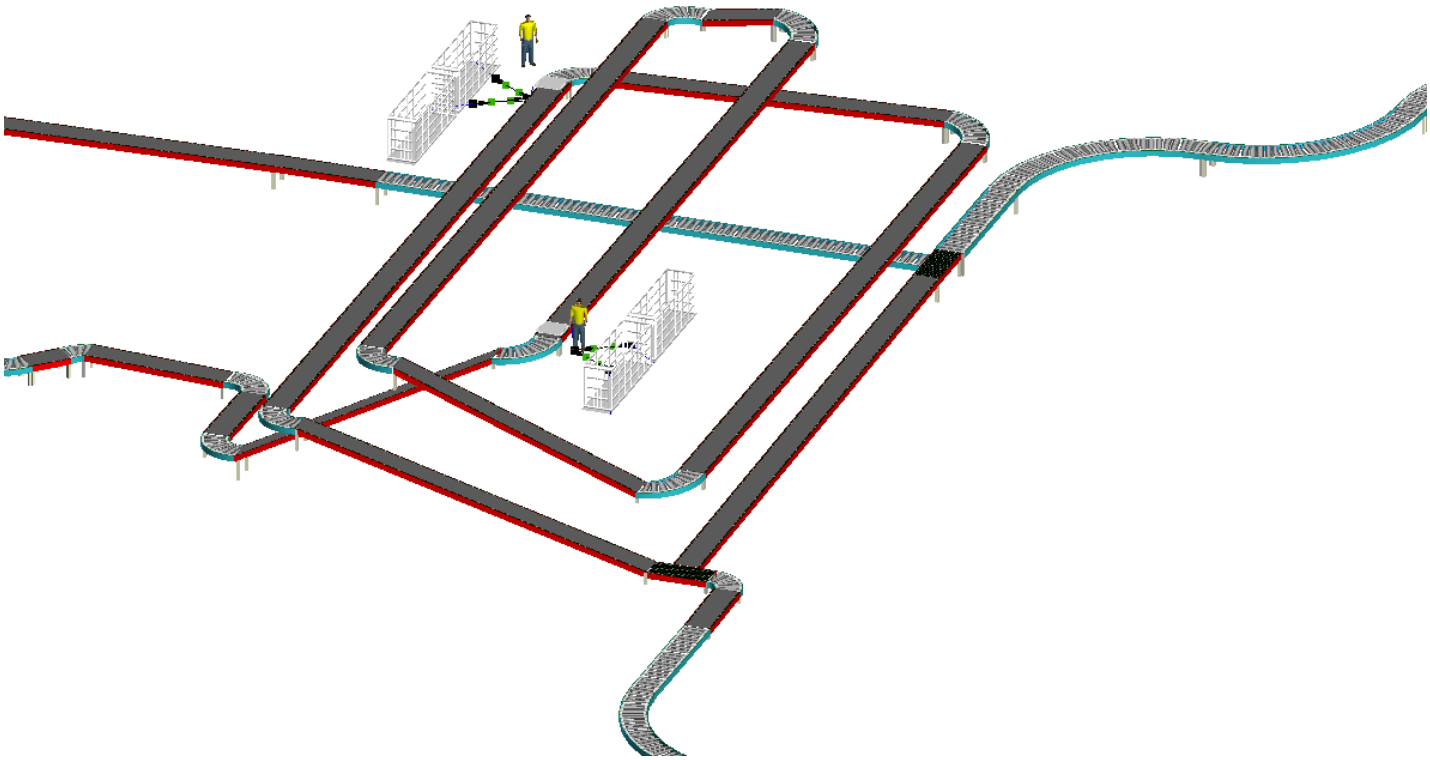
Siemuflex

Flexibel simuleren/emuleren



Afstudeerder 1:
Bart Holtermans
1520310

Afstudeerder 2:
Cedric Molthoff
1507692

Opdrachtgever:
Van Riet
Industrial Automation

Datum:
27 mei 2010

HOGESCHOOL UTRECHT
Industriële Automatisering

VANRIET

SIEMENS

FlexSim

Siemuflex

Flexibel simuleren/emuleren

Deze afstudeerscriptie is geschreven en uitgevoerd binnen het kader
van de opleiding Industriele Automatisering door

Bart Holtermans
Studentnummer 1520310

Cedric Molthoff
Studentnummer 1507692

Hogeschool:	Hogeschool Utrecht (HU)
Opleiding:	Industriële Automatisering
1ste docentbegeleider:	Dhr. J. Schouten
2de docentbegeleider:	Dhr. E. Karsemeijer
Bedrijf:	Van Riet IA
Adres:	Kaagschip 8
Postcode:	3991 CS
Plaats:	Houten
Bedrijfsbegeleider:	Dhr. P. Bosma
Inleverdatum scriptie:	25 mei 2010
Afstudeerzitting:	18 jun 2010
Vertrouwelijk:	Nee

Voorwoord

Voor u ligt een scriptie geschreven in het kader van de opleiding Industriële Automatisering aan de Hogeschool Utrecht. Deze scriptie is het resultaat van de afstudeeropdracht die wij bij Van Riet hebben uitgevoerd. Deze afstudeeropdracht bestaat uit het ontwikkelen van een interface tussen een Siemens S7 PLC en het simulatiepakket Flexsim.
Tijdens deze opdracht zijn wij veel aan het werk geweest met C++. Hierdoor hebben wij over deze programmeertaal veel kennis en ervaring opgedaan.

Wij willen vanuit deze plaats Piet Bosma bedanken voor zijn begeleiding vanuit Van Riet. Zijn vragen t.a.v. de voortgang van het project hebben ons alert gehouden.

Ook willen wij Jos Schouten bedanken voor zijn begeleiding vanuit de Hogeschool Utrecht. Zijn snelle en adequate antwoorden op onze vragen hebben ons zeer geholpen.

Verder willen wij de medewerkers van Van Riet bedanken voor hun raad, adviezen en interesse. Door hun ongecompliceerde manier van werken en omgang hebben wij een prettige werksfeer gehad.

Daarnaast willen wij Steven Harmoen, Dirk-Jan Moens en Martijn van Oostenbrugge van Talumis bedanken voor hun support bij het Flexsim pakket.

Tot slot willen wij Henk Molthoff hartelijk danken voor zijn taalkundige blik evenals iedereen die ons ook maar op enige wijze heeft bijgestaan bij het schrijven van deze scriptie.

Cedric Molthoff & Bart Holtermans

Houten, mei ’10

Samenvatting

Van Riet is een bedrijf gespecialiseerd in het ontwerp, de realisatie en de installatie van interne transportsystemen. Zij ontwikkelt haar software in eigen huis, zowel op PC- als op PLC-niveau. Deze software wordt op locatie (on-site) getest. Omdat Van Riet internationaal opereert kan deze locatie zich op grote afstand bevinden. Het bereid vinden van personeel die langere tijd op grotere afstand wil werken is lastig. Dit zet namelijk druk op hun sociale leven. Tevens is het een prijzige aangelegenheid om personeel naar het buitenland te sturen in verband met uurtoeslagen, tickets en andere zaken. Het is Van Riet er dus alles aan gelegen om de testduur on-site, en daarmee de complete doorlooptijd van een project, zo kort mogelijk te houden.

Vanuit dit oogpunt ontstond de behoefte om het PLC-programma van een bepaald transportsysteem grotendeels “in-house”, op de thuislocatie in Houten, te kunnen testen. Dat is het doel van deze opdracht.

Om dit te bewerkstelligen moeten alle inputs naar en alle outputs van de PLC gesimuleerd worden.

Van Riet beschikt over een licentie voor het 3D simulatiepakket Flexsim. Binnen Flexsim wordt het transportsysteem compleet met sensoren en actuatoren nagebouwd waardoor er een visualisatie van het transportsysteem ontstaat.

De PLC wordt nu via een interface gekoppeld aan Flexsim. De PLC bevat het voor een bepaald transportsysteem geschreven PLC-programma. De inputs die normaal gesproken van sensoren in het transportsysteem naar de PLC worden gestuurd, worden nu vanuit Flexsim gestuurd. De outputs die normaal gesproken vanuit de PLC naar de actuatoren in het transportsysteem worden gestuurd, worden nu ook naar Flexsim gestuurd.

In deze afstudeeropdracht is de interface gerealiseerd tussen de PLC en Flexsim. Deze interface is geschreven in de programmeertaal C++ en heeft de naam Siemuflex gekregen.

Deze interface is naar tevredenheid op werking getest en opgeleverd.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1. Inleiding

1.1. Het Bedrijf

1.1.1. Van Riet

De naam Van Riet is al meer dan 60 jaar een goede bekende in de wereldmarkt voor interne transportsystemen. De systemen zijn geïnstalleerd van Mexico tot Shanghai, van Praag tot Istanbul en van Los Angeles tot Amsterdam.

Het merendeel van de mechanische systeemelementen ontwikkelt Van Riet in eigen huis, ondersteund door haar eigen R&D-afdeling.

Met de juiste systeembesturing wordt het transportsysteem afgemaakt. Van Riet ontwikkelt haar software in eigen huis, zowel op PC- als op PLC-niveau.

Van Riet biedt met dit pakket totale systeemoplossingen, waarbij een Turn-Key systeem wordt opgeleverd. Van Riet denkt actief mee, vanaf de eerste oriëntatie tot en met de implementatie. Om de systeemwerking voor jaren te garanderen, biedt Van Riet een uitgebreid scala aan Service Packs aan.

Van Riet is gespecialiseerd in het ontwerp, de realisatie en de installatie van Pallet-, Order Pick-, Sorteer-, Robot- en Transportsystemen. In eerste instantie werden alleen kolen en mest getransporteerd. Inmiddels heeft Van Riet een grote variëteit aan producten getransporteerd.
Enkele voorbeelden hiervan worden in de tabel hieronder gegeven.

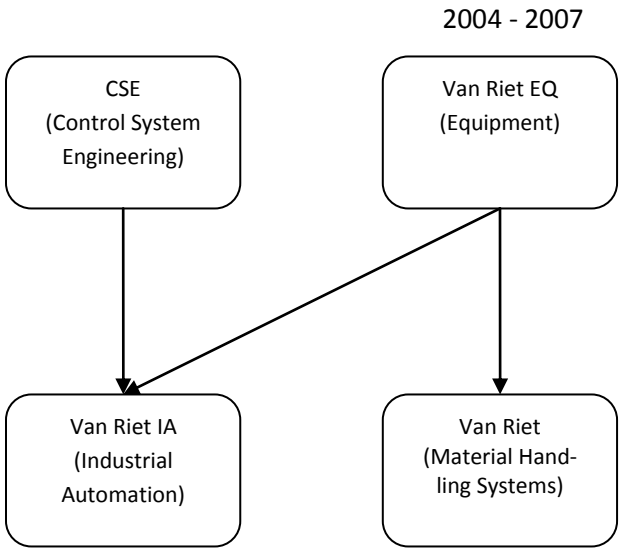
Getransporteerde producten			
Autobanden	Automotoren	Bakken	Blikjes bier
Bloemen	Boeken	Boutjes	Camera's
Consumentenelektronica	Emballage kratten	Fietsbanden	fornuizen
Fruit	Gereedschap	GSM toestellen	Grafstenen
Isolatiemateriaal	Kantoorartikelen	Keukenmachines	Kleding
Koelkasten	Medicijnen	Montage materiaal	Olievaten
Pallets	Postpakketten	Stoelen	Velgen
Verf	Verpakt vlees	Wasmachines	Zonnehemels

Tabel 1: Overzicht producten getransporteerd door Van Riet

1.1.2. Van Riet Industrial Automation

In ± 2002 waren de financiële middelen van Van Riet helaas niet meer toereikend om in zijn geheel door te gaan. Van Riet heeft een doorstart gemaakt onder de naam Van Riet Material Handling Systems (Van Riet MHS). Van Riet is doorgegaan met het bouwen van interne transportsystemen maar zonder besturingsafdeling.

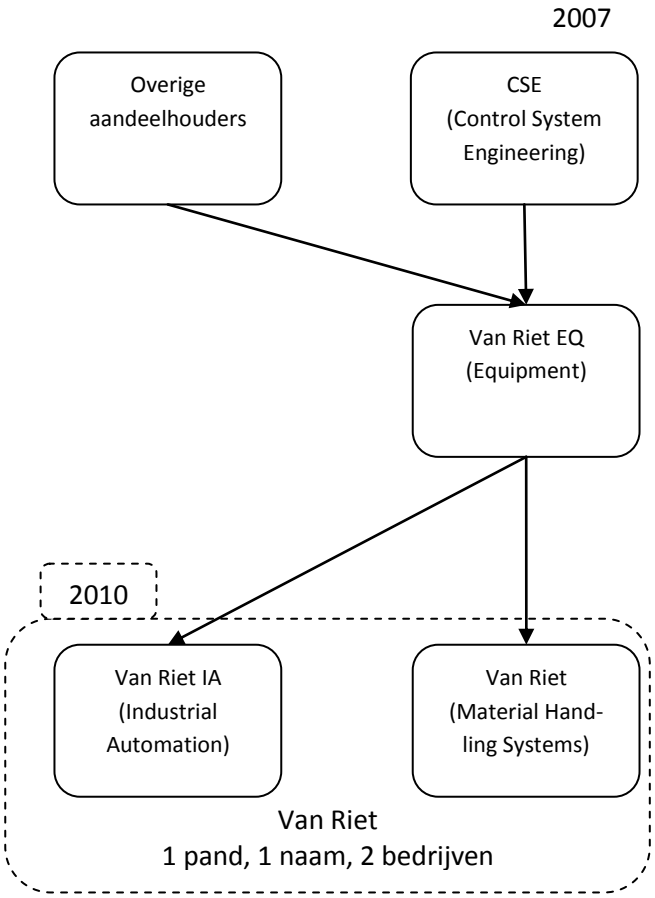
De kennis en expertise die aanwezig was/is bij de mensen die voorheen op de besturingsafdeling van Van Riet werkten was toch van belang voor Van Riet MHS. Een deel van deze werknemers hadden hun loopbaan voortgezet bij het bedrijf Control System Engineering (CSE). CSE en Van Riet Equipment (EQ) hebben toen het bedrijf Van Riet Industrial Automation (IA) opgezet. CSE en Van Riet (EQ) waren hier dan ook de aandeelhouders van.



Figuur 1: Structuur Van Riet 2004 - 2007

In 2007 zijn de verhoudingen veranderd. Van Riet EQ werd volledig aandeelhouder van Van Riet IA. CSE heeft in ruil voor zijn aandelen in Van Riet IA aandelen in Van Riet EQ gekregen.

Vanaf medio maart 2010 treden Van Riet MHS en Van Riet IA als één bedrijf vanuit één pand naar de buitenwereld. Alleen op papier achter de schermen zullen de twee bedrijven blijven bestaan.

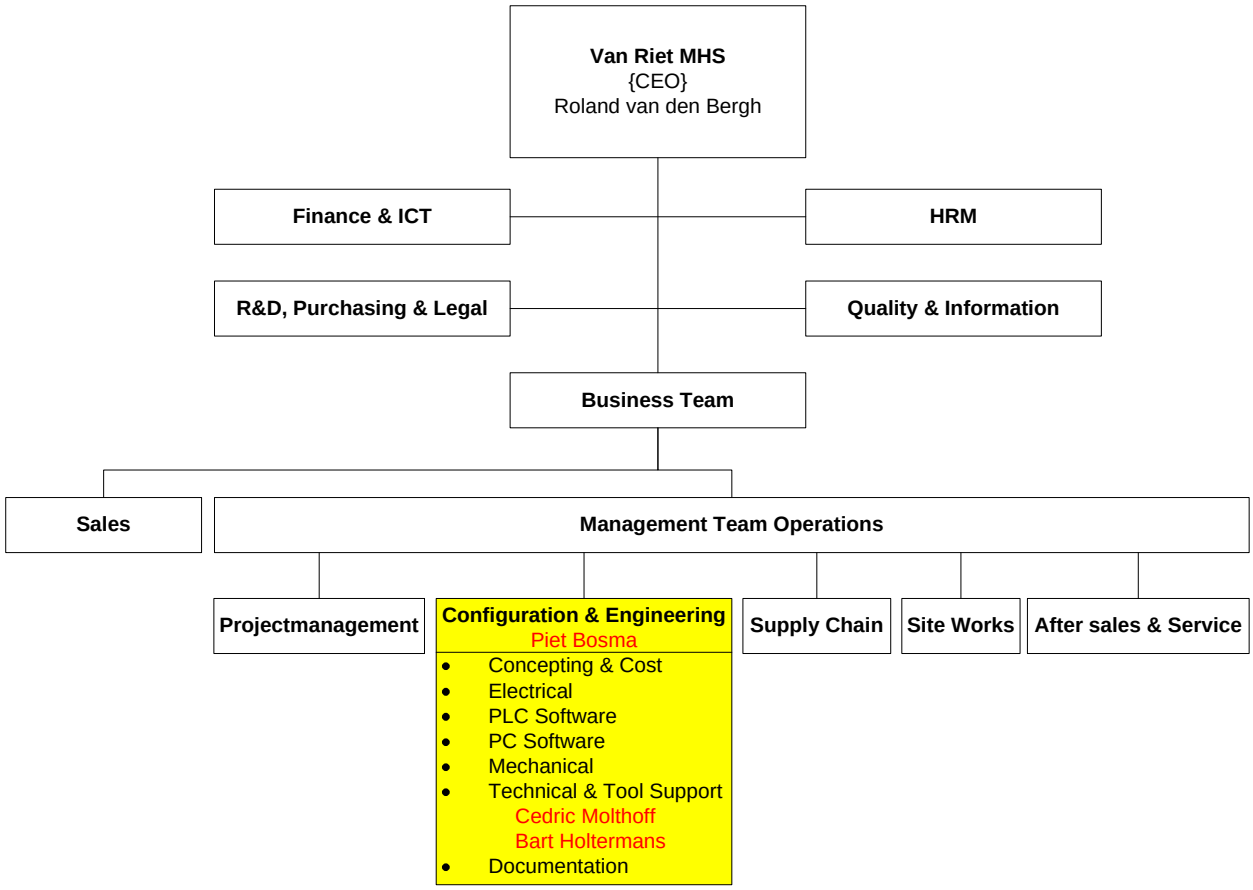


Figuur 2: Structuur Van Riet 2007 - heden

1.1.2.1. Afdelingen Van Riet IA

Van Riet IA bestaat uit twee afdelingen: soft- en hardware-engineering. Deze afstudeeropdracht is uitgevoerd bij de afdeling software-engineering. Op de afdeling software-engineering zijn er twee takken van sport, PLC- en PC-programmering. Dit project valt onder beide takken van sport, aangezien er een koppeling gelegd wordt tussen een PLC en PC. Vanaf medio maart 2010 valt Van Riet IA binnen de nieuwe structuur onder de afdeling Configuration & Engineering. De afstudeerders vallen onder de subcategorie Technical & Tool Support.

Hier volgt een overzicht van de structuur van het bedrijf vanaf maart 2010, met alleen de namen van personen relevant voor deze afstudeeropdracht:



Figuur 3: Interne bedrijfsstructuur

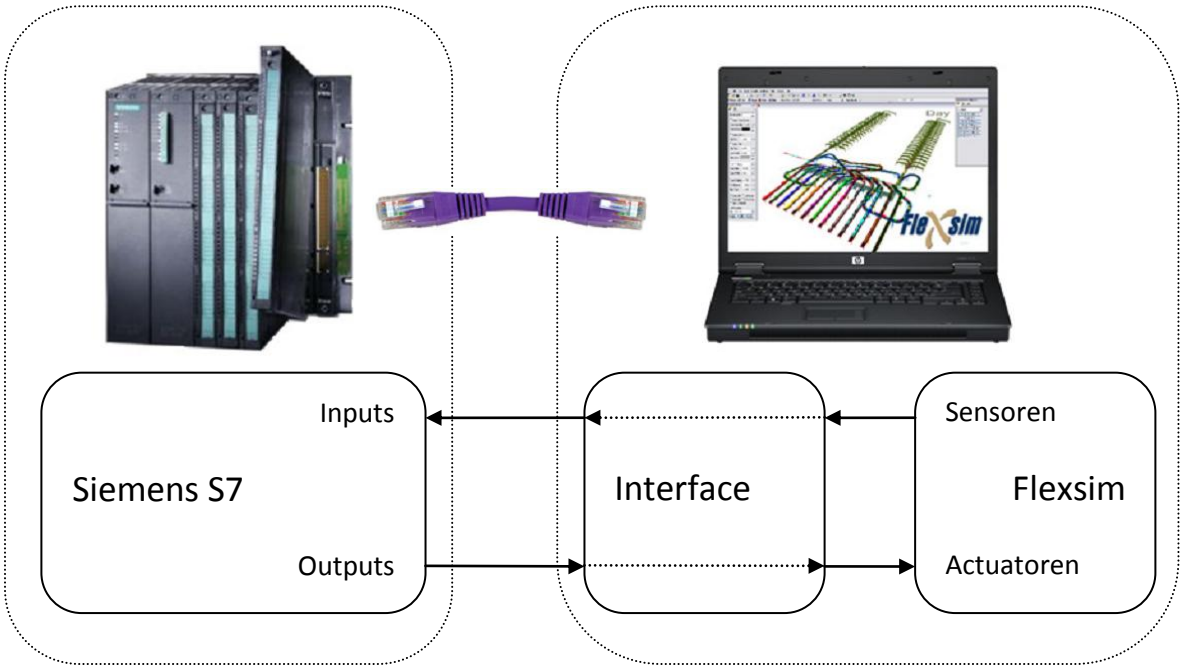
1.2. De opdracht

1.2.1. Omschrijving

1.2.1.1. Hoofdopdracht

De opdracht kan globaal als volgt worden omschreven: Een manier vinden om de PLC-software grotendeels in-house te testen. Daarbij moet gebruik gemaakt worden van het 3D simulatiepakket Flexsim. Om dit mogelijk te maken moet er een koppeling/interface tussen een Siemens PLC en het 3D simulatiepakket Flexsim gerealiseerd worden. Doordat de PLC aan de PC gekoppeld wordt, en via de te creëren interface kan communiceren met het simulatiepakket, ontstaat er een emulatie.

Het testen van de PLC-software werd voorheen on-site gedaan en wordt op deze manier in-house gehaald.



Figuur 4: Voorstelling hoofdopdracht

1.2.1.2. Deelopdracht

Voor Van Riet is de hoofdopdracht onderdeel van een groter geheel. De modellen die het transportsysteem simuleren in Flexsim moeten handmatig gecreëerd worden. Het uiteindelijke doel is om deze modellen automatisch te creëren vanuit het technical sales en planningpakket P'X5. Naast de bovengenoemde interface zal dan ook gekeken worden naar verdere automatisering in het hogere proces. De bedoeling is om uiteindelijk met P'X5 eenmalig de gegevens van de componenten in te voeren, waarna Flexsim de juiste gegevens overneemt en hieruit een model creëert. De deelopdracht is om, indien het qua tijd mogelijk is, de gegevens in kaart te brengen die Flexsim hiervoor nodig heeft.

1.2.2. Belang voor het bedrijf

Van Riet heeft drie voordelen bij de in de hoofdopdracht beschreven manier van testen:

1. Het personeel van Van Riet kan in grote lijnen de PLC-software op kantoor testen. Hierdoor bevindt men zich kortere tijd on-site, waardoor het sociale leven van de werknemers minder onder druk komt te staan.
2. Interne uren zijn minder kostbaar dan externe uren waardoor kosten bespaard kunnen worden.
3. Met het in-house testen wordt de totale doorlooptijd van een project verkort.

1.2.3. Globaal pakket van eisen, randvoorwaarden en prioriteiten

In de volgende tabel wordt weergegeven aan welke eisen en wensen de oplevering van het project moet voldoen. Tevens is de prioriteit aangegeven aan de rechterzijde volgens de MoSCoW methode. De prioriteiten zijn:

- **M**ust have this - deze eis moet in het eindresultaat terugkomen.
- **S**hould have this if at all possible - deze eis is zeer gewenst, maar een vergelijkbare eigenschap is ook goed genoeg.
- **C**ould have this if it does not affect anything else - deze eis mag alleen aan bod komen als er tijd genoeg is.
- **W**ould have - deze eis zal nu niet aan bod komen maar kan in de toekomst interessant zijn.

Eis	Prioriteit
Er wordt een interface gemaakt tussen een Siemens PLC en Flexsim	M
De interface kan outputs uit de PLC overbrengen naar Flexsim	M
De interface kan inputs vanuit Flexsim overbrengen naar de PLC	M
De communicatie gaat real time	M
Er wordt een werkende testopstelling opgeleverd	M
Er kunnen verschillende types PLC (S7-300 en S7-400) gebruikt worden	S
De specificaties benodigd voor de interface tussen P'X5 en Flexsim worden onderzocht	C

Tabel 2: Eisen

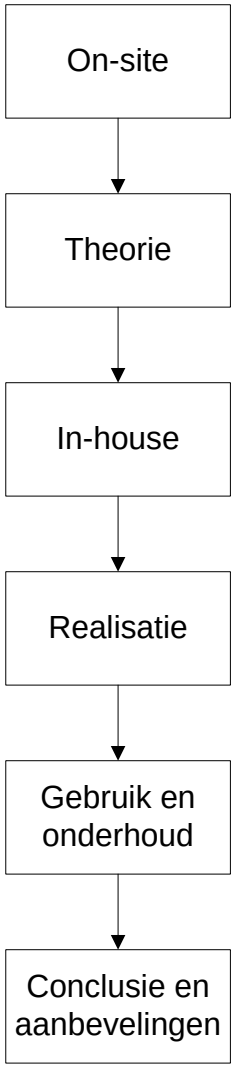
In de volgende tabel wordt weergegeven welke randvoorwaarden aan het project verbonden zijn.

Randvoorwaarde
Het project zal alleen de koppeling tussen Siemens PLC en Flexsim bevatten, en geen koppelingen naar andere lagen
Het zal niet zeker zijn of aan het eind van het project alles automatisch zal werken, hier wordt uiteraard wel naar gestreefd
Het project wordt in eerste instantie ontwikkeld voor één type PLC
De interface zal gecreëerd worden maar het kan zijn dat het programma Flexsim per project bijgesteld dient te worden

Tabel 3: Randvoorwaarden

1.3. Documentstructuur

Voor optimaal leesgemak zal in deze paragraaf de documentstructuur kort en bondig worden uitgelegd aan de hand van de belangrijkste hoofdstukken.



In dit hoofdstuk zal uiteengezet worden hoe de huidige situatie is. In dit hoofdstuk zal extra duidelijk worden wat de nadelen van de huidige situatie zijn.

Dit document en de gemaakte keuzes zijn uiteraard gebaseerd op een stuk theoretische kennis. Om het lezen van dit document te ondersteunen zal in dit hoofdstuk de basis theorie uiteengezet worden.

In dit hoofdstuk zal Siemuflex verder worden toegelicht doormiddel van een functionele beschrijving en hierbij de onderbouwing van gemaakte keuzes.

Het doel van de opdracht is het opleveren van een werkende interface. Hier zal dan ook de geschreven software behandeld worden aan de hand van flowcharts. De broncode is terug te vinden in de bijlagen.

Met deze opdracht is de basis van Siemuflex gelegd. Om te zorgen dat Siemuflex verder uitgebreid kan worden met toekomstige wensen zal een duidelijk overzicht geschetst worden welke bestanden nodig zijn en waar deze dienen te staan. Ook het gebruik van Siemuflex wordt hier uiteengezet.

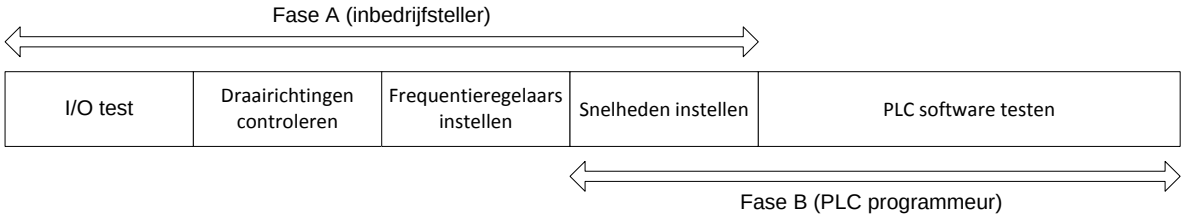
Het laatste essentiële hoofdstuk voor de opdracht is Conclusie en aanbevelingen. Tijdens het doorlopen van deze opdracht zijn er een aantal nieuwe inzichten ontstaan. Deze informatie zal hier worden uiteengezet.

Figuur 5: Documentstructuur

2. On-site

2.1. Procedure inbedrijfstelling

De inbedrijfstelling is op te delen in twee fasen: Fase A en fase B. Hieronder wordt een schematische voostelling van de inbedrijfstelling gegeven. In de volgende paragrafen worden de fasen van de inbedrijfstelling uiteengezet.



Figuur 6: Opzet inbedrijfstelling

2.1.1.1. Fase A

Fase A wordt uitgevoerd door een inbedrijfsteller. In deze fase worden de volgende handelingen uitgevoerd:

- *I/O test.*
  Met de I/O test worden de inputs en outputs van de PLC gecontroleerd. Er wordt gekeken of de ingangen (sensoren) binnenkomen op de PLC als ze bediend worden. Er wordt ook gekeken of de outputs reageren als ze aangestuurd worden. Hierbij moet gedacht worden aan controleren of motoren draaien, of uitstoters functioneren enz.

- *Draairichtingen controleren.*
  Van motoren wordt de draairichting gecontroleerd.

- *Frequentieregelaars instellen.*
  Van de frequentieregelaars wordt de configuratie juist ingesteld. Hierbij moet gedacht worden aan het instellen van het type motor die aan de frequentieregelaar is aangesloten, of er een remweerstand aanwezig is enz.

- *Snelheden instellen.*
  De snelheden van de frequentieregelaars moeten aangepast worden zodat de motoren op de snelheden draaien waarvoor het systeem ontworpen is. Dit wordt meestal in samenspraak met de PLC programmeur gedaan waardoor hier overlap ontstaat.

2.1.1.2. Fase B

Fase B wordt uitgevoerd door een PLC-programmeur. In deze fase worden de volgende handelingen uitgevoerd:

- *Snelheden instellen.*
  In samenspraak met de inbedrijfsteller worden de snelheden van de frequentieregelaar aangepast zodat de motoren op de snelheden draaien waarvoor het systeem ontworpen is.

- *PLC-software testen.*
  Hier wordt de functionaliteit van het PLC-programma getest.

2.2. Knelpunten & voordelen

Een PLC-programma van Van Riet bestaat voor een deel uit standaard softwareblokken en een deel specifieke software.

2.2.1.1. Standaard softwareblokken

Voor een aantal onderdelen in een transportsysteem zijn door Van Riet standaard PLC-programma blokken geschreven. Er zijn bijvoorbeeld een aantal standaard transportbanden met enkele sensoren en motoren die altijd dezelfde functie hebben, ongeacht in welk transportsysteem ze geplaatst zijn. Hiervoor is de software gestandaardiseerd.

Het implementeren van deze blokken in het programma is voor een groot deel kopieerwerk. Hierbij kunnen fouten gemaakt worden. Een voorbeeld is dat een deel van een motoraansturing gekopieerd wordt maar dat vergeten wordt het outputnummer aan te passen.
Het voordeel dat Siemuflex kan bieden is dat een dergelijke fout al in-house eruit gehaald kan worden, in plaats van on-site.

Ook moet tijdens de inbedrijfstelling naar voren komen of de afzonderlijke blokken goed gekoppeld zijn. Hierbij kan Siemuflex in een vroeg stadium, in-house, fouten naar voren brengen.

2.2.1.2. Specifieke software

Een deel van de PLC-software is altijd specifiek voor een bepaald systeem. Dat deel wordt geheel nieuw bedacht en geschreven. De juistheid van het ontwerp van de software wordt on-site getest. Door het gebruik van Siemuflex kan dit in-house gebeuren. Bijkomend voordeel is het feit dat het on-site arbeidsintensief is om de juistheid van het ontwerp te testen. Als er bijvoorbeeld een fout in de software zit waardoor het testobject (meestal een pakketje) op een verkeerde plek terechtkomt, moet deze daar opgehaald worden en na aanpassing van de software opnieuw in het systeem geplaatst worden. Met Siemuflex is dit een kwestie van de simulatie resetten en opnieuw starten, dit kost aanzienlijk minder tijd.

3. Theorie

3.1. Siemens

3.1.1. Inleiding

Siemens AG is een concern afkomstig uit Duitsland dat zich bezighoudt met elektrotechniek en elektronica. Siemens is werkzaam in de sectoren industrie, energie, gezondheid en consumentenelektronica.

3.1.2. SIMATIC

De producten van Siemens die Van Riet gebruikt komen uit de industriesector. Deze divisie is weer opgedeeld in een aantal subdivisies, waar Automation er één van is.

Onder Automation vallen een aantal automatiseringsconcepten, waaronder SIMATIC. SIMATIC is een productgroep binnen de automatiseringstak van Siemens. De naam SIMATIC is een geregistreerd handelsmerk, en is een samenvoeging van "**Si**emens" en "Auto**matic**".

3.1.3. Siemens S7

De SIMATIC S7 serie bestaat uit een aantal modulaire PLC's. Deze worden gebruikt om systemen mee te besturen en regelen. De S7 serie bestaat uit de volgende types:

- SIMATIC S7-200.
- SIMATIC S7-300.
- SIMATIC S7-400.
- SIMATIC S7-1200.

Van Riet gebruikt voornamelijk de S7-300 en S7-400 serie. De Siemuflex interface is ontwikkeld voor de S7-300 en S7-400 PLC. Siemuflex is uitvoerig getest op een S7-400 PLC.

Figuur 7: Siemens S7-400 PLC

3.1.4. Programma doorloop en process image

De CPU van een S7 PLC doorloopt elke cyclus een aantal stappen:

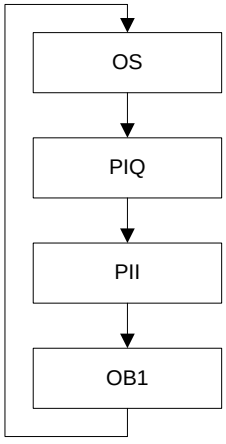
Programmadoorloop
Het Operating System (OS) start de cyclus monitor tijd
De CPU schrijft de waarden van de outputs van de process image output tabel (PIQ) naar de output modules
De CPU leest de waarden van de inputs van de input modules en schrijft deze naar de process image input tabel (PII)
De CPU verwerkt het PLC-programma en voert de instructies uit
Aan het eind van de cyclus, voert het OS nog wachtende taken uit, bijvoorbeeld blokken downloaden en verwijderen, en het ontvangen en verzenden van globale data
De CPU begint weer aan het begin van de cyclus en herstart de monitor tijd

Tabel 4: Programmadoorloop S7 CPU

Als de inputs (I) en de outputs (Q) in het PLC-programma worden aangesproken, kijkt het programma niet rechtstreeks naar de signaalstatus op de digitale signaalmodules, maar in een deel van het systeemgeheugen van de CPU en gedistribueerde I/Os. Dit geheugengebied staat bekend als het process image.

In vergelijking met rechtstreekse toegang tot de inputs en outputs op de verschillende modules, is het belangrijkste voordeel van het process image dat de CPU een consistent beeld van de proces signalen voor de duur van een programmacyclus heeft. Als een signaalstatus op een ingangsmodule verandert terwijl het programma wordt uitgevoerd, wordt de signaalstatus in het process image vastgehouden totdat het process image weer is bijgewerkt in de volgende cyclus. Dit zorgt ervoor dat niet de status van een ingangssignaal tijdens de uitvoering van een programma kan veranderen, wat eventueel problemen kan opleveren als dit ingangssignaal meerdere keren wordt gebruikt in één PLC-programma.

