Specificatie en implementatie van protocollen in VLSI

Beulen, A.J.F.

Award date:
1990

FACULTEIT DER ELEKTROTECHNIEK
TECHNISCHE UNIVERSITEIT EINDHOVEN
VAKGROEP DIGITALE SYSTEMEN EB

SPECIFICATIE EN IMPLEMENTATIE VAN PROTOCOLLEN IN VLSI

door A.J.F. Beulen

Verslag van afstudeerwerk
Augustus 1990
Afstudeerhoogleraar : prof. ir. M.P.J. Stevens
Begeleider : ir. F.P.M. Budzelaar

SAMENVATTING

Dit verslag beschrijft het ontwerp van een ISDN datalink protocol controller. Het ontwerp wordt uitgevoerd met de Real-Time ontwerpmethode van Hatley en Pirbai. Doel was onderzoeken van de bruikbaarheid van deze methode voor het ontwerpen van hardware. Hierbij bleek dat deze methode wel geschikt is voor hardware ontwerp. Er moeten dan wel strikte afspraken voor het beschrijven van de informatie stromen tussen processen gemaakt worden.

LIJST VAN GEBRUIKTE AFKORTINGEN

C : Command

CCD : Control Context Diagram

CCITT : International Consultative Committee on Telegraphy and Telephony

CFD : Control Flow Diagram

CRC : Cyclic Redundancy Check

CSPEC : Control Specification

DCD : Data Context Diagram

DFD : Data Flow Diagram

DL : Datalink

F : Final

FD : Flow Diagram

FRMR : Frame reject response

HDLC : High-Level Data Link Control

I : Information transfer format

ISDN : Integrated Services Digital Network

ISO : International Organization for Standardization

LAPD : Link Acces Protocol D-Channel

M : Management

MDL : Management Datalink Layer

OSI : Open Systems Interconnection

P : Poll

Ph : Physical

PSPEC : Proces Specification

P/F : Poll of Final

R : Response

RD : Requirements Dictionary

REJ : Reject command/response

RNR : Receiver not ready command/response

RR : Receiver ready command/response

S : Supervisory format

SAP : Service Acces Point

Sapi : Service Acces Point Identifier

SIx : SI0 of SI1

Tei : Terminal End Point Identifier

Rec : Received

U : Unnumbered format

UA : Unnumbered acknowledgement response

UI : Unnumbered information command

INHOUD

1. INLEIDING

Bij de vakgroep Digitale Systemen (EB) van de faculteit Electrotechniek aan de TU Eindhoven houdt men zich ondermeer bezig met onderzoek naar ontwerpmethoden. Tevens worden hulpmiddelen die voor het gebruik van deze ontwerpmethoden gebruikt worden ontworpen. Doel van dit onderzoek naar ontwerpmethoden is het verlichten van de taak van de ontwerper.

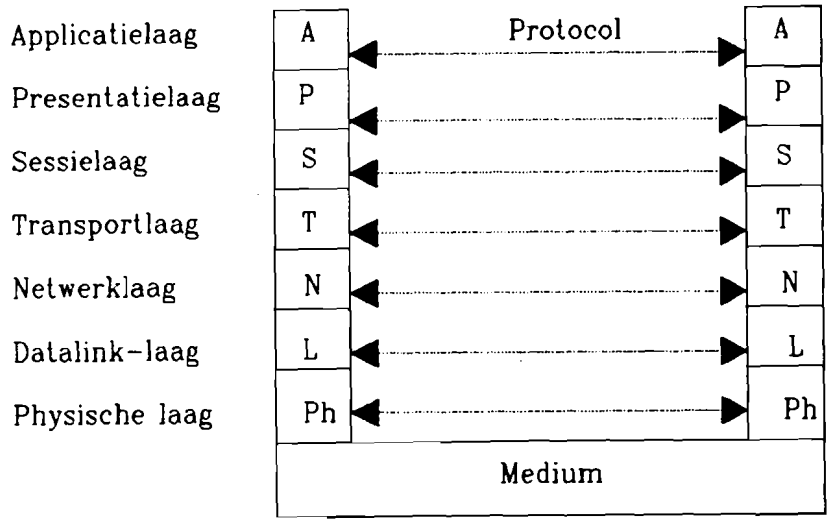
Een methode die hierbij tegen het licht wordt gehouden is de methode van Structured Analysis and Structured Design. Deze methode vindt zijn oorsprong in het ontwerpen van systemen die op software zijn gebaseerd. De vraag is nu of deze methoden ook toegepast kunnen worden bij het ontwerpen van hardware systemen.
Om dit te onderzoeken zullen een aantal ontwerpen met de methode gemaakt moeten worden. Een van die ontwerpen is de Datalink Controller waarvan het ontwerp in dit verslag wordt gepresenteerd. Dit gebeurt in hoofdstuk 4. Hoofdstuk 3 geeft een korte inleiding in de gebruikte ontwerpmethode. Dit is de ontwerpmethode voor Real-Time systemen van Hatley en Pirbai. Een inleiding met betrekking tot het OSI lagenmodel en de datalink laag is te vinden in hoofdstuk 2. De ervaringen opgedaan bij het gebruik van de methode en de hieruit te trekken conclusies vormen hoofdstuk 5.

2. HET OSI MODEL

Bij de beschrijving van ISDN maakt de CCITT gebruik van het OSI lagen model. Hoe de OSI lagen structuur er uit ziet en hoe de communicatie plaats vindt wordt hierna besproken. Daarna wordt de datalink laag aan de orde gesteld.

2.1. De lagen

De door OSI gedefinieerde lagen structuur bestaat uit zeven lagen (figuur 2.1). Voor de entiteiten in de zelfde laag lijkt het of ze rechtstreeks met elkaar communiceren.



figuur 2.1 : OSI-referentiemodel

Het protocol beschrijft hoe een entiteit communiceert met zijn peer entiteit. Deze communicatie zal niet rechtstreeks plaatsvinden, ze gebeurt namelijk door gebruikmaking van de eigenschappen van de onderliggende lagen. Deze bieden een service aan de hogere lagen [V&L]. Deze communicatie met de onderliggende lagen vindt plaats door middel van primitieven. Dit zijn opdrachten die al dan niet vergezeld van parameters tussen de twee lagen uitgewisseld worden. Hoe deze elementen geimplementeerd [Svo] moeten worden is hiermee niet beschreven.

2.2. De primitieven

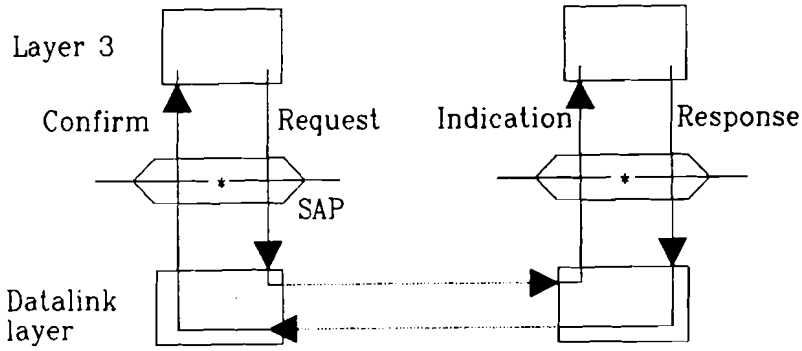
Binnen het OSI model zijn vier soorten primitieven gedefinieerd. Dit zijn: REQUEST, INDICATION, RESPONSE en CONFIRM. Hun functie word aan de hand van figuur 2.2 [I440] uit gelegd.
De functies van de primitieven zijn:

REQUEST wordt gebruikt door een hogere laag om een service van de onderliggende laag aan te vragen.

INDICATION wordt gebruikt om de bovenliggende laag op de hoogte te stellen dat een peer REQUEST geserviced moet worden.

RESPONSE bevestigd de ontvangst van een INDICATION aan de onderliggende laag.

CONFIRM geeft aan dat de met REQUEST aangevraagde service geleverd is.



figuur 2.2 : Illustratie primitieven

2.3. De datalink laag

De datalink laag stelt aan laag 3 een transparant foutvrij kanaal voor de overdracht van data pakketten terbeschikking. Hiervoor wordt gebruik gemaakt van de functies die laag 1 hiervoor aanbiedt (figuur 2.3) [I440]. De primitieven die hiervoor noodzakelijk zijn worden gegeven in tabel 2.1. Hoe deze primitieven gerealiseerd worden komt later aan de orde. De informatie die laag drie aanbiedt wordt verstuurd als HDLC [Sta,Tan] frames. Welke frametypes gebruikt worden is te zien in paragrafen 3.5 en 3.6 van I.441 [I440].

2.4. Gedrag van laag twee

Op welke wijze data uitwisseling mogelijk is hangt af van de toestand (figuur 2.4) [I440] waarin laag twee zich bevindt. Zolang de TEI nog niet vastgesteld is kunnen slechts UI management frames verzonden worden. Als de TEI wel vast staat is unacknowledged data transfer mogelijk met UI frames. Het opzetten van een multiple frame of een single frame datalink maak acknowledged data transfer op respectievelijk multiple en single frame basis mogelijk.

