niet goed verbonden. Kies van het Edit menu de optie “Remove Bad Wires” om de slechte draden te verwijderen. Dit kun je ook doen door Ctrl-B in te tikken. Als de draden nu wel goed aangesloten zijn, verandert de gebroken pijl in een gewone pijl. Het Diagram ziet er dan als volgt uit:



Klik met de Operating Tool op de Continuous Run Knop. Voer een paar keer een getal in. Om de VI stop te zetten klik je nogmaals op de Continuous Run Knop of klik je op de Stop Knop. Standaard loopt de schaal van de Gauge van 0 tot 10. Om deze te wijzigen beweeg je de muis over de “10” van de schaal heen. De cursor verandert dan van een handje in een text-cursor. Klik nu met de muis en voer het nieuwe maximum in. Het is dus erg gemakkelijk om invoer en uitvoer van een VI te regelen. Belangrijk is dat de eigenlijke VI (het Diagram) niet gewijzigd hoeft te worden.

Sluit de VI met de optie “Close” van het File menu.

Instellingen van Controls en Indicators spelen in LabVIEW de rol van declaraties in “gewone” programmeertalen. Ze worden bijvoorbeeld gebruikt om data type[3] en data range[4] te definiëren. In het hierboven gegeven voorbeeld zijn zowel invoer als uitvoer **double-precision** getallen (DBL). Roep, om hiervan 32-bit (long) **integers** te maken, het pop-up menu van de “Invoer” control op (rechtsklikken op de variabele) en kies I32 (long integer) uit het “Representation” palet. De bijbehorende terminal op het Diagram verandert automatisch: hij wordt blauw met aanduiding “I32”.
In het menu kan meteen ook data range in het pop-up menu worden gekozen. Hiermee worden de maximum en minimum waarden die het getal aan kan nemen aangegeven. Voor “unsigned byte” (U8) bijvoorbeeld, loopt de standaard data range van 0 tot en met 255; voor “byte” (I8) van -128 tot en met +127. Verder kun je je eigen limieten instellen binnen deze standaardwaarden.
NB: *Merk op dat oranje altijd bewerkingen op/met reële getallen aangeeft en blauw op/met gehele getallen.*

Andere data types (booleans, strings, arrays, lists en rings, clusters) zijn te vinden in aparte vakjes van het Controls Palette. Text rings (of Menu rings, of Dialog rings, ze werken allemaal op hetzelfde principe) zijn handig als je de gebruiker onverenigbare opties wilt geven, bijvoorbeeld trigger types. Ze zijn te vinden in het **Controls→Text Controls→Text Ring** subpalet. Om de naam van een item in te vullen, gebruikt men het Labeling Tool (A'tje). Verder kan een “digital display” zichtbaar worden gemaakt en kan “Add item after, Add item before” worden gebruikt om meer artikelen op de lijst te zetten.

[3] Data types zijn bijvoorbeeld geheel getal (**integer**), reëel getal (**real** of **double precision**), tekst.
[4] Geeft aan wat het bereik van een getal is, een integer (I16) heeft bijvoorbeeld een bereik van -32768 tot en met +32767, een long integer (I32) van -2147483648 tot en met +2147483647. Voor integers geldt de oplopende reeks I8, I16, I32, I64 en voor reals geldt **single precision** (SGL), **double precision** (DBL), **extended precision** (EXT).

Soms is het handig om een hele verzameling elementen (getallen, strings, etc.) over één draad te laten lopen. Hierbij kan gebruik worden gemaakt van arrays (geordende verzamelingen van gelijke typen data) of clusters (een verzameling van ongelijke typen data, bijvoorbeeld een combinate van een getal, een string en een Boolean).

Functions & Terminals

Bij het volgende voorbeeld krijgen we voor het eerst te maken met de ingebouwde functies van LabVIEW. Deze functies kunnen meer dan één aansluitpunt (terminal) terminal hebben. Vaak is het noodzakelijk om alle terminals van een functie aan te sluiten voordat deze goed is opgenomen in de VI. Het komt ook voor dat terminals niet aangesloten hoeven te worden. In dat geval wordt een standaard waarde (default) gebruikt voor de terminal.

In dit voorbeeld krijgen we te maken met de ingebouwde functie MULTIPLY, die twee getallen met elkaar vermenigvuldigt. De input terminals zijn X en Y, op de output terminal komt het resultaat X*Y te staan. Deze functie heeft dus drie terminals. Als men het klosje over de MULTIPLY functie heen beweegt, gaat de dichtstbijzijnde terminal knipperen. Door nu met de muis te klikken wordt de knipperende terminal aangesloten. Alle functies hebben aan de linkerkant de input terminals en aan de rechterkant de output terminals. De meeste VI's houden zich ook aan deze conventie.

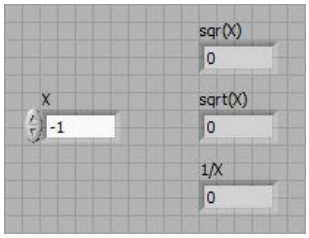
Context Help

Om te zien wat op welke terminal aangesloten moet worden kun je het **Context Help** venster openen. Dit doe je door van het Help menu de optie “Show Context Help” te kiezen. Het Context Help venster geeft ook informatie over de terminals van gemaakte sub-VI's. Net zoals bij ingebouwde functies en meegeleverde sub-VI's knippert in het Context Help venster de terminal waarover het klosje bewogen wordt.

Voorbeeld 2: Berekeningen

In deze oefening gaan we een VI maken die invoer aan de gebruiker vraagt en vervolgens het kwadraat, de wortel en de reciproke waarde (1/x) van deze invoer bepaalt. We beginnen met een nieuwe VI. Kies hiervoor de optie “New” van het File menu in het Front Panel venster. Creëer op het Front Panel venster een object van het type **Controls→Numeric Controls→Num Ctrl** en drie van het type **Controls→Numeric Indicators→Num Ind**. Noem het Control object X en de Indicator objecten sqr(X), sqrt(X) en 1/X respectievelijk.

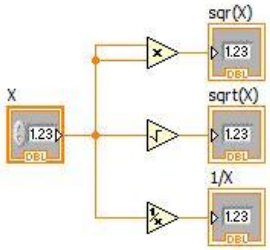
Ga naar het Tools Palette en kies de Labelling Tool. Klik hiermee op het label van één van de objecten. Het label kan dan veranderd worden. Druk niet op “Enter” als je klaar bent; want hiermee krijgt het object een label van twee regels omdat “Enter” als nieuwe regel in het label wordt geïnterpreteerd. Klik in plaats daarvan met de muis ergens op het werkblad. Positioneer de objecten op het werkblad waar je ze wilt hebben. Het Front Panel ziet er dan ongeveer als volgt uit:



Ga naar het Block Diagram. Hierin zijn nu vier objecten: één Control (invoer) en drie Indicators (uitvoer) objecten. Kies uit het **Functions→Programming→Numeric** subpalet de functie MULTIPLY. Let op: het kan zijn dat het Functions Pallet uitgeklapt moet worden door onderin op de twee V’s te klikken.