Toegang tot het process image vergt ook veel minder tijd dan rechtstreekse toegang tot de signaalmodules, aangezien het process image zich in het interne geheugen van de CPU bevindt.



Figuur 8: Schematische weergave programmadoorloop S7 CPU

3.1.5. Step 7

STEP 7 is het standaard software pakket dat wordt gebruikt voor het configureren en programmeren van SIMATIC PLC's. Het is een onderdeel van de SIMATIC-industrie software. Met STEP 7 kan:

- de hardware geconfigureerd worden.
- communicatie gedefinieerd worden.
- een PLC-programma geschreven worden.
- de instellingen en het PLC-programma gedownload worden.
- getest en onderhouden worden.
- gedocumenteerd en gearchiveerd worden.
- een diagnose uitgevoerd worden.

In bijlage H "Step 7" worden de afzonderlijke computerprogramma's kort toegelicht en worden de verschillende programmeertalen waarin een PLC-programma geschreven kan worden beschreven.

3.1.6. Communicatie met S7 PLC

Er kan op verschillende manieren gecommuniceerd worden met een S7 PLC. Er wordt in dit hoofdstuk alleen ingegaan op de in deze opdracht gebruikte communicatiemethoden en –protocollen.

3.1.6.1. MPI interface

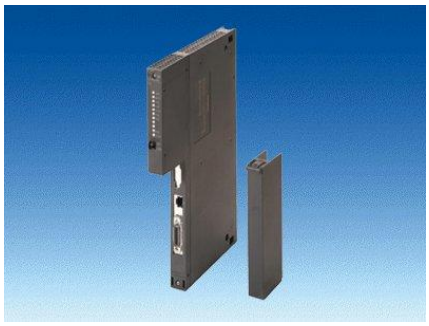
De Multi Point Interface (MPI) is een interface van Siemens om Siemens apparaten met elkaar te laten communiceren. Voorbeelden hiervan zijn controllers (PLC), panels (HMI), programmeerapparaten en PC's. MPI is een seriële verbinding die een variant is op de Profibus standaard (IEC 61158 en EN 50170). De data-uitwisseling is bi-directioneel, wat betekent dat de data beide kanten uit verzonden kan worden. Dit wordt ook wel een duplex verbinding genoemd. De ondersteunde baud rate loopt van 187,5 kbps tot en met 12 Mbps. De fysieke laag bestaat uit een 2 draads RS485 verbinding die gelijk is aan Profibus.

3.1.6.2. MPI adapter

Om een MPI verbinding tussen een Siemens PLC en een PC tot stand te brengen moet aan de PC kant een MPI adapter gebruikt worden. Deze adapter bevat een standaard Profibus aansluiting voor verbinding richting PLC en een andersoortige aansluiting voor verbinding richting PC. Dit kunnen verschillende fysieke verbindingen zijn. In deze opdracht wordt in de PC gebruik gemaakt van een PCMCIA kaart van Siemens, een CP5512, waarop de adapter aangesloten kan worden.

3.1.6.3. CP443

De CP443 is een kaart voor gebruik in een S7-400 PLC. CP staat voor Communications Processor. Met deze kaart kan de PLC op Ethernet aangesloten worden. De kaart beschikt over o.a. een standaard RJ-45 Ethernet aansluiting, welke in deze opdracht gebruikt wordt. De data uitwisseling verloopt via het TCP protocol.



Figuur 9: Siemens CP443-1 Ethernet kaart

3.1.6.4. S7 communicatie

In deze opdracht wordt gebruik gemaakt van S7 communicatie van en naar de PLC. S7 communicatie is een protocol van Siemens waarin vastgelegd is in welk formaat en welke syntaxis de data aangeleverd moet worden. Er is weinig bekend over dit protocol omdat Siemens hier geen informatie over verstrekt. Wel is bekend dat een pakket bestaat uit een 10 of 12 Byte header, een parameter blok en een data blok dat de te versturen data bevat.

3.2. Flexsim

Figuur 10: Flexsim logo

3.2.1. Inleiding

Flexsim is 3D simulatie software voor modelleren, visualiseren, analyseren en optimaliseren van zowel 'batch' processen als continu processen. Met dit softwarepakket kan men fabricage- en materiaalbehandelingen simuleren. Flexsim is zeer flexibel, waardoor men vrijwel elk proces variërend van dienstverlenende industrieën tot volledige distributiecentra kan modelleren.

3.2.2. Rapporteren / scenario's

Door meerdere scenario's uit te voeren kunnen systemen geoptimaliseerd worden tot maximale prestatie. Deze scenario's kunnen allerlei variaties omvatten, van het versnellen of vertragen van de algemene doorvoer tot het volledig uitvallen van machines binnen het systeem. Alle gegevens en trends kunnen worden weergegeven in grafieken en diagrammen.

3.2.3. Communicatie

Flexsim kan ook op diverse wijzen communiceren met de buitenwereld. Een van de manieren is met spreadsheets. Een tweede mogelijkheid van communiceren is via COM (waarover meer verderop in dit hoofdstuk). In principe kunnen alle mogelijke waarden en parameters weggeschreven en opgehaald worden van en naar applicaties van derden.

3.2.4. Programmeren

Doordat Flexsim C++ volledig ondersteunt is niets onmogelijk, maar om te zorgen dat de gebruiker geen C++ programmeur hoeft te zijn heeft Flexsim haar eigen script 'Flexscript' ontwikkeld. Flexscript is een vereenvoudigde versie van C++ en alle functies die hierbij horen zijn online duidelijk beschreven. Deze commando's worden bij installatie van het programma (ook de demoversie) mee geïnstalleerd in de help.

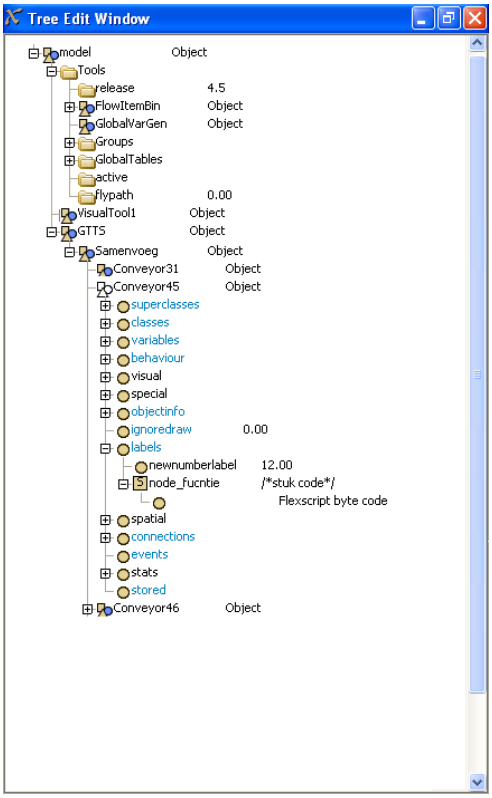
3.2.4.1. Triggers

Alle onderdelen die binnen Flexsim worden toegevoegd worden gezien als objecten. Een object heeft diverse eigenschappen en variabelen. Een dergelijk object is bijvoorbeeld een lopende band met als toevoeging één of meerdere fotocellen. Wanneer een pakketje een fotocel passeert wordt deze fotocel geactiveerd en gedeactiveerd. Aan deze activatie en deactivatie kunnen zogenaamde triggers gehangen worden. Deze triggers voeren dan een klein stukje software uit.

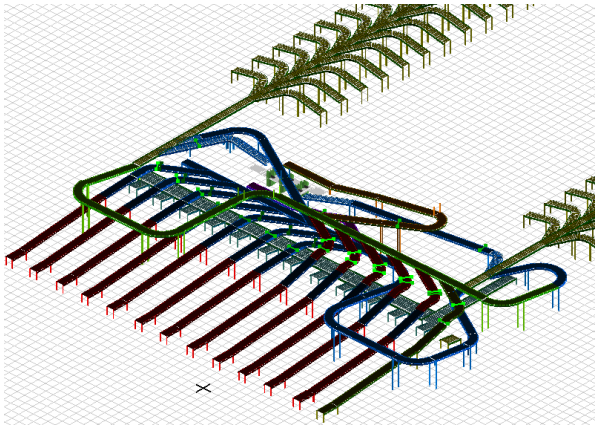
3.2.4.2. Nodefuncties

Binnen Flexsim worden alle objecten met bijbehorende variabelen in een boomstructuur ondergebracht. Alle objecten en variabelen kunnen dan ook aangesproken worden op een vergelijkbare wijze als de bestandsstructuur van Microsoft Windows. Elk onderdeel binnen de boomstructuur wordt ook wel een 'node' genoemd. Naast standaard variabelen van diverse typen, kunnen nodes ook omgeschakeld worden dat deze behandeld worden als functies (node functie), zowel in C++ als in Flexscript.

Het voordeel van deze functies is dat elk object zijn eigen functies bij zich kan dragen. Wanneer een object volledig is, kunnen deze inclusief alle functies opgeslagen worden in eigen bibliotheken.



Figuur 11: Boomstructuur Flexsim



Figuur 12: Voorbeelden Flexsim projecten

3.3. Libnodave

3.3.1. Inleiding

Libnodave is een software bibliotheek met functies waarmee data uitgewisseld kan worden met Siemens PLC's van de S7-200, S7-300 en S7-400 serie. Het is software die onder de GNU General Public License[1] valt. Deze licentie houdt in dat de software vrij verspreidt en uitgewisseld mag worden.

3.3.2. Werking

Libnodave verzendt en ontvangt data via het S7 communicatie protocol. De verbinding kan via Ethernet en MPI. Zie voor een algemene uitleg over MPI en S7 communicatie hoofdstuk 3.1.6 "Communicatie met S7 PLC".
In deze opdracht wordt gebruik gemaakt van een Ethernet verbinding met een Siemens CP443 kaart. Deze kan in een S7-400 PLC systeem geplaatst worden. Op deze kaart bevindt zich een standaard 8-pins RJ-45 Ethernet aansluiting, waarmee de kaart met een Ethernet netwerk verbonden kan worden.
Zoals eerder gezegd is Libnodave een bibliotheek met functies. De maker heeft de datapakketten die verzonden worden via S7 communicatie opgevangen en ontleed en zo het S7 communicatie protocol in kaart gebracht. Zo konden er voor veel voorkomende taken functies geschreven worden. Hierbij moet gedacht worden aan de PLC in een andere modus zetten, waarden lezen en schrijven.

3.3.3. Gebruik

De maker heeft de functies in een DLL geplaatst, die vanuit verschillende programmeertalen zijn aan te roepen. Dit zijn C, C++, C#, Delphi, Pascal, Perl en VB(A).
Een aantal functies zal altijd aangeroepen moeten worden om de verbinding met de PLC op te zetten en – na data uitwisseling – te verbreken. De normale gang van zaken:

Actie	Functie
1. Initialiseer de seriële interface	setPort
2. Initialiseer een daveInterface	daveNewInterface
3. Initialiseer de MPI adapter	daveInitAdapter
4. Initialiseer een daveConnection	daveNewConnection
5. Verbindt met een PLC met MPI adres	daveConnectPLC
6. Wissel data uit met die PLC	daveReadBytes daveWriteBytes
7. Verbreek verbinding met de PLC	daveDisconnectPLC
8. Verbreek verbinding met de adapter	daveDisconnectAdapter

Tabel 5: Gebruik Libnodave

[1] Zie: http://www.gnu.org/copyleft/gpl.html

3.4. Microsoft Component Object Model

3.4.1. Inleiding

Component Object Model (vanaf nu COM) is een binaire interface standaard welke begin jaren negentig ontwikkeld is door Microsoft. Deze standaard maakt interne communicatie tussen verschillende applicaties in verschillende programmeertalen mogelijk. De term COM wordt binnen de Microsoft ontwikkelomgeving vaak als algemene term gebruikt voor andere technieken (o.a. de OLE, OLE Automation, ActiveX, COM+ en DCOM technieken).

3.4.2. De essentie

De essentie van COM is het taalonafhankelijk hergebruiken van objecten, ook in omgevingen waar het object niet in is geprogrammeerd. Voor goed geprogrammeerde componenten staat COM hergebruik van objecten toe, zonder dat een programmeur kennis hoeft te bezitten van de interne implementatie. COM dwingt programmeurs van deze objecten goed gedefinieerde interfaces te creëren welke los staan van de implementatie. De verschillende toekenningen van betekenissen door programmeertalen wordt ondervangen door objecten zelf verantwoordelijk te maken voor hun eigen creatie en vernietiging. Het schakelen tussen verschillende interfaces gebeurt met de functie `QueryInterface()`. Deze functie wordt binnen COM automatisch mee geërfd omdat alle interfaces (in)direct moeten erven van de basis interface `IUnkown`.

Hoewel de interface op diverse computerplatformen standaard is geïmplementeerd, wordt COM voornamelijk toegepast op het Microsoft Windows platform.

3.4.3. Geschiedenis

Een van de eerste methodes van interne communicatie binnen Windows was Dynamic Data Exchange (DDE), deze techniek werd geïntroduceerd in 1987. DDE staat het zenden en ontvangen van berichten (‘praten’) tussen programma’s toe.

Object Linking and Embedding (OLE), Microsoft’s eerste object gebaseerde framework, was ontwikkeld bovenop DDE. OLE was voornamelijk ontwikkeld voor compound documents (documenten met gecombineerde inhoud) en werd geïntroduceerd samen met Word en Excel in 1991. Later werd OLE ook opgenomen binnen Windows, vanaf versie 3.1 in 1992.

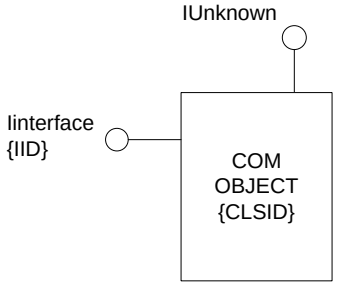
In 1993 werd de eerste versie van COM uitgegeven als onderliggende structuur voor OLE2. OLE2 moest in combinatie met COM naast ‘compount documents’ ook met softwarecomponenten in het algemeen werken.

Later werd COM uitgebreid naar DCOM zodat communicatie tussen applicaties ook over het netwerk mogelijk werd.

3.4.4. De techniek

COM programmeurs ontwikkelen hun software gebruikmakend van COM-geschikte componenten. Verschillende componenttypes worden geïdentificeerd met 'class ID's' (CLSIDs) welke wereldwijd uniek zijn (Globally Unique Indentifiers, GUID). Elk component stelt zijn functionaliteit beschikbaar door middel van één of meerdere 'interfaces'. De verschillende interfaces die ondersteund worden door een component worden elk geïdentificeerd met een Interface ID (IID), ook in dit geval met een GUID. Een Globally Unique Indentifier (GUID) is een pseudowillekeurig getal dat verondersteld wordt wereldwijd uniek te zijn. Hoewel elke GUID niet gegarandeerd 100% uniek is, is het totaal aantal unieke sleutels zo groot (2^{128}) dat de kans op de creatie van twee keer de dezelfde GUID erg klein is. Daarnaast is er een combinatie van twee GUID's (de CLSID en de IID) nodig om een bepaalde functie aan te roepen, hierdoor is de kans op duplicatie helemaal gering.

Grafisch wordt een COM-Object als volgt weergegeven:



Figuur 13: COM-object

3.4.4.1. Interfaces

Zoals eerder gezegd, alle COM componenten moeten de standaard `IUnknown` interface implementeren, hierdoor erven ook alle COM componenten automatisch de functionaliteit van deze standaard interface. De interface `IUnknown` bestaat uit drie functies:

- `AddRef()`
- `Release()`
- `QueryInterface()`

`AddRef()` en `Release()` implementeren 'reference counting' (zie ook: 3.4.4.3) en beheren de levensduur van een interface. `QueryInterface()` verzorgt aan de hand van opgegeven IID's referenties naar de (toekomstige) interfaces van een component. Naast de interface `IUnknown` zijn er meerdere standaard interfaces die gebruikt kunnen worden, zoals het ophalen en wegschrijven van bestanden (`IStream`). Voor het ontwikkelen van COM-objecten heeft o.a. Microsoft volledige bibliotheken geschreven, welke direct mee gecompileerd kunnen worden.

3.4.4.2. Klassen

De werking van COM-objecten is vergelijkbaar met de werking van objecten in een objectgeoriënteerde programmeertaal. Zowel binnen COM als binnen object georiënteerd programmeren wordt gebruikgemaakt van 'klassen'. Een klasse binnen COM is de taalonafhankelijke manier voor het definiëren van een klasse in objectgeoriënteerde zin. Een klasse kan gezien worden als een 'blueprint' die het object beschrijft.

Een co-klasse (coclass) levert de concrete implementatie(s) van een of meerdere interfaces. Het maakt niet uit in welke programmeertaal deze coclasses worden geschreven, zolang deze programmeertaal maar COM ondersteunt.

Een van de belangrijkste bijdrages van COM is dan ook de scheiding tussen interfaces en implementatie. Hierdoor is de mogelijkheid voor het ontstaan van een interface met meerdere implementaties. Dit betekent dat tijdens 'runtime' een applicatie kan kiezen tussen diverse concrete implementaties om een interface te benaderen. Concreet betekent dit dat verschillende applicaties andere functies kunnen gebruiken binnen het COM-object zonder dat alle functies bekend hoeven te zijn of geïmplementeerd hoeven te worden.

3.4.4.3. Reference Counting

De meest fundamentele interface van allemaal, `IUnknown` (waarvan elke interface zal erven), ondersteunt zoals eerder genoemd drie functies verdeeld in twee methoden: Toekomstige exploratie door `QueryInterface()` en levensduur beheer door `AddRef()` en `Release()`.

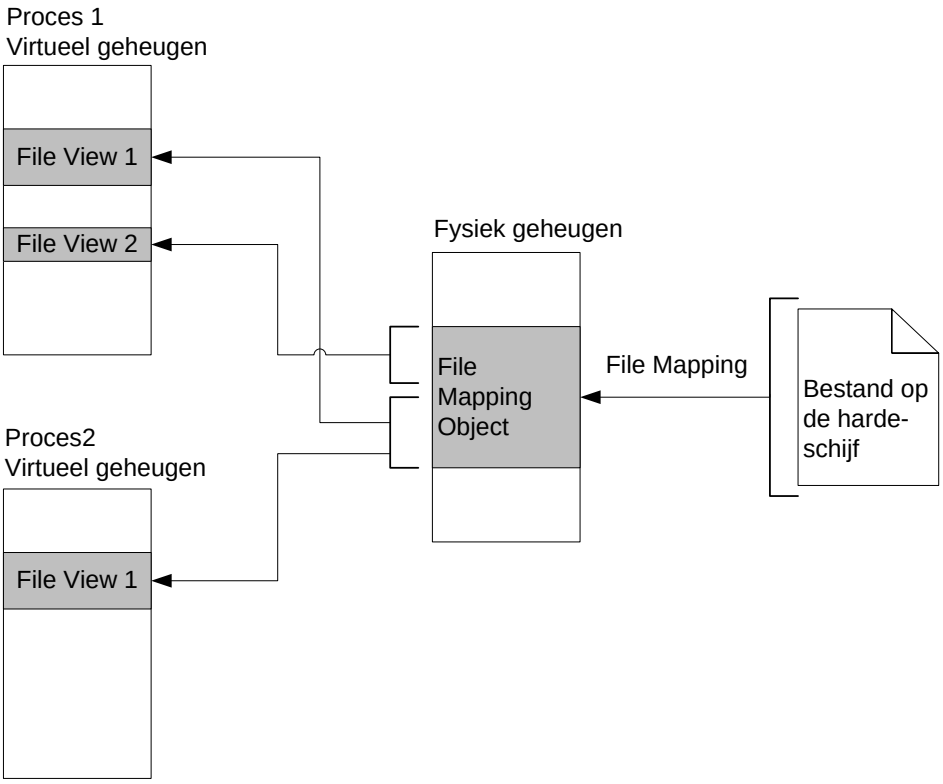
Voor de levensduur wordt de techniek 'Reference counting' vereist. Deze techniek zorgt ervoor dat individuele objecten in 'leven' blijven zolang er clients zijn die toegang tot de interface hebben verkregen. En andersom wanneer alle clients de verbinding hebben opgezegd het object op juiste wijze wordt verwijderd. Een COM-Object is dus verantwoordelijk voor het vrijgeven van zijn eigen geheugen wanneer zijn 'reference count' de nul heeft bereikt.

3.5. File Mapping

Normaliter zijn geheugenstukken toegewezen aan één of geen proces. Hierdoor zou het lastig zijn om gegevens tussen meerdere processen te delen. Om dit probleem op te lossen zijn er meerdere technieken om dit toch te bereiken. Een efficiënte en gemakkelijke manier is File Mapping. File mapping creëert een stuk gedeeld geheugen (shared memory) en staat het toe om met grote bestanden zoals een database te werken zonder dat dit bestand in zijn geheel in het geheugen geplaatst dient te worden.

Het besturingssysteem creëert hiervoor een 'File Mapping Object' om deze mapping in goede banen te leiden. Voor het bereiken van de gedeelde inhoud heeft een proces een 'File View' nodig. Een 'File View' is een stuk virtueel geheugen waaruit een proces de gedeelde gegevens kan bereiken. Processen lezen en schrijven via pointers. Deze pointers werken op dezelfde manier als dat deze toegepast worden in het dynamisch geheugen. De efficiëntie komt hier duidelijk uit naar voren omdat het bestand met gegevens op de harde schijf blijft staan en alleen de 'File view' zich in het geheugen bevind.

Op de onderstaande afbeelding is duidelijk te zien hoe de gegevens, het 'File Mapping Object' en de 'File Views' met elkaar in verband staan.



Figuur 14: De relatie tussen het bestand op de HDD, File Mapping Object en File View

Naast de optie om de inhoud van alle type bestanden te delen, kunnen gegevens ook gedeeld worden via de 'System Page File'. De System Page File is een verborgen bestand in de hoofdmap van de harde schijf welke door Windows als het RAM-geheugen gebruikt wordt.

4. In-house

In dit hoofdstuk wordt beschreven welke ontwerpkeuzes er zijn gemaakt. Dit hoofdstuk is in chronologische volgorde opgezet. Eerst is er een functionele beschrijving van de opdracht gemaakt. Daarna is aan de hand van deze functionele beschrijving gekeken hoe de afzonderlijke delen gerealiseerd kunnen worden. De verschillende opties die bekeken zijn worden beschreven, evenals de uiteindelijke keuzes en deze worden beargumenteerd.

4.1. Functionele beschrijving Siemuflex

4.1.1. Beschrijving interface

De interface die opgeleverd gaat worden met deze opdracht zal een Siemens S7 PLC verbinden met het simulatiepakket Flexsim. De interface krijgt de naam “Siemuflex”.

De achtergronden van Siemens en Flexsim staan beschreven in hoofdstuk 3 “Theorie”. De interface zendt en ontvangt data van de PLC en Flexsim en “vertaalt” deze zodat beide met de data om kunnen gaan. Door deze vertaling ontstaat er een emulatie.

De interface moet ervoor zorgen dat de PLC haar programma kan uitvoeren. Dit programma is voor een bepaald transportsysteem geschreven. Dit transportsysteem is dan niet fysiek aanwezig, maar zal gesimuleerd worden door Flexsim. Bij normaal gebruik van Flexsim zit alle intelligentie van de simulatie in Flexsim zelf, in de vorm van Flexscript. Deze intelligentie wordt eruit gehaald, en het PLC-programma neemt deze taak over.

De interface moet signalen over en weer kunnen zenden en vertalen tussen de PLC en Flexsim. Deze signalen bestaan uit - vanuit de PLC gezien – inputs en outputs.

4.1.1.1. Outputs van de PLC

De outputs van de PLC zullen cyclisch uitgelezen moeten worden. Deze worden ergens in een stuk computergeheugen geplaatst, waarschijnlijk in een soort array. In de volgende cyclus worden nieuwe output waarden opgehaald. Deze worden vergeleken met de vorige waarden. Als er een waarde van een output veranderd is moet deze doorgestuurd worden naar het corresponderende object in Flexsim.

4.1.1.2. Inputs naar de PLC

Bij een verandering in Flexsim van één van de inputs naar de PLC zal deze verandering vanuit Flexsim doorgestuurd moeten worden naar de interface. De interface zal ook voor de inputs een soort van array bevatten met actuele waarden. Bij een verandering van de waarde van een input zal deze vanuit de interface doorgestuurd worden naar de PLC.

4.1.2. De PLC

Aan de PLC zal zo min mogelijk aangepast moeten worden. Er wordt geprobeerd de interface zodanig te maken, en het echte transportsysteem zo te simuleren, dat de PLC zonder aanpassing van zijn programma kan functioneren. De hardware- en netwerkconfiguratie zullen waarschijnlijk wel voor elk project moeten worden aangepast. Er wordt naar gestreefd het PLC-programma volledig intact te laten.

4.1.3. Flexsim

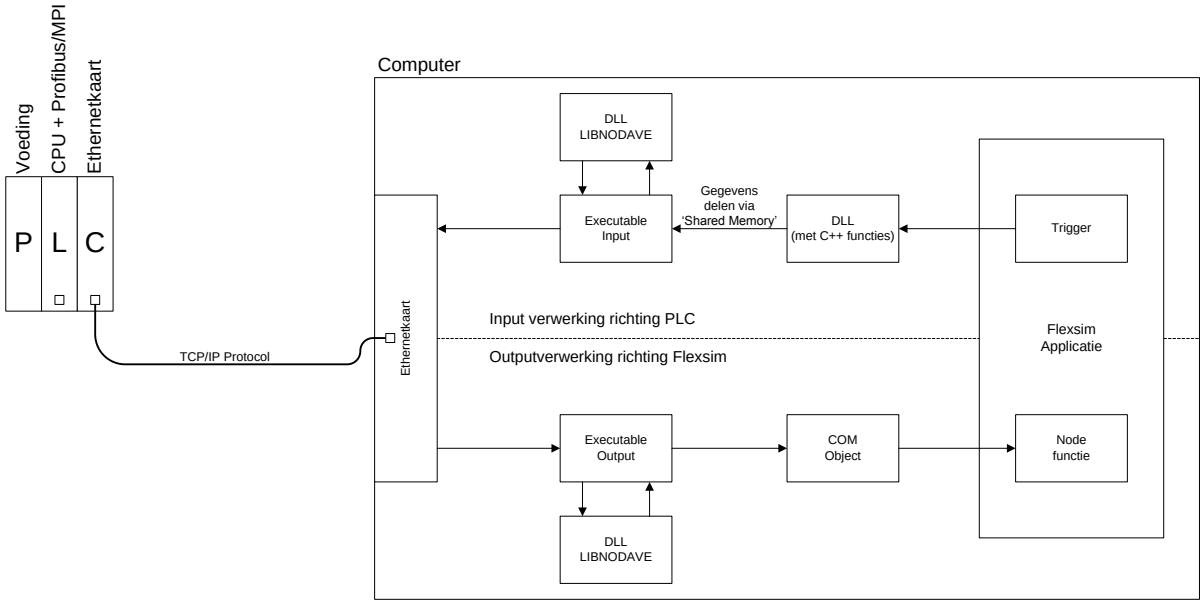
In Flexsim zal het echte transportsysteem nagebouwd worden. Voor de testopstelling met het TTI-project zal dit handmatig gebeuren. Bij een eventueel vervolg op de opdracht kan er gekeken worden of er een library gecreëerd kan worden waar standaard transportbanden en dergelijke in zitten.

4.1.4. Linken in- en outputs met Flexsim

De inputs naar en de outputs van de PLC, moeten worden gelinkt met de bijbehorende modelobjecten in Flexsim. Dit zullen voor de inputs voornamelijk sensoren zijn, en voor de outputs voornamelijk motoren. Het linken zal voor de testopstelling met het TTI-project hoogstwaarschijnlijk handmatig gebeuren. Bij een eventueel vervolg op de opdracht kan er gekeken worden of hier een automatische methode voor ontwikkeld kan worden.

4.2. Ontwerpkeuzes

Na het opzetten van de functionele beschrijving is er nagedacht over de verschillende onderdelen van de opdracht en is er onderzoek gedaan naar de beste optie voor elk onderdeel. Deze delen leverden samengevoegd een schematisch basis ontwerp dat hieronder wordt weergegeven. Het geheel van de onderdelen tussen de Ethernetkaart en Flexsim wordt “Siemuflex” genoemd.



Figuur 15: Basis ontwerp Siemuflex

De opdracht is door twee afstudeerders uitgevoerd. Omdat er veel uitgezocht moest worden is er besloten de taken te verdelen. Één afstudeerder heeft gekeken naar de PLC kant van de opdracht, en de andere afstudeerder heeft naar de Flexsim kant gekeken.
Na enig uitzoekwerk en enkele brainstormsessies kwam naar voren dat de inputs en de outputs twee verschillende verhalen zouden worden die elk een eigen aanpak vereisten. Daarom is besloten dit gescheiden te houden.

In de volgende paragrafen worden eerst de ontwerpkeuzes voor de algemene onderdelen besproken die niet specifiek voor in- of outputs zijn. Daarna worden afzonderlijk de ontwerpkeuzes voor de onderdelen van de inputs en de outputs besproken. Als laatste wordt de gebruikte programmeertaal besproken.

4.2.1. Algemeen

4.2.1.1. Type PLC

Zoals eerder vermeld werkt Van Riet voornamelijk met de S7-300 en S7-400 serie PLC’s van Siemens. Het lag daarom voor de hand één van deze types te kiezen voor de testopstelling. Siemuflex is voor beide typen PLC’s ontwikkeld, echter kon er maar op één type PLC getest worden.

De reden dat er voor de S7-400 serie is gekozen ligt in het feit dat de CPU in deze PLC hogere specificaties heeft dan de CPU in de S7-300 serie. PLC-programma’s die geschreven zijn voor een S7-300 zijn zonder probleem te gebruiken in een S7-400, terwijl dit andersom niet het geval is.

Mocht er in de toekomst een PLC-programma getest worden dat voor een S7-400 is geschreven, dan is zeker dat deze zonder problemen getest kan worden met een S7-400 PLC. Een PLC-programma geschreven voor een S7-300 kan na kleine aanpassingen in de hardware configuratie in een S7-400 gebruikt worden.

4.2.1.2. Communicatie tussen PLC en Siemuflex

Bij de communicatie tussen de PLC en Siemuflex is gekeken naar het communicatieprotocol en de fysieke verbinding.

Siemens heeft een softwarepakket genaamd Softnet DP slave. Met dit pakket kan een PC fungeren als een Profibus slave. De PLC is Profibus master. Bij het pakket wordt een OPC server geleverd.
Aanvankelijk is er gestart met dit pakket. Dit is gedaan omdat met dit pakket de in- en outputs die benodigd zijn zeer eenvoudig te bereiken zijn in de PLC. Er zou dan alleen nog een OPC client geprogrammeerd moeten worden die de informatie bij de OPC server ophaalt en wegschrijft. Ook hoeft er aan het PLC-programma niets aangepast te worden, wat zeer gewenst is. Daarnaast is Softnet DP Slave een relatief goedkoop pakket.
Een belangrijk onderdeel van het project is het zo snel mogelijk kunnen communiceren. Hier is in het begin van de opdracht onderzoek naar gedaan, en er is met Siemens hierover gesproken. De snelheid leek voldoende.

Na aanvullend onderzoek kwam echter naar voren dat de minimale cyclustijd van de OPC server die door Siemens meegeleverd wordt 100 ms bedraagt. Dit is voor de toepassing van deze opdracht te hoog. Daarom is besloten niet meer met de OPC server verder te gaan.

De definitieve keuze is gevallen op Libnodave. Dit is gedaan omdat de functies die voor dit project nodig zijn, zoals inputs schrijven en outputs lezen, allemaal in de bibliotheek zitten. Ook hoeft in het geval van Libnodave niets aan het PLC-programma aangepast te worden.
Daarbij is Libnodave vrijelijk verspreidbare software onder de GNU General Public License (vrijelijk moet gezien worden als vrij van rechten). Software onder deze licentie is doorgaans ook gratis, en dat is in het geval van Libnodave ook zo.

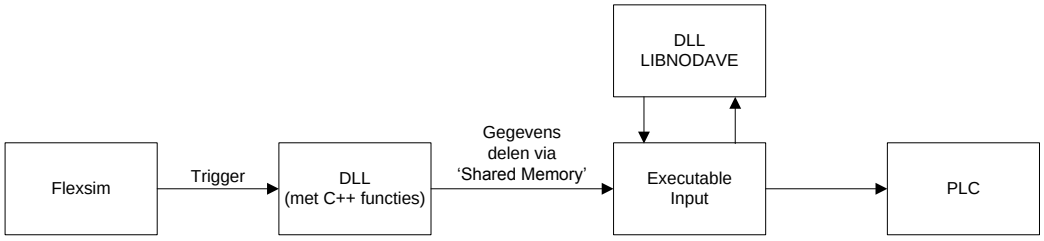
Libnodave kan via twee typen fysieke verbinding communiceren met een PLC. Dit zijn MPI en Ethernet. Er is om twee redenen gekozen voor Ethernet.

1. Ethernet is sneller dan MPI. De gebruikte Ethernet kaart van Siemens (CP443) heeft een maximale snelheid van 100 Mb/s, terwijl MPI maximaal 12 Mb/s haalt.
2. Voor MPI is een speciale adapter nodig die niet goedkoop is. Een Ethernet aansluiting zit op elke moderne laptop en desktop PC.

4.2.2. Inputs

4.2.2.1. Basis ontwerp inputs

Het basis ontwerp van de input verwerking ziet er als volgt uit:



Wanneer in Flexsim een sensor (Photo eye) wordt getriggerd zullen deze signalen worden doorgestuurd naar de PLC
Stap1: Flexsim roept de DLL aan
Stap2: De DLL reageert op Flexsim en zal de gegevens via shared memory in een array opslaan
Stap3: De executable vergelijkt cyclisch de gegevens uit het array met de vorige cylus en kopieert deze gegevens bij verandering naar de process image input

Figuur 16: Input verwerking

4.2.2.2. Flexsim → DLL

Er zijn verschillende manieren om in Flexsim functies aan te roepen als er een statusverandering (trigger) binnen een bepaald modelobject is. In de eigenschappen van het object kan code geplaatst worden. Deze code kan C++ code zijn, Flexscript (een afgeleide van C++), of er kan een DLL aangeroepen worden. Na onderzoek op de Flexsim site bleek dat een DLL aanroepen de beste keuze was omdat dit de hoogste snelheid heeft.

Daarom is besloten zelf een DLL te schrijven. Deze DLL bevat functies die de statusgegevens van de objecten in een array in het computergeheugen plaatst.

4.2.2.3. DLL → executable

Zonder speciale maatregelen is hetzelfde computergeheugen maar door één proces te benaderen. Mochten er twee processen toegang willen verkrijgen tot dezelfde geheugenlocatie, dan is die alleen toegankelijk voor het proces die het geheugen aangevraagd heeft. Het tweede proces krijgt een zogenoemde "Memory acces violation" foutmelding. Vanwege deze eigenschap moest er een manier gevonden worden om de gegevens vanuit de DLL in een ander proces – de Siemuflex input executable – te kunnen lezen. Er diende een gedeeld stuk geheugen gecreëerd te worden, het zogenaamde "shared memory".