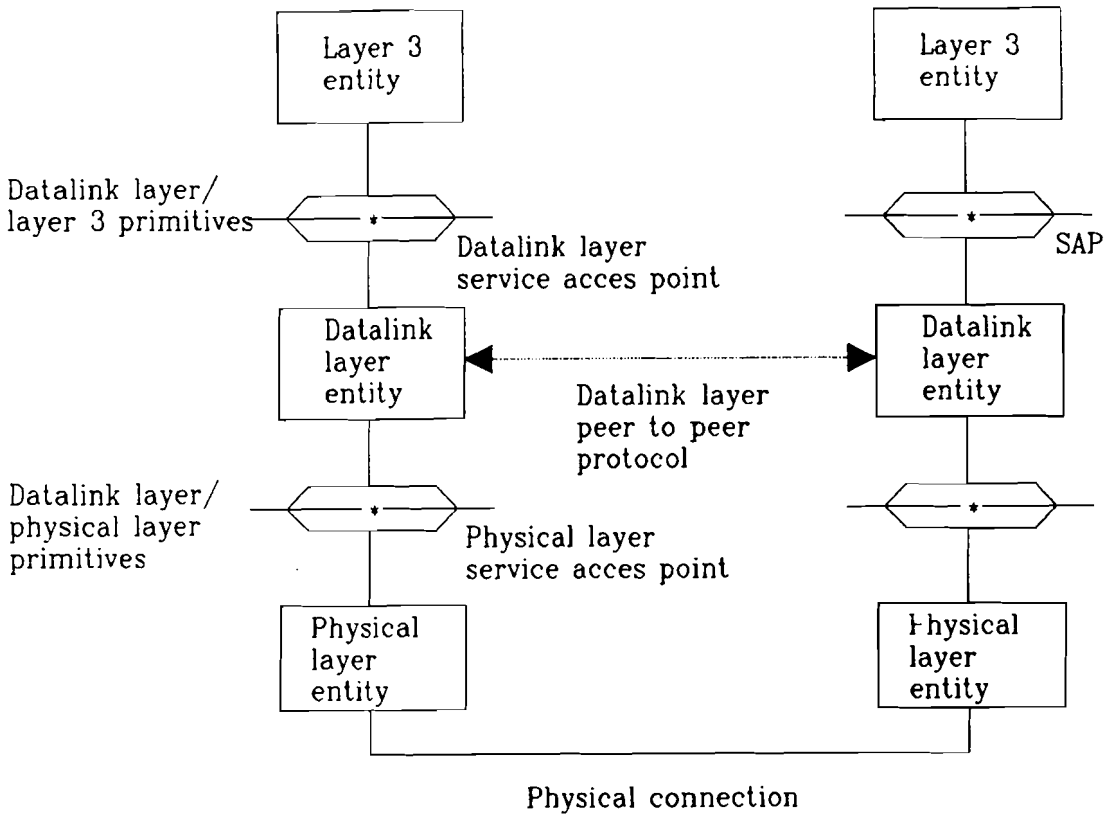
2.4.1.TEI Assignment

De TEI is naast de SAPI een van de componenten van het adres veld in een HDLC frame. Hun betekenis wordt in [I440] uitvoerig beschreven. Om datatransfer voor een user mogelijk te maken moet de TEI eerst vastgesteld worden. De MDL-ASSIGN

tabel 2.1 : primitieven van de datalink laag

soort	request	indication	response	parameters
L3 <--> L2				
DL-ESTABLISH	x	x		keuze single/ multiple frame overdracht
DL-RELEASE	x	x		keuze single/ multiple frame overdracht
DL-DATA	x	x		netwerk laag data
DL-UNIT DATA	x	x		netwerk laag data
M <--> L2				
MDL-ASSIGN	x	x		TEI waarde
MDL-REMOVE	x	x		TEI waarde
MDL-ERROR		x	x	oorzaak van fout
MDL-UNIT DATA	x	x		management data
L2 <--> L1				
Ph-DATA	x	x		datalink data
Ph-ACTIVATE	x	x		
Ph-DEACTIVATE	x	x		

primitieve initieert deze TEI verificaties. Met deze primitieve kan de gewenste TEI waarde aan gegeven worden. Hierna wordt een UI frame met de in tabel 2.2 gegeven structuur verzonden. Het gebruik van de management identifier is nog niet vastgelegd. Het referentienummer is een random getal dat voor de duur van de TEI vast stelling als identificatie dienst doet. Message type geeft het type management informatie aan. De action indicator heeft de waarde van de gewenste TEI of de toegewezen TEI. De details voor wat betreft de uitwisseling van de frames en de primitieven zijn te vinden in [I440].



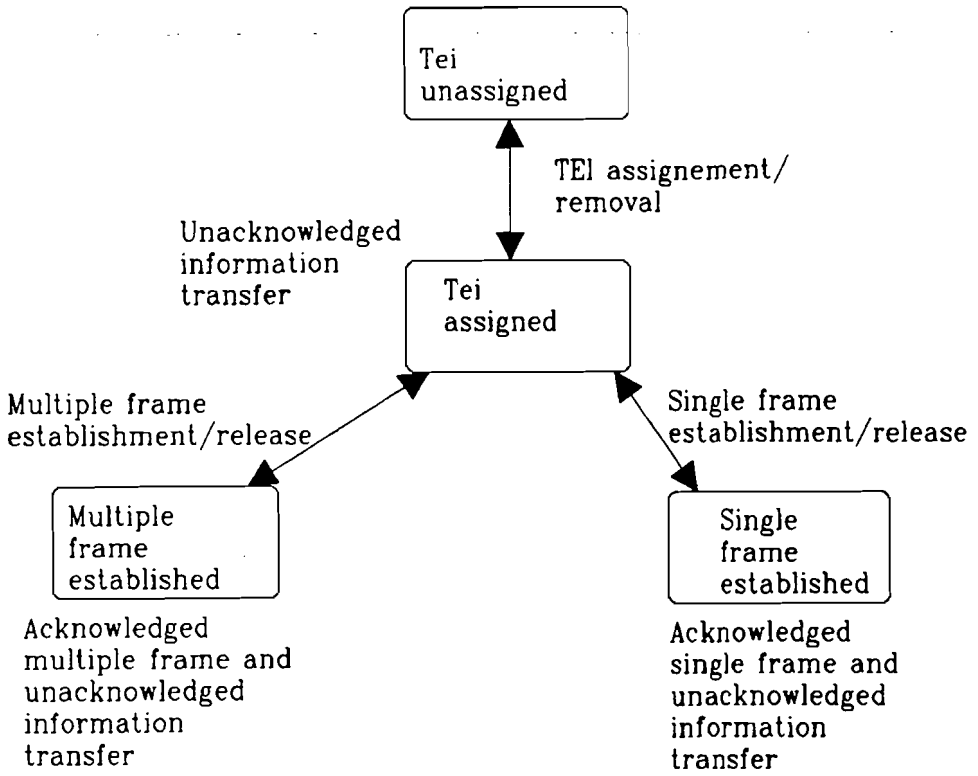
figuur 2.3 : Datalink laag en aangrenzende lagen

Tabel 2.2 : Inhoud van boodschap bij TEI verificatie

byte 1	Management entity identifier	1 byte
byte 2	Referentie nummer	2 bytes
byte 3	Message type	1 byte
byte 4	Action indicator	1 byte

2.4.2.Single frame data transfer

Single frame data transfer vindt plaats als met de DL-ESTABLISH primitieve wordt aangegeven dat deze wijze van overdracht gewenst is. Bij single frame data transfer vindt acknowledged data transfer plaats. Voor de data overdracht worden SIx command en response frames gebruikt. Deze frames worden ook gebruikt bij de initialisatie en bij het opzetten en verbreken van single frame data transfer. Bij de initialisatie wisselen de twee partners hun huidige toestand uit. Deze kan dan bij het opzetten en gebruiken



figuur 2.4 : Toestand laag twee en mogelijke data transfer

van de datalink gebruikt worden. Het opzetten van een link gebeurt met SIx frames die geen informatie deel bevatten. Meer details over single frame overdracht worden gegeven is [I440]. De flow control vindt hier plaats volgens de start stop methode.

2.4.3.Multiple frame data overdracht

Bij een DL-ESTABLISH primitieve kan worden aangegeven dat de overdracht op multiple frame basis moet plaats vinden. Het opzetten van de link gebeurt met SABM,SABME frames. Als deze met een UA frame bevestigd zijn is de datalink opgezet. Dan kan de overdracht van data over de link plaats vinden. Dit gebeurt met I frames. Behalve I frames is het ook mogelijk dat RNR, RR of REJ frames verzonden worden. Deze frames hebben een functie bij de flow control die volgens het sliding window mechanisme werkt, en bij het herstellen van fouten die opgetreden zijn. Het verbreken van de multiple frame link gebeurt na een aanvraag met DL-RELEASE. Op frame niveau heeft dit een uitwisseling van een DISC frame en een UA of DM frame tot gevolg. Hierover is nadere informatie te vinden in [I440].

2.4.4.Gedrag bij fouten

Bij de boven gegeven beschrijving is er van uit gegaan dat de overdracht foutloos verloopt. In de praktijk kunnen er echter wel fouten optreden. Binnen de datalink laag

wordt onderscheid gemaakt tussen twee soorten fouten:

-door hertransmissie herstelbare fouten
-niet door hertransmissie herstelbare fouten.

