Bijlagenboek

Hardware Simulator

Een stap dichter richting totale “plant” simulatie.

Opdrachtgever:	**Imtech Vonk** **Modem 30** **7741 MJ Coevorden**
Plaats:	Coevorden
Datum:	4 Januari 2011
Auteur:	Mark Haanstra
Studentnummer:	1531233
Studierichting:	Industriële Automatisering
Hogeschool:	Hogeschool Utrecht
Bedrijfsbegeleider:	J. Töllner
1ste docent begeleider:	J. Schouten
2de docent begeleider:	E. van Akkeren

Afstudeerproject:

Uitbreiding hardware simulator

Bijlagenboek

Opdrachtgever:	Imtech Vonk Modem 30 7741 MJ, Coevorden
Bedrijfsbegeleider:	J. Töllner
Plaats:	Coevorden
Datum:	4 Januari 2011
Auteur:	Mark Haanstra
Studentnummer:	1531233
Afstudeerperiode:	30 augustus 2010 t/m 28 januari 2011
School:	Hogeschool Utrecht Oudenoord 700 3513 EX, Utrecht
Studierichting:	Industriële Automatisering
Inleverdatum:	4 januari 2011
1ste docent begeleider:	J. Schouten
2de docent begeleider:	E. Karsemeijer
Versie:	1

Samenvatting

Dit bijlagenboek is geschreven als ondersteuning voor de scriptie. In de scriptie staan de hoofdlijnen uitgewerkt en welke keuzes er zijn gemaakt. In de bijlagenboek wordt er dieper op de stof in gegaan, zal het programma worden uitgelegd en zal de code van het programma in staan.

De scriptie is voor degene die geïntresseerd is in het totale proces en de gemaakte keuze.
Het bijlagenboek is bestemd voor degene die geïnteresseerd is in het programma, de werking ervan, welke stappen het programma doorloopt en hoe de programma structuur eruit ziet.

Inhoudsopgave

Plan van Aanpak

Hardware Simulator

Uitbreiding Hardware Simulatie
Versie 1.2

Student:	Mark Haanstra
Docent begeleider:	Jos Schouten
Bedrijfsbegeleider:	Jürgen Töllner
Plaats:	Coevorden
Datum	05-10-2010

Inhoudsopgave

1. Achtergronden

1.1 De Naam

Uitbreiding Hardware Simulator.

1.2 De opdrachtgever

Het bedrijf heet Imtech Vonk bv. Mijn begeleider is dhr. Jürgen Töllner.

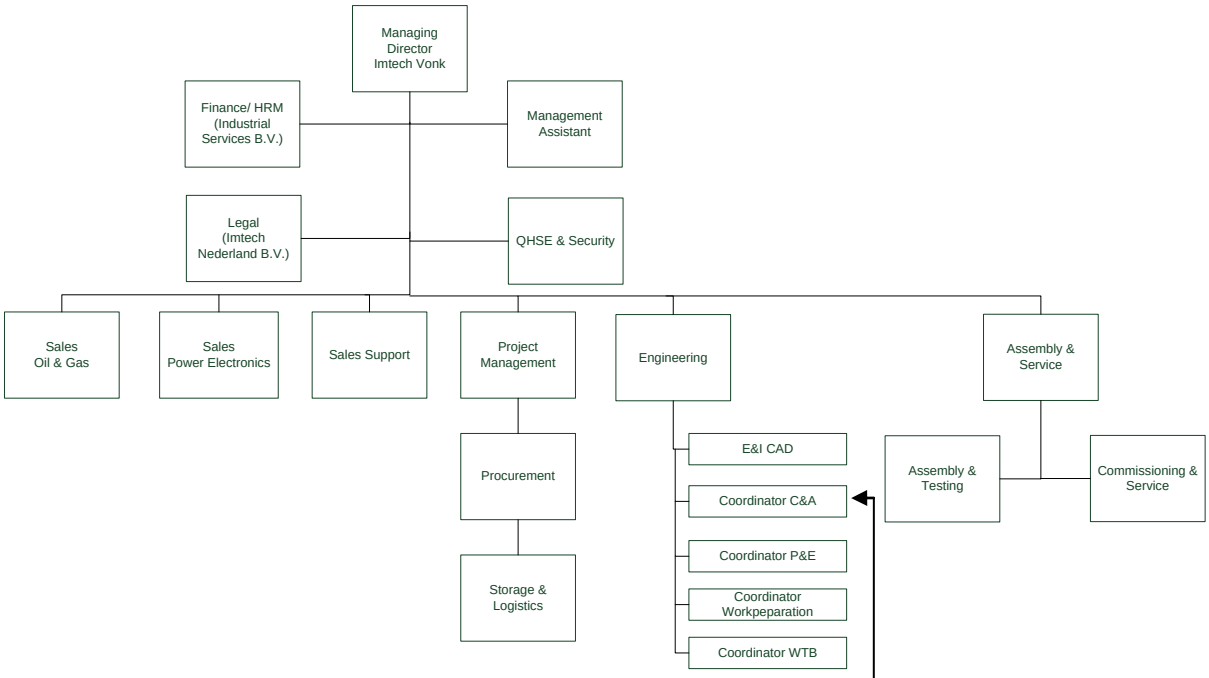
Het huidige Imtech Vonk is in 1937 opgericht door de dhr. Arjen Vonk. Arjen Vonk begon in 1939 met zijn werkzaamheden op het gebied van elektronica en installatiewerk voor onder andere voor de Nederlandse Aardolie Maatschappij (NAM). Zijn werkzaamheden groeiden langzaam uit naar het ontwerpen en fabriceren van elektrotechnische schakel componenten en controls-panels. Begin jaren 90 werd de toen nog Vonk Systems onderneming, een specialistisch onderdeel van Van Rietschoten & Houwens Noord-Oost (genaamd Imtech Projects Noord-Oost).
Vanaf 2002 opereert het bedrijf onder de naam Imtech-Vonk.

Imtech Noord-Oost heeft vestigingen in Leeuwarden, Groningen, Arnhem, Hengelo en heeft het hoofdkantoor in Coevorden. Bij Imtech Noord-Oost werken ca. 250 mensen in de utiliteit en industrie.

Momenteel werken circa 130 medewerkers bij Imtech Vonk. Imtech Vonk levert installaties aan de internationale olie- en gas industrie en doet daarnaast projecten in de vermogenselektronica.

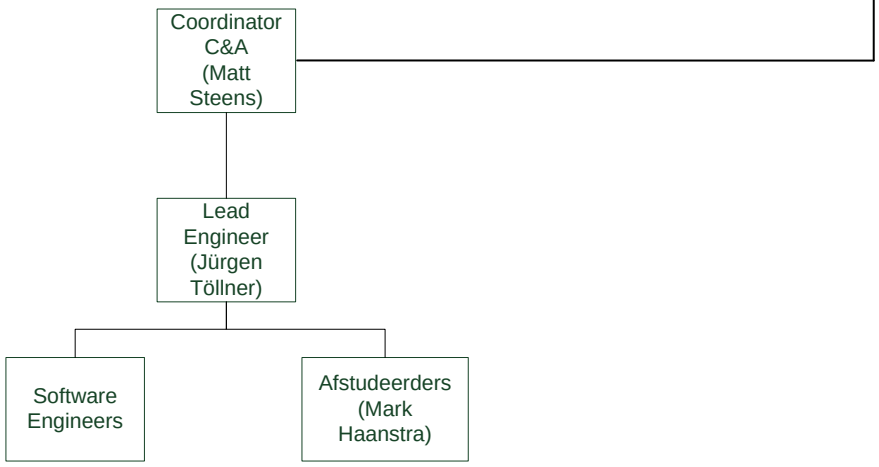
Hieronder (figuur 1.1 en figuur 1.2) staat twee organogrammen van het bedrijf. Één van het hele bedrijf Imtech Vonk en één van de afdeling Control en Automation (C&A). Hierin is de plaats van de afstudeerder in de organisatie ook duidelijk.

Organogram Imtech Vonk



figuur 1.1. organogram Imtech Vonk.

Organogram afdeling Control & Automation:



figuur 1.2. organogram afdeling C&A.

1.3 Opdrachtnemer

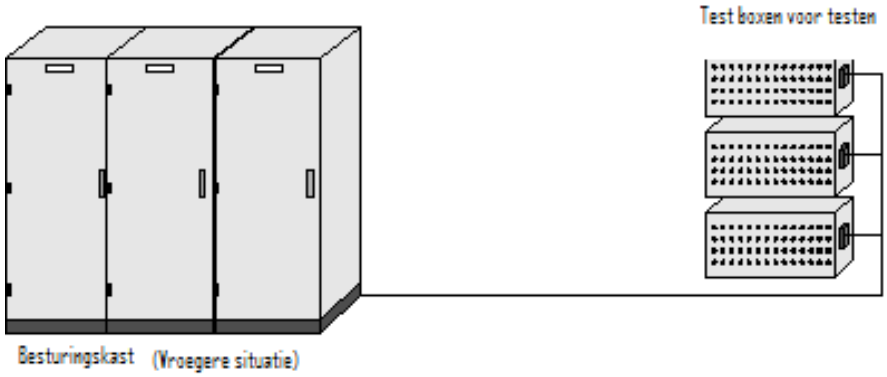
Mark Haanstra, student Industriële Automatisering Hogeschool Utrecht. 4de jaarstudent.

1.4 Algemene Informatie

1.4.1 Geschiedenis van hardware Simulator

Vroeger werd er bij Imtech Vonk getest met een testsysteem dat nogal complex en onoverzichtelijk was. Bij de vroege testopstelling werd er op de besturingskast kastjes aangesloten. Op deze kastjes konden schakelaars, potmeter, lampjes enz. worden geplaatst.
Hiermee kon je het proces simuleren.

Figuur 1.3 hieronder laat zien hoe het vroeger was.



figuur 1.3. vroegere situatie.

Omdat dit proces heel omslachtig is, is er een nieuw testsysteem ontworpen dat op een eenvoudigere en efficiëntere wijze diverse IO types kan simuleren.
Dit is de Hardware Simulator geworden.

Dit project is gedaan door een voormalige student aan de Hogeschool Utrecht.

2. Projectopdracht

Momenteel wordt er bij Imtech-Vonk getest met een hardware IO simulator om functionele testen uit te kunnen voeren op regelkasten. De simulator is opgebouwd met REMOTE IO, aangestuurd door een PC via MODBUS TCP.

Momenteel wordt er LabVIEW gebruikt om te simuleren. Hierdoor kunnen van simpele tot zeer complexe simulaties worden uitgevoerd.

2.1 Probleemstelling van het project

Momenteel zijn er een paar problemen/ tekortkomingen waardoor het werken met de hardware simulator erg complex en abstract is.
De problemen/ tekortkomingen zijn:

- Configuratie van IO is vrij complex en kan alleen door een ervaren engineer worden uitgevoerd.
- Huidige 4..20mA loops alleen passief.
- Niet makkelijk om NAMUR schakelaars te simuleren.
- Huidige IO te traag voor snelle proces simulatie.

2.2 Doelstelling van het project

De doelstelling van het project is het uitbreiden van de functionaliteit en het uitbreiden van de gebruiksvriendelijkheid van de hardware simulator.

Hierdoor worden de simulatie mogelijkheden beter waardoor de besturingskasten beter getest kunnen worden.
Door de gebruiksvriendelijkheid te vergroten kunnen niet alleen ervaren engineers maar ook monteurs gebruik maken van de hardware simulator.

Het doel van de opdracht is onderverdeeld in de onderdelen “Gebruiksvriendelijkheid” en “Uitbreiden functionaliteit”.
Per onderdeel zijn er subonderdelen die de uiteindelijke opdracht beschrijven.

2.2.1 Gebruiksvriendelijkheid

- *"User-friendly"IO configuratie software.*

  Dit houdt in dan de IO gegevens die in EPLAN staan, met 1 druk op de knop kunnen worden ingeladen in de IO simulator, de IO's zijn dan geconfigureerd en klaar voor gebruik. Momenteel moet elke tac/ IO nog handmatig worden ingevoerd en geconfigureerd. De tweede stap is dat met één druk op de knop in het simulatie programma er een lijst uit rolt waarop staat hoe de IO simulator moet worden aangesloten op de besturingskast, dit komt de gebruiksvriendelijkheid voor de monteurs ter goede.

  De lijst met IO TAGS zal door de tekenaar worden aangeleverd en zal op de pc van de hardware simulator te komen staan.

  Zie figuur 2.1.

2.2.2 Uitbreiden functionaliteit

- *MODBUS RTU/ ASCII simulatie (RS232/ RS485)*

  Sommige componenten worden aangestuurd via MODBUS RTU RS232/ RS485. Om de MODBUS te kunnen testen moet er eerst een component worden aangeschaft, aangesloten worden en dan kan er past getest worden. Dit is omslachtig en kost veel geld.
  De bedoeling is dat er een simulatie software/ programma komt die het component vervangt, via de simulatie kan dan toch de MODBUS worden getest. De software zal draaien op de pc waar ook de software van de hardware simulator op draait. De communicatie tussen de te testen besturingskast en de pc zal via de com-/seriële poort van de pc lopen.

  Zie figuur 2.1.

- *MODBUS TCP simulatie (ETHERNET)*

  In de toekomst komen er steeds meer componenten die werken op MODBUS TCP. Om in de toekomst niet met hetzelfde probleem te komen als bij MODBUS RTU/ ASCII (het testen van de verbinding) moet er een software programma komen om de MODBUS TCP te kunnen simuleren om zo deze te kunnen testen. Ook deze software zal draaien op de pc van de hardware simulator.
  De communicatie tussen de te testen besturingskast en de pc zal via de Ethernet poort van de pc lopen.

  Zie figuur 2.1.

- *OPC koppeling voor alle IO*

  Er moet een OPC koppeling komen die alle IO waarden opslaat in een database. Deze database moet door verschillende computers kunnen worden uitgelezen.

  De OPC koppeling wordt beheert en geprogrammeerd in LabVIEW.

  Zie figuur 1.4.

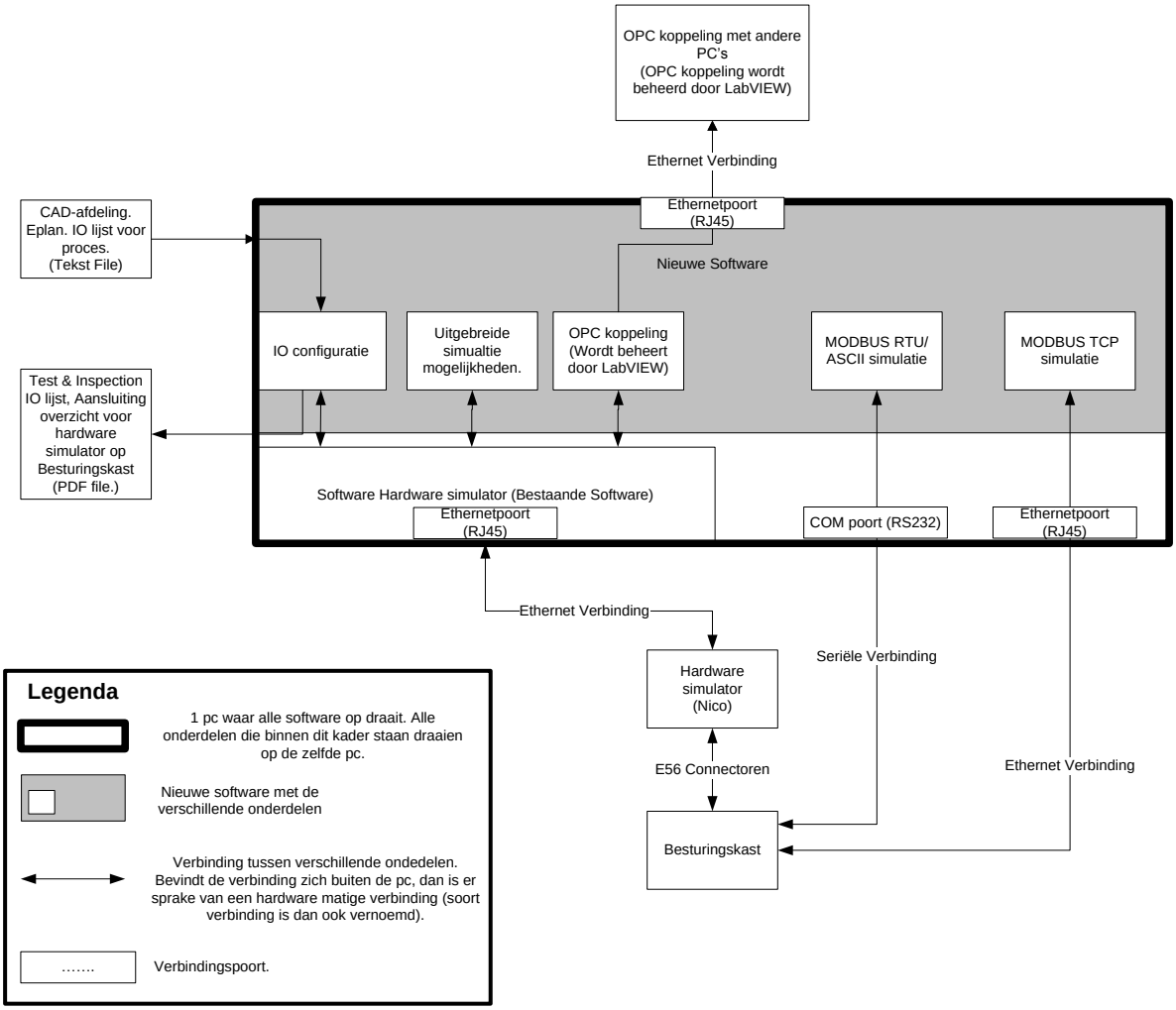
- *Uitbreiding van simulatie mogelijkheden*

  Het kunnen simuleren van kleppen, motoren, level transmittors, temperatuurgestuurde kleppen.