Er is onderzoek gedaan naar manieren om in Windows geheugen te delen. Er bleken een aantal opties te bestaan voor communicatie tussen processen. Dit wordt door Microsoft "InterProcess Communication" genoemd. De volgende technieken zijn beschikbaar:

- Clipboard
- COM
- Data Copy
- DDE
- File Mapping
- Mailslots
- Pipes
- RPC
- Windows Sockets

Na bestudering van de eigenschappen van de verschillende technieken bleek dat File Mapping de beste optie was, omdat het een makkelijke en efficiënte manier is voor twee of meer processen om data te delen. De grootte van de data heeft geen invloed op de efficiëntie waardoor uitbreiding zeer eenvoudig is. De andere methoden waren vaak voor andere doeleinden ontworpen dan data delen en/of zeer uitgebreid en daardoor lastig te doorgronden.

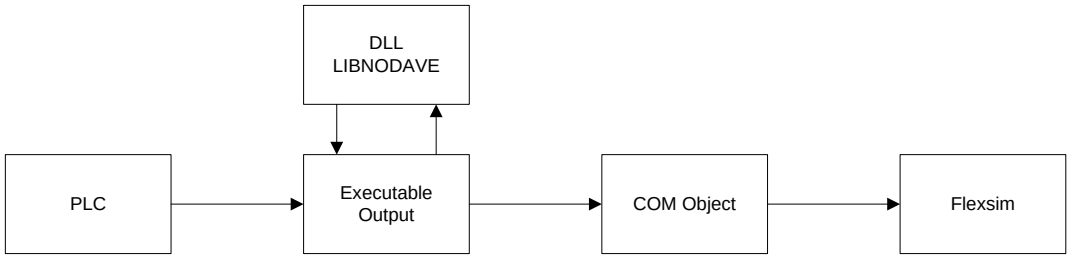
4.2.2.4. Executable → PLC

De wijze van communicatie tussen de executable en de PLC is voor de inputs en outputs gelijk. Daarom wordt deze keuze beschreven in het algemene deel, in paragraaf 4.2.1.2 “Communicatie tussen PLC en Siemuflex”.

4.2.3. Outputs

4.2.3.1. Basis ontwerp outputs

Het basis ontwerp van de input verwerking ziet er als volgt uit:



Wanneer er een verandering in de outputs plaats vindt zullen deze doorgegeven worden aan Flexsim
Stap1: De executable haalt cyclisch de process image output op uit de PLC en slaat deze in een array op.
Stap2: De executable kijkt of er veranderingen zijn t.o.v. de vorige cyclus en roept bij verandering het COM object aan
Stap3: Het COM object roept de juiste functie in Flexsim aan

Figuur 17: Output verwerking

4.2.3.2. PLC → executable

De wijze van communicatie tussen de PLC en de executable is voor de inputs en outputs gelijk. Daarom wordt deze keuze beschreven in het algemene deel, in paragraaf 4.2.2.5 “Communicatie tussen PLC en Siemuflex”.

4.2.3.3. Executable → COM object → Flexsim

Er moest een manier gevonden worden om functies in modelobjecten binnen Flexsim aan te roepen. Flexsim is oorspronkelijk niet ontworpen om van buiten de applicatie benaderd te worden. Op het forum van Flexsim wordt een COM object aangeboden dat kan communiceren met Flexsim. Via dit COM object kunnen functies rechtstreeks worden aangeroepen. Dit is een belangrijke eigenschap, omdat dit een vereiste is voor Siemuflex. Ook is het eenvoudig om vanuit een in C++ geschreven executable een COM object aan te roepen.

4.2.4. Programmeertaal

Er waren meerdere opties voor de programmeertaal waarin de executables geschreven konden worden. Libnodave, de gebruikte bibliotheek voor communicatie richting PLC, bevat functies in een DLL die vanuit verschillende programmeertalen te bereiken zijn. Dit zijn:

- C
- C++
- C#
- Delphi
- Pascal
- Visual Basic

Het gebruikte Component Object Model (COM) is zo ontworpen dat het programmeertaalonafhankelijk is. Het is zelfs de essentie van COM, een platform- en programmeertaalonafhankelijke standaard.

Ook de DLL die vanuit Flexsim wordt aangeroepen wanneer er een input verandert kan in diverse programmeertalen geschreven worden.

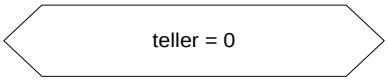
De keuze voor de programmeertaal is gevallen op C++. Deze taal is gekozen omdat de afstudeerders die het project hebben uitgevoerd onderwezen zijn in C++ en deze taal hun dus bekend is. Daarnaast zijn voor de bibliotheek Libnodave, gebruikt voor de communicatie tussen de PLC en Siemuflex, een aantal voorbeelden beschikbaar die geschreven zijn in C++. Deze voorbeelden zorgen ervoor dat Libnodave eenvoudiger te begrijpen is, aangezien de documentatie gebrekkig is.

4.3. Basis software ontwerp

4.3.1. Inleiding

Nadat de ontwerpkeuzes en de basis ontwerpen van de in- en outputs waren gemaakt is er een basis ontwerp van de software gemaakt. Daarna is de software geschreven en getest. De uiteindelijk geschreven software staat beschreven in hoofdstuk 5 “Realisatie”. Voor het begrijpen van deze flowcharts en flowcharts in volgende hoofdstukken is het handig om de betekenis van de figuren te kennen. Daarom hier een korte uitleg van de gebruikte figuren:

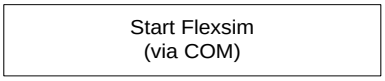
4.3.1.1. Voorbereiding/initialisatie



Figuur 18: Initialisatie blok

In deze blokken worden (standaard) waarden aan variabelen toegekend. Deze variabelen zijn dus niet direct afhankelijk van het proces.

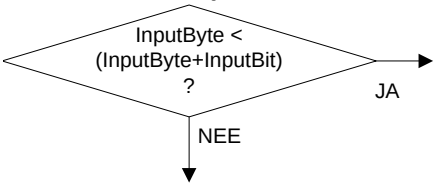
4.3.1.2. Proces/actie



Figuur 19: Proces blok

In deze blokken worden acties uitgevoerd. Deze acties kunnen zeer divers zijn.

4.3.1.3. Keuze/voorwaarde



Figuur 20: Keuze blok

In deze blokken worden ‘vragen’ gesteld. De vraag wordt altijd met JA of NEE beantwoord.

4.3.1.4. Symbolenlijst

Hier volgt een kleine verklarende symbolenlijst:

Symbool	Verklaring
x == y	x gelijk aan y
x > y	x groter dan y
x < y	x kleiner dan y
x != y	x is ongelijk aan y

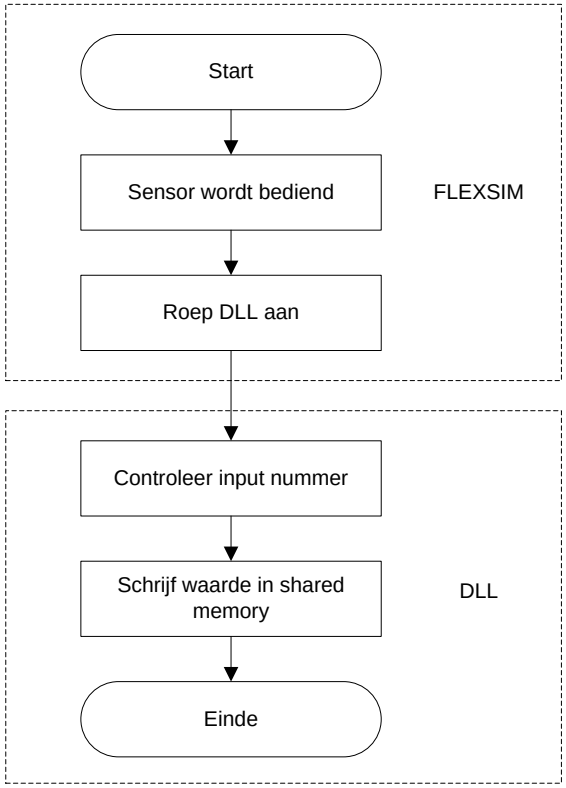
Symbool	Verklaring
=	toekenning
x ++	x = x + 1

Tabel 6: Flowchart symbolenlijst

4.3.1.5. Verdieping

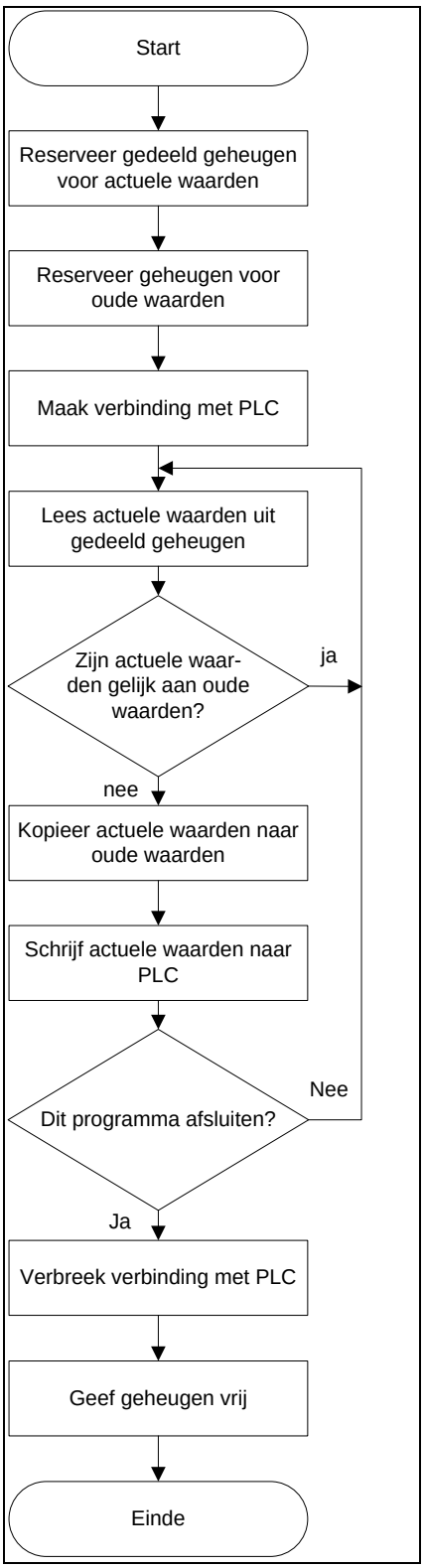
Wanneer verdieping gewenst is zal dit worden aangegeven met het volgende symbool: (2.1)
Verderop in het document zal dan meer informatie te vinden zijn corresponderend met het ingevulde nummer.

4.3.2. Input DLL



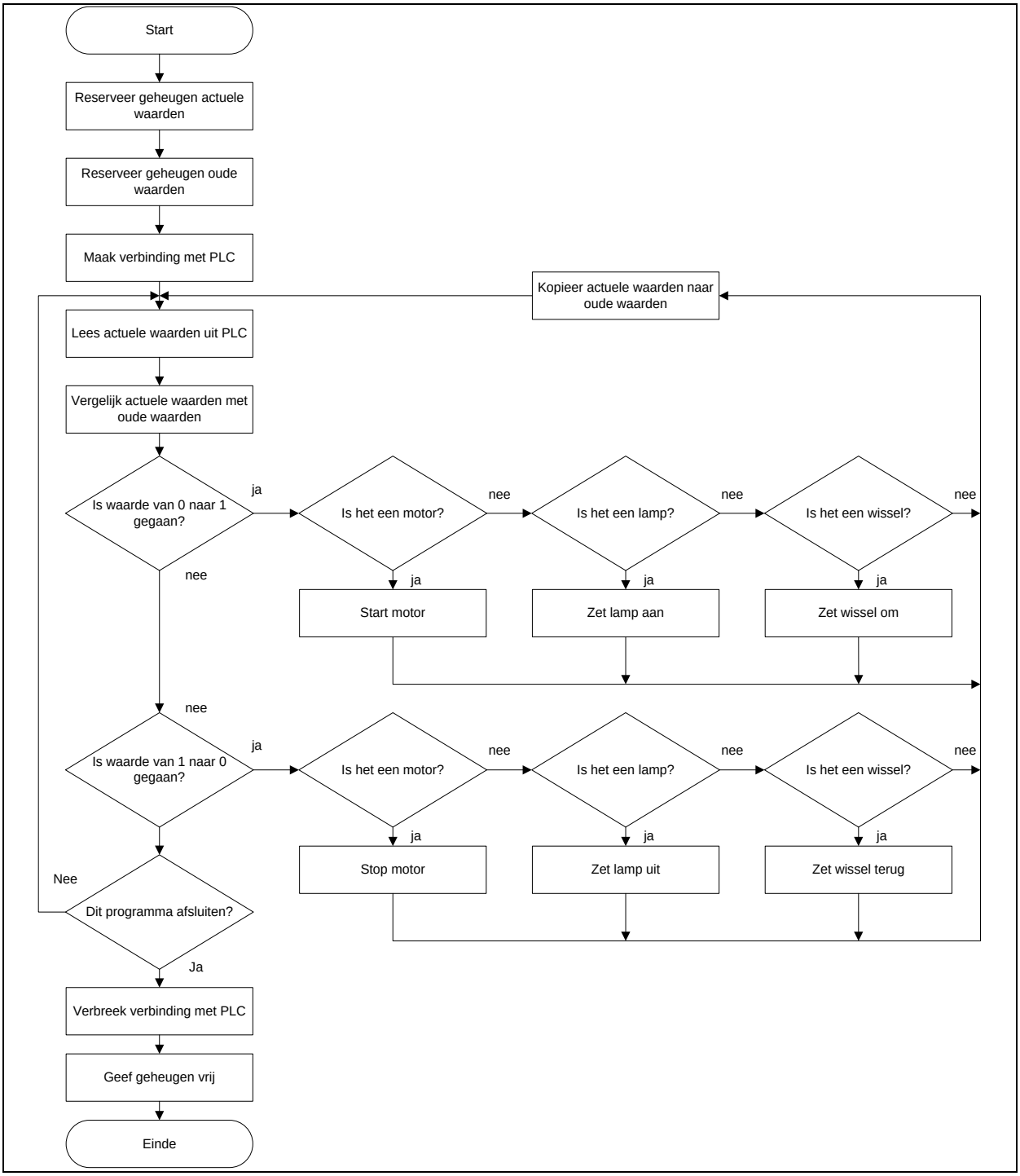
Figuur 21: Software ontwerp DLL

4.3.3. Input executable



Figuur 22: Software ontwerp input executable

4.3.4. Output executable



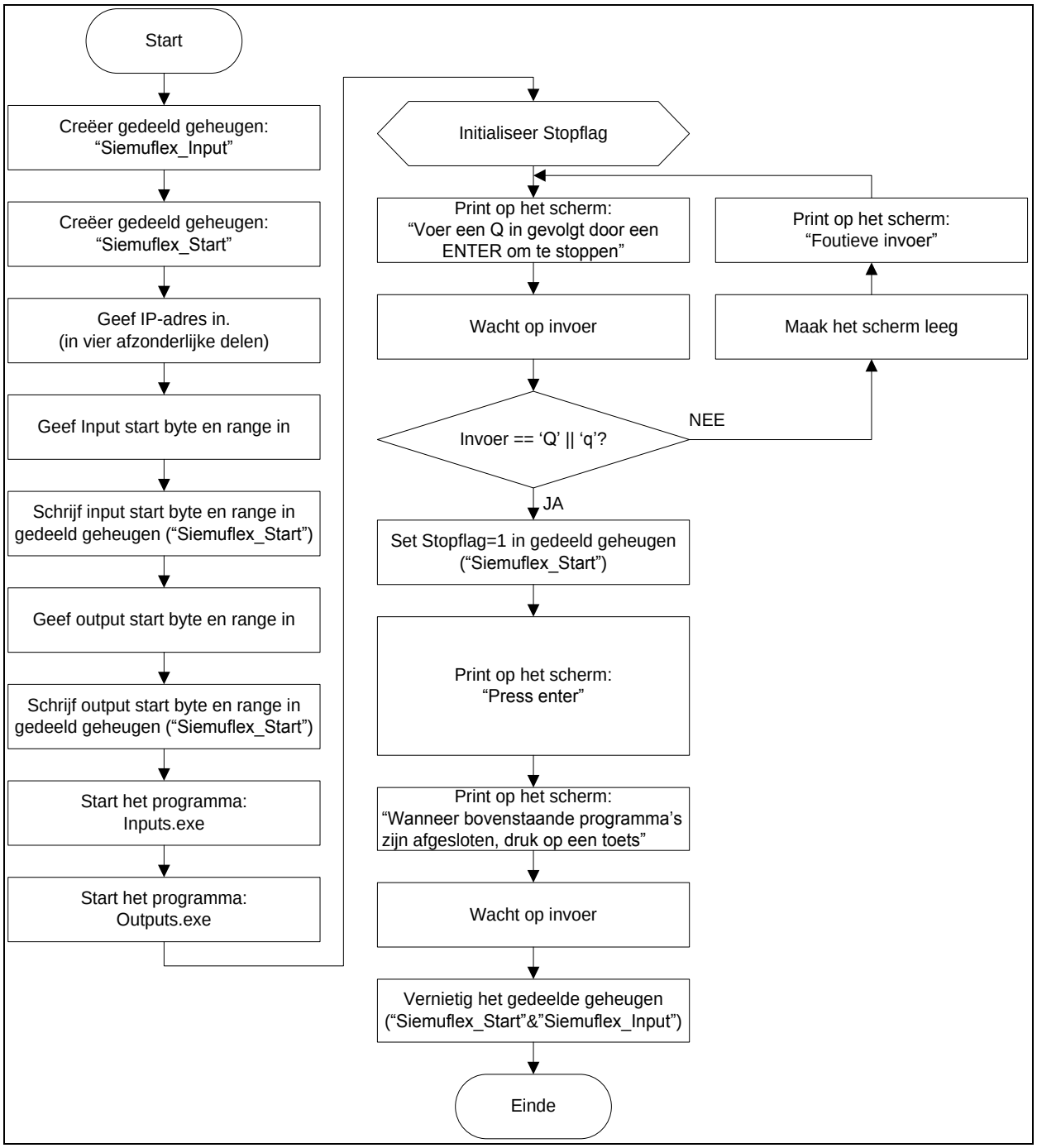
Figuur 23: Software ontwerp output executable

5. Realisatie

Zoals in het voorgaande hoofdstuk te lezen is, is Siemuflex voornamelijk een softwarematige opdracht. De gerealiseerde software zal in dit hoofdstuk worden verklaard doormiddel van flowcharts.

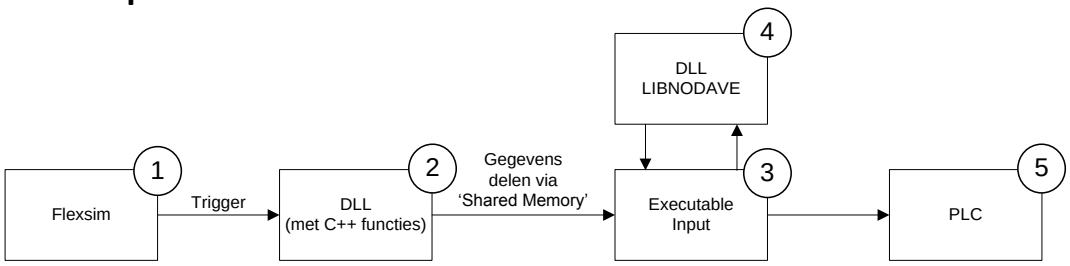
5.1. Overkoepelend programma (Start.exe)

Om het gebruik van Siemuflex te vereenvoudigen is er een overkoepelend programma (Start.exe) ontwikkeld. Dit programma verzorgt de start en stop van de andere programma's en de creatie en vernietiging van het shared memory . Door de toepassing van deze executable hoeven de gezamenlijke gegevens maar in een venster ingevoerd te worden waardoor het een overzichtelijker geheel wordt. Hier onder staat de flowchart van dit programma:



Figuur 24: Flowchart Start.exe

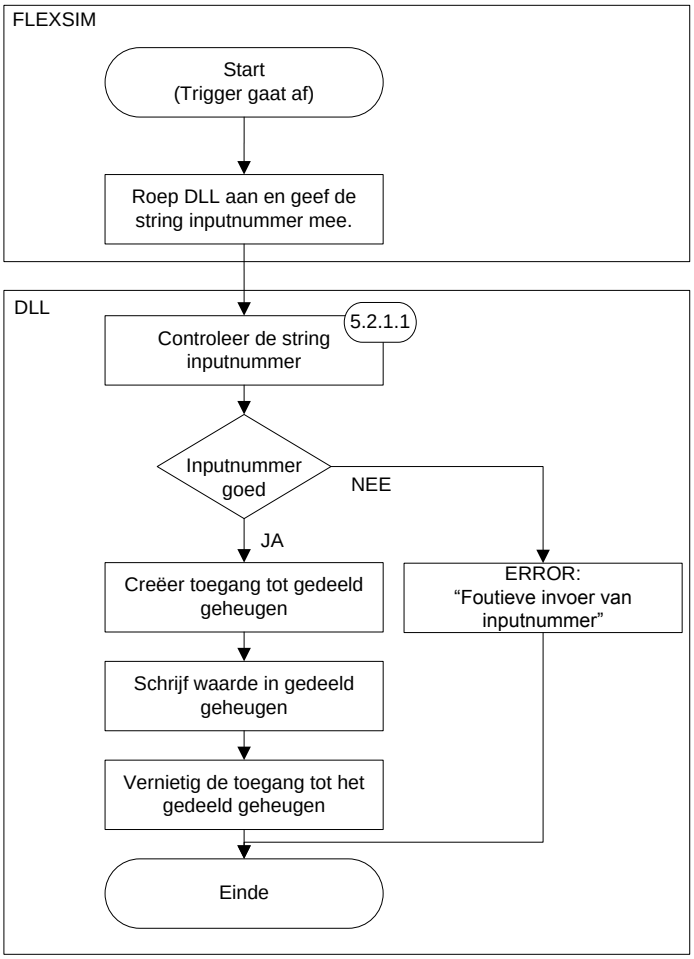
5.2. Inputs



Figuur 25: Input verwerking

5.2.1. Flexsim (1) en de DLL (2)

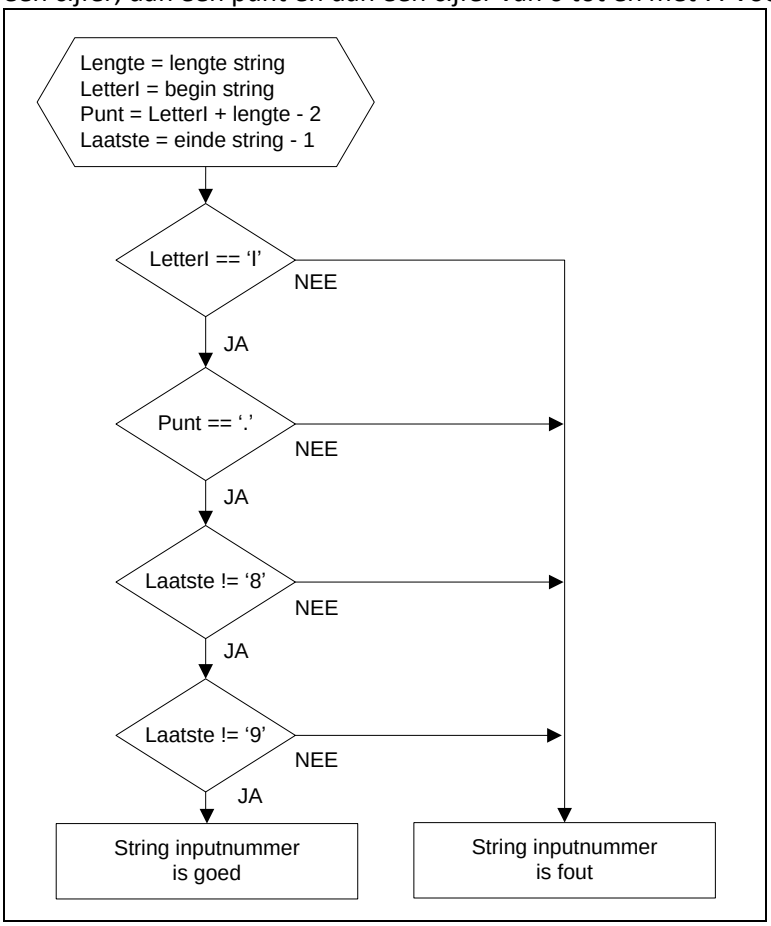
Wanneer binnen Flexsim een trigger afgaat, zal het volgende gebeuren (afhankelijk van de trigger, OnCover of OnDeCover, zal er een 1 of een 0 weggeschreven worden):



Figuur 26: Flowchart Flexsim (1) en de DLL (2)

5.2.1.1. Controle string inputnummer

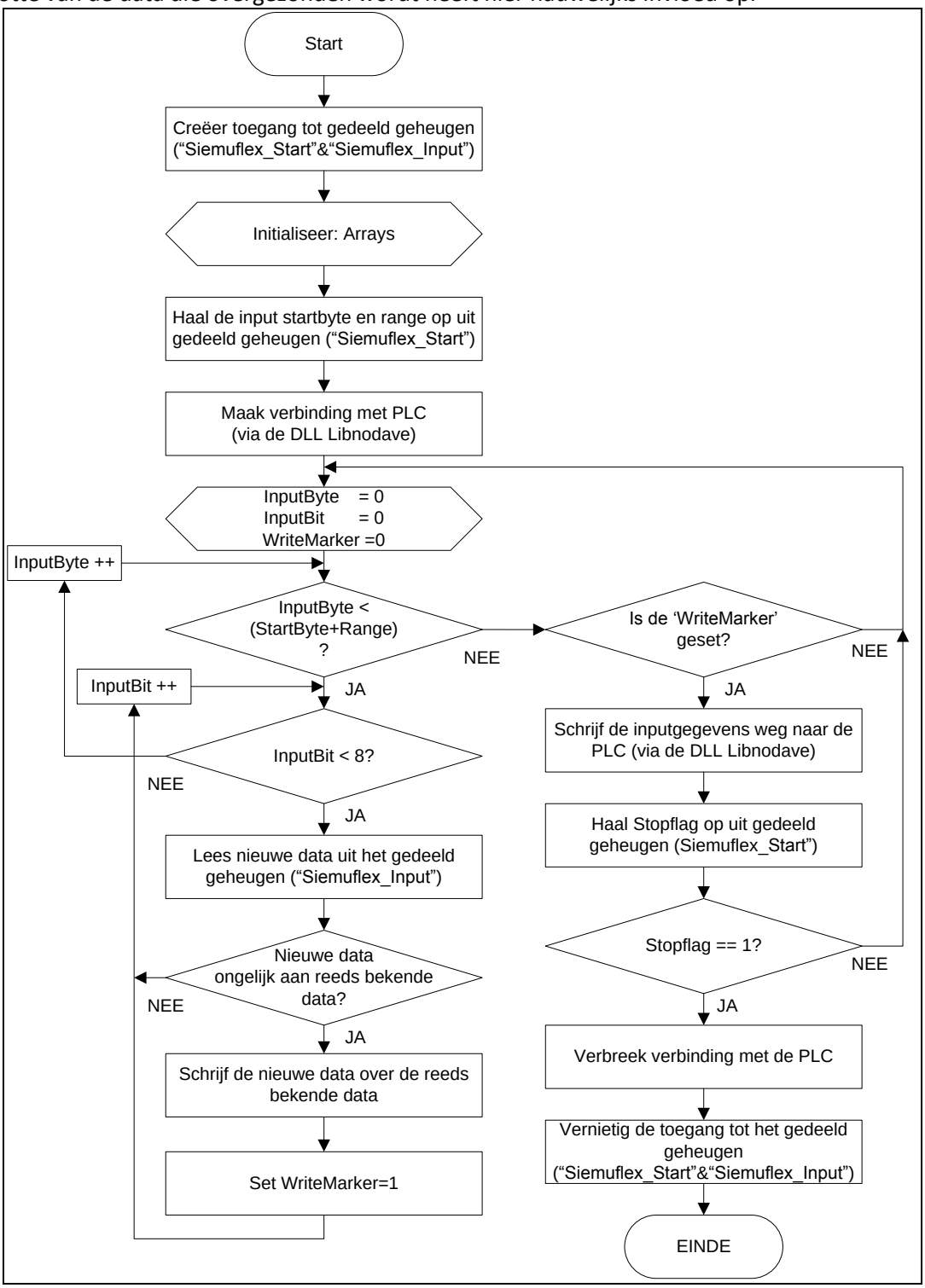
De actie 'controleer string inputnummer' wordt aangeroepen om er zeker van te zijn dat een input binnen Flexsim de juiste benaming heeft gekregen. Dit is belangrijk omdat het setten en resetten van een input richting PLC hier afhankelijk van is. Het inputnummer moet beginnen met een I, gevolgd door een cijfer, dan een punt en dan een cijfer van 0 tot en met 7. Voorbeeld: I12.6.



Figuur 27: Flowchart controle string inputnummer

5.2.2. Executable Input (3) en DLL Libnodave (4)

Deze executable kijkt of er veranderingen zijn, zo ja, dan worden deze naar de PLC gestuurd. De volledige range wordt in één keer over gezonden naar de PLC, en niet direct per verandering. De keuze om alles in een keer over te zenden is gemaakt omdat het versturen van de TCP-IP pakketjes tijd kost. De grootte van de data die overgezonden wordt heeft hier nauwelijks invloed op.

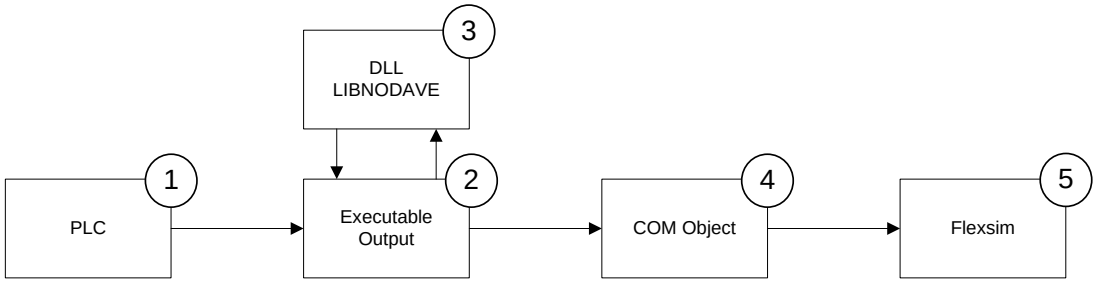


Figuur 28: Flowchart executable (3) en Libnodave DLL (4)

5.2.3. PLC (5)

Hier worden de inputgegevens naar toe geschreven. Via Libnodave wordt dit direct in de 'process image input (PII)' geplaatst. De PLC merkt hier niets van.

5.3. Outputs

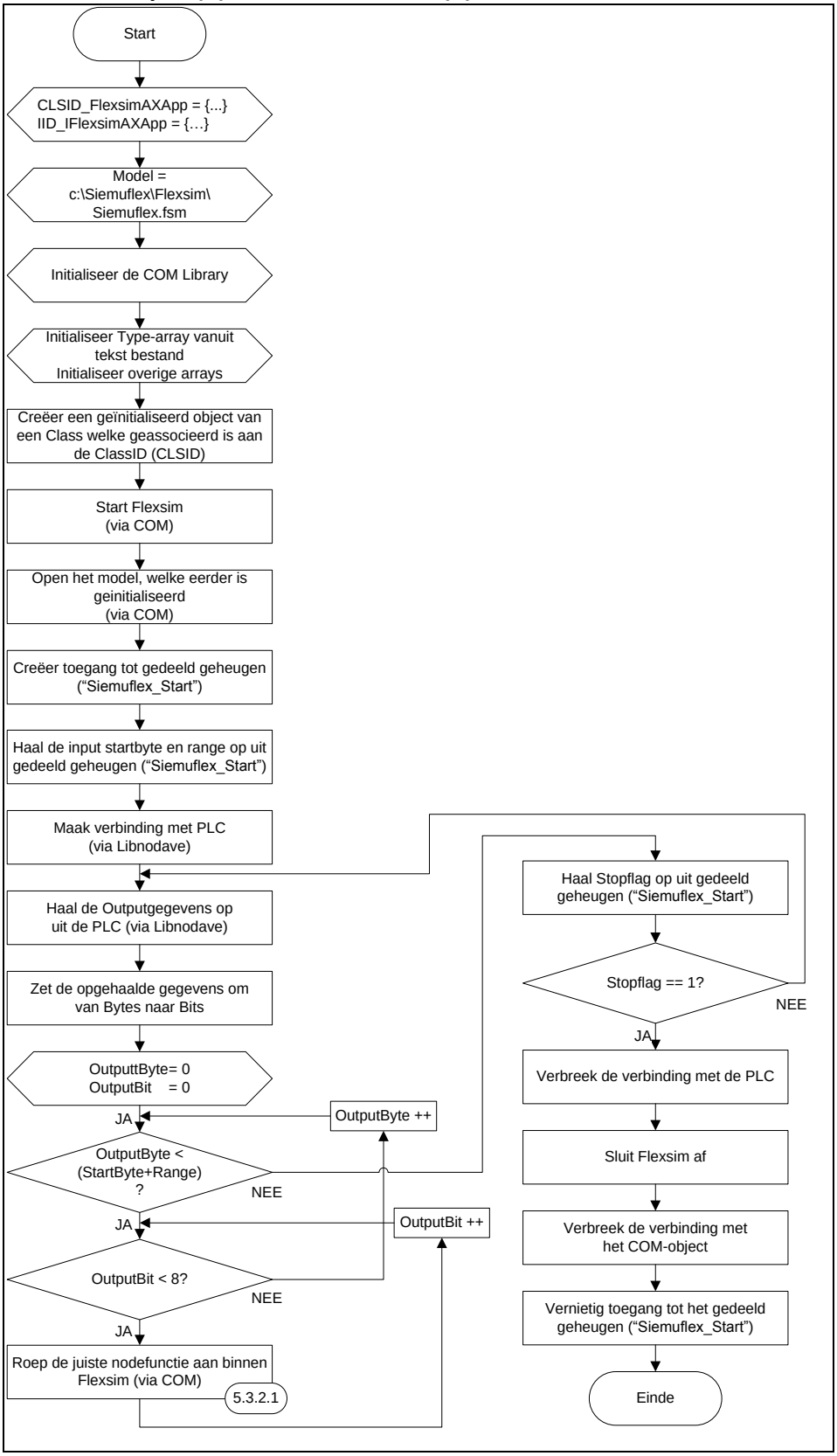


Figuur 29: Output verwerking

5.3.1. PLC (1)

Net als bij de inputs, zal in de PLC-software geen verandering doorgevoerd hoeven te worden. Libnodave (welke vanuit de executable Output (2) wordt aangeroepen) zal de gegevens ophalen.