Het eerste type is, zoals de naam al aangeeft, te herstellen door het frame waarin de fout is opgetreden nog een keer te verzenden. Fouten van dit type zijn:

-fout in het volgorde nummer van een frame
-fout in CRC
-aankomst van een ongeldig frame

Bij het optreden van onherstelbare fouten zal laag twee een herstart procedure van de link uitvoeren. Fouten van dit type zijn:

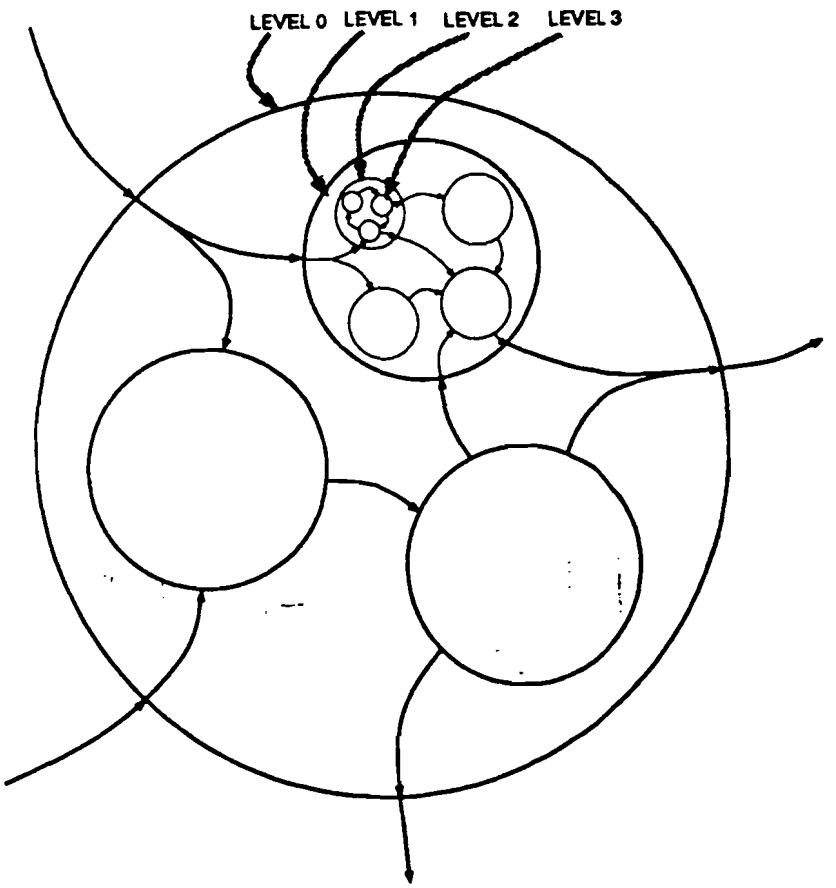
-een ontvangst bevestiging die niet in het window ligt
-een maximaal aantal hertransmissies heeft plaats gevonden bij het opzetten van de link
-een niet verwacht frame is aangekomen
-een FRMR frame signaleert dat de peer een onherstelbare fout heeft gedetecteerd.

Voor details wordt weer verwezen naar [I440].

3. ONTWERPMETHODE VAN HATLEY AND PIRBHAI

De ontwerpmethode zoals die wordt beschreven door Hatley en Pirbhai [H&P] wordt gebruikt voor het ontwerp van Real-Time systemen gebaseerd op software. De grotere ingewikkeldheid van de te ontwikkelen systemen maakte een methode voor het ontwerpen noodzakelijk. Een methode voor het ontwerpen van systemen is structured analysis. Deze beschrijf voornamelijk de wijze waarop informatie verwerking plaats vindt. Hatley en Pirbhai hebben deze methode uitgebreid met een state machine benadering. Deze wordt gebruikt om de besturing van de informatie verwerkende processen te beschrijven. Doel van de methode is een eenduidige beschrijving van een systeem te geven. Om dit te bereiken wordt het systeem opgedeeld in subprocessen. Hierbij worden niveaus gebruikt. Op elk dieper niveau worden meer details beschreven (figuur 3.1). Om deze opsplitsing te realiseren worden twee modellen opgesteld. Namelijk het Requirements Model en het Architecture Model. Het Requirements Model moet beschrijven wat een systeem moet doen. Het Architecture Model beschrijft hoe het ontwerp gestructureerd moet worden. Het opstellen van beide modellen moet parallel en in onderling wisselwerking gebeuren.

Een kort overzicht van de methode van Hatley en Pribhai [H&P] volgt nu. Hierbij zal



figuur 3.1 : Subprocessen op verschillende niveaus

de door H&P gebruikte Engelstalige terminologie gebruikt worden.

3.1. Het Requirements Model

Het Requirements Model beschrijft wat een systeem moet doen. De belangrijkste hulpmiddelen hierbij zijn de Flow Diagrams. Een Flow Diagram is een grafische weergave van informatie stromen (flows) en processen die deze flows verwerken. Er zijn twee te onderscheiden flow diagrammen mogelijk: Data Flow Diagrams (DFD) en Control Flow Diagrams (CFD). De andere elementen komen nog aan de orde.

De merites van de methode zijn:

-het is een abstracte beschrijving van de functionele eisen van het systeem.
-er wordt gebruik gemaakt van een grafische weergave waardoor de begrijpelijkheid verhoogd wordt.
-het voldoet aan de eisen die bij het ontwerpen van grote systemen gesteld worden.
-het opgestelde model is door zijn opbouw consistent.
-de indexering van de componenten vindt van zelf plaats.

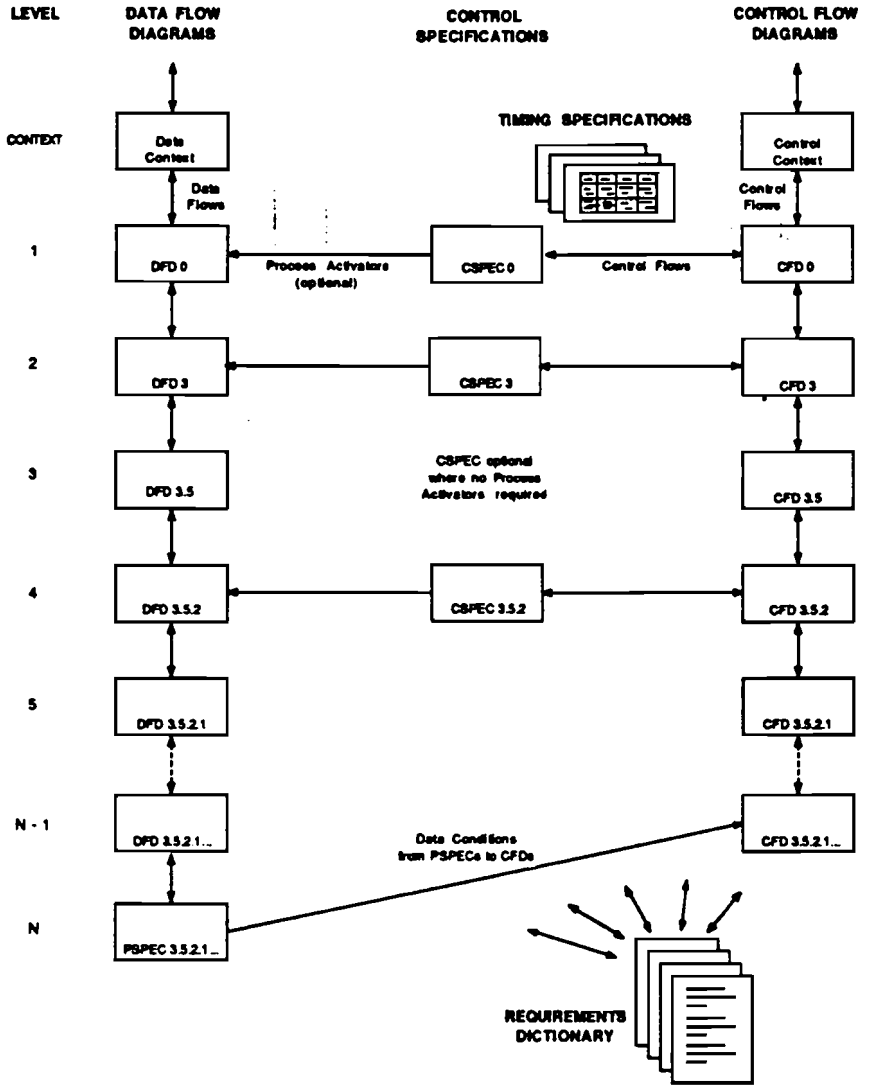
Behalve de reeds genoemde flow diagrammen komen in het Requirements Model de volgende componenten voor: Data Context Diagram (DCD), Control Context Diagram (CCD), Proces Specification (PSPEC), Control Specification (CSPEC), Timing Specification en Requirements Dictionary (RD). Hier is dus een scheiding gemaakt tussen data en control items. Data flows worden gebruikt om informatie overdracht weer te geven. Control flows geven aan dat het informatie voor de besturing betreft. Een uitgebreidere beschrijving van data en control flows wordt nog gegeven.

Hoe deze componenten onderling samen hangen wordt aangegeven in figuur 3.2.
In het Data Context diagram en het Control Context Diagram worden de flows gegeven die het systeem verbinden met de omgeving. Het proces in deze Context Diagrams zal opgedeeld worden in subprocessen die worden gegeven in flow diagram FD0. Deze processen laten gedetailleerder zien hoe het systeem opgebouwd is. Deze processen zullen op hun beurt weer opgesplitst worden in subprocessen. Dit wordt weer gegeven in de flow diagrammen van een lager niveau. Als een niveau bereikt is waarbij de processen zo eenvoudig zijn dat ze niet meer opgesplitst kunnen worden, dan wordt dit proces in een Process Specification geschreven. Voor Control Flow Diagrammen wordt op elk niveau een beschrijving in de vorm van CSPEC's gegeven.