  Maar ook complexe processen, bijvoorbeeld een blok configureren waar je in- en uitgangen definieert en in het blok een formule kunt zetten die het proces aangeeft.

  Zie figuur 2.1.

Ter verduidelijking staat hieronder een schema hoe alle onderdelen van de opdracht onderling met elkaar zijn verbonden:



figuur 2.1. schema projectopdrachten.

3. Projectactiviteiten

3.1 Onderzoek Hardware simulator

Als eerste projectactiviteit moet de hardware simulator worden onderzocht. Deze hardware simulator is totaal nieuw voor mij, dus ik zal nog enige kennis van de simulator(is gewenst) moeten opdoen.

Het onderzoek wordt op de volgende wijze belopen:

- Het doornemen van de documenten die beschikbaar zijn van de hardware simulator.
- Een gesprek voeren met de desbetreffende persoon die over de hardware simulator gaat.

3.2 Inventaris hardware en software van de hardwaresimulator

Om de in's en out's van de hardware simulator te kennen is het verstandig om de hardware en software te gaan inventariseren. Zo krijg je een duidelijk beeld van de hardware simulator, waar bestaat de simulator uit, wat kan de hardware simulator allemaal en wat is de exacte werking van de hardware simulator.
Momenteel zijn er geen elektrische tekeningen aanwezig van de hardware simulator. Om een goed beeld te krijgen van de hardware wordt er aan Reverse Engineering. Dit houdt in de hardware te inventariseren en dit op tekening te zetten zodat er een goed beeld van de aanwezige hardware ontstaat.

3.3 Ontwikkelen van software

3.3.1 Kennis opdoen van LabVIEW

Momenteel wordt er gesimuleerd met het programma LabVIEW. Hierin moeten ook de software worden geschreven van de MODBUS simulaties, de uitgebreide simulaties.
Om dit goed te kunnen doen, moet er eerst "gespeeld" worden met LABVIEUW, het programma leren kennen en het onder de knie te krijgen, voorbeelden van simulaties namaken zodat bij het uiteindelijke simulatie software het programma goed eigen is om zo goede software te kunnen schrijven.

3.3.2 Kennis opdoen van MODBUS protocollen

Een onderdeel van de opdracht is, dat er software wordt geschreven voor het simuleren van een MODBUS interface. Omdat mijn kennis over MODBUS summier is, moet ik me gaan verdiepen in de theorie van MODBUS protocollen, de werking en het versturen van data ervan.

3.3.3 Kennis opdoen van OPC koppeling

Een onderdeel van de opdracht is dat er een OPC koppeling moet worden gemaakt voor de IO's van de hardware simulator. Deze IO's moeten in een database worden weggeschreven en deze database moet door verschillende computers kunnen worden uitgelezen.

Om deze opdracht goed te kunnen voltooien moet ik mij verdiepen in de theorie van een OPC koppeling.
Wat houdt een OPC koppeling in, hoe is een OPC koppeling opgebouwd enz..

3.3.4 Ontwerpen van software voor simulatie

Nadat er voldoende kennis van het programma LabVIEW is opgedaan en de kennis van de hardware simulator eigen is kan er worden begonnen met het schrijven van de software programma's.
De volgende software programma's moeten worden geschreven:

- MODBUS TCP/ ASCII simulatie.
- MODBUS RTU ETHERNET simulatie.
- Simulatie mogelijkheden zoals kleppen, pompen en complexe processen.
- OPC koppeling.

3.3.5 Het testen en installeren van de software

Als de software geschreven is moet deze worden geïnstalleerd in de hardware simulator en moet deze worden getest.
Om dit te kunnen doe moet de hardware simulator aanwezig en beschikbaar zijn. En er moet een besturingskast aanwezig zijn waar de hardware simulator op aangesloten kan worden.
Dit moet goed worden overlegd met de desbetreffende personen.

3.4 Uitbrengen van een scriptie

Aan het eind van het half jaar moet er een scriptie worden ingeleverd waar alle bevindingen, onderzoeksstappen, problemen waar tegen aan gelopen is en het eindresultaat in vermeldt staan.

Om een goed technisch scriptie te kunnen afleveren, moet er op tijd worden begonnen met het schrijven van een scriptie.

De scriptie zal deel uitmaken van de eindbeoordeling van het afstudeerproces en zal voor ongeveer 80% van het eindcijfer zijn.

3.5 Examenzitting

Als afsluitend geheel zal er een examenzitting zijn. Deze zitting duurt ongeveer 45 minuten en zal bestaan uit de volgende onderdelen.
Bij examenzitting zullen de 1ste en 2de docentbegeleider, lid van het College van Toezicht (CvT) en de bedrijfsbegeleider aanwezig zijn.

- Een presentatie.
- Een eventuele demo.
- Verdediging.
- Overleg/ beoordeling tussen de docenten begeleiders, de bedrijfsbegeleider en het lid van het College van Toezicht.

4. Projectgrenzen

De startdatum van het afstuderen is op maandag 30 augustus 2010 en de einddatum van het afstuderen is op 28 januari 2011.

Het maken van software in LabVIEW zal door de student op zijn ter beschikking gestelde computer gebeuren.
Het echter uitvoerig testen zal op de computer van dhr. Jürgen Tüllner gebeuren.

De volgende punten worden uitgevoerd tijdens het afstuderen:

- “User-friendly” IO configuratie.
- MODBUS RTU/ ASCII simulatie (RS232/ RS485).
- MODBUS TCP simulatie (Ethernet).
- OPC koppeling voor alle IO.
- Uitbreiding van simulatie mogelijkheden.

Deze punten zijn uitvoerig beschreven in Hoofdstuk 2.2 Doelstelling van de opdracht.

5. Producten

Ieder project heeft een einddoelstelling: een werkend product, een document of een presentatie. Zo wordt er een einddoel gesteld voor wat er ingeleverd moet worden.

Voor deze afstudeeropdracht geldt dit ook. Er zal op het eind een aantal producten moeten worden opgeleverd. Wat er precies ingeleverd en geproduceerd moet worden, wordt hieronder weergegeven:

5.1 Producten

- “User-friendly” IO configuratie software.
- MODBUS RTU/ ASCII simulatie (RS232/ RS485).
- MODBUS TCP simulatie (ETHERNET).
- OPC koppeling voor alle IO.
- Uitbreiding van simulatie mogelijkheden.

5.2 Documenten

- Plan van Aanpak.
- Een scriptie aan het eind van het afstuderen.
- Een presentatie voor het verdedigen van het scriptie.
- Beoordelingsformulier bedrijfsbegeleider.

6. Kwaliteitsbewaking

Hierbij wordt er gekeken naar de kwaliteit die er geleverd wordt aan de opdrachtgever. Dit kan door op verschillende punten terug te kijken naar de gestelde eisen van het product. Door deze controles regelmatig uit te voeren kan er in een vroeg stadium gezegd worden als het bepaalde onderdeel op schema loopt, of er kan gezegd worden dat het onderdeel niet realiseerbaar is (binnen de gegeven eisen).

6.1 Kwaliteit eindproduct

De kwaliteit van het eindproduct (scriptie) moet van een HBO niveau zijn. Dat houdt in dat het document zowel technisch als taalkundig van een HBO niveau moet zijn.

6.1.1 Controle kwaliteit eindproduct

Om er zeker van te zijn dat de kwaliteit van het eindproduct gewaarborgd is, wordt in de tussen tijd het concept van zowel het Plan van Aanpak als het concept van de Scriptie vroegtijdig gecontroleerd door zowel de docent begeleider als de bedrijfsbegeleider.
Dit zorgt ervoor dat er vanaf het begin een goede kwaliteit wordt neergezet.

De kwaliteit van alle documenten worden door zowel de docent begeleider als de bedrijfsbegeleider beoordeeld.

6.2 Normen van de Techniek

In het project zal er gebruik worden gemaakt van verschillende programma's die helpen de einddoelen te realiseren. Deze programma's zijn:

- MS word 2003 & MS word 2007: Hiermee worden de teksten verwerkt.
- LabVIEW: Hierin worden de softwareprogramma's geschreven.
- MS Visio 2007: Hiermee zullen simpele hiërarchieën en blokschema's gemaakt kunnen worden.
- MS Excel 2003 & MS Excel 2007: Hierin kunnen tabellen, grafieken e.d. gemaakt worden.
- MS Project 2003 & MS Project 2007: Hierin zal de planning worden gemaakt.

7. Projectorganisatie

De volgende personen zijn intern betrokken bij het project:

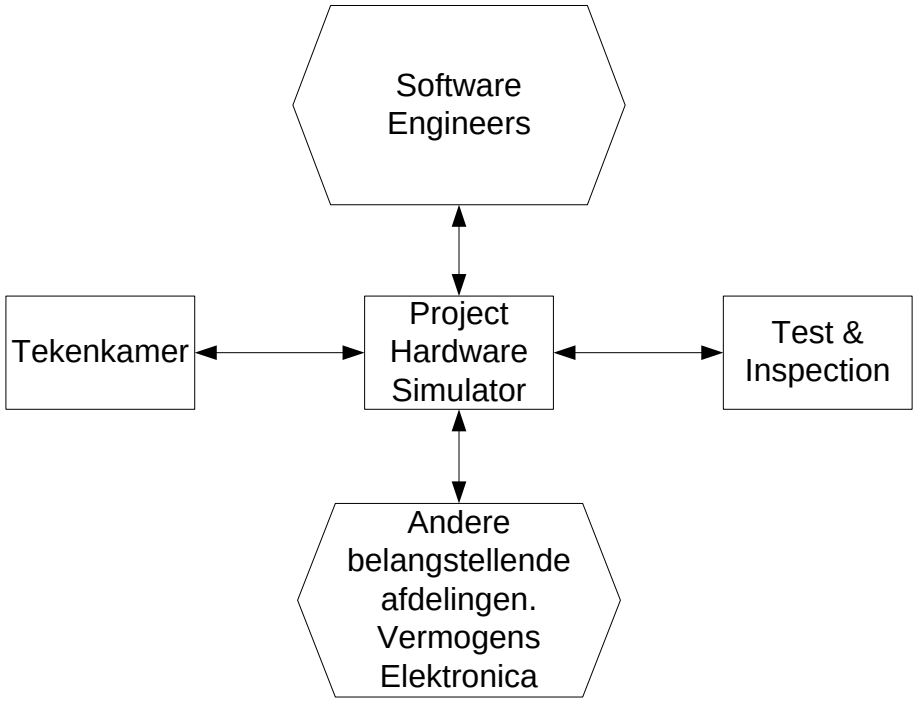
Naam:	Mark Haanstra
Functie:	Afstudeerder
E-mail:	mark.haanstra@student.hu.nl mark.haanstra@imtech.nl
Telefoon:	06-53433601

Naam:	Jürgen Töllner
Functie:	Bedrijfsbegeleider
E-mail:	jurgen.tollner@imtech.nl
Telefoon:	0524-599122

Naam:	Jos Schouten
Functie:	Docent begeleider
E-mail:	jos.schouten@hu.nl
Telefoon:	030-2388581

Met dit project zijn ook andere afdelingen betrokken.

In het schema hieronder staat schematisch weergegeven met welke afdelingen het project betrokken is:



figuur 7.1 Betrokken partijen overzicht.

8. Planning

Om het project goed te laten verlopen zal er een planning moeten worden gemaakt. Zo is er duidelijk wat er moet gebeuren en wanneer bepaalde onderdelen af moeten zijn.

In bijlage 1 is de planning te vinden.

9. Kosten en baten

In het project moet er rekening worden gehouden met de kosten en baten. Aan dit project zijn de volgende kosten en baten verbonden.

9.2 Kosten

De hardware simulator bestaat al, daar hoeven geen componenten voor worden gekocht.

9.3 Baten

Natuurlijk zitten er ook baten aan het project. Een project moet natuurlijk wel iets opleveren.

De volgende opbrengsten (baten) zijn aan het project verbonden:

- Een hardware simulator waarvan de gebruiksvriendelijkheid en functionaliteit van zijn verhoogd.
- Door het project voldoende af te sluiten en daarmee 30 ECT's te verkrijgen.

10. Risico's

Ook aan dit project zijn risico's verbonden die ervoor kunnen zorgen dat het project niet op tijd afgerond is of niet van voldoende kwaliteit is. Onder andere de volgende factoren spelen een rol:

10.1 Interne Risico's

- Een definitieve deadline.
- Onervarenheid met werken in projecten.
- Onvoldoende kennis/ niveau voor het project.
- Onvoldoende inbreng vanuit eindgebruiker/ opdrachtgever.
- Geen toezicht van hogere rang.

10.2 Externe Risico's

- Afhankelijk van andere projecten en dus ook die van derden.
- Onvoldoende medewerking vanuit de organisatie.
- Onduidelijke of tegenstrijdige projectgrenzen afbakening.
- Een andere doelstelling dan die geformuleerd is in het plan van aanpak.

In de bijlage is een spreadsheet te zien waarin een risicoanalyse opgemaakt is, waarin aan de risico's een cijfer is toegekend, waaruit weer een risicopercentage is.

10.3 Conclusie

Uitkomst van deze analyse is dat er een risicopercentage van 22% is. Bij meer dan 50% is het advies om af te zien van de opdracht of op bepaalde onderdelen/onderwerpen te herzien.