5.3.2. Executable Output (2) en DLL Libnodave (3)



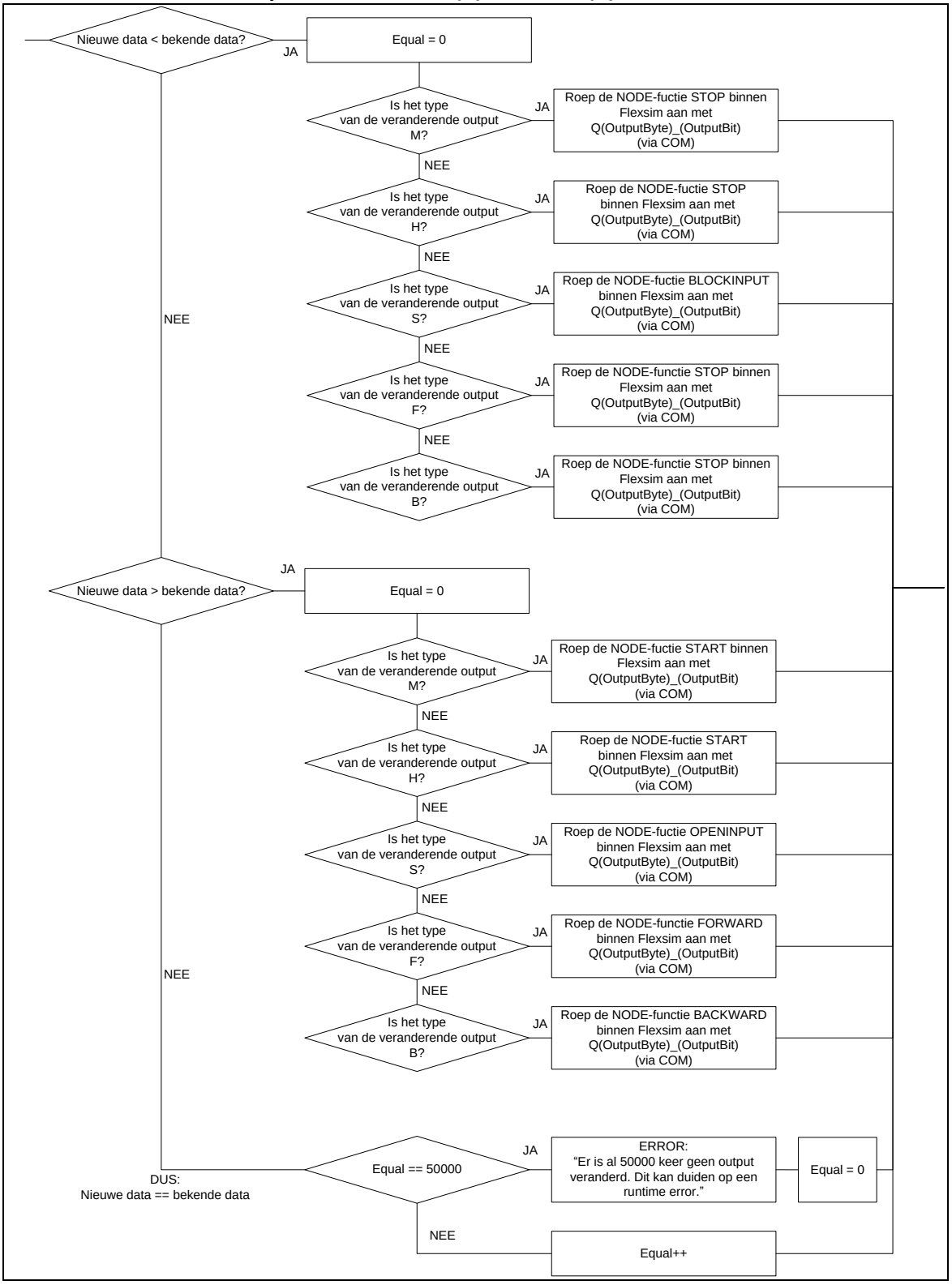
Figuur 30: Flowchart executable (2) en Libnodave DLL (3)

De executable Output maakt voor communicatie met Flexsim geen gebruik van shared memory zoals de executable Input. Echter maakt executable Output gebruik van een COM object. Zoals te lezen is bij de gemaakte keuzes is dit COM object gemaakt door een andere Flexsim gebruiker.

Net als bij het shared memory dient er ook een verbinding tussen de executable en het COM object tot stand te worden gebracht.

Wanneer deze verbinding tot stand is gebracht kunnen functies van de IFlexsimAXApp worden aangeroepen.

5.3.2.1. Functie aanroep binnen Flexsim (5) via COM (4)



Figuur 31: Flowchart functie aanroep binnen Flexsim (5) via COM (4)

Afhankelijk van het toegekende type zal de output de hierbij horende functie aanroepen.

5.3.2.2. Stabiliteitscheck

Wanneer er geen verandering tussen de nieuwe en oude waarden is hoeft er ook geen verandering binnen Flexsim doorgevoerd te worden. Uit testen is gebleken dat wanneer Flexsim op een te hoge snelheid draait het kan voorkomen dat Flexsim de COM opdrachten mist en ook niet meer oppakt. In de executable Output zit een teller ingebouwd die bijhoudt hoe vaak er geen verandering voorkomt. Als er vijftig keer geen output verandert, zal er een melding op het scherm verschijnen. Deze geeft een indicatie van deze runtime error. Op bovenstaande flowchart is dit onderin weer gegeven.

5.3.2.3. Nodefuncties

Zoals te zien is in bovenstaande afbeelding, wordt een nodefunctie aangeroepen wanneer er een verandering plaatsvindt. Alle nodefuncties zijn vooraf gedefinieerd binnen Flexsim en later beschreven in bijlage G. De standaard Siemuflex conveyor binnen de Siemuflex Library bevat deze functies al. De aanroep van deze functies is afhankelijk van het output nummer, deze dienen dus goed te zijn ingevoerd binnen Flexsim en wel op de volgende manier: Qx_y, waarbij x en y variabel zijn, let wel: y mag alleen de waarden 0 t/m 7 bevatten.

Voor vrijwel alle objecten gelden de bovenstaande regels. Maar wanneer een lopende band in twee richtingen kan bewegen levert dit een probleem op. Binnen Flexsim heeft een lopende band één naam, maar binnen de PLC programmeer omgeving wordt een dergelijke lopende band met twee verschillende outputs aangestuurd. Om dit probleem op te vangen, is er de keus gemaakt om dit binnen de Output executable te doen. De volgende aannamen zijn gedaan:

- Wanneer een output van het type F veranderd wordt de functie FORWARD of STOP aangeroepen met hetzelfde outputnummer.
- Wanneer een output van het type B veranderd wordt de functie BACKWARD of STOP aangeroepen met het outputnummer Qx_y−1.

Kortom, men dient er altijd voor te zorgen dat de outputs voor vooruit en achteruit direct na elkaar in de PLC geprogrammeerd worden en wel in die volgorde!

6. Gebruik en onderhoud

In dit hoofdstuk wordt uitgelegd hoe Siemuflex gebruikt kan worden en hoe Siemuflex aangepast/onderhouden kan worden.

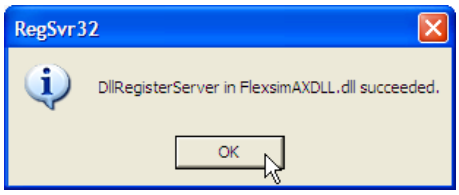
6.1. Gebruikershandleiding

6.1.1. Installatie

Siemuflex wordt geleverd als een verzameling van bestanden in een map genaamd “Siemuflex”. Deze map bevat alle benodigde bestanden om Siemuflex te installeren, uit te voeren en eventueel uit te breiden. Voor installatie is één handeling vereist. Voer het bestand **Install.bat** uit. Er worden bestanden naar de Flexsim installatie directory gekopieerd.

NB: Flexsim moet in de standaard installatie map geïnstalleerd zijn. Dit is C:\Program Files\Flexsim4\.

Zodra de bestanden zijn gekopieerd wordt het bestand “FlexsimAXDLL.dll” geregistreerd. De volgende melding verschijnt:



Figuur 32: Registratie FlexsimAXDLL.dll

6.1.2. Voor gebruik Siemuflex

6.1.2.1. PLC gebruiksklaar maken

Om Siemuflex te kunnen gebruiken zal de hardware configuratie van de PLC van het S7 project dat getest gaat worden aangepast moeten worden. De hardware configuratie moet aangepast worden omdat modules die in het echte transportsysteem aan de PLC verbonden zijn er tijden het testen op kantoor niet zullen zijn. Dit zijn normaliter de input en output modules (I/O). Deze zullen uit de hardware configuratie verwijderd moeten worden. Ook kan het zijn dat er voor het testen een ander type PLC/CPU wordt gebruikt dan in het echte transportsysteem, omdat deze bijvoorbeeld op kantoor beschikbaar is en de PLC uit het echte transportsysteem niet. In het S7 project zal dan het type PLC/CPU aangepast moeten worden.

De PLC dient met de PC te worden verbonden via een Siemens Ethernet kaart. Het IP adres voor de PLC kan vrij gekozen worden. Bij de start van Siemuflex dient het IP adres ingegeven te worden. Zorg wel dat het IP adres van de PC in dezelfde range zit als het IP adres van de Siemens Ethernet kaart.

6.1.2.2. Siemuflex gebruiksklaar maken

Voordat Siemuflex gestart kan worden moeten de types van de PLC-outputs gedefinieerd worden zodat de juiste functies binnen Flexsim aangeroepen kunnen worden. Dit moet op dit moment handmatig gebeuren. Ga naar de map “Siemuflex”. Open de snelkoppeling naar het bestand Typelijst.txt. Het programma kladblok opent. Typelijst.txt is een simpel tekstbestand. Per regel wordt een output gedefinieerd en ziet er ongeveer als volgt uit: Qx.y[spatie]type

Op dit moment zijn de volgende typen beschikbaar:

- M Motor
- F Bidirectionele motor, forward
- B Bidirectionele motor, backward
- H Lamp
- S Toegangsblokkade van een transportband

Zie het volgende stukje tekstbestand waar Q0.0 en Q0.1 motoren, Q5.0 en Q5.1 lampen en Q10.0 en Q10.1 de aansturingen voor één bidirectionele motor (FORWARD en BACKWARD) zijn.

```
Q0.0 M
Q0.1 M
Q5.0 H
Q5.1 H
Q10.0 F
Q10.1 B
```

Figuur 33: Voorbeeld Typelijst.txt

Nadat de types zijn gedefinieerd kan het document op normale wijze opgeslagen worden. Indien naar verwachting hetzelfde project nog een keer getest gaat worden kan het handig zijn om het tekstbestand ook ergens anders op te slaan met “opslaan als”.

NB. Zorg dat er geen overbodige tekens of spaties in het tekstbestand staan, dit kan problemen opleveren.

6.1.2.3. Flexsim model bouwen

Alvorens er getest kan worden moet een systeem gemodelleerd worden in Flexsim. Bij het bouwen van het Flexsim model moeten een aantal regels in acht genomen worden.

- De namen van de transportbanden moeten overeenkomen met de uitgang van de PLC waaraan deze verbonden zijn. De punt moet vervangen worden door een underscore (‘_’). Voorbeeld:
  De transportband die wordt aangestuurd door Q0.1 wordt Q0_1.
- In het geval van een bidirectionele motor krijgt de lopende band de naam van de output die de motor FORWARD aanstuurt. De BACKWARD moet in de PLC de direct opvolgende output zijn. Voorbeeld:
  - Output Q1.0 is FORWARD.
  - Output Q1.1 is dezelfde motor maar dan BACKWARD.
  - ➢ Dan moet de transportband de naam Q1_0 krijgen.
- De namen van de photoeyes (sensoren) in Flexsim moeten overeenkomen met de inputnummers waarmee deze verbonden zijn (hier moet de punt *niet* vervangen worden door een underscore).

6.1.3. Siemuflex starten/gebruik

Voordat Siemuflex gestart kan worden moet er aan een aantal vereisten worden voldaan.

- Flexsim 4 moet geïnstalleerd zijn in de standaard directory.
- Siemuflex moet geïnstalleerd zijn.
- Er moet een PLC verbonden zijn met de PC (netwerk) via Ethernet.
- De PLC moet een werkend programma bevatten en in de RUN modus staan.

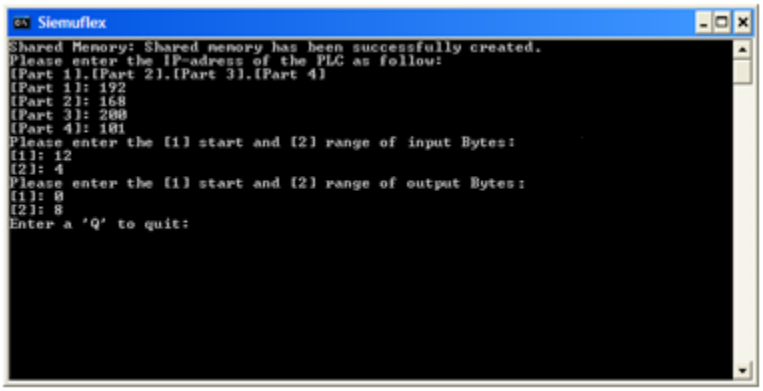
Siemuflex kan gestart worden door op de snelkoppeling “Siemuflex” in de hoofdmap van Siemuflex te dubbelklikken. Er start één executable (Start.exe). In het venster worden meldingen gegeven over het shared memory en daarna de volgende vragen gesteld:

Start executable:

- *Please enter the IP-address as follow:*
  [Part 1].[Part 2].[Part 3].[Part 4]
  Vul hier het IP adres van de Siemens Ethernet kaart in vier delen zonder punten in.

- *Please enter the [1] start and [2] range of input Bytes:*
  [1] Vul hier de start Byte in van de inputs. Als bijvoorbeeld de input range van I12.0 tot I16.0 loopt is deze 12.
  [2] Vul hier het aantal input Bytes dat aan het project verbonden is in. In dit voorbeeld is deze 4.

- *Please enter the [1] start and [2] range of output Bytes:*
  [1] Vul hier de start Byte in van de outputs. Als bijvoorbeeld de output range van Q0.0 tot Q8.0 loopt is deze 0.
  [2] Vul hier het aantal output Bytes dat aan het project verbonden is in. In dit voorbeeld is deze 8.

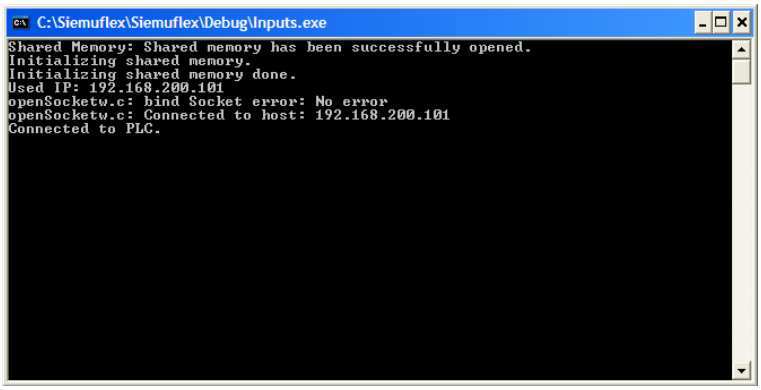


Figuur 34: Start executable

Nu starten Inputs.exe en Outputs.exe.

Input executable:

Er worden enkele meldingen gegeven over het shared memory en het gebruikte IP adres. Daarna wordt verbinding gemaakt met de PLC. Als dit geslaagd is verschijnt de melding “Connected to PLC”. Kan Siemuflex geen verbinding tot stand brengen verschijnt er de volgende foutmelding: “Error: Could not connect to PLC”. Of wanneer geen toegang tot het shared memory verkregen kan worden verschijnt dit ook in beeld.

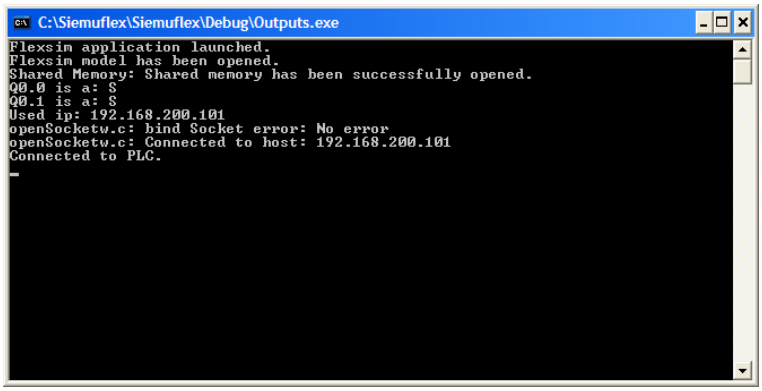


Figuur 35: Input executable

De Input executable is klaar voor gebruik.

Output executable:

Flexsim wordt gestart. Het Flexsimmodel “Siemuflex.fsm” wordt automatisch geladen. Flexsim minimaliseert. In het venster wordt gemeld dat Flexsim is gestart en het model is geopend. Daarna wordt gemeld dat het shared memory is geopend.
De types van de outputs die ingelezen zijn vanaf het Typelijst.txt bestand worden weergegeven en ook het gebruikte IP adres wordt weergegeven als extra controlemogelijkheid.
Nu wordt verbinding gemaakt met de PLC. Als dit geslaagd is verschijnt de melding “Connected to PLC”. Kan Siemuflex geen verbinding tot stand brengen verschijnt er de volgende foutmelding: “Error: Could not connect to PLC”.



Figuur 36: Output executable

De Output executable is klaar voor gebruik.
De simulatie in Flexsim kan nu gestart worden door op run 

te klikken.

6.2. Onderhoudshandleiding

6.2.1. Siemuflex compileren

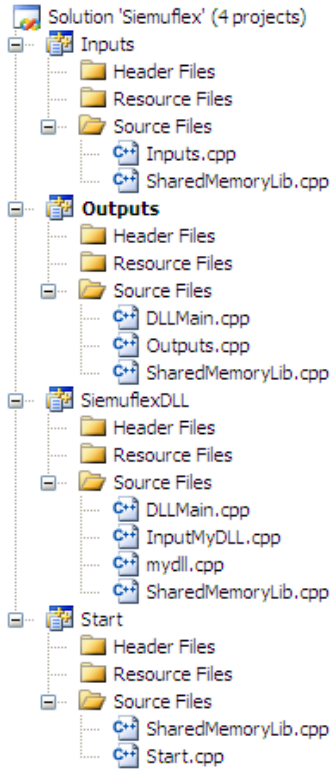
Om Siemuflex eenvoudig te kunnen compileren moeten de volgende software pakketten geïnstalleerd zijn:

- Microsoft Visual Studio (Express Edition).
- Microsoft Software Development Kit (SDK).

Siemuflex bestaat gecompileerd uit meerdere executables en een DLL. Dit zijn:

- *Start.exe*
  Deze executable roept de executables Inputs.exe en Outputs.exe aan en vormt de gebruikersinterface.
- *Inputs.exe*
  Deze executable zorgt voor de verwerking van de inputs.
- *Outputs.exe*
  Deze executable zorgt voor de verwerking van de outputs.
- *SiemuflexDLL.dll*
  Deze DLL bevat functies om de inputs in het shared memory (Siemuflex_Input) te schrijven en wordt aangeroepen door Flexsim. Het shared memory (Siemuflex_Input) kan gelezen worden door Inputs.exe.

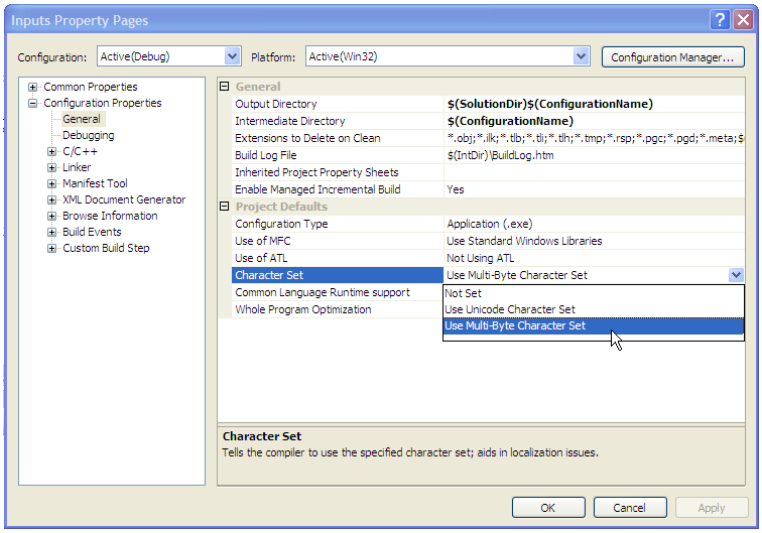
De broncode van de bestanden staan in de map “Siemuflex” onderverdeeld in mappen waarvan de naam gelijk is aan de executable/DLL. Er is in de map “Siemuflex” een solution bestand genaamd “Siemuflex.sln”. Een solution is in Visual Studio een structuur voor Visual Studio projecten. In één solution kunnen meerdere Visual Studio projecten gemaakt worden.



Figuur 37: Solution "Siemuflex"

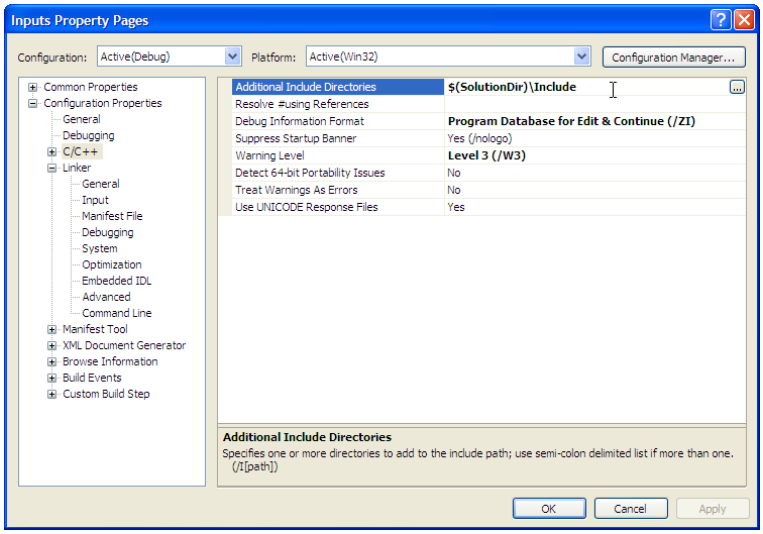
In de solution “Siemuflex” staat alles juist geconfigureerd. Mocht er onverhoopt iets misgaan waardoor dit verloren gaat kan er een nieuwe solution aangemaakt worden. Er moeten dan enkele parameters geconfigureerd worden. De betreffende bestanden staan in de map “Include”. Zie ook de lijst van bestanden in bijlage G.

Voor alle Siemuflex Visual Studio projecten (Outputs, Inputs, SiemuflexDLL en Start :
De gebruikte character set moet op Multi-Byte staan. Ga met de muis op een Visual Studio project staan en klik met de rechtermuisknop. Klik op **Properties**. Ga naar **Configuration Properties → General** en zet de Character Set op **Use Multi-Byte Character Set**.



Figuur 38: Instellen “Character Set”

Er moet een map geïnclude worden waarin diverse header-bestanden staan. Deze map is: “Siemuflex\Include”. Ga om deze map te includen met de muis op een Visual Studio project staan en klik met de rechtermuisknop. Klik op **Properties**. Ga naar **C/C++ →General**. Vul bij **Additional Include Directories** het volgende in: **$(SolutionDir)\Include.**

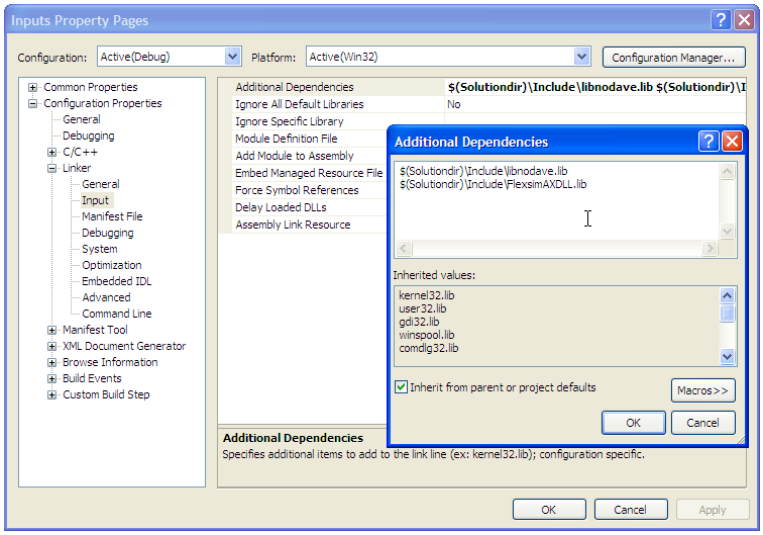


Figuur 39: Instellen include map

Voor Siemuflex Visual Studio projecten Inputs en Outputs:

Er moeten twee library (.LIB) bestanden toegevoegd worden aan het project. Ga met de muis op een Visual Studio project staan en klik met de rechtermuisknop. Klik op **Properties**. Ga naar **Linker → Input**. Vul bij **Additional Depedencies** het volgende in:

- **$(Solutiondir)\Include\libnodave.lib**
- **$(Solutiondir)\Include\FlexsimAXDLL.lib**



Figuur 40: Toevoegen library bestanden

Voor Siemuflex Visual Studio project Outputs:

Bij het Output programma moeten nog drie additionele library bestanden toegevoegd worden. Dit kan op gelijke wijze als bij de vorige stap. De bestanden zijn:

- **rpcrt4.lib**
- **comsuppw.lib**
- **comsupp.lib**

Deze bestanden zijn niet meegeleverd met Siemuflex omdat deze al bij Microsoft Visual Studio en Microsoft Software Development Kit meegeleverd worden.

7. Conclusies en aanbevelingen

7.1. Conclusies

De eerste conclusie die getrokken kan worden is dat de software lastig is voor studenten. Ondanks dat de afstudeerders onderwezen zijn in de programmeertaal C++, was de kennis aan de start van deze afstudeeropdracht beperkt tot basiskennis. Naast de moeilijkheden met de programmeertaal kwamen er ook een aantal specifieke programmeertechnieken aan bod, zoals COM, waar de afstudeerders nooit eerder in aanraking mee waren geweest.

Ten tweede was de software ook lastig voor de medewerkers van Van Riet, dit omdat deze medewerkers normaliter niet in C++ programmeren en weinig ervaring hebben met de specifieke programmeertechnieken. Ondanks dat de medewerkers van Van Riet IA zeer toegankelijk en bereid zijn om te helpen moesten veel zaken zelf uitgezocht worden.

De derde conclusie die getrokken kan worden is dat Flexsim niet gemaakt is om op deze wijze gebruikt te worden. Vanuit Talumis – distributeur van Flexsim in Nederland – was de wil om support te geven ook duidelijk aanwezig. Helaas is over het van buitenaf besturen van Flexsim nog weinig bekend, ook bij de werknemers van Talumis.

De vierde en belangrijkste conclusie is dat alle Must haves uit het pakket van eisen zijn behaald! Ook de Could have is behaald, Siemuflex is immers geschikt voor de S7-300 en S7-400 serie PLC's. Ondanks de moeilijkheidsgraad en omstandigheden is er toch een werkend eindproduct afgeleverd

7.2. Aanbevelingen

Tijdens het doorlopen van de afstudeeropdracht zijn er een aantal nieuwe ontwikkelingen en inzichten gekomen. Uit die ontwikkelingen en inzichten zijn een aantal aanbevelingen geformuleerd.

- Op dit moment worden de types van de outputs handmatig in een tekstbestand ingevoerd. Er kan worden gekeken naar de mogelijkheid om dit automatisch te doen. Één manier waarop dit gedaan zou kunnen worden is om de gegevens uit een Excel lijst te halen. Van Riet maakt bij het ontwerpen van een systeem een Excel lijst met de symboliek van de PLC in- en outputs. Uit de symboliek kunnen grotendeels de types afgeleid worden. Een motor is bijvoorbeeld altijd genummerd met een M (bv. 200M0). En een lamp altijd met een H. Daar is rekening mee gehouden, zodat de types in Siemuflex zoveel mogelijk overeenkomen met de symboliek van Van Riet.
- In de huidige opzet worden door de executable Start.exe twee andere executables – Input.exe en Output.exe – gestart. Er kan worden gekeken naar een andere opzet. Een optie die aanbevolen wordt is een multithreaded executable. Meerdere threads kunnen binnen één executable uitgevoerd worden. Zo kunnen de inputverwerking en outputverwerking nog steeds onafhankelijk van elkaar draaien, maar wordt alles binnen één executable gehouden. De voordelen hiervan zijn:
    - Verbinding maken met de PLC hoeft maar één keer te gebeuren.
    - Er is maar een executable dus, indien goed geschreven, stabieler.
    - Gemakkelijk uitwisselen van gegevens tussen de threads.

- Op dit moment is de Siemuflex library binnen Flexsim redelijk beperkt. Deze kan uitgebreid worden met meer standaard onderdelen uit een Van Riet transportsysteem. Op het moment dat er een complete library is, wordt het zeer eenvoudig om een volledig transportsysteem na te bouwen.
- Door tijdgebrek is er geen tijd overgebleven voor de deelopdracht, het onderzoeken van de koppeling tussen P'X5 en Flexsim. Uit toekomstig oogpunt kan het verstandig zijn om hier alsnog naar te kijken.

8. Literatuurlijst

8.1. Gedrukte media

Titel	Auteur	Jaar	Druk	ISBN
Projectmanagement	Roel Grit	2005	4de	90 – 01 – 34703 – 7
Aan de slag met C++	Gertjan Laan	2007	1, oplage 4	90 – 395 – 1689 – 8
Mastering Delphi 5	Marco Cantù	1999	1ste	0 – 7821 – 2565 – 4
Essenstial COM	Don Box	1998	2de	0 – 201 – 63446 – 5

Tabel 7: Gedrukte media

8.2. Online media

8.2.1. Online documenten

Titel	Auteur	Download locatie
Programming with STEP 7	Siemens	http://support.automation.siemens.com/WW/view/en/18652056
Working with STEP 7	Siemens	http://support.automation.siemens.com/WW/view/en/18652511
Flexsim brochure		http://www.flexsim.com/products/flexsim/FlexsimBrochure.pdf

Tabel 8: Online documenten

8.2.2. Internetpagina's

Titel	URL
Siemens AG - Wikipedia	http://nl.wikipedia.org/wiki/Siemens_AG
LIBNODAVE, a free communication library for Simatic S7 PLCs	http://libnodave.sourceforge.net/
OLE for Process Control - Wikipedia	http://nl.wikipedia.org/wiki/OLE_for_Process_Control
Component Object Model - Wikipedia	http://en.wikipedia.org/wiki/Component_Object_Model
Visual Basic - Wikipedia	http://nl.wikipedia.org/wiki/Visual_Basic
Compound document - wikipedia	http://en.wikipedia.org/wiki/Compound_document
Component Object Model (COM) - MSDN	http://msdn.microsoft.com/en-us/library/ms680573%28v=VS.85%29.aspx
Fred's COM (Component Object Model) Tutorials - JRS	http://www.jose.it-berater.org/smfforum/index.php?board=362.0
Connecting to COM object with C++ - C++ Forums	http://www.cplusplus.com/forum/windows/19738/
Flexsim forum - Flexim.com	http://www.flexsim.com/community/forum/
Interprocess Communications (Windows)	http://msdn.microsoft.com/en-us/library/aa365574%28v=VS.85%29.aspx
String to Char Array Conversion	http://www.programmingforums.org/thread16393.html
Get string length	http://www.cplusplus.com/reference/clibrary/cstring/strlen/
How to configure paging files	http://support.microsoft.com/kb/314482
CreateFileMapping Function	http://msdn.microsoft.com/en-us/library/aa366537%28VS.85%29.aspx
*.exe file in a c-program	http://www.daniweb.com/forums/thread10686.html#
pointer for shared memory	http://www.boost.org/doc/libs/1_37_0/doc/html/interprocess/quick_guide.html#interprocess.quick_guide.qg_offset_ptr

Tabel 9: Internetpagina's

9. Begrippenlijst

9.1. Definities

Term	Verklaring
Array	Lijst van elementen.
Compound Document	Vrij vertaald: Samengesteld document. Een voorbeeld hiervan is een spreadsheet ingevoegd in een tekstdocument.
DLL	Bibliotheek met functies, die door meerdere applicaties gebruikt kunnen worden.
Emulatie	Omzetten van data zodat een andere toepassing ermee om kan gaan.
Executable	Uitvoerbaar programma.
Flexscript	Een programmeerscript voor Flexsim. Dit script is afgeleid van de basis programmeertaal C++.
HW Config	Configuratieprogramma van Siemens voor hardware instellingen PLC.
In-house	In het thuispand.
Interface	Middel waarmee twee systemen met elkaar kunnen communiceren.
Libnodave	Software bibliotheek met functies voor data uitwisseling met Siemens S7 PLC's.
Model	Flexsim simulatiemodel.
NetPro	Configuratieprogramma van Siemens voor netwerk instellingen PLC.
Nodefunctie	Een functie binnen Flexsim geschreven in Flexscript; kan worden aangeroepen via COM.
On-site	Op de plek waar het systeem geplaatst wordt.
Orthographic View	Een grafische weergaven binnen Flexsim om een model te bouwen.
P'X5	Softwarepakket voor technical sales en projectplanning.
Pointer	Verwijzing naar een adres in het geheugen.
Profibus	Industriële datacommunicatiestandaard (veldbus) voor informatie uitwisseling.
Project	Project van Van Riet waarbij een transportsysteem ontworpen, gefabriceerd, geplaatst en in bedrijf gesteld wordt.
Rack	Grondplaat van een PLC waarin de onderdelen modulair opgebouwd kunnen worden.
Runtime	De tijd tijdens het uitvoeren van een programma.
Runtime error	Fout bij de uitvoering van een programma.
SIMATIC	Automatiseringsysteem van Siemens.
Sink	Tegenstelling van een source; Een afvoerplek voor flowitems binnen flexsim.
Source	Een bron; verzorgt flowitems (zoals pakketjes) binnen flexsim.
Transport-systeem	Een transportsysteem ontwikkeld, gemaakt, geplaatst en getest door Van Riet.
Trigger	Een stuk code dat ervoor zorgt dat bij een bepaalde gebeurtenis een actie wordt ondernomen.
TTI-project	Project van Van Riet bij een bandendistributeur waar autobanden getransporteerd worden.
Visual Basic	Programmeeromgeving en programmeertaal, uitgebracht door Microsoft. Het doel van Visual Basic is de ondersteuning van het bouwen van grafische applicaties op een visuele manier, dat wil zeggen, zo veel mogelijk via directe grafische manipulatie van elementen in plaats van het expliciet invoeren van programmacode.

Tabel 10: Definities

9.2. Afkortingen en acroniemen

Afkorting / acroniem	Verklaring
CLSID	Class Identification
COM	Component Object Model
CP	Communications Processor
CPU	Central Processing Unit
DCOM	Distributed Component Object Model
DDE	Dynamic Data Exchange
FBD	Function Block Diagram
HMI	Human Machine Interface
I/O	Input/Output
IEC	International Electrotechnical Commission
IID	Interface Identification
IP	Internet Protocol
LAD	Ladderdiagram
MPI	Multi Point Interface
OLE	Object Linking and Embedding
OPC	OLE voor Process Control
OS	Operating System
PCMCIA	Personal Computer Memory Card International Association
PI	Process Image
PII	Process Image input
PIQ	Process Image output
PLC	Programmable Logic Controller
RAM	Random Access Memory
SF	Siemuflex
STL	Statement List
TCP	Transmission Control Protocol
TTI	Tyre Trading International

Tabel 11: Afkortingen en acroniemen

10. Persoonlijke evaluatie

10.1. Persoonlijke evaluatie Cedric Molthoff

De afstudeerperiode heb ik met plezier doorlopen. In het begin waren er twijfels bij mij over de opdracht maar na de start kon ik mij helemaal vinden in deze opdracht. Het feit dat we iets moesten creëren waar wij en het bedrijf weinig tot geen ervaring mee hadden, maakte deze opdracht zeer uitdagend.

