Departement Industriële en Biowetenschappen

Master in de industriële wetenschappen:

elektronica-ICT - ICT

Event driven CAN & Video logger for unambiguous vehicle diagnostics

A tool for DAF trucks Eindhoven

CAMPUS

Geel

Sven Van Dyck

Academiejaar 2007-2008

VOORWOORD

Aan het begin van mijn stage op het CIT had ik de opdracht gekregen een competitie tussen scholen op te zetten met de Lego Mindstorms robot. De uitdaging bestond erin om de competitie tussen de robots op een centrale locatie te laten plaatsvinden en dat studenten van scholen over heel België hun robot via het internet konden programmeren.

Uiteraard moesten ze de wedstrijd via webcams kunnen volgen. Daardoor was ik reeds begonnen met het bestuderen van ASP.NET 2.0 en te onderzoeken hoe het geheel er zou moeten uitzien. Helaas kwam na 2/3 weken de spijtige melding dat dit project afgeblazen werd, dus kreeg ik na verloop van tijd (ongeveer in het begin van oktober) een nieuwe opdracht voor DAF Eindhoven. Deze opdracht is nu met succes afgerond, dankzij de steun van verschillende personen die ik wil bedanken.

Vooraleerst wil ik de heer Adriaan Brebels bedanken om mij de kans te geven bij het CIT stage te lopen.

In het bijzonder wil ik mijn externe begeleider, de heer Tom Limerkens, bedanken omdat hij mij zeer goed begeleid en ondersteund heeft. Zijn ondersteuning en ervaring waren onmisbare factoren in het succesvol afronden van dit project.

Omdat ik er steeds met al mijn vragen terecht kon, wil ik alle medewerkers van het CIT, medestudenten, docenten, vrienden en mijn stagebegeleider, de heer Rob Schoenmakers, bedanken voor hun hulp.

Mijn ouders wil ik bedanken voor hun jarenlange financiële en morele steun en het vertrouwen dat ze in mij stelden.

Van Dyck Sven, april 2008.

SAMENVATTING

De werkwijze die ze bij DAF Eindhoven hanteren om een nieuw systeem in ontwikkeling te testen is als volgt. Het te testen systeem wordt gemonteerd in een testvrachtwagen. Deze vrachtwagen wordt alleen gebruikt om deze nieuwe systemen te testen. Nadat het systeem ingebouwd is, gaat de testvrachtwagen er mee rondrijden. Tijdens deze proef wordt het systeem in werking getest. Zo een proef als deze kan bijvoorbeeld één week duren.

Vóór dit project schreef de testchauffeur al zijn bevindingen en rare gedragingen van het te testen systeem tijdens zijn testrit op. Na de testrit probeerde hij voor de testingenieurs de situaties, waarbij hij iets raar had opgemerkt, te reconstrueren. Dit leidde tot veel misverstanden, waardoor men bij DAF graag een systeem had dat alle CAN-data samen met de beelden van het verkeer voor de vrachtwagen logde. Bovendien mocht er alleen gelogd worden als de chauffeur op een hardware-knop duwde. De gelogde data moest bestaan uit een periode vóór dat er op de knop gedrukt was en een periode nadat er op de knop gedrukt was. Daardoor was het noodzakelijk om continu een buffer van alle data bij te houden. De ontwikkelde logger in dit project voldoet aan al deze specificaties.

Om al deze gelogde data te analyseren had men bij DAF ook een applicatie nodig die deze data overzichtelijk en gesynchroniseerd (video met CAN-data) weergaf. Deze applicatie heeft zijn concrete realisatie gekregen in de vorm van de “Viewer”.

Zowel de logger als de Viewer zijn ontwikkeld in LabVIEW.

INHOUDSTAFEL

LIJST MET ILLUSTRATIES

AFKORTINGEN

ABS	Antiblokkeersysteem
ACK	Ackownledgement
API	Application Programming Interface
BAM	Broadcast Announce Message
CAN	Controller Area Network
CANH	CAN high
CANL	CAN low
Cia	CAN in Automation
CIT	Centrum voor Industriële Toepassingen
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear To Send
DAF	van Doorne's Automobiel Fabriek
DAQ	Data Acquisition
DBC	DataBase CAN
DEL	Delete
DIO	Data In/Out
ECaVilo	Event driven CAN & Video logger for unambiguous vehicle diagnostics
ECU	Electronic Control Unit
EOF	End Of Frame
FIFO	first in, first out
Fps	Frames per second
Fpse	frames per second edit
GLI	Global variable with Local Intelligence
GUI	Graphical User Interface
HD	Hard Disk
ID	Identifier
IDE	Identifier Extension
INS	Insert
IPT	Information Processing Time
ISO	International Organization for Standardization
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
LED	Light Emitting Diode
LLC	Logical Link Control
LSB	Least Significant Bit
MAC	Medium Access Control
MCMP	Motion CMP (Compression), CMP = image compression technology
MDI	Medium Dependant Interface
MJPEG	Motion 'Joint Photographic Experts Group'
MPEG	Moving Pictures Experts Group
MSB	Most Significant Bit
NC	Normal Closed
NI	National Instruments
NI-IMAQ	National Instruments Image Aquistion
NO	Normal Open
NRZ	Non Return to Zero
NRZI	Non Return to Zero, Invert
OS	Operating System
OSI	Open Systems Interconnection
PC	Personal Computer
PDU	Protocol Data Unit
PF	PDU format

PG	Parameter Group
PGN	Parameter Group Number
PLS	Physical Signaling
PMA	Physical Medium Attachment
PS	PDU specific
REC	Received Error Counter
RTR	Remote Transmission Request
RTS	Request To Send
SAE	Society of Automotive Engineers
SJW	Synchronization Jump Width
SO-DIMM	Small Outline Dual In-line Memory Module
SOF	Start Of Frame
SRR	Substitute Remote Request
TCP/IP	Transmission Control Protocol (TCP) en het Internet Protocol (IP)
TEC	Transmitted Error Counter
TS	TimeStamp
USB	Universal Serial Bus
VAN	Vehicle Area Network
VF	VideoFile
Vi	Virtual Instrument

INLEIDING

De titel van dit werkstuk luidt:

"Event driven CAN & Video logger for unambiguous vehicle diagnostics".

"Event driven" betekent dat de logger alleen maar gaat loggen tijdens een bepaald event. Dit event of deze trigger wordt gegeven door een hardware-knop.

"CAN & Video logger" betekent dat er zowel Video- als CAN-data gelogd worden.

"for unambiguous vehicle diagnostics" betekent dat er geen twijfel kan bestaan over de gelogde data. De Viewer geeft de gelogde data immers gesynchroniseerd weer, daardoor kan de testingenieur een juiste diagnose stellen van het systeem dat getest wordt.

Voor de gebruiksvriendelijkheid kan de afkorting ECaViLo gebruikt worden, als men naar dit project verwijst.

De eerste twee hoofdstukken zijn theoretische hoofdstukken die handelen over de CAN-bus en USB. Beide zijn technologieën die in dit project gebruikt worden. Het derde hoofdstuk beschrijft het project zonder in detail te gaan. Voor lezers die alleen geïnteresseerd zijn in dit project en niet in de theorie van de gebruikte technologieën is het aan te raden bij dit hoofdstuk te beginnen. Het vierde hoofdstuk introduceert een paar LabVIEW begrippen die in hoofdstuk 6 en 7 gebruikt worden. Hoofdstuk 5 beschrijft alle gebruikte hardware waaruit het systeem van de logger bestaat. Hoofdstuk 6 en 7 beschrijven tenslotte de software van respectievelijk de logger en de Viewer. Hier wordt er dieper op de gebruikte principes ingegaan.

Tijdens de ontwikkeling van de logger en de Viewer zijn we 4 keer op verplaatsing geweest naar DAF Eindhoven. Daar konden we testen uitvoeren op de vrachtwagensimulator die men bij DAF ter beschikking stelde. Op het CIT konden we immers alleen maar de videologging testen en niet de CAN-logging. Door deze bezoeken werd DAF op de hoogte gehouden van de ontwikkeling. Tijdens ons laatste bezoek hebben we het volledige project getest en de laatste (kleine) bugs verholpen. We hebben tijdens dat bezoek het project opgeleverd zodat men het bij DAF kon inbouwen in hun testvrachtwagen.

Omdat ze bij het CIT reeds een schijnbare identieke videologger ter beschikking hadden, is het nuttig om de verschillen met dit project even aan te halen.

- Het loggen in dit project gebeurt pas als er een trigger (hardware-knop) gegeven wordt en de gegevens die gelogd worden komen uit een buffer. Het principe van het loggen is dus fundamenteel anders.

- Heel het gebeuren rond CAN, zowel bij de logger als bij de Viewer was niet aanwezig op de reeds beschikbare videologger/editor. Rond dit gedeelte is een groot stuk van de software van dit project opgebouwd.

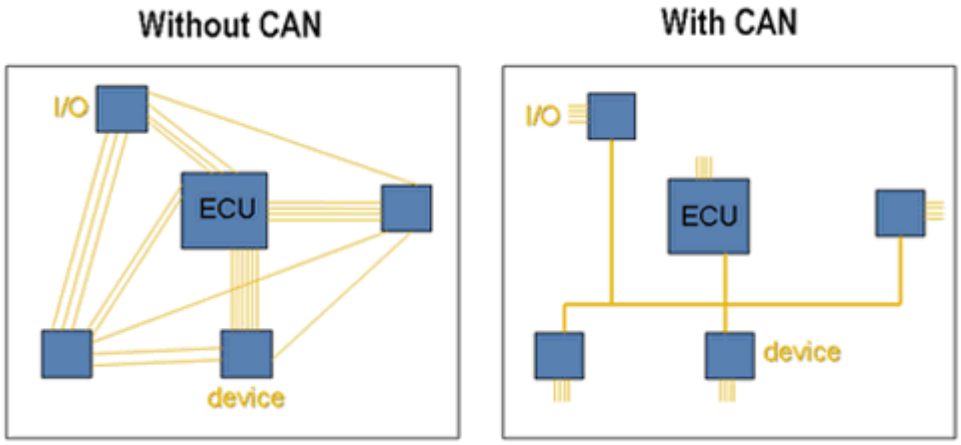
- De gebruikte hardware is verschillend, voor het videogedeelte bijvoorbeeld was in dit project een gewone webcam geschikt.

Deze verschillen zorgden ervoor dat er niet kon voortgebouwd worden op de al aanwezige videologger en -editor.

1 CAN - Bus

1.1 Inleiding tot de CAN-bus

Het Controller Area Network is in de jaren ’80 ontwikkeld door Bosch. De drijfveer van Bosch was om een robuust en betrouwbaar netwerk te ontwikkelen dat kon toegepast worden in voertuigen. Begin jaren ’90 nam de populariteit van CAN enorm toe. Dit was te wijten aan het feit dat de bedrading in voertuigen enorm complex was geworden. Er werd namelijk meer gebruik gemaakt van elektronica. Deze diende tot de verbetering van de veiligheid zoals het ABS-systeem. Anderzijds wou de klant steeds meer luxe, wat zich uitte in systemen zoals bijvoorbeeld cruisecontrol en airbags. Al deze extra elektronische componenten moesten gestuurd worden met als gevolg dat de kabelboom in een voertuig alsmaar langer en dikker werd. Dit was omdat er voor elke toegevoegde component een 1 op 1 verbinding nodig is met zijn ECU (Electronic Control Unit). Figuur 1 toont dit. De prijs en het gewicht van de kabelboom begon te stijgen. Een ECU bepaalt bijvoorbeeld wanneer en hoeveel brandstof er in de cilinders gespoten wordt bij elke motorcyclus.

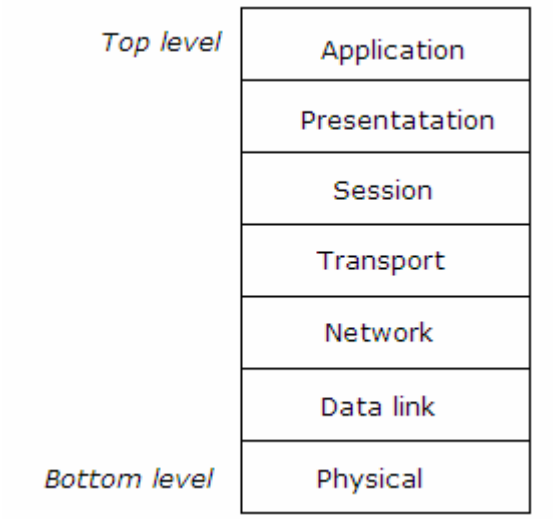


Figuur 1: Evolutie bedrading

Er werd dus naar een oplossing gezocht en die vond men in de CAN-bus. Op Figuur 1 merkt men meteen de verbetering in de bedrading. Andere voordelen van CAN zijn de real-time mogelijkheden en het goed functioneren in ruwe omgevingen. Er waren in de jaren ’90 ook nog andere soortgelijke protocollen voorhanden zoals bijvoorbeeld het VAN-systeem, maar die werden voor het grootste deel opzijgeschoven ten voordele van CAN.

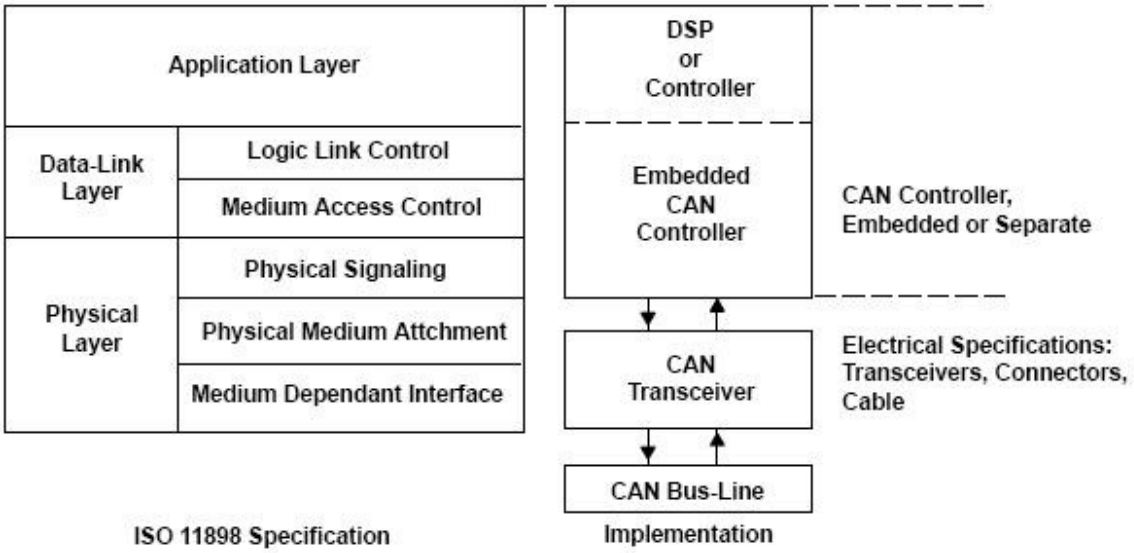
1.2 Het OSI-model

Het CAN-protocol is gebaseerd op het OSI 7-lagenmodel. Dit referentiemodel is ontwikkeld door de ISO en wordt ondermeer gebruikt bij Ethernet (Figuur 2).



Figuur 2: OSI 7-lagenmodel

Voor de beschrijving van het CAN-protocol zijn vooral de Fysische, Data Link en de Applicatie laag belangrijk, zoals weergegeven in figuur 3



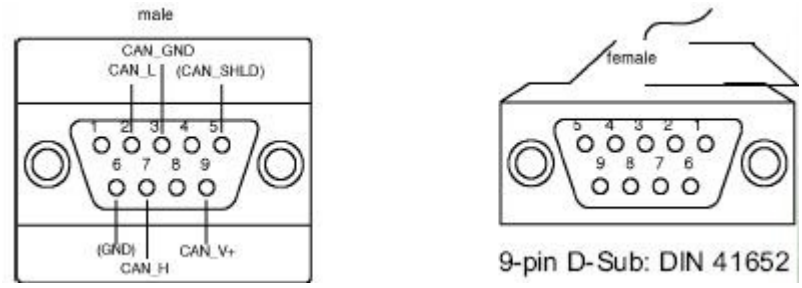
Figuur 3: Implementatie van het OSI 7-lagenmodel bij CAN

1.3 Fysische laag

1.3.1 De MDI sublaag

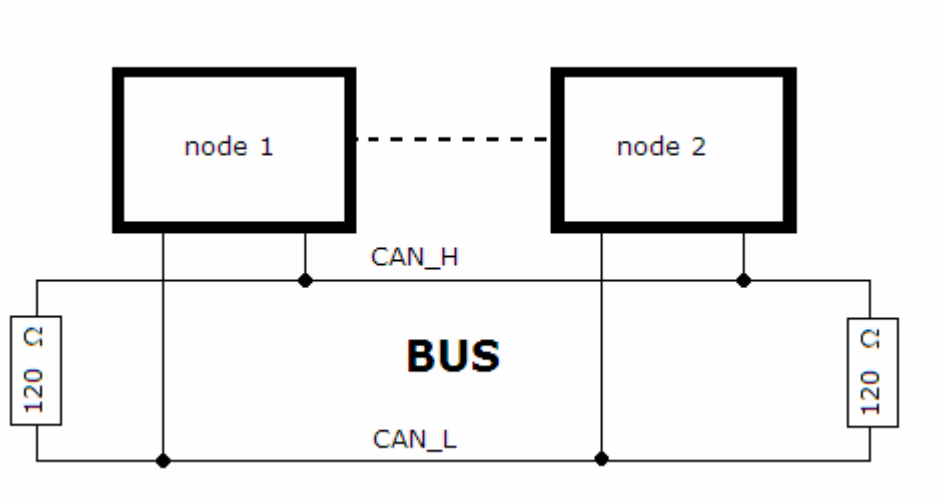
In Figuur 3 ziet men dat de fysische laag uit 3 componenten bestaat.

De onderste laag is de "Medium Dependant Interface" (MDI) laag, deze specificeert de karakteristieken van de kabel en de connector. De connector is een 9-pin D-sub connector (DB-9). In figuur 4 is de betekenis van de verschillende pinnen weergegeven.



Figuur 4: 9-pins D-sub connector

Het CAN-systeem gebruikt een busarchitectuur (figuur 5). Omwille van de benodigde robuustheid en met het oog op het verminderen van de kabelboom, is dit de beste keuze. Bij een sterarchitectuur bijvoorbeeld, moet al het verkeer langs een centrale node passeren, wat het aantal kabels niet verminderd. Een andere optie was de keuze voor een ringarchitectuur, maar deze is niet robuust. Als er 1 node uitvalt dan stopt het systeem met functioneren. Daarom was een bus de meest logische keuze. Een ander voordeel van een busarchitectuur is dat men nodes kan bijpluggen, zonder het netwerk te hinderen (hot-pluggable). De uiteinden van de bus dienen afgesloten te worden met de karakteristieke weerstand. Bij CAN bedraagt die ongeveer 120 Ω. Deze weerstand zorgt ervoor dat er geen reflecties van signalen optreden.



Figuur 5: CAN-busarchitectuur

Er wordt een twisted pair kabel gebruikt, deze kan shielded zijn. Het gebruik van differentiële spanningen en een twisted-pair zorgt ervoor dat het systeem minder storingsgevoelig is. Stel dat er een uitwendige storing plaatsvindt waardoor de spanning in CANL en CANH bijvoorbeeld +0,1V naar omhoog gaat. Dan zal het verschil tussen CANH en CANL geen verkeerd resultaat opleveren. 3.6V-1.6V is immers gelijk aan 3.5V-1.5V.

1.3.2 De PMA sublaag

De volgende laag is de PMA sublaag (figuur 3), dewelke de karakteristieken voor de zender/ontvanger vastlegt. Het is in de specificaties niet bepaald welk medium er voor CAN zou moeten gebruikt worden. Er kan gebruik gemaakt worden van "single-wire", draadloos of glasvezel bekabeling. Meestal worden differentiële lijnen gebruikt. De volgende standaarden gebruiken differentiële lijnen:

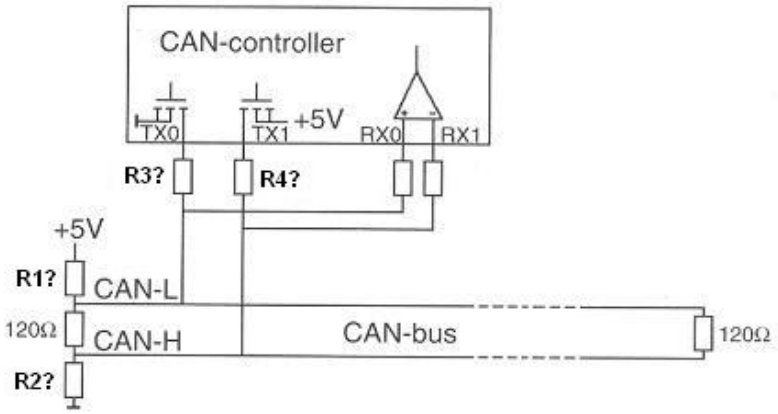
- IS011519: Low-Speed-CAN. Hier zijn snelheden van 5 kbit/s tot 125 kbit/s mogelijk.
- IS011898: High-Speed-CAN. Hierbij worden snelheden tot 1 Mbit/s ondersteund.
- SAE J1939-11: snelheid is 250kbit/s en maakt gebruik van een "shielded twisted pair" of STP kabel.
- SAE J1939-11: snelheid is 250kbit/s en maakt gebruik van een "unshielded twisted pair" of UTP kabel.

Voor de overdracht van een logische 0 of 1, respectievelijk een dominante of recessieve bit, wordt er gebruik gemaakt van differentiële spanningen. De benamingen dominant en recessief zullen hun nut nog bewijzen bij onderdeel 1.4.1.5 (Arbitrage). De huidige bit is recessief als het spanningsverschil tussen CANH en CANL (zie figuur 6) hoogstens 0,5V bedraagt. Anderzijds bestaat er een dominante toestand als het spanningsverschil tussen CANH en CANL minstens 0,9 V is. Bij een recessieve toestand is de CAN-bus in rust.

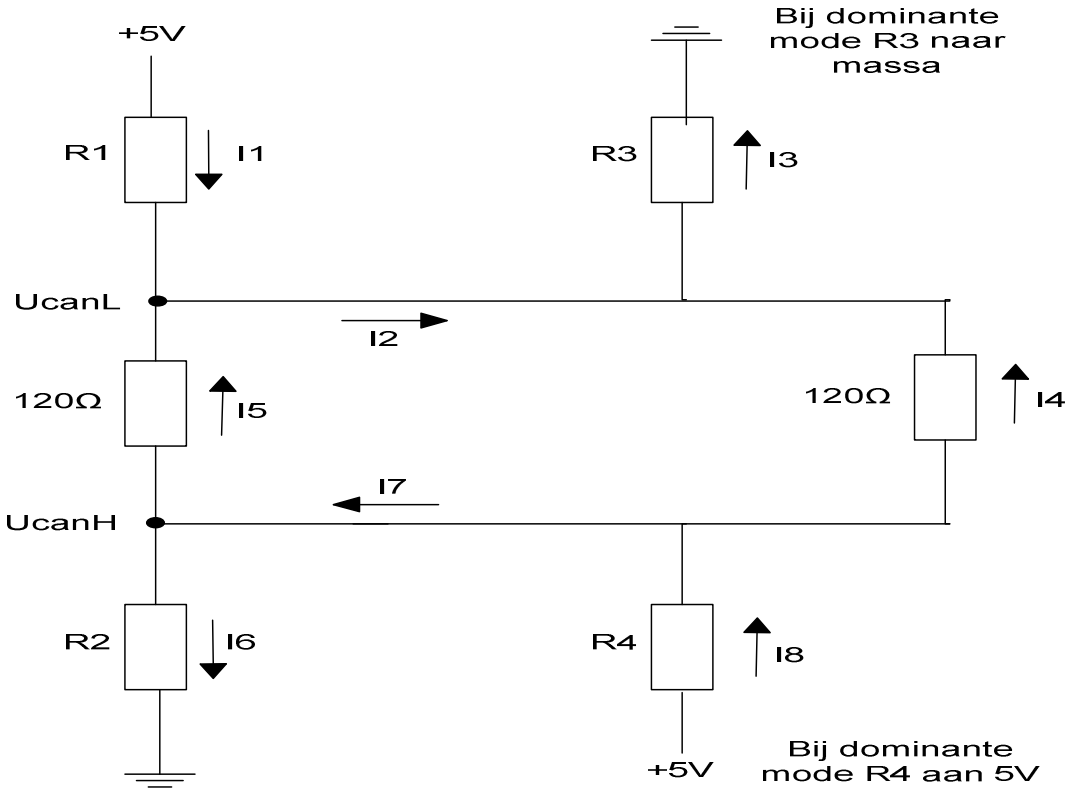
spanning op...\\bustoestand	recessief	dominant
CANH	2,5 V	3,5 V
CANL	2,5 V	1,5 v
toelaatbaar spanningsverschil: Ud = CANH - CANL	0... 0,5V	0,9...2,0V

Figuur 6: De spanningen ten opzichte van de massa

Hoe bekomt men nu deze spanningen? Een eenvoudige manier om een CAN-bus elektrisch te realiseren, is een aantal weerstanden aan de CAN-controllers hangen. Bij de meer complexere CAN-interfaces vindt men ook nog actieve componenten, zoals stubs, terug die kunnen zorgen voor een langere buslengte of een hogere bussnelheid. Een stub wordt ergens op een transmissielijn (in dit geval de CAN-bus) geplaatst om daar de reactieve component van die transmissielijn te neutraliseren; men kan deze stub parallel of in serie met de transmissielijn plaatsen. Hieronder volgt een korte berekening van de eenvoudige methode om de juiste spanningen te bekomen.



Figuur 7: CAN-interface



Figuur 8: Vereenvoudigde tekening CAN-interface

Voor het verzenden van een **recessieve** bit worden TX0 en TX1 niet gekoppeld (Figuur 7). De weerstanden die de bus afsluiten worden 120Ω genomen, vermits dit de karakteristieke weerstand van de bus is. Wil men de CANL en de CANH spanningen bij recessieve toestand respectievelijk bijvoorbeeld 2,7V en 2,3V, dan berekent men R1 en R2 als volgt:

$$I = \frac{U}{R} = \frac{2,7 - 2,3}{\frac{1}{1/120 + 1/120}} = \frac{0.4}{60} = 1/150A$$

$$R1 = \frac{U}{I} = \frac{5-2,7}{1/150} = 345\Omega$$
$$R2 = \frac{U}{I} = \frac{2,3-0}{1/150} = 345\Omega$$

Vervolgens worden de weerstanden R3 en R4 berekend die in de **dominante** mode actief zijn. Voor het verzenden van een dominante bit wordt TX0 gekoppeld aan de massa en TX1 aan de 5V. De CANL en CANH spanningen worden respectievelijk 1,5V en 3,5V gekozen. Deze waarden werden in de specificaties gegeven (Figuur 6).

R3 wordt eerst berekend:

De spanning over de beide parallelle weerstanden van 120Ω is dezelfde, dus ook de stroom door de beide weerstanden is dezelfde:

$$I4 = I5 = \frac{3,5-1,5}{120} = 0.0167A$$

De stroom door R1 kan men ook berekenen:

$$I1 = \frac{5-1,5}{345} = 0.01A$$

Nu kan men I2 vinden:

$$I2 = I1 + I5 = 0.01 + 0.0167 = 0.0267A$$

En tenslotte I3:

$$I3 = I2 + I5 = 0.0268 + 0.0167 = 0.0435A$$

R3 is dus:

$$R3 = \frac{U3}{I3} = \frac{1,5}{0,0435} = 34,5\Omega$$

Vervolgens wordt R4 berekend:

$$I6 = \frac{3,5V}{345\Omega} = 0.01A$$

$$I7 = I6 + I5 = 0,01 + 0,0167 = 0,0267A$$

$$I8 = I4 + I7 = 0{,}0167 + 0{,}0267 = 0{,}0434A$$

$$R4 = \frac{U}{I8} = \frac{5 - 3{,}5}{0{,}0434} = 34{,}5\Omega$$

Aan de uitgang van de CAN **ontvanger** bevindt zich een subtractor. Deze component berekent het verschil tussen CANH en CANL en geeft dit resultaat door aan de CAN controller die op de data-link laag opereert (Figuur 3 en Figuur 7).

1.3.3 De PLS sublaag

De laatste sublaag in de fysische laag is de PLS (zie figuur 3)

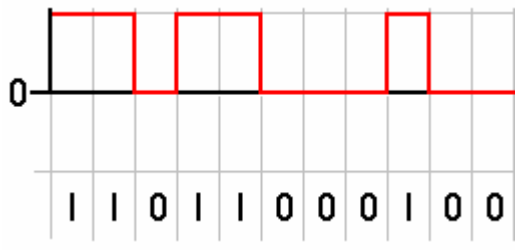
1.3.3.1 Bit Encoding/Decoding

De bit-code die bij de CAN-bus gebruikt wordt is een volledig-binaire unipolaire non return to zero basisbandcode (figuur 8).

Volledig-binair wil zeggen dat de 2 logische toestanden, 0 en 1, worden overgedragen door 2 verschillende niveaus. In dit geval levert de subtractor de 2 verschillende niveaus af. (zie figuur 6). Dus niet zoals bijvoorbeeld bij de half-binaire vorm, waar men een 1 voorstelt door een overgang van niveau en waar men een 0 voorstelt door het niet wijzigen van het niveau.

Unipolair betekent dat de impulsen maar uit 1 polariteit bestaan, in dit geval een positieve polariteit.

Non return to zero wil zeggen dat de overgang van een logische 0 naar een logische 1 of omgekeerd overeenkomt met een niveauverandering. Het nadeel van deze methode is dat de synchronisatie tussen de verschillende nodes op de CAN-bus verkeerd kan lopen. De nodes hersynchroniseren zich immers via de flanken. Als ze dus een tijdje geen flank meer gezien hebben, wanneer er bijvoorbeeld een groot aantal "1" -en na elkaar worden verstuurd, kan het zijn dat door de onnauwkeurigheid van het oscillator kristal in elke node, een verkeerde synchronisatie ontstaat. Dit probleem lost men op door middel van de techniek genaamd "bitstuffing" (figuur 9).



Figuur 9: volledig-binaire unipolaire NRZ-codering

"Bitstuffing" werkt als volgt: als er 5 identieke bits na elkaar op de bus verschijnen dan wordt de 6de bit tegengesteld aan de 5 vorige. In figuur 10 staat de toegevoegde bit tussen underscores.

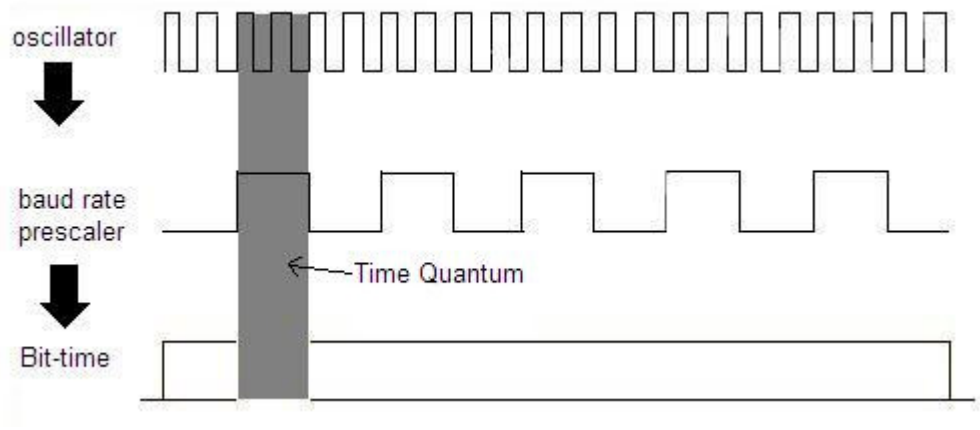
1111111000000
wordt na bitstuffing:
11111_0_1100000_1_0

Figuur 10: Bitstuffing

Bitstuffing wordt niet toegepast op de CRC delimiter en de ACK, EOF velden van het data of remote frame. Het wordt ook niet toegepast op het error en overload frame. De betekenis van de velden en de frames wordt later besproken.

1.3.3.2 De bittijd

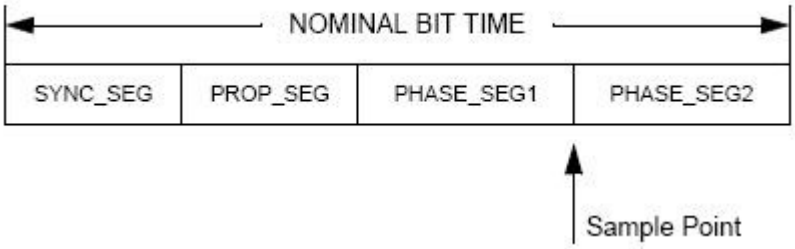
Met "Bit Timing" of bittijd bedoelt men de tijd die doorlopen wordt van het begin tot het einde van een bit. De kleinste eenheid van tijd in een CAN-systeem is een "Time Quantum" of tijdskwantum (figuur 11). Dit tijdskwantum bestaat uit een instelbaar aantal clockcycli van de oscillator die in elke CAN-controller aanwezig is. De lengte van een bittijd moet minimaal 8 en maximaal 25 tijdskwantum lang zijn. Men kan de lengte van de bittijd per CAN-controller instellen door enerzijds de "baud rate prescaler" te veranderen (en daarmee ook de lengte van het tijdskwantum), anderzijds kan men het aantal tijdskwantums per bittijd aanpassen. Alle CAN-controllers op de bus moeten natuurlijk werken met eenzelfde instelling.



Figuur 11: Visualisatie time quantum

De bitsnelheid bepaalt men met de volgende formule:

Nominale bittijd = 1 / nominale bitsnelheid



Figuur 12: Bittijd in detail

In figuur 12 kan men de verschillende onderdelen van een bittijd in detail zien. Het eerste segment noemt men het SYNC_SEG en is altijd 1 tijdskwantum lang. Zoals de naam al doet vermoeden wordt dit segment gebruikt om te synchroniseren. Elke controller verwacht tijdens dit segment de start van een bit. Als de vorige bit van een tegengestelde logische staat was, ziet de controller hier een flank. Als er een flank buiten het SYNC_SEG optreedt, noemt men het verschil tussen die flank en het SYNC_SEG de fasefout. Die kan zowel negatief als positief zijn(zie synchronisatie). Het volgende segment is het PROP_SEG dat men kan laten variëren van 1 tot 8 tijdskwantums. Dit segment gebruikt men om de vertraging op de bus te compenseren. De vertragingen ontstaan door de som van de interne vertraging in de CAN-controllers en door de vertraging van de bus. PHASE_SEG1 en PHASE_SEG2 worden gebruikt bij het hersynchroniseren (zie 1.3.3.3). Het punt waarop tijdens elke bittijd gesampled wordt, ligt tussen deze 2 laatstgenoemde segmenten. De "Information Processing Time" of IPT is de tijdsduur die de controller nodig heeft om zijn gesamplede punt te verwerken en is meestal kleiner dan of gelijk aan 2 tijdskwantums.

Voor men kan overgaan naar een voorbeeld is het nuttig om eerst de status van de bus toe te lichten als er 2 of meer nodes tegelijk willen verzenden.