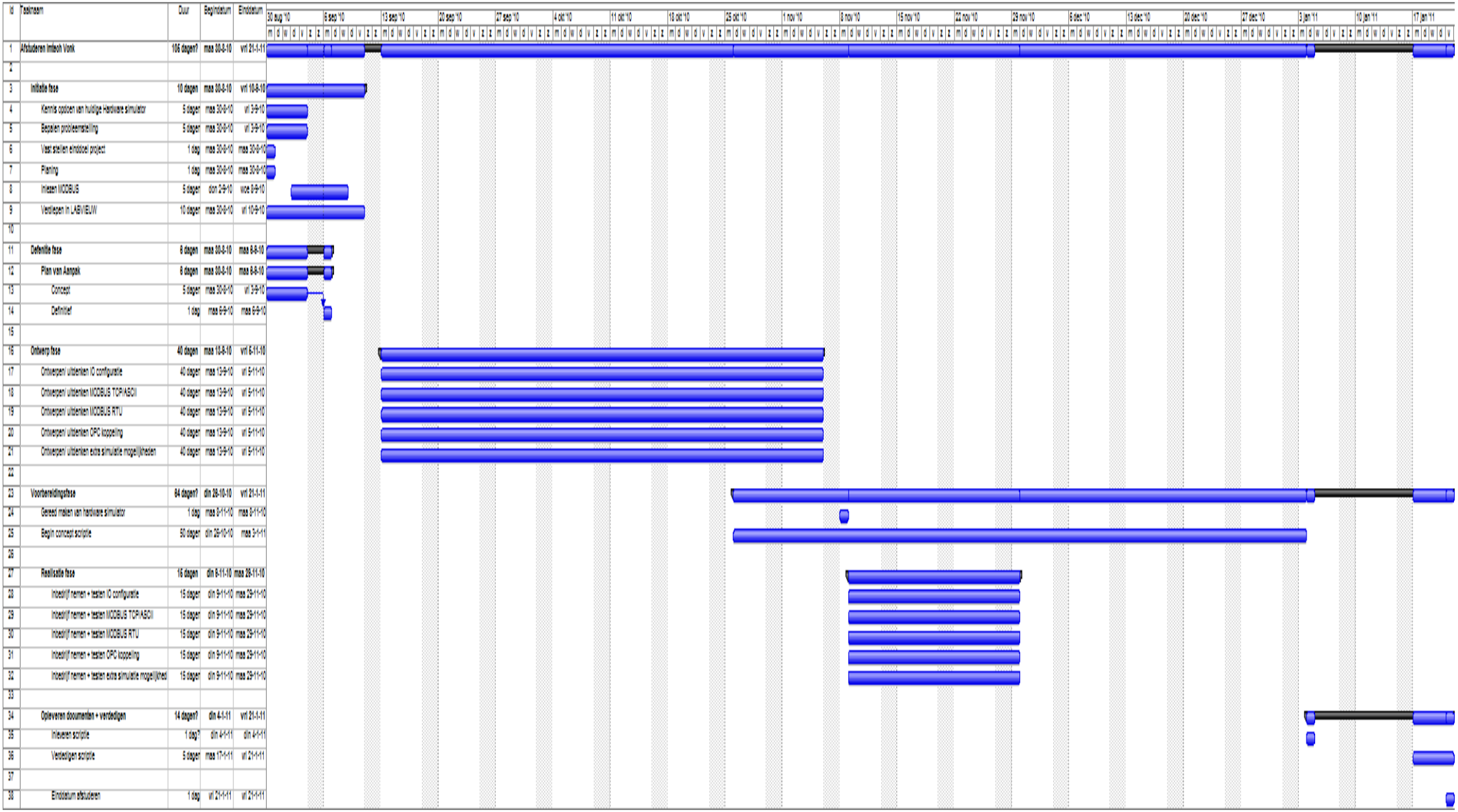
In de bijlage is te zien dat van de 22% 7.5% gaat zitten in de complexiteit van de opdracht.
De complexiteit van de opdracht is niet aan te passen, wel kan er op tijd aan de bel worden getrokken als er problemen qua complexiteit ontstaan zodat er op tijd naar een oplossing kan worden gezocht. Hierdoor wordt er zo minmogelijk tijd verspild waardoor het halen van het project in de afgesproken tijd geen gevaar loopt.

Aangezien het bepaalde percentage lager is dan 50% kan het project zijn doorgang vinden.

De risico analyse is te vinden in bijlage 2.

11. Bijlagen

11.1 Planning



Id	Taaknaam	Duur	Begindatum	Einddatum
1	Afstuderen Imtech Vonk	106 dagen?	ma 30-8-10	vri 21-1-11
2				
3	Initiële fase	10 dagen	ma 30-8-10	vri 10-9-10
4	Kennis opdoen van huidige Hardware simulator	5 dagen	ma 30-8-10	vri 3-9-10
5	Bepalen probleemstelling	5 dagen	ma 30-8-10	vri 3-9-10
6	Vast stellen einddoel project	1 dag	ma 30-8-10	ma 30-8-10
7	Planing	1 dag	ma 30-8-10	ma 30-8-10
8	Inlezen MODBUS	5 dagen	don 2-9-10	wo 8-9-10
9	Verdiepen in LABVIEW	10 dagen	ma 30-8-10	vri 10-9-10
10				
11	Definitie fase	6 dagen	ma 30-8-10	ma 6-9-10
12	Plan van Aanpak	6 dagen	ma 30-8-10	ma 6-9-10
13	Concept	5 dagen	ma 30-8-10	vri 3-9-10
14	Definitief	1 dag	ma 6-9-10	ma 6-9-10
15				
16	Ontwerp fase	40 dagen	ma 13-9-10	vri 5-11-10
17	Ontwerpen uitbreiden IO configuratie	40 dagen	ma 13-9-10	vri 5-11-10
18	Ontwerpen uitbreiden MODBUS TCP/ASCII	40 dagen	ma 13-9-10	vri 5-11-10
19	Ontwerpen uitbreiden MODBUS RTU	40 dagen	ma 13-9-10	vri 5-11-10
20	Ontwerpen uitbreiden OPC koppeling	40 dagen	ma 13-9-10	vri 5-11-10
21	Ontwerpen uitbreiden extra simulatie mogelijkheden	40 dagen	ma 13-9-10	vri 5-11-10
22				
23	Voorbereidingsfase	64 dagen?	din 28-10-10	vri 21-1-11
24	Gereed maken van hardware simulator	1 dag	ma 8-11-10	ma 8-11-10
25	Begin concept scriptie	50 dagen	din 26-10-10	ma 3-1-11
26				
27	Realisatie fase	15 dagen	din 9-11-10	ma 29-11-10
28	Inbedrijf nemen + testen IO configuratie	15 dagen	din 9-11-10	ma 29-11-10
29	Inbedrijf nemen + testen MODBUS TCP/ASCII	15 dagen	din 9-11-10	ma 29-11-10
30	Inbedrijf nemen + testen MODBUS RTU	15 dagen	din 9-11-10	ma 29-11-10
31	Inbedrijf nemen + testen OPC koppeling	15 dagen	din 9-11-10	ma 29-11-10
32	Inbedrijf nemen + testen extra simulatie mogelijkhed	15 dagen	din 9-11-10	ma 29-11-10
33				
34	Opleveren documenten + verdedigen	14 dagen?	din 4-1-11	vri 21-1-11
35	Inleveren scriptie	1 dag?	din 4-1-11	din 4-1-11
36	Verdedigen scriptie	5 dagen	ma 17-1-11	vri 21-1-11
37				
38	Einddatum afstuderen	1 dag	vri 21-1-11	vri 21-1-11

11.2 Risico analyse

Bij een risicopercentage > 50% dient het project niet in deze vorm te worden uitgevoerd.

Categorie	Risico	Waarde *	Factor **	Zwaarte **	Risicotot.
Tijdsfactor		↓maak keuze↓			
1	Geschatte looptijd van het project	3 - 6 maanden	1	4	4
2	Kent het project een definitieve deadline?	Ja	2	4	8
3	Is de tijd voldoende om het project te realiseren?	Voldoende	1	4	4
Complexiteit van het project		↓maak keuze↓			
4	Aantal functionele deelgebieden dat betrokken is	3+	3	4	12
5	Aantal functionele deelgebieden dat gebruik gaat maken van de resultaten	4	2	2	4
6	Gaat het om een aanpassing of een nieuw project?	Grote aanpassingen	2	5	10
7	In hoeverre zullen bestaande verantwoordelijkheden moeten wijzigen?	Niet	0	5	0
8	Zijn er andere projecten afhankelijk van dit project?	Nee	0	5	0
9	Wat zal de houding zijn van de gebruikers?	Positief	0	5	0
10	Zijn er deelprojecten, is de voortgang afhankelijk van de coordinatie hiertussen?	Enigszins	2	3	6
De projectgroep		↓maak keuze↓			
11	Welke medewerkers werken aan het project mee?	Voorn. interne	0	4	0
12	Wat is het geografische spreiding van de projecten?	1	0	2	0
13	Aantal projectleden dat op piektijden > 80% betrokken is	1-5	0	5	0
14	Verhouding materiedeskundigen tov projectdeskundigen	Goed	0	5	0
15	Nemen gebruikers deel aan de projectgroep?	In beperkte mate	3	3	9
De projectleiding		↓maak keuze↓			
16	Is de projectleiding materiedeskundig?	Zeer deskundig	0	3	0
17	Hoe deskundig is de projectleiding mbt de projectplanning?	Redelijk deskundig	2	3	6
18	Hoeveel ervaring heeft de projectleider met projecten als deze?	Veel ervaring	0	3	0
19	Hoe deskundig zijn de adviseurs op het te onderzoeken gebied?	Zeer deskundig	0	5	0
20	Hoe deskundig zijn de materiedeskundigen op het te onderzoeken gebied?	Zeer deskundig	0	5	0
21	Hoe betrokken zijn de verantwoordelijke lijnmanagers bij het project?	Sterk betrokken	0	5	0
22	Is de kans groot dat de samenstelling van de projectgroep wijzigt tijdens het project?	Kleine kans	0	5	0

23	Worden door de projectgroep standaardmethoden gebruikt?	Ja, een aantal	2	4	8

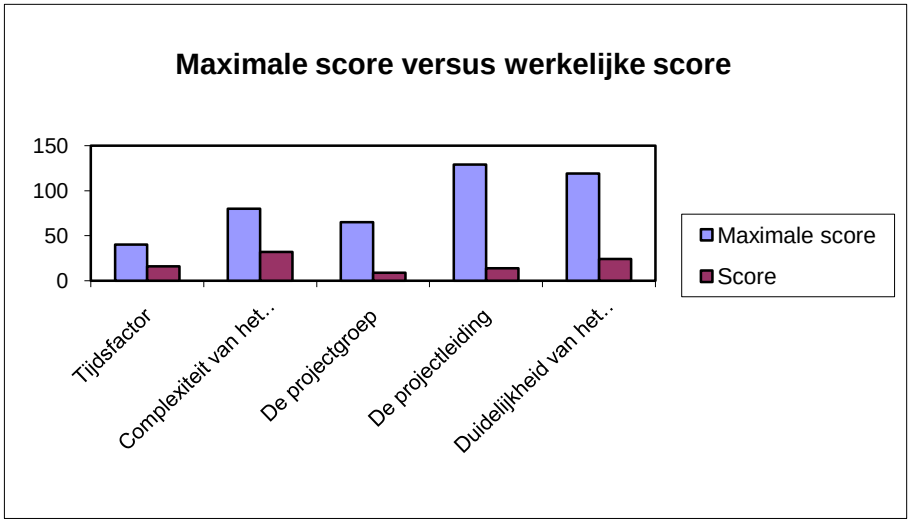
Vervolg risicoanalyse

Categorie	Risico	Waarde *	Factor **	Zwaarte **	Risicotot.
Duidelijkheid van het project		↓maak keuze↓			
24	Zijn probleem en doelstelling voldoende bekend bij alle projectleden?	Ja, iedereen	0	5	0
25	Is het onderzoeksgebied nauwkeurig vastgelegd?	Redelijk	2	5	10
26	Is er voldoende afbakening met andere projecten?	Voldoende	0	4	0
27	Is er voldoende tijd gepland voor afstemming en besluitvorming?	Voldoende	0	4	0
28	Zijn de randvoorwaarden duidelijk?	De meeste wel	1	4	4
29	Werken de randvoorwaarden beperkend genoeg?	Redelijk	2	5	10
			Totaal		**95**
			Risicopercentage ***		**21,94%**

** Waarde gekozen door projectleider.*
*** Hoogte factor en waarde staan vast.*
**** Risicopercentage is de totaalscore gedeeld door 433 (maximale score) maal 100.*

Aangezien het risicopercentage een totaalbeeld geeft, kan het zijn dat een bepaalde categorie wel voor een hoog risico zorgt. Hieronder een specificatie per categorie om eventuele verbeterpunten zichtbaar te maken.

Categorie (met maximale score versus werkelijke score)						%
Tijdsfactor		Maximaal	40	Score	16	3,70
Complexiteit van het project		Maximaal	80	Score	32	7,39
De projectgroep		Maximaal	65	Score	9	2,08
De projectleiding		Maximaal	129	Score	14	3,23
Duidelijkheid van het project		Maximaal	119	Score	24	5,54



Bijlage B MODBUS

1. Uitwerking MODBUS protocollen

1.1 Uitwerking MODBUS RTU protocol

Het MODBUS RTU staat voor Remote Terminal Unit en wordt veel gebruikt in de industrie. MODBUS RTU is opgebouwd uit hexadecimale getallen en heeft een bepaald frame formaat. Het frame formaat verschild tussen Master en Slave. Het Master frame formaat is als volgt opgebouwd: (zie tabel B.1).

Naam	Lengte	Functie
Start	3.5c idle	Tenminste 3.5 karaktertijden of stilte (MARK condition)
Slave adres	8 bits	Slave adres
Functie code	8 bits	Geeft aan van welke functie er gebruik wordt gemaakt.
Adres Hi	8 bits	Om begin adres tot 65535 te gebruiken
Adres Low	8 bits	Om begin adres tot 255 te gebruiken.
Plaatsen Hi	8 bits	Om aantal register/ coils tot 65535 te gebruiken
Plaatsen Low	8bits	Om aantal register/ coils tot 255 te gebruiken.
CRC check	16 bits	Error check, hiermee kan worden gekeken of het bericht in zijn geheel is over gekomen en of er geen data mist.
End	3.5c idle	Einde bericht, minstens 3.5 karaktertijden of stilte.

tabel B.1 MODBUS RTU Master frame formaat

Het Slave frame formaat is als volgt opgebouwd: (zie tabel B.2).

Naam	Lengte	Functie
Start	3.5c idle	Tenminste 3.5 karaktertijden of stilte (MARK condition)
Slave adres	8 bits	Slave adres
Functie code	8 bits	Geeft aan van welke functie er gebruik wordt gemaakt.
Aantal databits	8 bits	Geeft aan hoeveel data bits er zijn.
Data	n* 8 bits	Hierin komt de desbetreffende data te staan.
CRC check	16 bits	Error check, hiermee kan worden gekeken of het bericht in zijn geheel is over gekomen en of er geen data mist.
End	3.5c idle	Einde bericht, minstens 3.5 karaktertijden of stilte.

tabel B.2 MODBUS RTU Slave frame formaat.

De onderdelen van zowel de Master als de Slave staan hieronder verder uitgewerkt.

Slave adres:
Het MODBUS protocol werkt volgens het Master Slave principe. De Master zet een bericht op het netwerk gericht aan een bepaalde Slave. Alle Slaves krijgen dat bericht binnen en alleen de Slave waarvoor het bericht bestemd is mag het bericht in behandeling nemen of op reageren.

Het slave adres bestaat uit 8 bits. Van $0000\ 0000_2$ tot aan $1111\ 1111_2$ (00_{16} t/m FF_{16}) hiermee kunnen de adressen van 0_{10} t/m 255_{10} worden gegenereerd. Adres nul_{10} mag niet gebruikt worden, dit is namelijk het adres van de Master.