Ondanks dat we vrijwel alle tijd op kantoor door brachten ervaar ik deze opdracht toch zeer praktisch, dit is in tegenstelling tot een vorige stage waar de opdracht puur theoretisch was. Het afstuderen sprak mij hierdoor meer aan dan de stage. Ook het feit dat we Bart en ik samen deze opdracht aannamen, uitvoerden en voltooiden heb ik ervaren als een groot plus punt. De samenwerking met Bart verliep goed en we zitten aardig op een lijn qua denkwijze. Deze samenwerking zorgde voor een goede stimulatie doordat we bij tegenslagen of een “bad day” elkaar goed konden motiveren.

Tijdens het afstuderen liep ik uiteraard tegen een paar problemen aan. Het grootste probleem dat ik tegen ben gekomen was het feit dat het aan de praat krijgen van het COM-object niet wou lukken. Na vele pogingen en tijd is het uiteindelijk toch zelf gelukt. Hieruit heb ik geleerd dat het (sneller) inschakelen van een derde persoon voor het oplossen van het probleem een verstandigere keuze was geweest dan te blijven hangen, zowel voor de tijdsdruk als voor mijn eigen stemming en motivatie.

Het projectgericht werken ging goed. Door een goed plan van aanpak en de basis stappen (initiatief-, definitie-, ontwerp-, voorbereidings-, realisatiefase) te blijven volgen verliep de planning op het COM-object na perfect. Ondanks onvoorziene veranderingen (zoals de omschakeling van OPC naar Libnodave) bleven we goed op koers en kleine achterstanden werden snel weer ingehaald. Ook met het schrijven van de scriptie waren we redelijk consequent waardoor de deadline van inleveren gemakkelijk gehaald is. Dit is ten opzichte tot de voorgaande projecten een goede verandering en een plezierige ervaring.

Veel van de vakken die ik aan de Hogeschool Utrecht heb gevolgd kwamen terug in deze opdracht. Het betreft o.a. de vakken C++, Datacom, Besturingstechniek, Digitale techniek, PLC/SCADA, Modelleringtechnieken en alle projecten. Wat tegen viel bij de uitvoering van de opdracht was het feit dat de kennis over C++ ondanks dat ik hierin veel ben onderwezen ver weggezakt was. Maar toen dit weer was opgehaald is dit geen probleem meer geweest.

Zowel op gebied van projectmatig werken & planning, programmeren, de durf om te vragen, samenwerken en communicatie heb ik veel geleerd en kan ik deze afstudeerperiode kan ik met een goed en tevreden gevoel afsluiten!

Cedric Molthoff,
mei 2010

10.2. Persoonlijke evaluatie Bart Holtermans

De afstudeerperiode heb ik als zeer prettig ervaren. Ik vond het een leuke opdracht, die veel van mijn technische interesses aanspreken. Ook vond ik de opdracht erg uitdagend, omdat we iets nieuws moesten gaan verzinnen.

Mijn voorgaande stages waren onderdeel van mijn MBO opleiding. Daar heb ik vooral de realisatiefase uitgevoerd, bijvoorbeeld assembleren van producten. Nu heb ik alle fases, van de initiatieffase tot en met de realisatiefase, uitgevoerd. Dit vond ik erg prettig. Vooral het ontwerpgedeelte spreekt mij erg aan.

Ook kon ik in de afstudeeropdracht de kennis, die ik heb opgebouwd met het uitvoeren van schoolprojecten, in de praktijk toepassen. Hier heb ik veel van geleerd, en ook inzicht gekregen wat ik al beheers en waar verbeterpunten zitten.

De opdracht bestond grotendeels uit programmeren in C++. Hier ben ik in onderwezen maar de kennis die ik daarmee heb vergaard is vooral basiskennis. Ook is dit al meer dan een jaar geleden en was de kennis weggezakt. Op het vlak van programmeren heb ik dan ook veel geleerd. Dit kan erg nuttig zijn in de toekomst omdat C++ veel gebruikt wordt, en andere veelgebruikte objectgeoriënteerde programmeertalen grote samenhang vertonen met C++.

Op het gebied van plannen ben ik ook vooruitgegaan. In het begin van de opdracht schatte ik vaak de tijdsduur die benodigd was om iets uit te voeren te positief in, waardoor de tijd soms begon te dringen. Dit ging op het einde van de opdracht stukken beter.

Wat ik als positief ervaar is dat ik in deze opdracht veel theoretische kennis van vakken uit het curriculum van Industriële Automatisering in de praktijk heb kunnen toepassen. Bijvoorbeeld onder andere besturingstechniek, datacom, en digitale techniek.

Ook de samenwerking met Cedric verliep goed. Door de samenwerking werd ik gedwongen goed te communiceren. Dit is iets waar ik in het verleden te weinig aan deed, maar nu beter het nut van in zie en in gegroeid ben.

Aan het eind van de afstudeeropdracht kan ik met een goed gevoel deze periode afsluiten. De opdracht is gehaald, de interface werkt, en er ligt een mooie scriptie.

Bart Holtermans,
mei 2010

Bijlagen

BIJLAGE A: Plan van Aanpak

Lijst van Bijlage A – figuren

Lijst van Bijlage A – tabellen

Plan van Aanpak

Siemuflex

Flexibel simuleren/emuleren…

Afstudeerder 1:	**Afstudeerder 2:**	**Opdrachtgever:**	**Datum:**
Bart Holtermans	Cedric Molthoff	Van Riet	25 februari 2010
1520310	1507692	Industrial Automation	

Inhoudsopgave

1. Achtergronden

1.1. Van Riet

De naam Van Riet is al meer dan 60 jaar een goede bekende in de wereldmarkt voor Material Handling Systemen. De systemen zijn geïnstalleerd van Mexico tot Shanghai, van Praag tot Istanbul en van Los Angeles tot Amsterdam.

Het merendeel van de mechanische systeemelementen ontwikkelt Van Riet in eigen huis, ondersteund door haar eigen R&D afdeling.

Met de juiste systeembesturing wordt het systeem afgemaakt. Van Riet ontwikkelt haar software in eigen huis, zowel op PC- als op PLC-niveau.

Van Riet biedt met dit pakket totale systeemoplossingen, waarbij een Turn-Key systeem wordt opgeleverd. Van Riet denkt actief mee, vanaf de eerste oriëntatie tot en met de implementatie. Om de systeemwerking voor jaren te garanderen, biedt Van Riet een uitgebreid scala aan Service Packs aan.

Van Riet is gespecialiseerd in het ontwerp, de realisatie en de installatie van Pallet-, Order Pick-, Sorteer-, Robot- en Transportsystemen. In eerste instantie werden alleen kolen en mest getransporteerd. Inmiddels heeft Van Riet een grote variëteit aan producten getransporteerd:

Aardappelen	Ansichtkaarten	Autobanden	Automotoren
Auto onderdelen	Baby artikelen	Bakken	Bio Stabiel
Blikjes bier	Blikken	Bloemen	Blokken zout
Boeken	Boutjes	Brillen glazen	Camera's
Cargo	CD-tjes	Chocolade	Consumenten elektronica
Cosmetica	Deodorant	Doe het zelf artikelen	DVD-spelers & DVD-tjes
Emballage kratten	Espresso apparaten	Euro's	Farmaceutische artikelen
Fietsbanden	Flyers	Fornuizen	Fototoestellen
Fruit	Gereedschap	Geld	GSM toestellen
Golfsticks	Grafstenen	Groente	Haarverzorgings-producten
Huidverzorgings-producten	Incontinentie-materiaal	Isolatiemateriaal	Kantoor artikelen
Kazen	Keukenmachines	Keukenmeubels	Kleding
Kleurenprinters	Koelkasten	Koffers	Koffie
Kranten	Kratten	Kunststof kratten	Kweekplantjes
Lampen	Lingerie	Losse lakens	Luiers
Magazines	Magnetrons	Medicijnen	Meubels
Mie	Mobieltjes	Moertjes	Montage materiaal

Olievaten	Overalls	Pallets	Pannen
Papier	Papier en karton afval	Pc's	Plantjes
Pons afval	(Post-) pakketjes	Reserve-onderdelen	Rolcontainers
Schoenen	Schoenendozen	Schoolartikelen	Schoonmaak-producten
Schroefjes	Shampoo	Sigaren	Sigaretten
Snijbloemen	Sokken	Spijkerbroeken	Spijkers
Spuitmonden	Stoelen	Tandpasta	Tapijtrollen
Tapijttegel	T-shirts	Theekratten	Theezakjes
Tijdschriften	Trays	TV's	Vaatwassers
Vaten	Vaten vruchtensap	Velgen	Verf
Verfbenodigdheden	Verlichting	Verpakt vlees	Verwarmings-radiateuren
Videobanden	Videorecorders	Vissers-benodigdheden	Vrachtwagen cabines
Wasgoed (vuil en schoon)	Waskoeken	Wasmachines	Wijn
Zakjes bloed	Zakken cement	Zakken veevoer	Zeepproducten
Zonnebrandcrème	Zonnehemels	Zuivel	

Bijlage A - Figuur 1: Markten van Van Riet

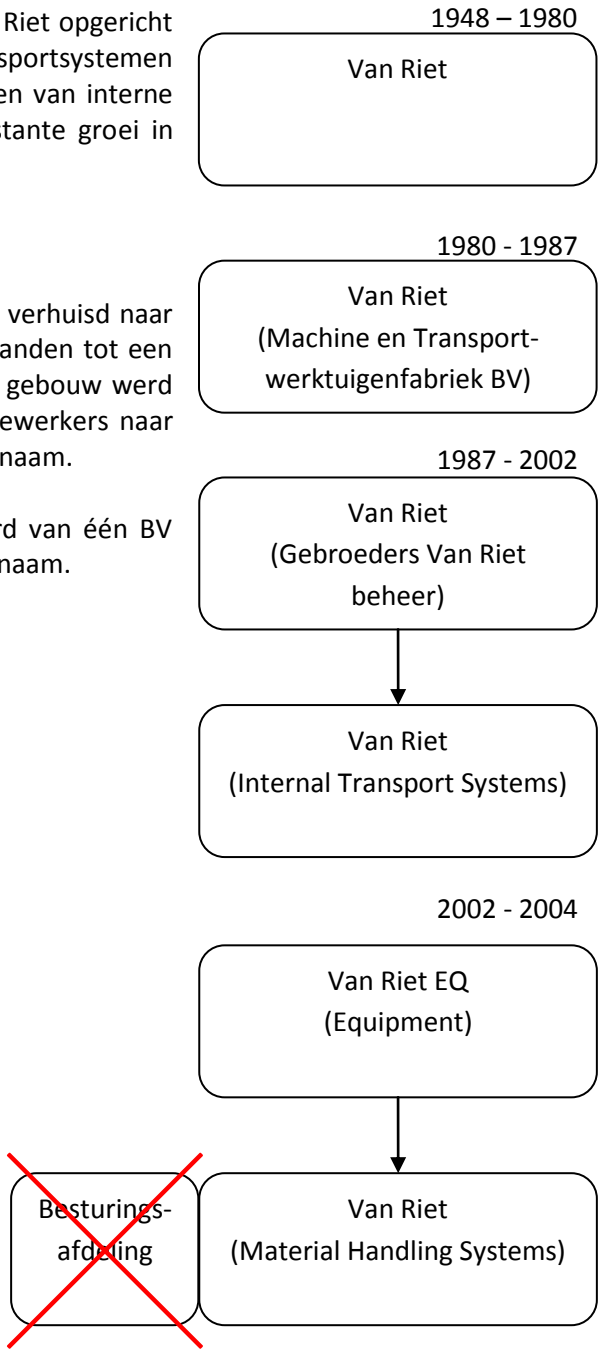
1.2. Historie

Het familiebedrijf Van Riet werd door de heer P. van Riet opgericht in 1948 in Utrecht. Via kolentransporteurs, mesttransportsystemen en bouwliften viel de keuze uiteindelijk op het maken van interne transportsystemen, met name omdat hier een constante groei in zat.

Het pand in Utrecht werd al snel te klein en er werd verhuisd naar Bilthoven. Hier werden in de loop van 25 jaar drie panden tot een bedrijfsgebouw van formaat samengevoegd. Ook dit gebouw werd te klein en in 1980 verhuisde Van Riet met 45 medewerkers naar Nieuwegein. Van Riet ging verder onder een 'nieuwe' naam.

In 1987 werd de structuur van het bedrijf veranderd van één BV naar twee BV's. Ook kreeg Van Riet weer een nieuwe naam.

In ± 2002 waren de financiële middelen helaas niet meer toereikend om in zijn geheel door te gaan. Van Riet heeft een doorstart gemaakt onder de naam Material Handling Systems (MHS). Van Riet is doorgegaan met het bouwen van 'prefab' transportsystemen maar zonder besturingsafdeling.

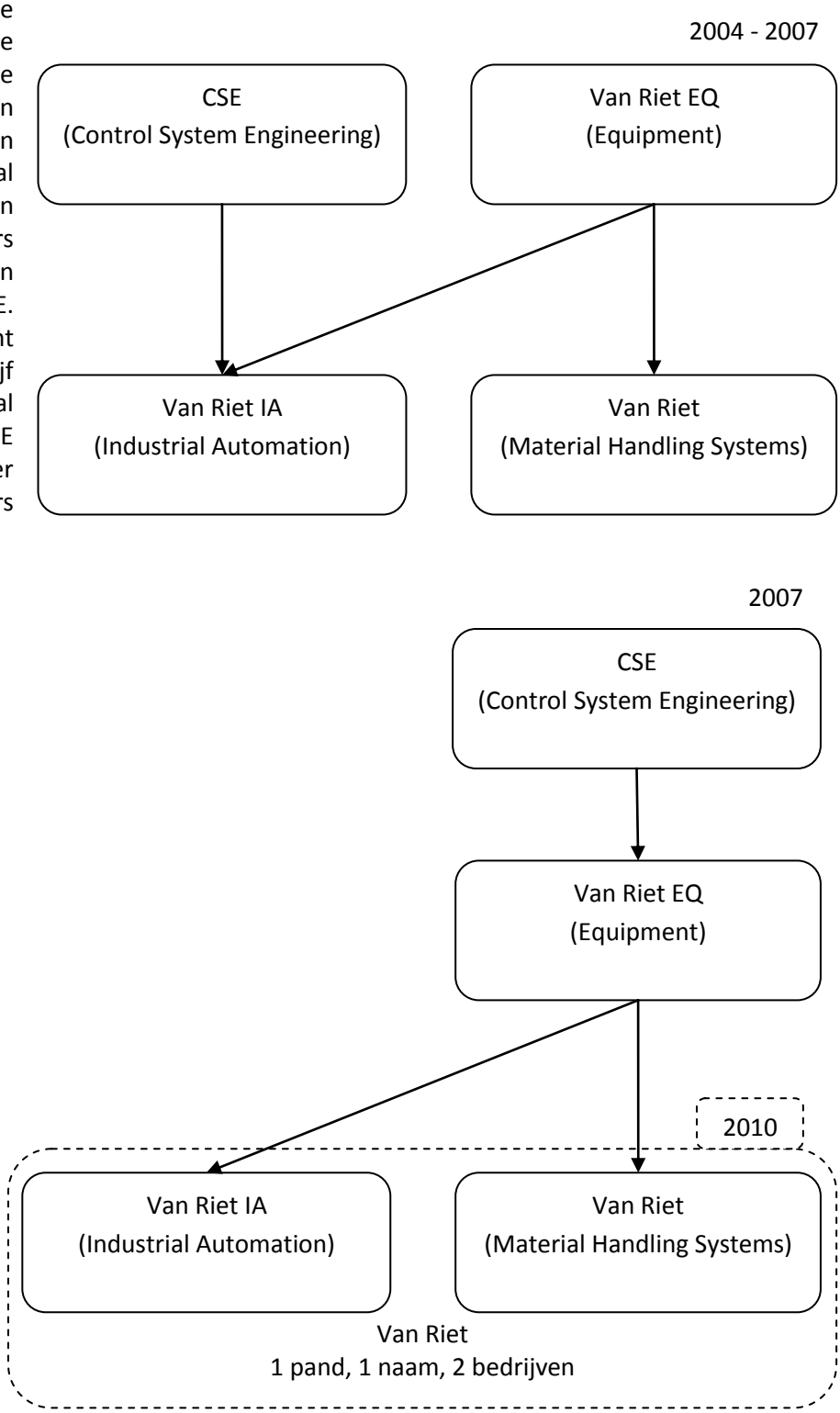


Bijlage A - Figuur 2: Van Riet door de jaren heen

De kennis en expertise die aanwezig was/is bij de mensen die voorheen op de besturingsafdeling van Van Riet werkten was toch van belang voor Van Riet Material Handling Systems (MHS). Een deel van deze werknemers hadden hun loopbaan voortgezet bij het bedrijf CSE. CSE en Van Riet Equipment (EQ) hebben toen het bedrijf Van Riet Industrial Automation (IA) opgezet. CSE en Van Riet (EQ) waren hier dan ook de aandeelhouders van.

In 2007 zijn de verhoudingen veranderd. Van Riet EQ werdt volledig aandeelhouder van Van Riet IA. CSE heeft in ruil voor zijn aandelen in Van Riet IA aandelen in Van Riet EQ gekregen.

Vanaf medio maart 2010 zullen Van Riet MHS en Van Riet IA als één bedrijf vanuit één pand naar de buitenwereld treden. Alleen op papier achter de schermen zullen de twee bedrijven blijven bestaan.



Bijlage A - Figuur 3: Van Riet door de jaren heen (vervolg)

1.3. Van Riet Industrial Automation

Van Riet Industrial Automation is gespecialiseerd in het ontwerpen, implementeren en onderhouden van hard- en software voor geautomatiseerde interne transportsystemen van stukgoederen. Van Riet Industrial Automation verricht besturingstechnische activiteiten ten behoeve van projecten van Van Riet Material Handling Systems.

1.4. Software engineering

Binnen Van Riet Industrial Automation bevinden zich twee afdelingen, software engineering en hardware engineering. Het project Siemuflex wordt uitgevoerd binnen de afdeling software engineering.

1.5. Siemuflex

Het project Siemuflex houdt in: met behulp van het simulatie pakket FlexSim wordt er vooraf getest of, en hoe de beoogde capaciteit van interne transportsystemen behaald kan worden. Door deze simulatie te koppelen aan de ontwikkelde PLC-software ontstaat er een emulatie waarmee vooraf de ontwikkelde PLC algoritmes getest kunnen worden.
Het project Siemuflex is in het leven geroepen door de heer Bosma. De heer Bosma is Managing Director bij Van Riet Industrial Automation. Vanaf nu zullen in dit verslag de bedrijven Van Riet Material Handling Systems en Van Riet Industrial Automation als één bedrijf gezien worden onder de naam Van Riet.

Het project werd door de heer Bosma aan ons (Bart Holtermans en Cedric Molthoff) aangeboden. Wij hebben dit project aangenomen. Dit doen wij in het kader van ons afstudeertraject. In samenwerking met de heer Bosma zullen de activiteiten, grenzen en dergelijke zaken bepaald worden.

Verder in dit plan van aanpak (PVA) zullen deze bepalingen vastgelegd worden.

2. De projectopdracht

2.1. Inleiding

Het project Siemuflex komt voort uit de behoefte om de volledige doorlooptijd van een project te verkorten. Een van de zaken die veel tijd in beslag neemt is de inbedrijfstelling. In het kader van het algehele doel wil Van Riet dus de inbedrijfstellingsduur on-site verkorten. Het project Siemuflex zal zich vooral hierop toespitsen.

Van Riet engineert software- en hardwarematig interne transportsystemen. Elk intern transportsysteem wordt klantspecifiek ontwikkeld en is hierdoor uniek. Alle bijbehorende PLC-software is daarom ook klantspecifiek en uniek. Standaardisatie is dus beperkt mogelijk. De PLC-software wordt op dit moment on-site getest tijdens de inbedrijfstelling.

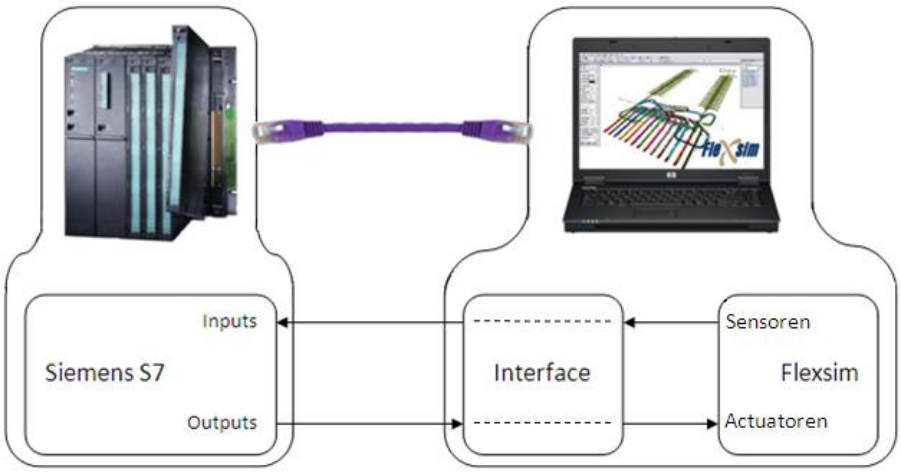
Van Riet bedient de internationale markt, waardoor de site zich vaak op grote afstand bevindt. Het bereid vinden van personeel die langere tijd op grotere afstand wil werken is lastig. Dit zet namelijk druk op hun sociale leven. Tevens is het een prijzige aangelegenheid om personeel naar het buitenland te sturen in verband met uurtoeslagen, tickets en andere zaken.
Het is dus zowel sociaal als financieel van belang dat de inbedrijfstellingsduur zo kort mogelijk wordt gehouden!

2.2. Interface A: Siemens ↔ Flexsim

Het doel is om een interface te ontwikkelen waarmee de PLC-software op kantoor getest kan worden. Deze testen zullen worden uitgevoerd op simulaties die gemaakt zijn in het 3D simulatiepakket Flexsim. In Flexsim wordt het echte systeem nagebouwd. Doordat de PLC aan de PC gekoppeld wordt, en via de te creëren interface kan communiceren met het simulatiepakket, ontstaat er een emulatie.

Door middel van deze emulatie kan het personeel van Van Riet in grote lijnen de PLC-software op kantoor testen. Hierdoor bevind men zich kortere tijd on-site, waardoor het sociale leven minder onder druk komt te staan en de kosten zullen dalen.

Naast het linken van inputs (voornamelijk sensoren) en outputs (voornamelijk motoraansturingen) moeten de reserveertabellen voor 'track and trace' zichtbaar worden om meer inzicht in de reserveringstechniek te krijgen. Uiteindelijk dient de emulatie gebruiksvriendelijk te zijn voor de PLC programmeurs. De basis is hieronder met een afbeelding en schematisch weergegeven.



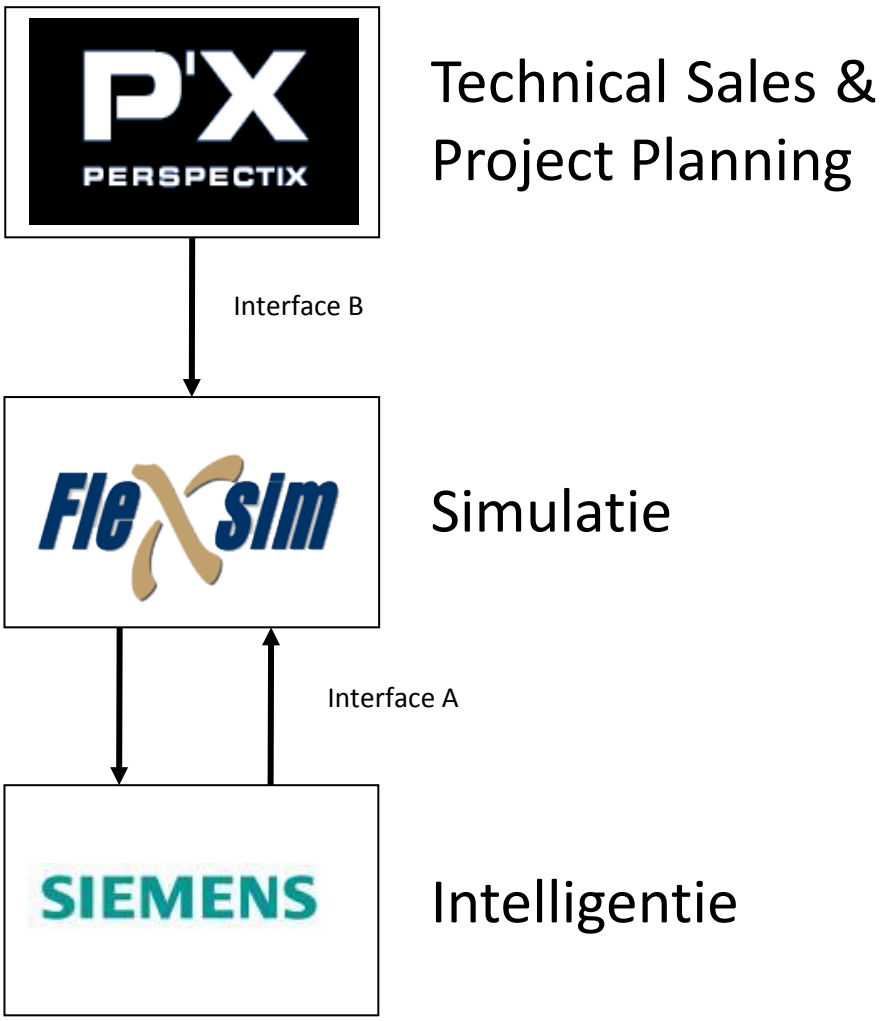
Bijlage A - Figuur 4: Voorstelling opdracht

2.3. Interface B: P'X5 → Flexsim

Met P'X5 kan snel en eenvoudig een basisschets worden gemaakt van een intern transportsysteem. Doordat P'X5 een standaard bibliotheek bevat kan gemakkelijk een prijs berekend worden. In deze bibliotheek staan standaard componenten waarvan alle technische gegevens bekend zijn. Naast de bovengenoemde interface zal dan ook gekeken worden naar verdere automatisering in het hogere proces. De bedoeling van deze interface is om uiteindelijk in P'X5 eenmalig de gegevens van de componenten in te voeren, waarna Flexsim de juiste gegevens overneemt en hieruit een model creëert. Tijdens deze afstudeerstage zal waar mogelijk gekeken worden naar wat Flexsim voor gegevens nodig heeft en in welk formaat.

2.4. Schematisch overzicht interfaces

Een schematisch overzicht van beide interfaces:



Bijlage A - Figuur 5: Hoofddoel van Van Riet

3. Projectactiviteiten

Initiatieffase

- Informatie inwinnen over de visie van de opdrachtgever.
- Inlezen in het onderwerp.
- Maken van de planning.
- Schrijven van het Plan van Aanpak.

Definitiefase

- Onderzoek doen naar de werking Siemens communicatieprotocol.
- Onderzoek doen naar de werking van Flexsim.
- Onderzoek doen naar de mogelijkheden van communicatie tussen verschillende protocollen.

Ontwerpfase

- Onderzoek doen naar het eenvoudig uitlezen in/outputs.
- Onderzoek doen naar het verwerken van deze gegevens.
- Onderzoek doen naar 'real-time' communicatie.
- Deze basis gegevens verwerken in prototype software.

Voorbereidingsfase

- Onderzoek naar de mogelijkheid naar het creëren van een overall interface.

Realisatiefase

- Het daadwerkelijk creëren van een complete interface.
- Onderzoek interface P'X5 → Flexsim.
- Het schrijven van een handleiding voor de eindgebruiker.
- Het schrijven van de scriptie.
- De interface testen.

4. Projectgrenzen en randvoorwaarden

4.1. Randvoorwaarden

Om het afstuderen goed te laten verlopen zijn er een aantal voorwaarden gesteld die tijdens het afstuderen van kracht zijn.

Het gaat hier om de volgende randvoorwaarden:

- De afstudeerders zorgen er voor dat de docentbegeleider op de hoogte wordt gehouden van de voortgang van het afstudeerproces.
- De bedrijfsbegeleider dient te zorgen voor een goede inhoudelijke begeleiding van de afstudeeropdracht.
- De afstudeerders moeten beschikken over de benodigde middelen zoals een laptop, werkplek etc.
- De afstudeerders zullen zelf actie ondernemen om informatie te verkrijgen. Indien nodig zal het bedrijf hierin sturen en/of helpen.
- Eventuele kosten, die gemaakt zijn door de afstudeerders voor de opdracht worden na goedkeuring door het bedrijf vergoed.
- Trainingsmateriaal moet aanwezig zijn voor de programma's waar de afstudeerders mee aan de slag moeten.

De einddata die overeengekomen zijn door de Hogeschool Utrecht en waar de afstudeerders zich aan moeten houden zijn hieronder weergegeven:

Inleveren eerste versie PVA	09-02-2010
Inleveren definitieve PVA	26-02-2010
Inleveren eerste concept scriptie	21-04-2010
Inleveren tweede concept scriptie	15-05-2010
Inleveren definitieve versie scriptie	25-05-2010
Verdediging scriptie	Week 24, 14-06 – 18-06, 2010

4.2. Grenzen

Omdat dit project al snel heel groot kan worden, zullen de volgende grenzen gesteld worden:

- Het project zal alleen de koppeling tussen Siemens PLC en Flexsim bevatten, en geen koppelingen naar andere lagen.

4.3. Afbakening

- Het zal niet zeker zijn of aan het eind van het project alles automatisch zal werken, hier wordt uiteraard wel naar gestreefd.
- Het project wordt in eerste instantie ontwikkeld voor één type PLC. Waar mogelijk wordt dit generiek gemaakt maar dit is niet de doelstelling.
- De interface zal gecreëerd worden maar het kan zijn dat het programma Flexsim per project bijgesteld dient te worden.

5. De producten

Aan het eind van het afstudeertraject moeten er door de afstudeerders een aantal producten worden opgeleverd. De op te leveren producten kunnen onderscheiden worden in de producten voor de Hogeschool Utrecht en voor het bedrijf. Hieronder staat een schematisch overzicht van de op te leveren producten.

Hogeschool Utrecht	Van Riet Industrial Automation
Plan van Aanpak	Bevindingen van het onderzoek
Scriptie	Interface (Siemens ↔ Flexsim)
Presentatie	Handleiding testopstelling
Beoordelingsformulier	Testopstelling
	Programma van eisen interface (Flexsim → P'X5)

Bijlage A - Tabel 1: Producten

6. Kwaliteit

Er zijn een aantal maatregelen getroffen zodat de afstudeerders een zo goed mogelijk afstudeertraject doorlopen.

Afspraken:
De afstudeerders zijn er verantwoordelijk voor dat zij de afspraken goed plannen en deze op de juiste wijze afhandelen. Ook zorgen de afstudeerders voor een goede oplossing als een van hen onverwacht verhinderd raakt.

Afwezigheid (ziekte):
Elke afwezigheid van een van de afstudeerders moet bekend zijn bij de bedrijfsbegeleider. Wanneer een van de afstudeerders thuis aan de opdracht of scriptie wil werken is dit mogelijk maar dan moet het wel bekend en goedgekeurd zijn bij/door de bedrijfsbegeleider. Als een van de afstudeerders ziek is moet dit gemeld worden aan de bedrijfsbegeleider, indien langer dan een week ook aan de docentbegeleider. Indien het laatste geval van toepassing is moet ook gemeld worden wanneer de afstudeerder weer aanwezig is.

Besprekingen:
Voor de vorderingen is het nuttig om wekelijks een bespreking te hebben met de bedrijfsbegeleider over het project. De afstudeerders zijn verantwoordelijk voor de communicatie met de docentbegeleider, zij dienen deze op de hoogte te houden van de vorderingen dan wel de tegenslagen. Ook is het de bedoeling dat de docentbegeleider minimaal 2 keer langs komt op het bedrijf om een gesprek te hebben met alle deelnemende partijen.

Geleverde producten:
De producten die geleverd worden door de afstudeerders moeten gecontroleerd zijn door hun zelf en het is ook verstandig dat de docentbegeleider en de bedrijfsbegeleider de afstudeerders controleert en, indien nodig, corrigeert.

Voortgang:
De grote lijnen van de opdracht zijn vastgelegd in de planning, deze planning is in hoofdstuk 8 toegevoegd. Het is belangrijk dat de afstudeerders zich zoveel mogelijk aan de planning houden maar het is goed mogelijk dat de planning gewijzigd wordt. De afstudeerders dienen tijdig aan te geven of de planning veranderd wordt en wat voor eventuele gevolgen dat heeft.