Bus state with two nodes transmitting

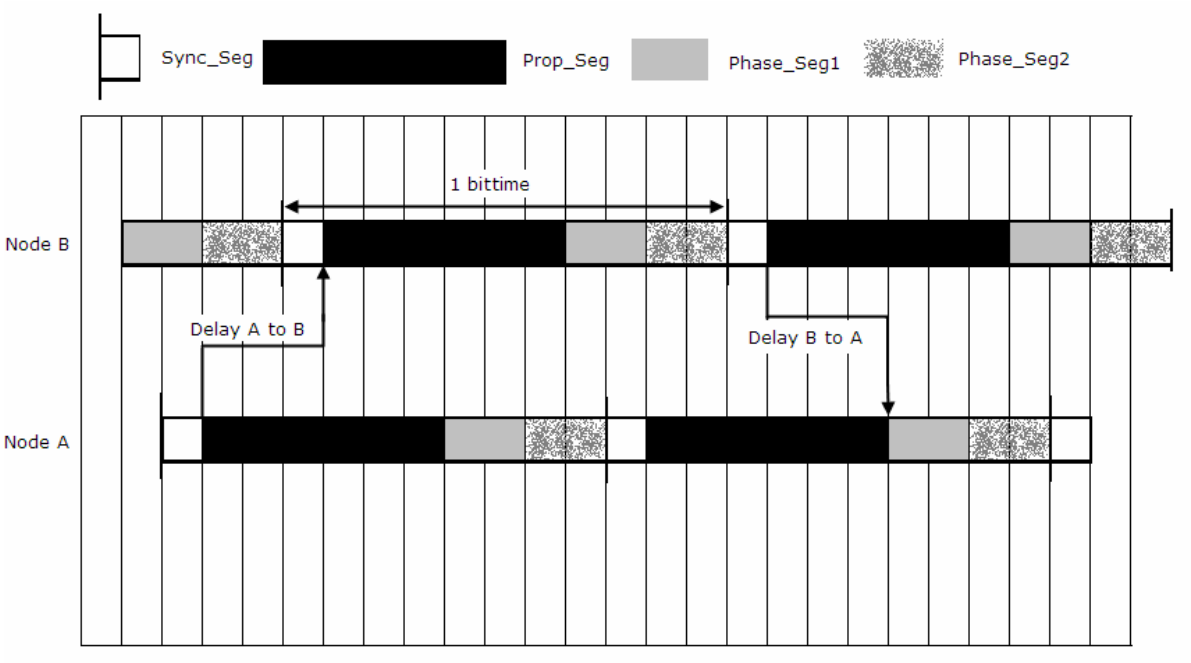
	dominant	**recessive**
dominant	dominant	dominant
recessive	dominant	recessive

Logical AND

	0	**1**
0	0	0
1	0	1

Figuur 13: Bus states

Men ziet dat de dominant bit altijd overheerst. Dit is logisch te verklaren omdat bij een recessieve bit de bus in rust wordt gelaten en bij een dominante bit gaat men een spanning op de bus zetten.



Figuur 14: De functie van PROP_SEG

De functie van het PROP_SEG zal verduidelijkt worden aan de hand van het voorbeeld in figuur 14:

In dit voorbeeld heeft men een fragment gekozen waar 2 nodes hun identifier (zie 1.4.1.5) tegelijkertijd willen verzenden. Het fragment in figuur 14 handelt over de eerste 2 verzonden bits. In dit voorbeeld is zowel de eerste als de tweede bit die node A verzendt recessief. Node B verzendt eerst een recessieve bit en daarna een dominante.

Node B en node A zijn gesynchroniseerd op de bitstroom van de CAN-bus, wat met zich meebrengt dat node B, die wat verder op de CAN-bus is geplaatst dan node A, niet in fase is met A. Dit komt, zoals eerder vermeld, door de vertraging van de bus en de vertraging die aanwezig is bij elke node. Dit wordt gecompenseerd door het PROP_SEG.

Als node A begint met het verzenden van zijn eerste bit, een recessieve, dan gaat het tijdens diezelfde bittijd de CAN-bus samplen. Als node A bij dit samplen detecteert dat de bit die hij verzonden heeft overeenkomt met de bit die hij samplet, dan begint hij met het versturen van zijn tweede bit, die ook recessief is. De samplewaarde tijdens de eerste bittijd van node A is in dit voorbeeld recessief (recessief van A + recessief van B = recessief op de bus).

Node B begint immers met het zenden van zijn recessieve bit tijdens de bittijd van node A. Ook node B samplet de bus en hij ziet een recessieve bit (recessief van A + recessief van B = recessief op de bus). Daarom hij gaat verder met het verzenden van zijn tweede bit, die dominant is.

Ondertussen heeft node A zijn tweede recessieve bit op de bus gezet en net voor hij gaat samplen in zijn tweede bittijd ontvangt hij de dominante bit van node B. Bijgevolg komt de gesamplede waarde niet meer overeen met de recessieve bit die hij net verstuurd heeft. Omdat node A een dominante bit samplet terwijl hij een recessieve bit op de bus gezet had, stopt hij met zenden. Let wel op dat voor een goede werking deze dominante bit van node B voor PHASE_SEG1 van node A moet aankomen. (De lengte van PHASE_SEG1 kan immers aangepast worden tijdens een hersynchronisatie). Dit gegeven bepaalt de lengte van PROP_SEG.

Vervolgens samplet node B zijn dominante bit en gaat door met verzenden. Als node B klaar is met verzenden van zijn frame, dan gaat node A nog eens proberen om zijn frame te versturen. Er is dus geen informatie verloren. Daarom is de CAN-bus zeer geschikt voor real-time toepassingen.

Het zojuist beschreven proces is een "non-destructive arbitration" proces. Later wordt er nog uitgelegd hoe dit werkt op de data link laag.

Het PROP_SEG wordt dus best niet te groot gekozen, want dit beïnvloedt de snelheid van de bus in de negatieve zin. Men mag het PROP_SEG ook niet te klein kiezen want dan moet men ook de maximale lengte van de bus verlagen. Hoe langer de bus, hoe groter de vertraging, hoe groter het PROP_SEG moet gekozen worden. De formule voor de lengte van PROP_SEG is als volgt:

$t_{propagation} = 2\ (t_{cable} + t_{canmodules})$

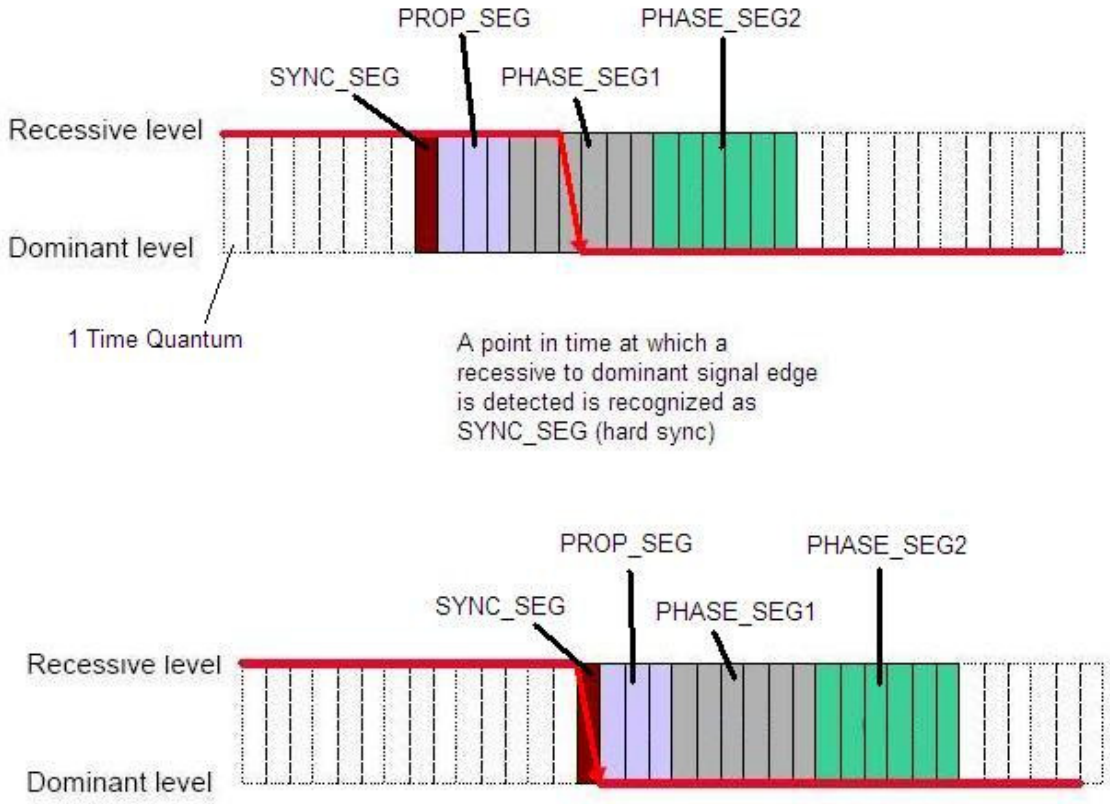
De som van de vertraging van de CAN-modules (+/- 400ns) en de vertraging op de kabel (+/- 5ns/m), moet verdubbeld worden. Neemt men het voorbeeld dat node A en node B aan de uiteinden van de bus liggen. Als node A een bit verzendt, ontvangt node B die met een vertraging die de som is van de vertragingen in de kabel, in node A en in node B. Als node A wil communiceren met node B dient hij te wachten op de ACK van node B die eenzelfde vertraging ondergaat. Of als node B ook iets tegelijkertijd wil gaan verzenden dan dient node A te wachten op de identifier bit van node B ("non-destructive arbitrage"), wat ook de reeds vermelde vertraging heeft. M.a.w. wanneer node A iets verzendt en hij moet wachten op een reactie van node B, dan dient de propagatie tijd dus zo groot te zijn dat hij het verkeer "heen en terug" omvat.

Bitsnelheid	Buslengte (m)	Nominale bittijd (microseconden)
1 Mbit/s	30	1
800 kbit/s	50	1,25
500 kbit/s	100	2
250 kbit/s	250	4
125 kbit/s	500	8
62,5 kbit/s	1000	20
20 kbit/s	2500	50
10 kbit/s	5000	100

Figuur 15: Praktische bittijden

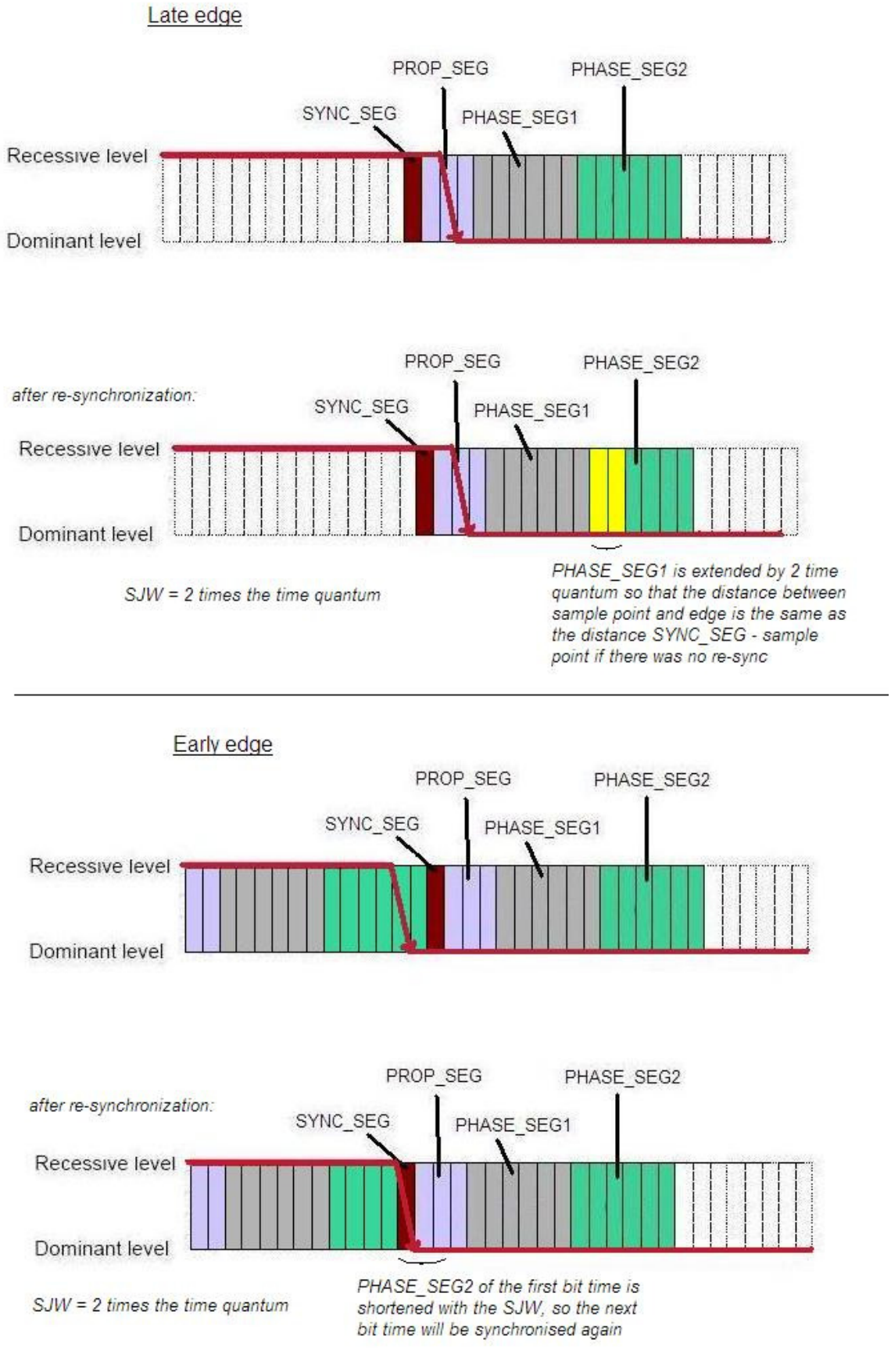
1.3.3.3 (her)synchronisatie

Er worden 2 soorten synchronisatie toegepast, harde synchronisatie en zachte synchronisatie of hersynchronisatie. Ze worden alle twee toegepast op een flank die ontstaat tijdens een recessieve naar dominante mode. Een flank is gesynchroniseerd indien hij binnen het SYNC_SEG valt. Wanneer de flank voor het SYNC_SEG optreedt, heeft men te maken met een negatieve fasefout. Als de flank na het SYNC_SEG optreedt, heeft men te maken met een positieve fasefout. De absolute waarde van de fasefout wordt berekend door het verschil te nemen van de flank en het einde van het SYNC_SEG. Synchronisatie is noodzakelijk omdat er kleine verschillen bestaan in de oscillators van de verschillende CAN-controllers.



Figuur 16: Harde synchronisatie

Als een CAN-controller een harde synchronisatie (figuur 16) uitvoert, herstart hij zijn interne bittijd aan het einde van SYNC_SEG. Dit zorgt ervoor dat de flank (recessief-dominant) die deze synchronisatie triggerde, binnen het SYNC_SEG van zijn nieuwe bittijd ligt. Harde synchronisatie wordt toegepast tijdens het SOF bit (altijd dominant) van een frame (zie 1.4.1.1) en wanneer er een flank (recessief-dominant) optreedt tussen 2 frames. Harde synchronisatie kijkt niet naar een eventuele fasefout.



Figuur 17: Hersynchronisatie

Hersynchronisatie (zie figuur 17) wordt toegepast tijdens de rest van het frame, altijd als er een flank (recessief-dominant) optreedt en wanneer de CAN-module als ontvanger functioneert. De belangrijkste segmenten zijn hier het PHASE_SEG1 en het PHASE_SEG2. Er kunnen 2 situaties optreden. Indien een zender traag uitzendt, dan kan het zijn dat er een flank gedetecteerd wordt na het SYNC_SEG. Dat is een positieve fasefout. Bij hersynchronisatie gaat men deze fout proberen weg te werken door het PHASE_SEG1 te verlengen met de absolute waarde van de fasefout. Het sample punt schuift dus op. Een andere situatie is wanneer er een flank gedetecteerd wordt voor het SYNC_SEG van de volgende bittijd (negatieve fasefout). Hersynchronisatie lost dit op door het PHASE_SEG2 in te korten met de absolute waarde van de fasefout.

Let wel op dat de verlenging en verkorting van respectievelijk PHASE_SEG1 en PHASE_SEG2 niet groter mag zijn dan "de resynchronization jump width" of SJW. Deze mag op zijn beurt niet groter zijn dan de lengte van PHASE_SEG1 en moet tussen 1 en 4 tijdskwantums liggen. Als de fasefout groter is dan de SJW, dan blijft er een fasefout bestaan die gelijk is aan de lengte van de fasefout verminderd met de lengte van het SJW.

Bij de volgende bittijd worden terug de oorspronkelijke lengten van PHASE_SEG1 en PHASE_SEG2 genomen. Voor PHASE_SEG1 is die gelijk aan 1-8 tijdskwantums en de lengte voor PHASE_SEG2 is gelijk aan grootste lengte van de twee volgende segmenten, IPT of PHASE_SEG1. Als IPT>PHASE_SEG2, dan wordt de lengte van PHASE_SEG2 gelijk aan de lengte van IPT. Analoog voor het geval IPT<PHASE_SEG2, waar de lengte van PHASE_SEG2 gelijk zal worden aan die van PHASE_SEG1. Deze waardes zijn instelbaar.

De twee PHASE_SEG zijn instelbaar daardoor kan men het sample punt verschuiven. Dit laat ons toe om de bittijd te optimaliseren, met een late sampling kan men de lengte van de bus verhogen en met een vroege sampling kan men de flanken trager laten stijgen of dalen.

Indien hersynchronisatie niet mogelijk is door bijvoorbeeld een reeks opeenvolgende identieke bits, worden de vooraf ingestelde waarden voor PHASE_SEG1 en PHASE_SEG2 genomen. Dit doet zich zelden of niet voor, want bitstuffing verhelpt dit probleem.

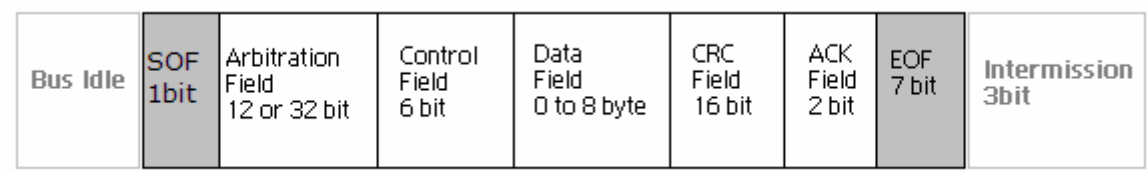
1.4 Data link laag

De data link laag bestaat uit 2 sublagen (figuur 3). Op de MAC sublaag worden de verschillende frames gedefinieerd, alsook het arbitrage proces en de foutdetectie. De LLC sublaag staat in voor het herverzenden van data tijdens een fout. Deze sublaag voorziet ook een gepaste reactie als een node overbelast is en tenslotte staat deze sublaag in voor het filteren van de frames.

1.4.1 De MAC sublaag

1.4.1.1 Het Data frame

De functie van het "Data frame" is data van een verzender over te dragen naar de ontvangers. Figuur 18 geeft de opbouw van zulk frame weer. Bij het verzenden van frames wordt de MSB (most significant bit) eerst verstuurd.

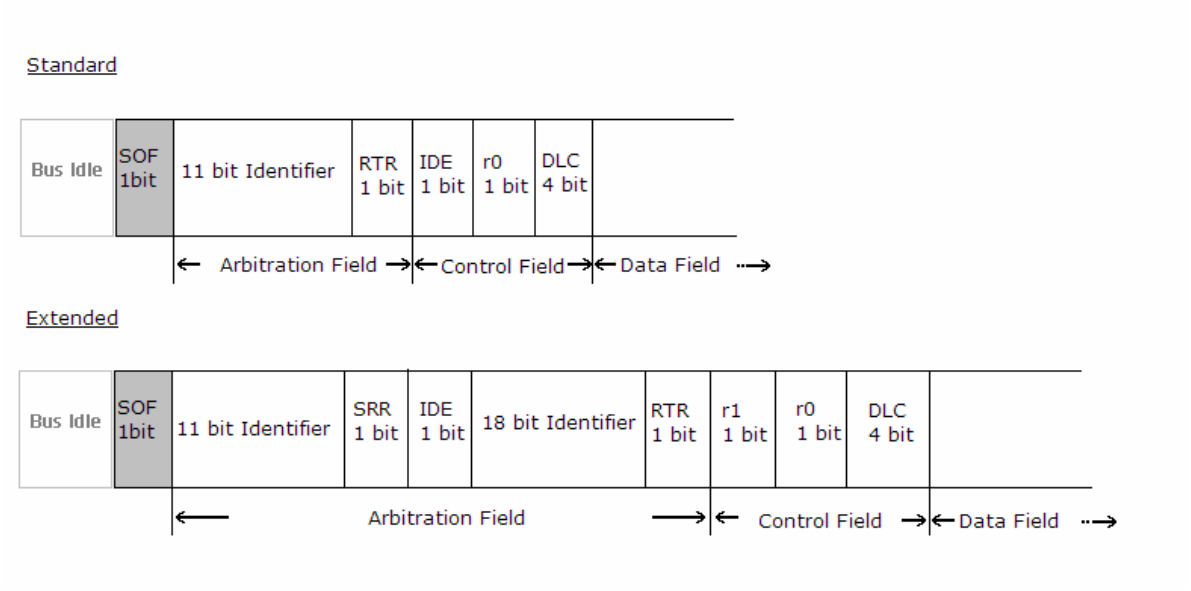


Figuur 18: Data frame

Het eerste veld is het "**Start Of Frame**" (SOF) veld en bestaat altijd uit 1 dominante bit. Zoals vermeld in puntje 1.3.3.3 gebeurt hier de harde synchronisatie.

Het "**Arbitration Field**" (Figuur 19) bestaat uit een "identifier" van 11 bit en een RTR of "Remote Transmission Request" bit in het standaard formaat. Dit formaat noemt men ook wel eens het CAN 2.0A formaat. In een data frame moet de RTR bit dominant zijn en in een remote frame moet hij recessief zijn. De RTR bit neemt ook deel aan het arbitrage proces waardoor een data frame altijd de overhand zal halen op een remote frame. Dit komt omdat een dominante bit het wint van een recessieve (voor verdere uitleg, zie het punt 1.4.1.5). Men kan aan de hand van een "identifier" ook weten wat voor soort bericht de data in het frame voorstelt. Elke node op een CAN-bus weet welke "identifiers" voor hem bestemd zijn.

Naast het standaard formaat heeft men ook een "extended" formaat (CAN 2.0B) of uitgebreid formaat ontwikkeld. De belangrijkste drijfveer voor dit tweede formaat was om de "identifier" uit te breiden. In het standaard formaat heeft men theoretisch 2^{11} mogelijkheden, maar de eerste 7 bits van de "identifier" mogen niet recessief (=logische 1) zijn, daarom houdt men in de praktijk nog 2^{11} -2^4 = 2032 mogelijke "identifiers" over. Dit wil zeggen dat de combinaties 1111111xxxx zijn gereserveerd en niet gebruikt mogen worden. In bepaalde toepassingen met veel verschillende soorten nodes en berichten heeft men niet voldoende aan 2032 "identifiers". In het uitgebreide formaat bestaat de "Identifier" uit 29 bit (11 bit + 18 bit), waardoor er theoretisch 2^{29} =536870912 mogelijkheden zijn. Maar zoals reeds vermeld, mag de 11 bit "identifier" (Figuur 19 "extended") niet bestaan uit de combinaties 1111111xxxx waardoor er in de praktijk 2032 x 2^{18} = 532676608 combinaties kunnen gebruikt worden.



Figuur 19: Arbitration field & Control field detail

In het uitgebreide formaat (Figuur 19 "extended") bestaat het "Arbitration Field" buiten de "identifiers" ook nog uit een SRR, IDE en een RTR bit. De SRR of de "Substitute Remote Request" bit staat op de plaats van de RTR bit. De SRR bit wordt altijd recessief verzonden en zal ervoor zorgen dat bij een botsing tussen een standaard en een uitgebreid frame (met dezelfde "11-bit identifier"), het standaard frame de overhand zal halen in het arbitrageproces. Op dit moment kan de ontvanger nog niet uitmaken met welk formaat hij te maken heeft, dit zal bepaald worden door de IDE of "Identifier Extension" bit. Als deze bit recessief is, weet de ontvanger dat er nog meerdere bits zullen volgen en dat de gehele "identifier" van het frame van het uitgebreide formaat is. De RTR bit bij het uitgebreide formaat vervult dezelfde functie als die bij het standaard formaat.

Het volgende veld in Figuur 18 is het "**Control Field**". De indeling van dit veld verschilt naargelang het formaat (standaard of uitgebreid). Zoals weergegeven in Figuur 19 bestaat het "Control Field" bij het standaard formaat uit een IDE bit die dezelfde functie heeft als de IDE bit in het "Arbitration Field" van het uitgebreide formaat, met als enige verschil dat hij in het standaard formaat dominant is. In het uitgebreide formaat is de eerste bit in het "Control Field" een gereserveerde bit (r0). Gereserveerde bits worden doorgaans dominant verstuurd, maar de ontvangers accepteren zowel dominant als recessief.

Vanaf de 2^{de} bit is het "Control Field" hetzelfde voor de 2 formaten. De 2^{de} bit (r0) is een gereserveerde bit en de 4 laatste bits van het "Control Field" bepalen de grootte van het "Data Field" dat na het "Control Field" zal volgen. Met 4 bits bestaan er 16 mogelijke combinaties, maar aangezien het "Data Field" maximum 8 bytes groot kan zijn, is er afgesproken dat de getallen met combinatie 1xxx (8 t.e.m 15), allemaal duidden op een 8 bytes lang "Data Field".

Het "**Data Field**" bevat de data van de verzendende node en kan 0 tot 8 bytes lang zijn. Men kan zich afvragen wat het nut is van 0 databytes te verzenden? Indien er bijvoorbeeld moet worden meegedeeld of een bepaald event heeft plaatsgevonden, kan men opteren om een leeg bericht te sturen. De ontvangende nodes weten aan de hand van de "identifier" welk event er plaatsgevonden heeft.

De data zelf hoeft niet te bestaan uit 1 welbepaalde soort data. Hij kan bestaan uit meerdere "channels" of kanalen. De eerste 4 bytes kunnen bijvoorbeeld een temperatuur voorstellen, terwijl de laatste 4 bytes het koppel voorstellen. Deze informatie is bij DAF vastgelegd in een DBC bestand (zie 7.1) dat ontwikkeld is door het bedrijf, Vector.

Na het "Data Field" volgt het "**CRC Field**" waarmee de ontvanger kan nagaan of er zich foute bits in het frame bevinden die opgetreden zijn tijdens het verzenden. Het veld bestaat uit een code van 15-bit lang, die men de CRC of "Cyclic redundancy check" noemt en een recessieve CRC "delimiter" van 1 bit lang. De totale lengte van het veld is dus 16 bit.

De CRC wordt door de verzender als volgt berekend:

$$\frac{X^{n-k}.M(X)}{G(X)} = Q(X) + \frac{R(X)}{G(X)}$$

M(X) is de bitstroom die bestaat uit de bits van het SOF, "Abitration", "Control" en "Data Field". De generatorveelterm G(X) is een 15-bit lange constante die voor CAN bepaald is op 1100010110011001, in veeltermvorm is dit $X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$. "n" is het aantal bits van het codewoord U(X) dat men gaat verzenden. "k" is het aantal bits van M(X). "n-k" geeft de lengte van G(X), bij CAN is dit altijd 15 bit. Q(X) is het quotiënt van de deling, maar deze waarde is onbelangrijk.

$$U(X) = X^{n-k}.M(X) + R(X) = Q(X).G(X)$$

U(X) is het codewoord dat men gaat versturen en bestaat uit een M(X) die 15 bit verschoven is + de rest R(X) van de deling. Men ziet in de bovenste formule, dat de rest van de deling van U(X) door G(X), nul als resultaat moeten opleveren.

De ontvanger maakt de volgende binaire deling:

Rest van de deling $\frac{U(X)}{G(X)} = 0?$

Wanneer de rest van deze deling niet nul bedraagt, is er een fout opgetreden in U(X) waardoor het frame genegeerd zal worden en de foutafhandeling van CAN zal optreden (zie 1.4.1.4).

De hamming distance bij CAN is 6, dit wil zeggen dat er 5 willekeurige foute bits in U(X) kunnen gedetecteerd worden. Bovendien kunnen blokken van maximum 15 aaneengesloten foute bits gedetecteerd worden.

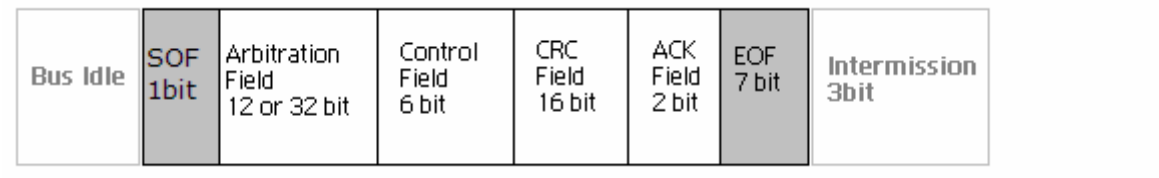
Het volgende veld is het "**ACK Field**". De eerste bit in dit veld is een bevestigingsbit en de tweede bit is een recessieve "delimiter", dewelke als functie heeft om het "ACK Field" af te sluiten.

De verzender verstuurt de 2 bits van dit veld altijd recessief, wanneer een ontvanger het frame van de verzender ontvangen heeft, overschrijft hij de recessieve waarde van de bevestigingsbit met een dominante. De ontvanger doet dit altijd als hij een frame ontvangen heeft, dus ook als er eventuele fouten in het frame staan. De verzender, die de bus na elke verzonden bit samplet (zie 1.3.3 De PLS sublaag), ziet de dominante bit en weet nu dat tenminste 1 node zijn frame correct ontvangen heeft.

Het laatste veld is het "**End Of Frame Field**" en bestaat uit 7 recessieve bits. Na het EOF veld komt er een "**Intermission**" periode. Tijdens deze periode moet de bus gedurende minstens 3 bits vrij zijn alvorens een volgend frame mag verzonden worden.

1.4.1.2 Het Remote Frame

Als een node bepaalde informatie nodig heeft dan kan het die expliciet opvragen bij andere nodes. Dit gebeurt door middel van een "remote frame". Als een bepaalde node een "remote frame" ontvangt, dan gaat hij een "data frame" met dezelfde "identifier" als die van het "remote frame" op de bus zetten.



Figuur 20: Remote frame

De indeling van een "remote frame" is dezelfde als die van een "data frame" met een "Data Field" van 0 bytes (Figuur 20). De RTR bit is bij een "remote frame" recessief en het "Control Field" geeft nu de grootte van de gevraagde data op.

1.4.1.3 Soorten fouten

Er zijn 5 soorten fouten die kunnen herkend worden en die optreden tijdens een dataoverdracht:

- Een bitfout
- Een stuff-bitfout
- Een ACK-fout
- Een formaatfout
- CRC-fout

Na elke fout wordt er een foutafhandeling procedure opgeroepen, de fouten zullen nu in detail besproken worden.

Iedere verzender samplet de bus na elke verzonden bit. Wanneer hij ziet dat er een andere bit op de bus staat dan diegene die hij verzonden heeft, neemt hij een bitfout waar. Er mag natuurlijk alleen op een bitfout gecheckt worden na het arbitrage proces en er mag ook niet gecheckt worden tijdens het verzenden van de bevestigingsbit in het "ACK Field".

Een stuff-bitfout wordt gedetecteerd als er meer dan 5 identieke bits op mekaar volgen. Dit zou normaal niet mogelijk zijn met bitstuffing.

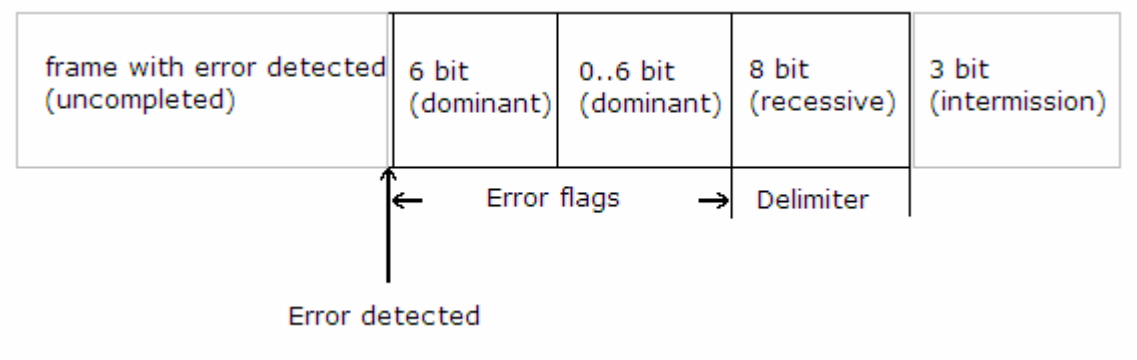
Als een zender ziet dat zijn recessive bevestigingsbit niet overschreven is door een dominante bit, dan treedt er een ACK-fout op. Dit wil immers zeggen dat geen enkele ontvanger zijn bericht ontvangen heeft (zie 1.4.1.1 "ACK Field").

Bepaalde velden van een CAN-frame hebben een vaste inhoud. Delimiters bijvoorbeeld, moeten altijd bestaan uit een recessieve bit en het "EOF Field" moet altijd bestaan uit recessieve bits. Indien er in deze velden een dominante bit wordt gedetecteerd, spreekt men van een formaatfout.

Er treedt een CRC-fout op bij een ontvanger als de rest van de binaire deling van het codewoord en de generatorveelterm niet 0 is. (zie 1.4.1.1 "CRC Field").

1.4.1.4 Foutafhandeling

Wanneer een node, dit kan de verzender of een ontvanger zijn, 1 van de vorige fouten herkent, verstuurt hij een "error flag" of foutsignaal (actief of passief, zie verder). Dit signaal bestaat bij een actieve node uit 6 dominante bits en bij een passieve node uit 6 recessieve bits. Omdat deze bits bij een actief foutsignaal (normale werking) dominant zijn, wordt het huidige frame overschreven. Alle nodes op de bus herkennen nu zeker een stuff-bitfout (of een bitfout of formaatfout) en zenden op hun beurt een foutsignaal. Er wordt even aangenomen dat alle nodes in een actieve toestand zijn. Er zal een superpositie optreden van alle foutsignalen op de bus, waardoor een bitstroom van minimum 6 en maximum 12 dominante bits zal ontstaan (Figuur 21)



Figuur 21: error frame with active error flags

Een lokaal gedetecteerde fout wordt dus globaal gemaakt voor de gehele bus. Dit zorgt ervoor dat geen enkele node verkeerde data ontvangt en daardoor blijft het geheel consistent. Na het verzenden van hun foutsignaal, beginnen de nodes met het verzenden van recessieve bits. Dit doen ze net zolang tot ze op de bus ook een recessieve bit zien. Dit wil zeggen dat alle nodes hun foutsignaal beëindigd hebben. Daardoor kunnen de recessieve bits niet meer overschreven worden door de dominante van de (in dit voorbeeld) actieve foutsignalen. Vervolgens zenden de nodes 7 recessieve bits, zodat de bus voldoende tijd heeft om opnieuw te starten. De laatste 8 bits noemt men de "Error delimiter" (Figuur 21).

Het "EOF Field" bij het data en remote frame zorgt ervoor dat een fout zeker herkend wordt voordat een frame beëindigd is. Na het aflopen van een "Error frame" zal de zender het bericht nog eens proberen te versturen, als hierbij een botsing optreedt, wordt het arbitrage proces hervat.

Maar wat als een node inwendig defect is zodat hij continu berichten met foute bits gaat versturen? Als dit voorkomt, gaat heel de bus blokkeren. Daarom heeft men 3 verschillende toestanden gedefinieerd waarin een node zich kan bevinden (Figuur 22). Het veranderen van 1 toestand naar een andere wordt bepaald door 2 tellers die per node bijgehouden worden, de "Transmitted Error Counter" of TEC en de "Received Error Counter" of REC.