Functie code:

MODBUS maakt gebruik van functie codes. Met deze codes wordt aangegeven welke handeling er moet gebeuren. De *schuin* gedrukte functie codes worden uitvoerig behandeld omdat dit de meest gebruite functie codes zijn. Uit de volgende functie codes kan worden gekozen:

- *01 Read coil status:* Hiermee kunnen coils (binair) waardes worden uitgelezen zoals open/ dicht meldingen of motor aan/ uit,
- *02 Read input status:* Hiermee kunnen inputs (binair) waardes worden uitgelezen.
- *03 Read holding registers:* Hiermee kunnen analoge holding registers worden uitgelezen zoals temperatuur, klep stand enz,
- *04 Read input registers:* Hiermee kunnen analoge input registers worden uitgelezen.
- *05 Write single coil:* Hiermee kan de waarde “true” of “false” naar een coil worden geschreven,
- *06 Write single register:* Hiermee kan een analoge waarde naar een register worden geschreven,
- 07 Read Exception status:
- 08 Diagnostics:
- 09 Program 484:
- 10 Poll 484:
- 11 Fetch Communication Event Counter:
- 12 Fetch Communication Event Log:
- 13 Program controller:
- 14 Poll controller:
- *15 Write multiple coils:* Hiermee kunnen in meerdere coils de waarde “true”of “false worden geschreven,
- *16 Write multiple registers:* Hiermee kunnen in meerdere registers analoge waarden worden geschreven,
- 17 Report slave adres:
- 18 Program 884/M84:
- 19 Reset Comm. Link:
- 20 Read General Reference:
- 21 Write General Reference:
- 22 Mask Write 4X Register:
- 23 Read/Write 4X Register:

- 24 Read FIFO Queue:

Adres Hi/ Adres Low;

Bij MODBUS is het mogelijk om vanaf een bepaalde register/ coil adres beginnen met uitlezen. Het begin adres kun je invullen bij Adres Low. Maar omdat adres low maar bestaat uit 8 bits, 0000 0000_2 t/m 1111 1111_2 (00_{16} t/m FF_{16}) kun je maar een begin adres tussen 0_{10} en 255_{10} uitkiezen (256 mogelijkheden). Het maximale adres zou dan zijn FF_{16}, 255_{10}.

Om dit probleem op te lossen is het bereik Adres Hi er. Dit bereik bestaat ook uit 8 bits, 0000 0000_2 t/m 1111 1111_2, hier heb je ook 256 mogelijkheden.
Dus wil je beginnen bij adres 256_{10} dan krijg je 0100_{16} waarbij de eerste twee getallen Adres Hi is en de laatste twee getallen Adres Low is.

Het maximum begin adres is dan 255 *256 (adres Hi) + 255 (adres low) = 65535_{10} ofwel $FFFF_{16}$.

Plaatsen Hi/ Plaatse Low;

Bij MODBUS is het mogelijk om een x aantal register/ coils plaatsen uit te lezen. Het aantal plaatsen kun je invullen bij Plaatsen Low. Maar omdat Plaatsen Low maar bestaat uit 8 bits, 0000 0000_2 t/m 1111 1111_2 (00_{16} t/m FF_{16}) kun je maar een x aantal plekken tussen de 0_{10} en 255_{10} uitkiezen (256 mogelijkheden). Het maximale aantal plaatsen dat je zou kunnen uitlezen zou dan zijn FF_{16} ofwel 255_{10}.

Om dit probleem op te lossen is het bereik Plaats Hi er. Dit bereik bestaat ook uit 8 bits, 0000 0000_2 t/m 1111 1111_2, hier heb je ook 256 mogelijkheden.
Dus wil je beginnen bij x aantap plaatsen 256_{10} dan krijg je 0100_{16} waarbij de eerste twee getallen Plaatsen Hi is en de laatste twee getallen Plaats is.

Het maximum aantal plaatsen dat je kunt uitlezen is dan 255 *256 (Plaats Hi) + 255 (Plaats low) = 65535_{10} ofwel $FFFF_{16}$.

Aantal data bits:

Geeft aan hoeveel databits zich bevinden in het bericht. Dit wordt gebruikt om de data uit het bericht te kunnen filteren.

Data:

Hierin wordt de data geschreven, aan de hand van de functie code wordt bepaalde data weg geschreven.

De data kan verschillen aan de hand van de functie code. Worden er registers uitgelezen, dan wordt er gebruik gemaakt van data HI en data Low, net zoals bij de adressering en de register plaatsen. Worden er coils uitgelezen, dan wordt er gebruik gemaakt van een 8 bits getal.
Voorbeeld: er worden 8 coils uitgelezen, de waarden zijn:

Coil 1= true (1_2),
Coil 2 = false (0_2),

Coil 3 = true(1_2),
Coil 4 = false (0_2),
Coil 5 = false (0_2),
Coil 6 = true (1_2),
Coil 7 = false (0_2),
Coil 8 = true (1_2).

Deze waarden worden in een 8bits binair getal gezet waarbij de LSB Coil 1 is, en de MSB Coil 8 is. Het 8 bits getal komt er dan als volgt uit te zien: 10100101. Dit getal wordt dan weer geconverteerd naar een hexadecimaal getal.

Worden er maar 6 coils uitgelezen i.p.v. 8, dan worden de overige twee plekken die dan over zijn in het 8bits getal opgevuld met 0 (spaceholders).

Als er registers worden uitgelezen, of er naartoe wordt geschreven dan wordt er gebruik gemaakt van de data Hi, data Low notatie.

CRC check:

CRC check staat voor cyclic redundancy check. De CRC is een berekening die wordt gedaan over het bericht dat moet worden verzonden. Deze CRC wordt dan aan het einde van het bericht toe gevoegd.

De CRC is een uniek getal omdat de CRC aan de hand van het te verzenden bericht wordt berekend. Een ander bericht houdt dan ook in een andere CRC.

De Master berekend de CRC en voegt deze toe aan het bericht, de Slave ontvangt dit bericht en haalt de CRC er weer af. De Slave berekend zelf de CRC en vergelijkt deze met de CRC van het bericht, komen deze twee CRC's met elkaar overeen dan weet de Slave dat het bericht in zijn geheel is overgekomen. Want zou het bericht incompleet zijn, dan zou er een andere CRC bij de berekening uitkomen dan de CRC die bij het bericht zit.

Omgekeerd als de Slave een bericht stuurt naar de Master worden de zelfde stappen uitgevoerd maar dan omgekeerd.

Een CRC berekenen is erg lastig. De berekening gaat als volgt:

Als voorbeeld is er het volgende bericht: 0B0300000003. Het bericht wordt opgesplitst in paren van twee, dit wordt dan 0B 03 00 00 00 03_{16}.

Er is een xor constante met de waarde 1010000000000001_2.

Er wordt begonnen met een 16 bit getal met trues 1111111111111111_2,

0B wordt in 16 bits uitgeschreven: 0000000000001011_2,
Deze twee getallen gaan door de functie xor, uitkomst: 1111111111110100_2,
Dan worden de bits naar rechts geschoven via een shift functie, richting de Last Significante Bit (LSB).
Aan de linker kant wordt er een 0 bij het bericht toegevoegd, de Most Significante Bit (MSB).
Dit getal wordt opgeschreven.

Maar als het LSB een 1 wordt, wordt de shift functie 1 keer uitgevoerd, de uitkomst daarvan wordt gexord met de xor constante en die uitkomst wordt opgeschreven.

Dit proces wordt 8 keer uitgevoerd.

Daarna wordt het tweede paar getallen uitgeschreven in 16 bits, in dit geval:
03 wordt in 16 bits uitgeschreven: 0000000000000011_2,
Dit getal wordt met de uitkomst van laatste shift actie gexord en dit getal wordt opgeschreven.
Dan wordt de shift functie zoals bij getal 0B ook uitgevoerd.
Dit proces wordt ook 8 keer gehaald.

Het totale proces wordt gedaan voor alle twee paar getallen van het data bericht. De uitkomst van de laatste shift actie van het laatste getal is dan de CRC.

De uitgewerkte CRC check is te zien in onderstaand tabel B.3.

input hex string: **0B0300000003**

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		xor constant	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
byte#	Hex	Start with 16 trues	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
1	**0B**	0000000000001011	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**
		xor the 2 lines above	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0
		shift xor 1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0
		shift xor 2	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1
		shift xor 3	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		shift xor 4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
		shift xor 5	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		shift xor 6	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
		shift xor 7	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
		shift xor 8	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0
2	**03**	0000000000000011	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
		xor the 2 lines above	1	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1
		shift xor 1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1
		shift xor 2	1	1	0	1	0	0	0	1	1	1	1	1	1	1	1	0
		shift xor 3	0	1	1	0	1	0	0	0	1	1	1	1	1	1	1	1
		shift xor 4	1	0	0	1	0	1	0	0	0	1	1	1	1	1	1	0
		shift xor 5	0	1	0	0	1	0	1	0	0	0	1	1	1	1	1	1
		shift xor 6	1	0	0	0	0	1	0	1	0	0	0	1	1	1	1	0
		shift xor 7	0	1	0	0	0	0	1	0	1	0	0	0	1	1	1	1
		shift xor 8	1	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0
3	**00**	0000000000000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		xor the 2 lines above	1	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0
		shift xor 1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	1
		shift xor 2	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
		shift xor 3	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0
		shift xor 4	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0
		shift xor 5	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0
		shift xor 6	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1
		shift xor 7	1	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1

		shift xor 8	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0
4	00	0000000000000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		xor the 2 lines above	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0
		shift xor 1	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
		shift xor 2	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0
		shift xor 3	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0
		shift xor 4	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0
		shift xor 5	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0
		shift xor 6	0	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0
		shift xor 7	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	0
		shift xor 8	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0
5	00	0000000000000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		xor the 2 lines above	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0
		shift xor 1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1
		shift xor 2	1	0	1	0	0	0	0	0	0	0	1	1	1	1	0	1
		shift xor 3	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1
		shift xor 4	1	1	0	1	1	0	0	0	0	0	0	0	1	1	1	0
		shift xor 5	0	1	1	0	1	1	0	0	0	0	0	0	0	1	1	1
		shift xor 6	1	0	0	1	0	1	1	0	0	0	0	0	0	0	1	0
		shift xor 7	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	1
		shift xor 8	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1
6	03	0000000000000011	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
		xor the 2 lines above	1	0	0	0	0	1	0	1	1	0	0	0	0	0	1	0
		shift xor 1	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	1
		shift xor 2	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1
		shift xor 3	1	1	1	0	0	0	0	0	1	0	1	1	0	0	0	1
		shift xor 4	1	1	0	1	0	0	0	0	0	1	0	1	1	0	0	1
		shift xor 5	1	1	0	0	1	0	0	0	0	0	1	0	1	1	0	1
		shift xor 6	1	1	0	0	0	1	0	0	0	0	0	1	0	1	1	1
		shift xor 7	1	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0
		shift xor 8	0	1	1	0	0	0	0	1	0	0	0	0	0	1	0	1

crc
0561

tabel B.3. CRC berekening.

1.2 MODBUS ACII protocol

ASCII is een afkorting van American Standard Code for Information Interchance. Dit houdt in dat elk karakter wordt vertegenwoordigt door een getal, decimaal of hexadecimaal.

Een ASCII bericht begint met een : gevolgd door het data bericht. In plaats van een CRC kent het ASCII protocol een lrc berekening. Het bericht wordt afgesloten met CR (carriage return) en LF (new line feed).

LRC check:

LRC check staat voor Longitudinal Rendundancy Check. De LRC is een berekening die wordt gedaan over het bericht dat moet worden verzonden. Deze LRC wordt dan aan het einde van het bericht toe gevoegd.

De LRC is een uniek getal omdat de LRC aan de hand van het te verzenden bericht wordt berekend. Een ander bericht houdt dan ook in een andere LRC.

De Master berekend de LRC en voegt deze toe aan het bericht, de Slave ontvangt dit bericht en haalt de LRC er weer af. De Slave berekend zelf de LRC en vergelijkt deze met de LRC van het bericht, komen deze twee LRC's met elkaar overeen dan weet de Slave dat het bericht in zijn geheel is overgekomen. Want zou het bericht incompleet zijn, dan zou er een andere LRC bij de berekening uitkomen dan de LRC die bij het bericht zit.

Omgekeerd als de Slave een bericht stuurt naar de Master worden de zelfde stappen uitgevoerd maar dan omgekeerd.

Een LRC wordt als volgt berekend, een bericht voor Slave 17, functie code 3, eerste registers 40108, aantal registers 3 ziet er als volgt uit:

- 17 3 0 107 0 3_{10},
- Al deze getallen moeten bij elkaar worden opgeteld: 130_{10} ,
- Dit getal wordt negatief gemaakt via de twee compliment methode[1], -130_{10},
- Dit getal moet worden omgezet naar hexadecimaal, $7E_{16}$.
- $7E_{16}$ is dan de LRC.

Het bericht is dan : 11 03 00 6B 00 03 7E CR LF geworden. Dit moet worden omgezet naar ASCII formaat.

Het grote verschil tussen het RTU protocol en het ASCII protocol is tevens een groot nadeel namelijk, wil je het RTU protocol in ASCII protocol hebben, dan moet je alle cijfers van het RTU bericht splitsen en dan voor elk cijfer het decimale getal uit het ASCII tabel halen.

Het bericht : 1 1 0 3 0 0 6 B 0 0 0 3 7 E CR LF wordt omgezet naar ASCII formaat en wordt dan:
3A 3131 3033 3642 3030 3033 3745 0D 0A.

Hierin is het duidelijke verschil te zien tussen RTU formaat en ASCII formaat, het ASCII formaat is vele maten groter en onleesbaar geworden.
Het RTU bericht (11 03 00 6B 00 03 76 87_{16}) is 8 bytes groot, het ASCII formaat is 17 bytes groot.

Het ASCII protocol maakt gebruik van het RTU protocol qua functies en opbouw. Het vindt alleen een vertaal slag plaats naar het ASCII formaat.

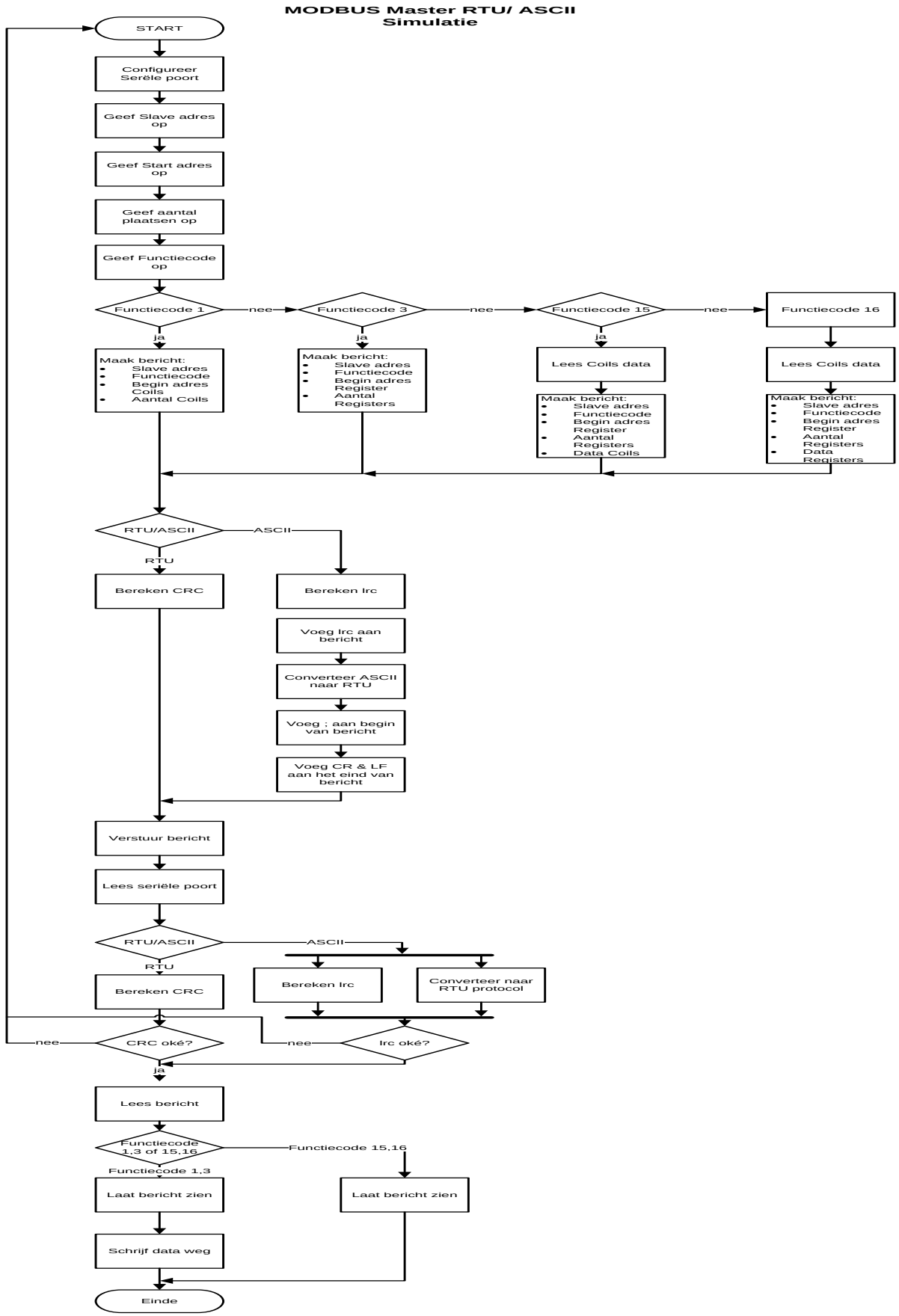
1.3 RTU & ASCII over TCP/ IP protocol.