7. De projectorganisatie

Hieronder volgt een opsomming van personen die deelnemen aan dit project:

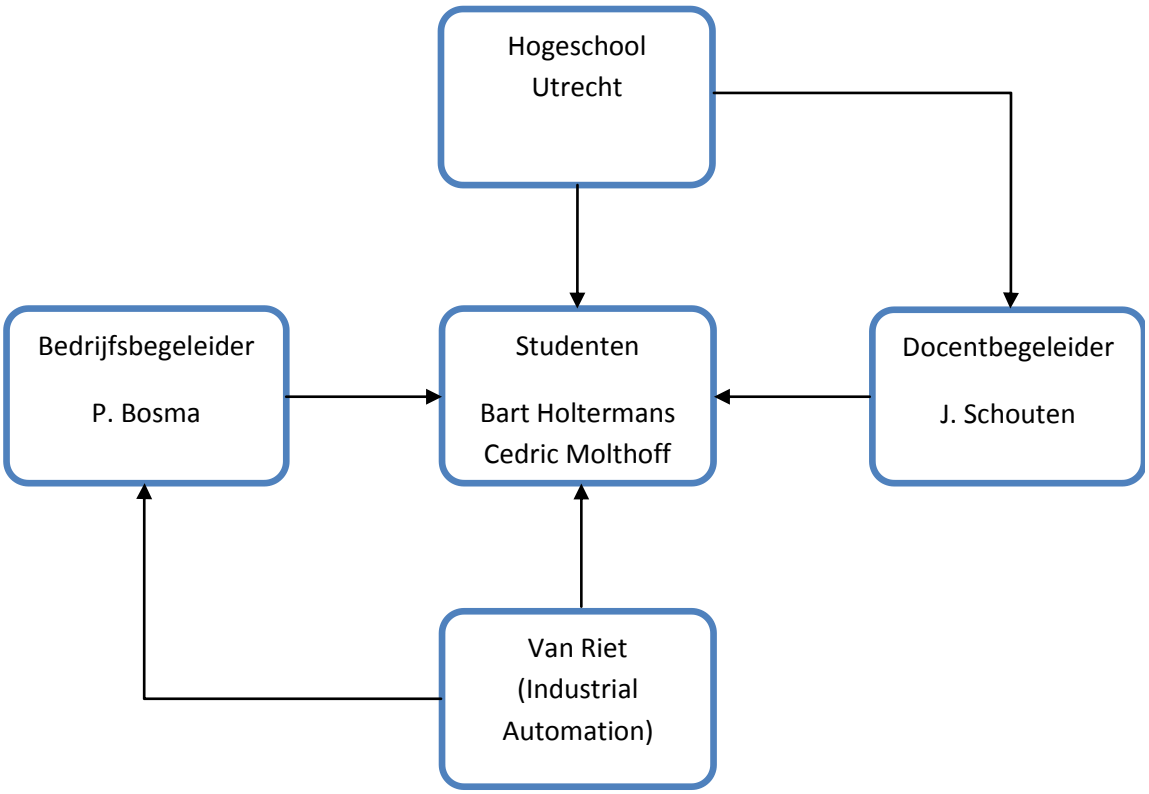
Afstudeerders/studenten:
Cedric Molthoff
cedric.molthoff@student.hu.nl
06 42370318

Bart Holtermans
bart.holtermans@student.hu.nl
06 33051126

Bedrijfsbegeleider:
Piet Bosma
piet.bosma@vria.nl
06 53746343

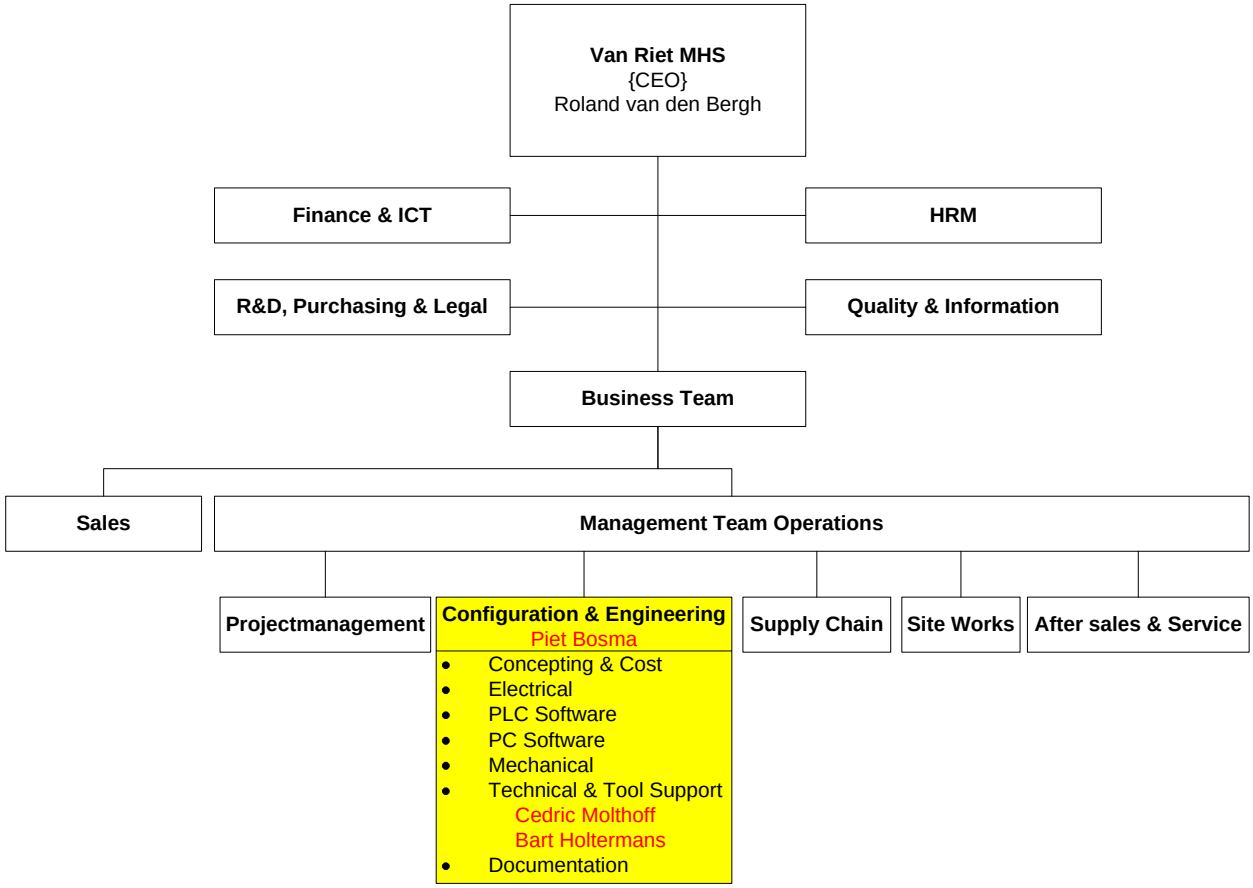
Docentbegeleider:
Jos Schouten
jos.schouten@hu.nl
030 238 8581

De afstudeerders zijn verantwoordelijk voor het project, zij worden hierbij ondersteund door de bedrijfsbegeleider voor het technische gedeelte en de docentbegeleider voor het projectmatige gedeelte.
De Hogeschool biedt informatie en documentatie vooraf aan het afstudeerproces van de afstudeerders. Het bedrijf biedt de specificaties en de richtlijnen voor het project. Zie onderstaande figuur voor een schematisch overzicht.



Bijlage A - Figuur 6: Projectorganisatie

Op de volgende pagina staat een overzicht van de bedrijfsstructuur van Van Riet. In het gearceerde vlak staan de afstudeerders en de bedrijfsbegeleider met rode letters aangegeven.



Bijlage A - Figuur 7: Bedrijfsstructuur Van Riet

8. Planning

Op de volgende vijf pagina's is de planning weergegeven. De planning is onderverdeeld in projectfases.

AFSTUDEER PLANNING (A) Bart Holtermans & Cedric Molthoff (B) Bart Holtermans (C) Cedric Molthoff	Week 5					Week 6					Week 7					Week 8				
	1-feb-2010	2-feb-2010	3-feb-2010	4-feb-2010	5-feb-2010	8-feb-2010	9-feb-2010	10-feb-2010	11-feb-2010	12-feb-2010	15-feb-2010	16-feb-2010	17-feb-2010	18-feb-2010	19-feb-2010	22-feb-2010	23-feb-2010	24-feb-2010	25-feb-2010	26-feb-2010
Activiteiten	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v
Initiatieffase																				
Plan van aanpak maken			A	A	A															
Planning maken				A		A														
Inlezen Profibus, -net					B		B	B	B	B	B	A	A							
Inlezen OPC Server							A	A	A	A	A	A	A							
Inlezen OPC Client							C	C	C	C	C	A	A							
Inlezen Flexsim							C	C	C	C	C	A	A							
Inlezen Siemens Step 7					B		B	B	B	B	B	A	A							
Overleg bedrijfsbegeleider	A	A	A	A	A	A				A	A									
Plan van aanpak inleveren																				A
Definitiefase																				
Mogelijkheden Flexsim bepalen													C	C	A					
Mogelijkheden gegevensoverdracht OPC													C	C	A					
Bezoek DHL															A	A				
Overleg bedrijfsbegeleider															A	A				
Mogelijkheden Siemens software/protocollen													B	B	A					
Ontwerpfase																				
Onderzoek uitlezen in/outputs																	B	B	B	B
Onderzoek verwerking van deze gegevens																	C	C	C	C
Onderzoek ‘real-time’ communicatie																	B	B	B	B
Prototype software																	C	C	C	C
Overleg bedrijfsbegeleider																				A

Bijlage A - Tabel 2: Planning (1/5)

AFSTUDEER PLANNING (A) Bart Holtermans & Cedric Molthoff (B) Bart Holtermans (C) Cedric Molthoff	Week 9					Week 10					Week 11					Week 12				
	1-mrt-2010	2-mrt-2010	3-mrt-2010	4-mrt-2010	5-mrt-2010	8-mrt-2010	9-mrt-2010	10-mrt-2010	11-mrt-2010	12-mrt-2010	15-mrt-2010	16-mrt-2010	17-mrt-2010	18-mrt-2010	19-mrt-2010	22-mrt-2010	23-mrt-2010	24-mrt-2010	25-mrt-2010	26-mrt-2010
Activiteiten	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v
Ontwerpfase																				
Onderzoek uitlezen in/outputs	B	B	B	B	B															
Onderzoek verwerking in/outputs	C	C	C	C	C															
Onderzoek 'real-time' communicatie	B	B	B	B	B															
Prototype software	C	C	C	C	C															
Resultaten bespreken						A	A													
Overleg bedrijfsbegeleider						A	A													
Voorbereidingsfase																				
Mogelijkheden evalueren								A	A	A										
Mogelijke keuzes voorbereiden								A	A	A										
Keuzes maken								A	A	A										
Overleg bedrijfsbegeleider								A	A	A										
Initiatieffase																				
Inlezen Remote Procedure Call (RPC)											C	C	C	C	C					
Inlezen LIBNODAVE											B	B	B	B	B					
Inlezen Mulithreading/Shared memory																B	B	B	B	B
Ontwerpfase																				
Onderzoek RPC																C	C	C	C	C
Onderzoek LIBNODAVE																B	B	B	B	B
Onderzoek Multithreading																B	B	B	B	B

Bijlage A - Tabel 3: Planning (2/5)

AFSTUDEER PLANNING

(A) Bart Holtermans & Cedric Molthoff
(B) Bart Holtermans
(C) Cedric Molthoff

Activiteiten	Week 13					Week 14					Week 15					Week 16				
	29-mrt-2010	30-mrt-2010	31-mrt-2010	1-apr-2010	2-apr-2010	5-apr-2010	6-apr-2010	7-apr-2010	8-apr-2010	9-apr-2010	12-apr-2010	13-apr-2010	14-apr-2010	15-apr-2010	16-apr-2010	19-apr-2010	20-apr-2010	21-apr-2010	22-apr-2010	23-apr-2010
	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v
Realisatiefase						P														
Configureren PLC						A														
Configureren Flexsim						S														
Programmeren LIBNODAVE	B	B	B	B	B	E	B	B	B	B	B	B	B	B	B					
Programmeren RPC	C	C	C	C	C	N	C	C	C	C	C	C	C	C	C					
Programmeren multithreading						-					B	B	B	B	B	B	B	B	B	B
Creëren Flexsim code						P					C	C	C	C	C	C	C	C	C	C
Testopstelling bouwen						A														
Schrijven scriptie	A	A	A	A	A	S	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Handleiding schrijven						E														
Overleg bedrijfsbegeleider		A				N	A					A					A			
Eerste concept scriptie inleveren[1]						-												A		
						P														
						A														
						S														
						E														
						N														
						-														
						P														
						A														
						S														
						E														
						N														
						-														

[1] Woensdag eind van de dag

Bijlage A - Tabel 4: Planning (3/5)

AFSTUDEER PLANNING (A) Bart Holtermans & Cedric Molthoff (B) Bart Holtermans (C) Cedric Molthoff	Week 17					Week 18					Week 19					Week 20				
	26-apr-2010	27-apr-2010	28-apr-2010	29-apr-2010	30-apr-2010	3-mei-2010	4-mei-2010	5-mei-2010	6-mei-2010	7-mei-2010	13-mei-2010	14-mei-2010	15-mei-2010	16-mei-2010	17-mei-2010	17-mei-2010	18-mei-2010	19-mei-2010	20-mei-2010	21-mei-2010
Activiteiten	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v
Realisatiefase					K									H						
Configureren PLC					O									E						
Configureren Flexsim					N									M						
Programmeren LIBNODAVE/multithreading	B	B	B	B	I									E						
Programmeren RPC					N									L						
Creëren Flexsim code	C	C	C	C	G									V						
Testopstelling bouwen					I	A	A	A	A	A				A						
Schrijven scriptie	A	A	A	A	N	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Handleiding schrijven					N	A	A	A	A	A				R						
Overleg bedrijfsbegeleider		A			E		A					A		T			A			
Feedback eerste concept scriptie verwerken[2]	A				D									D						
Tweede concept scriptie inleveren[1]					A								A	A						
Feedback tweede concept scriptie verwerken[2]					G									G			A			
					-									-						
					K									H						
					O									E						
					N									M						
					I									E						
					N									L						
					G									V						
					I									A						
					N									A						
					N									R						
					E									T						

[1] Woensdag eind van de dag
[2] Feedback geleverd door: Dhr. Schouten

Bijlage A - Tabel 5: Planning (4/5)

AFSTUDEER PLANNING

(A) Bart Holtermans & Cedric Molthoff
(B) Bart Holtermans
(C) Cedric Molthoff

Activiteiten		Week 21					Week 22					Week 23					Week 24				
		24-mei-2010	25-mei-2010	26-mei-2010	27-mei-2010	28-mei-2010	31-mei-2010	1-jun-2010	2-jun-2010	3-jun-2010	4-jun-2010	7-jun-2010	8-jun-2010	9-jun-2010	10-jun-2010	11-jun-2010	14-jun-2010	15-jun-2010	16-jun-2010	17-jun-2010	18-jun-2010
		m	d	w	d	v	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v
Realisatiefase		P	S														A				
Presentatie maken		I	C									A	A	A			F		A		
Presentatie voorbereiden		N	R												A	A	S		F		A
		K	I														T		S		F
		S	P														U		T		S
Nazorgfase		T	T														D		U		T
Testen		E	I	A	A	A	A	A									E		D		U
Uitloop vorige fasen		R	E	A	A	A	A	A	A	A	A	A	A	A	A	A	E		E		D
Informatie overdracht aan bedrijf		E							A	A	A						R		E		E
Overleg bedrijfsbegeleider		N	I					A					A						R		E
		-	N														V				R
		P	L														E		V		
		I	E														R		E		V
		N	V														D		R		E
		K	E														E		D		R
		S	R														D		E		D
		T	E														I		D		E
		E	N														G		I		D
		R															I		G		I
		E															N		I		G
		N															G		N		I
		-																	G		N
		P																			G
		I																			

Bijlage A - Tabel 6: Planning (5/5)

9. Kosten en baten

9.1. Kosten

De kosten van dit project zijn:

- Uren Bart Holtermans.
- Uren Cedric Molthoff.
- Eventuele hard- en software (in overleg met bedrijfsbegeleider).

9.2. Baten

De baten van dit project zijn:

- Naar verwachting, een vermindering van de testuren on-site. (zie: Hfst. 2, De opdracht)
  - → Het behoudt van werknemers.
  - → Kostenbesparing.

10. Risico's

10.1. Inleiding

Er zijn een aantal risico's die het afstuderen van de afstudeerder kunnen belemmeren.

- Het te zwaar zijn van de opdracht.
- Miscommunicatie tussen alle 3 de deelnemende partijen.
- Het moeizaam opdoen van kennis.
- Te veel dagen missen.
- Te veel tijdsverlies door de verhuizing van het bedrijf.

Meest van de boven genoemde risico's worden veroorzaakt door externe factoren. Dit zijn dus oorzaken waar de afstudeerders niets aan kunnen doen maar wel rekening mee moeten houden. De afstudeerders kunnen de meeste risico's vermijden door tijdig met de verschillende werkzaamheden te beginnen zodat er eventueel tijd is voor uitloop. Er kan vertraging worden opgevangen door wat meer uren te draaien, dit door middel van extra uren op een dag of een dag in het weekend aan de opdracht besteden. Mocht er echter een te grote vertraging zijn opgelopen en het echt niet meer te redden is om in mei de scriptie in te leveren kan er nog voor gekozen worden om de scriptie in augustus in te leveren, dit is een noodoplossing.

10.2. Risicoanalyse

De risicoanalyse is voltooid aan de hand van het boek van Project management, Roel Grit. In de bijlage 1 is deze risicoanalyse verder uitgesplitst.

Categorie	Risico	Waarde *	Factor **	Zwaarte **	Risicotot.
Tijdsfactor					
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					
4	Aantal functionele deelgebieden dat betrokken is	2	1	4	4
5	Aantal functionele deelgebieden dat gebruik gaat maken van de resultaten	1	0	2	0
6	Gaat het om een aanpassing of een nieuw project?	Geheel nieuw	3	5	15
7	In hoeverre zullen bestaande verantwoordelijkheden moeten wijzigen?	Minimaal	1	5	5
8	Zijn er andere projecten afhankelijk van dit project?	Ja, er is tijd genoeg	1	5	5
9	Wat zal de houding zijn van de gebruikers?	Positief	0	5	0
10	Zijn er deelprojecten, is de voortgang afhankelijk van de coördinatie hiertussen?	Nee	1	3	3

Bijlage A - Tabel 7: Risicanalyse

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

Risicoanalyse vervolg

Categorie	Risico	Waarde *	Factor **	Zwaarte **	Risicotot.
De projectgroep					
11	Welke medewerkers werken aan het project mee?	Beperkt intern	1	4	4
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					
16	Is de projectleiding materiedeskundig?	Beperkt deskundig	4	3	12
17	Hoe deskundig is de projectleiding mbt de projectplanning?	Zeer deskundig	0	3	0
18	Hoeveel ervaring heeft de projectleider met projecten als deze?	Weinig ervaring	3	3	9
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?	Redelijk betrokken	2	5	10
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, alleen maar	0	4	0
Duidelijkheid van het project					
24	Zijn probleem en doelstelling voldoende bekend bij alle projectleden?	Ja, iedereen	0	5	0
25	Is het onderzoeksgebied nauwkeurig vastgelegd?	Ja	0	5	0
26	Is er voldoende afbakening met andere projecten?	Voldoende	0	4	0
27	Is er voldoende tijd gepland voor afstemming en besluitvorming?	Redelijk	1	4	4
28	Zijn de randvoorwaarden duidelijk?	Ja	0	4	0
29	Werken de randvoorwaarden beperkend genoeg?	Ja	0	5	0
			Totaal		**96**
			Risicopercentage ***		**22,17%**

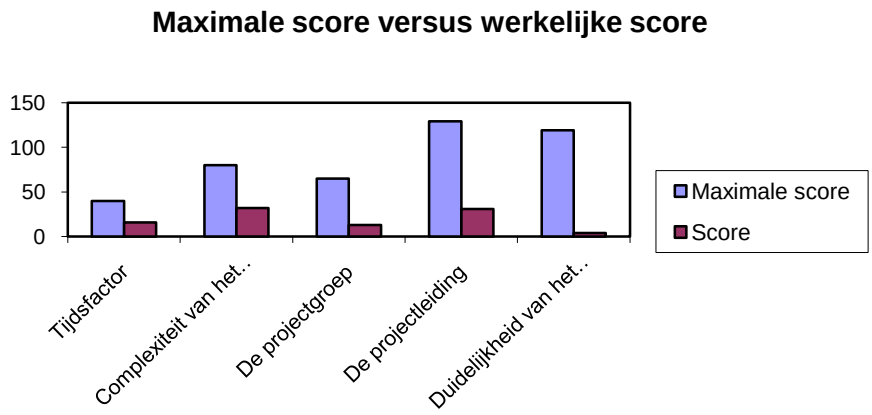
Bijlage A - Tabel 8: Risicoanalyse (vervolg)

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

Risicoanalyse vervolg

Categorie (met maximale score versus werkelijke score)				
Tijdsfactor	Maximaal	40	Score	16
Complexiteit van het project	Maximaal	80	Score	32
De projectgroep	Maximaal	65	Score	13
De projectleiding	Maximaal	129	Score	31
Duidelijkheid van het project	Maximaal	119	Score	4

Bijlage A - Tabel 9: Risicoanalyse (vervolg 2)



Bijlage A - Figuur 8: Resultaat risicoanalyse

Conclusie: Het risicopercentage ligt binnen de grenzen van een aanvaardbaar project. Dit wil zeggen, het percentage ligt onder de 50%. Wanneer het risicopercentage boven de 50% zou komen was het project niet aanvaardbaar.

Bijlagen

BIJLAGE A-1. Template risicoanalyse[i]

Risico	Waarde	Factor		Zwaarte	Risico totaal
Tijdsfactor					
1. Geschatte doorlooptijd van het project	0-3 maanden	0		4	-
	3-6 maanden	1	x		
	6+ maanden	3			
2. Kent het project een definitieve deadline	Nee	0		4	-
	Flexibel	2	x		
	Ja	4			
3. Is de tijd voldoende om het project binnen de gestelde termijn te realiseren	Ruim voldoende	0		4	-
	Voldoende	1	x		
	Onvoldoende	3			
Complexiteit					
4. Aantal disciplines (deelgebieden) dat betrokken is	1	0		4	-
	2	1	x		
	3+	3			
5. Aantal functionele deelgebieden dat gebruikt gaat maken van de resultaten	1	0		2	-
	2-3	1	x		
	4	2			
6. Gaat het om een aanpassing of om een nieuw project	Kleine aanpassing	0		5	-
	Grote aanpassing	2	x		
	Geheel nieuw	3			
7. In hoeverre moeten bestaande verantwoordelijkheden wijzigen (nieuwe)	Niet	0		5	-
	Minimaal	1			
	Gemiddeld	2	x		
	Sterk	3			
8. Zijn er andere projecten afhankelijk van dit project	Nee	0		5	-
	Ja, er is tijd genoeg	1	x		
	Ja, er is weinig tijd	3			
9. Wat zal de houding zijn van de gebruikers	Positief	0		5	-
	Geïnteresseerd	1	x		
	Gereserveerd	2			
10. Zijn er deelprojecten, is de voortgang afhankelijk van de coördinatie hiertussen	Nee	1		3	-
	Enigszin	2	x		
	Sterk	3			

Bijlage A-1 - Tabel 1: Template risicoanalyse

Risico	Waarde	Factor		Zwaarte	Risico totaal
De projectgroep					
11. Welke medewerkers werken aan het project mee	Voor interne Beperkt interne Vooral externe	0 1 3	 x 	4	-
12. Wat is de geografische spreiding van de projecten	1 plaats 1-3 plaatsen 3+	0 1 2	 x 	2	-
13. Aantal projecten dat op piektijden >80% betrokken is	1-5 5-10 10+	0 2 4	 x 	5	-
14. Verhouding materiedeskundigen / projectdeskundigen	Goed Redelijk Slecht	0 2 4	 x 	5	-
15. Nemen gebruikers deel aan de projectgroep	In sterke mate In redelijke maten In beperkte maten	0 1 3	 x 	3	-
De projectleiding					
16. Is de projectleiding materiedeskundig	Zeer deskundig Redelijk deskundig Beperkt deskundig	0 2 4	 x 	3	-
17. Hoe deskundig is de projectleiding m.b.t. projectplanning	Zeer deskundig Redelijk deskundig Beperkt deskundig	0 2 4	 x 	3	-
18. Hoeveel ervaring heeft de projectleiding met projecten als deze	Veel ervaring Redelijk veel ervaring Weinig ervaring	0 1 3	 x 	3	-
19. Hoe deskundig zijn de adviseurs op het te onderzoeken gebied	Zeer deskundig Redelijk deskundig Beperkt deskundig	0 1 3	 x 	5	-
20. Hoe deskundig zijn de materiedeskundigen op het te onderzoeken gebied	Zeer deskundig Redelijk deskundig Beperkt deskundig	0 1 3	 x 	5	-
21. Hoe betrokken zijn de verantwoordelijke lijnmanagers bij het project	Sterk betrokken Redelijk betrokken Beperkt betrokken	0 2 5	 x 	5	-
22. Is de kans groot dat de samenstelling van de projectgroep wijzigt tijdens het project	Kleine kans Gemiddelde kans Grote kans	0 2 5	 x 	5	-
23. Worden door de projectgroep standaardmethoden gebruikt of kiest / maakt men eigen methoden	Ja, alleen maar Ja, een aantal Nee	0 2 4	 x 	4	-

Bijlage A-1 - Tabel 2: Template risicoanalyse (vervolg)

Risico	Waarde	Factor		Zwaarte	Risico totaal
Duidelijkheid van het project					
24. Zijn problemen en doelstellingen voldoende bekend bij alle projectleden	Ja, iedereen De meeste wel Nee	0 1 5	x	5	-
25. Is het onderzoeksgebied nauwkeurig vastgelegd	Ja Redelijk Niet nauwkeurig	0 2 5	x	5	-
26. Is er voldoende afbakening met andere projecten	Voldoende Redelijk Onvoldoende	0 1 3	x	4	-
27. Is er voldoende tijd ingepland voor afstemming en besluitvorming	Voldoende Redelijk Onvoldoende	0 1 3	x	4	-
28. Zijn de randvoorwaarden duidelijk	Ja De meeste wel nee	0 1 5	x	5	-
				Totaal:	.....
Risicopercentage					

$$Risicopercentage^{1} = \frac{TotaalMax.score\ (433)}{} \times 100\% =$$

[1] Indien het percentage hoger dan 50% ligt, dient het project *in deze vorm* niet te worden uitgevoerd.

Bijlage A-1 - Tabel 3: Template risicoanalyse (vervolg 2)

[i] Bron: Roel Grit, Projectmanagement, Wolters Noordhoff, ISBN 90-01-34703-7

BIJLAGE B: Evaluatie bedrijfsbegeleider

EVALUATIE BEDRIJFSBEGELEIDER

EINDEXAMEN 2010

Namen examinandi:	Bart Holtermans	Student ID: 1520310
	Cedric Molthoff	Student ID: 1507692
Opdracht uitgevoerd bij:	Van Riet IA	
Onder leiding van:	Dhr. P. Bosma	

Oordeel over :

A. Het product / resultaat:

Voldoet het eindresultaat aan de opdrachtomschrijving?

Bart en Cedric hebben behoorlijk wat moeite gedaan om een voor hen totaal onbekend gebied binnen het spectrum van de Industriële Automatisering eigen te maken.

Dat is met vallen en opstaan gebeurd, maar uiteindelijk met veel succes gelukt!

Het eindresultaat is een goed werkend geheel, waarmee een zeer goede basis is gelegd voor de verdere implementatie van Siemuflex binnen de Van Riet organisatie.

B. Het proces:

Is het proces op de juiste wijze verlopen en heeft de student voldoende initiatief getoond, o.a. contact met deskundigen, ontwikkelen van concepten, bewaken van de voortgang, etc.

Bart en Cedric waren tijdens de start wat onzeker over hun eigen kunnen, maar hebben gaande weg de smaak te pakken gekregen en zelf initiatieven ondernomen om uiteindelijk daar te komen waar ze nu zijn.

Cedric en Bart hadden wellicht zich iets beter moeten/willen houden aan de door hen zelf gemaakte planning. Daarmee hadden ze zichzelf nog beter kunnen sturen en beter grip gehad op voortgang.

Cedric en Bart zijn gegroeid in hun vakvolwassenheid en het nemen van verantwoordelijkheid t.a.v. beslissingen en doorzetten van een gekozen koers!

Met name hebben zij een grote stap gemaakt in het vastleggen van de uit te voeren en uitgevoerde activiteiten.

Doorzettingsvermogen is een grote kracht van beide heren!

C. Het verslag:

Wordt in het verslag een helder en overtuigend beeld gegeven van de opdracht en de bijdrage van de student?

Het verslag is na wat redigeerwerk door de heer Schouten, ondergetekende, familie en kennissen van Bart en Cedric gegroeid tot een prima document.

De scriptie geeft nu een helder en leesbaar beeld van het doel, de werkzaamheden, het resultaat en het vervolg van de opdracht.

Ik vind Bart en Cedric beiden prettige personen in de omgang en heb evenals alle andere Van Riet medewerkers zeer prettig met hen samen samengewerkt!

Handtekening bedrijfsbegeleider(s):

BIJLAGE C: Programmacode start programma

Hier volgt de programmacode van Start.exe. Het kan zijn dat bepaalde functies uit andere bestanden komen. Deze functies worden hier niet verder behandeld maar zijn wel digitaal terug te vinden op de meegeleverde CD-Rom.

```
1.  // ------------------------------------------------------------------------
2.  // Start.cpp
3.  // This program calls the programs inputs.exe and outputs.exe
4.  //
5.  // ------------------------------------------------------------------------
6.
7.  //include standard C library header files
8.  #include <windows.h>
9.  #include <iostream>
10. #include <conio.h>
11.
12. #include "SharedMemoryLib.h"
13.
14. using namespace std;
15.
16. //prototypes
17. void UserInput(int& Istart, int& Irange);
18. void IPInput(int& IP_Part1, int& IP_Part2, int& IP_Part3, int& IP_Part4);
19.
20. int main()
21. {
22.         //variable declaration
23.         int I_range, I_start, Q_range, Q_start;
24.         int IP_Part1, IP_Part2, IP_Part3, IP_Part4;
25.         int Stopflag = 0;
26.         char Stop;
27.
28.         //create shared memory
29.         CreateSharedMemoryArea();
30.
31.         //ask IP address
32.         IPInput(IP_Part1, IP_Part2, IP_Part3, IP_Part4);
33.
34.         cout << "Please enter the [1] start and [2] range of input Bytes you want to
                write: "<<endl;
35.         UserInput(I_start, I_range);
36.         WriteOnSMStart(I_start,0);
37.         WriteOnSMStart(I_range,1);
38.
39.         cout << "Please enter the [1] start and [2] range of output Bytes you want to
                read: "<<endl;
40.         UserInput(Q_start, Q_range);
41.         WriteOnSMStart(Q_start,3);
42.         WriteOnSMStart(Q_range,4);
43.
44.         //launch inputs.exe and outputs.exe
45.         ShellExecute(NULL, NULL, "Inputs.exe", NULL, NULL, SW_SHOW );
46.         ShellExecute(NULL, NULL, "Outputs.exe", NULL, NULL, SW_SHOW );
47.
48.         //loop for quitting Siemuflex
49.         while(Stopflag == false)
50.         {
51.                 cout << "Enter a 'Q' to quit: ";
52.                 cin >> Stop;
53.                 if (Stop == 'Q' || Stop == 'q')
54.                 {
55.                         Stopflag = 1;
56.                         WriteOnSMStart(Stopflag,9);
57.                 }
58.                 else
59.                 {
60.                         system("cls");
61.                         cout << "Input fault"<<endl;
62.                 }
63.         }
64. cout << "Press ENTER" <<endl;
65. getch();
66. DestroySharedMemoryArea ();
67. }
```

```
68. //FUNCTIONS
69. //UserInput
70. void UserInput(int& start, int& range)
71. {
72.         cout << "[1]: ";
73.         cin >> start;
74.         cout << "[2]: ";
75.         cin >> range;
76.
77. }
78. //IPInput
79. void IPInput(int& IP_Part1, int& IP_Part2, int& IP_Part3, int& IP_Part4)
80. {
81.         cout << "Please enter the IP-adress of the PLC as follow:" << endl;
82.         cout << "[Part 1].[Part 2].[Part 3].[Part 4]" <<endl;
83.         cout << "[Part 1]: ";
84.         cin >> IP_Part1;
85.         cout << "[Part 2]: ";
86.         cin >> IP_Part2;
87.         cout << "[Part 3]: ";
88.         cin >> IP_Part3;
89.         cout << "[Part 4]: ";
90.         cin >> IP_Part4;
91.
92.         //write IP parts in shared memory
93.         WriteOnSMStart(IP_Part1,5);
94.         WriteOnSMStart(IP_Part2,6);
95.         WriteOnSMStart(IP_Part3,7);
96.         WriteOnSMStart(IP_Part4,8);
97. }
```

BIJLAGE D: Programmacode input programma

Hier volgt de programmacode van Inputs.exe. Het kan zijn dat bepaalde functies uit andere bestanden komen. Deze functies worden hier niet verder behandeld maar zijn wel digitaal terug te vinden op de meegeleverde CD-Rom.

```
1.  // ----------------------------------------------------------------------------
2.  // Inputs.cpp
3.  // This program writes inputs from shared memory to a Siemens S7 PLC
4.  // via a TCP/IP connection
5.  // ----------------------------------------------------------------------------
6.
7.  //Specify OS, this is important for the header files
8.  //because the library is written for multiple Operating Systems
9.  #define BCCWIN
10.
11. //include standard C library header files
12. #include <stdio.h>
13. #include <conio.h>
14. #include <iostream>
15. #include <string>
16. #include <vector>
17. #include <bitset>
18. #include <stdlib.h>
19.
20. //include libnodave header files
21. #include "nodave.h"
22. #include "opensocket.h"
23.
24. //include shared memory header file
25. #include "SharedMemoryLib.h"
26.
27. using namespace std;
28.
29. //pointers to structures
30. daveInterface * di;
31. daveConnection * dc;
32.
33. //prototypes functions
34. int Connect();
35. void Disconnect();
36. void InitializeArray();
37.
38. _daveOSserialType fds;
39.
40. void main(int argc, char* argv[])
41. {
42.         //variable declaration
43.         int I_range, I_start, OldState[500], WriteFlag = 0, Stopflag = 0, NewState
44.         char * pc;
45.         const int Width = 8;
46.
47.         //create and initialize the shared memory area (new state array)
48.         OpenSharedMemory();
49.         InitializeArray();
50.
51.         //initialize old state array
52.         for (int i = 0; i < 500; i++)
53.         {
54.                 OldState[i] = 0;
55.         }
56.
57.         //read entered start Byte and range from shared memory
58.         ReadFromSMStart(&I_start,0);
59.         ReadFromSMStart(&I_range,1);
60.
61.         pc = new char[I_range];
62.
63.         //connect with PLC
64.         if (Connect() == 0)
65.         {
66.
67.                 while (Stopflag == false)
68.                 {
69.                 WriteFlag = 0;
70.                         for (int InputByte = I_start; InputByte < (I_start + I_range);
                                InputByte++)
```

```
71.                         {
72.                                 for (int InputBit = 0; InputBit < Width; InputBit++)
73.                                 {
74.                                         ReadFromArray(&NewState, InputByte, InputBit);
75.                                         if (NewState != OldState[InputByte * 8 + InputBit])
76.                                         {
77.                                                 OldState[InputByte * 8 + InputBit] =
                                                    NewState;
78.                                                 WriteFlag = 1;
79.                                         }
80.                                 }
81.                         }
82.                         if (WriteFlag == 1)
83.                         {
84.                                 //write character array (1 character = 1 Byte)
85.                                 for (int InputByte = I_start; InputByte <
                                            (I_start + I_range); InputByte++)
86.                                 {
87.                                         bitset<8> mybits;
88.                                         for (int InputBit = 0; InputBit < Width;
                                                InputBit++)
89.                                         {
90.                                                 //write bits per character/Byte
91.                                                 mybits.set(InputBit,
                                                            OldState[InputByte * 8 + InputBit]);
92.
93.                                         }
94.                                         pc[InputByte] = mybits.to_ulong();
95.                                 }
96.                         //write inputs
97.                         daveWriteBytes(dc, daveInputs, 0, I_start, I_range, pc);
98.                         }
99.                 ReadFromSMStart(&Stopflag,9);
100.                }
101.        }
102.        //disconnect from PLC
103.        Disconnect();
104.        delete[] pc;
105.        DestroySharedMemoryArea ();
106. }
107.
108. //FUNCTIONS
109.
110. //connect to PLC
111. int Connect()
112. {
113.        //variable declaration
114.        int result, IP_Part1, IP_Part2, IP_Part3, IP_Part4, i, offset = 0;
115.        char ip[16];
116.        const int STRSIZE = 4;
117.        char IP_Part1_char[STRSIZE];
118.        char IP_Part2_char[STRSIZE];
119.        char IP_Part3_char[STRSIZE];
120.        char IP_Part4_char[STRSIZE];
121.
122.        //read ip address from shared memory
123.        ReadFromSMStart(&IP_Part1,5);
124.        ReadFromSMStart(&IP_Part2,6);
125.        ReadFromSMStart(&IP_Part3,7);
126.        ReadFromSMStart(&IP_Part4,8);
127.
128.        //convert ip address from int to char
129.        itoa(IP_Part1,IP_Part1_char,10);
130.        itoa(IP_Part2,IP_Part2_char,10);
131.        itoa(IP_Part3,IP_Part3_char,10);
132.        itoa(IP_Part4,IP_Part4_char,10);
133.
134.        //combine up address parts to c string
135.    for (i=0; i<STRSIZE && IP_Part1_char[i]!=0; i++)
136.        ip[offset++]=IP_Part1_char[i];
137.    ip[offset++]='.';
```

```
138.     for (i=0; i<STRSIZE && IP_Part2_char[i]!=0; i++)
139.         ip[offset++]=IP_Part2_char[i];
140.     ip[offset++]='.';
141.     for (i=0; i<STRSIZE && IP_Part3_char[i]!=0; i++)
142.         ip[offset++]=IP_Part3_char[i];
143.     ip[offset++]='.';
144.     for (i=0; i<STRSIZE && IP_Part4_char[i]!=0; i++)
145.         ip[offset++]=IP_Part4_char[i];
146.        ip[offset++]=0;
147.        cout << "Used IP: " << ip <<endl;
148.
149.        fds.rfd = openSocket(102, ip);        //open socket 102 from entered IP adress
150.        fds.wfd = fds.rfd;                    //equal settings for sending and receiving
151.        di = daveNewInterface(fds, "IF1", 0, daveProtoISOTCP, daveSpeed187k);
152.        dc = daveNewConnection(di, 2, 0, 2);  //initialise structure "daveConnection"
153.        result = daveConnectPLC(dc);          //connect to PLC
154.        if (result == 0)
155.        {
156.                cout << "Connected to PLC." << endl;
157.                return 0;
158.        }
159.        else
160.        {
161.                cout << "Error: Could not connect to PLC." << endl;
162.                return 1;
163.        }
164. }
165.
166. //disconnect from PLC
167. void Disconnect()
168. {
169.        daveDisconnectPLC(dc);         //disconnect PLC
170.        daveDisconnectAdapter(di);     //disconnect adapter
171.        closeSocket(fds.rfd);          //close socket
172. }
173.
174. //initialize array
175. void InitializeArray()
176. {
177.        cout << "Initializing shared memory." << endl;
178.        for (int i = 0; i < 100; i++)
179.        {
180.                for(int j = 0; j < 8; j++)
181.                {
182.                        //performs the writing on the shared memory
183.                        WriteOnArray(0, i, j);
184.                }
185.        }
186.        cout << "Initializing shared memory done." << endl;
187. }
```