In de linker kolom in figuur 3.2 worden gegeven: Data Context Diagram, Data Flow Diagrams en Process Specifications. Deze componenten vormen het Process Model. Ze beschrijven de informatie verwerking binnen het systeem. De rechter kolom bestaat uit: Control Context Diagram en Control Flow Diagrammen. Deze vormen samen met de Control Specifications in de middelste kolom het Control Model. Hierin wordt beschreven hoe de besturing van de processen binnen het systeem gerealiseerd wordt. Deze twee modellen komen nu nader aan de orde.

3.1.1.Het Process Model

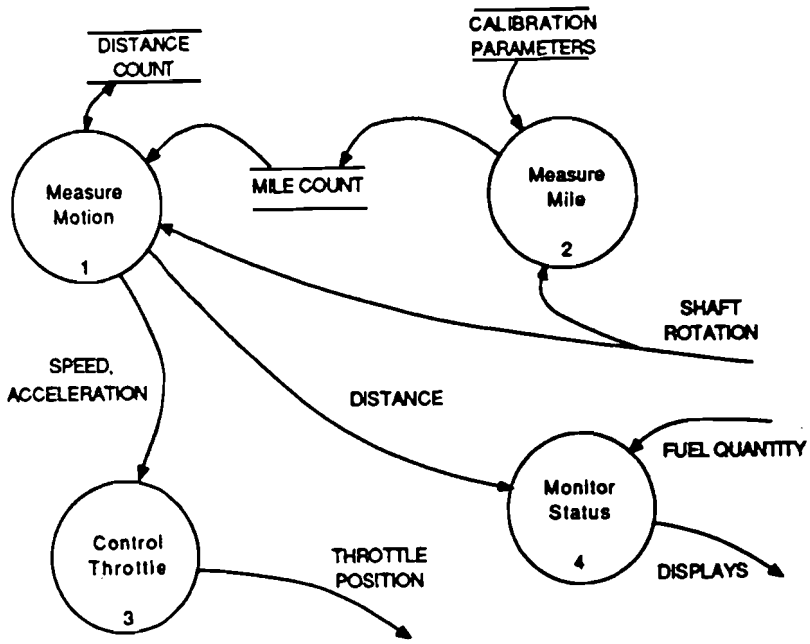
Het Process Model bestaat uit Data Context Diagram (DCD), Data Flow Diagrams (DFD) en Proces Specifications (PSPEC).



figuur 3.2 : Samenhang van de elementen uit het requirements model

De elementen die in de Data Flow Diagrammen (figuur 3.3) kunnen voorkomen zijn: processen, data flows en data stores.

Processen : transformeren de inkomende data flows in de uitgaande data flows. Dit gebeurt volgens het model oneindig snel. Tevens worden de subprocessen parallel uitgevoerd. Als processen niet parallel uitgevoerd mogen worden dan kan de volgorde door het



figuur 3.3 : voorbeeld van een Data Flow Diagram

opnemen van "next :" in de beschrijving worden vast gelegd. Processen worden in diagrammen met een cirkel aangegeven.

Data flows : worden gebruikt om aan te geven hoe informatie tussen de processen uitgewisseld moet worden. De richting wordt aan gegeven met een pijl. Flows kunnen informatie overdracht tussen processen in hetzelfde diagram verzorgen. Het is echter ook mogelijk dat pijlen aan een einde niet met een proces verbonden zijn. Hiermee wordt aangegeven dat de flow van een hoger niveau komt of naar een hoger niveau gaat.
Data flows kunnen voor wat hun waarde betreft continue of discreet zijn. In de tijd beschouwd kunnen ze ook continue of discreet zijn. Tijd discrete flows worden transiënt flows genoemd. Ze worden door een proces gegenereerd door aan te geven: "issue <dataflow> =". Tijd continue flows worden aangegeven door "set <dataflow> =". <dataflow> is hierbij de naam van de flow. Gebruik van de notatie: "naam1(naam2)" geeft aan dat de flow naam1 deel uitmaakt van de flow naam2 en dat deze aan de periferie is opgesplitst.

Datastores: behouden de waarde die door een flow is aangeboden totdat een andere waarde wordt aangeboden. Dit in tegenstelling tot flows die hun waarde slechts tijdelijk hebben. De in- en uitgaande flows van stores kunnen in ander diagrammen als gewone data flows gebruikt worden. Een store mag slechts in een enkel diagram gebruikt worden. Stores worden in de diagrammen

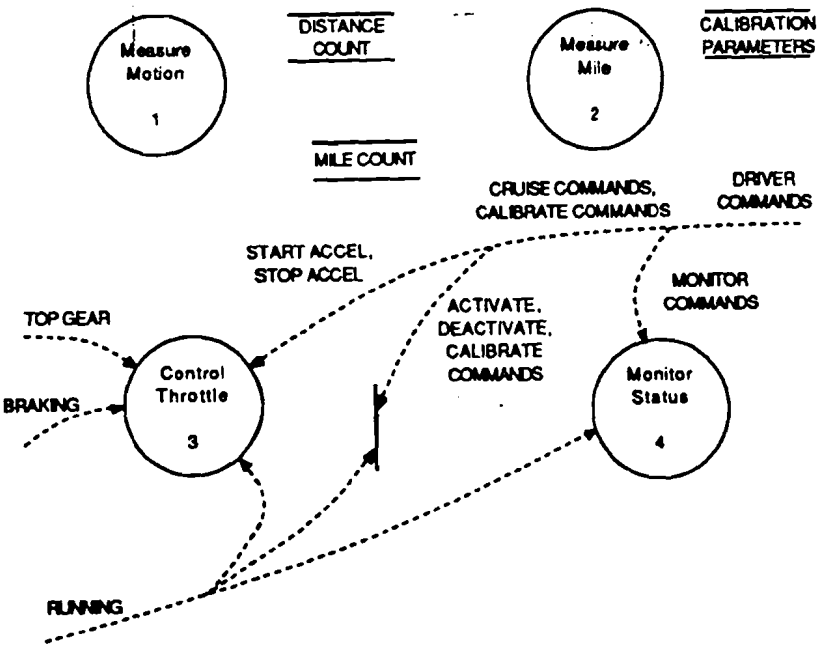
aangegeven met twee evenwijdige lijnen waartussen de naam van de opgeslagen informatie flows staat.

Het Data Context Diagram (DCD) is een speciaal Data Flow Diagram. In het DCD komt slechts een enkel proces voor. Dit proces is het gehele te realiseren systeem. Van dit systeem wordt aangegeven hoe de data flows naar en van de omgeving er uit zien. Deze omgeving wordt weergegeven als terminators. In het DCD zijn deze te herkennen als rechthoeken.

Process Specifications vormen het laatste onderdeel van het Process Model. Als de opdeling van processen door decompositie van processen een niveau heeft bereikt waarop de processen op eenvoudige wijze te beschrijven zijn, dan wordt deze beschrijving in de vorm van een PSPEC gegeven. Een PSPEC zal gewoonlijk een beschrijving in Structured English bevatten. Het is echter ook mogelijk dat naar een algemeen bekent algorithme wordt verwezen. Het is ook toegestaan om op grafische wijze de werking van een proces te verduidelijken. Als het doel van de PSPEC, namelijk het eenduidig vastleggen van de functie eisen maar bereikt wordt. Voor de duidelijkheid kan ook nog commentaar aan PSPEC's toegevoegd worden. Dit gebeurt door het commentaar tussen sterretjes in de beschrijving op te nemen, dus: " *<commentaar>* ".

3.1.2.Het Control Model

Het Control Model beschrijft hoe de besturing van de processen plaatsvindt. Hiervoor wordt gebruik gemaakt van: Control Context Diagram (CCD), Control Flow Diagram (CFD) en Control Specifications (CSPEC).



figuur 3.4 : Voorbeeld van een Control Flow Diagram

De Control Flow Diagrammen (figuur 3.4) worden gebruikt om grafisch de samenhang tussen de verschillende elementen van een CFD weer te geven. Deze elementen zijn: processen, CSPEC bar's, Control Stores en Control flows.

Processen : zijn reeds beschreven bij het Process Model.

Control flows : worden gebruikt voor het besturen van processen. Bij control flows kunnen twee soorten flows onderscheiden worden, namelijk: process activators en toestand flows. Met de process activators wordt bepaald of een proces al dan niet geactiveerd wordt. Dit kan op twee manieren gebeuren. Zo is het mogelijk om een proces expliciet te activeren en te deactiveren. Verder is het mogelijk om een proces te triggeren en op deze wijze te activeren. Uit het voorgaande blijkt dat control flows ook een verschillend karakter kunnen hebben. Ze kunnen namelijk continue of transiënt zijn. Voor wat hun waarde betreft kunnen control flows slechts discreet zijn. Control flows worden in de diagrammen met een gestippelde pijl aangegeven. Ze kunnen op de zelfde wijze gebruikt worden als data flows. Control flows kunnen ook door processen gegenereerd worden. Er is dan sprake van Data Conditions.

Control Stores : hebben een zelfde functie als de Data Stores nu echter voor control flows. Verder gelden voor het gebruik er van de zelfde regels als voor data stores.