TCP is een Transmission Control Protocol en IP is een Internet Protocol. Deze twee protocollen worden samen gebruikt als transport protocol voor het internet. Omdat er RTU/ ASCII data over TCP protocol wordt verstuurd wordt er alleen gebruik gemaakt van de lagen 1 (fysieke laag) en 2 (internetlaag).

2. Ontwerp MODBUS simulatie

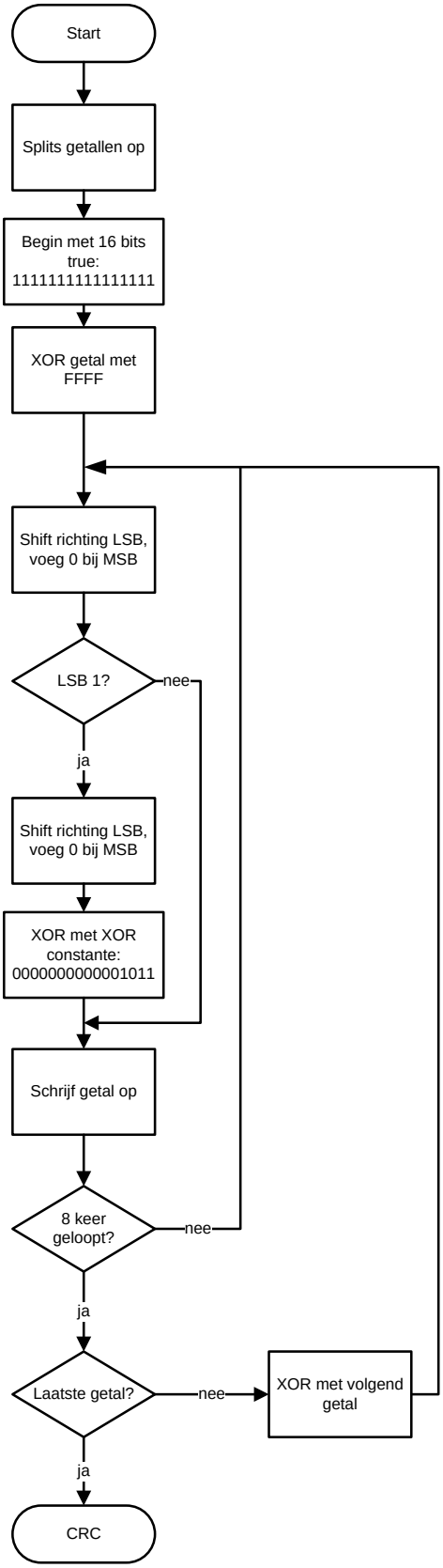
2.1 Ontwerp MODBUS Master Simulatie.

In figuur B.4 is te flowchart van de MODBUS RTU/ ASCII over RS232 Master simulatie te zien.
In figuur B.5 is de flowchart van de MODBUS RTU/ ASCII over RS232 Slave simulatie te zien.
In figuur B.6 is de flow



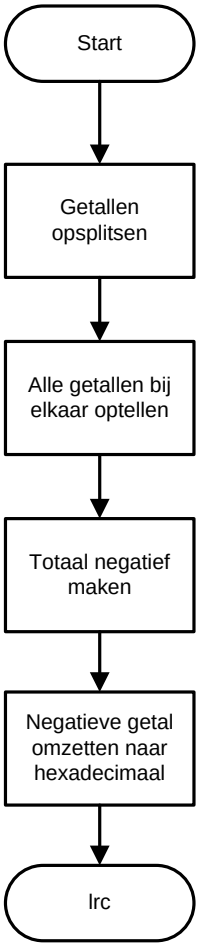
figuur B.4. Koninklijke weg MODBUS Master Simulatie.

Cyclic Redundancy Check (CRC)

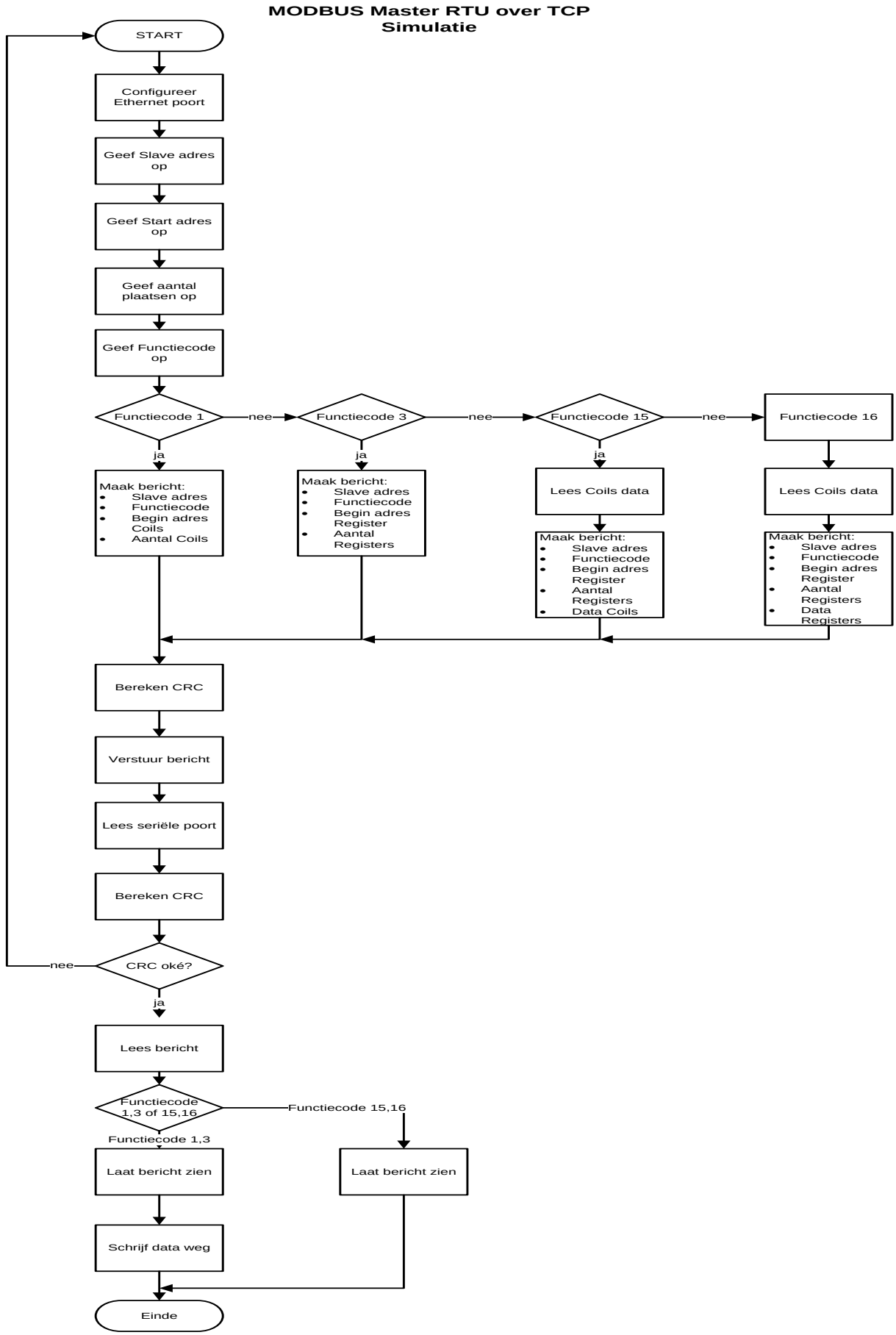


figuur B.5. Koninklijke weg CRC check.

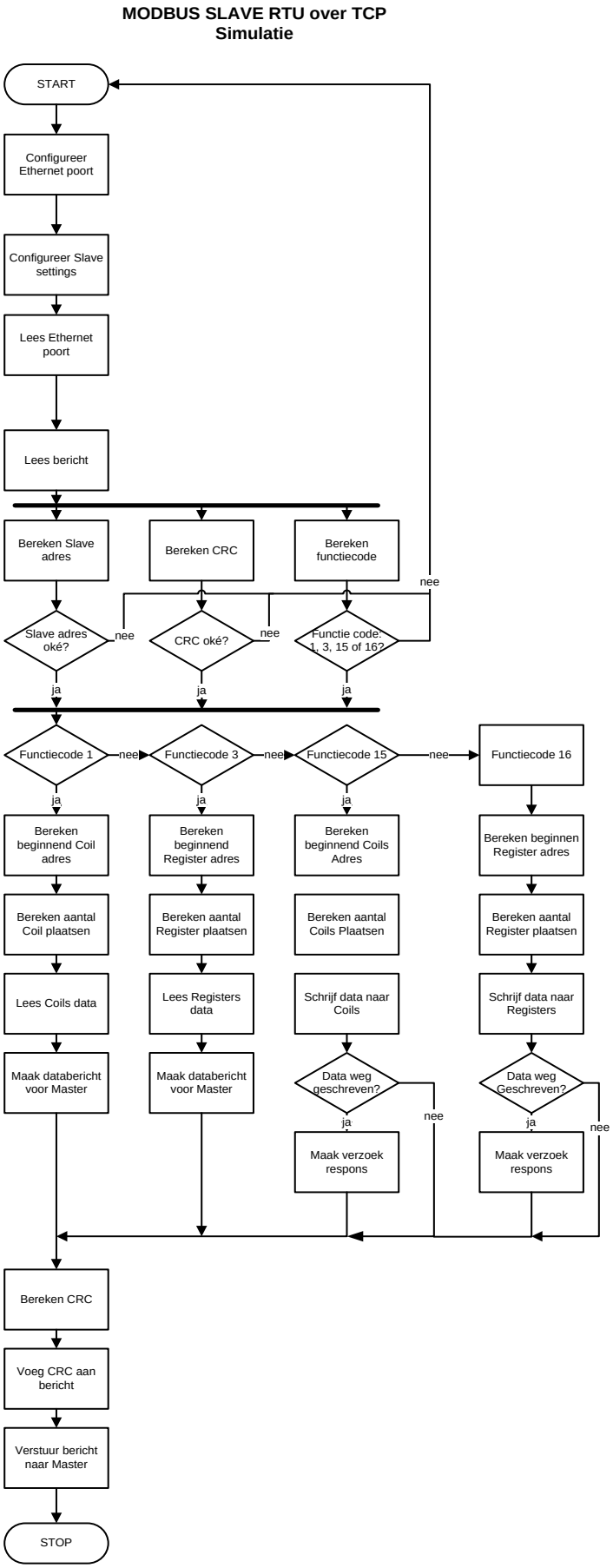
Longditudinal Rendundancy Check (lrc)



figuur B.6. Koninklijke weg lrc check.



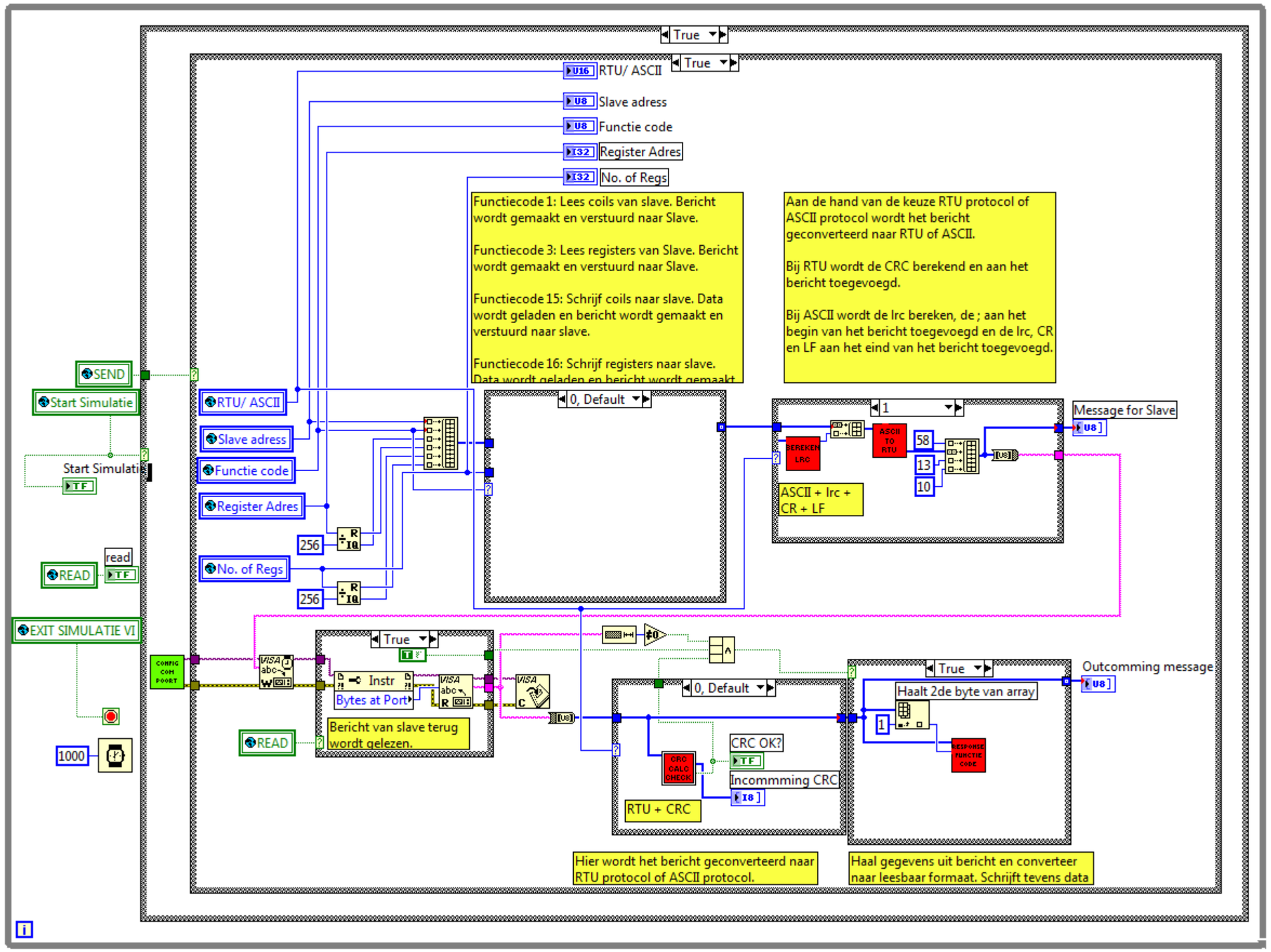
figuur B.7. MODBUS RTU over TCP Master



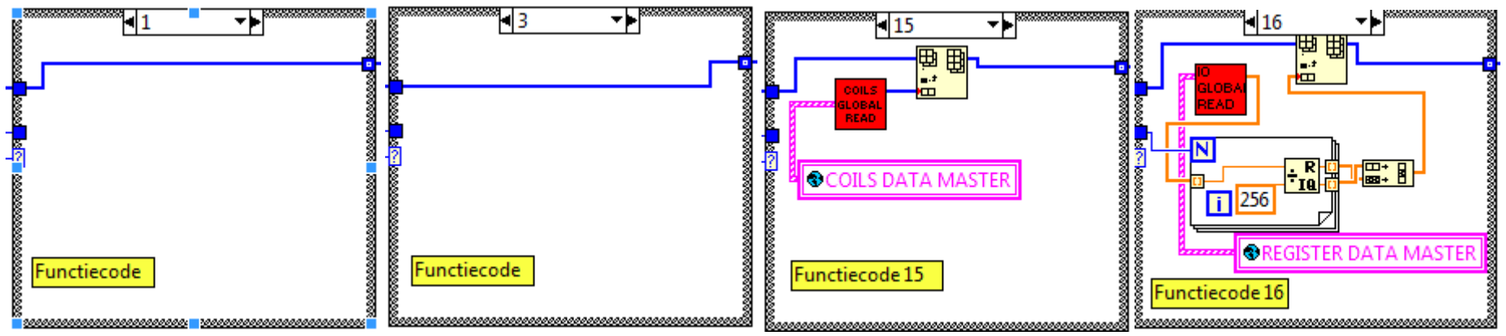
figuur B.8. MODBUS RTU over TCP Slave

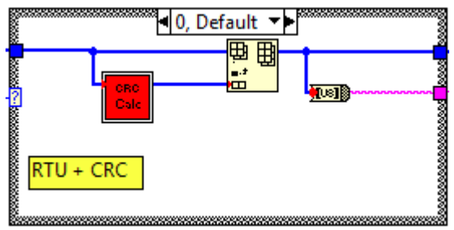
3. Beschrijving MODBUS simulaties

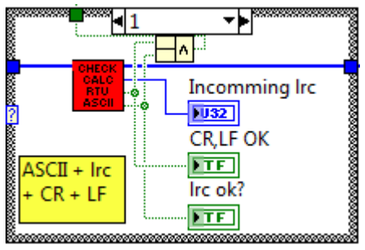
In dit hoofdstuk worden de MODBUS broncodes weergegeven.
In figuur B.9 is de broncode van de MODBUS RTU/ ASCII over RS232 Master te zien.
In figuur B.10 is de broncode van de MODBUS RTU/ ASCII over RS232 SLAVE te zien.
In figuur B.11 is de broncode van de MODBUS RTU over TCP MASTER te zien.
In figuur B.12 is de broncode van de MODBUS RTU over TCP SLAVE te zien.