BIJLAGE E: Programmacode output programma

Hier volgt de programmacode van Outputs.exe. Het kan zijn dat bepaalde functies uit andere bestanden komen. Deze functies worden hier niet verder behandeld maar zijn wel digitaal terug te vinden op de meegeleverde CD-Rom.

```
1.  // ----------------------------------------------------------------------------
2.  // Outputs.cpp
3.  // This program reads outputs from a Siemens S7 PLC via a TCP/IP connection
4.  // and passes them to Flexsim using COM
5.  // ----------------------------------------------------------------------------
6.
7.  //Specify OS, this is important for the Libnodave header files
8.  //because the library is written for multiple Operating Systems
9.  #define BCCWIN
10.
11. //include standard C library header files
12. #include "stdio.h"
13. #include <windows.h>
14. #include <iostream>
15. #include <bitset>
16. #include <stdlib.h>
17. #include <iomanip>
18. #include <vector>
19. #include <oleauto.h>
20. #include <comutil.h>
21. #include <string>
22. #include <fstream>
23.
24. //include libnodave header files
25. #include "nodave.h"
26. #include "opensocket.h"
27.
28. #include "SharedMemoryLib.h"
29.
30. using namespace std;
31.
32. //pointers to structures
33. daveInterface * di;
34. daveConnection * dc;
35.
36. _daveOSserialType fds;
37.
38. //create identifiers
39. static const CLSID CLSID_FlexsimAXApp         =
40. {0x315D4C85,0x9BC9,0x4479,0xB0,0x14,0x2A,0xB9,0x0A,0x3C,0x20,0x6E};
41. static const IID   IID_IFlexsimAXApp          =
42. {0x308FBBF5,0x2854,0x4F47,0xA7,0x60,0x8F,0x92,0x17,0x16,0x0A,0x47};
43. static const IID   LIBID_FlexsimAXDLLLib     =
44. {0x559F2E1B,0xEFD5,0x48A6,0xB1,0xAB,0xD5,0x7D,0x03,0xA9,0xAE,0xC8};
45. static const IID   DIID__IFlexsimAXAppEvents =
46. {0x10AA9419,0xB7DE,0x4A70,0xA3,0xEA,0xC6,0x98,0x02,0x74,0x36,0x1F};
47.
48. //prototypes functions
49. int Connect();
50. void Disconnect();
51. void InitTypeArray(char typearray[][8]);
52. void InitStateArray(int statearray[][8], int Q_start, int Q_range, int Width);
53. void CharToInt(int statearray[][8], int Q_start, int Q_range, int Width, char * pc);
54. void SendNodeFunction(char Task[], int OutputByte, int OutputBit, HRESULT& hrs, struct
                         IFlexsimAXApp * pstruc);
55.
56. //inherit class IFlexsimAXApp from IDispatch
57. interface IFlexsimAXApp : public IDispatch
58.         {
59.     public:
60.
61.         virtual /* [helpstring][id] */ HRESULT __stdcall getinterfacename(
                /* [out] */ CHAR *copyto) = 0;
62.
63.          virtual /* [helpstring][id] */ HRESULT __stdcall OpenModel(
                /* [in] */ BSTR pszString) = 0;

64.          virtual /* [helpstring][id] */ HRESULT __stdcall Go( void) = 0;

65.          virtual /* [helpstring][id] */ HRESULT __stdcall Reset( void) = 0;
66.
```

```
67.         virtual /* [helpstring][id] */ HRESULT __stdcall Stop( void) = 0;
68.
69.         virtual /* [helpstring][id] */ HRESULT __stdcall SetNodeNum(
                /* [string][in] */ BSTR path,
                /* [in] */ double value) = 0;
70.
71.         virtual /* [helpstring][id] */ HRESULT __stdcall SetNodeStr(
                /* [in] */ BSTR path,
                /* [in] */ BSTR value) = 0;
72.
73.         virtual /* [helpstring][id] */ HRESULT __stdcall GetNodeNum(
                /* [in] */ BSTR path,
                /* [retval][out] */ double *retval) = 0;
74.
75.         virtual /* [helpstring][id] */ HRESULT __stdcall GetNodeStr(
                /* [in] */ BSTR path,
                /* [retval][out] */ BSTR *value) = 0;
76.
77.         virtual /* [helpstring][id] */ HRESULT __stdcall SetTableNum(
                /* [in] */ BSTR table,
                /* [in] */ int row,
                /* [in] */ int column,
                /* [in] */ double value) = 0;
78.
79.         virtual /* [helpstring][id] */ HRESULT __stdcall SetTableStr(
                /* [in] */ BSTR table,
                /* [in] */ int row,
                /* [in] */ int column,
                /* [in] */ BSTR value) = 0;
80.
81.         virtual /* [helpstring][id] */ HRESULT __stdcall GetTableNum(
                /* [in] */ BSTR table,
                /* [in] */ int row,
                /* [in] */ int column,
                /* [retval][out] */ double *retval) = 0;
82.
83.         virtual /* [helpstring][id] */ HRESULT __stdcall GetTableStr(
                /* [in] */ BSTR table,
                /* [in] */ int row,
                /* [in] */ int column,
                /* [retval][out] */ BSTR *retval) = 0;
84.
85.         virtual /* [helpstring][id] */ HRESULT __stdcall NodeFunction(
                /* [in] */ BSTR node,
                /* [in] */ double p1,
                /* [in] */ double p2,
                /* [in] */ double p3,
                /* [in] */ double p4,
                /* [in] */ double p5,
                /* [in] */ double p6,
                /* [in] */ double p7,
                /* [in] */ double p8,
                /* [in] */ double p9,
                /* [retval][out] */ double *retval) = 0;
86.
87.         virtual /* [helpstring][id] */ HRESULT __stdcall StartApplication(
                /* [in] */ BOOL minimized) = 0;
88.
89.         virtual /* [helpstring][id] */ HRESULT __stdcall ExitApplication( void) = 0;
90.
91.         virtual /* [helpstring][id] */ HRESULT __stdcall ExecuteString(
                /* [in] */ BSTR execString,
                /* [retval][out] */ double *retval) = 0;
92.
93.         virtual /* [helpstring][id] */ HRESULT __stdcall SetRunSpeed(
                /* [in] */ double speed) = 0;
94.
95.
96.         virtual /* [helpstring][id] */ HRESULT __stdcall SetNotificationTime(
                /* [in] */ double simtime) = 0;
97.
```

```
98.         virtual /* [helpstring][id] */ HRESULT __stdcall SetUpdateInterval(
                /* [in] */ int milliseconds,
                /* [in] */ BOOL onlywhenrunning) = 0;
99.
100.        virtual /* [helpstring][id] */ HRESULT __stdcall OpenLibrary(
                /* [in] */ BSTR pszString) = 0;
101.
102.        virtual /* [helpstring][id] */ HRESULT __stdcall AddNodeInto(
                /* [string][in] */ BSTR path,
                /* [string][in] */ BSTR name,
                /* [in] */ int type) = 0;
103.
104.        virtual /* [helpstring][id] */ HRESULT __stdcall AddNodeAfter(
                /* [string][in] */ BSTR path,
                /* [string][in] */ BSTR name,
                /* [in] */ int type) = 0;
105.    };

106. int main()
107. {
108.        //variable declaration
109.        const int Height = 100, Width = 8;
110.        int Q_range = 0, Q_start = 0, OldState[Height][Width], NewState[Height][Width];
111.        int Equal = 0;
112.        char * pc, Type[Height][Width];
113.        int Stopflag = 0;
114.
115.        //Flexsim model location
116.        BSTR ModelPath =
            SysAllocString(OLESTR("c:\\\\Siemuflex\\\\Flexsim\\\\Siemuflex.fsm"));
117.        //Handle
118.        HRESULT hr;
119.        IFlexsimAXApp * pIFAXA = NULL;
120.        //Start COM and Flexsim
121.        hr = CoInitialize(NULL);
122.        hr = CoCreateInstance(CLSID_FlexsimAXApp, NULL, CLSCTX_INPROC_SERVER,
                                  IID_IFlexsimAXApp, (void**)&pIFAXA);
123.        hr = pIFAXA -> StartApplication(true);
124.        cout << "Flexsim application launched." << endl;
125.        hr = pIFAXA->OpenModel(ModelPath);
126.        cout << "Flexsim model has been opened." << endl;
127.        OpenSharedMemory();
128.        ReadFromSMStart(&Q_start,3);
129.        ReadFromSMStart(&Q_range,4);
130.        pc = new char[Q_range];
131.
132.        //initialize array OldState & NewState
133.        InitStateArray(OldState, Q_start, Q_range, Width);
134.        InitStateArray(NewState, Q_start, Q_range, Width);
135.
136.        //initalize array Type
137.        InitTypeArray(Type);
138.
139.        //connect with PLC
140.        if (Connect() == 0)
141.        {
142.                while (Stopflag == false)
143.                {
144.                        //read outputs
145.                        daveReadBytes(dc,daveOutputs, 0, Q_start, Q_range, pc);
146.
147.                        //outputs from char to integer array
148.                        CharToInt(NewState, Q_start, Q_range, Width, pc);
149.
150.                        //compare arrays
151.                        for (int OutputByte = Q_start; OutputByte < (Q_start + Q_range);
                                OutputByte++)
152.                        {
153. for (int OutputBit = 0; OutputBit < Width; OutputBit++)
154. {
155.        //if value has changed from 0 to 1
```

```
156.        if (NewState[OutputByte][OutputBit] > OldState[OutputByte][OutputBit])
157.        //to true
158.        {
159.                Equal = 0;
160.                switch(Type[OutputByte][OutputBit])
161.                {
162.                        case 'M':
163.                        {
164.                        char Task[30] = "//Qx_x>labels//START";
165.                        SendNodeFunction(Task, OutputByte, OutputBit, hr, pIFAXA);
166.                        }break;
167.
168.                        case 'F':
169.                        {
170.                        char Task[30] = "//Qx_x>labels//FORWARD";
171.                        SendNodeFunction(Task, OutputByte, OutputBit, hr, pIFAXA);
172.                        }break;
173.
174.                        case 'B':
175.                        {
176.                        char Task[30] = "//Qx_x>labels//BACKWARD";
177.                        SendNodeFunction(Task, OutputByte, OutputBit - 1, hr, pIFAXA);
178.                        }break;
179.
180.                        case 'H':
181.                        {
182.                        char Task[30] = "//Qx_x>labels//START";
183.                        SendNodeFunction(Task, OutputByte, OutputBit, hr, pIFAXA);
184.                        }break;
185.
186.                        case 'S':
187.                        {
188.                        char Task[30]="//Qx_x>labels//OPENINPUT";
189.                        SendNodeFunction(Task, OutputByte, OutputBit, hr, pIFAXA);
190.                        }break;
191.
192.                        default:
193.                        cout << "Error: No type found." << endl;
194.                }
195.                OldState[OutputByte][OutputBit] = NewState[OutputByte][OutputBit];
196.        }
197.        //if value has changed from 1 to 0
198.        else if ( NewState[OutputByte][OutputBit] < OldState[OutputByte][OutputBit] )
199.        //to false
200.        {
201.                Equal = 0;
202.                switch(Type[OutputByte][OutputBit])
203.                {
204.                        case 'M':
205.                        {
206.                        char Task[30] = "//Qx_x>labels//STOP";
207.                        SendNodeFunction(Task, OutputByte, OutputBit, hr, pIFAXA);
208.                        }break;
209.
210.                        case 'F':
211.                        {
212.                        char Task[30] = "//Qx_x>labels//STOP";
213.                        SendNodeFunction(Task, OutputByte, OutputBit, hr, pIFAXA);
214.                        }break;
215.
216.                        case 'B':
217.                        {
218.                        char Task[30] = "//Qx_x>labels//STOP";
219.                        SendNodeFunction(Task, OutputByte, OutputBit - 1, hr, pIFAXA);
220.                        }break;
221.
222.
223.                        case 'H':
224.                        {
225.                        char Task[30] = "//Qx_x>labels//STOP";
226.                        SendNodeFunction(Task, OutputByte, OutputBit, hr, pIFAXA);
```

```
227.                        }break;
228.
229.                        case 'S':
230.                        {
231.                        char Task[30]="//Qx_x>labels//BLOCKINPUT";
232.                        SendNodeFunction(Task, OutputByte, OutputBit, hr, pIFAXA);
233.                        }break;
234.
235.                        default:
236.                        cout << "Error: No type found." << endl;
237.                }
238.                OldState[OutputByte][OutputBit] = NewState[OutputByte][OutputBit];
239.        }
240.        //if value has not changed
241.        else if ( NewState[OutputByte][OutputBit] == OldState[OutputByte][OutputBit] )
242.        {
243.                Equal++;
244.                if (Equal == 50000)
245.                {
246.                        cout << "Runtime error: No output has changed
                                for 50000 cycles." << endl;
247.                        Equal = 0;
248.                }
249.        }
250.        }
251.        }
252.        ReadFromSMStart(&Stopflag,9);
253.        }
254.        }
255.        Disconnect();  //Disconnect from PLC
256.        delete[] pc;   //destroy dynamic array
257.
258.        hr = pIFAXA->ExitApplication();       //Close Flexsim
259.        CoUninitialize();                     //Disconnect from COM-Object
260.        DestroySharedMemoryArea ();           //Destroy shared memory
261.        return 0;
262. }
263.
264. //FUNCTIONS
265. //connect to PLC
266. int Connect()
267. {
268.        //variable declaration
269.        int result, IP_Part1, IP_Part2, IP_Part3, IP_Part4, i, offset = 0;
270.        char ip[16];
271.        const int STRSIZE = 4;
272.        char IP_Part1_char[STRSIZE];
273.        char IP_Part2_char[STRSIZE];
274.        char IP_Part3_char[STRSIZE];
275.        char IP_Part4_char[STRSIZE];
276.
277.        //read ip address from shared memory
278.        ReadFromSMStart(&IP_Part1,5);
279.        ReadFromSMStart(&IP_Part2,6);
280.        ReadFromSMStart(&IP_Part3,7);
281.        ReadFromSMStart(&IP_Part4,8);
282.
283.        //convert ip address from int to char
284.        itoa(IP_Part1,IP_Part1_char,10);
285.        itoa(IP_Part2,IP_Part2_char,10);
286.        itoa(IP_Part3,IP_Part3_char,10);
287.        itoa(IP_Part4,IP_Part4_char,10);
288.
289.        //combine up address parts to c string
290.        for (i=0; i<STRSIZE && IP_Part1_char[i]!=0; i++)
291.                ip[offset++]=IP_Part1_char[i];
292.        ip[offset++]='.';
293.        for (i=0; i<STRSIZE && IP_Part2_char[i]!=0; i++)
                ip[offset++]=IP_Part2_char[i];
294.        ip[offset++]='.';
295.        for (i=0; i<STRSIZE && IP_Part3_char[i]!=0; i++)
```

```
296.                ip[offset++]=IP_Part3_char[i];
297.        ip[offset++]='.';
298.        for (i=0; i<STRSIZE && IP_Part4_char[i]!=0; i++)
299.                ip[offset++]=IP_Part4_char[i];
300.        ip[offset++]=0;
301.        cout << "Used ip: " << ip <<endl;
302.
303.        fds.rfd = openSocket(102, ip);        //open socket 102 from entered IP adress
304.        fds.wfd = fds.rfd;                    //equal settings for sending and receiving
305.        di = daveNewInterface(fds, "IF1", 0, daveProtoISOTCP, daveSpeed187k);
306.        dc = daveNewConnection(di, 2, 0, 2);  //initialise structure "daveConnection"
307.        result = daveConnectPLC(dc);          //connect to PLC
308.        if (result == 0)
309.        {
310.                cout << "Connected to PLC." << endl;
311.                return 0;
312.        }
313.        else
314.        {
315.                cout << "Error: Could not connect to PLC." << endl;
316.                return 1;
317.        }
318. }
319.
320. //disconnect from PLC
321. void Disconnect()
322. {
323.        daveDisconnectPLC(dc);                //disconnect PLC
324.        daveDisconnectAdapter(di);            //disconnect adapter
325.        closeSocket(fds.rfd);                 //close socket
326. }
327.
328. //initialize Type array
329. void InitTypeArray(char typearray[][8])
330. {
331.        string line;
332.        ifstream myfile ("..//Include//Typelijst.txt");
333.
334.        if (myfile.is_open())
335.        {
336.                while (! myfile.eof() )
337.                {
338.                        //read line from file "Typelijst"
339.                        getline (myfile, line);
340.                        if (line.size() >= 6 && line.size() <= 8 && *line.begin() == 'Q')
341.                        {
342.                                string::iterator nrtype = line.end() - 1;
343.                                if (*nrtype == ' ')
344.                                {
345.                                        const char* CByte = &line[1];
346.                                        int IByte = atoi(CByte);
347.                                        cout    << "Q" << IByte << "." << "x"
348.                                                <<" Syntax error: Wrong output number,
                                                  please remove whitespaces." << endl;
349.                                }
350.                                else
351.                                {
352.                                        const char* CByte = &line[1];
353.                                        const char* CBit = &line[ line.size() - 3  ];
354.
355.                                        int IByte = atoi(CByte);
356.                                        int IBit  = atoi(CBit);
357.                                        typearray[IByte][IBit] = *nrtype;
358.                                        cout << "Q" << IByte << "." << IBit << " is a: "
359.                                             << typearray[IByte][IBit] << endl;
360.                                }
361.                        }
362.                }
363.        myfile.close();
364.        }
365.        else cout << "Unable to open file";
```

```
366. }
367.
368. //initialize State array
369. void InitStateArray(int statearray[][8], int Q_start, int Q_range, int Width)
370. {
371.         for (int OutputByte = Q_start; OutputByte < Q_range; OutputByte++)
372.         {
373.                 for (int OutputBit = 0; OutputBit < Width; OutputBit++)
374.                 {
375.                         statearray[OutputByte][OutputBit] = 0;
376.                 }
377.         }
378. }
379.
380. //outputs from char to integer array
381. void CharToInt(int statearray[][8], int Q_start, int Q_range, int Width, char * pc)
382. {
383.         const size_t bits = 8;
384.         bitset< bits > bitwise_value;
385.         for (int OutputByte = Q_start; OutputByte < (Q_start + Q_range); OutputByte++)
386.         {
387.                 for (int OutputBit = 0; OutputBit < Width; OutputBit++)
388.                 {
389.                         //set bits per char
390.                         bitwise_value[OutputBit] = (pc[OutputByte] >> OutputBit) & 1;
391.                         statearray[OutputByte][OutputBit] = bitwise_value[OutputBit];
392.                 }
393.         }
394. }
395.
396. //send nodefunction via COM to Flexsim
397. void SendNodeFunction(char Task[], int OutputByte, int OutputBit, HRESULT& hrs, struct
                         IFlexsimAXApp * pstruc)
398. {
399.         char TaskOutputByte[2];
400.         char TaskOutputBit[2];
401.
402.         itoa(OutputByte, TaskOutputByte, 10);
403.         itoa(OutputBit, TaskOutputBit, 10);
404.         Task[3] = TaskOutputByte[0];
405.         Task[5] = TaskOutputBit[0];
406.
407.         BSTR CloseInputString = _com_util::ConvertStringToBSTR(Task);
408.
409.         double p1 = NULL, p2 = NULL, p3 = NULL, p4 = NULL, p5 = NULL;
410.         double p6 = NULL, p7 = NULL, p8 = NULL, p9 = NULL;
411.         double * pNF;
412.         double sjaak = 0;
413.         pNF = &sjaak;
414.
415.         hrs = pstruc->NodeFunction(CloseInputString,p1, p2, p3, p4, p5, p6, p7, p8, p9,
416.                                    pNF);
417.         SysFreeString(CloseInputString);
418. }
```

BIJLAGE F: Programmacode SiemuflexDLL

Hier volgt de code van SiemuflexDLL.DLL. Het kan zijn dat bepaalde functies uit andere bestanden komen. Deze functies worden hier niet verder behandeld maar zijn wel digitaal terug te vinden op de meegeleverde CD-Rom.

1. Hoofdcode

Hier volgt de programmacode van SiemuflexDLL.dll.

```
1.  // ------------------------------------------------------------------------
2.  // InputMyDLL.cpp
3.  // This program contains functions for writing inputs in shared memory
4.  //
5.  // ------------------------------------------------------------------------
6.
7.  //include standard C library header files
8.  #include <stdio.h>
9.  #include <sstream>
10. #include <string>
11.
12. //include Flexsim functions and definitions header files
13. #include "FlexsimDefs.h"
14.
15. //include shared memory header files
16. #include "SharedMemoryLib.h"
17.
18. //include custom DLL functions header files
19. #include "InputMyDLL.h"
20.
21. using namespace std;
22.
23. visible double SetInput(FLEXSIMINTERFACE)
24. {
25.         string InputNr = parstr(1);
26.         int lengte = InputNr.size();
27.         int pos1 = 0, pos2 = 0, data = 1;
28.
29.         if (lengte >= 4 && lengte <= 7)
30.         {
31.                 bool juist = CheckInputNr(lengte, InputNr);
32.                 if(juist)
33.                 {
34.                         pt(InputNr); pt(" = 1"); pr();
35.                         int int1 = CreateInputRowNr(InputNr);
36.                         int int2 = CreateInputColNr(lengte, InputNr);
37.                         pos1 = int1;
38.                         pos2 = int2;
39.                 }
40.                 else
41.                 {
42.                         pt("Syntax error: Wrong input number"); pr();
43.                 }
44.                 OpenSharedMemory();
45.                 WriteOnArray(1, pos1, pos2); // performs the writing
46.                 DestroySharedMemoryArea ();
47.         }
48.         else
49.         {
50.                 pt("Syntax error: The length of the input number is not valid");
51.                 pr();
52.         }
53.         return 0;
54. }
55.
56. visible double ResetInput(FLEXSIMINTERFACE)
57. {
58.         string InputNr = parstr(1);
59.         int lengte = InputNr.size();
60.         int pos1 = 0, pos2 = 0, data = 1;
61.
62.         if (lengte >= 4 && lengte <= 7)
63.         {
64.                 bool juist = CheckInputNr(lengte, InputNr);
65.                 if(juist)
66.                 {
67.                         pt(InputNr); pt(" = 0"); pr();
```

```
68.                         int int1 = CreateInputRowNr(InputNr);
69.                         int int2 = CreateInputColNr(lengte, InputNr);
70.                         pos1 = int1;
71.                         pos2 = int2;
72.                 }
73.                 else
74.                 {
75.                         pt("Syntax error: Wrong input number"); pr();
76.                 }
77.                 OpenSharedMemory();
78.                 WriteOnArray(0, pos1, pos2); // performs the writing
79.                 DestroySharedMemoryArea ();
80.         }
81.         else
82.         {
83.                 pt("Syntax error: The length of the input number is not valid");
84.                 pr();
85.         }
86.         return 0;
87. }
```

2. Functiecode

Hier volgen de bijbehorende functies.

```
1.  // ----------------------------------------------------------------------------
2.  // InputMyDLL.cpp
3.  // This program contains functions for checking the input string
4.  //
5.  // ----------------------------------------------------------------------------
6.
7.  //include standard C library header files
8.  #include <sstream>
9.  #include <string>
10.
11. using namespace std;
12.
13. bool CheckInputNr(int &len, string &number)
14. {
15.                                                             //inputstring:
16.         string::iterator      LetterI = number.begin(),     //-> start (I)
17.                               Lpunt   = LetterI + len - 2,  //-> start + length - 2 (.)
18.                               Laatste = number.end() - 1;   //-> last number
19.
20.         //Check the input string for the following properties:
21.         //-> Is the first character an I?
22.         //-> Is the point on the correct position
23.         //-> Is the last (numeral) character smaller than 8?
24.         if(*LetterI ==(char) 'I' && *Lpunt == (char) '.' &&
               *Laatste != (char) '8' && *Laatste != (char) '9')
25.         {
26.                 return true;
27.         }
28.         else
29.         {
30.                 return false;
31.         }
32. }
33. int CreateInputRowNr(string &number)
34. {
35.         const char* str_int1 = &number[1];
36.         int int1 = atoi(str_int1);
37.         return int1;
38. }
39.
40. int CreateInputColNr(int &len, string &number)
41. {
42.         const char* str_int2 = &number[len-1];
43.         int int2 = atoi(str_int2);
44.         return int2;
45. }
```

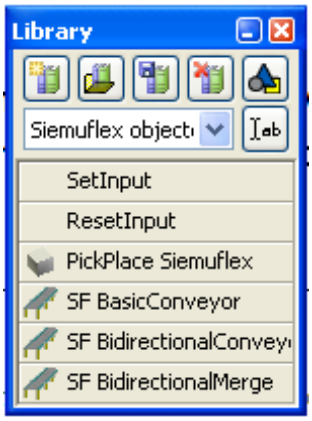
BIJLAGE G: Flexsim bibliotheek

Inhoudsopgave

Lijst van figuren bijlage G

1. Inleiding

In deze bijlage worden de objecten uit de bibliotheek Siemuflex Library behandeld. Per object wordt bekeken welke functie hij bevat en op welke manier met hem omgegaan dient te worden.



Bijlage G - Figuur 1: Siemuflex Library

2. SetInput/ResetInput

2.1. Functie(s)

Deze twee objecten binnen de bibliotheek zorgen er voor dat de juiste DLL aangeroepen wordt. Deze DLL verzorgt het wegschrijven van de inputs.

2.2. Gebruik

Het gebruik van deze twee objecten is zeer eenvoudig. Sleep de objecten een voor een op de Orthographic View. Er zal een melding verschijnen dat het object is toegevoegd is aan de User Commands.

3. Siemuflex Conveyor

3.1. Functies

Dit is een standaard lopende band welke aan en uit gezet kan worden via COM. Wanneer het model gereset wordt zal de lopende band automatisch in stop vallen.

Bijlage G - Figuur 2: Siemuflex Conveyor

De nodefuncties:

- START

```
1.  /**Start motor*/
2.  treenode current = ownerobject(c);                    //Wijst naar zich zelf
3.
4.  if (getlabelnum(current,"BLOKKADE") == 0)             //Indien al gestart
5.  {                                                     //Voorkomt dubbele deblokkering
6.     pt(getname(current));
7.     pt(" is reeds gedeblokkeerd");
8.     pr();
9.  }
10. else                                                  //Indien niet gestart
11. {                                                     //Laad de texture met richting
12.    loadimage("Textures\\VRtransportbandVooruit.bmp","VRtransportbandVooruit");
                                                          //pas de texture toe
13.    setobjecttextureindex(current,gettextureindex("Textures\\VRtransportbandVooruit
                                                          .bmp"));
14.
15.    resumeobject(current);                             //Start de motor
16.    setlabelnum(current,"BLOKKADE",0);
17. }
```

- STOP

```
/**Stop motor*/
treenode current = ownerobject(c);                          //Wijst naar zich zelf

if (getlabelnum(current,"BLOKKADE") == 1)                   //Indien al gestopt
{                                                           //Voorkomt dubbele blokkering
   pt(getname(current));
   pt(" is reeds geblokkeerd");
   pr();
}
else                                                        //Indien niet gestart
{                                                           //Laad de basis texture
   loadimage("Textures\\VRtransportband.bmp","VRtransportband");
                                                            //pas de texture toe
   setobjecttextureindex(current,gettextureindex("Textures\\VRtransportband.bmp"));

   stopobject(current, STATE_BLOCKED);                      //Stop de motor
   setlabelnum(current,"BLOKKADE",1);
}
```

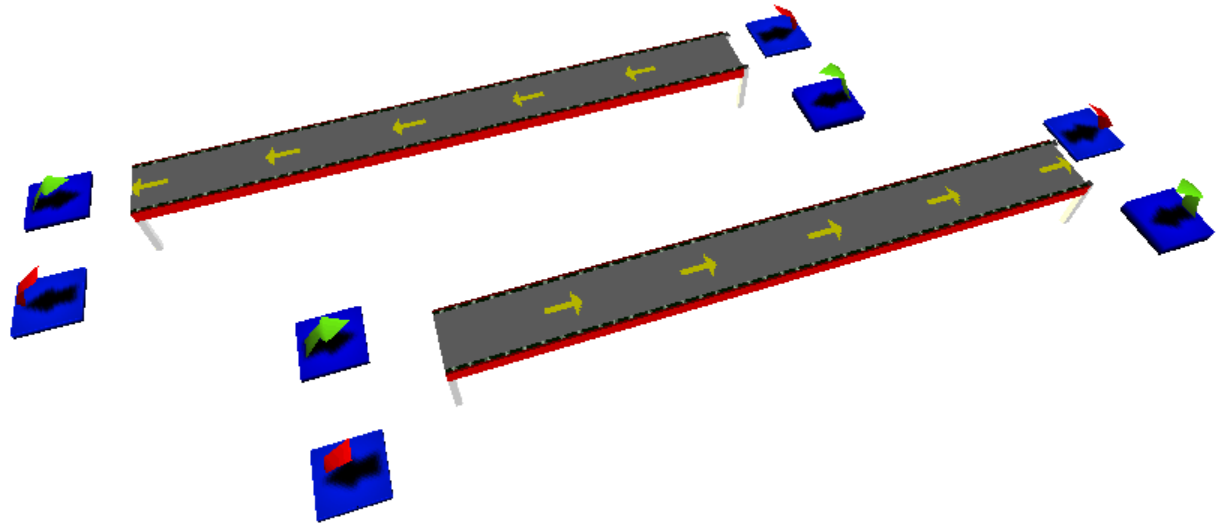
3.2. Gebruik

Dit object kan ook in de Orthographic View gesleept worden. De in- en output van dit object kan naar wens worden ingevuld. Een Flowitem, bijvoorbeeld een pakketje, zal van het begin naar het eind gaan met een constante snelheid. Aan het eind verlaat het pakketje de band via de eerst beschikbare output (mochten meerdere outputs beschikbaar zijn, zal het pakketje de bovenste output van de outputs nemen. Zie voor meer informatie: Flexsimhandleiding → "ranking").

4. Siemuflex Bidirectional Conveyor

4.1. Functies

Dit object lijkt op het voorgaande object maar kan ook achteruit.



Bijlage G - Figuur 3: Siemuflex Bidirectional Conveyor

De nodefuncties met code zijn dan:

- STOP

```
/**Stop motor*/
treenode current = ownerobject(c);                    //Wijst naar zich zelf

if (getlabelnum(current,"BLOKKADE") == 1)             //Indien al gestopt
{                                                     //Voorkomt dubbele blokkering
   pt(getname(current));
   pt(" is reeds geblokkeerd");
   pr();
}
else                                                  //Indien niet gestart
{                                                     //Laad de basis texture
   loadimage("Textures\\VRtransportband.bmp","VRtransportband");
                                                      //pas de texture toe
   setobjecttextureindex(current,gettextureindex("Textures\\VRtransportband.bmp"));

   stopobject(current, STATE_BLOCKED);                //Stop de motor
   setlabelnum(current,"BLOKKADE",1);
}
```

- FORWARD

```
treenode current = ownerobject(c);                          //Wijst naar zichzelf
treenode conv = current;                                    //Hernoeming

if (getlabelnum(current,"BLOKKADE") == 0)                   //Vorige situatie geen stop
{
   setlabelnum(current,"RICHTING",1);                       //Set de variabele RICHTING
   bcsetdirection(conv,1);                                  //Zet de band in vooruit
                                                            //Laad de texture "vooruit"
                                                            //en pas toe
   loadimage("Textures\\VRtransportbandVooruit.bmp","VRtransportbandVooruit");
   setobjecttextureindex(current,gettextureindex("Textures\\VRtransportbandVooruit.bmp"));
   receiveitem(current,1);                                  //Ontvang een item

for(int i = 1; i <= content(conv); i++)                     //Stuur alle items die al
   {                                                        //op de band lagen de goede
   treenode item = rank(conv, i);                           //richting op
   bcsetitemconveystate(conv, item, bcgetitemposition(conv, item), 0, 1, 0);
   }
}
Else                                                        //Vorige situatie stop
{
   setlabelnum(current,"RICHTING",1);                       //Set de variabele RICHTING
                                                            //Laad de texture "vooruit"
                                                            //en pas toe
   loadimage("Textures\\VRtransportbandVooruit.bmp","VRtransportbandVooruit");
   setobjecttextureindex(current,gettextureindex("Textures\\VRtransportbandVooruit.bm p"));
   bcsetdirection(conv,1);                                  //Zet de band aan (vooruit)
   receiveitem(current,1);                                  //Ontvang een item
   resumeobject(current);                                   //Start het object
   setlabelnum(current,"BLOKKADE",0);
for(int i = 1; i <= content(conv); i++)                     //Stuur alle items die al
   {                                                        //op de band lagen de goede
   treenode item = rank(conv, i);                           //richting op
   bcsetitemconveystate(conv, item, bcgetitemposition(conv, item), 0, 1, 0);
   }
}
```

- BACKWARD

```
1.  treenode current = ownerobject(c);                          //Wijst naar zichzelf
2.  treenode conv = current;                                    //Hernoeming

3.  if (getlabelnum(current,"BLOKKADE") == 0)                   //Vorige situatie geen stop
4.  {
5.     setlabelnum(current,"RICHTING",0);                       //Set de variabele RICHTING
6.     bcsetdirection(conv,0);                                  //Zet de band in vooruit
                                                                //Laad de texture "vooruit"
                                                                //en pas toe
7.     loadimage("Textures\\VRtransportbandAchteruit.bmp","VRtransportbandAchteruit");
8.     setobjecttextureindex(current,gettextureindex("Textures\\VRtransportbandAchteruit.bmp")
       );
9.     receiveitem(current,2);                                  //Ontvang een item

10. for(int i = 1; i <= content(conv); i++)                     //Stuur alle items die al
11.    {                                                        //op de band lagen de goede
12.    treenode item = rank(conv, i);                           //richting op
13.    bcsetitemconveystate(conv, item, bcgetitemposition(conv, item), 0, 1, 0);
14.    }
15. }
16. else
17. {
18.    setlabelnum(current,"RICHTING",0);                       //Set de variabele RICHTING
                                                                //Laad de texture "vooruit"
                                                                //en pas toe
19.    loadimage("Textures\\VRtransportbandAchteruit.bmp","VRtransportbandAchteruit");
20.    setobjecttextureindex(current,gettextureindex("Textures\\VRtransportbandAchteruit.bmp")
    );
21.    bcsetdirection(conv,0);                                  //Zet de band in achteruit
22.    receiveitem(current,2);                                  //Ontvang een item
23.    resumeobject(current);                                   //Start het object
24.    setlabelnum(current,"BLOKKADE",0);

25. for(int i = 1; i <= content(conv); i++)                     //Stuur alle items die al
26.    {                                                        //op de band lagen de goede
27.    treenode item = rank(conv, i);                           //richting op
28.    bcsetitemconveystate(conv, item, bcgetitemposition(conv, item), 0, 1, 0);
29.    }
30. }
```

In tegenstelling tot de Siemuflex Conveyor heeft de Siemuflex Bidirectional Conveyor naast de nodefuncties ook een OnEntry trigger. De OnEntry trigger wordt geactiveerd wanneer een item de lopende band op komt en zorgt er voor dat een volgend item de band op mag komen. De eerste keer zal deze geactiveerd worden wanneer de nodefunctie FORWARD of BACKWARD wordt aangeroepen.