Doe hetzelfde nog twee keer, maar kies nu de functie SQUARE ROOT resp. RECIPROCAL Nu zijn alle basiselementen voor de VI aanwezig.

Ga naar het Tools palet en druk op de Wiring Tool. Verbind de objecten zoals aangegeven in het volgende Diagram:



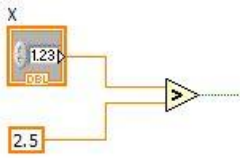
Voer de VI uit door op de Continuous Run knop te drukken en controleer de werking van de VI door verschillende invoerwaarden te geven. Bewaar de VI.

Logische structuren

In iedere programmeertaal is het noodzakelijk om beslissingen te kunnen nemen. Zo ook in LabVIEW. Er zijn een groot aantal functies voorgedefinieerd om vergelijkingen uit te kunnen voeren: $x > 0$, $x \geq 0$, $x = y$, $x \neq y$, $x > y$, $x \geq y$, enz. Al deze functies hebben een icon waarbij TF (**True** of **False**) aangeeft dat er een boolean waarde uitkomt: true of false, waar of niet waar. Het stukje code:

ALS (x > 2.5) DAN . . .

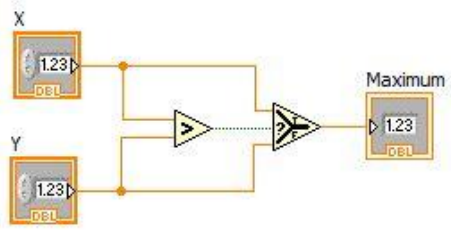
komt er in LabVIEW als volgt uit te zien:



waarbij de functie GREATER? uit het **Functions→Programming→Comparison** subpalet werd gebruikt en de constante 2.5 gemaakt werd met **Functions→Programming→Numeric→Numeric Constant**. Control X en de constante zijn beide van data type DBL.

Een andere handige functie is **Functions→Programming→Comparison→Select**:

Hiermee kan men de waarde van de uitvoerterminal kiezen tussen die van de bovenste invoerterminal (T,TRUE) in het geval de conditie waar is, of die van de onderste terminal (F,FALSE) in geval de conditie niet waar is. Een voorbeeld hiervan is een functie om het maximum te bepalen van twee getallen:



While Loop en For Loop

In veel programma's is het noodzakelijk om handelingen meerdere malen te verrichten. Hiervoor heeft LabVIEW de While Loop en de For Loop. We zullen nu deze loops onder de loep nemen.

While Loop

Een WHILE LOOP wordt gevonden onder **Functions→Programming→Structures**. Zodra deze optie is aangeklikt verschijnt er in het diagram een klein vierkantje. De rechteronderzijde daarvan kan met de muis om het te herhalen stuk schakeling worden gesleept. Er verschijnt in het diagram dan de volgende rechthoek:

Hierbij is het vierkantje met de "i" de Loop Counter. Iedere keer dat de loop doorlopen wordt, wordt de teller met één verhoogd. Het vierkantje met de cirkel-pijl is de Loop Condition. Zolang deze conditie True is wordt de loop uitgevoerd. Zodra de waarde False wordt aangeboden wordt de loop afgebroken.

De loop wordt dus altijd ten minste één keer uitgevoerd. In een ingewikkeld programma kan dat tot ongewenste resultaten lijden. Om dit te voorkomen, dient men alle behandelingen binnen de While Loop in een Case Structure zetten:

Als een bepaalde variabele, die binnen een loop wordt gegenereerd, doorgegeven moet worden aan een indicator of een andere functie buiten de loop, dan dient men eerst de bijbehorende draad te verbinden met de rand van de loop en vervolgens met de indicator of het functieblokje. Er verschijnt nu een zwart vierkantje op de rand van de loop. Soms is het handig om alle waarden die binnen de loop worden gegenereerd samen in één array naar buiten te laten komen. In dat geval kan met rechts op het zwarte vierkantje geklikt worden en daarna op "Enable Indexing". In andere gevallen is het handig om alleen de laatste waarde uit de lus te laten komen. In dat geval moet juist "Disable Indexing " worden geselecteerd.

NB: *LabVIEW probeert alle instructies binnen een While Loop of een For Loop zo snel mogelijk uit te voeren. Dit kan elke 1 of 10 msec betekenen, wat in de meeste gevallen helemaal niet nodig is. Om een loop te laten uitvoeren met een bepaalde vertraging, bijvoorbeeld één keer per seconde, gebruikt men timing functies. Deze zijn te vinden in het* **Functions→Programming→Timing** *subpalet.*

Shift Registers

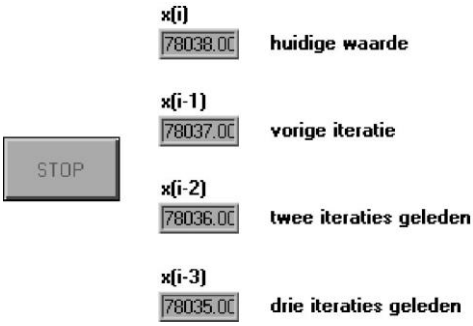
Shift Registers zijn lokale variabelen van een loop. Deze variabelen transporteren waarden van de ene iteratie van de loop naar de volgende. We zullen zo meteen zien wat het voordeel hiervan kan zijn.

Een Shift Register wordt gecreëerd door rechtsklikken op de linker- of rechterkant van de loop en de optie “Add Shift Register” te kiezen van het menu dat verschijnt. Een Shift Register met een pijl naar boven verschijnt aan de rechterkant. Aan de linkerkant van de loop verschijnt een Shift Register met een pijl naar beneden. Als een Shift Register aan de rechterkant omhoog of omlaag wordt verplaatst gaat het register aan de linkerkant mee en vice versa. Aan de linkerkant staat de waarde van de grootheid na de vorige iteratie. Aan de rechterkant staat de nieuwe waarde. Om een waarde door te geven aan de volgende iteratie moet deze dus aan de rechterkant op de loop gezet worden en aan de linkerkant van de loop afgehaald worden. Ook is het mogelijk om waarden van meerdere voorgaande iteraties te bekijken. Hiervoor klikt men aan de linkerkant van de loop op het Shift Register en kiest men de optie “Add Element”. Men moet de Shift Registers initiële waarden meegeven anders treden er tijdens de eerste loop problemen op met de waarden van de shift registers.