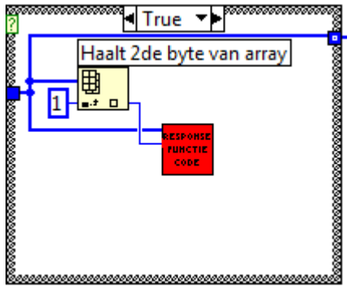


figuur B.9 MODBUS MODBUS RTU/ ASCII over RS232 MASTER.

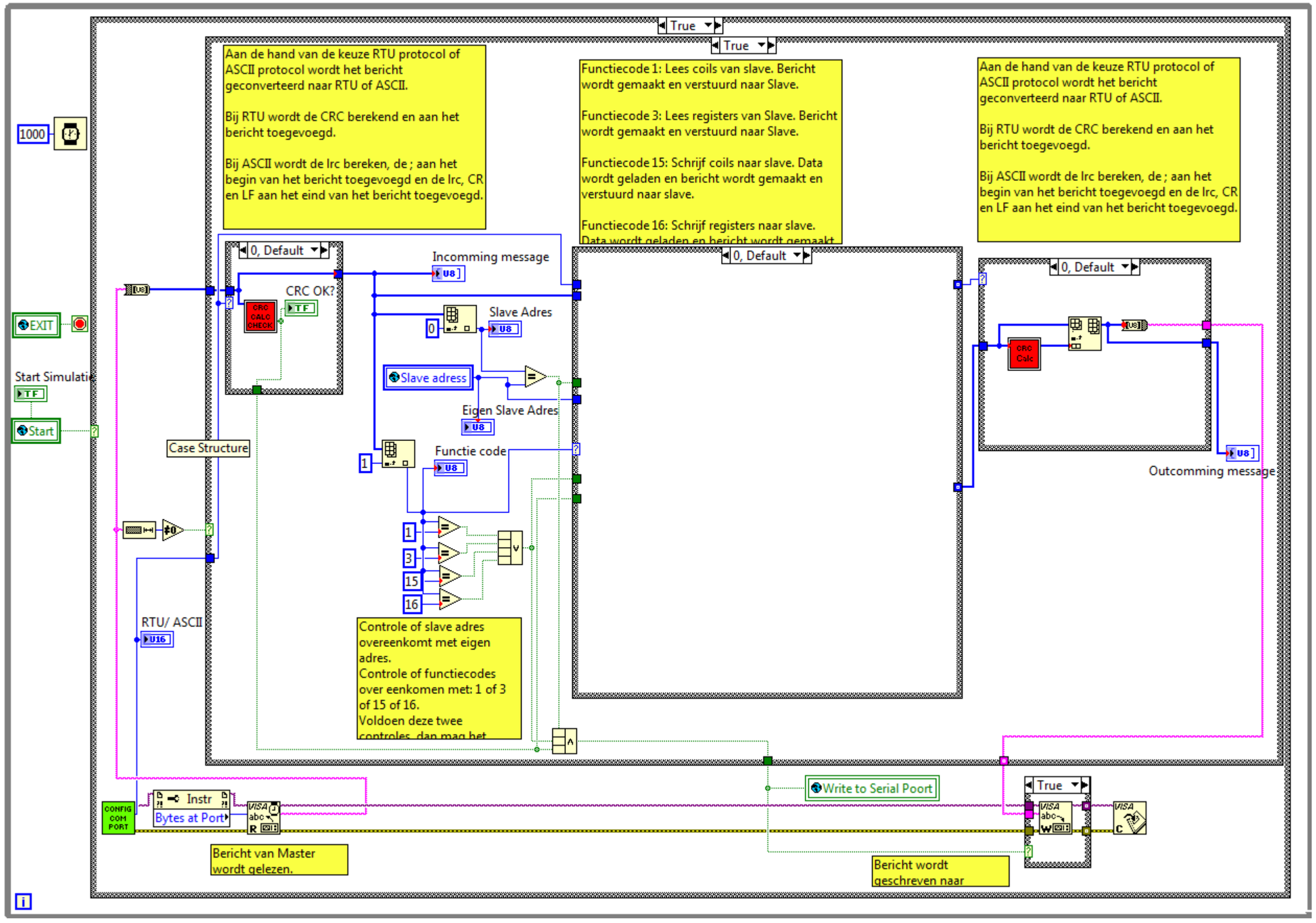




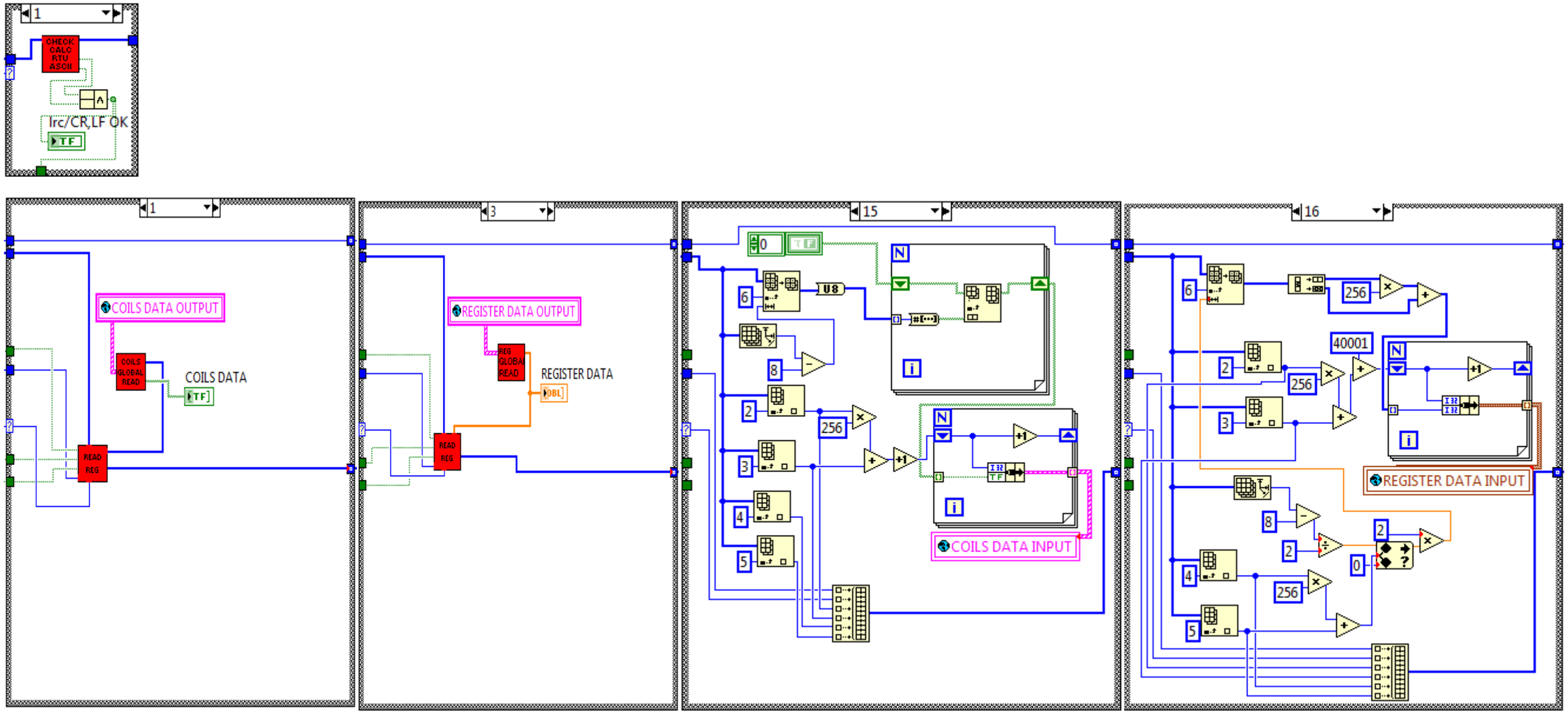


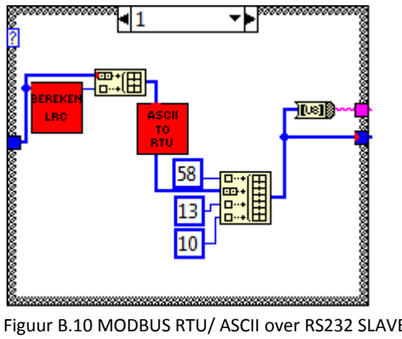


figuur B.9 MODBUS MODBUS RTU/ ASCII over RS232 MASTER (vervolg).

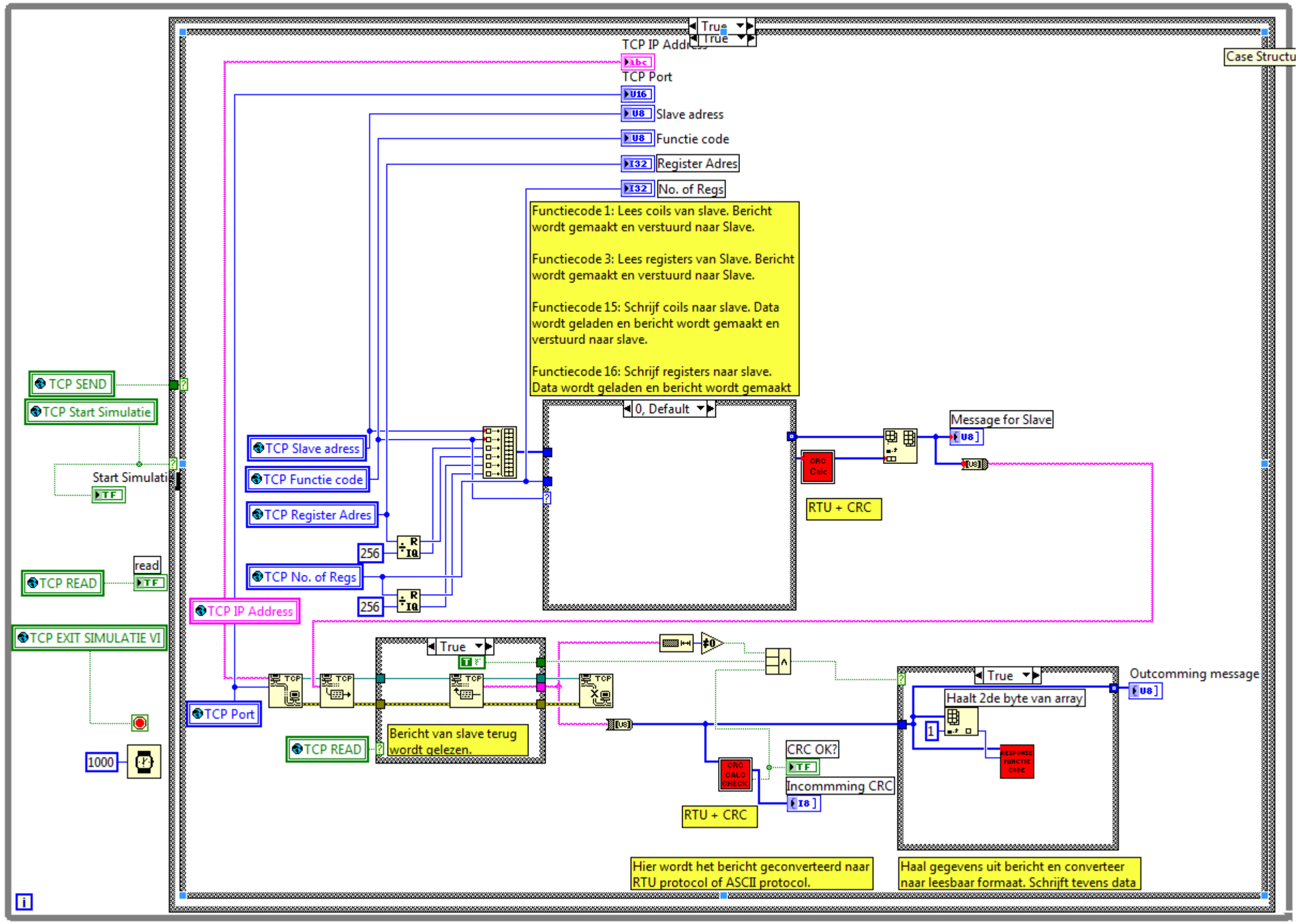


Figuur B.10 MODBUS RTU/ ASCII over RS232 SLAVE.

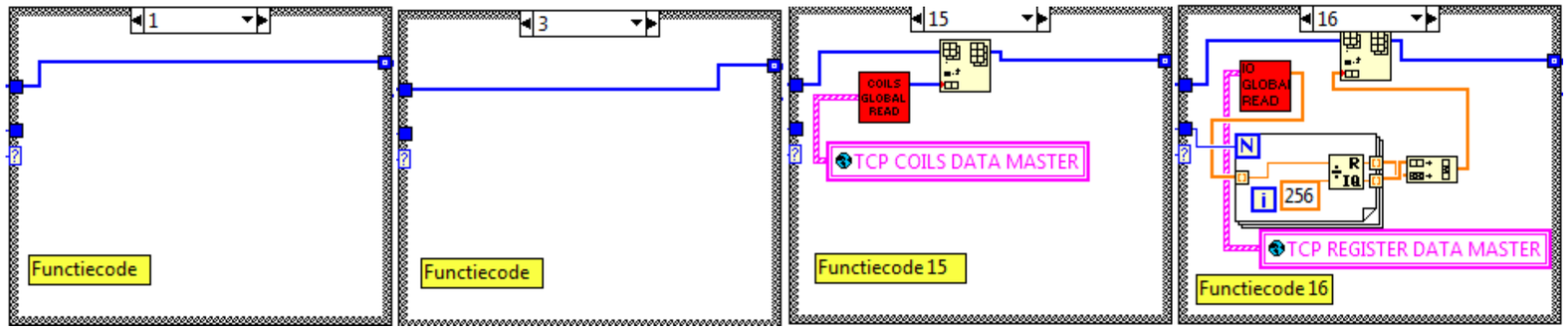


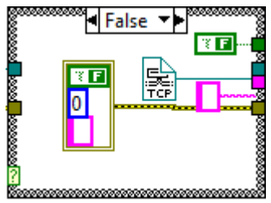


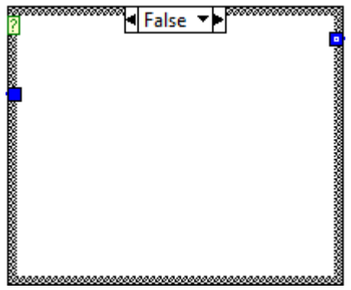
Figuur B.10 MODBUS RTU/ ASCII over RS232 SLAVE (vervolg).