CSPEC Bar's : geven aan dat de control flows die er naar toe gaan gebruikt worden om de uitgaande flows te genereren. In de CSPEC wordt beschreven hoe deze flows uit de flows aan de ingang wordt gegenereerd. In een CFD kunnen meerdere CSPEC Bar's voor komen, deze worden echter allemaal in de bij dat diagram behorende CSPEC beschreven.

Het Control Context Diagram (CCD) is een bijzonder soort CFD. Hierin wordt namelijk aangegeven hoe het proces met control flows met de omgeving communiceert. Het proces en de terminators in het CCD zijn de zelfde als die uit het DCD.

In de Control Specifications (CSPECS) wordt beschreven hoe de uitgaande control flows gegenereerd worden uit de inkomende flows. Deze uitgaande flow kan een combinatorische functie zijn van de binnen komende flows. Het is echter ook mogelijk dat een state machine beschrijving nodig is om de flow aan de uitgang te genereren. In de CSPEC kunnen derhalve naast eenvoudige beschrijvingen in Structured English ook tabellen, figuren, state diagrammen en state tables voorkomen. Welke representatie gekozen wordt is niet zo zeer van belang mits de beschrijving de functies die vervult wordt maar goed weergeeft.

3.1.3.Completering van het Requirements Model

Om de beschrijving van het Requirements Model volledig te maken moet nog twee onderdelen van het model aan de orde komen. Dit zijn: de Timing Requirements en het Requirements Dictionary (RD).

Het Requirements Model legt slechts vast wat er moet gebeuren. De manier waarop dit moet gebeuren is niet van belang. Derhalve hebben timing aspecten binnen het Requirements Model eigenlijk geen plaats. Timing aspecten zijn wel van belang bij de communicatie met de buiten wereld. Twee punten die door de buiten wereld opgelegd worden zijn: de herhaling frequentie van signalen en de response snelheid van uitgang signalen op ingang signalen. De gestelde eisen worden in de Timing Requirements vastgelegd.

In de vorige paragraaf zijn de PSPEC's en de CSPEC's genoemd als de plaatsen waar de operaties die op de flows uitgevoerd worden beschreven worden. De flows en de stores werden echter nog niet beschreven in het model. Dit gebeurt in het Requirements Dictionary (RD), hier worden alle flows en stores beschreven.

In het RD worden twee soorten flows onderscheiden: elementaire flows (primitive flows) en samengestelde flows (group flows). Group flows zijn samengesteld uit meerdere elementaire flows. Behalve de samenstelling worden in het RD ook nog enkele eigenschappen van flows genoemd, zoals: eenheden of waarde. Tabel 3.1 geeft een overzicht van de notaties die bij het opstellen van een RD worden gebruikt.

tabel 3.1 : Requirements Dictionary notaties

Symbool	(Betekenis)Beschrijving
=	(bestaat uit) geeft aan hoe een flow samengesteld is of welke waardes hij heeft.
+	(samen met) geeft aan dat flows samengevoegd worden
{ }	(iteraties) hiermee wordt het aantal keren dat een flow voorkomt aangegeven
[|]	(keuze) slechts een van de flows kan voor komen
( )	(optioneel) de flow kan maar hoeft niet voor te komen
" "	(letterlijk) de ingesloten tekst komt letterlijk in de flow voor
* *	commentaar
\ \	(primitieve) geeft aan dat een flow een primitieve flow is

3.2. Architecture Model

Het Architecture model beschrijft hoe de functies die in het Requirements Model gedefinieerd zijn gerealiseerd moeten worden. Hiervoor geven H&P de volgende hulpmiddelen: Architecture Context Diagram, Architecture Flow Diagram, Architecture Module Specification, Architecture Interconnect Diagram, Architecture Interconnect Specification en Architecture Dictionary. Aangezien echter bij hardware ontwerp de opdeling in fysieke modules reeds als onderdeel van het opstellen van het Requirements model plaatsvindt, is het niet nodig deze opdeling nog eens expliciet aan de orde te laten komen. Het Architecture model wordt daarom hier ook niet verder besproken.

3.3. Gebruikte conventies

Bij het ontwerpen van de datalink controller wordt er op sommige plaatsen afgeweken van de conventies die H&P gebruikten. Dit wordt gedaan om de duidelijkheid te vergroten. De afwijkingen worden hier na beschreven.

3.3.1.Flow diagrammen

Het Control Flow Diagram en het Data Flow Diagram van een bepaald niveau worden samen gevoegd tot een Flow Diagram (FD). Het zelfde geldt ook voor de twee context diagrammen die samengevoegd worden tot een Context Diagram (Context). De genoemde samenvoegen wordt gedaan om de verwarring die kan ontstaan bij het gebruik van gescheiden flow diagrammen te voorkomen.

3.3.2.Control processen in het flow diagram

Control processen worden niet met bar's maar met gestippelde bollen weergegeven [W&M]. Dit heeft als voordeel dat een naam en nummer toegekend kan worden. Hiermee is het mogelijk om processen die bij H&P uit de CSPEC gevist moesten worden duidelijk aan te wijzen. De CSPEC's hebben nu namelijk een label.

3.3.3.Proces activering

De proces activering vanuit een control proces wordt gebruikt naast een andere methode. Hierbij wordt een proces geactiveerd als een juiste flow aangeboden wordt.

3.3.4.Het gebruik van proces beschrijvingen

Bij de methode van H&P zijn slechts de proces beschrijvingen nodig bij processen die het laagste niveau van decompositie bereikt hebben. Het is echter handig om op hogere niveaus al inzicht te krijgen in het doel van processen. Om dit inzicht te krijgen wordt op elk niveau van elk proces al een korte beschrijving in normale taal gegeven. Deze proces beschrijving bestaat dan slechts uit commentaar.

3.3.5.Grens tussen requirements en design

Waar deze grens moet liggen wordt door H&P niet duidelijk aangegeven. Het hangt volgens hun af van de eisen die gesteld worden. Voor het ontwerp van hardware zal tot op een zeer laag niveau door gegaan moeten worden met decompositie. Dit niveau is bereikt als de processen gerealiseerd kunnen worden door gegeven elementaire componenten.

4. DATALINK LAAG CONTROLLER

In dit hoofdstuk wordt de decompositie van een datalink laag controller beschreven. De datalink laag controller vormt een point-point datalink controller [I440] die tevens is voorzien van laag niveau functies die voor het samenstellen en decoderen van HDLC frames zorgen. De Controller zal services verlenen aan een laag 3 entiteit (Host) en services vragen van een laag 1 entiteit (Modem). Behalve de protocol functies zullen ook enkele management functies door de controller vervult worden.

De flows die de controller gebruikt bij zijn communicatie met de omgeving worden eerst beschreven. Daarna worden de functies die de controller moet vervullen beschreven. De eerste stappen van de decompositie van de controller worden ook in dit hoofdstuk beschreven. De rest van de decompositie en de procesbeschrijvingen worden gegeven in appendix A. In appendix B wordt het Requirements Dictionary gegeven.

4.1. De flows

Tabel 2.1 geeft uitsluitsel over de primitieven die voor de Datalink laag van belang zijn. Voor de laag drie - laag twee interface zijn dit de DL primitieven. De Ph primitieven worden gebruikt voor informatie overdracht tussen laag twee en laag een. De communicatie tussen datalink management en de laag twee entiteit (DatalinkLayerController) gebeurt met MDL primitieven. Op welke wijze deze primitieven worden gerealiseerd door de flows van de controller wordt nu kort beschreven. Figuur 4.1 is het Context Diagram van de controller.