Het nut van Shift Registers blijkt uit het volgende voorbeeld.

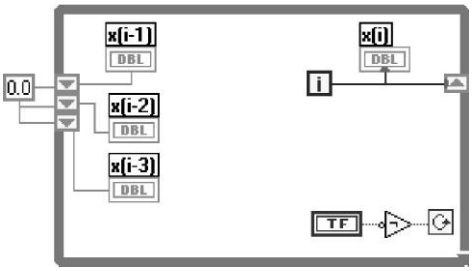
Voorbeeld 3: Shift Registers

Maak een VI met het volgende Front Panel:



Hierbij worden vier elementen van het type Numerical Indicator (Num Ind) gemaakt. Merk op dat het data-type van kan worden gekozen: 8-bit integer (I8), 16-bit (I16), 32-bit (I32), 32-bit single precision real (SGL), extended presicion (EXT), enz. Hiertoe kan men rechts-klikken op de indicator en uit het pop-up menu Representation kiezen.

De Stop knop (Stop Button) is te vinden in het Controls→Buttons&Switches palet. De Stop knop is een zogenaamde Boolean Control en wordt aangegeven met het symbool TF. De waarde die de ingedrukte Stop knop teruggeeft is True. Om de While Loop te beëindigen is echter een waarde False nodig. Daarom wordt de functie NOT uit Functions→Programming→Boolean ertussen geplaatst om de juiste waarde te genereren. Dit is een veel gebruikte constructie in LabVIEW. De labels naast de vier Indicators zijn losse labels. Deze worden gemaakt door met de Labeling Tool (A'tje) te klikken op het werkblad en het label in te tikken. Sluit het Diagram aan als getekend in de volgende figuur.

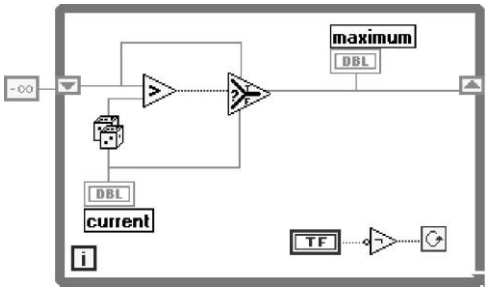


De WHILE LOOP is te vinden in Functions→Programming→Structures. Hier is de index i van de While Loop direct aangesloten op het Shift Register van de loop. De initiële waarde 0 is gemaakt met behulp van Functions→Programming→Numeric→Numeric Constant. Deze waarde wordt aangesloten op alle drie de Shift Registers. Deze laatste zijn gemaakt door eerst aan de rechterkant van de While Loop te rechts-klikken en de optie “Add Shift

Register" te kiezen van het pop-up menu. Vervolgens kan door aan de linkerkant van de While Loop te klikken een extra Shift Register worden toegevoegd.

Voer de VI uit door op de "Run" knop te drukken. Zoals te zien is, heeft deze VI een STOP optie. Zodra op de STOP knop gedrukt wordt eindigt de VI en geeft de controle aan de gebruiker terug. Hierdoor is het ook niet noodzakelijk om de "Continuous Run" button te gebruiken. Bewaar de VI.

De Shift Registers zijn een handig hulpmiddel om bepaalde bewerkingen uit te voeren. Zo is het een gemakkelijke manier om het lopende maximum van een functie bij te houden. In het volgende Diagram is als functie Functions→Programming→Numeric→Random number gekozen.



For Loop

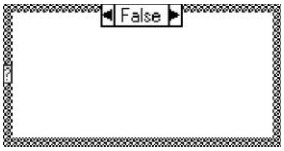
Een FOR LOOP wordt gevonden onder Functions→Programming→Structures en werkt analoog aan een WHILE LOOP. Een FOR LOOP wordt weergegeven met

De loop conditie ontbreekt, maar het aantal iteraties moet van te voren worden opgegeven. De loop begint dus bij 0 en loopt tot N-1. Dit is de conventie, die consequent in LabVIEW is doorgevoerd. Ook bij For Loops kan met Shift Registers worden gewerkt. Het gebruik is verder identiek aan dat bij While Loops.

Case Structure en Sequence Structure

Case Structure

Een CASE STRUCTURE wordt gevonden onder Functions→Programming→Structures→Case Structure en wordt weergegeven met



De CASE STRUCTURE heeft bij aanmaak een enkele terminal met daarin een vraagteken (een case selector). Deze structuur kan op twee manieren gebruikt worden:

- Als op de terminal een boolean waarde wordt aangesloten, fungeert de CASE STRUCTURE als een compound if ... then ... else statement. We hebben dan twee Diagrammen achter elkaar. Heeft de aan de terminal toegevoerde boolean de waarde True dan wordt het Diagram dat binnen het True-kader is opgenomen uitgevoerd. Idem voor False. We kunnen

aan de rechterzijde zelf uitvoerterminals aanbrengen. Een uitvoerterminal moet vanuit beide terminals van signaal worden voorzien. Men wisselt tijdens het programmeren tussen het True en False venster door een van de driehoekjes aan de bovenzijde van het CASE kader aan te klikken.

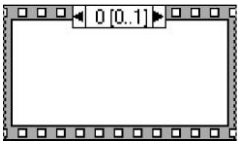
- Wanneer op de terminal een numerieke waarde wordt aangesloten, is de werking equivalent aan het boolean CASE geval, alleen zijn er nu nog meer vensters achter elkaar. Elk venster heeft midden-boven een identifier (een getal 0, 1, 2, 3, etc., of een woord). Om meer vensters toe te voegen, dient men ergens op de structuur te rechts-klikken en vervolgens de optie “Add Case” te kiezen. Als case selectors gebruikt men in dit geval verschillende numerieke selectors:
  1. Numeric controls
  2. Text rings/Menu Rings/Dialog Rings
  3. Lists
  4. Enumerated types

  Numeric controls en rings kwamen al eerder aan bod. Enumerated types zijn handig, want de identifier van een venster bevat in dit geval een stukje tekst (en niet alleen een getal).

Denk er bij het aanmaken van een numerieke CASE STRUCTURE aan, dat voor elk item in een lijst of ring een bijbehorend venster van de CASE STRUCTURE bestaat.

Sequence Structure

Een SEQUENCE STRUCTURE wordt gevonden onder Functions→Programming→Structures->Stacked Sequence en wordt weergegeven met



De SEQUENCE STRUCTURE lijkt qua opbouw enigszins op de CASE STRUCTURE. Bij LabVIEW heeft de programmeur weinig invloed op de volgorde van acties in zijn programma. Zodra een signaal op de ingangen van een functie verschijnt, wordt deze geactiveerd. Soms is het echter noodzakelijk dat handelingen in een bepaalde volgorde uitgevoerd dienen te worden. Hiervoor is de SEQUENCE STRUCTURE beschikbaar. Het getal middenboven in het kader van de structure geeft het Frame Number. Schakelingen binnen de frames worden in volgorde van Frame Number afgewerkt. Om frames toe te voegen dient men te rechts-klikken op het Frame Number en vervolgens de optie “Add Frame After” te kiezen.