De werking van de tellers is onderhevig aan de volgende regels:

- Indien een ontvanger een bericht zonder fouten ontvangt (en na het sturen van de bevestigingsbit), wordt de REC verminderd met 1 als hij tussen 1 en 127 lag, een waarde toegekend tussen 119 en 127 als hij groter was dan 127 en hij blijft 0 als hij reeds 0 was.

- Wanneer een ontvanger of verzender een bitfout detecteer terwijl hij een "Active Error Flag" of een "Overload Frame" (zie 1.4.1.6) aan het versturen is, worden de REC of TEC verhoogd met 8, afhankelijk of het respectievelijk de ontvanger of verzender was die de bitfout detecteerde.

- Als een ontvanger een dominante bit detecteert na het versturen van een foutsignaal, zal de REC verhoogt worden met 8.

- Wanneer een verzender een foutsignaal verstuurd, wordt zijn TEC verhoogd met 8. Er zijn 2 situaties waarbij deze regel niet geldig is. Ten eerste als de verzender in een passieve toestand is en een passief foutsignaal verstuurd ten gevolge van een bericht dat niet bevestigd is. Ten tweede als de verzender een foutsignaal verstuurt tengevolge van een stuff-bitfout die optrad tijdens het arbitrage proces.

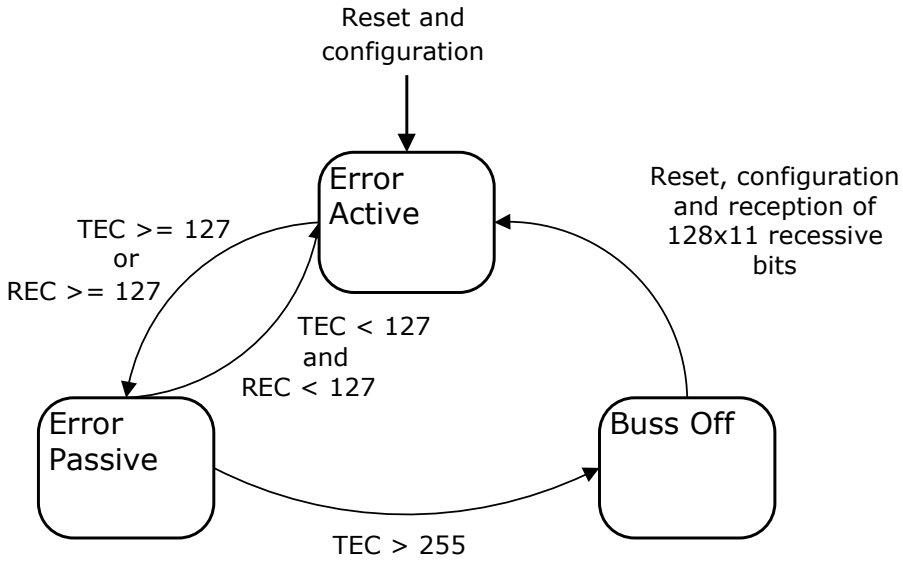
- Na het versturen van een "overload flag" (zie 1.4.1.6) of een actief of passief foutsignaal door eender welke node, zal de desbetreffende node 7 opeenvolgende dominante bits toelaten. Na het detecteren van de 14de opeenvolgende dominante bit of na het detecteren van 8 opeenvolgende dominante bits gevolgd door een passief foutsignaal en na elke blok van 8 opeenvolgende dominante bits, zal elke verzender zijn TEC met 8 verhogen en elke ontvanger zijn REC met 8 verhogen.

- Bij alle andere fouten die optreden tijdens het ontvangen, wordt de REC verhoogd met 1.

- Na het succesvol verzenden van een bericht zal de TEC van de verzender met 1 verminderd worden.

Een node is bij normale werking (beide tellers < 127) in een "Error Active State" of een actieve toestand.

Als 1 van beide tellers gelijk is of groter dan 127 wordt, verandert de node zijn toestand naar "Error Passive". Nu kan hij alleen nog "Passive error flags" of passieve foutsignalen versturen. Deze bestaan, zoals reeds vermeld, uit een reeks van 6 recessieve bits. Als dit passief foutsignaal het enige signaal op de bus is, zien de andere nodes een stuff-bitfout en beginnen ze op hun beurt foutsignalen te verzenden (passief of actief). Als de node in passieve toestand niet de enigste verzender is (of hij is ontvanger), dan heeft het passief foutsignaal dat hij verstuurt geen invloed op de bus omdat het recessief is. Een bijkomende beperking dat een passieve node ondervindt, is dat het na een "Intermission" nog eens 8 recessieve bits moet verzenden/wachten vooraleer hij terug mag beginnen met het versturen van zijn volgende bericht. Als in tussentijd een andere node begint met verzenden, dan zal de passieve node ontvanger worden van dit bericht. Deze extra wachttijd zorgt ervoor dat een defecte node de bus niet zal blokkeren.



Figuur 22: Toestandsdiagramma foutafhandeling

Een node gaat naar de "Bus-Off" toestand als zijn TEC groter wordt dan 255. In deze toestand kan de node niets meer versturen of ontvangen, hij is volledig afgesloten van de bus. Een node gaat terug naar de actieve toestand als hij 128 keer een blok van 11 recessieve bits op de bus ziet.

De kans op fouten die niet gedetecteerd worden is zeer klein. Er is door de Cia of "CAN in Automation" berekend dat die kans kleiner is dan $4{,}7 \times 10^{-11}$. Er wordt aangenomen dat een CAN-bus 8 uur per dag operationeel is en dat een jaar 365 dagen telt. Voor een CAN-bus met 1 fout om de 0,7s betekent dat $1{,}5 \times 10^{10}$ fouten op een tijdspanne van 1000 jaar. Als we dit aantal fouten vermenigvuldigen met de kans op fouten die niet gedetecteerd worden ($1{,}5 \times 10^{10} \times 4{,}7 \times 10^{-11}$), dan wordt er 1 fout om de 1000 jaar niet gedetecteerd.

1.4.1.5 Arbitrage

Als zenders iets willen verzenden moeten ze wachten tot het lopende bericht beëindigd is of ze moeten gelijktijdig beginnen met verzenden waardoor er een arbitrage zal plaatsvinden.

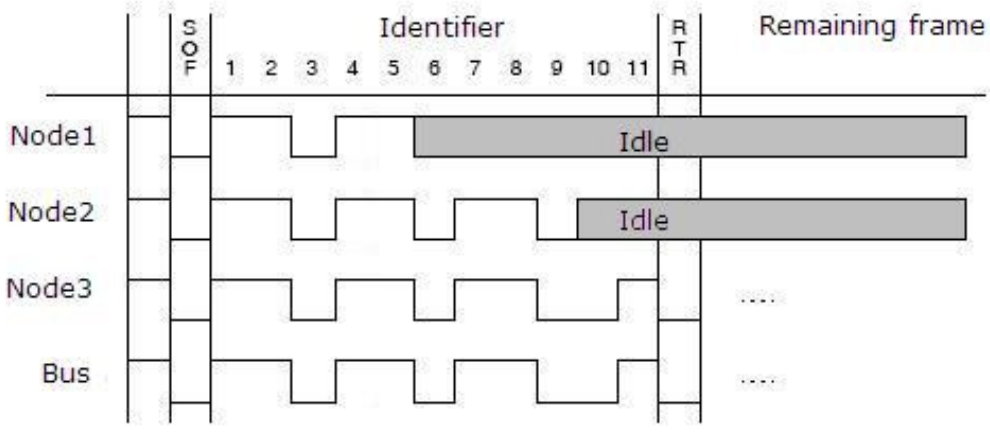
CAN hanteert het principe van "Carrier Sense Multiple Access with Collision Avoidance" (CSMA/CA). Wanneer de CAN-bus vrij of idle is, dan kan elke node beginnen met verzenden. Elke node checkt de bus voor hij begint met verzenden (Carrier Sense). Alle nodes hebben evenveel recht om te beginnen verzenden (multiple access), daardoor kan het zijn dat er 2 of meerdere nodes tegelijk beginnen te verzenden. Men lost dit probleem op met "non-destructive arbitration" (Collision Avoidance). Dit soort arbitrage zorgt ervoor dat er geen enkel frame verloren gaat. Een ander belangrijk kenmerk van deze arbitrage is dat er geen kostbare tijd verspild wordt.

Arbitrage wordt uitgelegd aan de hand van een voorbeeld (Figuur 26). Het proces wordt toegepast op de ID's of "identifiers" van elk frame. Het ID bepaalt de prioriteit van het desbetreffende frame. Stel dat er 3 nodes tegelijk willen verzenden en dat het frame dat ze willen verzenden volgende ID's hebben:

ID1 = 11011**1 010 10**

ID2 = 110110 110 **11**

ID3 = 110110 110 01



Figuur 23: Arbitrage

Zoals vermeld bij puntje 1.3.3.3 wint een dominante bit (=logische 0) het van een recessieve bit (=logische 1). De eerste 5 bits (11011) zijn bij alle drie de zenders hetzelfde, vanaf de 6^{de} bit doet ID1 niet meer mee in het arbitrage proces, want daar verliest zijn 6^{de} bit het tegen de dominante 6^{de} bit van ID2 en ID3. De volgende bits ('110') zijn bij ID2 en ID3 hetzelfde, vanaf de 10^{de} bit verliest ID2 het tegen de 10^{de} bit van ID3 die dominant is. De winnaar van deze arbitrage is node3 die het frame met ID3 verstuurde. Men kan opmerken dat in dit proces (arbitrage) geen tijd verloren is gegaan.

De verliezende nodes zullen wachten tot de bus weer idle is en ze zullen daarna opnieuw proberen om het frame te verzenden, in tussentijd functioneren ze als ontvanger. Er is bijgevolg geen informatie verloren gegaan.

De frames met een hoge prioriteit (dus met het kleinste ID) winnen het arbitrage proces. Dit proces is zeer geschikt voor real-time toepassingen. Een nadeel van deze arbitrage is dat de berichten met lage prioriteit een grote vertraging kunnen ondervinden als er op dezelfde tijd veel berichten met hoge prioriteit verstuurd worden.

Zoals reeds vermeld zal in het geval dat een data frame en een remote frame met dezelfde ID op hetzelfde moment beginnen met uitzenden, het data frame de overhand halen.

1.4.1.6 Het Overload Frame

Dit frame wordt door een node gebruikt om de bus te waarschuwen dat hij nog niet klaar is om berichten te ontvangen. Het frame zorgt voor een bijkomende vertraging tussen 2 berichten. Een "Overload Frame" heeft exact dezelfde indeling als Figuur 21 en wordt dus ook geglobaliseerd door alle deelnemende nodes.

De volgende regels vertellen in detail wanneer een "Overload frame" verstuurd zal worden.

De eerste regel zegt dat wanneer een ontvanger ziet dat hij, ten gevolge van een inwendige vertraging, een volgend bericht niet meer tijdig kan verwerken, hij een vertraging zal aanvragen door middel van een "Overload frame".

De tweede voorwaarde waarop dit frame zal verstuurd worden, is voldaan als een node een dominante bit ziet op de eerste of tweede plaats van een "Intermission" (de periode tussen 2 berichten).

De derde en laatste voorwaarde om dit frame te versturen is voldaan als een node een dominante bit ziet (samplet) op 8^{ste} bit van een "Error Delimiter" (Figuur 21).

Bij het versturen van een "Overload frame" behouden de REC en de TEC (zie puntje 1.4.1.4 Foutafhandeling) hun waarde.

1.4.2 De LLC sublaag

Communicatie tussen de verschillende nodes is niet gebaseerd op adres informatie, zoals bij TCP/IP, maar elk verzonden bericht wordt ontvangen door elke node ("broadcast") en elke node moet voor zichzelf uitmaken of dat bericht voor hem bedoeld is ("acceptance filtering"). Dit is vooral handig wanneer bepaalde data op meer dan 1 locatie dient ontvangen te worden. Sommige hogere lagen bouwen wel een soort adressering in, maar die is altijd gebaseerd op "acceptance filtering".

Op deze sublaag worden ook de waarschuwingen (“Overload frames”) gegeven als een node overbelast is en ze staat ook in voor het herverzenden van foute berichten.

1.4.3 Hogere lagen

Het CAN protocol definieert de fysische en de data link laag. Voor de hogere lagen zijn er andere protocols ontwikkeld. Enkele van deze hogere protocollen zijn DeviceNet, CANopen en SAE J1939. Dit zijn gestandaardiseerde protocollen, het is ook mogelijk een eigen protocol te ontwikkelen. DeviceNet en CANopen worden toegepast in industriële toepassingen.

Meestal gebruikt men bij voertuigen het SAE J1939 protocol. SAE staat voor “Society of Automotive Engineers” en is een Amerikaanse organisatie van automobieldeskundigen.

1.4.3.1 Het SAE J1939 protocol: Inleiding

Bij DAF baseert men zich op het SAE J1939 protocol. Voor onze toepassing is het belangrijk dat men weet dat dit protocol opereert op een snelheid van 250kbps (1850 berichten per seconde) en dat er gebruik wordt gemaakt van het uigebreide formaat (“29-bit identifier”, zie 1.4.1.1).

Het SAE J1939 protocol is vastgelegd in verschillende specificaties, die ruwweg het 7-lagen OSI model volgen (Figuur 24). Dit protocol heeft buiten de hogere lagen ook nog zijn specifieke afspraken voor de fysische laag (die vastgelegd zijn in SAE J1939-11 en die besproken zijn in het punt “1.3.2 De PMA sublaag”) en de data link laag (SAE J1939-21). Deze specifieke afspraken bouwen verder op hetgeen besproken werd in punt 1.3 en 1.4.

J1939/11 Physical layer (250 kbit/s, twisted shielded pair)
J1939/13 Off-board diagnostic connector
J1939/15 Reduced physical layer (250 kbit/s, unshielded twisted pair)
J1939/21 Data link layer
J1939/31 Network layer
J1939/71 Vehicle application layer
J1939/73 Application layer diagnostics
J1939/74 (Draft) Application - Configurable messaging
J1939/75 Application layer - Generator sets and industrial
J1939/81 Network management

Figuur 24: SAE J1939 Specificaties

1.4.3.2 Het SAE J1939 protocol: Parameter groups

Vooraleer de specifieke indeling van de identifier van dit protocol behandeld wordt, eerst een woordje uitleg over “Parameter Groups” of PG’s. De specificaties van PG’s zijn vastgelegd in J1939-71. In een PG zitten signalen die op mekaar gelijken of die met mekaar verwant zijn. In een CAN-frame kan men maximaal 8 bytes aan data versturen, als de 8 bytes grens overschreden wordt, zal er gebruik gemaakt worden van een transport protocol (zie verder). Elk soort data wordt ondergebracht in een bepaalde PG die de parameters zoals onder andere de grootte, locatie van de LSB, transmissiesnelheid en prioriteit van het bericht bepaald.

Men kan de PG van een bericht achterhalen door de “Parameter Group Number” of de PGN. Deze bestaat uit een 18 bit waarde en die is samengesteld uit de “Reserved” en “Data page” bit, het “PDU format” en het “PDU specific” veld van de “identifier” (zie

Figuur 26). Wanneer een node een bericht ontvangt weet hij dus aan de hand van het PGN welk type data het bericht met zich meedraagt.

Voorbeeld van een PG
Name: Engine temperature
Transmission rate: 1s
Data length: 8 bytes
Data page: 0
PDU format: 254
PDU specific: 238
Default priority: 6
PG Number: 65,262 (FEEE16)
Description of data:
Byte: 1 Engine coolant temperature
Byte: 2 Fuel temperature
Byte: 3,4 Engine oil temperature
Byte: 5,6 Turbo oil temperature
Byte: 7 Engine intercooler

Figuur 25: Voorbeeld van een PG

De data in een DBC bestand is op soortgelijke manier opgevat (zie 7.1).

1.4.3.3 Het SAE J1939 protocol: de data link laag

Priority	Reserved	Data page	PDU format	PDU specific	Source Address
3 bits	1 bit	1 bit	8 bits	8 bits	8 bits

Figuur 26: SAE J1939 29-bit identifier

Bij SAE J1939 heeft men een specifieke indeling van de beschikbare bits van de "29-bit identifier" (Figuur 26) en dit is vastgelegd in SAE J1939-21.

De eerste 3 bits kan men gebruiken om de prioriteit van het desbetreffende bericht aan te geven. Deze 3 bits zijn de eerste die het arbitrage proces ondergaan,"000" is de hoogste prioriteit.

De volgende bit is gereserveerd voor toekomstige doeleinden. Hij wordt voorlopig op 0 gezet.

Data page bit zorgt ervoor dat het aantal mogelijke PG's nog uitgebreid kan worden.

Het volgende veld is het PDU ("Protocol Data Unit") format of PF veld, m.a.w. het formaat van het bericht. De inhoud van het "PDU specific" of PS veld hangt af van de waarde van PF veld.

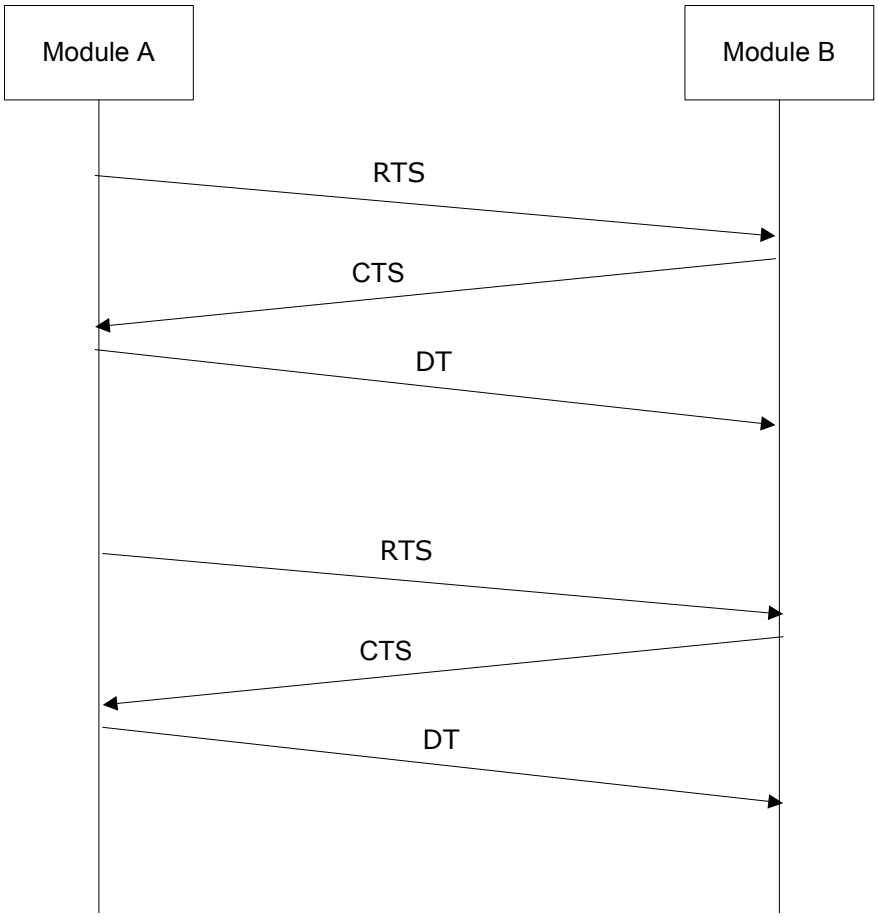
Ligt de waarde van het PF veld tussen 0 en 239, dan zal het PS veld het adres van de bestemming bevatten en noemt men het bericht een "PDU1 bericht" . Indien het adres 0xFF wordt gebruikt, dan is het bericht bestemd voor iedereen ("broadcast").

Wanneer de waarde van het PF veld tussen 240 en 250 ligt, dan zal het PS veld een GE ("Group Extension") bevatten. Deze GE breidt het aantal PG uit. Dit soort bericht kan

alleen maar naar iedereen gestuurd worden (“broadcast”) en men noemt dit een PDU2 bericht.

Het laatste veld is het “Source Address” veld en dit bevat een label dat uniek is voor elke node die een bericht verstuurt. Op die manier zorgt men ervoor dat elke identifier uniek is. Elke node kan een uniek adres bekomen via de procedure “address claiming” dewelke vastgelegd is in SAE J1939-81. Gedetailleerde uitleg van dit proces zou ons te ver leiden.

Er wordt een speciale procedure in werking gezet als de data uit meer dan 8 bytes bestaat. Een CAN-frame kan immers maar maximum 8 bytes aan data bevatten. De procedure is ruw geschetst in Figuur 27.



Figuur 27: Transport protocol

Er wordt eerst een “request to send” of RTS gestuurd naar module B, waarna die antwoord met “clear to send”. Vervolgens stuurt module A het eerste segment over (DT = data transmission). Als module B alle segmenten heeft ontvangen, kan hij de data reconstrueren. Het “Data Field” van zo een segment bestaat uit 7 bytes data en 1 byte die de sequentienummer van het segment bevat.

Een andere manier om data die uit meer dan 8 bytes te versturen is de “broadcast” dewelke wordt aangekondigd met een BAM of “Broadcast Announce Message”. Deze BAM ontvangt elke node en daardoor weten ze dat er broadcast berichten zullen volgen.

Deze 2 methodes zijn vastgelegd in SAE J1939-21

2 USB

2.1 Geschiedenis

De eerste specificatie van de Universal Serial Bus, USB 1.0, werd geïntroduceerd eind 1995 door Intel, Microsoft, Philips en US Robotics.

De originele drijfveer voor de ontwikkeling van USB, was het verminderen van het kluwen van verschillende kabels die de verscheidene randapparatuur met de PC verbond.

Eind 1998 werd de USB 1.1 standaard geïntroduceerd dewelke een reeks problemen van USB 1.0 verhielp.

In 2000 kwam de USB 2.0 standaard uit, die ontwikkeld is door Hewlett-Packard, Intel, Lucent (Alcatel-Lucent), Microsoft, NEC en Philips. Het doel van USB 2.0 was om een hogere datasnelheid dan USB 1.1 te bekomen. USB 2.0 is "backwards compatible" wat betekent dat elk apparaat dat voldoet aan eender welke versie van USB, kan samenwerken met eerdere versies.

2.2 Algemene kenmerken

USB is hot-pluggable wat betekent dat er op eender welke moment een nieuw apparaat op de USB-bus kan aangesloten worden. De juiste driver zal automatisch geladen worden, en na het verwijderen van het apparaat zal die driver vervolgens automatisch afgesloten worden.

Er kan maar 1 host tegelijk op de bus aanwezig zijn, meestal is dit de PC. Deze host heeft het commando over de bus.

Er kunnen maximum 127 apparaten op een USB bus aangesloten worden, dit gebeurt door middel van hubs (2.3.3).

2.3 USB componenten

2.3.1 Elektrisch

USB is een seriële bus en zijn kabel bestaat uit 4 draden die shielded zijn:

- +5V
- D+
- D-
- Ground

Omdat USB apparaten gevoed kunnen worden via de bus is er een 5V lijn meegeleverd. Als er apparaten zijn die meer dan 500 mA nodig hebben, is het aangeraden die apart te voeden. Een andere oplossing is om deze laatstgenoemde apparaten met een 2de USB kabel te voorzien, hiermee kunnen ze dan 500 mA + 500 mA van de host of hub ontvangen. Het nadeel is echter dat er in de host of hub een extra USB slot bezet is.

D+ en D- worden gebruikt voor dataoverdracht en vormen een twisted pair. De dataoverdracht is differentieel. Bij USB 1.1 is een verschilspanning van 200 mV nodig en bij USB 2.0 400 mV.

De maximum vertraging dat een signaal op de USB bus mag ondervinden is 5.2 ns per meter. De maximum totale vertraging is vastgelegd op 26 ns, dus een USB kabel mag niet langer zijn dan 5 meter.

De USB bus heeft een karakteristieke weerstand van 90 Ω.

De snelheden van de verschillende specificaties van de USB bus zijn als volgt:

USB 1.1

- 1,5 Mbps (low speed mode)
- 12 Mbps (full speed mode)

USB 2.0

- 480 Mbps (high speed mode)

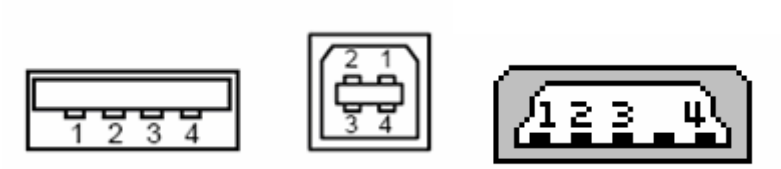
Men kan opmerken dat USB 2.0 40 keer sneller is dan USB 1.1 (full speed mode).

USB 3.0 dat op deze moment in ontwikkeling is, zal een datasnelheid van ongeveer 4,8 Gbps hebben.

De codering van de bits gebeurt met een NRZI (Non return to zero, inverted) codering. Deze bepaalt dat er alleen bij een logische 0 van niveau dient veranderd te worden. Bij een logische 1 zal er geen niveauverandering optreden. Het veranderen van niveau gebeurt enkel op een flank van de klok. Net zoals bij CAN zal er ook hier bitstuffing toegepast worden.

2.3.2 Connectors

Aan de kant van de host (meestal PC) zijn de kabels voorzien van een type A aansluiting, aan de kant van het apparaat zijn ze voorzien van een type B aansluiting. Op deze manier kan men geen verkeerde aansluitingen maken.



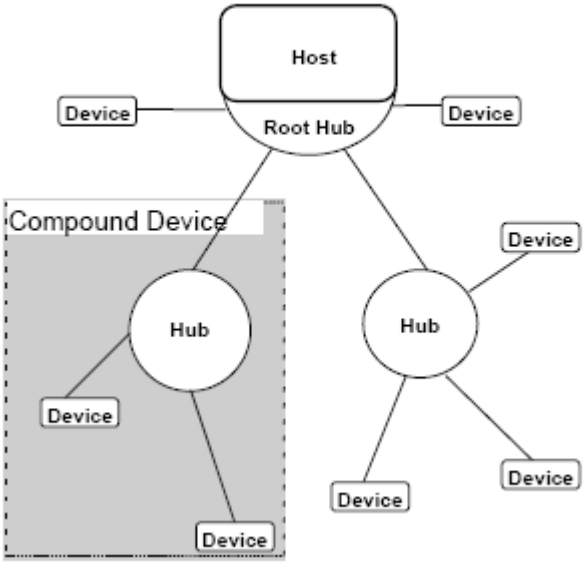
Figuur 28: In volgorde Type A, Type B, Mini-B

Een recente ontwikkeling is die van de mini-B aansluiting, deze aansluiting vind zijn toepassing in kleine mobiele aparaten (zoals GSM).

2.3.3 Hubs

Men spreekt van de Universal Serial Bus, maar in wezen is USB geen echte bus. Er kan immers maar 1 host zijn die het commando over de "bus" voert. De host regelt alle transacties en hij verdeelt de beschikbare bandbreedte over de aangesloten apparaten.

In realiteit is de topologie van de USB-bus een boomstructuur (Figuur 29).



Figuur 29: Topologie USB

Hubs spelen een essentiële rol in deze topologie. De hub heeft 1 aansluiting die van de host afkomstig is (al dan niet via een andere hub) en aansluitingen waar men USB apparaten of een andere hub op kan aansluiten. Apparaten die op een hub aangesloten zijn, hebben nog steeds de mogelijkheid om gevoed te worden via de bus

Door de opbouw van de topologie kan men vertrekkende van 1 USB aansluiting d.m.v. hubs veel meer USB aansluitingen voorzien. Men kan opmerken dat er telkens maar 1 host aanwezig is die het commando over de bus voert.

2.4 USB transfer modes

De USB-bus kan men onderscheiden van een gewone seriële bus o.a. door middel van zijn verschillende transfer modes. Tijdens de communicatie van een apparaat met zijn host worden er pipe(s) opgezet. Pipes zijn logische connecties die gebruikt worden om data te verzenden. Ze hebben elk hun eigen parameters (toegestane bandbreedte, type transfer, richting van het versturen en buffergroottes). Pipes kunnen zowel unidirectioneel als bidirectioneel zijn.

Algemeen zijn er 2 soorten pipes in USB gedefinieerd:

- Stream Pipes zijn unidirectioneel en ze bezitten geen vastgelegd USB formaat Door dit soort pipe kan men eender welk type data versturen. De transfer modes die dit type pipe gebruiken zijn de bulk, isochronous and interrupt transfer modes.

- Message Pipes zijn bidirectioneel en ze bezitten een vastgelegd USB formaat. De control transfer mode gebruikt dit type van pipe.

2.4.1 Control

Dit type pipe wordt gebruikt voor operaties die te maken hebben met het configureren van het (zonet) aangesloten apparaat.

Het verzenden van pakketten in deze pipe gebeurt meestal in bursts.

Deze pipe kan ook andere type pipes controleren.

2.4.2 Interrupt

Indien een USB apparaat de aandacht wil opeisen van de host, moet hij wachten totdat de host hem polt. Pas nadat dit gebeurt is, kan hij zijn interrupt transfer pipe met de host opzetten.

Deze pipe heeft de volgende specificaties

- Vaste vertraging.
- Stream Pipe dus unidirectioneel.
- Foutdetectie en herverzenden van foute pakketten.

De aard van deze transfers zijn meestal niet-periodisch.

2.4.3 Isochronous

De aard van dit type van transfer is dat ze continu en periodisch verloopt. Ze bevat meestal tijdskritische data zoals audio of video data (bijvoorbeeld webcam).

De specificaties zijn:

- Een op voorhand vastgelegde bandbreedte
- Een vertraging die binnen bepaalde grenzen blijft.
- Stream Pipe dus unidirectioneel.
- Alleen foutendetectie via CRC maar geen herverzending van foute pakketten of de zekerheid dat de verzonden data goed afgeleverd werd.
- Alleen beschikbaar in full of high speed modes.

2.4.4 Bulk

Deze pipe wordt gebruikt als er een grote hoeveelheid data moet verzonden worden die plots opkomt (bursts). Een voorbeeld van dit soort data is de data afkomstig van een printer of een beeld van een scanner.

De specificaties zijn:

- Wordt gebruikt tijdens het verzenden van plotse grote hoeveelheden data.
- Foutdetectie en herverzenden van foute pakketten.
- Zekerheid van ontvangst
- Er is geen minimum vertraging vastgelegd, de vertraging hangt af van de load van de bus
- De bandbreedte is niet vastgelegd, de beschikbare bandbreedte op de bus wordt benut
- Stream pipe dus unidirectioneel.
- Alleen beschikbaar in full of high speed modes.

Deze mode mag niet gebruikt worden voor het verzenden van tijdskritische data. Een bulk transfer is afgerond wanneer het de hoeveelheid data verstuurd heeft die aangevraagd werd.

3 ALGEMENE BESCHRIJVING VAN HET PROJECT

3.1 De opdrachtgevers

3.1.1 CIT

Het CIT is gesitueerd in de Kleinhoefstraat 6 te Geel en staat voor "Centrum voor Industriële Toepassingen". Als spin-off van de KHK zijn ze gestart en ze hebben ondertussen vestigingen in België, Polen en Nederland. Ze hebben o.a. een grote expertise in LabVIEW opgebouwd en ze zijn actief op verschillende domeinen zoals:

- "Vision and motion"
- data-acquisitie
- mechanica
- "Industrial and Measurement Automation"
- …

En in de volgende sectoren:

- Metingen en testen
- Process controle
- Productie
- Bouw
- Voeding en medicijnen
- Medisch
- Energie
- Onderzoek

Bij het CIT bestaat ook de mogelijkheid om trainingen te volgen. Voor meer informatie kunt u terecht op www.citengineering.com.

3.1.2 DAF Trucks Eindhoven

DAF Trucks is onderdeel van de PACCAR groep waar o.a. ook Kenworth en Peterbilt toe behoren. Deze 2 laatste vrachtwagenmerken genieten het meeste aandacht in Amerika.

DAF staat voor "van Doorne's Automobiel Fabriek" en werd in 1932 opgericht door de Nederlander Hub Van Doorne. Na een bewogen geschiedenis werd DAF in 1996 overgenomen door PACCAR.

De productie van motoren en componenten gebeurt in Eindhoven, waar het hoofdkwartier van het bedrijf gevestigd is. De cabines en de assen krijgt de assemblagelijn in Eindhoven toegeleverd vanuit Westerlo.

Meer informatie vind u op www.daftrucks.com.

3.2 De opdracht

De opdracht bestond er in om een synchrone video/CAN-bus logger te maken. Deze logger zal ingebouwd worden in een testvrachtwagen. Die vrachtwagen dient uitsluitend om testen/proeven uit te voeren op nieuwe systemen, bijvoorbeeld een nieuwe versnellingsbak.

Wanneer een nieuw systeem dient getest te worden, wordt dit ingebouwd in de testvrachtwagen. Waarna die er mee gaat rondrijden. Zodra de chauffeur iets raar opmerkt, zoals bijvoorbeeld het even niet functioneren van het te testen systeem, dan dient dit gelogd te worden. De trigger voor het starten van een logfile wordt gegeven door het indrukken van een hardware-knop die de chauffeur ter beschikking heeft.

Wanneer de testvrachtwagen, bijvoorbeeld na een week, terugkeert van zijn rit, is het de bedoeling dat de testingenieurs bij DAF de gelogde data kunnen analyseren.

Een "Viewer" is daarom noodzakelijk om de gelogde data op een overzichtelijke manier op het scherm weer te geven. Hij verzorgt ook de synchronisatie tussen video en CAN-data. De Viewer is dus een onmisbare tool om de gelogde data te analyseren.

Er zullen 2 zaken moeten gerealiseerd worden:

- Logger
- Viewer

3.3 De oplossing

3.3.1 De logger

3.3.1.1 Onderdelen

- een embedded PC met speciale voeding voor gebruik in een vrachtwagen
- een USB-webcam
- een hardware trigger-knop met status LED's
- een USB-CAN DAQ kaart
- een USB DAQ kaart met digitale in –en uitgangen (aanstuurbaar vanuit LabVIEW)
- een externe USB hard disk
- een USB2.0 Hub

Alle onderdelen worden in de vrachtwagen geplaatst en op de embedded PC aangesloten (zie Figuur 32). De software is volledig in LabVIEW geschreven.

3.3.1.2 Algemene werking logger

Men start met het inbouwen van het te testen systeem in de testvrachtwagen. In deze vrachtwagen is de logger al reeds ingebouwd (Figuur 32), behalve de externe HD.

Voor het starten van de proef stelt de testingenieur de pretrigger en posttrigger tijd in. Dit gebeurt door het manueel aanpassen of aanmaken van een .ini bestand. In Figuur 30 kan men de inhoud van zo een bestand waar de post –en pretrigger tijd op 60 seconden zijn ingesteld, zien.

```
[TriggerSettings]
PreTrigger=60
PostTrigger=60
```

Figuur 30: Voorbeeld van het configuratiebestand

Vervolgens hernoemt de testingenieur het bestand naar "DataLoggerConfiguration.experiment" en kopieert het naar de volgende folder op de externe schijf: \Configuration\. Als deze map nog niet bestaat, dan dient de testingenieur deze zelf aan te maken. Wanneer deze procedure afgehandeld is, plaatst de testingenieur de externe HD in de vrachtwagen en koppelt hij hem aan de embedded PC via USB. De logger weet nu welke instellingen hij moet gebruiken voor de post –en pretrigger tijd.

De logger is op dit moment operationeel en de proef kan starten. Wanneer de chauffeur zijn vrachtwagen start, zal de embedded PC automatisch mee opstarten. De logger start op zijn beurt mee op. Het programma zal meteen beginnen met het bijhouden van een buffer op de interne schijf van de embedded PC. De buffer bestaat uit video en CAN-data en bedraagt x-aantal(instelbaar) minuten. Deze instelbare tijd wordt vanaf nu de pretrigger time genoemd.