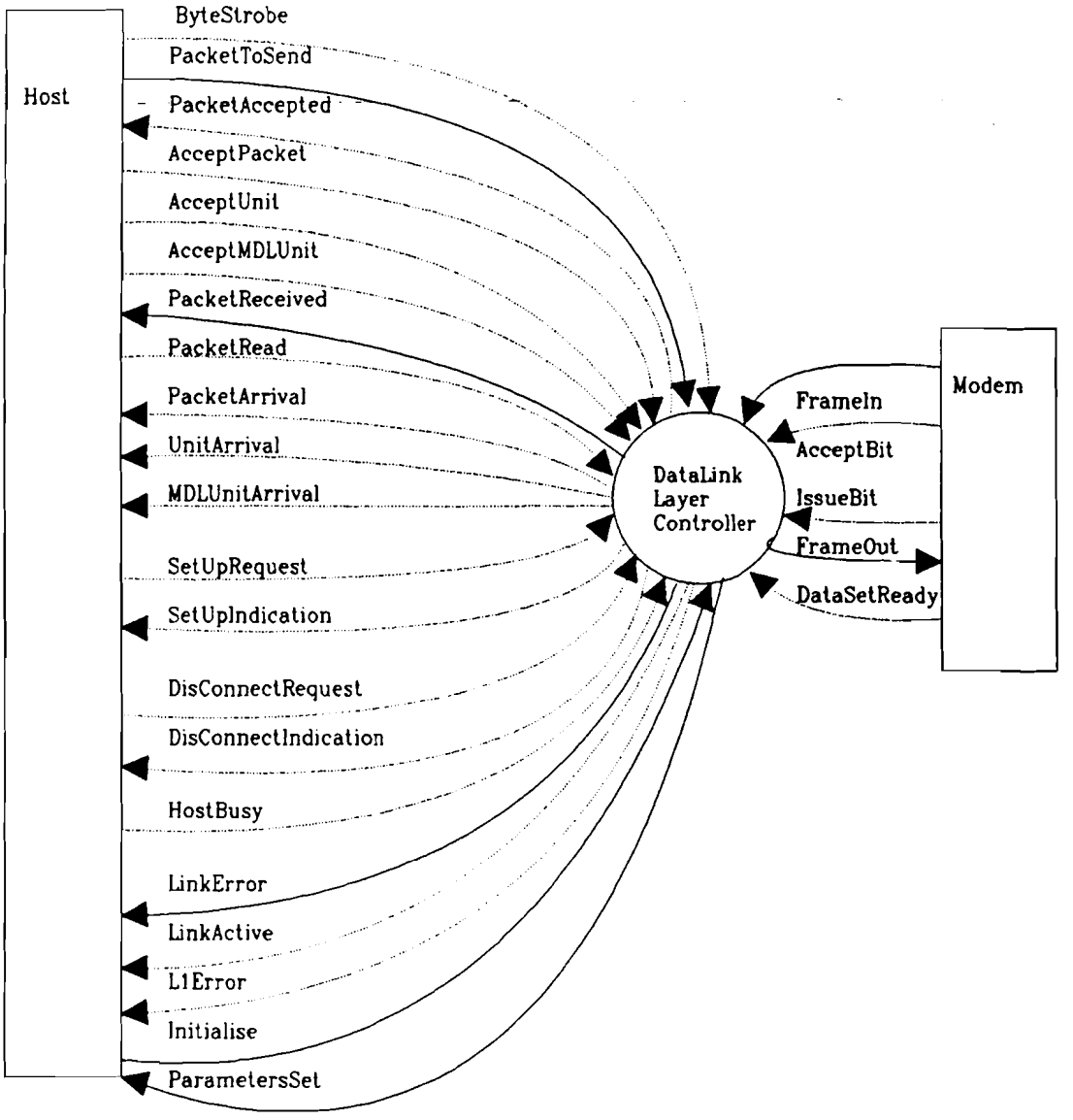
4.1.1.Flows tussen Modem en controller

De flows die de controller verbinden met de laag 1 entiteit (Modem) realiseren de primitieven als volgt:

Ph-DATA Indication
: De bits die de Ph-DATA Indication vormen, worden door het modem aan de controller aangeboden als FrameIn. AcceptBit geeft aan wanneer een bit van FrameIn geaccepteerd moet worden (figuur 4.2).

Ph-DATA Request
: De bits die Ph-DATA Request vormen verlaten de controller als FrameOut. IssueBit geeft aan wanneer bits in FrameOut van waarde moeten veranderen (figuur 4.2).

Ph-ACTIVATE, Ph-DEACTIVATE
: De Ph-ACTIVATE en Ph-DEACTIVATE primitieven worden niet als zodanig gerealiseerd. Hiervoor wordt DataSetReady gebruikt. Deze flow signaleert aan de controller of het modem wel of niet in staat is om de bits van de Datalink te transporteren. Dit gebeurt met de waarden ModemReady en ModemNotReady.



figuur 4.1 : Context Diagram

4.1.2.Flows tussen Host en controller

De communicatie tussen Host en DatalinkLayerController gebeurt met een groot aantal flows. Tabel 4.1 geeft een overzicht van deze flows, hun richting, hun type en van de primitieven die ze mede realiseren.

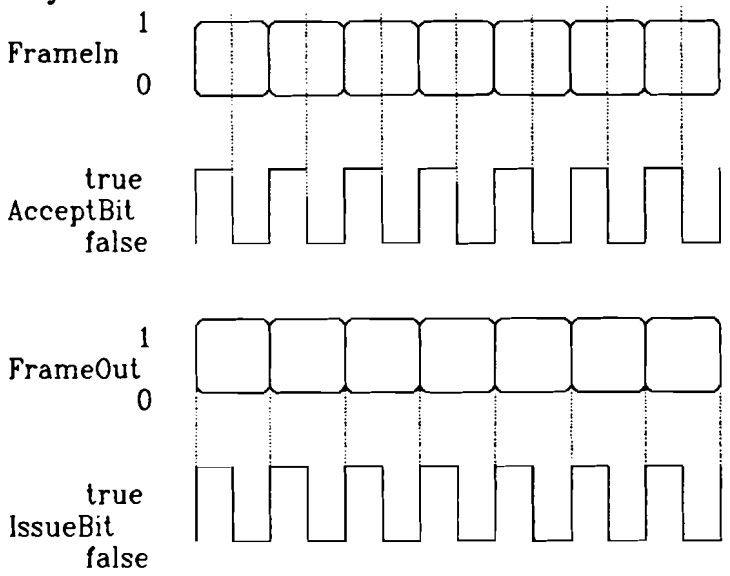
Hoe de DL primitieven zijn gerealiseerd wordt nu beschreven. Bij acknowledged data transfer worden de primitieven DL-DATA Request en DL-DATA Indication gebruikt.

DL-DATA Request

De DL-DATA Request wordt gerealiseerd door de flows: AcceptPacket, PacketAccepted, ByteStrobe en PacketToSend (figuur 4.3). Met AcceptPacket stelt Host de controller ervan op de hoogte dat er een packet klaar staat (PacketToSend) om verzonden te worden. Als controller het pakket accepteert wordt dit met PacketAccepted aan de host bevestigd. ByteStrobe geeft aan wanneer de bytes geldig zijn.

DL-DATA Indication

DL-DATA Indication wordt gerealiseerd door de flows: PacketArrival, PacketRead, ByteStrobe en PacketReceived (figuur 4.3). PacketArrival stelt de host van de aanwezigheid van een ontvangen packet (PacketReceived) op de hoogte. De host zal het accepteren van het packet aan de controller doorgeven met PacketRead. Wanneer de bytes doorgegeven moeten worden wordt bepaald door ByteStrobe.



figuur 4.2 : Verband tussen FrameIn, FrameOut en de clocks

De bovenstaande stromen worden gebruikt voor acknowledged datatransfer. Het is ook mogelijk om pakketten zonder bevestiging door de controller te laten transporteren. De DL-UNIT DATA primitieven waarmee deze service wordt aangevraagd, worden op de zelfde wijze gerealiseerd als de DL-DATA primitieven.

DL-UNIT DATA Request

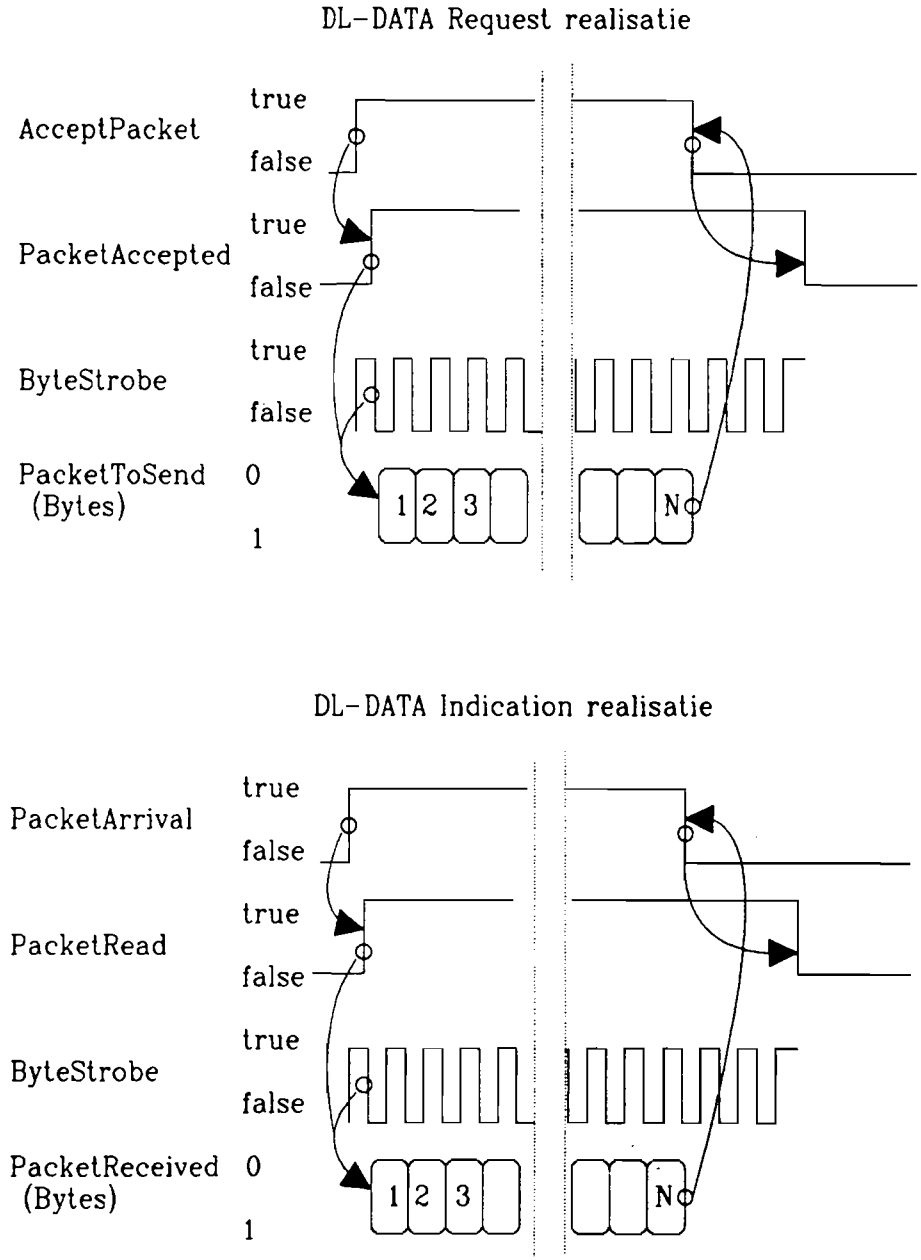
De DL-UNIT DATA Request wordt gerealiseerd door de flows: AcceptUnit, PacketAccepted, ByteStrobe en PacketToSend. AcceptUnit stelt de controller ervan op de hoogte dat er een packet klaar staat (PacketToSend) om verzonden te worden. Als dit packet door de controller geaccepteerd wordt, dan wordt dit door PacketAccepted aan de host bevestigd. ByteStrobe geeft aan wanneer de bytes geldig zijn.