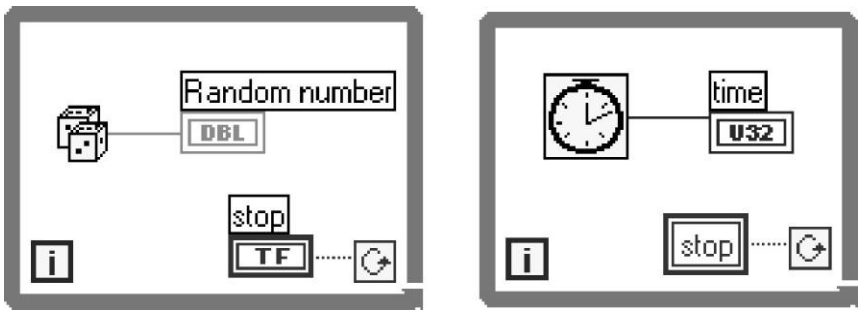
Om gegevens van het ene naar het andere frame te sturen, gebruikt men sequence locals: rechts-klik op een frame en kies “Add Sequence Local”.

Een voorbeeld van het gebruik van de Sequence Structure is te vinden in de LabVIEW→Programmeren→Self-study map: “Timing Template.vi”.

De hierboven beschreven Sequence is een zogenaamde Stacked Sequence. Dat wil zeggen dat de verschillende Frames allemaal achter elkaar op het Block Diagram staan. In sommige gevallen is het handiger als alle Frames naast elkaar staan. In dat geval kan worden gekozen voor een Flat Sequence.

Local variables

Local variables (of locale variabelen) maken het mogelijk om hetzelfde object toegankelijk te maken op verschillende plaatsen van het diagram. Daarnaast kan men m.b.v. locale variabelen een object als control en tegelijkertijd als indicator gebruiken. Een klassiek voorbeeld van het gebruik van locale variabelen in LabVIEW is om controle te krijgen over twee of meer parallel lopende lussen:



In dit voorbeeld worden beide lussen op hetzelfde moment met één STOP knop gestopt.

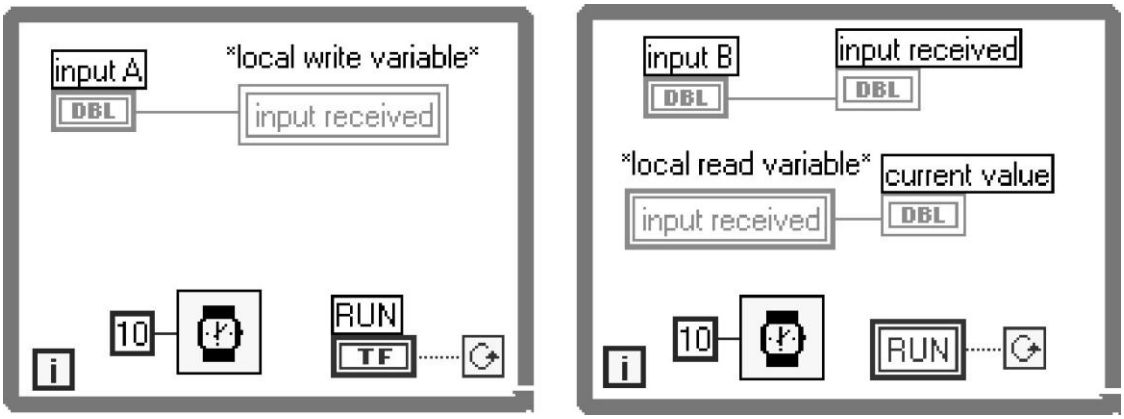
Om een locale variabele te creëren dient men op het object (control of indicator, of een terminal op het Diagram) te klikken en de optie “Create Local Variable” te kiezen. Let wel op dat een aldus gemaakte locale variabele de naam krijgt van het object, dus het object moet een naam (label) hebben. Verder dient een juiste mode van de locale variabele worden gekozen. Locale variabelen in read mode zijn te vergelijken met controls (invoer) en locale variabelen in write mode zijn te vergelijken met indicators (uitvoer):

READ mode = CONTROL

WRITE mode = INDICATOR

Een locale variabele is in feite een kopie van de huidige toestand van een Front Panel control of indicator. Het is mogelijk om voor hetzelfde object meerdere locale variabelen te creëren, sommige in write mode en sommige in read mode. Als dat het geval is, kan bijvoorbeeld naar een Front Panel control geschreven worden: een control fungeert dus zowel als invoer en als uitvoer. De gegevens worden namelijk gelezen uit de locale variabele in read mode en geschreven naar de locale variabele in write mode. Een voorbeeld is te vinden in de LabVIEW→Examples→General map: “locals.llb→One Switch Control with Reset.vi”.

Bij het gebruik van locale variabelen moet je altijd oppassen voor de zogenaamde race condition. Hierbij probeert men tegelijkertijd naar een object te schrijven en van hetzelfde object te lezen:

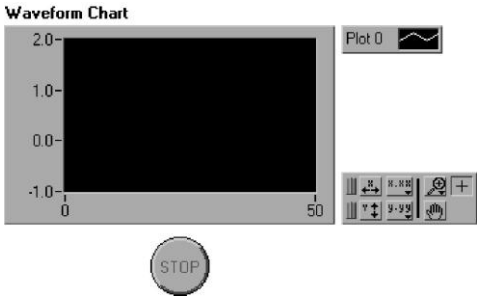


LabVIEW Visual Displays: Charts & Graphs

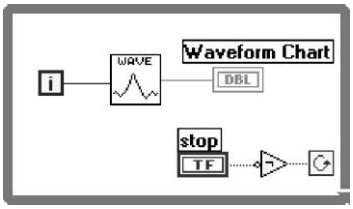
Het grafisch weergeven van data is heel simpel. In plaats van de data naar een object van het type Numerical Indicator te sturen, sturen we deze nu naar een object van het type Waveform Chart. Het gebruik wordt in het volgende voorbeeld geïllustreerd.

Voorbeeld 4: Charts

Maak een VI met het volgende Front Panel:



De grafiek is gemaakt met Controls→Graphs→Waveform Chart. De "STOP" knop is gemaakt met Controls→Buttons&Switches→Stop Button. Het Block Diagram ziet er als volgt uit:



De WHILE LOOP vind je onder Functions→Programming→Structures. Het signaal wordt gesimuleerd met de functie Functions→Tutorial→Generate Waveform.vi.