Wanneer de logger een trigger detecteert, die gegeven wordt door de hardware-knop, is het de bedoeling dat hij de data van de webcam en de CAN-bus gaat loggen op de externe harde schijf. De gelogde data per trigger zal bestaan uit een tijdsspanne van de pretrigger time + x-aantal instelbare minuten nadat de trigger gegeven is. De instelbare tijd na de trigger wordt de post-trigger time genoemd. Dus met andere woorden, de tijdspanne van de gelogde data van een trigger bestaat uit de pretrigger time + de posttrigger time.

Tijdens het loggen worden van zowel de video als van de CAN data, timestamps bijgehouden. Zo is het mogelijk om nadien in de "Viewer" te bepalen welke videoframe er bij welke CAN-data hoort. Op die manier kan men video met CAN synchroniseren.

Dus samengevat: wanneer de chauffeur een fout in het te testen systeem detecteert, duwt hij op de hardware-knop, waarna de logger de data begint te loggen op een wijze die hierboven beknopt beschreven staat (met pre –en posttrigger).

Alle gelogde data komen op de externe HD te staan onder de folder: \Logs\. Voor elke trigger wordt een nieuwe map aangemaakt die als naam het formaat heeft van <Trigger_"jaar"-"maand"-"dag"_"uur"-"min"-"microseconden">. Het volledige pad is bijvoorbeeld \Logs\Trigger_2008-02-28_14-20-36.375.

In elke triggermap komen 3 bestanden te staan:

- Een videobestand met de extensie .avi
- Een bestand dat een tijdstip bijhoudt van de start van elk videoframe. Men noemt dit de timestamp file met de extensie .ts
- Een bestand waarin alle data van de CAN-bus vervat zit. (extensie .CANlog)

Trigger_2008-02-28_14-20-36.375.avi

Trigger_2008-02-28_14-20-36.375.ts

Trigger_2008-02-28_14-20-36.375.CANlog

Figuur 31: Voorbeeld van de triggerbestanden in een triggermap

Men hoeft voor het CANlog bestand geen aparte timestamp file bij te houden, omdat de timing bij in het CANlog bestand zit. Een CANlog bestand wordt in een binair formaat opgeslagen op de harde schijf.

Als de chauffeur zijn vrachtwagen afzet, detecteert de logger dit en zorgt het programma ervoor dat de embedded PC correct afgesloten wordt.

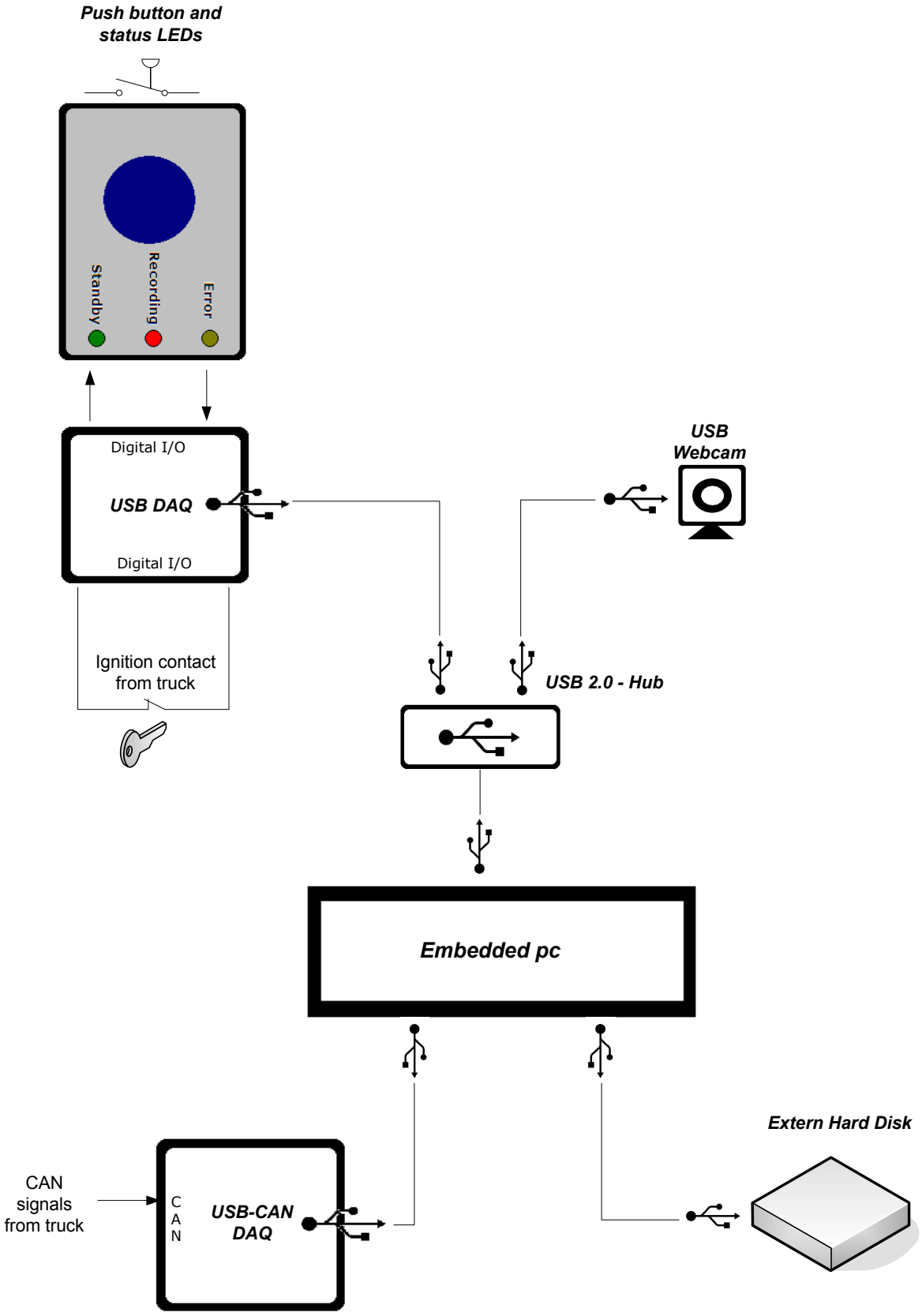
Na de proefrit wordt de externe HD uit de vrachtwagen gehaald en aangesloten op een gewone PC waar de Viewer op geïnstalleerd staat. De Viewer herkent alle triggers op de externe HD en de testingenieur kan beginnen met zijn analyse.

3.3.2 De Viewer

Zoals reeds vermeld, wordt (wanneer de vrachtwagen teruggekeerd is van zijn test) de externe harde schijf uit de vrachtwagen genomen en aangesloten op een computer waar de Viewer is geïnstalleerd.

In de Viewer zal men moeten specificeren waar het experiment, dat men wil analyseren, zich bevindt. In dit geval staat het dus op de externe harde schijf. Voor archief doeleinden kan men het ganse experiment, dat bestaat uit gelogde data en configuratie bestanden, verplaatsen naar een meer geschikte locatie.

Eenmaal het experiment gekozen, detecteert de Viewer alle triggers, waarna men kan selecteren welke trigger men van naderbij wil bekijken. De Viewer heeft een grafiek en een videovenster. Zo kan men CAN en video gesynchroniseerd bekijken. Een "play" functie mag natuurlijk ook niet ontbreken. Bovendien bestaat de mogelijkheid om een tweede scherm met het videobeeld op te roepen. Op deze manier is het mogelijk om met een “dual screen setup” te werken, dewelke het analyseren vereenvoudigt.



Figuur 32: Schets opstelling hardware van logger

4 ENKELE LABVIEW BEGRIPPEN

LabVIEW betekent voluit "Laboratory Virtual Instrument Engineering Workbench" en is een grafische programmeertaal die ontwikkeld is door National Instruments. LabVIEW is o.a. zeer geschikt voor communicatie met hardware (DAQ toepassingen).

4.1 Het vi

Een vi of "Virtual Instrument" is het basisonderdeel van de LabVIEW programmeertaal. Men kan zelf vi's aanmaken of men kan standaard vi's van de LabVIEW omgeving gebruiken. Men kan ook extra toolkits installeren die vi's voor een bepaalde functie of type hardware bevatten.

Een vi bestaat uit 3 onderdelen

- Frontpaneel
- Blokdiagram
- Connectorvlak

Men kan in een vi een ander vi oproepen, het opgeroepen vi noemt men een subvi. Men kan de werking van een vi nog het best omschrijven als een functie in c++.

4.2 Het frontpaneel

Het frontpaneel verzorgt de grafische interface met de gebruiker. Op het frontpaneel kunnen controls geplaatst worden. Deze controls zijn bijvoorbeeld LED's, lijsten, verschillende soorten knoppen, tekstvakken, grafieken,… De controls kan men selecteren in het "Controls palette" (Figuur 33). Dit palette kan men altijd oproepen door met de rechtermuisknop in het frontpaneel te klikken.

In Figuur 52 en Figuur 69 kan men respectievelijk het frontpaneel van de logger en de Viewer bekijken.

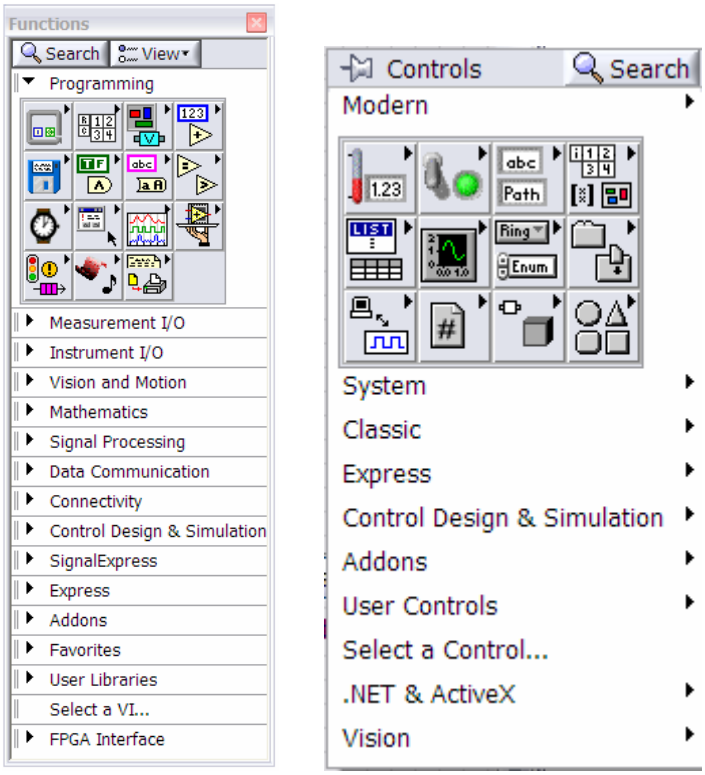
4.3 Het blokdiagram

In het blokdiagram komt de code te staan. Elk vi heeft zijn eigen frontpaneel en blokdiagram. De controls die men op het frontpaneel plaatst, zullen ook in codevorm beschikbaar zijn in het blokdiagram. Op die manier kan men via het blokdiagram de controls op het frontpaneel aansturen.

De verschillende soorten datatypes worden elk weergegeven door hun eigen specifieke kleur. Het datatype boolean heeft bijvoorbeeld een groene kleur.

Men verbindt de verschillende controls of vi's met een draad die de kleur krijgt van het datatype dat hij "overbrengt". In Figuur 58 kan men een blokdiagram met code bekijken.

De verschillende vi's die beschikbaar zijn in de LabVIEW omgeving kan men kiezen in het "Functions palette" (Figuur 33). Dit palette kan men altijd oproepen door met de rechtermuisknop in het blokdiagram te klikken.



Figuur 33: Functions palette en controls palette

4.4 Het connectorvlak

Het connectorvlak bevindt zich in de rechterbovenhoek van het frontpaneel of het blokdiagram. Het connectorvlak wordt gebruikt om de controls (of indicators) te koppelen aan de in – of uitgangen van het vi. In het connectorvlak kan men tevens het icoon dat de vi voorstelt aanpassen. Als het vi als subvi functioneert, dan kan men het herkennen aan het icoon.

Voorbeelden van zelfgemaakte iconen kan men bekijken in Figuur 53.

4.5 De GLI

Waneer men een globale variabele gebruikt, kunnen er race condities ontstaan.

Deze condities kunnen ontstaan wanneer verschillende parallelle processen dezelfde globale variabele gaan uitlezen en aanpassen.

Eerst leest men de data in de globale variabele, daarna voert men er een bewerking op uit. De uitkomst van deze bewerking schrijft men terug naar diezelfde globale variabele. Wanneer 2 parallelle processen dit principe op ongeveer hetzelfde tijdstip gaan toepassen, kan het gebeuren dat de data in de globale variabele niet meer de verwachte of correcte waarde zal bevatten.

Bijvoorbeeld 2 parallelle processen:

- Proces 1 vermeerderd de globale variabele met 1.
- Proces 2 verminderd de globale variabele met 1.
- We nemen de waarde van de globale variabele 5.

Indien proces 2 uitgevoerd wordt na het beëindigen van proces 1, wordt de waarde van de globale variabele 5 (= 5 +1 – 1). Dit is de correcte waarde.

Wanneer proces 2 uitgevoerd wordt ongeveer tijdens proces 1, dan zal de globale variabele een verkeerde finale waarde krijgen. Proces 1 leest de waarde 5 van de globale variabele, proces 2 doet dit op ongeveer dezelfde tijd. Elk proces gaat met de waarde 5 aan de slag, dit is de oorzaak van een race conditie. Het finale resultaat van de globale variabele zal nu 4 (5-1) of 6 (5+1) zijn, afhankelijk van het proces dat als laatste zijn uitkomst weggeschreven heeft naar de globale variabele. Dit is een onjuiste waarde.

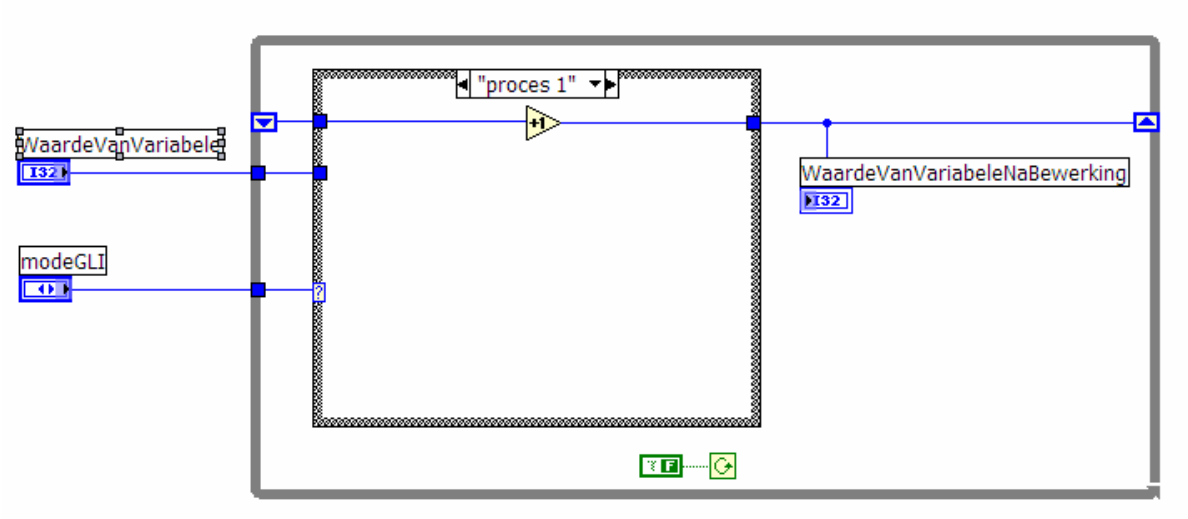
De oplossing voor dit probleem is een GLI is een "Global variable with Local Intelligence". De verschillen met een gewone globale variabele zijn:

- De GLI kan uitsluitend door 1 proces tegelijk aangesproken worden.
- De GLI bevat intelligentie (of code) die bewerkingen op de interne data kan uitvoeren.

Proces 1 en 2 uit het vorige voorbeeld zullen nu intern in de GLI plaatsvinden. Men kan kiezen welk proces er uitgevoerd wordt door de mode van de GLI te kiezen. Het is belangrijk om te weten dat er maar 1 mode tegelijk uitgevoerd kan worden.

Heeft men nu 2 parallelle processen dan zal in elk proces de GLI met de juiste mode opgeroepen worden. Stel nu dat de 2 processen de GLI tegelijk oproepen (ongeacht de gekozen mode). Het proces dat eerst was met het oproepen van de GLI, zal voorrang krijgen. Het andere proces zal dus noodgedwongen moeten wachten totdat het eerste voltooid is (het heeft de "race" verloren). Op die manier gebeurt de uitvoering serieel, wat ervoor zorgt dat er geen race condities kunnen optreden.

In LabVIEW wordt de werking van GLI gesimuleerd door een aparte subvi. Een subvi kan standaard maar door 1 ander vi opgeroepen worden, dus aan die voorwaarde is reeds voldaan. In die subvi zet men een while-lus die bij het oproepen van het subvi 1 keer uitgevoerd wordt. De data wordt bijgehouden in de shift register(s) van de while-lus. Wanneer men het subvi een volgende keer gaat oproepen, dan zullen de shift register(s) dezelfde data sinds de vorige aanroep bevatten. De verschillende modes worden bepaald aan de hand van een "case structure".



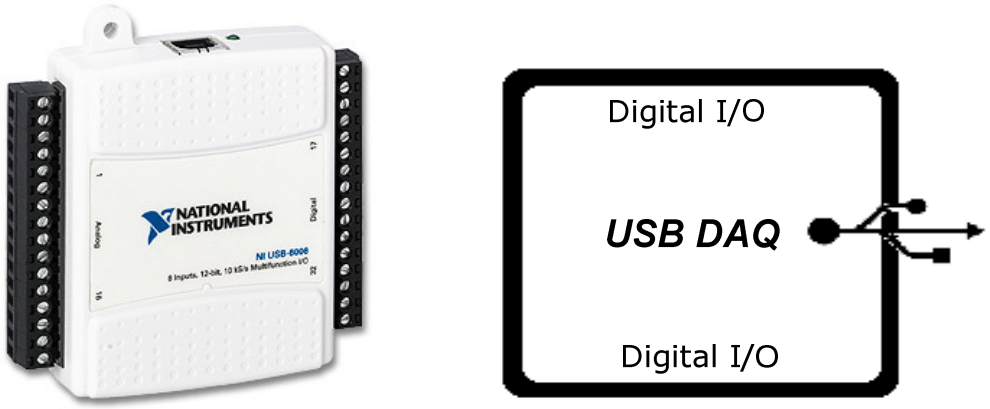
Figuur 34: GLI

In Figuur 34 kan men de code voor een GLI bekijken. Proces 1 gaat de waarde van het shift register met 1 vermeerderen en terug wegschrijven. De "WaardeVanVariabele" is tijdens een andere case in het shift register weggeschreven.

5 HARDWARE VAN DE LOGGER

5.1 De USB-DAQ module

Voor de USB-DAQ module is geopteerd voor een NI USB 6008 die verkrijgbaar is bij National Instruments. In Figuur 35 ziet u de module samen met zijn schematische voorstelling, zoals die gebruikt is in Figuur 32.

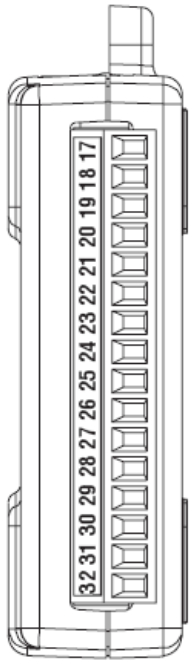


Figuur 35: NI USB 6008

De integratie van deze module in LabVIEW wordt verzorgt door de DAQmx toolkit.

Figuur 36: DAQmx toolkit

Deze module heeft zowel analoge als digitale in –en uitgangen. Voor dit project zijn alleen de digitale in –en uitgangen gebruikt. De betekenis van de verschillende pinnen ziet u in Figuur 37. Het is mogelijk om softwarematig (met behulp van de DAQmx toolkit) in te stellen of een bepaalde pin als ingang of als uitgang moet functioneren.

Module	Terminal	Signal
	17	P0.0
	18	P0.1
	19	P0.2
	20	P0.3
	21	P0.4
	22	P0.5
	23	P0 6
	24	P0.7
	25	P1.0
	26	P1.1
	27	P1.2
	28	P1.3
	29	PFI 0
	30	+2.5 V
	31	+5 V
	32	GND

Figuur 37: Digitale in - en uitgangen NI USB 6008

Vanuit deze module vertrekken 6 verbindingen naar het bakje met de drukknop en de statusLED's. Deze 6 verbindingen zijn:

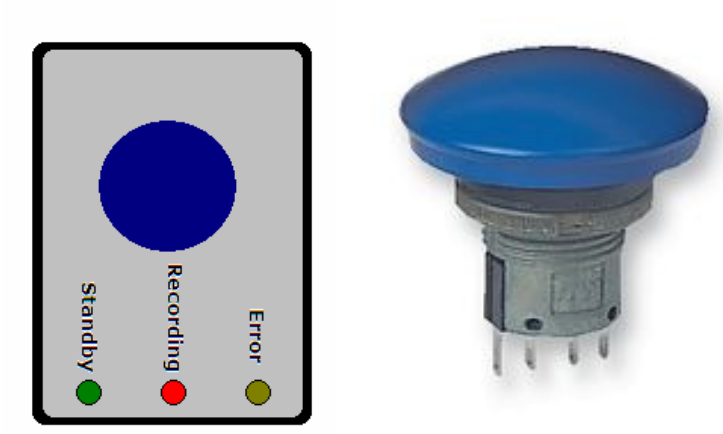
- 3 digitale uitgangen
- 1 digitale ingang
- Massa
- +5 V

Er is ook 1 digitale ingang gebruikt om het contact van de vrachtwagen te monitoren.

Deze verbindingen worden nader toegelicht in punt 5.2.

In de bijlage kan u een datasheet van dit apparaat terugvinden.

5.2 De drukknop met de statusLED’s



Figuur 38: Drukknop + statusLED’s

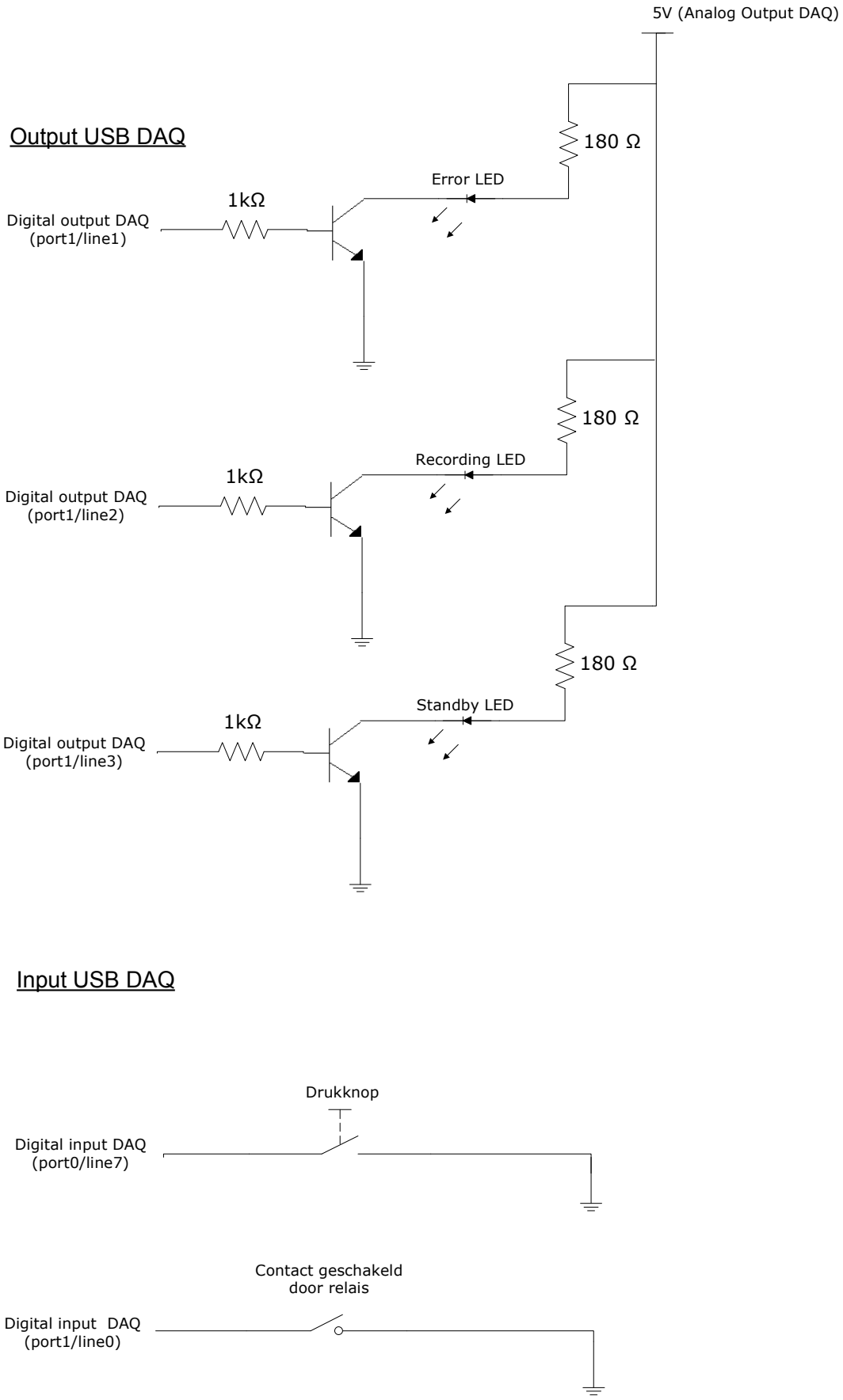
De drukknop is samen met de statusLED’s in een plastieken behuizing ondergebracht.

De behuizing ontvangt van de USB-DAQ module 6 verbindingen. Deze 6 verbindingen + de aansluiting van het contact van de vrachtwagen worden gegeven in Figuur 39.

Omdat LED’s gemiddeld 20mA (is afhankelijk van hun kleur) nodig hebben zijn ze niet rechtstreeks aan hun respectievelijke digitale ingang gekoppeld, deze levert immers maar 0.6mA. Om de LED’s van voldoende stroom te voorzien wordt er gebruikt gemaakt van de +5V die beschikbaar is op de USB-DAQ (Figuur 37, pin 31). Er is gekozen voor een weerstand van 180Ω, waardoor de stroom door de LED’s ongeveer 27 mA (5V/180Ω) bedraagt. Dit is meer dan voldoende om ze duidelijk te laten oplichten. De NPN-transistor functioneert hier als schakelaar en gaat “schakelen” als hij een signaal krijgt van de digitale uitgang die aan zijn basis gekoppeld is.

Omdat een digitale ingang intern 5V bedraagt als hij niet aangesloten is, wordt de drukknop aan de massa gehangen. Zodra de knop ingedrukt wordt zal de digitale ingang naar 0V getrokken worden. Zo weet de module dat er een overgang van een logische 1 naar een logische 0 geweest is. De knop is van het merk Farnell en kan zowel als normaal open (NO) en normaal gesloten (NC) aangesloten worden. Hier is hij NO aangesloten. In de bijlage kan men de datasheet van deze knop terugvinden.

Het laatste contact in Figuur 39 werkt op dezelfde manier als de drukknop maar hier wordt de schakelaar door een relais bedient. Deze relais schakelt als het contact van de vrachtwagen “aan” staat.



Figuur 39: schema van de drukknop + statusLED's + contactaansluiting

5.3 De embedded PC

De embedded PC moet aan de volgende voorwaarden voldoen:

- Schokbestending
- Speciale voeding
- Voldoende krachtige processor
- USB2.0 aansluitingen

De voeding moet overweg kunnen met het grote spanningsverschil dat de batterij van de vrachtwagen levert. Vooral bij het starten is er een grote stroom nodig waardoor de spanning even gaat dalen. De voeding van de gekozen PC kan overweg met een spanning die ligt tussen 6..36 V. De PC wordt rechtstreeks gevoed door de batterij, ook wanneer de motor van de vrachtwagen niet draait of uitgeschakeld wordt. Op die manier kan de PC correct afgesloten worden. Als de voeding detecteert dat het contact van de vrachtwagen wordt afgesloten, dan bestaat de mogelijkheid om de PC automatisch te laten uitvallen na een instelbare tijd. Deze tijd wordt ingesteld door intern jumpers op de juiste plaats te zetten, in dit project is er gekozen voor een afsluittijd van 5 minuten. Op deze manier valt echter de voeding plotseling uit, waardoor Windows en het programma van de logger niet correct afgesloten worden. Omdat dit kan leiden tot corrupte logbestanden is er gekozen om de PC softwarematig te laten afsluiten (nadat alle logbestanden weggeschreven zijn). Door de reeds besproken aansluiting van het contact van de vrachtwagen aan de NI USB 6008 (Figuur 39), is het mogelijk om softwarematig te detecteren wanneer het contact uitgeschakeld wordt. De voeding detecteert ook wanneer het contact van de vrachtwagen ingeschakeld wordt, wat er voor zorgt dat de PC mee gaat opstarten.

De gekozen PC, is de VTC 3300 (Figuur 40). Deze PC heeft een sterke behuizing en is schokbestendig gemonteerd. Omdat er gekozen is voor een Intel Pentium M 1.8 Ghz (niet standaard bij de VTC 3300) als processor, werden er onderaan 2 extra fans meegeleverd/gemonteerd. De processor bleek krachtig genoeg om de logger correct te laten functioneren. Het testsysteem was namelijk uitgerust met dezelfde processor. Deze cpu kenmerkt zich door het lage verbruik (3-25 W) en is daarom geschikt voor gebruik in laptops of embedded computers, de M van Pentium M staat immers voor "mobile".

Er is gekozen voor 1 gigabyte aan SO-DIMM DDR266 RAM. Tijdens tests is gebleken dat ongeveer 30% van het geheugen gebruikt werd. De voornaamste slokop van het geheugen was de softwarematige circulaire buffer van de videobeelden, waarover later meer.

De interne harde schijf is van het type 2.5" (voornamelijk gebruikt in laptops) en is 80 gigabyte groot. Dit is meer dan voldoende voor onze interne buffer. De opslagsnelheid van de data van de logger is ongeveer 21 MB/min, dus voor een pretrigger tijd van 60 seconden, zal de buffer ongeveer 21MB groot zijn. De buffer-bestanden die ouder zijn dan pretrigger tijd worden verwijderd, behalve als ze nog in gebruikt zijn tijdens het editeren van een trigger (waarover later meer). Hierdoor kan de buffer tijdelijk meer dubbel zo groot worden (afhankelijk van de posttrigger tijd), maar met 80 GB zal de interne harde schijf nooit volledig gevuld kunnen raken met de buffer. Zelfs met een pretrigger –en posttrigger tijd van 50 min zal de buffer op zijn maximum maar iets groter zijn dan 2100MB (21 x 50 x 2). De testingenieurs bij DAF Eindhoven zijn alleen geïnteresseerd in een pre en posttrigger tijd die niet groter is dan 10 minuten. De grootte van het videogedeelte van de buffer hangt natuurlijk af van de gebruikte codec (zie 6.1).

Figuur 40: VTC 3300 + schematische voorstelling

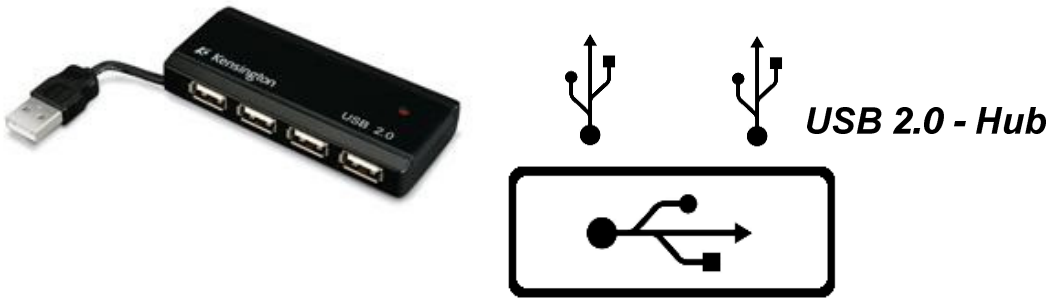
Het operating systeem is een volledige up to date Microsoft Windows XP. Dit OS is al verscheidene jaren op de markt en is zeer stabiel.

De VTC 3300 heeft standaard 3 USB 2.0 aansluitingen. Omdat dit niet voldoende is voor onze applicatie, is er een extra USB-Hub gebruikt. Voor meer informatie over de specificaties, kan u in de bijlage een datasheet vinden over de VTC 3300.

5.4 De USB-Hub

De 2 voorwaarden waaraan een USB-Hub voor ons project moet voldoen zijn

- Ondersteuning voor USB 2.0.
- Voeding via USB-Bus (Buspowered).



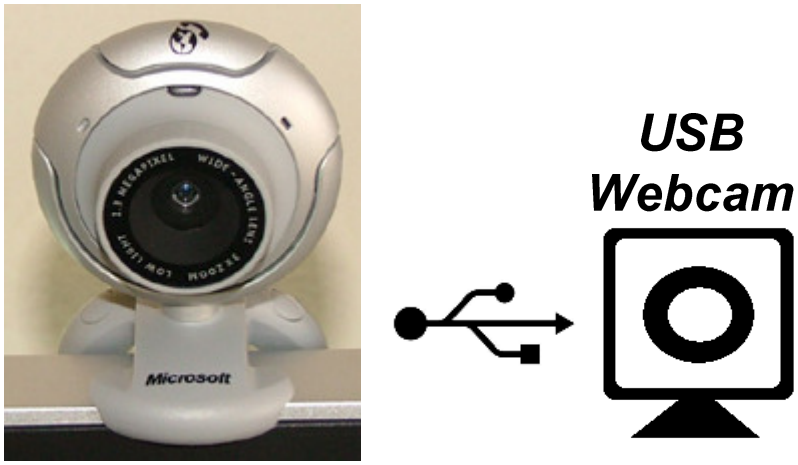
Figuur 41: Kensington K33399 + schematische voorstelling

Er is gekozen voor een Kensington K33399. Figuur 41 toont hem en zijn schematische voorstelling zoals die gebruikt is in de algemene schets van het totale project (Figuur 32).

5.5 De USB-webcam

De voorwaarden voor een webcam zijn:

- Minimum 10 frames per seconde.
- Elk frame moet minstens een resolutie hebben van 640×480 (0.3 Megapixels).
- Kijkhoek die breed genoeg is.



Figuur 42: Microsoft LifeCam VX-6000+ schematische voorstelling

De Microsoft LifeCam VX-6000 voldoet aan al die eisen. Hij kan 30 frames per seconde afleveren en dit aan een maximum van 1.3 megapixels per frame. Zijn kijkhoek is 71°, wat voldoende is voor onze toepassing. Deze webcam heeft een USB 2.0 aansluiting. Aan 30 fps en een resolutie van 640x480 (RGB 24 mode is 3 bytes) verstuurd de webcam zijn data aan ongeveer 27,648 MB/s (640x480x3x30). USB 2.0 heeft een maximale datasnelheid van 480 Mbps (= 60 MB/s).

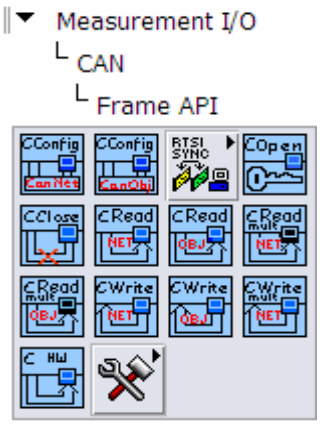
Men plaatst de webcam afhankelijk van het te testen systeem. Voor de meeste systemen zal hij het verkeer voor de vrachtwagen filmen, maar het is evengoed mogelijk om de webcam op iets anders te richten.