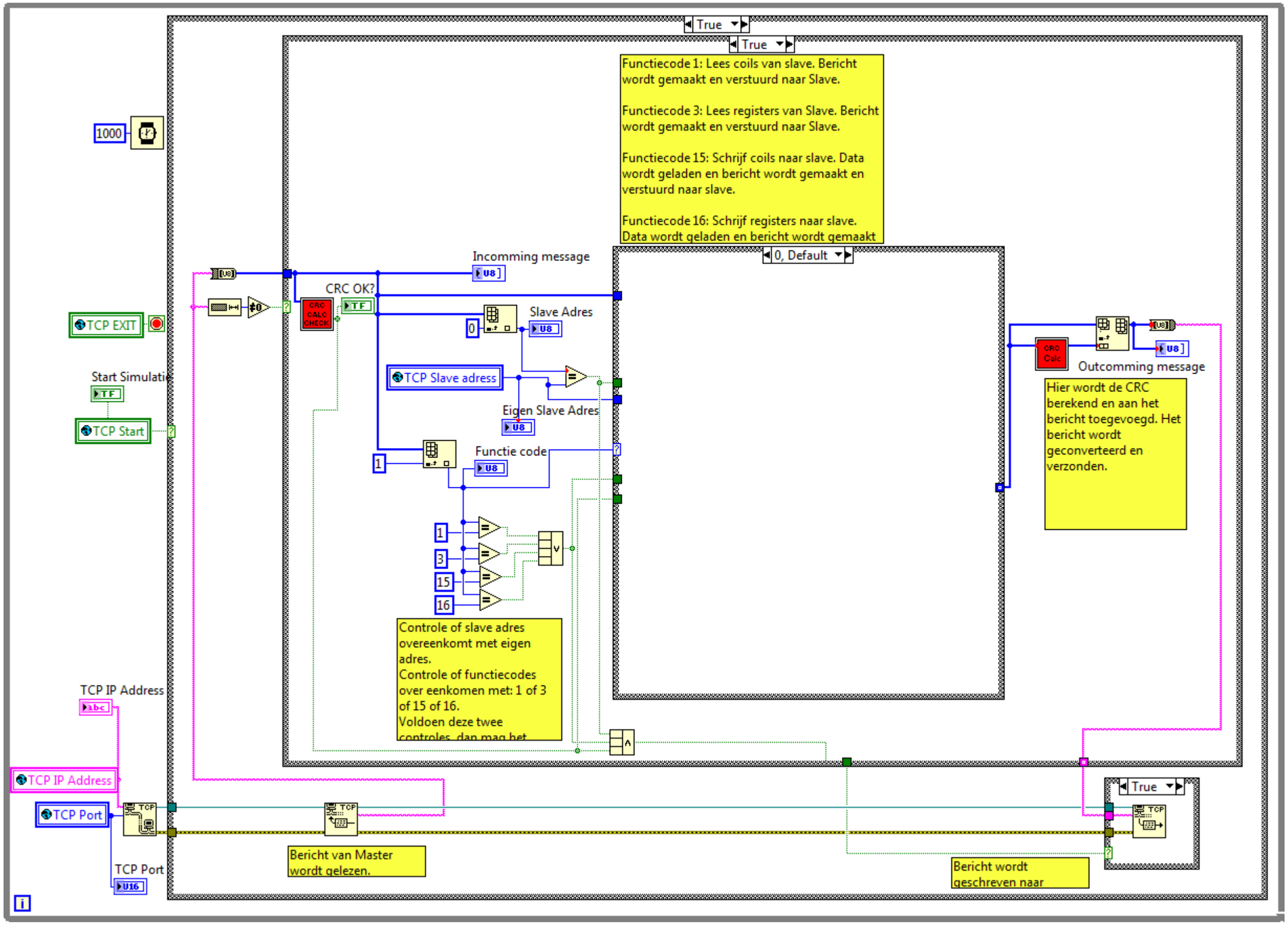
figuur B.11. MODBUS RTU over TCP MASTER



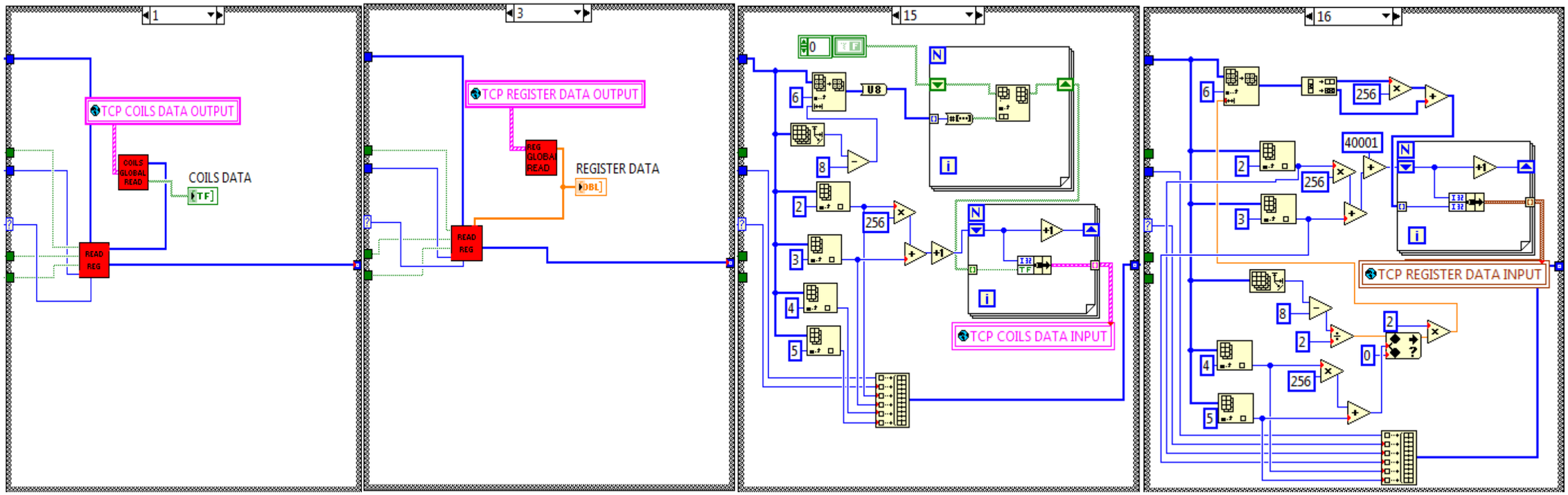


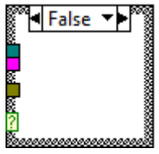


figuur B.11. MODBUS RTU over TCP MASTER (vervolg).



figuur B.12. MODBUS RTU over TCP SLAVE.

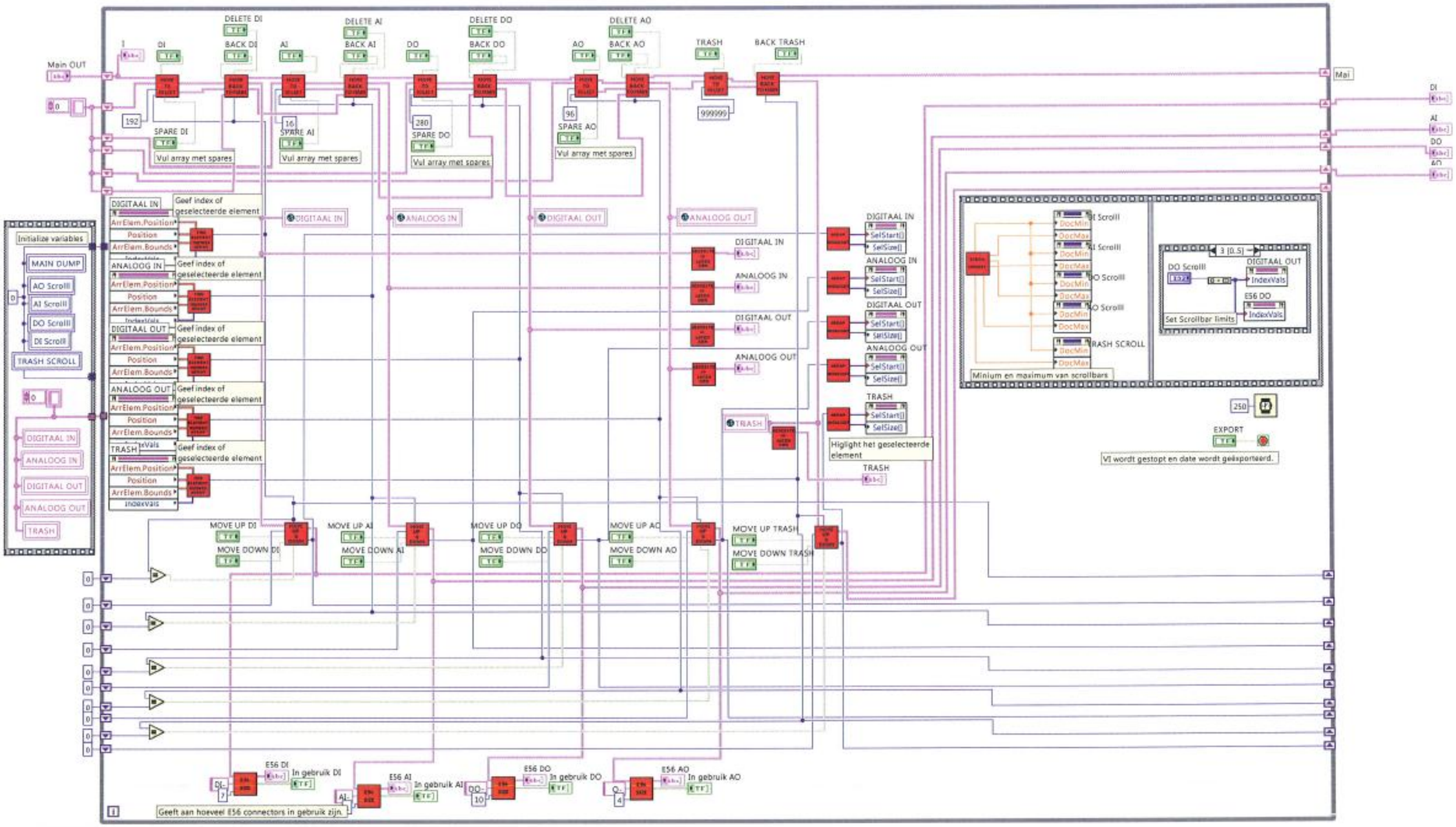




figuur B.12. MODBUS RTU over TCP SLAVE.

Bijlage C User-friendly IO configuratie

In figuur C.1 is de broncode van de User-friendly IO configuratie te zien.

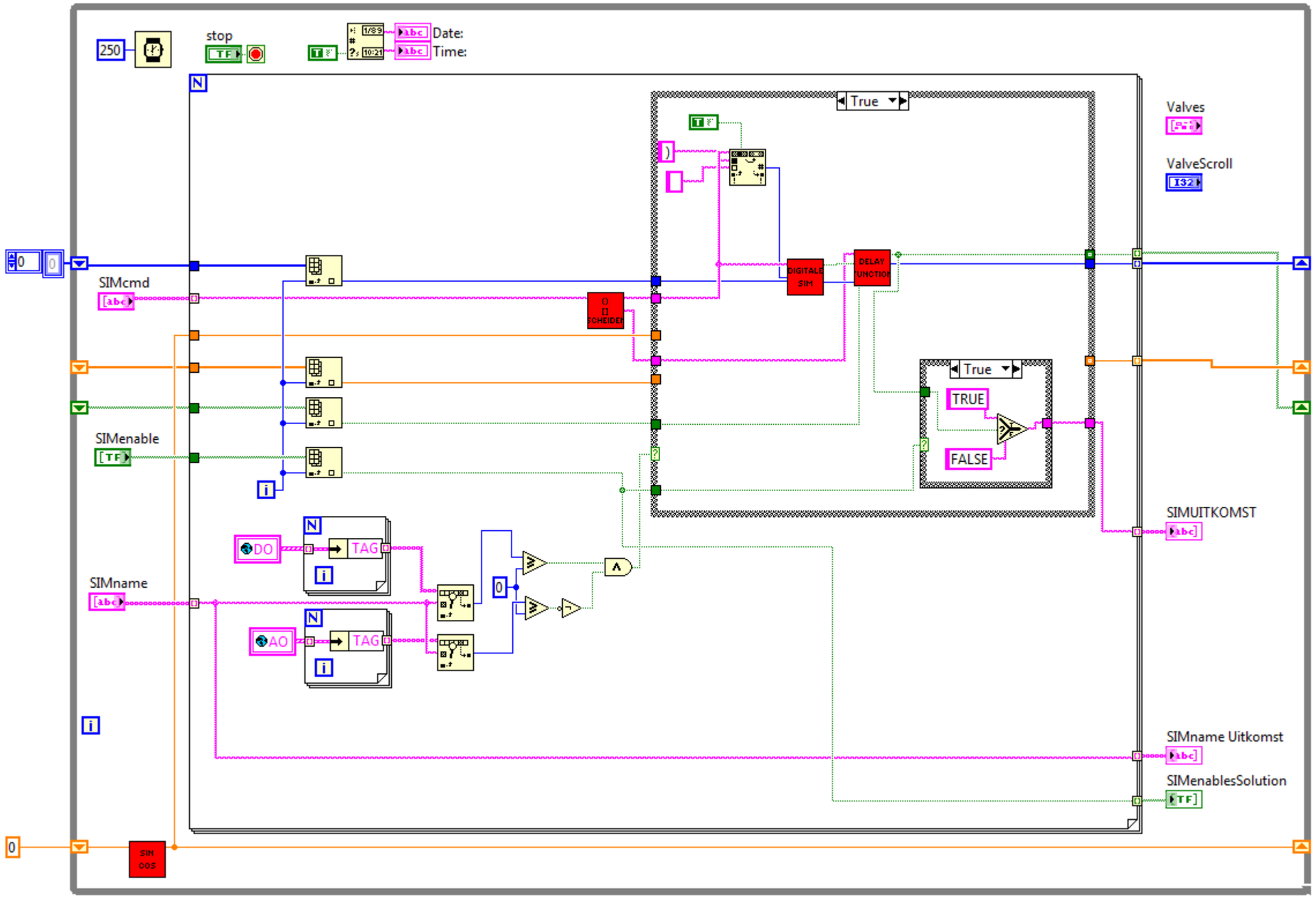


figuur C.1. Broncode User-friendly IO configuratie.

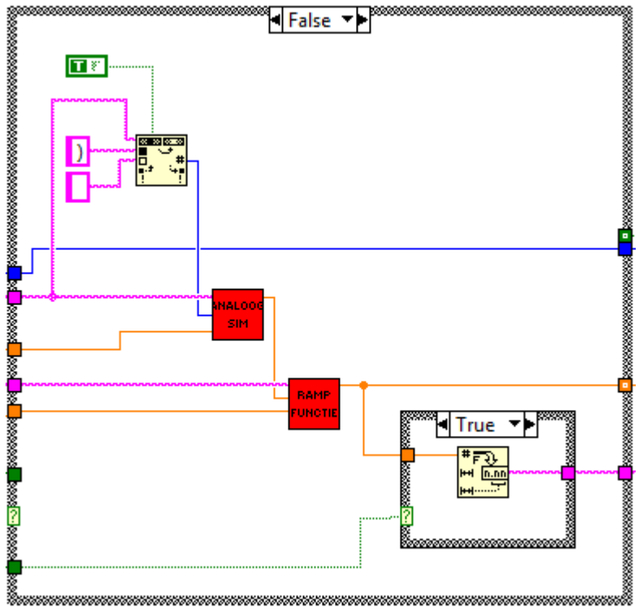
Bijlage D Uitbreiding simulatie mogelijkheden

In deze bijlage staan de broncodes van de simulatiemogelijkheden.