Voer de VI uit door op de "Run" knop te drukken. Bewaar de VI.

Probeer de diverse mogelijkheden uit om de schaal van de chart aan te passen. Dit kan ofwel door de getallen langs de assen te overschrijven, ofwel met behulp van het kleine knoppenpaneeltje dat met de chart op het Front Panel verschijnt. (Wanneer dit paneeltje niet zichtbaar is dan kan het gemaakt worden door rechts-klikken op de Chart en selecteren van Visible Items→Graph Palette).

In de meeste gevallen wordt de Waveform Chart binnen een While loop gezet, om nieuwe data punten meteen na acquisitie (of berekeningen) punt-voor-punt weer te geven. Er bestaan drie operation modes voor Waveform Charts: Strip Chart, Scope Chart en Sweep Chart. De laatste twee zijn vergelijkbaar met een oscilloscoop-scherm en de Strip Chart met een XT-schrijver (chart recorder).

Om meer dan één plot op een Chart weer te geven, dient men de Bundle functie gebruiken (Functions→Cluster palet).

Een aantal voorbeelden van het gebruik van Waveform Charts in verschillende modes zijn te vinden in de LabVIEW→Examples→General→graphs map: "charts.llb ".

LabVIEW kent ook twee andere types van "visual displays": Waveform Graph and XY graph. De Waveform Graph wordt gebruikt om een array van getallen weer te geven tegen het bijbehorende array van index positions van deze getallen. In tegenstelling tot de Waveform Chart, is het in dit geval ook mogelijk om een bepaald deel van een array weer te geven (bijvoorbeeld van index 10 tot index 50) of een "forced index" te gebruiken. De XY Graph zet het ene array tegen het andere uit. Meer details over deze twee vormen van "visual displays" zijn te vinden in de LabVIEW boeken.

File I/O

File I/O operaties en vaak ook instrument I/O en netwerk communicatie, gebruiken strings om data uit te wisselen. Een string is een verzameling van "displayable" (letters, cijfers) en "non-displayable" (Tab, Return, Spatie) tekens. String controls en indicators zijn te vinden in het Controls→Text Controls subpalet. String controls en indicators hebben vier verschillende display modes: normal, "\"

code (bijvoorbeeld {\t voor Tab), password, of hex (hexadecimal). Je kunt de display mode kiezen uit de pop-up menu van een string control of indicator:

De meest gebruikte "\" codes zijn:

\b=Backspace (ASCII BS)

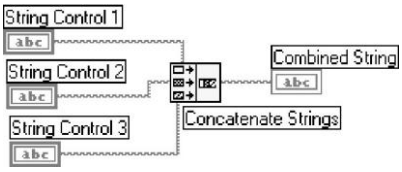
\s=Space (ASCII SP)

\r=Return (ASCII CR)

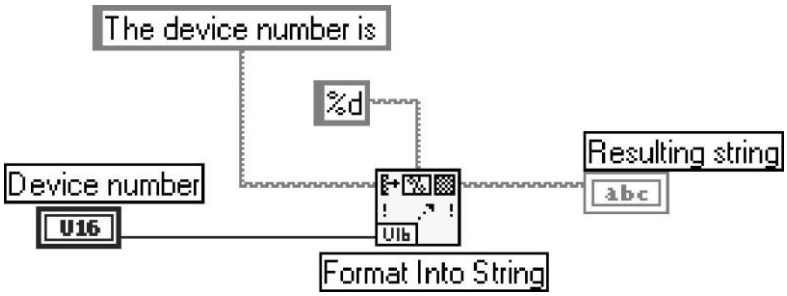
\n=Newline (ASCII LF)

\t=Tab (ASCII HT)

LabVIEW heeft een grote aantal functies om strings te behandelen. Deze String functies zijn te vinden in het subpalet Functions→Programming→String. De CONCATENATE STRINGS functie kan bijvoorbeeld worden gebruikt om een aantal strings samen te voegen:



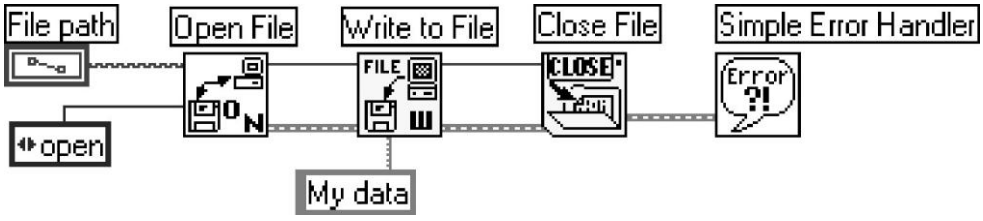
Een andere voorbeeld is de FORMAT INTO STRING functie, die elk argument in een string van het gewenste formaat omzet, bijvoorbeeld de reële waarde van een digital indicator in een string:



Open "Build String.vi" uit de LabVIEW→Programmeren→Self-study map en volg de instructies op het scherm om het bouwen van strings te oefenen.

File I/O operaties schrijven informatie in een bestand of lezen uit een bestand. File I/O functies zijn te vinden in het Functions→Programming→File I/O palet. Met behulp van de WRITE TO SPREADSHEET FILE en READ FROM SPREADSHEET FILE functies, bijvoorbeeld, kun je één- of tweedimensionale arrays in een bestand schrijven of uit een bestand lezen.

Daarnaast kan je de vier meest belangrijke handelingen van bestanden (Open File, Write to File, Read from File, Close File) zelf definiëren door gebruik te maken van overeenkomstige functies:



De bovenste draad (refnum) is hier gebruikt om aan te geven dat het om hetzelfde bestand gaat. De onderste draad geeft eventuele fouten door naar volgende functies. De SIMPLE ERROR HANDLER functie meldt deze fouten op het scherm in de vorm van dialoog (deze functie bevindt zich in het Functions→Programming→Dialog & User interface subpalet).

Meer voorbeelden van hoe je File I/O functies kan gebruiken zijn te vinden in de LabVIEW→Programmeren→Self-study map: "Write Temperature to File.vi", "Read Temperature from File.vi", "Using Spreadsheet Format.vi".

Het opsporen van fouten (debuggen)

LabVIEW heeft een aantal hulpmiddelen om programmeerfouten in een VI op te sporen. Deze zogenaamde "debugging tools" bevinden zich in het Tools palet en op de knoppenbalk van het Block Diagram venster. Deze hulpmiddelen zijn vooral handig bij de wat grotere VI's. Als een VI niet doet wat verwacht wordt, is en blijft de eerste stap: CONTROLEER HET DIAGRAM.

Als de fout op deze manier niet snel gevonden wordt, is Execution Highlighting (het lampje) een handig hulpmiddel. Het lampje laat de gegevens zien die over de draden van de VI gaan. Hiermee is ook goed zichtbaar welke onderdelen van de VI eerst worden uitgevoerd.