5.6 De USB-CAN DAQ module

Voor de USB-CAN DAQ kaart is gekozen voor een NI USB-8473 die verkrijgbaar is bij National Instruments. In Figuur 43 ziet u de module samen met zijn schematische voorstelling, zoals die gebruikt is in Figuur 32. Deze module kan werken met 11 en 29 bit "identifiers" en hij is compatibel met de SAE J1939 standaard. Zijn maximale snelheid is 1Mbits/s. In dit project en bij DAF gebruiken ze de SAE J1939 standaard die aan 250 kbit/s opereert. De module wordt gevoed via de USB-bus. De datasheet van deze module is te vinden bij de bijlagen.

Figuur 43: NI USB-8473 + schematische voorstelling

Voor de communicatie met de LabVIEW is het noodzakelijk een driver en de "Frame API" te installeren. Beide worden ter beschikking gesteld door National Instruments. De "Frame API" is een verzameling van vi's die de communicatie met de module verzorgen. De belangrijkste vi voor dit project is diegene die alle frames van de CAN-bus kan binnenlezen in LabVIEW. De "Frame API" biedt ook de mogelijkheid om zelf frames op de CAN-bus te zetten, maar deze functionaliteit is niet gebruikt in dit project. In Figuur 44 ziet u de vi's van de Frame API.

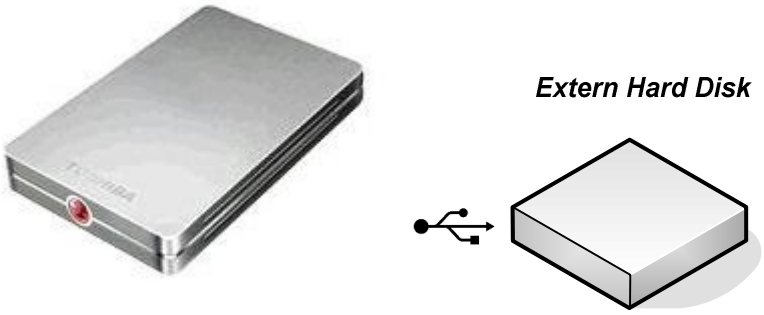


Figuur 44: De Frame API

5.7 De externe harde schijf

De 3 voorwaarden aan dewelke de externe HD moet voldoen zijn:

- Voldoende capaciteit
- Voeding via USB-bus
- USB 2.0



Figuur 45: Externe HD van Toshiba + schematische voorstelling

De 200 GB externe USB mini harde schijf van Toshiba voldeed aan onze voorwaarden.

Met een pre en posttrigger tijd van bijvoorbeeld 30 seconden, kunnen er aan 21MB/min ongeveer 9500 (200000/21) verschillende triggers opgeslagen worden. Elke trigger bestaat uit 3 bestanden (Video, CANlog, timestamp, zie 3.3.1.2).

Het formaat van de HD is van het type 2.5” (mini) en ze heeft een snelheid van 4200 toeren per minuut.

Deze HD wordt gevoed door de USB-bus.

6 DE LOGGER SOFTWARE

6.1 De codec

Omdat er met videobeelden gewerkt wordt, is het nodig de beelden te comprimeren om plaats te besparen.

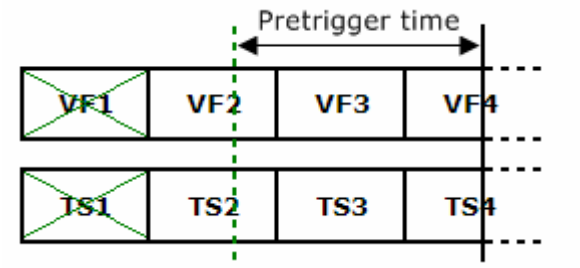
De gebruikte codec is de "LEAD MCMP/MJPEG Video Codec". Hij behoort tot de groep van de MJPEG codecs, MJPEG staat voor Motion JPEG. Dit soort codecs coderen per beeld, ze bekijken de video als een reeks stilstaande beelden. Dit in tegenstelling tot de MPEG codecs. Zij gebruiken, voor het coderen van het huidige beeld, informatie uit de vorige en de volgende beelden. Op deze manier zal het coderen afhangen van de bewegingen die in de video plaatsvinden. Voor het editeren van videobestanden is een MPEG codec niet geschikt omdat hij voor een bepaalde frame altijd informatie nodig heeft uit andere frames, daardoor kan het zijn dat bij het knippen van een videobestand, informatie verloren gaat. MCMP, of Motion CMP is een speciale techniek die LEAD gebruikt om gecodeerde videobestanden nog kleiner te maken en met een betere beeldkwaliteit dan bij een andere MJPEG codec.

6.2 Principe

Er wordt continu een buffer bijgehouden op de interne harde schijf van de embedded PC. Deze buffer bestaat uit:

- Blokken van video bestanden die telkens 200 frames bevatten. Er wordt gelogd aan 10 frames per seconde, dus de blokken zijn 20 seconden lang. Zo een blok wordt vanaf nu aangeduid met "VF" (VideoFile).
- Per VF wordt er ook een timestamp bestand bijgehouden (TS) waarin het tijdstip van het begin van elk frame van de corresponderende VF wordt bijgehouden. Deze bestanden zijn nodig voor het editeren.
- Dit geldt analoog voor de CAN frames. Een CAN blok bestaat uit 10 CAN-frames.
- Per CAN blok wordt er ook een TS bijgehouden, analoog aan de videoTS.

Het principe van de logger wordt uitgelegd aan de hand van de VF en de bijhorende TS bestanden. Uiteraard is dit principe analoog voor de CAN data en zijn TS bestanden. In de logger zal het proces van de CAN data parallel met het proces van de video draaien.

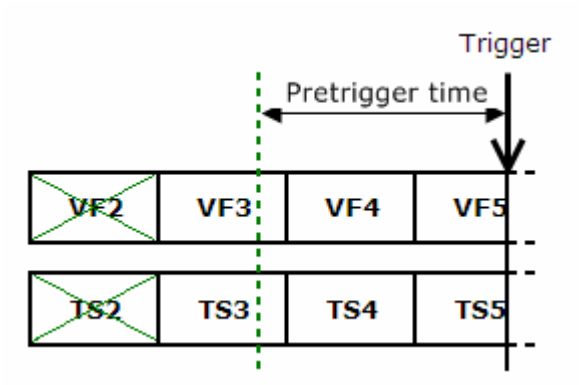


Figuur 46: Werking interne buffer zonder trigger

In Figuur 46 ziet u de werking van de interne buffer zonder dat er een trigger gedetecteerd is.

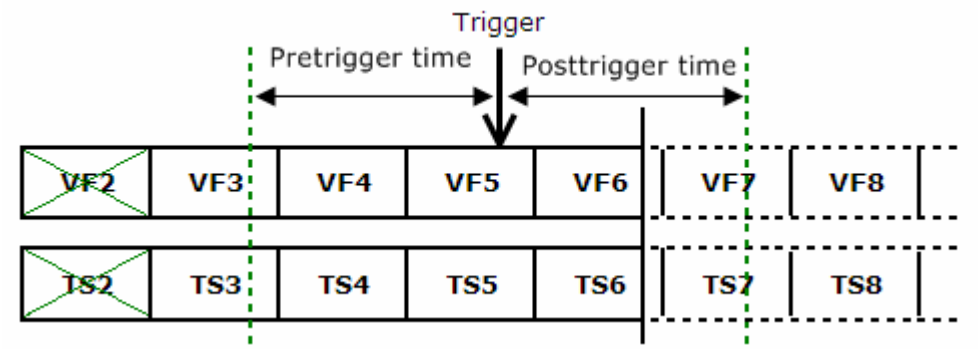
De volle verticale zwarte lijn duidt het huidige tijdstip aan. De groene verticale stippellijn duidt in deze figuur het begin aan van de pretrigger (= huidige tijd – pretrigger tijd).In de software wordt continu bijgehouden wanneer welk bestand werd aangemaakt. Alle bestanden ouder dan de huidige tijd – pretrigger tijd worden verwijderd, in dit geval zijn dat VF1 en TS1. Als er geen trigger wordt gedetecteerd, blijft de interne buffer een grootte hebben van 3 blokken, op het tijdstip van Figuur 46 zijn dat VF2, VF3 en VF4 + hun corresponderende TS bestanden.

In Figuur 47 wordt er een trigger gedetecteerd. Op dit moment wordt een pre en posttrigger tijd, die bij deze trigger horen, "bewaart" in de software. VF2 en TS2 worden verwijderd, omdat deze ouder zijn dan de oudste pretrigger. Er kunnen meerdere triggers na mekaar gegeven worden, daarom is het belangrijk dat men alleen de bestanden verwijdert die ouder zijn dan de oudste pretrigger. In dit voorbeeld is er voor de duidelijkheid maar 1 trigger.



Figuur 47: Werking interne buffer met trigger (1)

In Figuur 48 zit men een beetje verder in de tijd, namelijk tussen de trigger en zijn posttrigger. Op dit moment wordt blok VF6 en zijn TS6 aangemaakt en er zijn geen bijkomende bestanden verwijderd. De software gaat nu wachten totdat de posttrigger tijd is afgelopen.

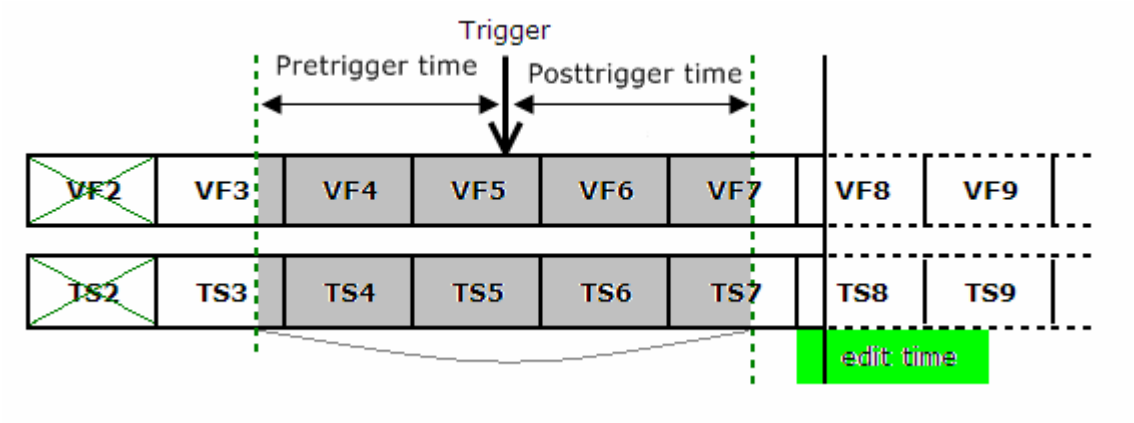


Figuur 48: Werking interne buffer met trigger (2)

Als de posttrigger tijd afgelopen is en als de blokken die nodig zijn voor het editeren correct aangemaakt zijn in de interne buffer, dan begint de software met het editeren (Figuur 49). Het is noodzakelijk dat alle blokken die nodig zijn, correct afgesloten worden door het proces dat ze aanmaakt. Als dit niet het geval is, zal het editeren niet

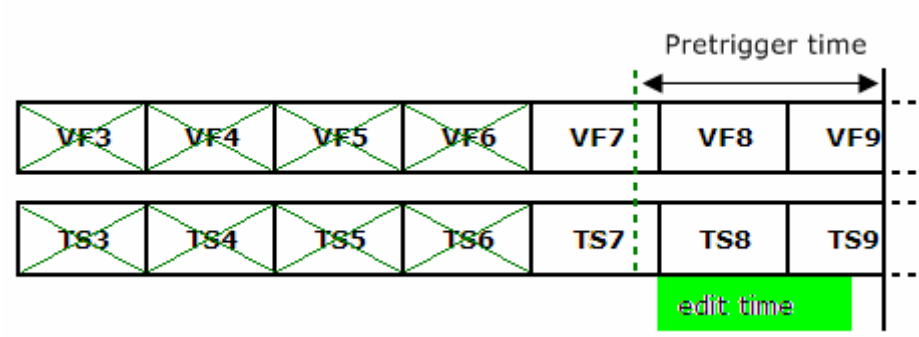
kunnen voltooid worden, omdat een blok maar door 1 proces tegelijk kan behandeld worden. In dit voorbeeld kan het editeren alleen beginnen als blok VF7 is afgerond.

Met editeren wordt er bedoelt dat de software een videobestand (en bijhorende TS) gaat aanmaken die een lengte heeft van pretrigger + posttrigger tijd. De software zoekt de juiste frames uit de begin en eindblokken (VF3 en VF7). Samen met VF4, VF5 en VF6 vormt dit het videobestand dat op de externe schijf zal geplaatst worden (en dit onder de juiste trigger folder, zoals besproken in 3.3.1.2). In Figuur 49 is dit voorgesteld door een grijze zone. Men kan opmerken dat de interne buffer op deze moment meer dan dubbel zo groot is dan in normale werking (zonder trigger).



Figuur 49: Werking interne buffer met trigger (3)

De bestanden mogen maar verwijderd worden als het editeren gedaan is en als ze ouder zijn dan de oudste van alle bewaarde pretriggers. Dit laatstgenoemde is vooral belangrijk als er meerdere triggers na mekaar gegeven worden. Het editeren neemt tijd in beslag en dit is voorgesteld door de groene band "edit time".



Figuur 50: Werking interne buffer met trigger (4)

Als het editeren voltooid is, wordt de normale werking van de buffer hervat en de bestanden die niet meer nodig zijn (ouder dan oudste pretrigger), worden verwijderd (Figuur 50). Indien er een tweede trigger gedetecteerd werd, zal deze natuurlijk eerst afgehandeld worden volgens hetzelfde principe. Als deze tweede trigger bovendien kort na de eerste gegeven werd, dan zullen niet alle bestanden in Figuur 50 verwijderd worden, maar dit werd reeds hierboven besproken.

Dit principe is analoog voor de CAN blokken en hun bijhorende TS. Met de uitzondering dat bij het editeren geen CAN TS naar de externe harde schijf gekopieerd zal worden. De tijdstippen van de CAN frames zitten immers bij in de data van de CAN blokken. De vraag kan gesteld worden waarom het noodzakelijk is om überhaupt een TS bestand voor de CAN blokken bij te houden op de interne HD. Dit kan men verklaren met het feit dat er tijdens het editeren veel sneller naar de juiste frames kan gezocht worden in een TS bestand dan in een blok van CAN data.

6.3 Praktische werking van de software

De software van de logger is opgebouwd uit 4 parallele vi's (Figuur 51). Dit zijn

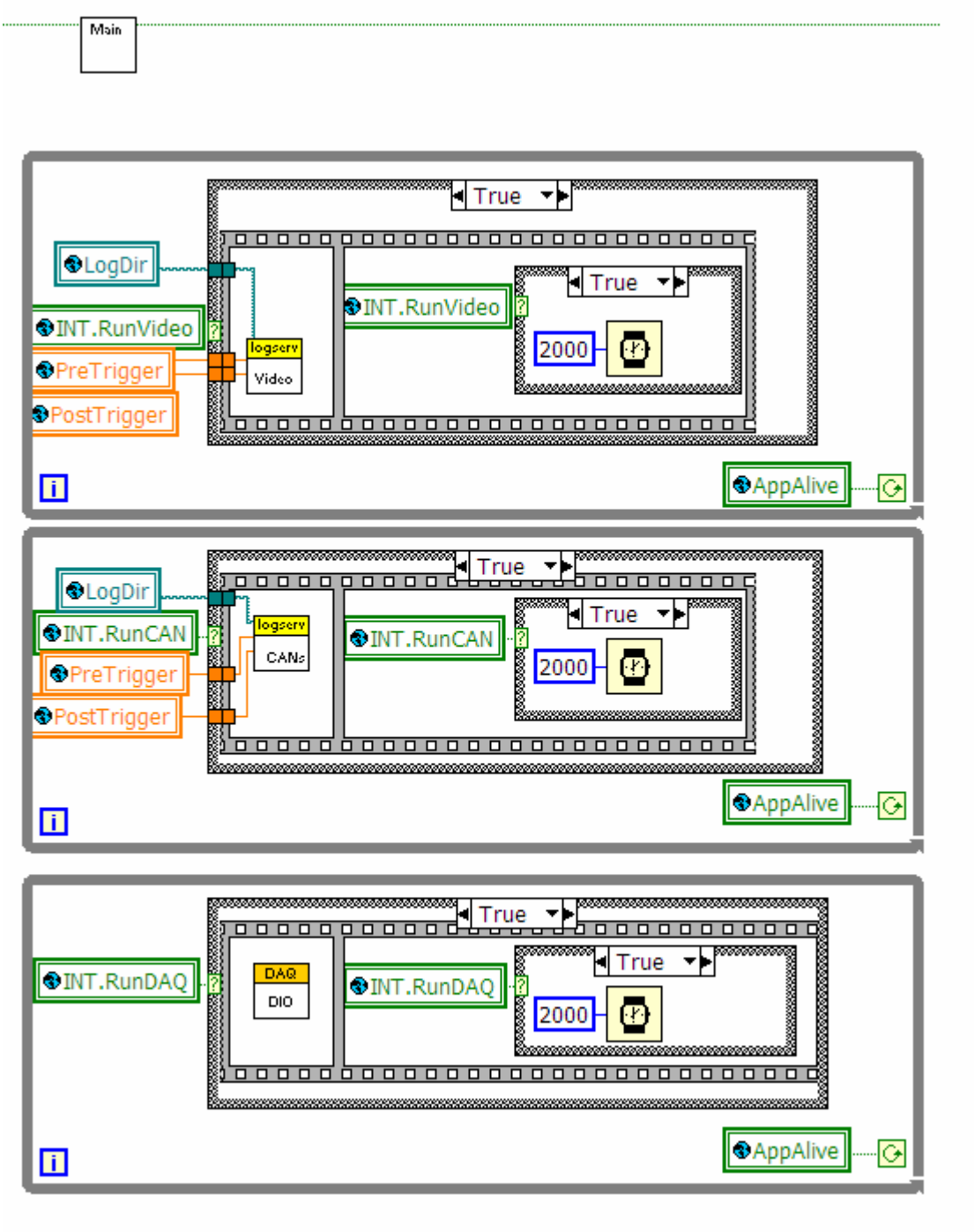
- Main
- Logserv Video
- Logserv CAN
- DAQ DIO (Data I/O)

Elke vi bestaat intern uit een lus. De externe lussen rond de 3 laatste vi's hebben als functie om bij een error het desbetreffende vi terug op te starten. Als er bijvoorbeeld in het vi "Logserv Video" een error optreedt waardoor het vi uit zijn interne lus wordt gehaald (en het vi dus beëindigt wordt), dan zal er 2 seconden gewacht worden (indien er ondertussen niet het sein gegeven is om de logger af te sluiten, in dat geval zal er niet gewacht worden), waarna er getracht wordt het vi terug op te starten.

6.3.1 Het "Main" vi

Het Main vi heeft de volgende functies:

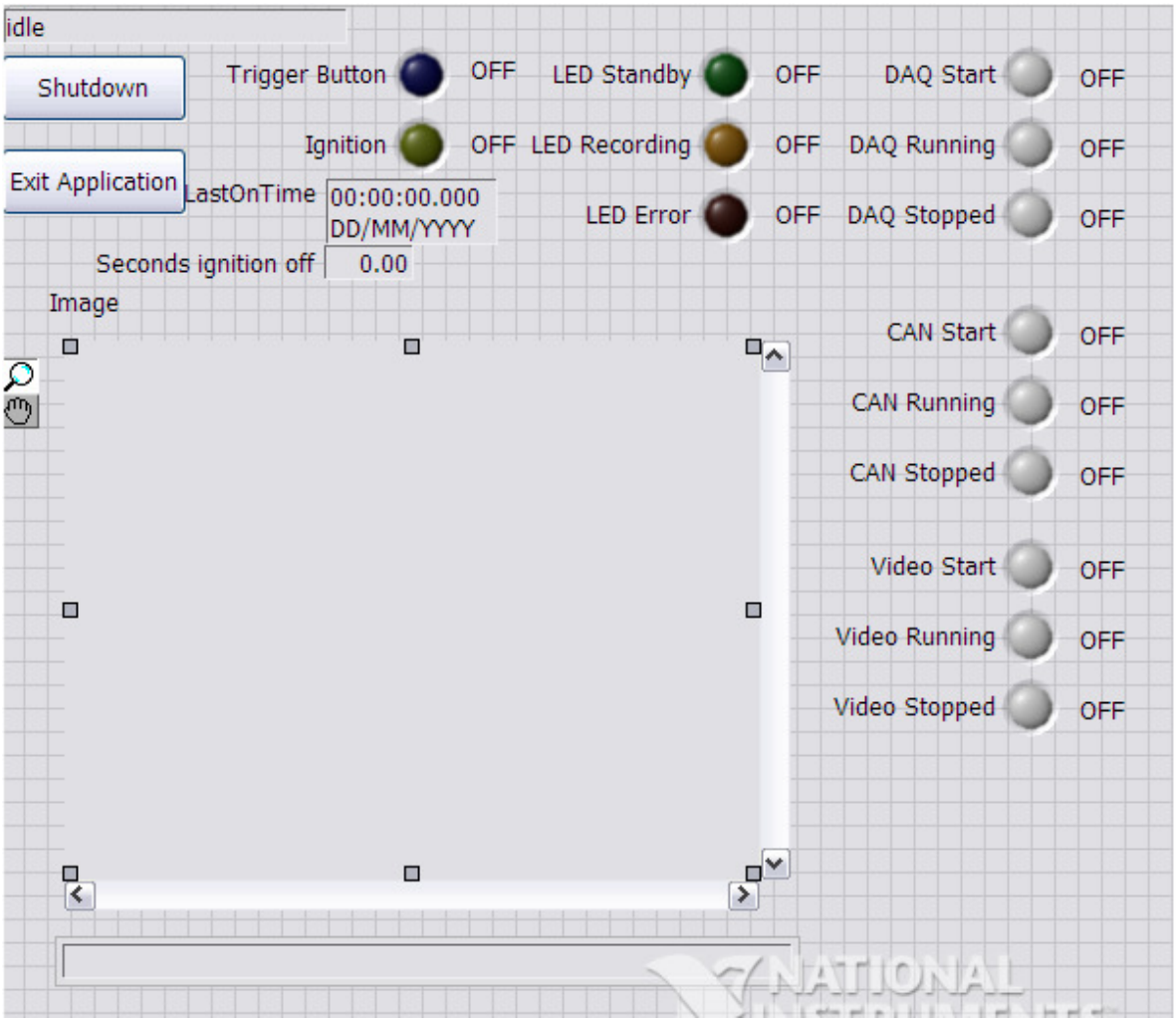
- De 3 andere vi's starten door hun desbetreffende globale variable (vb: INT.RunVideo) de waarde "true" te geven. Bij het opstarten wordt er ook gezocht naar het "DataLoggerConfiguration.experiment" bestand op de externe HD waarin de waardes voor de pre en posttrigger zijn bewaart.
- Continu checken wat de status van het contact is. (via een globale variabele, IN.Ignition, die in het DAQ DIO "vi" beheert wordt).
- Het correct afsluiten van de 4 parallelle vi's en daaruit volgend het gehele programma. Bij het afsluiten wordt de buffer op de interne harde schijf verwijderd.
- Dit vi stuurt de GUI van de logger.



Figuur 51: 4 parallelle processen/vi's

6.3.1.1 De GUI van de logger

In Figuur 52 kan u de GUI van de logger bekijken. De werking van de verschillende controls zal nu kort toegelicht worden.



Figuur 52: GUI van de logger

De "Shutdown" knop zorgt ervoor dat het programma en de embedded PC afgesloten worden. Het afsluiten van de embedded PC is intern (in het main vi) geregeld door LabVIEW het commando < shutdown -s -t 30 -c "Shutdown by LabVIEW" > naar de "command prompt" van Windows XP te laten sturen.

- "-s" staat voor een gewone shutdown.
- "-t 30" wil zeggen dat er na een timeout van 30 seconden mag afgesloten worden.
- -c "Shutdown by LabVIEW" zet een opmerking in het eventlog van Windows XP.

De "Exit Application" knop sluit alleen het programma af en niet de embedded PC.

In het grote grijze vlak "Image" komt het real-time beeld van de webcam te staan.

De resterende controls zijn allemaal LED's die gestuurd worden door globale variabelen, die op hun beurt van waardes (true of false) worden voorzien die afkomstig zijn van de processen in de 4 parallele vi's.

De "Trigger" LED licht op als er een signaal van de hardware-knop is ontvangen. De globale variabele die deze LED stuurt wordt van een waarde voorzien in het DIO vi.

De "Ignition" LED licht op indien het contact van de vrachtwagen ingeschakeld is. De globale variabele die deze LED stuurt wordt van een waarde voorzien in het DIO vi

De "Standby" LED licht op wanneer de logger volledig is opgestart en operationeel is.

De "Recording" LED is gedoofd wanneer er geen trigger gedetecteerd wordt. Indien er een trigger gedetecteerd wordt dan begint deze LED te knipperen. Het knipperen duurt net zolang totdat de posttrigger tijd overschreden is. Tijdens het editeren is deze LED continu opgelicht.

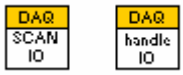
De "Error" LED begint te knipperen als "Logserv Video" of/en het "Logserv Video" vi onverwacht beëindigt is/zijn.

Deze 3 laatstvernoemde LED's vindt men ook terug op het bakje waar de knop is gemonteerd (zie punt 5.2). Hun werking is dezelfde.

De resterende LED's geven de status weer van de "Logserv Video", "Logserv Video" en "DAQ DIO" vi's.

6.3.2 Het DIO DAQ vi

De kern van dit vi bestaat intern uit 2 parallel uitgevoerde vi's/processen (Figuur 53).



Figuur 53: interne processen DIO DAQ vi

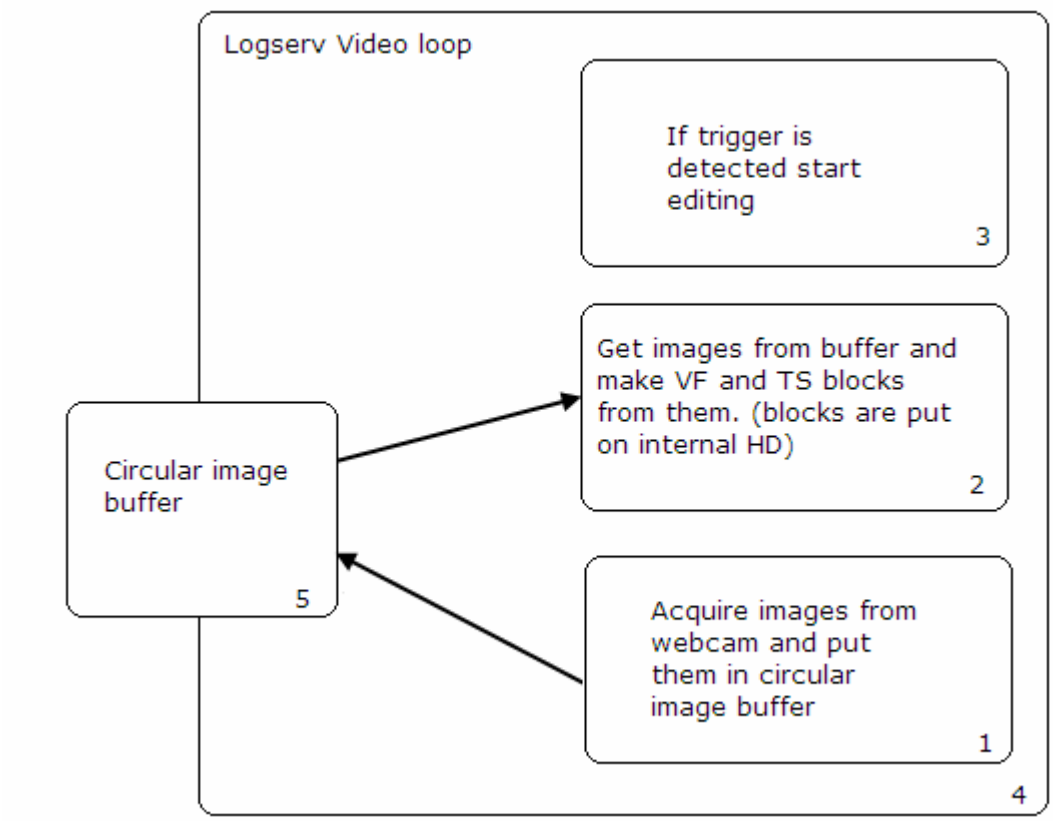
Het "SCAN IO" vi verzorgt de communicatie met de USB-DAQ module. De digitale in - en uitgangen van Figuur 39 worden continu gemonitored. Dit gebeurt met behulp van de DAQmx toolkit (Figuur 36). De communicatie met de rest van het programma gebeurt door middel van LabVIEW globale variabelen. Detecteert het "SCAN IO" vi bijvoorbeeld dat de hardware-knop ingedrukt is, dan zal de globale variabele "IN.button" "true" worden (zolang de knop ingedrukt blijft tenminste). Dit vi zal ook de 3 status LED's aansturen en het contact van de vrachtwagen monitoren.

Het "handle IO" vi checkt continu de globale variabele "IN.button". Indien deze variabele true wordt, dan zal het "handle IO" vi het tijdstip van die trigger in de triggerbuffer zetten. Deze buffer is een standaard LabVIEW FIFO buffer. De triggerbuffer wordt uitgelezen in de editeer-lussen van het "Logserv Video" vi en het "Logserv CAN" vi (zie 6.3.3 en 6.3.4).

6.3.3 Het "Logserv Video" vi

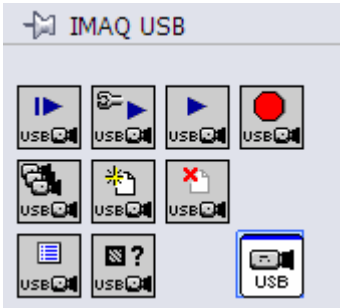
Dit vi gaat te werk volgens het principe uitgelegd in punt 6.2. Het doet dit met behulp van 3 interne parallelle processen/lussen. In Figuur 54 ziet u deze lussen. Blok 1,2,3,4 stellen lussen of loops voor en blok 5 stelt een buffer voor. Deze buffer is een GLI. Alle

lussen werken onafhankelijk en parallel van/aan elkaar. De data uitwisseling tussen lus 2 en lus 1 gebeurt via de buffer 5.



Figuur 54: Werking Logserv Video vi

Lus 1 gaat continu beelden van de webcam in een circulaire buffer wegschrijven. Deze buffer bestaat enkel in het geheugen (RAM) van de PC. Het binnenhalen van de beelden van de webcam gebeurt met de IMAQ toolkit van National Instruments, meer bepaald het onderdeel van de toolkit dat met USB webcams overweg kan. Een speciale NI-IMAQ USB webcam driver moet ook geïnstalleerd worden. In Figuur 55 ziet u de verschillende vi's.



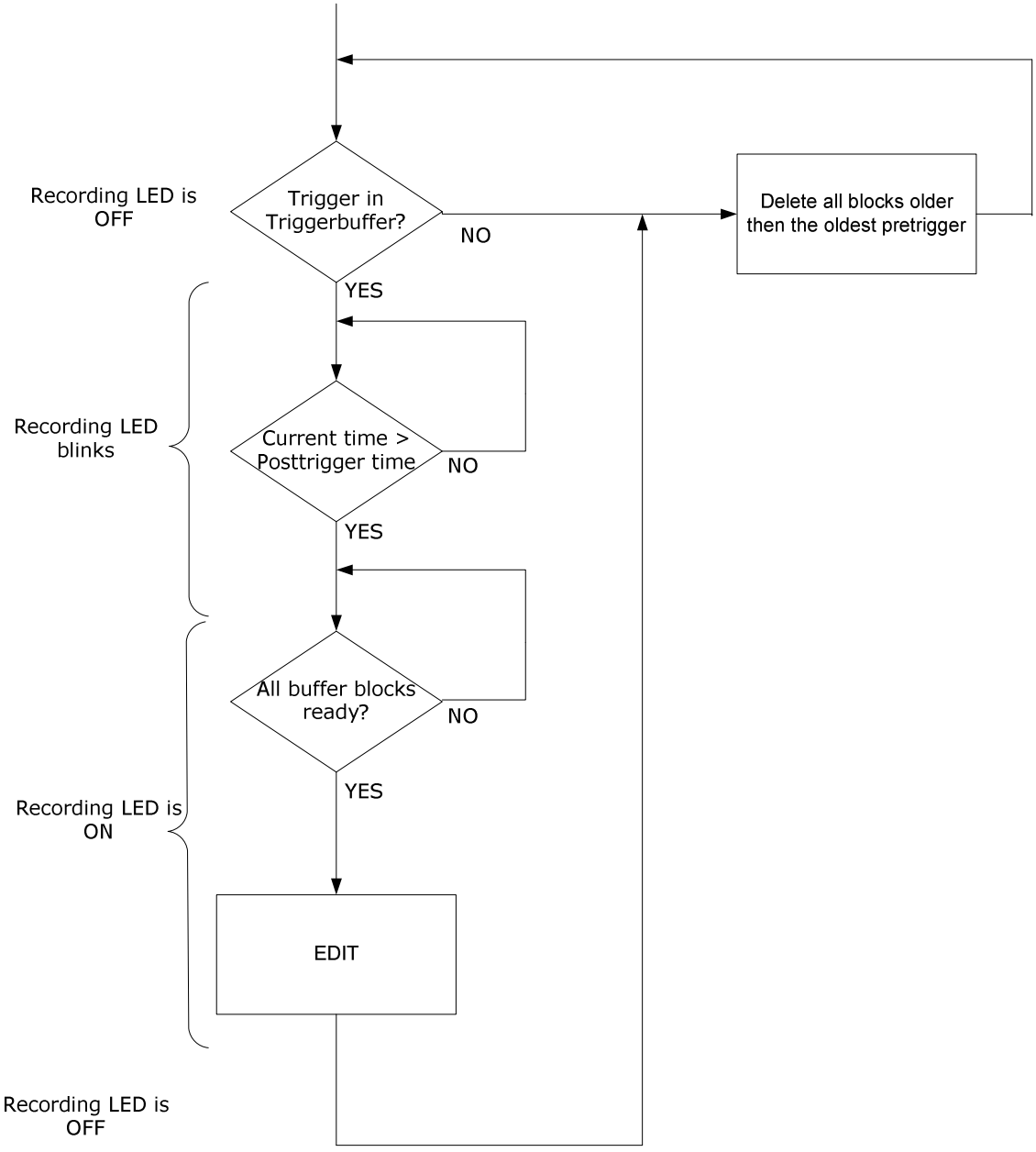
Figuur 55: NI IMAQ USB webcam toolkit

Er zijn vi's voor de verbinding met de webcam te openen of te sluiten. Een ander vi neemt een snapshot van 1 beeld van de webcam en nog een ander vi kan continu beelden van de webcam binnenhalen. Het is dit laatste vi wat er in deze lus gebruikt wordt. De webcam levert aan de PC 30 fps, maar niet alle 30fps worden gebruikt. Om

de 100 miliseconden gaat er een frame uit de 30fps genomen worden en opgeslagen worden in de circulaire buffer. Dit komt neer op een framerate van 10 fps.

Lus 2 gaat deze beelden uit de buffer halen en ze, samen met een TS, wegschrijven op de interne harde schijf. Dit wegschrijven gebeurt in blokken zoals uitgelegd in punt 6.2. Informatie over de blokken zoals het begintijdstip en positie op de harde schijf worden in de software bijgehouden in de GLI "file buffer".

Lus 3 is de edit loop en zijn werking is afgebeeld in Figuur 56.