DL-UNIT DATA Indication

Tabel 4.1 : Flows tussen host en controller

flow	richting	type	primitieve
PacketToSend	host->contr	D	DL-DATA Request DL-UNIT DATA Request MDL-UNIT DATA Request
PacketAccepted	contr->host	C	DL-DATA Request DL-UNIT DATA Request MDL-UNIT DATA Request
AcceptPacket	host->contr	C	DL-DATA Request
AcceptUnit	host->contr	C	DL-UNIT DATA Request
AcceptMDLUnit	host->contr	C	MDL-UNIT DATA Request
PacketReceived	contr->host	D	DL-DATA Indication DL-UNIT DATA Indication MDL-UNIT DATA Indication
PacketRead	host->contr	C	DL-DATA Indication DL-UNIT DATA Indication MDL-UNIT DATA Indication
PacketArrival	contr->host	C	DL-DATA Indication
UnitArrival	contr->host	C	DL-UNIT DATA Indication
MDLUnitArrival	contr->host	C	MDL-UNIT DATA Indication
SetUpRequest	host->contr	C	DL-ESTABLISH Request
SetUpIndication	contr->host	C	DL-ESTABLISH Indication
DisConnect Request	host->contr	C	DL-RELEASE Request
DisConnect Indication	contr->host	C	DL-RELEASE Indication
HostBusy	host->contr	C	DL-ESTABLISH Indication
LinkError	contr->host	D	MDL-ERROR Indication
LinkActive	contr->host	C	-
L1Error	contr->host	C	-
Initialise	host->contr	D	MDL-ASSIGNE Request

DL-UNIT DATA Indication wordt gerealiseerd door de flows: UnitArrival, PacketRead, ByteStrobe en PacketReceived. UnitArrival stelt de host van de aanwezigheid van een ontvangen packet (PacketReceived) op de hoogte. De host zal het accepteren van het packet aan de controller doorgeven met PacketRead. Wanneer een byte doorgegeven moet worden bepaalt ByteStrobe.



figuur 4.3 : Realisatie DL-DATA primitieven

Voor het opzetten van een datalink worden de primitieven DL-ESTABLISH Request en DL-ESTABLISH Indication gebruikt. Hun realisatie wordt hier onder beschreven.

DL-ESTABLISH Request

DL-ESTABLISH Request wordt gerealiseerd door SetUpRequest. Deze flow heeft twee waarden: Single en Multiple. Met deze waarden wordt aan gegeven of de datalink moet worden opgezet voor single of multiple frame operatie.

DL-ESTABLISH Indication

DL-ESTABLISH Indication wordt gerealiseerd door SetUpIndication. Deze flow heeft ook twee waarden (Single, Multiple) om aan de host door te geven of de data transfer in single of in multiple frame mode gebeurt. Als de host de link niet wil openen, dan kan hij dit met HostBusy signaleren.

Voor het verbreken van de datalink staan de primitieven DL-RELEASE Request en DL-RELEASE Indication terbeschikking.

DL-RELEASE Request

DL-RELEASE Request wordt gerealiseerd door DisConnectRequest. Deze flow heeft ook twee waarden namelijk Single en Multiple om aan te geven welk type data transfer afgebroken moet worden.

DL-RELEASE Indication

DL-RELEASE Indication wordt door DisConnect Indication aan de host gemeld. Dit gebeurt weer met gelijktijdig meegeven van de parameter Single of Multiple.

Een aantal flows is nu nog niet aan de orde gekomen. Dit zijn de flows: AcceptMDLUnit, MDLUnitArrival, ParametersSet, LinkError, LinkActive, L1Error, Initialise. Deze flows worden voor een deel gebruikt voor het realiseren van MDL primitieven (management). Tevens worden enkele van de flows gebruikt bij de initialisatie van de controller. De MDL-UNIT DATA Request en de MDL-UNIT DATA Indication die gebruikt worden voor de overdracht van peer to peer management informatie, worden op soortgelijke wijze als de DL-DATA primitieven gerealiseerd. Dit wordt nu beschreven.

MDL-UNIT DATA Request

De MDL-UNIT DATA Request wordt gerealiseerd door de flows: AcceptMDLUnit, PacketAccepted, ByteStrobe en PacketToSend. AcceptMDLUnit stelt de controller ervan op de hoogte dat er een packet klaar staat (PacketToSend) om verzonden te worden. Als dit packet door de controller geaccepteerd wordt, dan wordt dit door PacketAccepted aan de host bevestigd. ByteStrobe geeft aan wanneer de bytes geldig zijn.

MDL-UNIT DATA Indication

MDL-UNIT DATA Indication wordt gerealiseerd door de flows: MDLUnitArrival, PacketRead, ByteStrobe en PacketReceived. MDLUnitArrival stelt de host van de aanwezigheid van een ontvangen packet (PacketReceived) op de hoogte. De host zal het accepteren van het packet aan de controller doorgeven met PacketRead. Wanneer de bytes door gegeven moeten worden bepaalt ByteStrobe.

De flow LinkError kan gezien worden als de implementatie van de MDL ERROR Indication. Deze flow bevat namelijk informatie omtrent de opgetreden fouten tijdens het gebruik van de controller.

Met ParametersSet meldt de controller de gebruikte parameters bij de initialisatie aan de host. Deze flow vervult dus de functie van MDL ASSIGN Indication.

De Initialise flow vervult een dubbele functie. In de eerst plaats zorgt hij voor het overdragen van de MDL ASSIGN Request primitieve. Een tweede functie is het doorgeven van een aantal operationele parameters aan de controller. Deze parameters moet de controller voor in gebruik name kennen. Tabel 4.2 geeft een overzicht van deze parameters en hun functie.

Tabel 4.2 : Initialisatie parameters

MaxNumberOfRetries
: is het maximaal aantal keren dat een frame opnieuw verzonden wordt voor dat een fout conditie optreedt.

MaxAcknowledgeTime
: is de tijd waarbinnen een ontvangstbevestiging van een frame moet zijn ontvangen.

MaxNumberOfBytes
: is het maximaal aantal bytes dat in een informatie frame mag voorkomen.

MaxWindowSize
: is het maximaal aantal frames dat op bevestiging mag staan wachten.

MaxIdleTime
: is de maximale tijd die mag verstrijken zonder dat er een frame is verzonden.

BasicExtended
: geeft aan of het controlfield van I en S frames acht of zestien bits lang is.

Ai
: Gewenste TEI waarde.

SAPI
: SAPI waarde.

ReferenceNrs
: random getal gebruikt voor identificatie bij TEI onderhandelingen.

L1Error zal na ontvangst van Initialise aan de host meedelen of het modem in staat is om bits door te geven.

4.2. De functies

De Datalink layer controller vormt de verbinding tussen een laag 3 entiteit(Host) en een laag 1 entiteit(Modem). De functies die door de controller vervult moeten worden door

de CCITT in X.200 [X200] vastgelegd (Tabel 4.3).

Tabel 4.3 : functies in de datalink laag (CCITT X.200)

Datalink layer

a) datalink connection establishment and release
b) datalink service unit mapping
c) datalink connection splitting
d) delimiting and synchronisation
e) sequence control
f) error detection
g) error recovery
h) flow control
i) identification and parameter exchange
j) control of data cirquit interconnection
k) data link management

Hoe deze functies gerealiseerd worden volgt nu.

ad.a Het opzetten en verbreken van de verbindingen gaat volgens Lapd protocol beschrijving in I.441 [I440] paragrafen 5.5 en 5.6. Hierin wordt het opzetten van een datalink op respectievelijk single frame of multiple frame basis beschreven.

ad.b De afbeelding van de pakketten data van de Host op de laag twee HDLC frames vindt op 1:1 basis plaats. Aan de pakketten wordt de laag 2 header toegevoegd.

ad.c In het opsplitsen van een datalink verbinding in meer physieke verbindingen wordt niet voorzien.

ad.d De data van het modem zal in bit-seriële vorm door het modem aan de controller aangeboden worden. De controller biedt te verzenden data bit-serieel aan het modem aan. Informatie omtrent de geldigheid van een bit wordt ook door het modem geleverd. De laag twee frames worden van elkaar gescheiden door sync karakters ("01111110"). Om een eenduidige herkenning van deze sync tekens mogelijk te maken zal bitstuffing toegepast worden. Als er geen frames uitgezonden worden zal er een continue reeks syncs uitgezonden worden om aan te geven dat de verbinding in active state bevindt.

ad.e De controller zal ervoor zorgen dat de volgorde waarin de pakketten bij de ontvanger aankomen de zelfde is als die waarin ze bij de zender zijn verzonden.

ad.f Fouten detectie wordt uitgevoerd door het berekenen van de CRC over de

ontvangen bitstroom. Verder zal het aantal bytes in een frame en het type van het frame gecontroleerd worden. Tevens kan een te lange wachttijd voor ontvangst van een ontvangstbevestiging van een frame of het wegvallen van het active channel oorzaak van fouten zijn. Tabel 4.4 geeft een overzicht van de mogelijk optredende fouten en hun type.

ad.g Minder ernstige fouten zullen door een hertransmissie mechanisme herstelt worden. Van fouten die niet herstelt kunnen worden door hertransmissie wordt de host op de hoogte gesteld. Bij het optreden van deze fouten zal heropstarten van de link moeten plaatsvinden.

ad.h Flowcontrol vindt plaats op de door CCITT in I.441 [I440] par. 5.5 en par. 5.7 beschreven wijze.

ad.i Tijdens de initialisatie fase zal informatie over de adressering uitgewisseld worden. Deze worden daarna gebruikt.

ad.j De controller biedt de host geen mogelijkheid om verbindingen op te zetten binnen laag 1.

ad.k Bij het opstarten van de verbindingen wordt eerst gekeken of er een modem is. Is dit het geval dan zal de controller de verbinding verder opzetten. Dit houdt in dat de controller geintialiseerd wordt. De parameters die hierbij overgedragen worden staan in tabel 4.2.