Een tweede veelgebruikte methode is het zogenaamde "Single Steppen" waarmee een Block Diagram stap voor stap kan worden uitgevoerd. Start een VI door op één van de single-step knoppen te klikken (in plaats van de Run knop), om single-steppen te beginnen. Je kunt ook de VI onderbreken door een Breakpoint te zetten of door op de Pause knop te klikken.

Gebruik de Probe tool uit het Tools Palette om de tussenliggende waarden in een VI te controleren, vooral als de VI twijfelachtige of onverwachte resultaten voortbrengt.

Tips & Hotkeys

Als men bezig is een VI te maken komen de volgende tips van pas (oefen ze even, de meeste zijn echt handig):

- Een kopie te maken van een object kan snel worden gedaan door het object met het Positioning Tool naar de plek voor de kopie te slepen, terwijl je de Control toets ingedrukt houdt.
- Houd de Shift toets ingedrukt om een object alleen te verplaatsen in de horizontale of verticale richting. De eerste richting waarin de muis wordt bewogen nadat de Shift toets is ingedrukt bepaalt of het object verticaal of horizontaal verplaatst moet worden.
- Gebruik de hotkeys om de menu-opties te vervangen; speciaal Ctrl-B om losse draden te wissen is handig.
- Een object op het Front Panel is terug te vinden in het Block Diagram venster door met rechts te klikken op het object en de optie "Find Terminal" te kiezen. In het Block Diagram venster kan men hetzelfde doen met de optie "Find Control".
- Met de spatiebalk wordt gewisseld tussen de twee belangrijkste elementen van de knoppenbalk: in het Front Panel venster zijn dat het pijltje en het handje. In het Block Diagram venster zijn dat het pijltje en het klosje.
- Een object op het Front Panel kan veranderd worden van een Control object in een Indicator object door met rechts te klikken op het object en de optie "Change to Indicator" te kiezen. Het omgekeerde kan ook, de optie heet dan "Change to Control".

5. Opdrachten

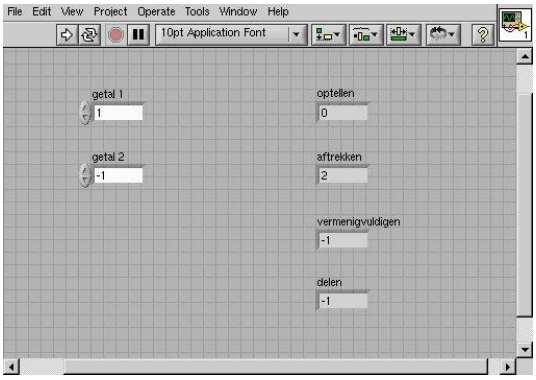
Dit appendix bevat opdrachten voor het onderdeel LabVIEW van het Practicum Natuurkunde 1. Op het college zal bekend gemaakt worden welke opdrachten gemaakt moeten worden.
De opdrachten kunnen tijdens de practicumdag gemaakt worden op PC's die daarvoor beschikbaar zijn.

1. De LabVIEW programmeeromgeving

Deze opdracht is vervallen! Zie de voorbeelden in hoofdstuk “Werken met LabVIEW”.

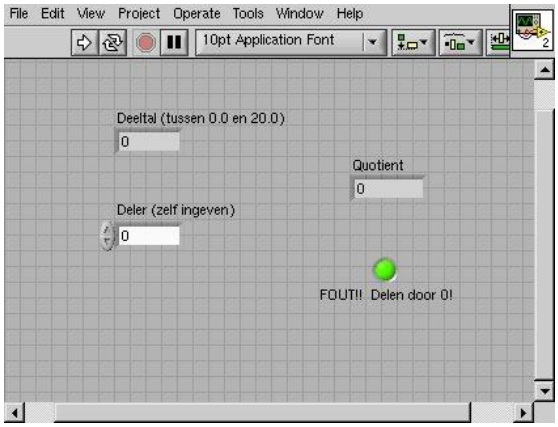
2. Eenvoudige rekenmachine

Bouw een VI die twee in te voeren getallen optelt, aftrekt, vermenigvuldigt en deelt en vervolgens het resultaat op het Front Panel weergeeft. Gebruik hiervoor het hieronder afgebeelde Front Panel.



3. Delen door nul

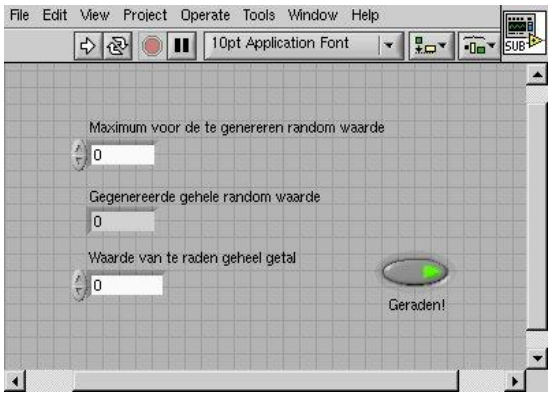
Bouw een VI die een willekeurige reëel getal (random number) tussen 0.00 en 10.00 genereert, het getal door een in te voeren getal deelt en tenslotte het resultaat op het Front Panel toont. Als het ingevoerde getal nul is, moet de VI een foutmelding geven (brandende LED indicator). De RANDOM NUMBER functie is te vinden in het **Functions→Programming→Numeric** subpalet en de EQUAL? functie in **Functions→Programming→Comparison**. Zie onderstaande figuur.



4. Loterij

Deel 1 opdracht 4:

Bouw een VI “Guess Number.vi” met het Front Panel van de afgebeelde figuur. Deze VI genereert een willekeurig geheel getal (random number) tussen 0 en een in te voeren getal Max number en vergelijkt dit met een ander in te voeren getal My number (My number moet een INTEGER zijn). Als de twee getallen gelijk zijn, gaat de indicator Guessed branden.
Sla deze VI op als sub-VI.

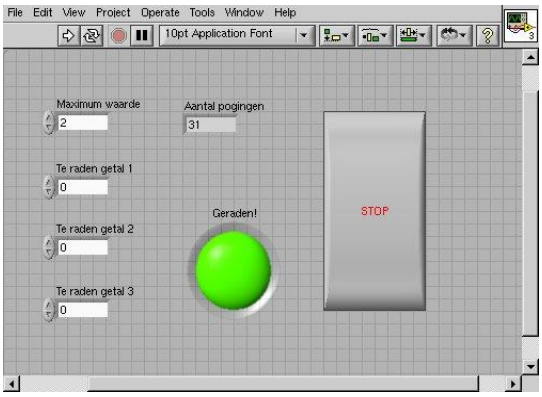


De RANDOM NUMBER functie is te vinden in het **Functions→Programming→Numeric** subpalet en de EQUAL? functie in **Functions→Programming→Comparison**.