Figuur 56: Lus 3 of "edit loop"

Als er door het DIO DAQ vi een trigger gedetecteerd wordt, zet hij die in een standaard LabVIEW FIFO buffer. De "edit loop" gaat deze buffer continu controleren op de aanwezigheid van een trigger. Als er geen trigger in de buffer zit, gaat deze "edit loop" alle bestanden ouder dan de oudste pretrigger deleten. Indien er een trigger in de

buffer aanwezig is, gaat hij wachten totdat de posttrigger van die trigger afgelopen is. Vervolgens gaat hij wachten tot alle benodigde blokken beschikbaar zijn in de buffer (op de interne harde schijf). Wanneer alle blokken klaar zijn, begint hij met editeren (zie 6.2). Dit proces gebruikt de informatie uit de "file buffer" GLI. Na het editeren begint het proces in Figuur 56 opnieuw. Voor een meer gedetailleerde uitleg van het principe, zie punt 6.2.

6.3.3.1 Circular image buffer GLI

Om geen dataverlies te hebben en om aan een constante 10 fps te kunnen blijven loggen, is er een buffer nodig tussen lus 1 en lus 2. Indien alleen de eerste 2 lussen actief zijn is er geen probleem, tenzij de HD eventjes niet meer kan volgen met het wegschrijven van data in lus 2. Wanneer nu alle lussen actief worden tijdens het editeren van een trigger, wordt de processor van de embedded PC sterk belast. Het kan op sommige momenten gebeuren dat hij niet tijdig alle data in lus 2 kan verwerken en wegschrijven naar de interne HD. Op die momenten gaat de buffer dus eventjes meer gevuld zijn. Moest er geen buffer aanwezig zijn, lus 1 en lus 2 zouden dan serieel moeten uitgevoerd worden, dan zouden er op deze moment frames verloren gaan. Lus 3 heeft geen tijdsprioriteit.

De buffer is een circulaire buffer d.w.z. dat deze buffer zich gedraagt als een FIFO buffer maar met een vast aantal geheugenplaatsen. De werking wordt uitgelegd aan de hand van een voorbeeld. Stel men heeft een circulaire buffer met 5 geheugenplaatsen.

X X X X X

X betekend een lege geheugenplaats, de buffer hierboven is dus leeg.

X 1 X X X

Er wordt een element "1" in de buffer geplaatst. De plaats waar dit gebeurt is onbelangrijk aangezien karakter van een circulaire buffer.

X X 1 X X

Stel nu dat het vorige element uit de buffer wordt gehaald en er een nieuw element "1" wordt weggeschreven, dan krijgt men een situatie zoals hierboven weergegeven. Het weghalen van elementen uit de buffer gebeurt steeds volgens het FIFO principe.

X X 1 2 X

Als er nu een element "2" bij in de buffer wordt geplaatst, dan zitten er op dit moment 2 elementen in. In dit project schrijft lus 1 altijd naar de buffer en lus 2 haalt de elementen uit de buffer. Met een ideale werking zou de buffer altijd gevuld zijn met 1 element. Indien lus 2 niet kan volgen met het weghalen van de elementen uit de buffer, dan zal die buffer na een bepaalde tijd "overlopen".

4 5 1 2 3

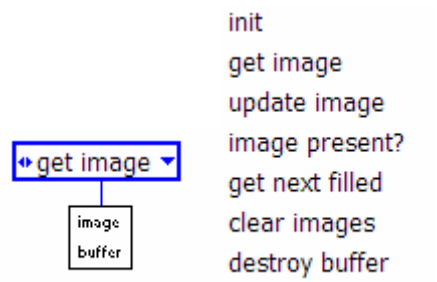
Hierboven is de buffer volledig gevuld.

4 5 6 2 3

Wanneer lus 2 gedurende lange tijd niet meer kan volgen met weghalen van elementen uit de buffer, dan zullen er uiteindelijk waardes in de buffer overschreven worden (door lus 1). Hierboven kunt u zien dat het element "1" overschreven was door het element "6". Met een element bedoelt men een frame of een beeld.

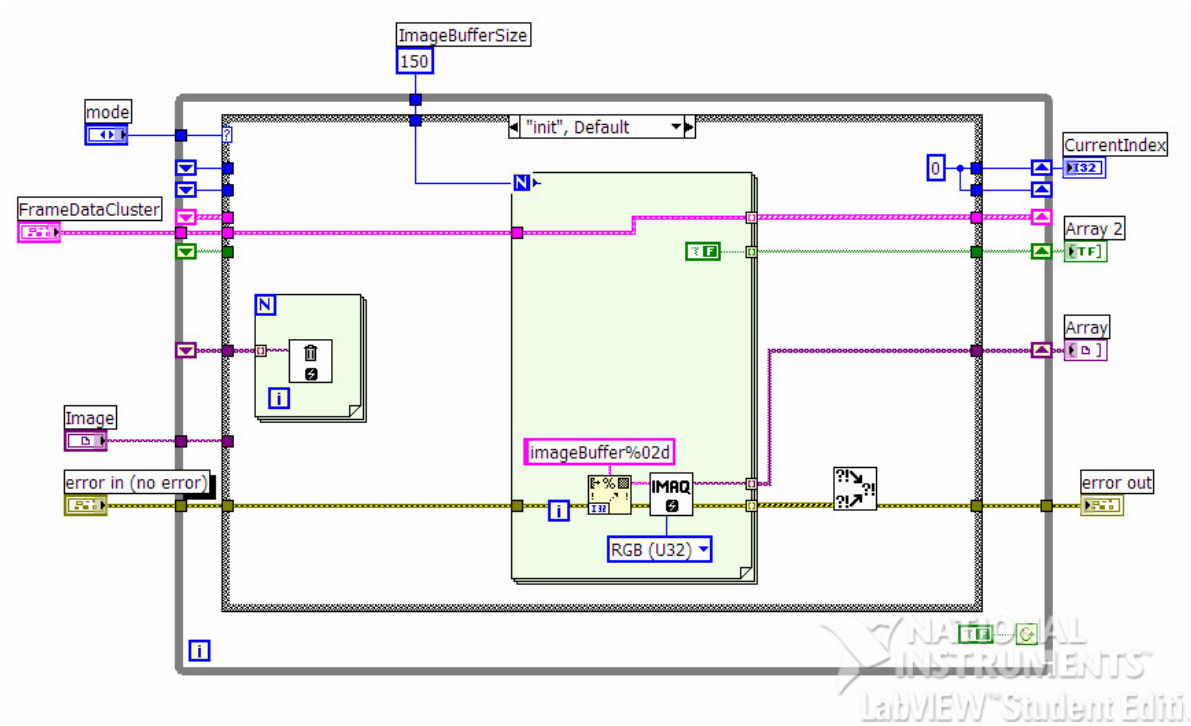
De keuze voor een circulaire buffer is te verklaren door het feit dat beelden of frames veel geheugen in beslag nemen. Indien lus 2 niet meer zou kunnen volgen met het weghalen van frames uit de buffer en de buffer zou een gewone FIFO buffer zijn, dan bestaat de kans dat het volledige geheugen overspoeld zou worden. Indien dit zou gebeuren dan zou de embedded PC niet meer kunnen functioneren en de logger zal niet meer operationeel zijn. Maar door het gebruik van een circulaire buffer worden er alleen frames overschreven, terwijl de logger operationeel blijft. Het nadeel is dat er in de finale videobestanden beelden zullen missen.

In LabVIEW is de buffer als een GLI aangemaakt. In Figuur 57 zijn zijn voorstelling + zijn verschillende modes weergegeven.



Figuur 57: Image buffer (get image mode) + andere modes

De eerste mode is de mode “init”, deze mode zal tijdens het opstarten uitgevoerd worden.



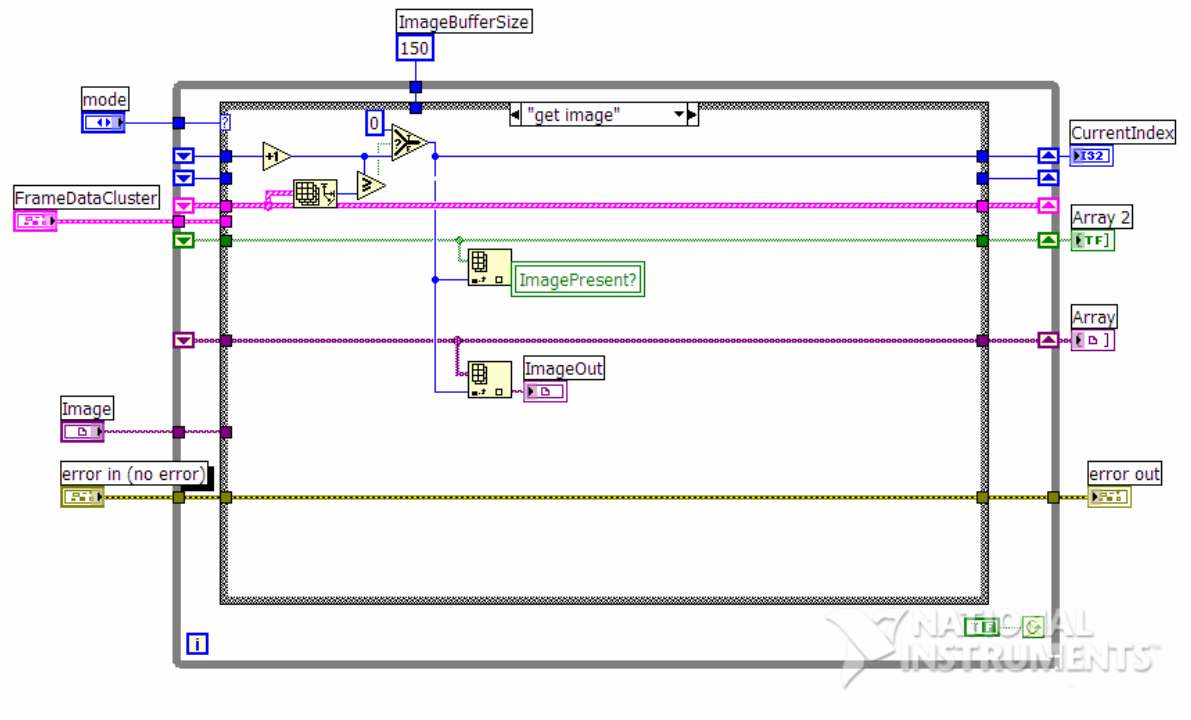
Figuur 58: Image buffer (init mode)

Indien er al geheugenplaatsen bestonden in de GLI, zal hij ze in de init mode eerst weer verwijderen om er nadien 150 nieuwe te maken. Deze geheugenplaatsen zijn van het type IMAQ image. Elke geheugenplaats kan aangesproken worden door een pointer. Ze worden als een array bewaart in het shift register (paars). Ook wordt er een leeg

array van "FrameData" clusters aangemaakt (roos). Alle elementen van dit array hebben een 1 op 1 relatie met de elementen (beelden) in het IMAQ image array (paars) Dit type cluster koppelt het tijdstip van een frame aan het framenummer van dat frame. Men weet van elk beeld in de buffer dus het tijdstip, het framenummer en de pointer.

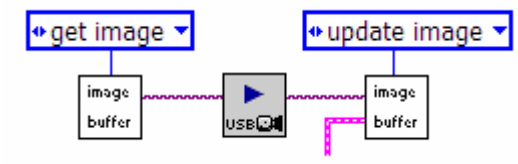
De procedure voor het wegschrijven van een beeld naar de buffer gaat als volgt:

Eerst wordt de mode "get image" (Figuur 59) opgeroepen. In deze mode wordt de huidige index met 1 vermeerderd. Op deze plaats/index zal het nieuwe element terechtkomen. Indien deze som een index oplevert die groter is dan 150 (aantal elementen), dan wordt er terug opnieuw begonnen bij index 0 (circulair). Men kan 2 blauwe shift registers opmerken, het bovenste shift register, dat bijhoudt op welke plaats er het laatst is weggeschreven, wordt vanaf nu de "writeIndex" genoemd. Het onderste shiftregister houdt de index bij wanneer er in de mode "get next filled" een frame gelezen wordt, deze index wordt vanaf nu de "readIndex" genoemd (waarover later meer).



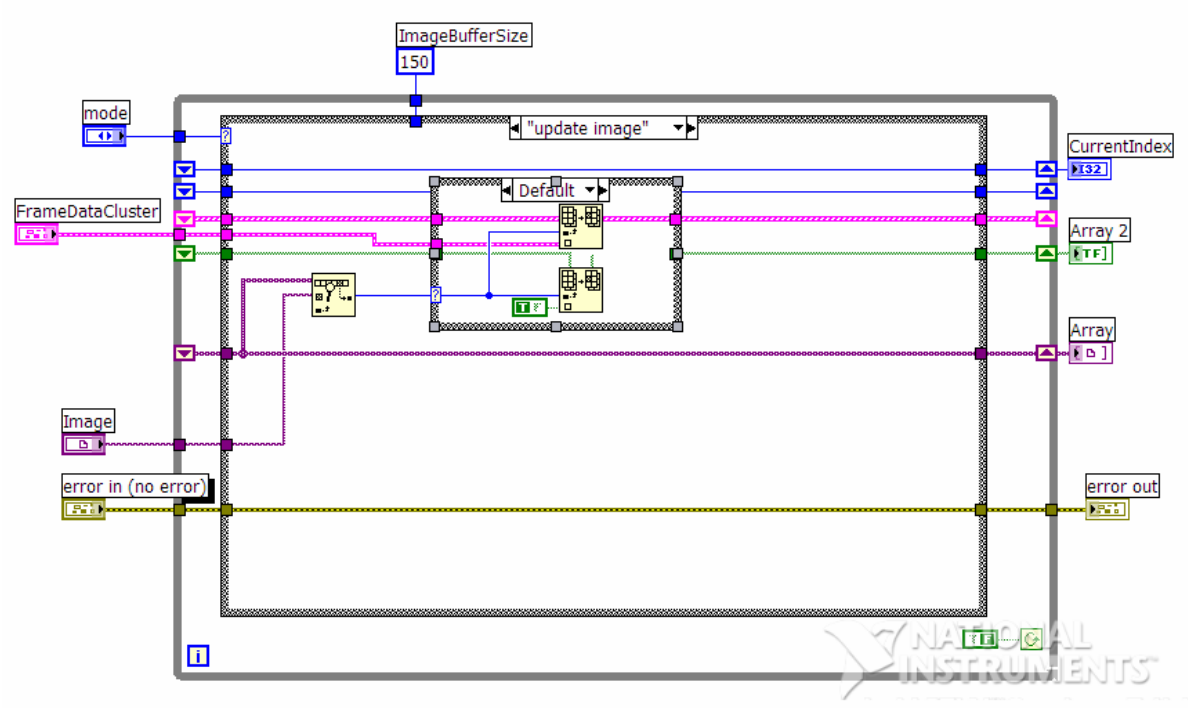
Figuur 59: Image buffer (get image mode)

ImageOut (paars) is een pointer naar die geheugenplaats. Deze pointer wordt doorgegeven aan het vi dat beelden van de webcam binnenhaalt. Dit vi zal zijn beeld op de geheugenplaats, naar waar de pointer wijst, wegschrijven. Vervolgens wordt de pointer doorgegeven aan de image buffer gli in de mode "update image" (Figuur 60).



Figuur 60: procedure wegschrijven beeld naar image buffer

In de mode “update image” (Figuur 61), heeft de GLI 2 ingangen. Dit zijn een “FrameData” cluster, waar zich het tijdstip en het framenummer (van het frame dat net is weggeschreven) in bevindt, en de image pointer van dat frame. In deze mode zoekt de GLI die image pointer op in zijn array van image pointers (elementen). Zo weet hij op welke index het zich bevindt. Omdat het image array (paars) en het “FrameData” cluster array (roos) een 1 op 1 relatie hebben, wordt de “FrameData” cluster (aan de ingang van de GLI) weggeschreven in het “FrameData” cluster array (roos) op dezelfde index als waar de corresponderende image pointer in zijn array (paars) zich bevindt. In de groene array wordt de status van de geheugenplaatsen bijgehouden, dit is het statusarray. Het statusarray heeft ook een 1 op 1 relatie met de 2 laatstgenoemde arrays en is van het type boolean. De taak van het statusarray bestaat erin om een up to date weergave te zijn van welke geheugenplaatsen er bezet zijn (“true” is bezet).



Figuur 61: Image buffer (update image)

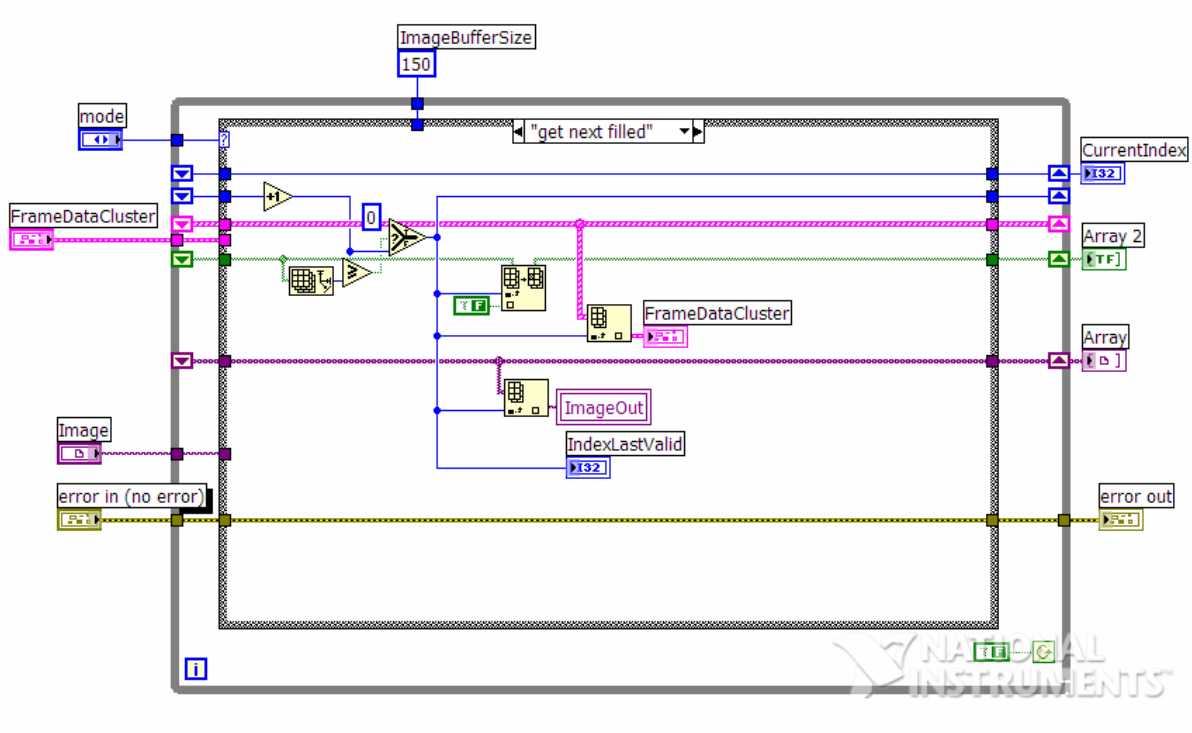
Het wegschrijven gebeurt in lus 1, de frames weer uit de buffer halen gebeurt in lus 2.

Als men een frame uit de buffer wil halen, gebruikt men de mode “get next filled” (Figuur 62). In deze mode wordt de “readIndex” met 1 verhoogt. Men gaat dus telkens als men deze mode oproept het volgende beeld (ImageOut) met zijn “FrameData” cluster aanbieden aan de uitgang van de GLI.

Indien de “readIndex” groter wordt dan het aantal elementen, begint hij terug vanaf 0. Omdat de elementen op dezelfde manier zijn weggeschreven (namelijk altijd op de volgende index), blijft het principe van first in, first out (FIFO) gehanteerd.

In deze mode wordt de corresponderende index in de statusarray op “false” gezet, d.w.z. dat die geheugenplaats nu leeg is.

In lus 2 wordt er tevens een overlay op het beeld geplaatst. Dit overlay laat in de linkerbovenhoek van elk beeld het tijdstip en het framenummer van dat beeld zien.



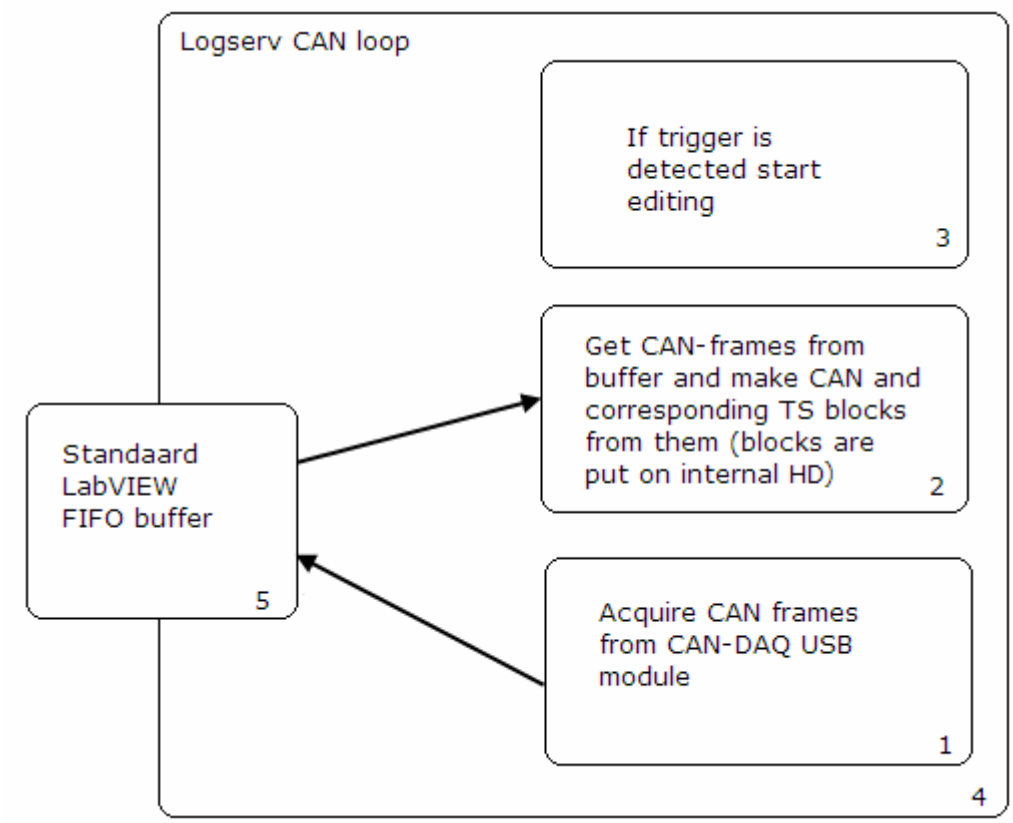
Figuur 62: Image buffer (get next filled)

De overige modes zijn "clear images" en "destroy buffer". De eerste maakt een nieuwe "FrameData" cluster array en een nieuwe statusarray aan (de array van image pointers blijft behouden). De tweede verwijdert alle data aanwezig in de GLI, deze mode wordt gebruikt bij het afsluiten.

6.3.4 Het "Logserv CAN" vi

De werking van dit vi is analoog aan het "Logserv Video" vi. In plaats van een circulaire buffer wordt er een gewone standaard LabVIEW FIFO buffer gebruikt, omdat de data van de CAN frames significant minder geheugen in nemen dan de beeldframes, is er geen of minder gevaar dat er een overflow van het geheugen optreed. Vanwege dezelfde reden bestaat een element in de LabVIEW FIFO CAN buffer uit 10 CAN frames i.p.v. 1 frame. In Figuur 63 kan men zien dat de structuur van dit vi analoog is aan dat van het "Logserv Video" vi.

In lus 1 worden de CAN frames binnengehaald met behulp van de CAN Frame API (Figuur 44). Er wordt geen onderscheid gemaakt tussen frames, ze worden allemaal gelogd.



Figuur 63: Werking Logserv CAN vi

7 VIEWER

De Viewer is de tool die het mogelijk maakt de gelogde data van de logger (op de externe HD) te analyseren. De Viewer wordt geïnstalleerd op een gewone computer. Op deze computer dient wel de LEAD codec geïnstalleerd te zijn, anders is de Viewer niet in staat om de gelogde videobeelden te decoderen.

7.1 DBC formaat

Het DBC formaat is ontwikkeld door Vector en heeft als bedoeling om een koppeling te leggen tussen de CAN frames en de verschillende channels in de frames. In het "Data field" van een CAN frame kunnen immers meerdere types data zitten. Een DBC file heeft verschillende entiteiten en zijn opbouw gelijkt op Figuur 25.

BO_ 2365484270 TCO1: 8 Vector__XXX

SG_ VehicleSpeed : 48|16@1+ (0.00390625,0) [0|250.996] "km/h" Vector__XXX

SG_ OutputShaftSpeed : 32|16@1+ (0.125,0) [0|8031] "rpm" Vector__XXX

SG_ DrawerStatus : 30|2@1+ (1,0) [0|0] "" Vector__XXX

SG_ TachographPerformance : 28|2@1+ (1,0) [0|0] "" Vector__XXX

SG_ HandlingInformation : 26|2@1+ (1,0) [0|0] "" Vector__XXX

SG_ SystemEvent : 24|2@1+ (1,0) [0|0] "" Vector__XXX

SG_ Driver2Card : 20|4@1+ (1,0) [0|0] "" Vector__XXX

SG_ Driver2RelatedStates : 16|4@1+ (1,0) [0|0] "" Vector__XXX

SG_ Driver1Card : 12|4@1+ (1,0) [0|0] "" Vector__XXX

SG_ Driver1RelatedStates : 8|4@1+ (1,0) [0|0] "" Vector__XXX

SG_ DriveRecognize : 6|2@1+ (1,0) [0|0] "" Vector__XXX

SG_ Driver2WorkingState : 3|3@1+ (1,0) [0|0] "" Vector__XXX

SG_ Driver1WorkingState : 0|3@1+ (1,0) [0|0] "" Vector__XXX

Figuur 64: entiteit uit DBC bestand

Deze entiteit behandelt alle CAN frames die een ID hebben van 2365484270 (0x8CFE6CEE). Het actuele ID is 0xCFE6CEE. Het meest linkse hexadecimale getal (hier 0x8) is een aanduiding die voor de software van Vector belangrijk is, maar voor dit project is hij onbelangrijk.

Elke lijn beginnende met "SG" stelt een channel in het frame "BO_ 2365484270 TCO1" voor. Er wordt als voorbeeld het channel met data over de snelheid van de vrachtwagen genomen.

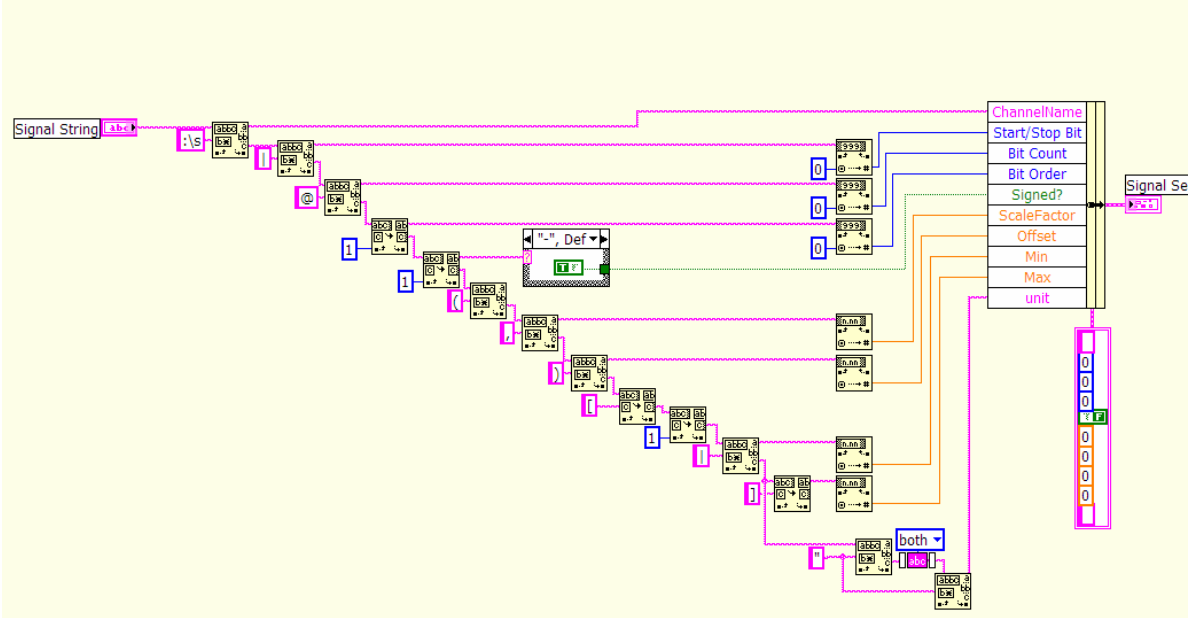
"SG_ VehicleSpeed : 48|16@1+ (0.00390625,0) [0|250.996] "km/h" Vector__XXX"

- 48 is de bit in het "data field" van het frame waar dit channel begint.

- 16 is de lengte van het channel uitgedrukt in bits.
- 1 is de bitorder (big of little endian).
- + betekend dat de data signed is, - betekend unsigned.
- 0.00390625 is een schaalfactor waarmee de waarde van het datapunt in het channel moet vermenigvuldigd worden.
- 0 stelt een eventuele offset voor, in dit geval is hij 0. Voor de reële waarde van het datapunt van het channel te kennen, dient men dus eerst te vermenigvuldigen met de schaalfactor en daarna de offset bij het resultaat op te tellen.
- 0|250.996 stelt respectievelijk een minimumwaarde van 0 en een maximumwaarde van 250.996 voor. Het datapunt in het channel mag deze grenzen niet overschrijden.
- Het laatste element stelt de eenheid van het datapunt voor. In dit voorbeeld zal die "km/h" zijn.

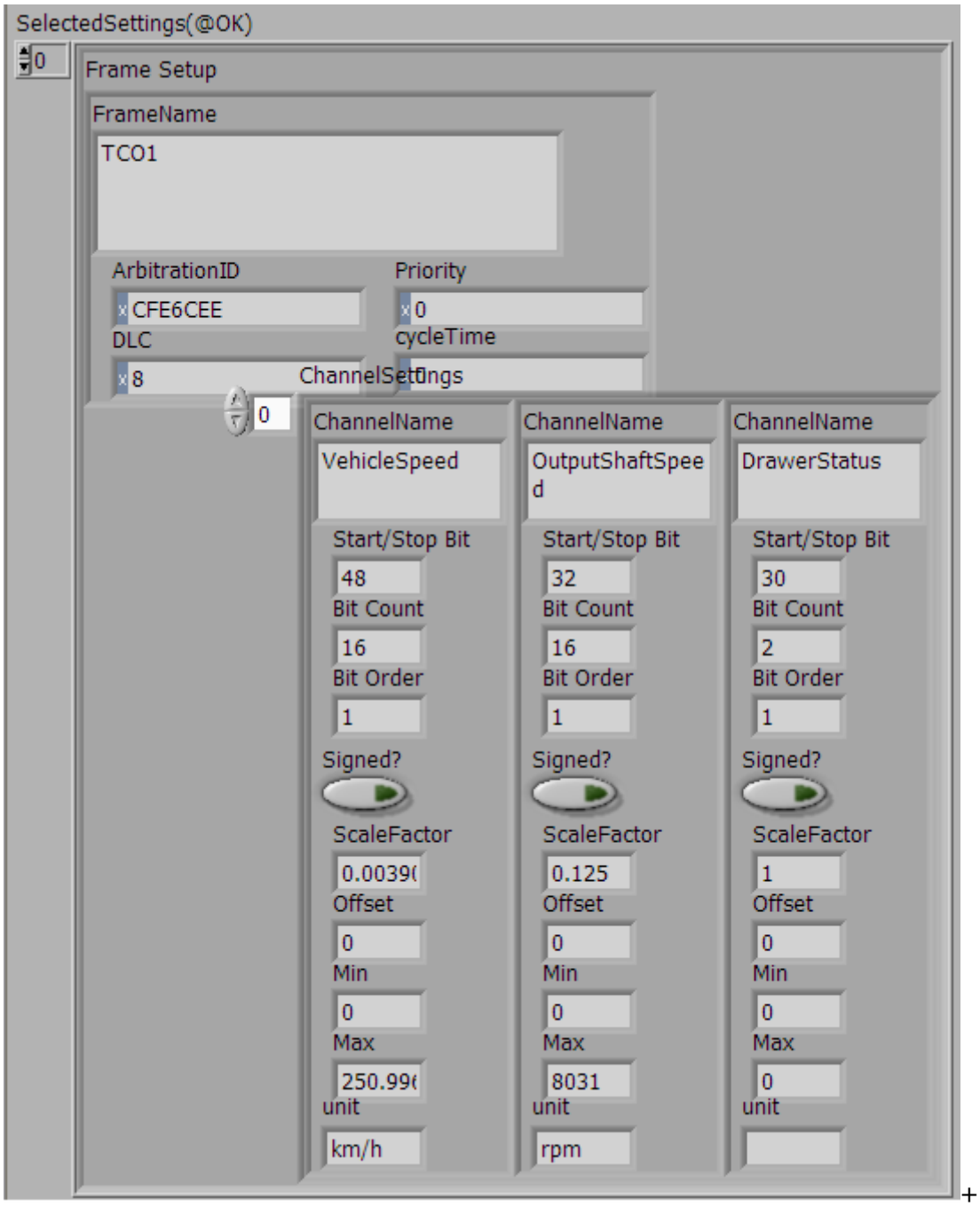
Zoals reeds vermeld, wordt er elke 50 ms een BO_ 2365484270 TCO1 frame verzonden.

Indien men alle lengtes van alle channels van een bepaald frame optelt, zou men een grootte moeten bekomen die niet groter is dan 8 bytes. Een "data field" in een CAN frame mag immers maar 8 bytes lang zijn. De proef wordt op de som genomen: 16+16+2+2+2+2+4+4+4+4+2+3+3 = 64 bits = 8 bytes = OK.



Figuur 65: LabVIEW code voor de structuur van 1 channel in een frame te analyseren

De code hierboven behandelt 1 channel van de DBC file. Uiteindelijk zal de gehele DBC file doorzocht worden en alle gevonden frames en channels met hun informatie zullen opgeslagen worden in een array.

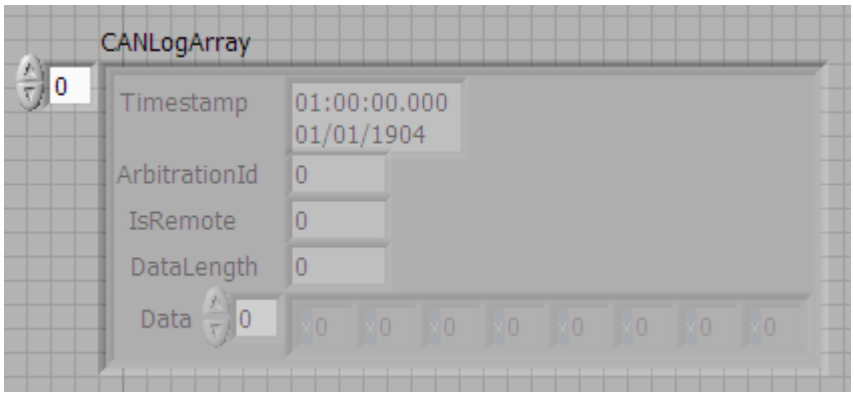


Figuur 66: array van informatie over de geselecteerde frames/channels

Men kan opmerken dat in dit array geen CAN frame data zit. Op dit moment zijn de koppelingen die in de DBC file vastgelegd zijn, vertaald naar de software. De gebruiker kan selecteren welke frames/channels hij wil zien in de grafiek, alleen de informatie van die frames/channels (Figuur 66) zal worden doorgegeven naar het proces dat het parsen beheert.