- OnEntry trigger

```
1.  /**Ontvang een item*/
2.  treenode item = parnode(1);                          //Verwijzing naar het item
3.  treenode current = ownerobject(c);                   //Verwijzing naar de lopende band
4.  int port = parval(2);                                //Verwijzing naar de poort
5.  treenode richting = getlabelnum(current,"RICHTING"); //Haal looprichting van de band op

6.  if (richting == 1)                                   //Wanneer de richting FORWARD is
7.  {
8.     receiveitem(current,1);                           //Ontvang een item van
                                                         //inputpoort 1
9.  bcsetitemconveystate(current,item,0,0,1,0);          //Geef het ontvangen item een
                                                         //beginpositie, een startsnelheid,
                                                         //een eindsnelheid en een
                                                         //acceleratie mee
10.
11. }
12. else if (richting == 0)                              //Wanneer de richting BACKWARD is
13. {
14. receiveitem(current,2);                              //Ontvang een item van inputpoort 2
15. bcsetitemconveystate(current,item,10,0,1,0);         //Geef het ontvangen item een
                                                         //beginpositie, een startsnelheid,
                                                         //een eindsnelheid en een
                                                         //acceleratie mee
```

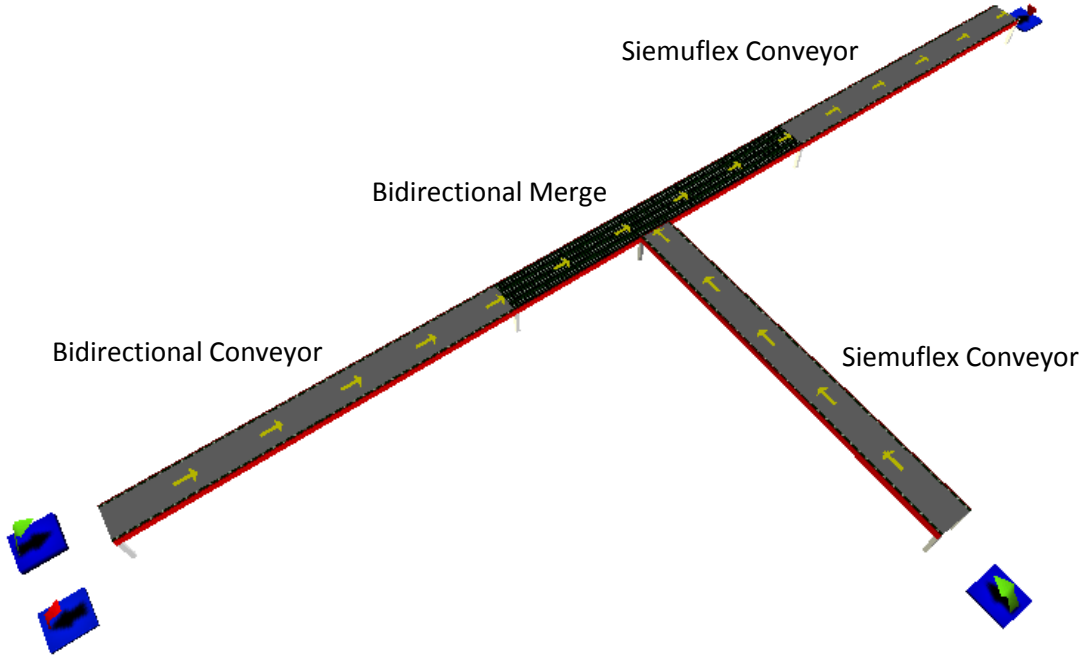
4.2. Gebruik

Het gebruik van dit object is hetzelfde als bij Siemuflex Conveyor. Hier dient wel goed opgelet te worden op de volgorde van in- en outputs. Sluit eerst de in- en output voor FORWARD aan, daarna de in- en output voor BACKWARD. Wanneer deze methode gevolgd word kan het eigenlijk niet fout gaan. Er zijn dus in totaal twee inputs en twee outputs.

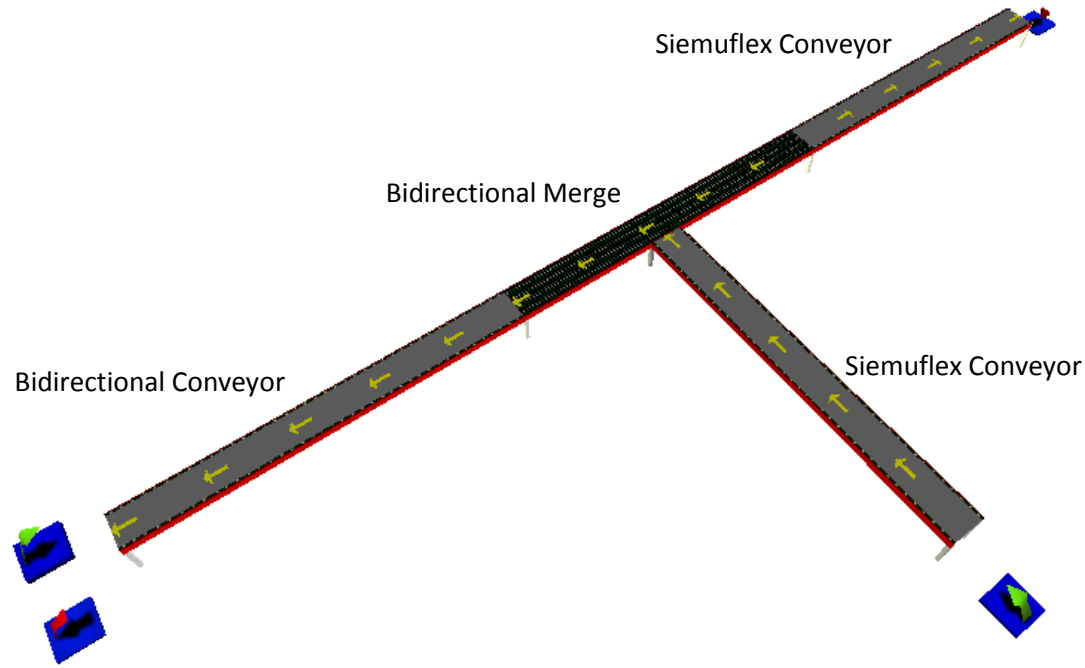
5. Siemuflex Bidirectional Merge

5.1. Functies

Dit is een speciaal object. De volgende situaties zijn mogelijk:



Bijlage G - Figuur 4: Siemuflex Bidirectional Merge (FORWARD)



Bijlage G - Figuur 5: Siemuflex Bidirectional Merge (BACKWARD)

Wanneer de eerste situatie zich voordoet zijn er twee input mogelijkheden en één output mogelijkheid. In de tweede situatie blijft er maar één input mogelijkheid over en één outputmogelijkheid.

De functies met code zijn dan:

- STOP

```
/**Stop motor*/
treenode current = ownerobject(c);                          //Wijst naar zich zelf

if (getlabelnum(current,"BLOKKADE") == 1)                   //Indien al gestopt
{                                                           //Voorkomt dubbele
                                                            //blokkering
   pt(getname(current));
   pt(" is reeds geblokkeerd");
   pr();
}
else                                                        //Indien niet gestart
{                                                           //Laad de basis texture
   loadimage("Textures\\VRtransportband.bmp","VRtransportband");
                                                            //pas de texture toe
   setobjecttextureindex(current,gettextureindex("Textures\\VRtransportband.bmp"));

   stopobject(current, STATE_BLOCKED);                      //Stop de motor
   setlabelnum(current,"BLOKKADE",1);
}
```

- FORWARD

```
treenode current = ownerobject(c);                          //Wijst naar zichzelf
treenode conv = current;                                    //Hernoeming

if (getlabelnum(current,"BLOKKADE") == 0)                   //Vorige situatie geen stop
{
   setlabelnum(current,"RICHTING",1);                       //Set de variabele RICHTING
   bcsetdirection(conv,1);                                  //Zet de band in vooruit
                                                            //Laad de texture "vooruit"
                                                            //en pas toe
   loadimage("Textures\\VRtransportbandVooruit.bmp","VRtransportbandVooruit");
   setobjecttextureindex(current,gettextureindex("Textures\\VRtransportbandVooruit.bmp"));
   receiveitem(current,1);                                  //Ontvang een item

for(int i = 1; i <= content(conv); i++)                     //Stuur alle items die al
   {                                                        //op de band lagen de goede
   treenode item = rank(conv, i);                           //richting op
   bcsetitemconveystate(conv, item, bcgetitemposition(conv, item), 0, 1, 0);
   }
}
Else                                                        //Vorige situatie stop
{
   setlabelnum(current,"RICHTING",1);                       //Set de variabele RICHTING
                                                            //Laad de texture "vooruit"
                                                            //en pas toe
   loadimage("Textures\\VRtransportbandVooruit.bmp","VRtransportbandVooruit");
   setobjecttextureindex(current,gettextureindex("Textures\\VRtransportbandVooruit.bm p"));
   bcsetdirection(conv,1);                                  //Zet de band aan (vooruit)
   receiveitem(current,1);                                  //Ontvang een item
   resumeobject(current);                                   //Start het object
   setlabelnum(current,"BLOKKADE",0);
for(int i = 1; i <= content(conv); i++)                     //Stuur alle items die al
   {                                                        //op de band lagen de goede
   treenode item = rank(conv, i);                           //richting op
   bcsetitemconveystate(conv, item, bcgetitemposition(conv, item), 0, 1, 0);
   }
}
```

- BACKWARD

```
1.  treenode current = ownerobject(c);                          //Wijst naar zichzelf
2.  treenode conv = current;                                    //Hernoeming

3.  if (getlabelnum(current,"BLOKKADE") == 0)                   //Vorige situatie geen stop
4.  {
5.     setlabelnum(current,"RICHTING",0);                       //Set de variabele RICHTING
6.     bcsetdirection(conv,0);                                  //Zet de band in vooruit
                                                                //Laad de texture "vooruit"
                                                                //en pas toe
7.     loadimage("Textures\\VRtransportbandAchteruit.bmp","VRtransportbandAchteruit");
8.     setobjecttextureindex(current,gettextureindex("Textures\\VRtransportbandAchteruit.bmp")
       );
9.     receiveitem(current,2);                                  //Ontvang een item

10. for(int i = 1; i <= content(conv); i++)                     //Stuur alle items die al
11.    {                                                        //op de band lagen de goede
12.    treenode item = rank(conv, i);                           //richting op
13.    bcsetitemconveystate(conv, item, bcgetitemposition(conv, item), 0, 1, 0);
14.    }
15. }
16. else
17. {
18.    setlabelnum(current,"RICHTING",0);                       //Set de variabele RICHTING
                                                                //Laad de texture "vooruit"
                                                                //en pas toe
19.    loadimage("Textures\\VRtransportbandAchteruit.bmp","VRtransportbandAchteruit");
20.    setobjecttextureindex(current,gettextureindex("Textures\\VRtransportbandAchteruit.bmp")
    );
21.    bcsetdirection(conv,0);                                  //Zet de band in achteruit
22.    receiveitem(current,2);                                  //Ontvang een item
23.    resumeobject(current);                                   //Start het object
24.    setlabelnum(current,"BLOKKADE",0);

25. for(int i = 1; i <= content(conv); i++)                     //Stuur alle items die al
26.    {                                                        //op de band lagen de goede
27.    treenode item = rank(conv, i);                           //richting op
28.    bcsetitemconveystate(conv, item, bcgetitemposition(conv, item), 0, 1, 0);
29.    }
30. }
```

Zoals te zien is, zijn de nodefuncties van de Bidirectional Conveyor en Bidirectional Merge gelijk. Het enige verschil zit in de trigger OnEntry. Bij de Bidirectional Conveyor is de inputpoort waar een item de lopende band op mag komen afhankelijk van de draairichting van de motor, de output volgt vanzelf. Bij de Bidirectional Merge mogen items van alle inputpoorten tegelijkertijd binnenkomen, afhankelijk van de inputpoort komen de items aan het begin of halverwege de band op. Ook hier volgt de output vanzelf.

- OnEntry trigger

```
1.  treenode current = ownerobject(c);                    //Verwijzing naar de lopende band
2.  int port = parval(2);                                 //Verwijzing naar de poort
3.  treenode richting = getlabelnum(current,"RICHTING");  //Haal de looprichting van de band
4.  treenode conveyorlength = getvarnum(current,"conveyorlength"); //haal lengte band op
5.  treenode helft = conveyorlength/2;                    //Bereken de de helft van de lengte

6.  receiveitem(current);                                 //Ontvang een item van willekeurige
                                                          //poort
7.  if(port == 1)                                         //Wanneer binnenkomend item van
8.     {                                                  //poort 1 komt
9.     bcsetitemconveystate(current,item,0,0,             //Geef dat item snelheid 1
10.    getvarnum(current,"speed"),0);                     //en beginpositie 0 mee
11.    }
12. else if(port ==2)                                     //Wanneer binnenkomend item van
13.    {                                                  //poort 1 komt
14.    bcsetitemconveystate(current,item,helft,0,         //Geef dat item snelheid 1 en
15.    getvarnum(current,"speed"),0);                     //beginopsitie op de helft
16.    }
```

5.2. Gebruik

Ook hier geldt weer dat eerst de input- en outputpoorten voorwaarts aangesloten moeten worden, waarbij de eerste input op het begin van de band aankomt en de tweede op de helft. Om te zorgen dat de nodefunctie BACKWARD nuttig is, dient de eerste input een Siemuflex Bidirectional Conveyor te zijn of er dient direct een sink aan deze output gekoppeld te worden.

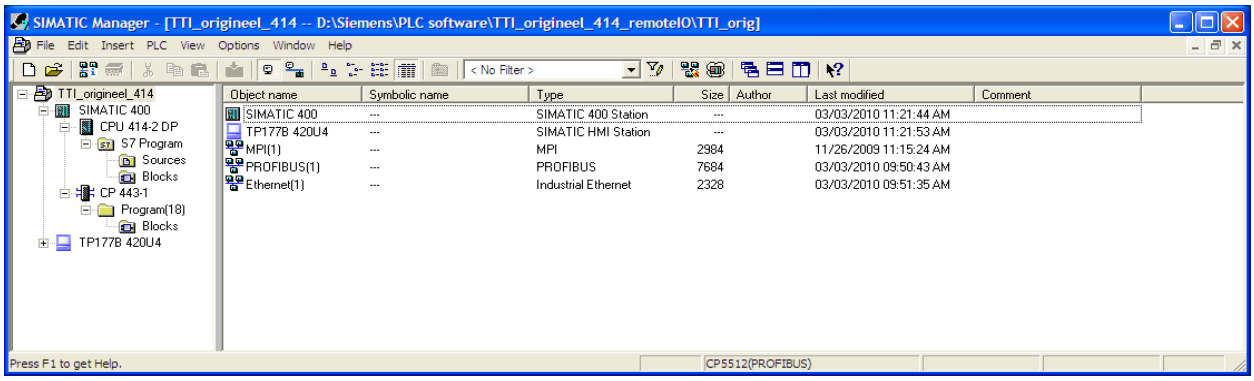
BIJLAGE H: Step 7

Inhoudsopgave

Lijst van figuren Bijlage H

1. SIMATIC manager

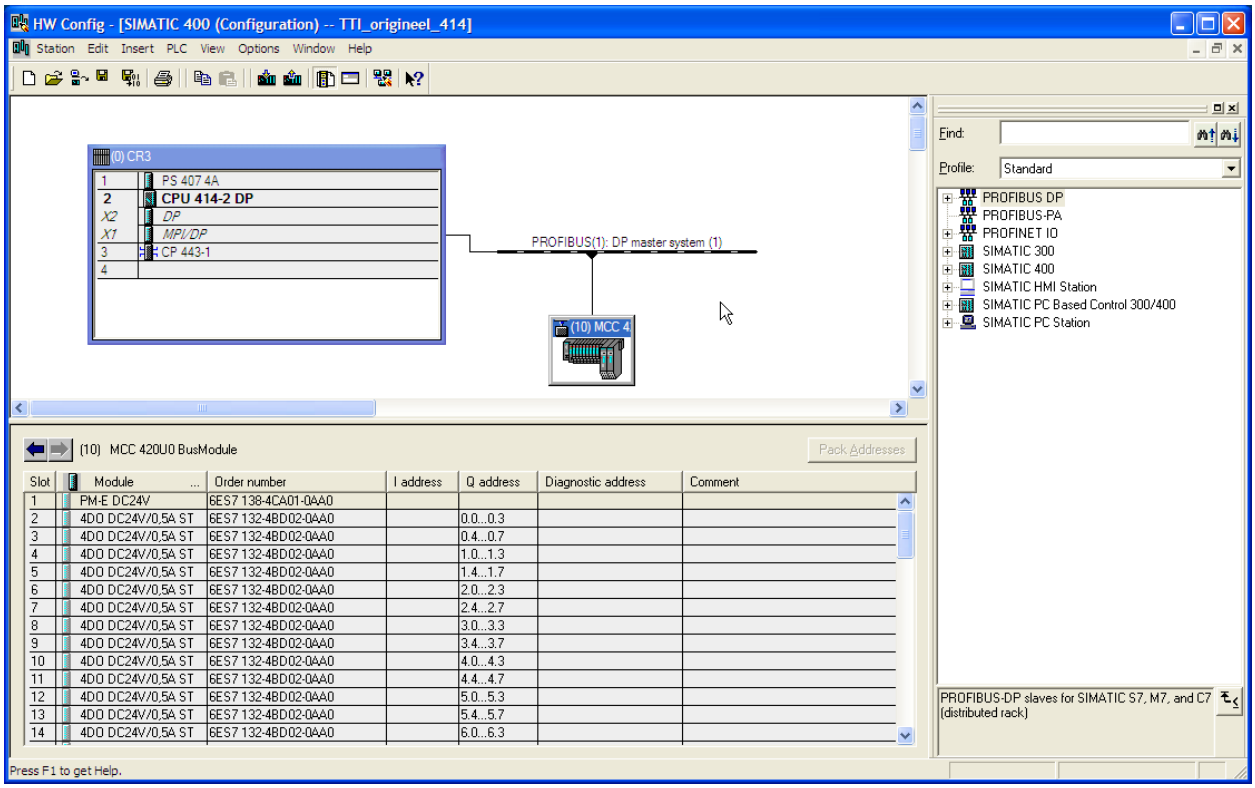
Het startpunt voor STEP 7 is het computerprogramma SIMATIC Manager. Vanuit hier kunnen de verschillende STEP 7 programma's bereikt worden.



Bijlage H - Figuur 1: Hoofdscherm van het computerprogramma Simatic Manager

2. HW Config

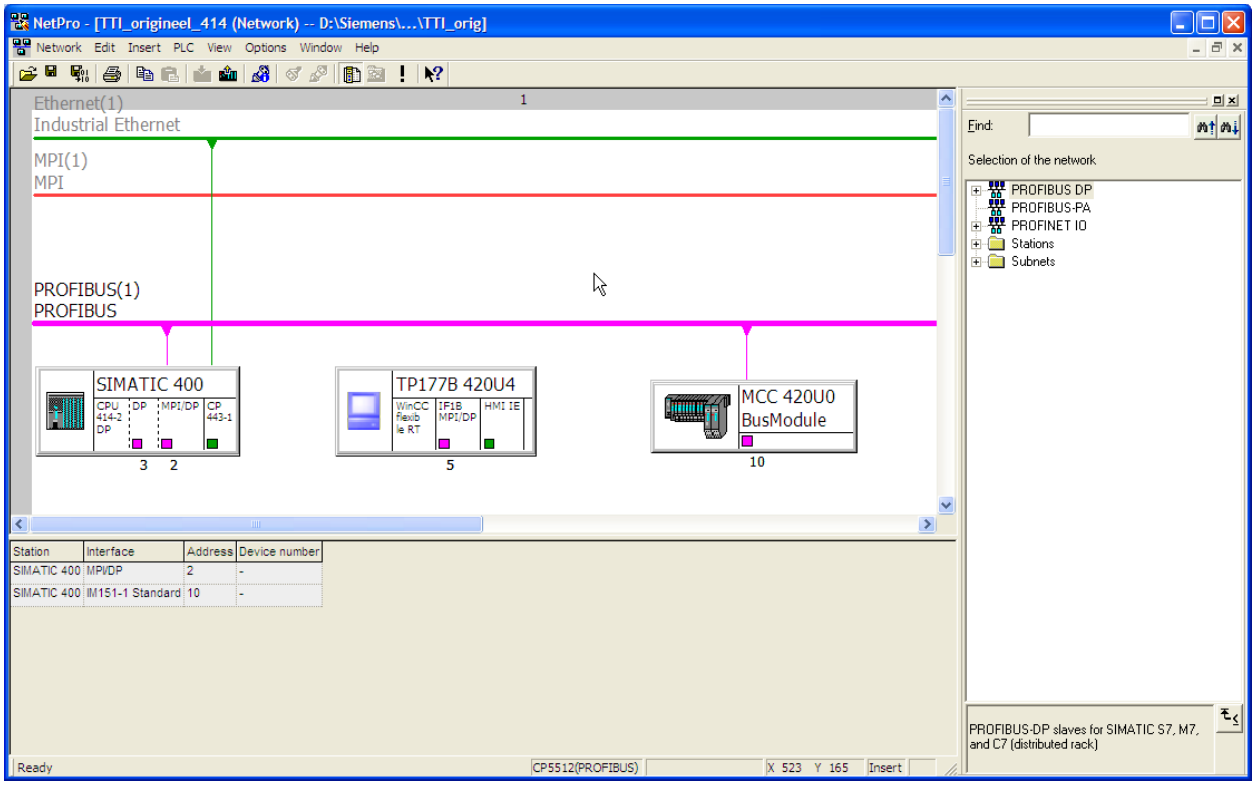
In het computerprogramma HW Config kunnen de verschillende hardware onderdelen van de PLC gekozen en geconfigureerd worden. Als eerste wordt het gebruikte rack geselecteerd. Hierna kunnen de modules, zoals de voeding, CPU en I/O toegevoegd worden. Bij het invoegen kunnen de configuratie instellingen aangepast worden.



Bijlage H - Figuur 2: Hardware configuratie in het computerprogramma HW Config

3. NetPro

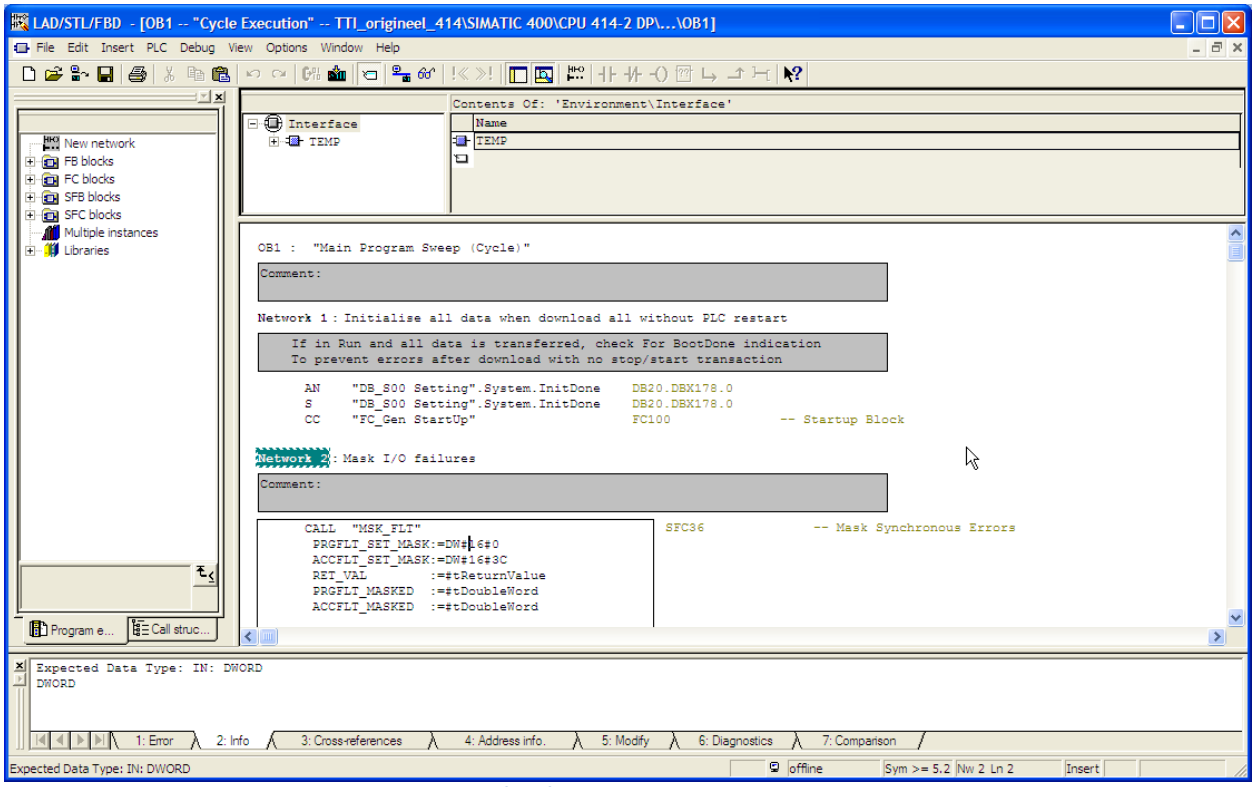
In het computerprogramma NetPro kan het netwerk geconfigureerd worden. De modules kunnen via diverse communicatieprotocollen verbonden worden. Voorbeelden zijn Profibus, Industrial Ethernet en MPI. MPI is het communicatieprotocol van Siemens waarmee een programmeerapparaat met een PLC gegevens kan uitwisselen.



Bijlage H - Figuur 3: Netwerk configuratie in het computerprogramma NetPro

4. LAD/STL/FBD

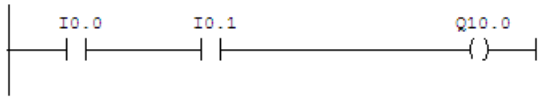
In het computerprogramma LAD/STL/FBD kan het PLC-programma geschreven worden. Dit kan in verschillende programmeertalen, te weten ladderdiagram, statement list, function block diagram en Structured Control Language. Deze staan beschreven het volgende hoofdstuk van deze bijlage.



Bijlage H - Figuur 4: Het computerprogramma LAD/STL/FBD met een deel van Organisatie Blok 1

5. PLC-programmeertalen

- Ladderdiagram (LAD) is een grafische voorstelling van de STEP 7 programmeertaal. De syntaxis voor de instructies is vergelijkbaar met een normaal stroomkringschema, maar dan een kwartslag linksom gedraaid.



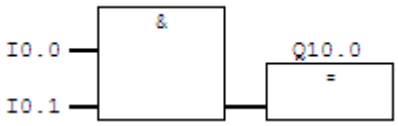
Bijlage H - Figuur 5: Voorbeeld serieschakeling in Ladderdiagram

- Statement list (STL) is een tekstuele weergave van de STEP 7 programmeertaal, vergelijkbaar met machine-code. Als een programma is geschreven in Statement List, komen individuele instructies overeen met de stappen waarmee de CPU het programma uitvoert. Om de programmering te vergemakkelijken, is Statement List uitgebreid met enkele hogere programmeertaal constructies (zoals gestructureerde toegang tot de gegevens en blokparameters).

```
A     I     0.0
A     I     0.1
=     Q    10.0
```

Bijlage H - Figuur 6: Voorbeeld serieschakeling in Statement List

- Function Block Diagram (FBD) is ook een grafische voorstelling van de STEP 7 programmeertaal en gebruikt de logica dozen bekend van Boole-algebra om de logica te vertegenwoordigen. Complexe functies (bijvoorbeeld wiskundige functies) kunnen rechtstreeks worden vertegenwoordigd in samenhang met de logica dozen.



Bijlage H - Figuur 7: Voorbeeld serieschakeling in Function Block Diagram

- Structured Control Language (SCL) is een tekstuele weergave van de STEP 7 programmeertaal. Deze programmeertaal lijkt qua syntaxis sterk op Pascal en C. Het verschil met Statement List is dat SCL een hogere programmeertaal is en daarom geschikt is voor de berekening van vergelijkingen, complexe algoritmen, of het beheer van grote hoeveelheden gegevens.

```
BEGIN
CONTROL:=FALSE;
FOR INDEX:= 1 TO ENDVALUE DO
  IQ1:= IQ1 * 2;
  IF IQ1 >10000 THEN
    CONTROL = TRUE
  END_IF
END_FOR;
END_FUNCTION_BLOCK
```

Bijlage H - Figuur 8: Voorbeeld PLC-programmacode in Structered Control Language

BIJLAGE I: Bestandsoverzicht Siemuflex

Op de volgende pagina's staat een overzicht van alle bestanden van Siemuflex. Het grootste gedeelte is in voorgaande hoofdstukken en bijlagen uiteengezet. Overige bestanden zijn automatisch gegenereerd, minder van belang voor het geheel of verkregen via derden.

Overzicht van alle Siemuflex bestanden:

```
Folder PATH listing
Volume serial number is 4CF0-C858
C:.
|   Install.bat
|   Siemuflex.lnk
|   Typelijst.txt.lnk
|
+---Flexsim
|   |   Siemuflex.fsm
|   |   Siemuflex.fsm!
|   |   SiemuflexLibrary.fsl
|   |   testing5.fsm
|   |   testing5.fsm!
|   |
|   +---DLL
|   |       FlexsimAXDLL.dll
|   |       FlexsimRPC.fsl
|   |       FlexsimRPC.fsl!
|   |       FlexsimRPC.fsm
|   |       FlexsimRPC.fsm!
|   |       FlexsimRPCServer.dll
|   |
|   \---Textures
|           newconveyorpictureFS.bmp
|           newconveyorpicturesmallFS.bmp
|           VRinvoegwielen.bmp
|           VRinvoegwielenAchteruit.bmp
|           VRinvoegwielenVooruit.bmp
|           VRrollenuitvoeger2.bmp
|           VRsorter.bmp
|           VRtransportband.bmp
|           VRtransportbandAchteruit.bmp
|           VRtransportbandinvoeg5.bmp
|           VRtransportbandVooruit.bmp
|           VRuitvoegPlateau.bmp
|           VRuitvoegwielen.bmp
|
\---Siemuflex
    |   Siemuflex.ncb
    |   Siemuflex.sln
    |
    +---Debug
    |       Inputs.exe
    |       Inputs.ilk
    |       Inputs.pdb
    |       libnodave.dll
    |       Outputs.exe
    |       Outputs.ilk
    |       Outputs.pdb
    |       SiemuflexDLL.dll
    |       SiemuflexDLL.exp
    |       SiemuflexDLL.ilk
    |       SiemuflexDLL.lib
    |       SiemuflexDLL.pdb
    |       Start.exe
    |       Start.ilk
    |       Start.pdb
    |
    +---Include
    |       array.h
    |       binding.h
    |       declaration.h
    |       definition.h
    |       FlexsimAXDLL.lib
    |       FlexsimDefs.h
    |       FlexsimFuncs.h
    |       InputMyDLL.h
    |       libnodave.lib
    |       nodave.h
    |       openSocket.h
    |       SharedMemoryLib.cpp
    |       SharedMemoryLib.h
    |       Shortcut to Typelijst.txt.lnk
    |       Typelijst.txt
    |       UserFuncs.h
    |
    +---Inputs
    |   |   Inputs.cpp
    |   |   Inputs.vcproj
    |   |   Inputs.vcproj.HQVANRIETNL.barhol.user
    |   |   Inputs.vcproj.HQVANRIETNL.cedmol.user
    |   |
    |   \---Debug
    |           BuildLog.htm
    |           Inputs.exe.embed.manifest
    |           Inputs.exe.embed.manifest.res
    |           Inputs.exe.intermediate.manifest
    |           Inputs.obj
    |           mt.dep
    |           SharedMemoryLib.obj
    |           vc90.idb
    |           vc90.pdb
    |
    +---Outputs
    |   |   DLLMain.cpp
    |   |   Outputs.cpp
    |   |   Outputs.vcproj
    |   |   Outputs.vcproj.HQVANRIETNL.barhol.user
    |   |   Outputs.vcproj.HQVANRIETNL.cedmol.user
    |   |
    |   \---Debug
    |           BuildLog.htm
```

```
|       DLLMain.obj
|       mt.dep
|       Outputs.exe.embed.manifest
|       Outputs.exe.embed.manifest.res
|       Outputs.exe.intermediate.manifest
|       Outputs.obj
|       OutputsNIEUWER.obj
|       SharedMemoryLib.obj
|       vc90.idb
|       vc90.pdb
|
+ SiemuflexDLL
|   |   DLLMain.cpp
|   |   InputMyDLL.cpp
|   |   mydll.cpp
|   |   SiemuflexDLL.vcproj
|   |   SiemuflexDLL.vcproj.HQVANRIETNL.barhol.user
|   |   SiemuflexDLL.vcproj.HQVANRIETNL.cedmol.user
|   |
|   \ Debug
|       BuildLog.htm
|       DLLMain.obj
|       InputMyDLL.obj
|       mt.dep
|       mydll.obj
|       SharedMemoryLib.obj
|       SiemuflexDLL.dll.embed.manifest
|       SiemuflexDLL.dll.embed.manifest.res
|       SiemuflexDLL.dll.intermediate.manifest
|       vc90.idb
|       vc90.pdb
|
\ Start
    |   Start.cpp
    |   Start.vcproj
    |   Start.vcproj.HQVANRIETNL.barhol.user
    |   Start.vcproj.HQVANRIETNL.cedmol.user
    |
    \ Debug
        BuildLog.htm
        mt.dep
        SharedMemoryLib.obj
        Start.exe.embed.manifest
        Start.exe.embed.manifest.res
        Start.exe.intermediate.manifest
        Start.obj
        vc90.idb
        vc90.pdb
```