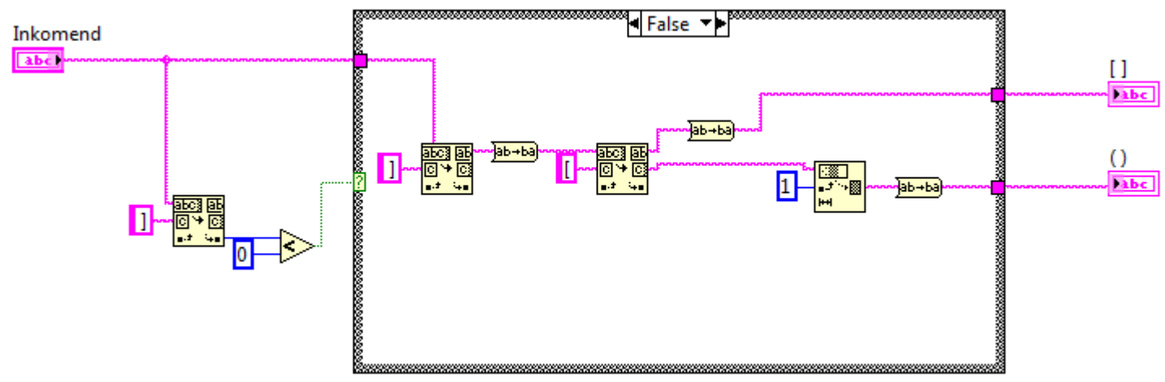
In figuur D.1 staat de broncode van de simulatie mogelijkheden.
In figuur D.2 staat de broncode van de ()[] scheiden.
In figuur D.3 staat de broncode van de digitale simulatie.
In figuur D.4 staat de broncode van de analoge simulatie.
In figuur D.5 staat de broncode van de digitale special.
In figuur D.6 staat de broncode van de analoge special.



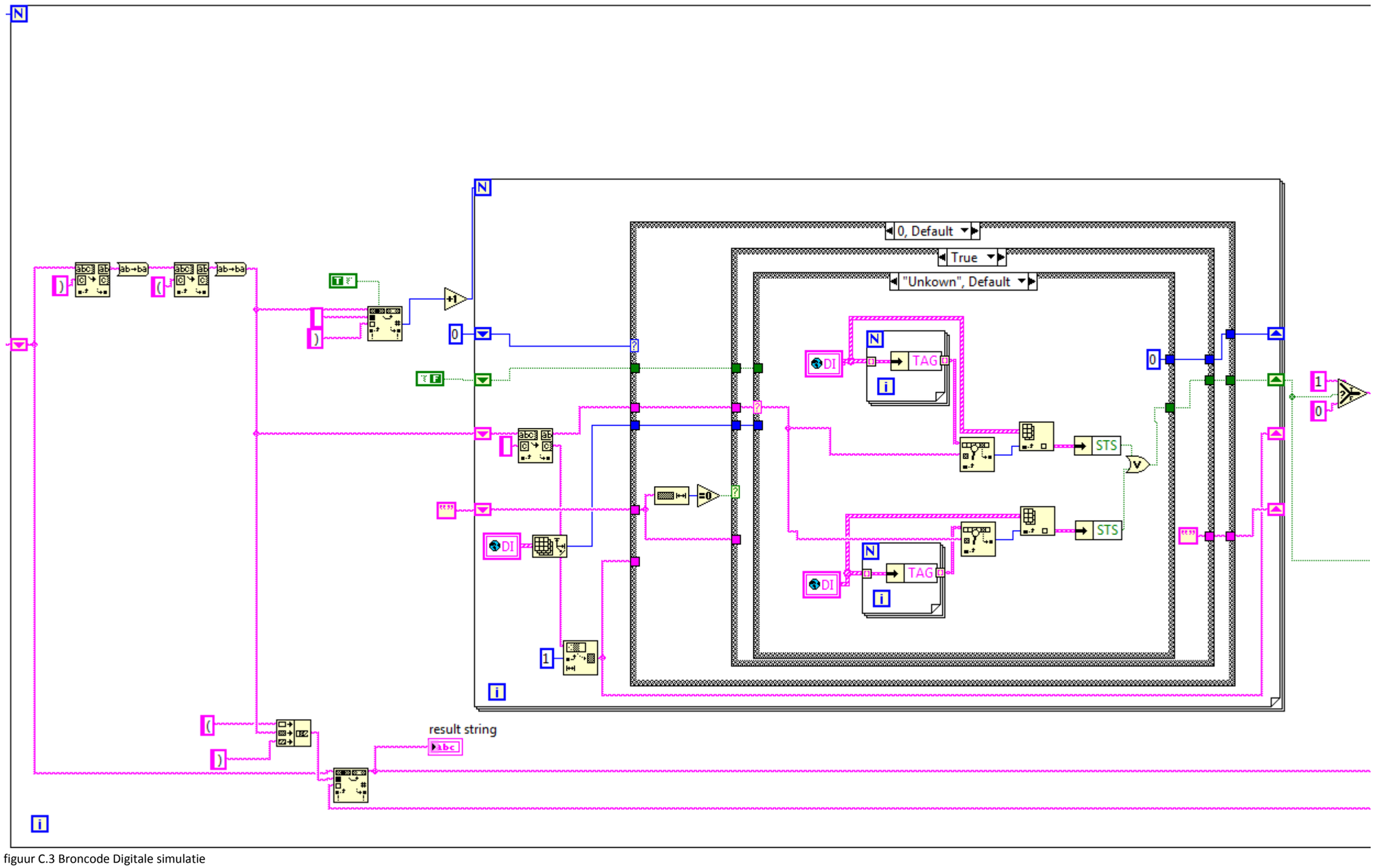
figuur C.1. Broncode Uitbreiding simulatie mogelijkheden.



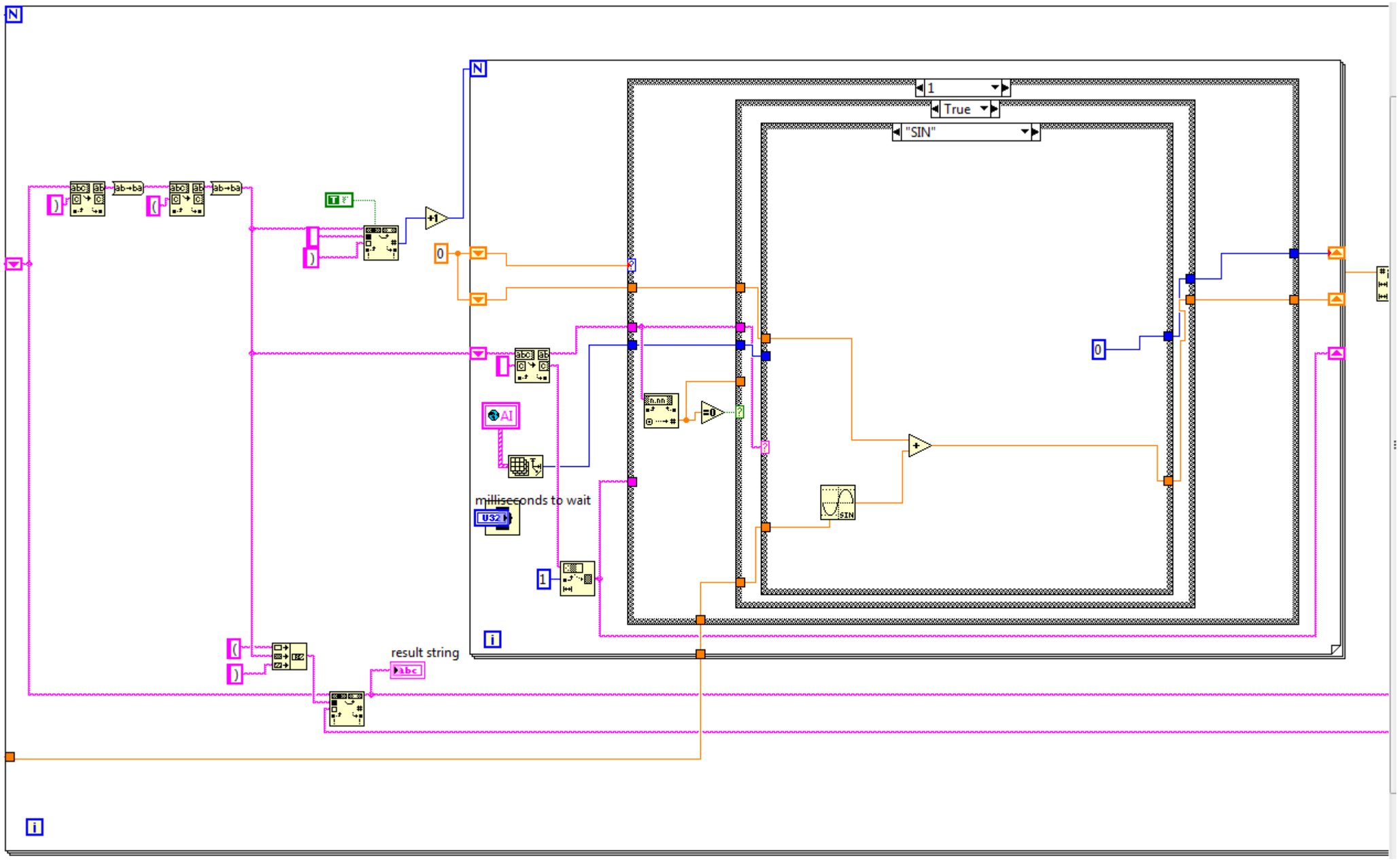
figuur C.1. Broncode Uitbreiding simulatie mogelijkheden (vervolg).



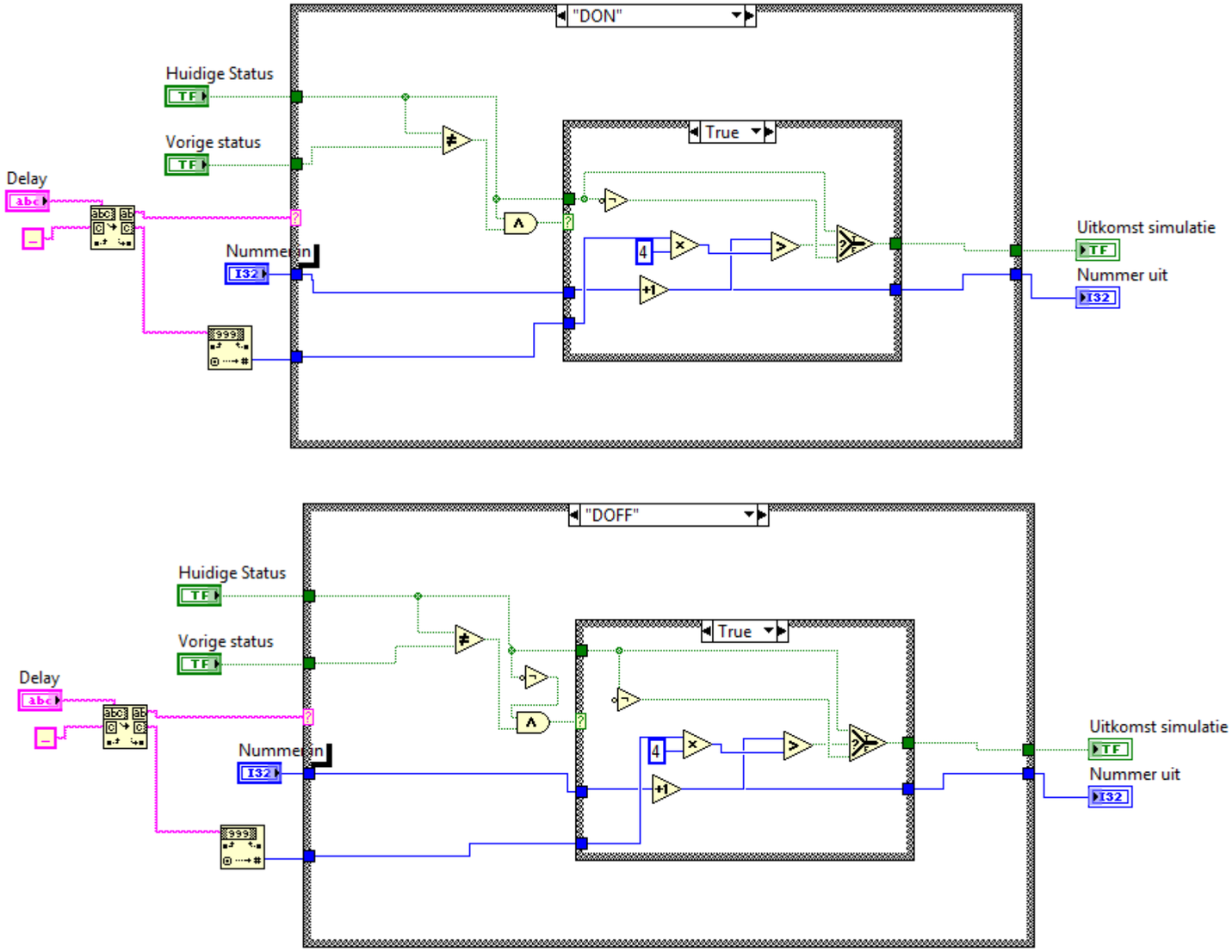
figuur C.2. Broncode () [] scheiden.



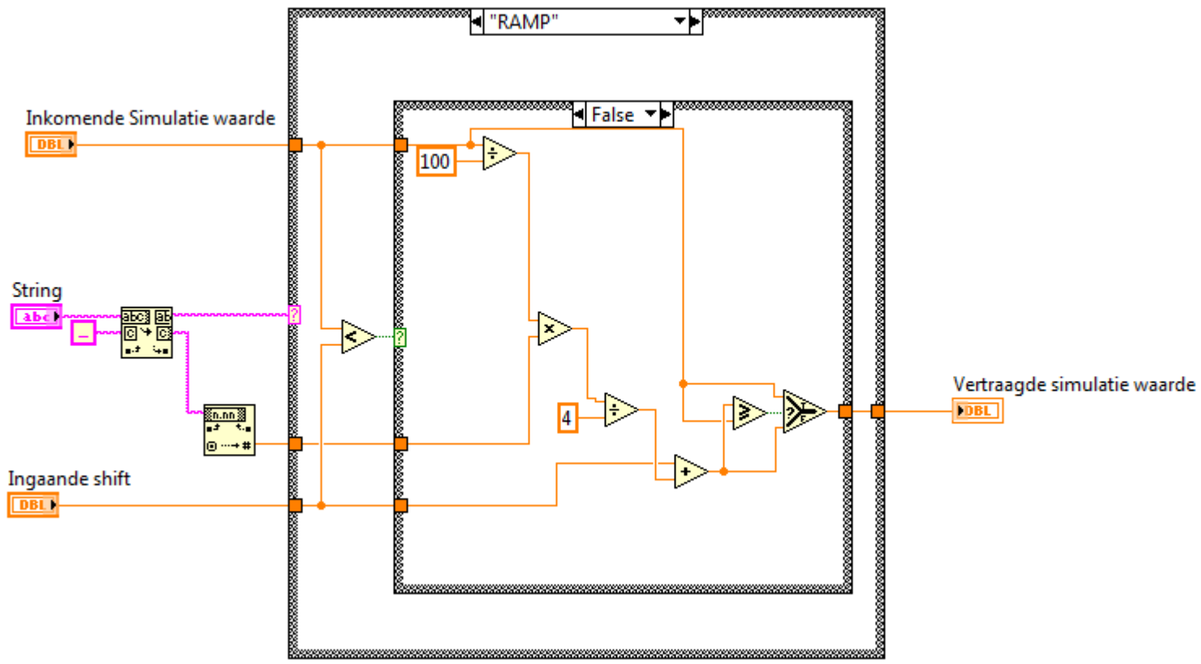
figuur C.3 Broncode Digitale simulatie



figuur C.4 Broncode analoge simulatie.



figuur C.5. Broncode Digitale Special.



figuur C.6. Broncode Analoge Special.