7.2 Parsen

Een CANlog bestand bestaat uit een array van alle gelogde frames. In Figuur 67 kan men de voorstelling zoals die in LabVIEW gebruikt is, bekijken.



Figuur 67: CANlog formaat

Tijdens het parsen gaat men in de gelogde CANlog bestanden op zoek naar de ID's (ArbitrationId van de frames) die overeenkomen met de ID's van de frames die de gebruiker wil weergeven op de grafiek. Indien het proces zo een frame tegenkomt in de CANlog van de huidige geselecteerde trigger, dan gaat het proces in het "data field" van dat frame op zoek naar alle datapunten van de verschillende channels. Dit is mogelijk omdat het proces over de informatie beschikt wat de startposities van de verschillende channels zijn en welke hun offsets en schaalfactors zijn (zie 7.1).

Elk channel zal in de software zijn eigen array toegewezen krijgen. Het proces gaat nu in elke channel array een element bijvoegen. Dit laatste toegevoegde element in elk channel array bestaat uit het respectievelijke datapunt van elk channel dat zonet gevonden was.

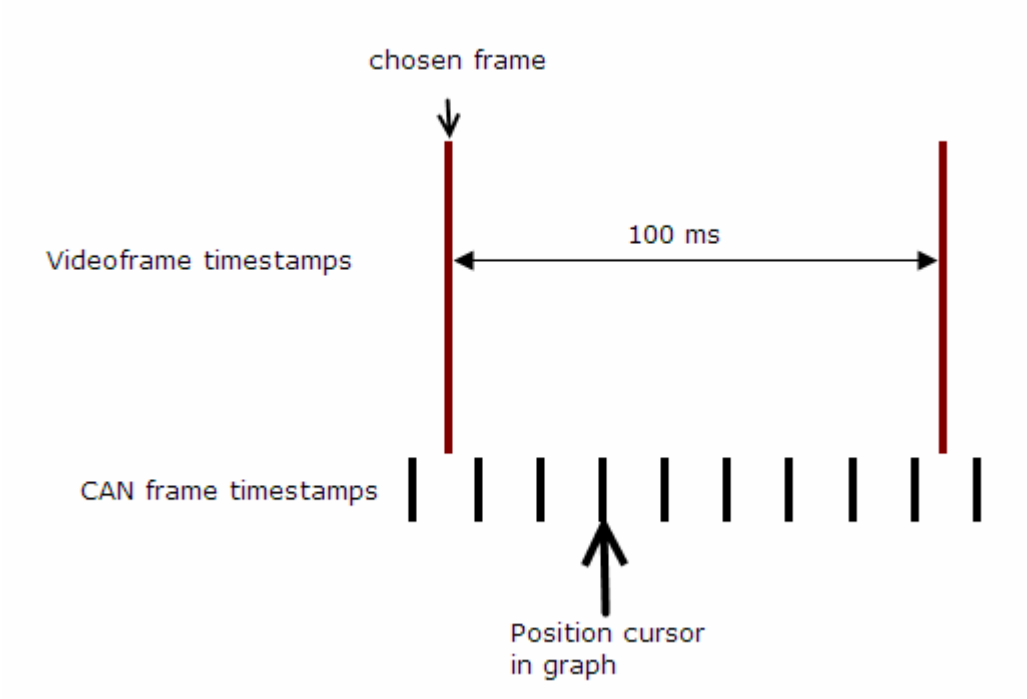
Vervolgens worden deze channel arrays met de datapunten afgeleverd aan de grafiek.

7.3 Synchronisatie

Het beeld dat getoond wordt in het interne en externe videovenster wordt als volgt bepaalt.

De bovenste timestamps in Figuur 68 zijn die van de verschillende videoframes, omdat er aan 10 fps gelogd is, zit er een tijdspanne van 100 ms tussen de videoframes. De tijdspanne tussen de CAN frames (onderste timestamps) wordt bepaalt door het type frame.

Stel nu dat de cursor van de grafiek op een bepaalde positie staat, dan wordt het frame dat het dichtste tegen de linkerkant van de cursor ligt, getoond in het videovenster. Het is immers tijdens dit videoframe dat het CAN frame op de positie van de cursor gelogd werd.

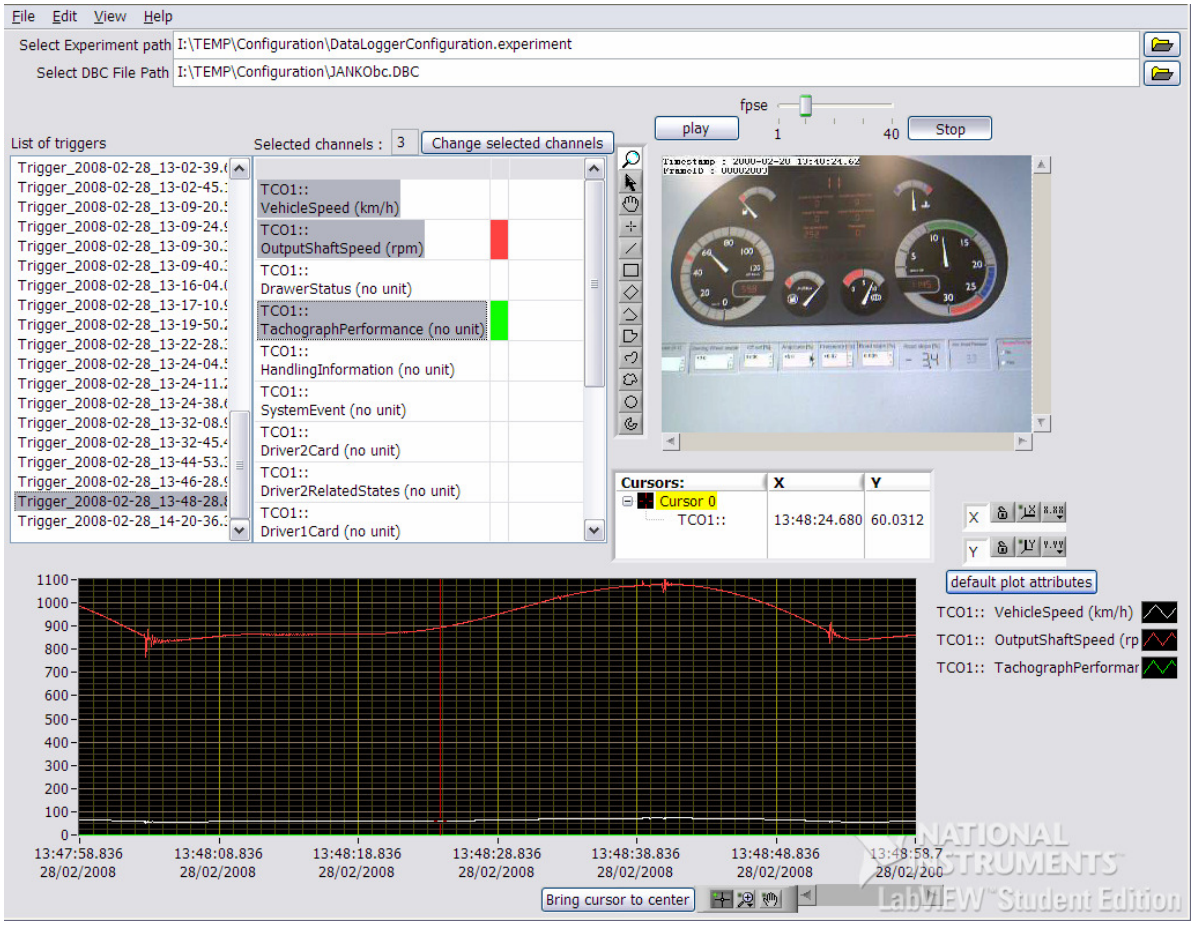


Figuur 68: Synchronisatie

De Viewer gebruikt voor het synchroniseren van video met CAN een video TS bestand. Het zoeken naar de juiste videoframes gebeurt immers veel sneller in een TS bestand dan in een "groot" videobestand. Voor de CANlog wordt de inwendige tijdsdata gebruikt.

7.4 GUI

De verschillende controls van het hoofdscherm (Figuur 69) zullen kort besproken worden. Het testen van het project gebeurde in DAF Eindhoven op een vrachtwagensimulator. De webcam filmt het dashboard van de simulator, op die manier kan men de synchronisatie van de CAN data met de video verifiëren. Tijdens het praktisch gebruik zal men de webcam het verkeer voor de vrachtwagen laten filmen.



Figuur 69: Hoofdscherm Viewer

Indien men het venster van het hoofdscherm groter of kleiner maakt, dan wordt de grafiek ook groter of kleiner. De grafiek schaalt dus mee met de grootte van het venster.

7.4.1 Menu

Helemaal bovenaan ziet u het menu van de trigger. Onder “File” kan men een nieuw experiment openen, een andere DBC file kiezen of het gelogde .CANlog formaat omzetten naar een .asc formaat. Het .asc formaat wordt gebruikt door het Vector Canalyzer software pakket wat ze bij DAF Eindhoven gebruiken voor CAN signalen te analyseren. Als laatste optie onder “File” kan men het programma beëindigen.

Onder “Edit” kan men kiezen om de laatst gemaakte actie(s) ongedaan te maken (Undo) of (Redo).

Onder “View” kan men het extra videovenster oproepen en onder “help” kan men een venster met informatie oproepen. De context van dit laatste venster past zich aan, aan de informatietekst van de control waar de muis op die moment opstaat.

7.4.2 Het Select Experiment Path

Met een experiment worden de folders /Logs/ en /Configuration/ tezamen met hun inhoud bedoeld (zie 3.3.1.2). Men start met het openen van een experiment via het menu of via deze control. In deze control dient het "DataLoggerConfiguration.experiment" .ini bestand geselecteerd te worden. Eenmaal dat het corresponderende pad gekozen is, detecteert de Viewer automatisch de .DBC file en alle triggers in het experiment.

Hij selecteert de eerste DBC file die hij tegenkomt in dezelfde folder als het .ini bestand. Deze folder noemt /Configuration/. Als de testingenieur wenst dat de DBC file automatisch geladen wordt na het selecteren van het experiment, dient hij manueel een .DBC file in de /Configuration/ folder van het experiment te zetten. Dit is aan te raden, omdat op deze manier altijd een (juiste) DBC file aanwezig is bij het 'experiment'.

7.4.3 Het Select DBC File path

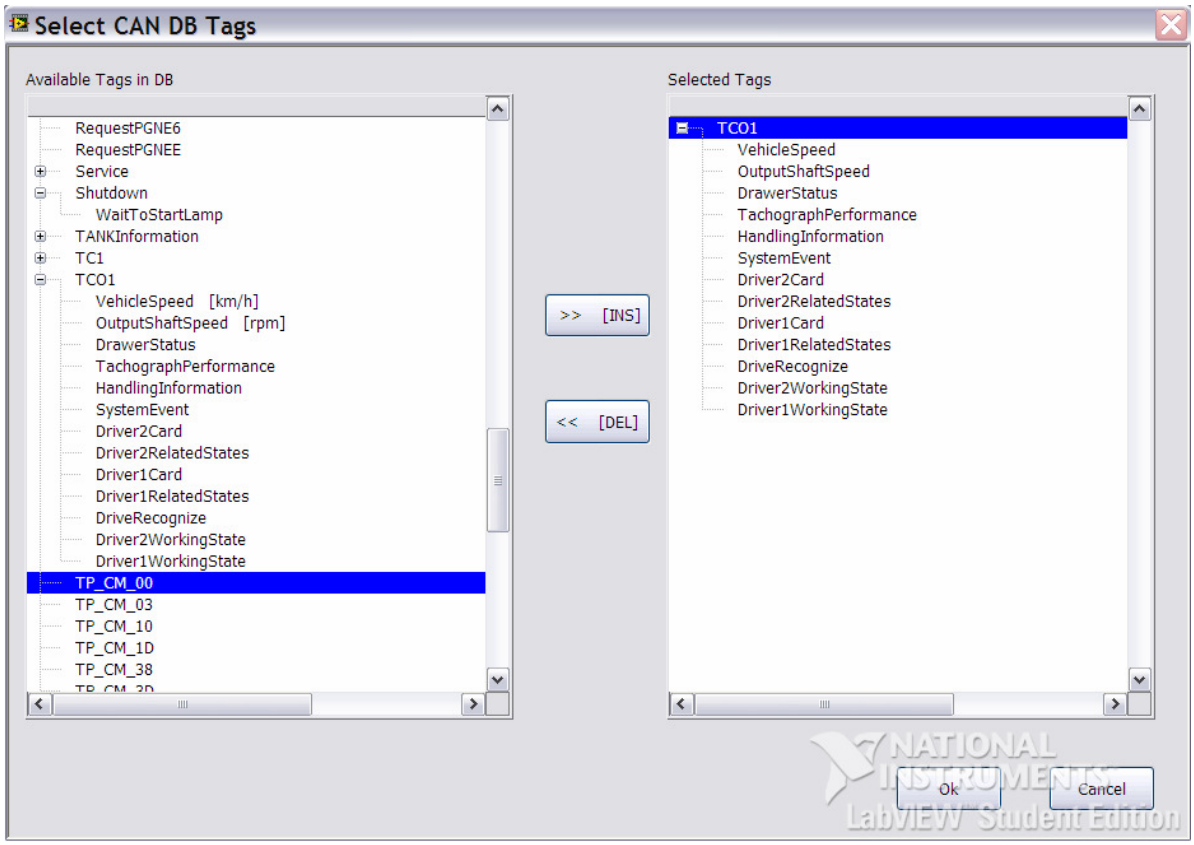
De testingenieur heeft ook de mogelijkheid om ten allen tijde een andere DBC file te laden. Daarvoor dient hij via deze control naar de gewenste DBC file te browsen. Na het kiezen van een andere DBC file, zullen de triggers geparsed worden aan de hand van deze nieuwe DBC file.

7.4.4 List of triggers

In deze lijst zullen na het open van een experiment alle triggers van dit experiment getoond worden. Indien men een trigger selecteert, wordt de data van die trigger in het geheugen geladen.

7.4.5 De Selected channels list

In deze lijst worden de geselecteerde CAN channels weergegeven, initieel is deze lijst leeg. Omdat alle soorten frames en bijhorende channels gelogd worden, is het niet overzichtelijk om ze allemaal in een grafiek te laten zien. Daarom is er geopteerd om aan de testingenieur de keuze te laten welke CAN channels hij in de grafiek wil analyseren. De testingenieur kan de inhoud van deze lijst veranderen door op de knop "change selected channels" te klikken. Hij krijgt dan Figuur 70 te zien. Hier kan hij de gewenste tags selecteren, in de linkse lijst staan alle channels die er in de geselecteerde DBC file voorkomen en in de rechtse lijst staan de channels die de testingenieur reeds geselecteerd heeft. Men kan channels van de ene lijst naar de andere kopiëren met de knoppen INS (insert) en DEL (Delete). De inhoud van de rechtse lijst zal ook in de "Selected channels list" van het hoofdscherm verschijnen.



Figuur 70: Select CAN Database tags

In Figuur 69 kan men opmerken dat er 3 channels geselecteerd zijn in de "Selected channels list", deze 3 channels zijn zichtbaar in de grafiek. Het selecteren van meerdere channels in de lijst gebeurt door middel van de "shift" of "ctrl" toets op het toetsenbord in combinatie met het enkel linkerklikken van de muis op het channel. Als men dubbelklikt dan zal de grafiek een "autoscale" doorvoeren.

De uiterst rechtse kolom die in Figuur 69 leeg is, zal een melding "NO DATA!" bevatten per CAN channel waar geen data voor gevonden is (in het CANlog bestand van de geselecteerde trigger). Het kan bijvoorbeeld gebeuren dat een DBC file meer entities beschrijft dan er verschillende types CAN frames gelogd werden.

7.4.6 Grafiek

In de grafiek worden alle channels weergegeven die in de "selected channels" list geselecteerd zijn. De grafiek biedt de mogelijkheid om in te zoomen op bepaalde stukken data. Men kan ook de dikte, type, kleur, … van de datalijnen kiezen door met de rechtermuisknop op de legende naast de grafiek te klikken.

Het belangrijkste element in de grafiek is de cursor. In Figuur 69 kan men zien dat deze is voorgesteld door een rode verticale lijn. Op dit moment is de cursor gekoppeld aan de witte datalijn dewelke de snelheid van de vrachtwagen voorstelt. Men kan de cursor aan andere datalijnen koppelen door te rechterklikken op "Cursor0" rechtsboven de grafiek. Hier kan men ook de waarde van de huidige positie van de cursor aflezen (X en Y as). Op deze moment bedraagt die 60,0312 km/h. De cursor kan men manueel verplaatsen door hem met de muis te bewegen.

De knop “bring cursor to center” zal de cursor naar het midden van de grafiek verschuiven ongeacht of er ingezoomd is of niet.

Men kan opmerken dat de som van de pretrigger en de posttrigger in Figuur 69, één minuut bedraagt. (13:48:58 – 13:47:58 = 1 min).

7.4.7 Intern videoscherm

Dit scherm bevat altijd het beeldframe dat overeenkomt met de positie van de cursor in de grafiek. Als men de cursor gaat verschuiven, zal het frame in dit videoscherm mee veranderen. Er zit een tolerantie van +/-100 ms op het beeldframe en de positie waar de cursor zich bevindt in het CAN channel in de grafiek. Dit is te verklaren door het feit dat men maar tegen 10 beelden per seconde gelogd heeft. De meeste CAN channels sturen een nieuw datapunt in een ordegrootte van milliseconden. Het witte CAN channel (Vehiclespeed) bijvoorbeeld stuurt om de 50 ms een nieuw frame met een nieuw datapunt in.

7.4.8 Play, stop knop en fpse slider

Een play functie mag natuurlijk niet ontbreken. Wanneer de testingenieur op de play knop klikt, zal de video van de geselecteerde trigger beginnen afspelen met een framerate die bepaald wordt door de waarde van de fpse (frames per second edit) slider. Samen met het afspelen van de video zal de cursor mee door de grafiek lopen. De positie van de cursor wordt per beeldframe bepaald door het tijdstip van het eerste datapunt dat ouder is dan het tijdstip van het beeldframe.

7.4.9 Extern videoscherm

Via het menu “View” kan men een tweede videovenster oproepen. Dit is vooral handig als men 2 computerschermen ter beschikking heeft. Op het ene scherm kan men het hoofdvenster met de grafiek maximaliseren en op het andere scherm kan met het tweede videovenster maximaliseren, zo bekomt men een volwaardige “dual screen“ setup.

De werking van dit externe videoscherm is identiek aan dat van het intern. Men zal dus ten allen tijde hetzelfde beeldframe in het interne en externe videoscherm kunnen waarnemen.

Figuur 71: extern videovenster

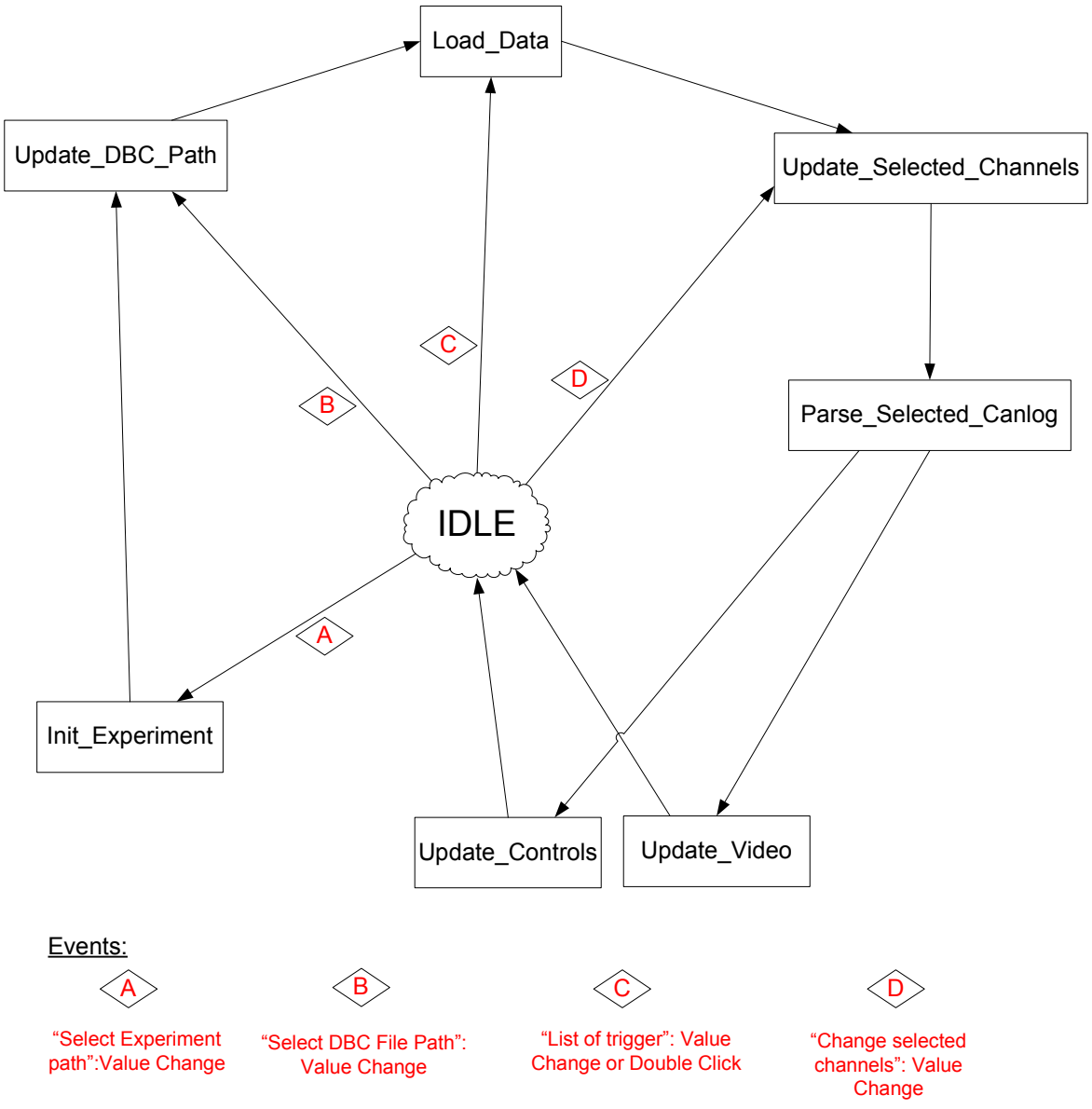
In dit tweede venster zijn er ook een play, stop knop en een fps slider geplaatst. Hun werking is identiek aan hun corresponderende controls in het hoofdvenster.

Men kan opmerken dat de snelheid op het dashboard van de simulator (links) 60 km/h aangeeft. Dit komt overeen met de waarde van de cursor in Figuur 69.

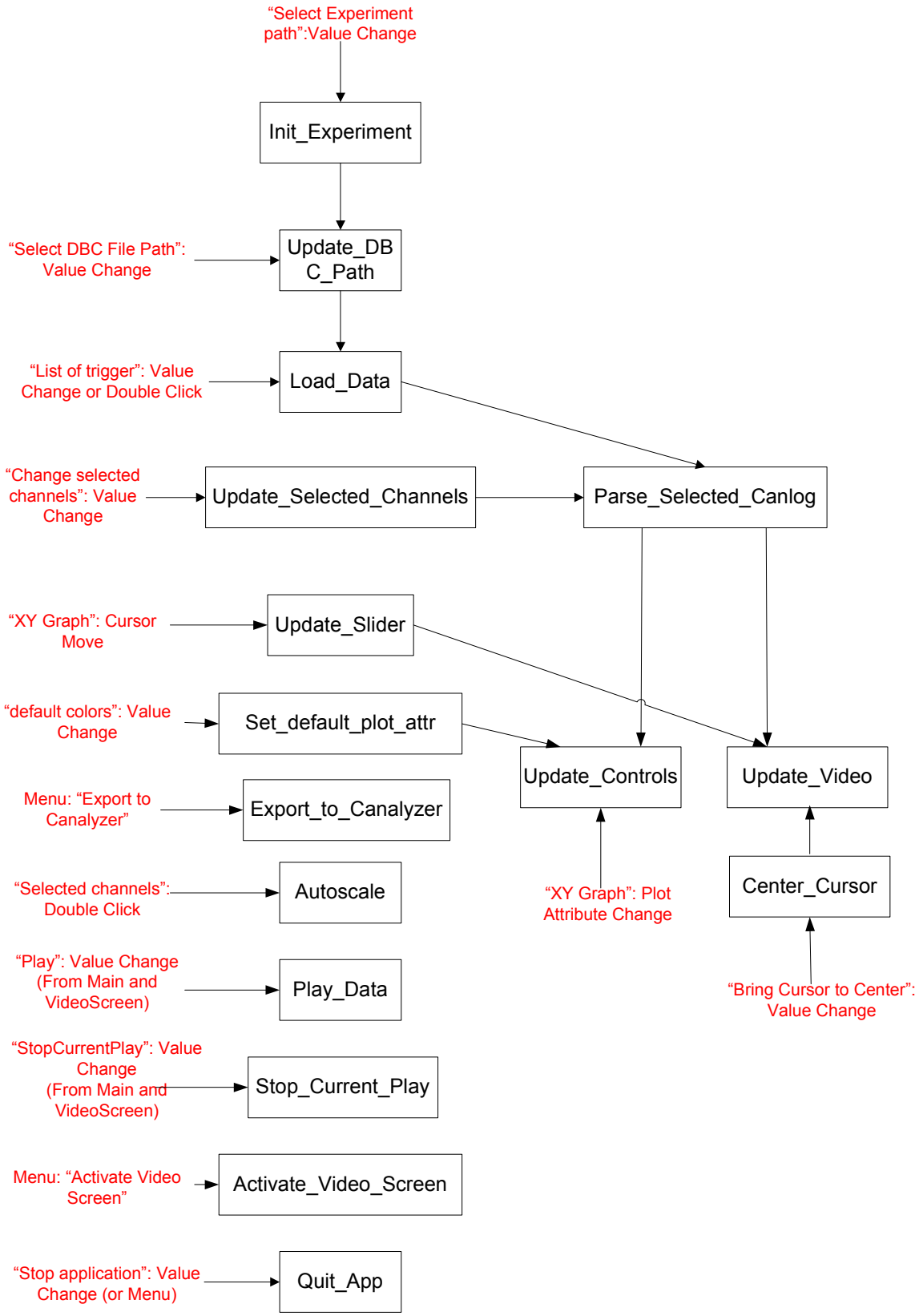
7.5 Principe van de Viewer

De Viewer is ontwikkeld op basis van een state machine en een GLI. De statemachine bepaalt **wanneer** er iets dient uitgevoerd te worden en de GLI bepaalt **wat** er in elke state uitgevoerd moet worden.

In Figuur 73 zijn de belangrijkste states weergegeven. De rode tekst (tekst niet in rechthoek) stelt GUI events voor. Het operating system detecteert deze events en geeft ze door aan LabVIEW. Alle events worden gedetecteerd in de "idle" of rust state. Deze state is de default state, m.a.w. wanneer de gebruiker niets aanklikt en hij dus geen event genereert. Stel dat de gebruiker het externe videovenster wil weergeven, dan klikt hij in het menu "View>Activate video screen". Dit event ("Menu: Activate video screen") wordt gedetecteerd waarna de state "Activate_Video_Screen" zal gestart worden. Als deze state beëindigd is, keert de statemachine terug naar zijn "idle" state, waarna hij weer een nieuw event kan ontvangen. Indien er na een state geen andere state meer volgt (er vertrekt geen pijl), dan wordt de "idle" state terug actief (voor het overzicht te behouden is de "idle" state in Figuur 73 weggelaten).



Figuur 72: kern states



Figuur 73: Statemachine Viewer

In Figuur 72 is er andere manier van weergeven gebruikt, hier zijn alleen de states weergegeven die voor de kern werking van de Viewer zorgen. De "Idle" state is in deze figuur wel weergegeven.

Zoals reeds vermeld, dient er begonnen te worden met het selecteren van een experiment. Het "Select Experiment path: Value Change" event of event A triggert de state "Init_Experiment". Wanneer de state "Init_Experiment" beëindigd is, wordt de state "Update_DBC_Path" gestart. Deze werkwijze gaat door totdat de "Idle" state terug bereikt is. Men kan dit zeer goed volgen in Figuur 73 en Figuur 72. Merk op dat de events A,B,C en D rechtstreeks een state opstarten. Event C bijvoorbeeld, laat de "Load_Data" state opstarten, wanneer deze laatstgenoemde state beëindigd is, wordt de "Update_Selected_Channels" state gestart. Dit proces gaat door totdat de "Idle" state terug actief is.

7.5.1 Beschrijving van de kern states

Figuur 73 geeft een volledig overzicht van alle states. Buiten de kern states zijn er nog states die het tweede videovenster oproepen, de applicatie beëindigen, of de play functie starten. Deze play functie gebeurt volledig parallel zodat de gebruiker acties kan blijven ondernemen in de GUI (zoals bijvoorbeeld inzoomen op de grafiek). Moest deze play functie niet parallel uitgevoerd worden, dan zou de GUI "hangen" tijdens het afspelen en zou de gebruiker het afspelen eerst moeten stoppen vooraleer hij andere acties in de GUI kan ondernemen.

De werking van de states van Figuur 72 zal nu kort toegelicht worden. In de Viewer wordt er 1 GLI gebruikt die de data van de geselecteerde trigger in het geheugen laadt en de GLI bevat de nodige intelligentie om die data te parsen en met mekaar te synchroniseren (Video met CAN).

7.5.1.1 Init_Experiment

Hier wordt de GLI geïnitialiseerd d.w.z. dat alle shift registers van de GLI leeggemaakt worden en er wordt ook een geheugenplaats aangemaakt waar het beeld van het videovenster in bewaard zal worden. De inhoud van deze geheugenplaats bevat steeds het beeldframe dat overeenkomt met de positie van de cursor in de grafiek.

In deze state wordt er gezocht naar de DBC file in de /Configuration/ folder van het experiment. Wanneer er een DBC bestand gevonden is, wordt de "Select DBC file path" control geüpdate met het path naar het gevonden DBC bestand. Indien er geen DBC bestand gevonden is, zal het programma aan de gebruiker vragen welk DBC bestand hij wil gebruiken.

7.5.1.2 Update_DBC_Path

Bij het uitvoeren van deze state wordt de waarde die zich bevindt in de "Select DBC file path" control, in de GLI opgeslagen. Zo weet de GLI altijd waar hij het DBC bestand kan terugvinden.

7.5.1.3 Load_Data

In deze state wordt alle data van de **geselecteerde trigger** in de "List of triggers" control in het geheugen (GLI) geladen. Dit zijn 3 bestanden. In het geval dat niet alle bestanden beschikbaar zijn, dan zullen de volgende acties plaatsvinden:

- Videobestand gevonden, maar geen bijhorende TS → maak een nieuwe TS aan gebaseerd op de tijdsdata in het videobestand.
- Videobestand en/of CANlog bestand niet gevonden → ga door met het gevonden bestand. Wanneer er bijvoorbeeld alleen een CANlog bestand gevonden wordt, dan zal de Viewer geen video laten zien bij deze trigger.

Na het uitvoeren van deze state heeft de GLI altijd beschikking over de data die bij de geselecteerde trigger hoort. Indien er een andere trigger geselecteerd wordt, zal deze data uiteraard aangepast worden (event C Figuur 72).

7.5.1.4 Update_Selected_Channels

Gedurende deze state zal de GLI intern het proces beschreven in 7.1 uitvoeren (koppelingen in het DBC bestand worden vertaalt naar de software). Tijdens dit intern proces zal er aan de gebruiker gevraagd worden welke CAN channels/frames hij wil analyseren (Figuur 70).

7.5.1.5 Parse_Selected_Canlog

De GLI weet nu welke CAN channels/frames de gebruiker wil bekijken. In deze state zal de GLI intern het parsen (punt 7.2) uitvoeren. Hij kan dit doen omdat hij tijdens de "Update_Selected_Channels" state, de informatie die nodig is voor het parsen, uit het DBC bestand gehaald heeft.

7.5.1.6 Update_Controls

Gedurende deze state zullen de channels, die weergegeven zijn in de grafiek, geüpdate worden met de geselecteerde channels in de "Selected channels" lijst.

De koppeling tussen de kleuren van de geselecteerde channels in deze lijst en de kleuren gebruikt door de channels in de grafiek, zal ook geüpdate worden.

7.5.1.7 Update_Video

In deze state zal de GLI de synchronisatie (punt 7.3) uitvoeren. De GLI beschikt intern immers over alle gegevens (timestamps) die nodig zijn bij het synchronisatie proces. Na het uitvoeren van deze state zal het videovenster gesynchroniseerd zijn met de positie van de cursor in de grafiek.

BESLUIT

Het project was een zeer leerrijke en boeiende ervaring. Het heeft mij veel kennis bijgebracht over LabVIEW en de CAN-bus.

Met deze logger en de Viewer kunnen ze bij DAF fouten of onvolmaaktheden tijdens de ontwikkeling van nieuwe systemen beter analyseren. De logger en de Viewer zijn ondertussen intensief door DAF getest geweest. Bovendien is het project momenteel in gebruik genomen door DAF Eindhoven. Het systeem is al zeer nuttig gebleken. Andere constructeurs hebben reeds hun interesse getoond.

De triggers voor het starten van het loggen worden momenteel enkel en alleen gegeven door het indrukken van een hardware-knop. Een mogelijke uitbreiding zou kunnen zijn dat er automatisch triggers worden gegeven indien bepaalde thresholds van bepaalde CAN-signalen overschreden worden. Dit zou het project een nog groter toepassingsgebied geven.

Het project werd door het CIT op de NIdays 2008 Expo tentoongesteld. Die Expo werd op 10 April 2008 te Zaventem gehouden (zie bijlage 4).

LITERATUURLIJST

BOSCH. (1991). CAN specification version 2.0. 73 p.

Cia. (s.a.). Controller Area Network. 46 p.

Cia. Can In Automation: http://www.can-cia.org (2008).

Colleman, P. (2007). Datacommunicatie, 298 p.

Colleman, P. (2007). *Datacommunicatie*: Digitale communicatie

Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. (2000). *Universal Serial Bus Specification*: Revision 2.0, 622 p.

Corrigan, S. (2003). Introduction to the Controller Area Network (CAN). 16 p.

Johansson, K., Törngren, M., Nielsen L. (s.a.). Vehicle Applications of Controller Area Network 25 p.

Martin, T. (2006). CAN Controller. In *the insider's guide to the Philips ARM 7*. (pp.105 - 119).

National Instruments. (s.a.). CAN: Controller Area Network. In *LabVIEW Introduction Course Manual.* (pp.9-1 - 9-49).

National Instruments. (2006). *CAN*: NI-CAN Hardware and Software Manual.

National Instruments: http://www.ni.com (2008)

Peacock, C. (2002). *USB in a Nutshell*: Making Sense of the USB Standard. 30 p. http://www.beyondlogic.org

Renesas. (2006). Introduction to CAN. 44 p.

VECTOR. (2004). Introduction to J1939: Version 1.0, 6 p.

Universal Serial Bus. http://www.usb.org (2008)

Wikipedia. http://en.wikipedia.org (2008)

Zeltwanger, H. J1939-based application profiles. 15 p.

BIJLAGEN

Bijlage 1: USB DAQ module datasheet

Low-Cost Multifunction DAQ for USB

NI USB-6008, NI USB-6009

- Small and portable
- 12 or 14-bit input resolution, at up to 48 kS/s
- Built-in, removable connectors for easier and more cost-effective connectivity
- 2 true DAC analog outputs for accurate output signals
- 12 digital I/O lines (TTL/LVTTL/CMOS)
- 32-bit event counter
- Student kits available
- OEM versions available

Operating Systems
- Windows 2000/XP
- Mac OS X[1]
- Linux®[1]
- Pocket PC
- Win CE

Recommended Software
- LabVIEW
- LabWindows/CVI

Measurement Services Software (included)
- NI-DAQmx
- Ready-to-run data logger

[1]Mac OS X and Linux users need to download NI-DAQmx Base.