AANWIJZING:
Gebruik de ROUND TO NEAREST functie uit het **Functions→Programming→Numeric** subpalet.

Deel 2 opdracht 4:

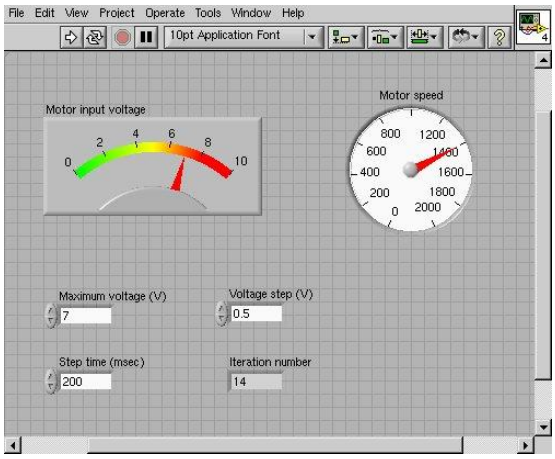
Bouw vervolgens een VI “Guess 3 Numbers.vi” met het Front Panel van onderstaande figuur. Deze VI gebruikt drie in te voeren getallen (alle drie INTEGER) en vergelijkt elk van deze getallen met een willekeurig getal, totdat alle drie de paren gelijk zijn. In feite laat deze VI zien hoeveel keer men moet proberen om drie juiste getallen in een loterij te raden. Gebruik de in deel 1 gemaakte VI “Guess Number.vi” als sub-VI.



AANWIJZING:
Gebruik de WHILE LOOP uit **Functions→Programming→Structures** en de COMPOUND ARITHMETIC functie uit **Functions→Programming→Numeric** in de AND mode.

5. Motor control simulation

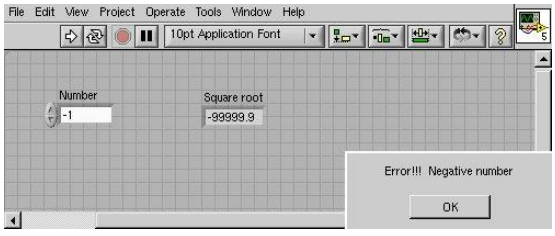
Bouw een VI die een DC motor control simuleert door trapsgewijs spanning te genereren, zie het Front Panel in onderstaande figuur. De spanning wordt opgebouwd in stapjes van Voltage step hoogte en Step time breedte. De indicator Iteration number geeft aan welke stap (iteration) op een gegeven moment wordt uitgevoerd en toont aan het eind van het programma het totale aantal iteraties tot de waarde Maximum voltage.
Neem aan dat een spanningsvariatie van 0.2 V overeenkomt met een verandering van de motorsnelheid (Motor speed) van 40 RPM (revolutions per minute).



AANWIJZING.
Gebruik de FOR LOOP uit het **Functions→Programming→Structures** subpalet.

6. Berekening vierkantswortel

Deel 1 opdracht 6:
Bouw een VI die de vierkantswortel van een in te voeren getal N berekent. Zie het Front Panel van onderstaande figuur. Bij N<0 geeft het programma een foutmelding “Error... Negative number” en geeft de Square Root Value indicator de waarde -99999.0 weer.

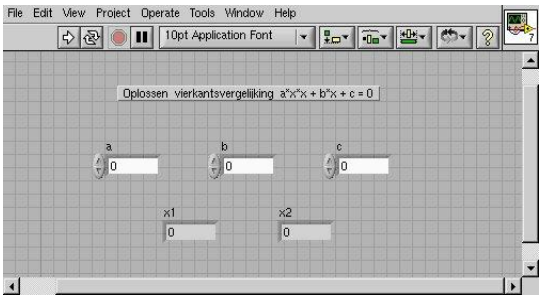


AANWIJZING:
Gebruik hiervoor de CASE structure uit het **Functions→Programming→Structures** subpalet en ONE BUTTON DIALOG uit **Functions→Programming→Dialog & User** interface.

Deel 2 opdracht 6:
Schrijf een andere variant van het programma zonder de CASE structure en de foutmelding, maar met gebruik van de SELECT functie uit het **Functions→Programming→Comparison** subpalet.

7. Oplossen kwadratische vergelijking

Bouw een VI die de oplossingen van een kwadratische vergelijking voor de variabele x berekent: $ax^2+bx+c=0$
Zie het Front Panel van onderstaande figuur. Bij a=0 geeft het programma een foutmelding “Error... Divide by Zero” en bij $b^2-4ac<0$ geeft het een foutmelding “Error... Negative Number”.



AANWIJZING:
Gebruik voor berekeningen FORMULA NODE uit het **Functions→Programming→Structures** subpalet.

8. Werken met arrays

Deze opdracht bestaat uit drie delen:

1. Bouw een VI die de volgorde van een array van 100 willekeurige getallen (random numbers) omkeert. Dus array[0] wordt array[99], array[1] wordt array[98], enzovoorts.
2. Bouw een VI die een array van 100 willekeurige getallen genereert en vervolgens een deel van dat array, bijvoorbeeld van index 10 tot index 50, op een digital indicator en als grafiek (wafeform graph) toont.
3. Bouw een VI die als invoer een 1D array met een even aantal elementen heeft, vervolgens paren van opeenvolgende elementen met elkaar vermenigvuldigt (beginnend met elementen 0 en 1, daarna 2 en 3, enzovoorts) en tenslotte het resulterend array van uitkomsten op het scherm toont. Bijvoorbeeld: een array met elementen 1, 23, 10, 5, 7 en 11 zal dus het array 23, 50, 77 als resultaat geven.

AANWIJZINGEN:
Gebruik bestaande functies om de berekeningen in deze opdracht uit te voeren!
(ad 2.) De waveform graph vind je in het **Controls→Graph** subpalet.
(ad 3.) Maak gebruik van de DECIMATE 1D ARRAY functie uit het **Functions→Programming→Array** subpalet.

9. Dobbelsteen gooien

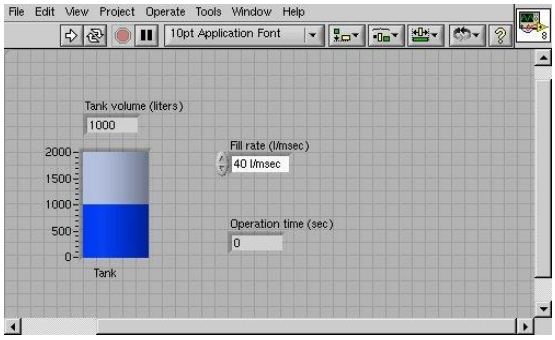
Bouw een VI die het gooien van een dobbelsteen simuleert (mogelijke waarden 1-6) en die het aantal malen dat elke waarde gegooid is, bijhoudt. Invoerwaarde is het aantal malen dat de dobbelsteen gegooid moet worden en uitvoerwaarden zijn de aantallen dat elk van de zes waarden gegooid zijn.