Product	Bus	Analog Inputs[1]	Input Resolution (bits)	Max Sampling Rate (kS/s)	Input Range (V)	Analog Outputs	Output Resolution (bits)	Output Rate (Hz)	Output Range (V)	Digital I/O Lines	32-Bit Counter	Trigger
USB-6009	USB	8 SE/4 DI	14	48	±1 to ±20	2	12	150	0 to 5	12	1	Digital
USB-6008	USB	8 SE/4 DI	12	10	±1 to ±20	2	12	150	0 to 5	12	1	Digital

[1]SE = single ended, DI = differential

Hardware Description

The National Instruments USB-6008 and USB-6009 multifunction data acquisition (DAQ) modules provide reliable data acquisition at a low price. With plug-and-play USB connectivity, these modules are simple enough for quick measurements but versatile enough for more complex measurement applications.

Software Description

The NI USB-6008 and USB-6009 use NI-DAQmx high-performance, multithreaded driver software for interactive configuration and data acquisition on Windows OSs. All NI data acquisition devices shipped with NI-DAQmx also include VI Logger Lite, a configuration-based data-logging software package.

Mac OS X and Linux users can download NI-DAQmx Base, a multiplatform driver with a limited NI-DAQmx programming interface. You can use NI-DAQmx Base to develop customized data acquisition applications with National Instruments LabVIEW or C-based development environments. NI-DAQmx Base includes a ready-to-run data logger application that acquires and logs up to eight channels of analog data.

PDA users can download NI-DAQmx Base for Pocket PC and Win CE to develop customized handheld data acquisition applications.

Recommended Accessories

The USB-6008 and USB-6009 have removable screw terminals for easy signal connectivity. For extra flexibility when handling multiple wiring configurations, NI offers the USB-6008/09 Accessory Kit, which includes two extra sets of screw terminals, extra labels, and a screwdriver.

In addition, the USB-6008/09 Prototyping Accessory provides space for adding more circuitry to the inputs of the USB-6008 or USB-6009.

Common Applications

The USB-6008 and USB-6009 are ideal for a number of applications where economy, small size, and simplicity are essential, such as:
- Data logging – Log environmental or voltage data quickly and easily.
- Academic lab use – The low price facilitates student ownership of DAQ hardware for completely interactive lab-based courses. (Academic pricing available. Visit ni.com/academic for details.)
- Embedded OEM applications.

NATIONAL INSTRUMENTS

Information for Student Ownership

To supplement simulation, measurement, and automation theory courses with practical experiments, NI has developed the USB-6008 and USB-6009 student kits, which include the LabVIEW Student Edition and a ready-to-run data logger application. These kits are exclusively for students, giving them a powerful, low-cost hands-on learning tool. Visit **ni.com/academic** for more details.

Information for OEM Customers

For information on special configurations and pricing, call (800) 813 3693 (U.S. only) or visit **ni.com/oem**. Go to the Ordering Information section for part numbers.

Ordering Information

NI USB-6008[1]	779051-01
NI USB-6009[1]	779026-01
NI USB-6008 OEM	193132-02
NI USB-6009 OEM	193132-01
NI USB-6008 Student Kit[1,2]	779320-22
NI USB-6009 Student Kit[1,2]	779321-22

[1] Includes NI-DAQmx software, NI ready-to-run data logger software, and a USB cable.

[2] Includes LabVIEW Student Edition.

BUY NOW!

For complete product specifications, pricing, and accessory information, call 800 265 9891 (U.S. only) or go to ni.com/usb.

Specifications

Typical at 25 °C unless otherwise noted.

Analog Input

Absolute accuracy, single-ended

Range	Typical at 25 °C (mV)	Maximum (0 to 55 °C) (mV)
±10	14.7	138

Absolute accuracy at full scale, differential[1]

Range	Typical at 25 °C (mV)	Maximum (0 to 55 °C) (mV)
±20	14.7	138
±10	7.73	84.8
±5	4.28	58.4
±4	3.59	53.1
±2.5	2.56	45.1
±2	2.21	42.5
±1.25	1.70	38.9
±1	1.53	37.5

Number of channels ..... 8 single-ended/4 differential
Type of ADC ..... Successive approximation

ADC resolution (bits)

Module	Differential	Single-Ended
USB-6008	12	11
USB-6009	14	13

Maximum sampling rate (system dependent)

Module	Maximum Sampling Rate (kS/s)
USB-6008	10
USB-6009	48

Input range, single-ended ..... ±10 V
Input range, differential ..... ±20, ±10, ±5, ±4, ±2.5, ±2, ±1.25, ±1 V
Maximum working voltage ..... ±10 V
Overvoltage protection ..... ±35 V
FIFO buffer size ..... 512 B
Timing resolution ..... 41.67 ns (24 MHz timebase)
Timing accuracy ..... 100 ppm of actual sample rate
Input impedance ..... 144 k
Trigger source ..... Software or external digital trigger
System noise ..... 0.3 LSB_{rms} (±10 V range)

Analog Output

Absolute accuracy (no load) ..... 7 mV typical, 36.4 mV maximum at full scale
Number of channels 2
Type of DAC ..... Successive approximation
DAC resolution ..... 12 bits
Maximum update rate ..... 150 Hz, software-timed

Output range ..... 0 to +5 V
Output impedance ..... 50 Ω
Output current drive ..... 5 mA
Power-on state ..... 0 V
Slew rate ..... 1 V/µs
Short-circuit current ..... 50 mA

Digital I/O

Number of channels ..... 12 total
8 (P0.<0..7>)
4 (P1.<0..3>)
Direction control ..... Each channel individually programmable as input or output
Output driver type
USB-6008 ..... Open-drain
USB-6009 ..... Each channel individually programmable as push-pull or open-drain
Compatibility ..... CMOS, TTL, LVTTL
Internal pull-up resistor ..... 4.7 kΩ to +5 V
Power-on state ..... Input (high impedance)
Absolute maximum voltage range ..... -0.5 to +5.8 V

Digital logic levels

Level	Min	Max	Units
Input low voltage	-0.3	0.8	V
Input high voltage	2.0	5.8	V
Input leakage current	–	50	µA
Output low voltage (I = 8.5 mA)	–	0.8	V
Output high voltage (push-pull, I = -8.5 mA)	2.0	3.5	V
Output high voltage (open-drain, I = -0.6 mA, nominal)	2.0	5.0	V
Output high voltage (open-drain, I = -8.5 mA, with external pull-up resistor)	2.0	–	V

Counter

Number of counters 1
Resolution ..... 32 bits
Counter measurements ..... Edge counting (falling edge)
Pull-up resistor ..... 4.7 kΩ to 5 V
Maximum input frequency ..... 5 MHz
Minimum high pulse width ..... 100 ns
Minimum low pulse width ..... 100 ns
Input high voltage ..... 2.0 V
Input low voltage ..... 0.8 V

Power available at I/O connector

+5 V output (200 mA maximum) ..... +5 V typical
+4.85 V minimum
+2.5 V output (1 mA maximum) ..... +2.5 V typical
+2.5 V output accuracy ..... 0.25% max
Voltage reference temperature drift ..... 50 ppm/°C max

[1]Input voltages may not exceed the working voltage range.

Physical Characteristics

If you need to clean the module, wipe it with a dry towel.

Dimensions (without connectors)	6.35 by 8.51 by 2.31 cm (2.50 by 3.35 by 0.91 in.)
Dimensions (with connectors)	8.18 by 8.51 by 2.31 cm (3.22 by 3.35 by 0.91 in.)
Weight (without connectors)	59 g (2.1 oz)
Weight (with connectors)	84 g (3 oz)
I/O connectors	USB series B receptacle (2) 16-position (screw-terminal) plug headers
Screw-terminal wiring	16 to 28 AWG
Screw-terminal torque	0.22 to 0.25 N•m (2.0 to 2.2 lb•in.)

Power Requirement

USB (4.10 to 5.25 VDC)	80 mA typical 500 mA maximum
USB suspend	300 µA typical 500 µA maximum

Environmental

The USB-6008 and USB-6009 are intended for indoor use only.

Operating environment	
Ambient temperature range	0 to 55 °C (tested in accordance with IEC-60068-2-1 and IEC-60068-2-2)
Relative humidity range	10 to 90%, noncondensing (tested in accordance with IEC-60068-2-56)
Storage environment	
Ambient temperature range	-40 to 85 °C (tested in accordance with IEC-60068-2-1 and IEC-60068-2-2)
Relative humidity range	5 to 90%, noncondensing (tested in accordance with IEC-60068-2-56)
Maximum altitude	2,000 m (at 25 °C ambient temperature)
Pollution degree	2

Safety and Compliance

Safety

This product is designed to meet the requirements of the following standards of safety for electrical equipment for measurement, control, and laboratory use:

- IEC 61010-1, EN 61010-1
- UL 61010-1, CAN/CSA-C22.2 No. 61010-1

Note: For UL and other safety certifications, refer to the product label or visit **ni.com/certification**, search by model number or product line, and click the appropriate link in the Certification column.

Electromagnetic Compatibility

This product is designed to meet the requirements of the following standards of EMC for electrical equipment for measurement, control, and laboratory use:

- EN 61326 EMC requirements; Minimum Immunity
- EN 55011 Emissions; Group 1, Class A
- CE, C-Tick, ICES, and FCC Part 15 Emissions; Class A

Note: For EMC compliance, operate this device according to product documentation.

CE Compliance

This product meets the essential requirements of applicable European Directives, as amended for CE marking, as follows:

- 73/23/EEC; Low-Voltage Directive (safety)
- 89/336/EEC; Electromagnetic Compatibility Directive (EMC)

Note: Refer to the Declaration of Conformity (DoC) for this product for any additional regulatory compliance information. To obtain the DoC for this product, visit **ni.com/certification**, search by model number or product line, and click the appropriate link in the Certification column.

Waste Electrical and Electronic Equipment (WEEE)

EU Customers: At the end of their life cycle, all products must be sent to a WEEE recycling center. For more information about WEEE recycling centers and National Instruments WEEE initiatives, visit **ni.com/environment/weee.htm**.

NI Services and Support

NI has the services and support to meet your needs around the globe and through the application life cycle – from planning and development through deployment and ongoing maintenance. We offer services and service levels to meet customer requirements in research, design, validation, and manufacturing. Visit ni.com/services.

Training and Certification

NI training is the fastest, most certain route to productivity with our products. NI training can shorten your learning curve, save development time, and reduce maintenance costs over the application life cycle. We schedule instructor-led courses in cities worldwide, or we can hold a course at your facility. We also offer a professional certification program that identifies individuals who have high levels of skill and knowledge on using NI products. Visit ni.com/training.

Professional Services

Our Professional Services Team is comprised of NI applications engineers, NI Consulting Services, and a worldwide National Instruments Alliance Partner program of more than 600 independent consultants and integrators. Services range from start-up assistance to turnkey system integration. Visit ni.com/alliance.

NATIONAL INSTRUMENTS™
Certified Alliance Partner

OEM Support

We offer design-in consulting and product integration assistance if you want to use our products for OEM applications. For information about special pricing and services for OEM customers, visit ni.com/oem.

Local Sales and Technical Support

In offices worldwide, our staff is local to the country, giving you access to engineers who speak your language. NI delivers industry-leading technical support through online knowledge bases, our applications engineers, and access to 14,000 measurement and automation professionals within NI Developer Exchange forums. Find immediate answers to your questions at ni.com/support.

We also offer service programs that provide automatic upgrades to your application development environment and higher levels of technical support. Visit ni.com/ssp.

Hardware Services

NI Factory Installation Services

NI Factory Installation Services (FIS) is the fastest and easiest way to use your PXI or PXI/SCXI combination systems right out of the box. Trained NI technicians install the software and hardware and configure the system to your specifications. NI extends the standard warranty by one year on hardware components (controllers, chassis, modules) purchased with FIS. To use FIS, simply configure your system online with ni.com/pxiadvisor.

Calibration Services

NI recognizes the need to maintain properly calibrated devices for high-accuracy measurements. We provide manual calibration procedures, services to recalibrate your products, and automated calibration software specifically designed for use by metrology laboratories. Visit ni.com/calibration.

Repair and Extended Warranty

NI provides complete repair services for our products. Express repair and advance replacement services are also available. We offer extended warranties to help you meet project life-cycle requirements. Visit ni.com/services.

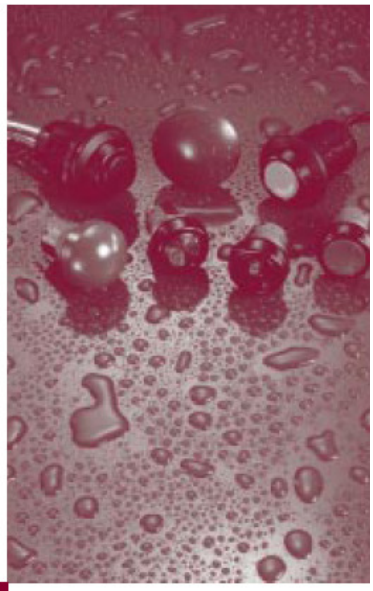
NATIONAL INSTRUMENTS™

ni.com • (800) 813 3693
National Instruments • info@ni.com

351378A-01 2006-7322-301-101-D

Bijlage 2: Hardware-knop datasheet

Data Sheet no. 76-94700



SERIES 76-94 – PANEL SEALED RANGE

FEATURES

- Non-spark material
- Explosive-proof materials used
- Wide temperature range
  – from -40°C to +125°C
- Panel sealed product
  – Protection against immersion
- Impact rating - 20J
- Choice of actuators and styles
  -Push-buttons, rotary and
  panel lamp

NON-STANDARD OPTIONS

- PCB terminals
- Gold contacts
- 2 poles available
- Colour variants
- In-filled engraving
  (Push-button only)

ITW Switches

Technical Information

Series 76-94

The 76-94 Series has been designed to cover a wide range of harsh environmental applications.

The range is available in 3 styles, push-button, rotary and panel lamp.

The push-button comes in 4 button styles with 3 colour options and has solder/faston terminals as standard. The rotary is a single pole, 2 position, snap action switch with green or red spot indicator on the black actuator. To complete the range an indicator version is available with red, green or amber lens.

The entire range is panel sealed to IP67 and due to the construction and the materials used, are all capable of surviving the impact of a 5kg mass dropped from 40cm (20J). All the bezels and the actuators of the push-buttons and rotaries are made from a non-sparking zinc alloy whilst the panel lamp lens is manufactured from polycarbonate, which maintains the specification of the range.

Non-standard options for the range include custom colours, PCB terminals, gold contacts and 2 pole variants.

The features of the Series makes it ideal for robust, potentially explosive environments, such as the mining industry.

Specification

The push-button and rotary variants utilise ITW Switches' proven Series 16 snap action microswitch.

Mechanical

Travel	2,5mm (PB)
Life (nom)	2 x 10^6 cycles (PB), 50,000 cycles (R) 1 x 10^5 cycles (Latched Push-button)
Operating force	7N (PB)
Contact bounce	5ms (PB, R)
Panel thickness (max)	6mm (PB, R, L)

Electrical

Dielectric strength	1000V RMS (PB, R)
Insulation resistance	1GΩ (PB, R)
Contact resistance (initial)	25mΩ (PB, R)
Current rating (max)	10A resistive 250V a.c. (PB, R)
Life	100,000 cycles 20°C (PB, R) 50,000 cycles 70°C (PB, R)
Illumination type	T1¾ Wedge base bulb (L)

Environmental & Physical

Ingress protection	IP67 (PB, R, L)
Impact rating	20J (PB, R, L)
Shock	50g (PB, R, L)
Operating temperature range	-40°C to +125°C (PB, R, L)
Body material	Zinc alloy (PB, R, L)
Actuator material	Zinc alloy (PB, R)
Lens material	Polycarbonate (L)
Contact material	Fine silver (PB, R)
Terminal material	5µm bright acid tin plate nickel flash undercoat (PB, R, L)

Key

PB	Push-button
R	Rotary
L	Panel Lamp Indicator

Ordering Information

76-94 [ ][ ] /xxxxxx [ ]

Button Style

Flush	1
Raised	2
Large circular (latching)	3
Large circular (momentary)	4
Domed	5
Panel lamp indicator	6
Rotary	7

Indicator Colour

Push-button & rotary	0
Red	R
Green	G
Amber	A

(No addition to the part number is required for the indicator after this)

Termination & No. of Poles

Solder/faston	1 pole	439088
Solder	1 pole	4044
Solder	2 poles	804

Indicator Colour

Black	B
Red	R
Green	G

(For rotary: standard black actuator, specify colour of dot indicator, red & green only)

Panel Mounting & construction information

The 76-94 mounts easily into panels not exceeding 6mm thickness. Front panel sealing to IP67 is achieved by use of the sealing washer which is fitted to the switch before it is inserted into the panel hole cut out. It is held into the panel by means of a zinc lock nut, tightened down to a torque of between 1,5Nm to 2,85Nm to achieve the correct sealing pressure.

It is mounted into a standard 22,5mm diameter hole and a double 'D' flat is provided to prevent rotation if required.

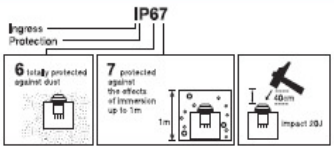
Lamp Construction & Multi-pole Information

The 76-94 panel lamp variant is supplied with a quick release bulb holder that will accept a T1¾ wedge base filament lamp. Because of the wide range of illumination available (different voltage lamps, multi chip LEDs etc.), we do not supply a bulb as standard although we are prepared to do so if required.

The push-button and rotary versions have a non-standard option of 2 poles. The 2 pole switch is provided separate to the switch body and must be fitted, via a spring clip, after the switch body has been mounted in the panel.

For more information, please contact the Sales Office.

Ingress Protection Impact Key

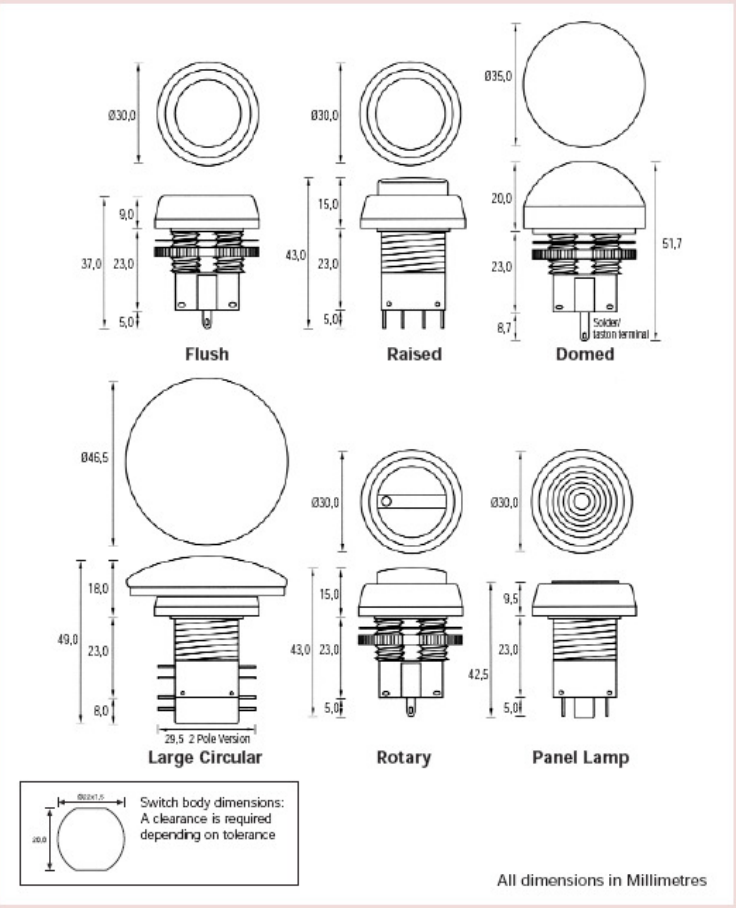


Data Sheet no. 76-94700

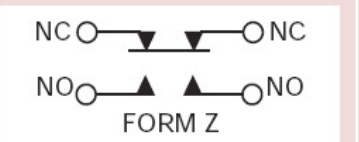
SERIES 76-94
Panel Sealed Range

Technical Information

Product Drawings



Circuit Form



Application References

- Military environments
- Robust industrial
- Vandal resistant requirements
- Mining industry
- Petrochemical
- Process control
- Material handling

Further Information

For further information on our complete range of switch products, visit our website – www.itwswitchcon.com or contact our Sales Office.

ITW Switches

ITW Switches, Division of ITW Limited, Norway Road, Hilsea, Portsmouth PO3 5HT, UK
Tel: +44 (0)2392 694971 Fax: +44 (0)2392 666352 Website: www.itwswitchcon.com

Bijlage 3: USB-CAN DAQ module datasheet

CAN and LIN Interfaces for Hi-Speed USB

NI USB-8472, NI USB-8472s, NI USB-8473, NI USB-8473s, NI USB-8476, NI USB-8476s,

- 1-port interfaces for high-speed CAN, low-speed/fault-tolerant CAN, and LIN
- Optional hardware synchronization
- Hi-Speed USB, bus-powered
- 500 V digital isolation
- Bus error logging
- Hardware timestamping – 1 µs resolution

CAN Interfaces

- Philips SJA1000 CAN controller
- Transmit/receive 100 percent bus load at 1 Mb/s
- ISO 11898 compliance for standard (11-bit) and extended (29-bit) arbitration IDs
- Software-selectable termination for low-speed/fault-tolerant CAN
- J1939 compliance

LIN Interfaces

- Atmel ATA6620 transceiver
- LIN 1.3/2.0 and J2602 compliance
- Software-selectable master/slave termination

Operating Systems

- Windows Vista/XP/2000

Recommended Software

- LabVIEW
- LabWindows™/CVI
- Visual C++ 6.0
- Visual Basic 6.0
- Borland C/C++

Application Software (included)

- CAN/LIN bus monitoring and logging utility

Driver Software (included)

- NI-CAN

Model	Physical Layer	Transceivers	Ports	Max Transfer Rate (kb/s)	Hardware Sync	Software Termination	API
USB-8472	Low-speed/fault-tolerant CAN	TJA1054AT	1	125	–	✓	Frame
USB-8472s	Low-speed/fault-tolerant CAN	TJA1054AT	1	125	✓	✓	Frame
USB-8473	High-speed CAN	TJA1041	1	1,000	–	–	Frame
USB-8473s	High-speed CAN	TJA1041	1	1,000	✓	–	Frame
USB-8476	LIN	ATA6620	1	20	–	✓	Frame
USB-8476s	LIN	ATA6620	1	20	✓	✓	Frame

Table 1. NI USB-847x Selection Guide

Overview

NI USB-847x devices provide Hi-Speed USB interfaces for Controller Area Network (CAN) and Local Interconnect Network (LIN) monitoring, logging, and testing. With high-speed CAN, low-speed/fault-tolerant CAN, and LIN interfaces featuring optional hardware synchronization, you can use two or more USB-847x devices together to interface to a wide variety of CAN and LIN networks.

The USB-847x interfaces are ideal for many types of applications, including:

- In-vehicle network monitoring and logging
- Bus load monitoring
- Device validation with synchronized data acquisition
- CAN device development and test
- CAN and LIN data correlation with external measurements

The convenient all-in-one design features a captive 2 m USB cable and built-in transceiver, requiring no extra cables or accessories to get applications running quickly.

With hardware timestamping, you can log messages with microsecond-accurate timestamps for reconstructing network events and correlating data across synchronized devices. All USB-847x interfaces use an industry-standard 9-pin male D-Sub (DB9) connector to interface to CAN and LIN buses.

Hi-Speed USB connectivity and onboard frame buffering ensure no dropped frames, even under 100 percent bus loads.

CAN Interfaces

USB-847x CAN interfaces feature the industry-standard Philips SJA1000 CAN controller, which implements ISO 11898 CAN functionality. The SJA1000 offers additional features to aid in system development, including listen-only mode, sleep/wakeup mode, error counter access, and self-reception (echo) mode. USB-847x CAN interfaces recognize standard (11-bit) and extended (29-bit) arbitration IDs and are compatible with J1939 networks.

NATIONAL INSTRUMENTS

CAN and LIN Interfaces for Hi-Speed USB

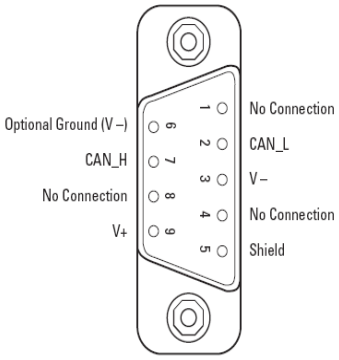


Figure 1. CAN DB9 Connector (USB CAN Modules)

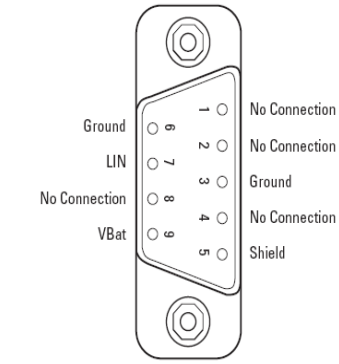


Figure 2. LIN DB9 Connector (USB LIN Modules)

Low-speed/fault-tolerant USB-847x CAN interfaces offer software-selectable termination for compatibility with all low-speed network configurations.

Onboard Physical Layer

The CAN physical layer connects the CAN controller to the physical bus wires. USB-847x CAN interfaces are available with high-speed and low-speed/fault-tolerant physical layers. All USB-847x devices have onboard physical layer transceivers and require no external dongles. The physical layers are internally powered via a DC-to-DC converter and electrically isolated up to 500 V.

LIN Interfaces

NI USB-8476 LIN interfaces, featuring the Atmel ATA6620 LIN transceiver, are compliant with the LIN 1.3/2.0 and J2602 specifications and offer software-selectable master/slave termination. The ATA6620 features baud rates up to 20 kb/s and offers advanced power management with a low-power sleep mode.

Hardware Synchronization

Many automotive applications demand tight integration of CAN, LIN, analog, and digital measurements. In applications such as CAN device development and validation, you need to synchronize different measurements to correlate data. You can program this synchronization in software, but OS latency and clock drift between the different onboard oscillators introduce unacceptable delays for certain automotive test applications.

NI offers USB-847x CAN and LIN interfaces with an optional hardware synchronization feature, with which the USB interfaces can share timing and triggering signals with other USB-847x interfaces, as well as data acquisition, vision, and motion devices, to achieve true hardware synchronization. Determinism is maintained between the trigger signal and the desired response because timing and triggering signals are handled in hardware. The host PC software interacts only to retrieve the data once it is acquired or to write new data.

NI-CAN Software

National Instruments provides NI-CAN driver software for Windows Vista/XP/2000. NI-CAN includes a detailed API and examples for NI LabVIEW and LabWindows/CVI, Microsoft Visual C/C++ 6.0 and Visual Basic 6.0, and Borland C/C++. USB-847x interfaces use the NI-CAN Frame API for frame-level access to messages on CAN and LIN networks. The NI-CAN driver software also includes the Bus Monitor and NI Spy utilities to aid in application development.

NI-CAN Frame API

The NI-CAN Frame API offers a powerful interface for accessing NI CAN hardware. The API enables full access to all traffic on CAN and LIN buses. It is a powerful tool for implementing a variety of applications, including CAN and LIN frame-level access, challenge response protocols, remote frame handling, and advanced synchronization.

Bus Monitor

To quickly monitor all CAN and LIN bus traffic, use Bus Monitor, a utility that offers an easy-to-use interface that displays all CAN and LIN frames on the network, and log the traffic to disk. Bus Monitor provides options to control, display, and view bus statistics.

NI Spy

NI Spy is a utility for monitoring NI-CAN API calls made by applications without recompiling or rebuilding. It is ideal for testing application functionality, troubleshooting problems, and verifying communication with CAN devices. NI Spy dynamically captures and displays all NI-CAN API calls made by applications running on the computer.

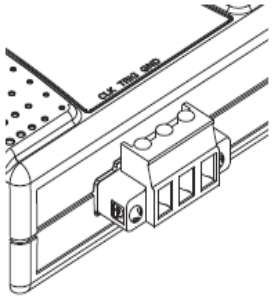


Figure 3. COMBICON Synchronization Connector for USB-847x Devices

Ordering Information

NI USB-8472
1-port, low-speed, USB CAN interface 779790-01

NI USB-8472s
1-port, low-speed, USB CAN interface with synchronization 779791-01

NI USB-8473
1-port, high-speed, USB CAN interface 779792-01

NI USB-8473s
1-port, high-speed, USB CAN interface with synchronization 779793-01

NI USB-8476
1-port, USB LIN interface 779794-01

NI USB-8476s
1-port, USB LIN interface with synchronization 779795-01

CAN Device Simulator
U.S. 120 VAC 779189-01
Euro 240 VAC 779189-02
Japan 100 VAC 779189-07

Cables and Brackets
USB cable strain relief bracket 777550-01
CAN single-termination, high-speed cable, 2 m 192017-02

BUY NOW!

For complete product specifications, pricing, and accessory information, call 800 813 3693 (U.S.) or go to ni.com/can.

Specifications

USB Power Requirements

USB-8472(s), USB-8473(s)	250 mA, 5 VDC
USB-8476(s)	200 mA, 5 VDC

Hardware Synchronization

Input clocks	1, 10, and 20 MHz
Output clock	1 MHz
Trigger lines	1 input/output
Clock lines	1 input/output

Examples included for sync to NI-DAQmx, LIN, CAN, and RTSI bus.

Supported Baud Rates

USB-8472(s)	5,000 to 125,000 baud
USB-8473(s)	40,000 to 1,000,000 baud
USB-8476(s)	2,400 to 20,000 baud

Safety

Galvanic CAN channel to USB isolation	500 V
Withstand	2 s maximum

Dimensions

Nonsynchronized versions	7.87 by 6.35 by 2.54 cm (3.1 by 2.5 by 1.0 in.)
Synchronized versions	7.87 by 7.11 by 2.54 cm (3.1 by 2.8 by 1.0 in.)
Cable length	2 m
I/O connector	9-pin male D-Sub, optional 3-pin COMBICON for synchronization

Operating Environment

Ambient temperature	0 to 55 °C
Relative humidity	10 to 90%, noncondensing (tested in accordance with IEC-60068-2-1, IEC-60068-2-2, IEC-60068-2-56)

NI Services and Support

NI has the services and support to meet your needs around the globe and through the application life cycle – from planning and development through deployment and ongoing maintenance. We offer services and service levels to meet customer requirements in research, design, validation, and manufacturing. Visit **ni.com/services**.

Training and Certification

NI training is the fastest, most certain route to productivity with our products. NI training can shorten your learning curve, save development time, and reduce maintenance costs over the application life cycle. We schedule instructor-led courses in cities worldwide, or we can hold a course at your facility. We also offer a professional certification program that identifies individuals who have high levels of skill and knowledge on using NI products. Visit **ni.com/training**.

Professional Services

Our NI Professional Services team is composed of NI applications and systems engineers and a worldwide National Instruments Alliance Partner program of more than 600 independent consultants and integrators. Services range from start-up assistance to turnkey system integration. Visit **ni.com/alliance**.

NATIONAL INSTRUMENTS
Certified Alliance Partner

OEM Support

We offer design-in consulting and product integration assistance if you want to use our products for OEM applications. For information about special pricing and services for OEM customers, visit **ni.com/oem**.

Local Sales and Technical Support

In offices worldwide, our staff is local to the country, giving you access to engineers who speak your language. NI delivers industry-leading technical support through online knowledge bases, our applications engineers, and access to 14,000 measurement and automation professionals within NI Developer Exchange forums. Find immediate answers to your questions at **ni.com/support**.

We also offer service programs that provide automatic upgrades to your application development environment and higher levels of technical support. Visit **ni.com/ssp**.

Hardware Services

NI Factory Installation Services

NI Factory Installation Services (FIS) is the fastest and easiest way to use your PXI or PXI/SCXI combination systems right out of the box. Trained NI technicians install the software and hardware and configure the system to your specifications. NI extends the standard warranty by one year on hardware components (controllers, chassis, modules) purchased with FIS. To use FIS, simply configure your system online with **ni.com/pxiadvisor**.

Calibration Services

NI recognizes the need to maintain properly calibrated devices for high-accuracy measurements. We provide manual calibration procedures, services to recalibrate your products, and automated calibration software specifically designed for use by metrology laboratories. Visit **ni.com/calibration**.

Repair and Extended Warranty

NI provides complete repair services for our products. Express repair and advance replacement services are also available. We offer extended warranties to help you meet project life-cycle requirements. Visit **ni.com/services**.

NATIONAL INSTRUMENTS

ni.com • 800 813 3693
National Instruments • info@ni.com

351581A-01 2008-9328-151-101-D

Bijlage 4: VTC 3300 datasheet

Embedded Server Intel Pentium M 1.8 Ghz w/VGA/LVDS, LAN, 1xPCMCIA, 2xCF Socket, Mini-PCI, 3xUSB, 4xCOM, Audio, 1x2.5" Drive Bay, +6...+36V DC Power Input, +5VDC, +12VDC Power Out

- CPU Board Form Factor: Special
- Processor (CPU): Intel Pentium M 1.8 Ghz
- Processor (Number of CPU): 1
- Processor (CPU Socket): 478
- System Memory (Maximum Memory Size): 1 GB
- Ethernet (Ethernet Channels): 1

Construction: Aluminium Chassis

Mounting Configuration: DeskTop, Wall

CPU Board Form Factor: Special

Processor:

- CPU:Intel Pentium M 1.8 Ghz
- Number of CPU:1

Chipset: Intel 855GM

System Memory:

- Maximum Memory Size:1 GB
- *1 gig of RAM included
- Memory Type:DDR266
- Memory Sockets:1xSODIMM 200pin
- ECC:No

VGA:

- VGA Controller:Integrated with Chipset
- Channels:1
- Interface:VGA, LVDS
- Connectors:DB15, DB26 (LVDS)

Ethernet:

- Ethernet Channels:1
- 100Mbps:1
- Connector:RJ45

IDE:

- Channels:1xIDE

COM:

- Ports, total:4
- RS-232 Ports:2
- RS-485/422 Ports:2

USB:

- Ports:3
- Specification:Rev. 2.0

PCMCIA:

- Slots:1

CompactFlash:

- Slots:2

Audio: Built-in Chipset

Additional Interfaces: Digital I/O (DB9)

Extension Slots:

- PC/104: 1 (PC/104+)
- Mini-PCI:1

2.5" Drive Bays: 1 (HDD) 80 gb

Front Panel

Switches: 1xOn/Off, 1xReset

LEDs: 1xOn/Off LED, 1xHDD LED, GPIO LEDs

Connectors: USB, PCMCIA, CompactFlash, RJ11

Rear Panel

Connectors: DB15 VGA, DB26 (LVDS), RJ45 Ethernet, 3xDB9, 2xCOM (Screw Terminal), 2xUSB, Mic-in, Line-in, Line-out, 3pin Power-In, Power-Out

Cooling

Fans: 2 Fans beneath chassis

Power Supply

Input Voltage Range:

- DC: 6...36 V

Operation:

- Temperature: -10...+50 (with CompactFlash)
- Humidity:10...90 %
- Vibration: MIL-STD-810F, Method 514.5, Category 20

Storage:

- Temperature: -20...+80
- Humidity:10...90 %

Dimensions and Weight

Dimensions:

- Width:260 mm
- Height:70 mm
- Depth:176 mm

Certifications: CE, FCC, e Mark

Standard Compliance: RoHS

Bijlage 4: Reclamefiche voor NIdays