AANWIJZING:
Gebruik een SHIFT REGISTER om de waarden van de ene iteratie naar de volgende te bewaren. Een dobbelsteen kan als een array van zes elementen gezien worden.

10. Liquid level control

Bouw een VI met het Front Panel van onderstaande figuur, die de controle over de inhoud van een waterreservoir simuleert: het reservoir wordt gevuld of geleegd in gewenste stapjes. Het proces moet als volgt verlopen:

1. Vul het reservoir met water tot het niveau van 2000 liter met snelheid Fill Rate. De gebruiker moet Fill Rate kunnen kiezen uit: 5, 10, 20 of 40 liter/msec.
2. Wacht 5 seconden.
3. Voer water af tot het niveau van 1000 liter met dezelfde snelheid.
4. Wacht 4 seconden.
5. Maak de reservoir leeg met dezelfde snelheid.
6. Geef de totaaltijd van deze handelingen (Operation time).



AANWIJZING.
Gebruik voor Fill Rate een TEXT RING control uit het **Controls→Text Controls** subpalet in combinatie met een CASE STRUCTURE. Een tank indicator en een numeric indicator vind je in het **Controls→Programming→Numeric** subpalet. Maak gebruik van lokale variabelen.

11. Sinus en cosinus

Bouw een VI die een sinus- en een cosinusfunctie genereert en deze naar een Waveform Chart schrijft. Gebruik een WHILE LOOP en de SINE en COSINE functies uit het **Functions→Programming→Numeric** subpalet om twee 1D arrays te genereren (sin x en cos x) die de waarden van beide functies bevatten.
Het Front Panel van de VI vind je in onderstaande figuur, waarin Number of Cycles het gewenste aantal perioden van de sin x en de cos x functies en Number of Points het totale aantal x punten is.
Noem de VI “Sine/Cosine Chart.vi”.

Voer de VI uit. Experimenteer met verschillende instellingen van digitale controls en met de drie Update Mode opties voor de grafiek: Strip Chart, Scope Chart en Sweep Chart.

12. Tekstuitvoer

De bedoeling van deze VI is een output string te maken door stukjes tekst en de lopende waarde van een getal samen te voegen. Zo'n procedure komt vaak voor in bijvoorbeeld drivers voor meetinstrumenten, waar bepaalde stukjes tekst naar een meetinstrument moeten worden gestuurd of samen met gemeten waarden op het computerscherm moeten worden vertoond.
De VI die je hier maakt moet 10 willekeurige getallen genereren en voor elk getal boodschap “Random number in iteration X is Y” tonen, waarbij X het nummer van een FOR LOOP iteratie en Y het bijbehorende random number is. Zie het Front Panel van onderstaande figuur. Bovendien moeten alle 10 willekeurige getallen in een Array of Strings worden vertoond.

13. Temperatuur-analyse

Bouw een VI die de temperatuur continu (één keer per seconde) meet en deze in een grafiek in Scope mode vertoont.
Gebruik de VI "Thermometer.vi" uit de **Labview→Programmeren→SubVI** library map om de temperatuurmetingen te simuleren. Als de temperatuur boven of beneden de ingestelde grenzen (High limit resp. Low limit) komt, zet het programma een front panel LED aan. De grafiek moet de temperatuur alsmede de onder- en bovengrenzen tonen. De gebruiker moet in staat zijn om de grenzen vanuit het Front Panel in te stellen.
Noem de VI "Temperature Limit.vi".

14. Temperatuur-analyse (vervolg)

Wijzig de VI die je in de voorgaande opdracht had gebouwd zodanig dat de minimum- en maximumwaarden, die de temperatuur heeft bereikt, ook continu op het Front Panel worden vertoond. Noem de VI "Temp Limit (max/min).vi".

AANWIJZING:
Gebruik SHIFT REGISTERS en een ARRAY MAX & MIN functie (Array palet).

15. On-line gegevensverwerking

Bouw een VI die 50 temperatuurwaarden uitleest (één waarde per 0.25 seconde), deze waarden meteen op een Waveform Chart uitzet en de waarden vervolgens naar een bestand schrijft. Gebruik "Thermometer.vi" uit de **Labview→Programmeren→SubVI Library** map om temperatuurmetingen te simuleren. Vóór het schrijven naar het bestand moet het programma elke gemeten waarde naar een tekststring omzetten en vervolgens deze string aanvullen met een Tab, een "time stamp" string en een End of Line.
Noem de VI "Temperature Log.vi".

Het bestand moet er ongeveer als volgt uit zien (je kunt het bekijken met behulp van elke tekstverwerker):

78.9 11:34:48
79.0 11:34:49
79.5 11:34:50

AANWIJZING:

- Gebruik de TAB en de END OF LINE uit het String palet.
- Gebruik een GET DATE/TIME STRING functie uit het Timing palet.
- Gebruik CONCATENATE STRINGS om alle strings samen te voegen.
- Gebruik WRITE CHARACTERS TO FILE om de gegevens op te slaan.
- Het is het eenvoudigste om elke regel apart naar het bestand te schrijven, maar het is veel sneller en efficiënter om alle regels in één keer te schrijven. Daartoe kan je alle regels met hulp van Shift Registers en de CONCATENATE STRINGS functie in één grote regel zetten en die in één keer naar het bestand schrijven.

16. Master Control

In veel toepassingen is het nuttig om een soort “master control” te hebben, die de waarden van andere controls kan wijzigen. Stel dat je een eenvoudige controlebord wilt bouwen voor jouw “home stereo volumes”. De computer heeft vermoedelijk, op één of andere manier, een aansluiting op het “stereo volume control”. De VI die je hier gaat bouwen simuleert een controlepaneel met drie slide controls: Left channel, Right channel en Master. Zie onderstaande figuur. De twee kanalen — Left channel and Right channel — kunnen onafhankelijk ingesteld worden terwijl iedere verplaatsing van de Master slide een evenredige vergroting of vermindering van de geluidssterkte op beide kanalen moet teweegbrengen.
Dat wil zeggen, als de gebruiker de “master slide” verplaatst, moeten de twee andere “slides” hun posities vanzelf veranderen. Noem deze VI “Master and Slave.vi”.

AANWIJZING:
Het is voor deze opdracht noodzakelijk om SHIFT REGISTERS te gebruiken.