Tabel 4.4 : mogelijke fouten

herstelbaar:
-ontvangen framenummer fout
-fout CRC
-ongeldig frame aangekomen

niet herstelbaar:
-aknowledge frame nummer buiten window
-maximaal aantal hertransmissies gedaan
-niet verwacht frame
-ontvangst FRMR

4.3. Gedrag van de controller

Bij het in gebruik nemen van de controller voor datatransmissie kunnen we vier fases herkennen. In de eerste fase zal de controller geïnitialiseerd worden. Na een succesvolle afsluiting van deze fase kan een verbinding worden opgezet met de host aan de andere zijde. Indien het opzetten van een verbinding is gelukt kan de eigenlijke datatransfer plaatsvinden. Nadat de datatransmissie is gebeurt kan de verbinding weer verbroken

worden.

Initialisatie van de controller vind plaats na ontvangst van de initialisatie opdracht (Initialise). De controller zal nu testen of het modem klaar is voor de overdracht van bits (DataSetReady). Indien dit niet het geval is zal de host hiervan op de hoogte worden gesteld door L1Error. Als de bit overdracht wel mogelijk is zal de initialisatie voort gezet worden. Dit resulteert in een continue stroom van sync tekens die FrameOut aan het modem aanbiedt. Indien FrameIn ook bestaat uit een aaneengesloten reeks syncs dan zal dit aan de host gesignaleerd worden met LinkActive. Het met Initialise aangeboden adres kan nu geverifieerd worden. Vervolgens wordt het TransferData proces geinitialiseerd met SetUpParameters. Na terug melding met ParametersSet is de initialisatie afgelopen.

De host kan nu een datalink opzetten via de controller. Dit wordt kenbaar gemaakt door SetUpRequest. Als parameter voor deze request wordt meegedeeld of de data transfer op single frame of op multiple frame basis moet plaats vinden. Bij datatransfer op multiple frame basis wordt gebruik gemaakt van en slidingwindow protocol met een zend window van 8 of 128 frames. Het ontvangst window heeft de grootte een. De pakketten worden verzonden met I frames. (CCITT I.440 [I440]). De acknowledgement van frames is ook in de I frames opgenomen. Verder is er ook nog acknowledgement mogelijk met RR frames. Dit frame type zorgt samen met RNR en REJ frames ook voor de flowcontrol. Datatransfer op single frame basis gebeurt met SI0 en SI1 command frames. Deze frames worden acknowledged met SI0 en SI1 response frames. Deze frames zorgen tevens voor de flow control omdat er geen frames verzonden kunnen worden voordat het vorige bevestigd is.

Bij het opzetten van een datalink op multiple frame basis zal na de SetUpRequest een SABM/SABME frame verzonden worden. Als de peer een verbinding aan wil gaan dan zal een UA frame terug gezonden worden. Dit zal resulteren in een SetUpIndication. Tevens zal de controller in de datatransfer toestand komen.Als de peer niet in staat is de verbinding op te zetten, dan zal hij dit door het sturen van een DM frame laten weten. Dit zal aan de host doorgeven worden met een DisConnectIndication.

Het is ook mogelijk dat de peer host de setup initieert. Dan zal er een SABM/SABME arriveren. Dit zal resulteren in een SetUpIndication. Als de host niet in staat is de verbinding op te stellen, dan zal hij dit signaleren middels HostBusy. Er wordt nu een DM frame verzonden. Is de host wel in staat de verbinding op te zetten dan wordt dit met een SetUpResponse gemeld. Nu wordt een UA frame verzonden. Tevens komt de controller in de datatransfer toestand. In deze datatransfer toestand kunnen pakketten via multiple frame acknowledged data transfer tussen de beide hosts uitgewisseld worden. Dit gebeurt door de overdracht van I frames. Om te voorkomen dat de ontvanger overloopt kunnen er ook nog RNR,RR frames verzonden worden om de datatransfer tijdelijk op te schorten. Fouten in de data transfer kunnen herstelt worden door een her-transmissie aanvraag REJ. Als de datatransfer gedaan is kan de verbinding verbroken worden. Dit gebeurt door een DisConnectRequest of door aankomst van een DISC frame. In het eerste geval zal een DISC frame verzonden worden, en zal na ontvangst van de bevestiging in de vorm van een UA of DM frame een DisConnectIndication aan de host gegeven worden. In het tweede geval zal aan de host een DisConnectIndication gegeven worden. DisConnectRequest zal dan als gevolg hebben dat een UA frame verzonden wordt als de verbinding nog verbroken moet

worden. Als de verbinding reeds verbroken was zal er een DM frame verzonden worden. Bij datatransfer op single frame basis gebeurt zowel opzetten als de datatransfer met SI0 en SI1 frames. Dit wordt nog nader beschreven.

4.4. Eerste decompositie van de datalink controller

Uitgaande van de bovenstaande functie en gedrag beschrijvingen kunnen er twee niveaus onderscheiden worden voor wat betreft de complexiteit van de functies. Dit zijn een laag niveau en een hoog niveau. In het laag niveau zijn de volgende functies te onderscheiden: frame generatie, frame decodering, sync detectie, sync generatie, bitstuffing, destuffing bits, CRC generatie, CRC checking, bij houden van timers, hertransmissie tellen en serieel-parallel omzetting. De andere laag twee functies zijn hoog niveau functies deze kunnen in drie groepen opgedeeld worden[Win]. In de eerste plaats functies die zorgen voor de initialisatie en error meldingen (ManageLayer). Een tweede groep zorgt voor het opzetten en verbreken van verbindingen (ControlConnection). Het verzenden van data pakketten is de derde hoog niveau functie van de controller (TransferData). Deze opdeling is ook terug te vinden in FD0. AssembleDisAssembleFrames zorgt hier voor de realisatie van de laag niveau functies. Tabel 4.5 geeft een overzicht hoe de functies uit tabel 4.3 op de processen in FD0 (figuur 4.4) zijn afgebeeld.

Tabel 4.5 : Afbeelding functies op processen

functie	processen
a	1,2,3
b	1
c	-
d	3
e	1
f	1,2,3
g	2,3,4
h	1
i	3,4
j	-
k	3,4

In FD0 (figuur 4.4) verschijnen een aantal nieuwe flows. Zij worden kort beschreven. Tevens zal hun interactie aangegeven worden.

4.4.1.Flows tussen TranferData en ControlConnection

De besturing van de multiple frame data transfer gebeurt met de volgende twee flows:

StartDataTransfer
: Als een data link op multiple frame basis is opgezet wordt met deze flow het verzenden van pakketten mogelijk gemaakt. Tevens wordt de toestand van het verzend proces in de reset toestand gebracht.

StopDataTransfer
: Met deze flow kan multiple frame data transfer gestopt worden.

Drie flows die een rol spelen bij het opzetten van single frame data transfer zijn:

ConnectState
: ConnectState geeft de momentele toestand van de controller weer. De volgende toestanden zijn mogelijk: Idle, TeiAssigned, Connecting, Connected, DisConnecting, Restarting, SingleFrame. Deze flow is van belang bij het opzetten van een datalink op single frame basis.

SICommand
: SICommand bestuurt het opzetten van een single frame datalink. De waarden die SICommand kan aannemen zijn: SISetUp en SIDisc. Dit zijn de opdrachten om respectievelijk een link op te zetten en te verbreken.

SIResponse
: SIResponse zorgt voor de bevestiging van SICommand met de waarden SISetUpDone en SIDiscDone. Tevens zal deze flow doorgeven als de peer controller een link wil opzetten of verbreken. Dit gebeurt met de opdrachten RSISetUp en RSIDisc.

De overige drie flows spelen een rol bij de fout afhandeling.

NRError
: NRError geeft aan dat het ontvangen bevestigings framenummer niet in het zend window valt. Deze fout kan niet door her-transmissie herstelt worden, daarom zal dit in een fout melding resulteren.

RequestStateVar
: Bij het afhandelen van de fout moet de toestand van TransferData bekent zijn. Met RequestStateVar wordt de toestand van TransferData aangevraagd.

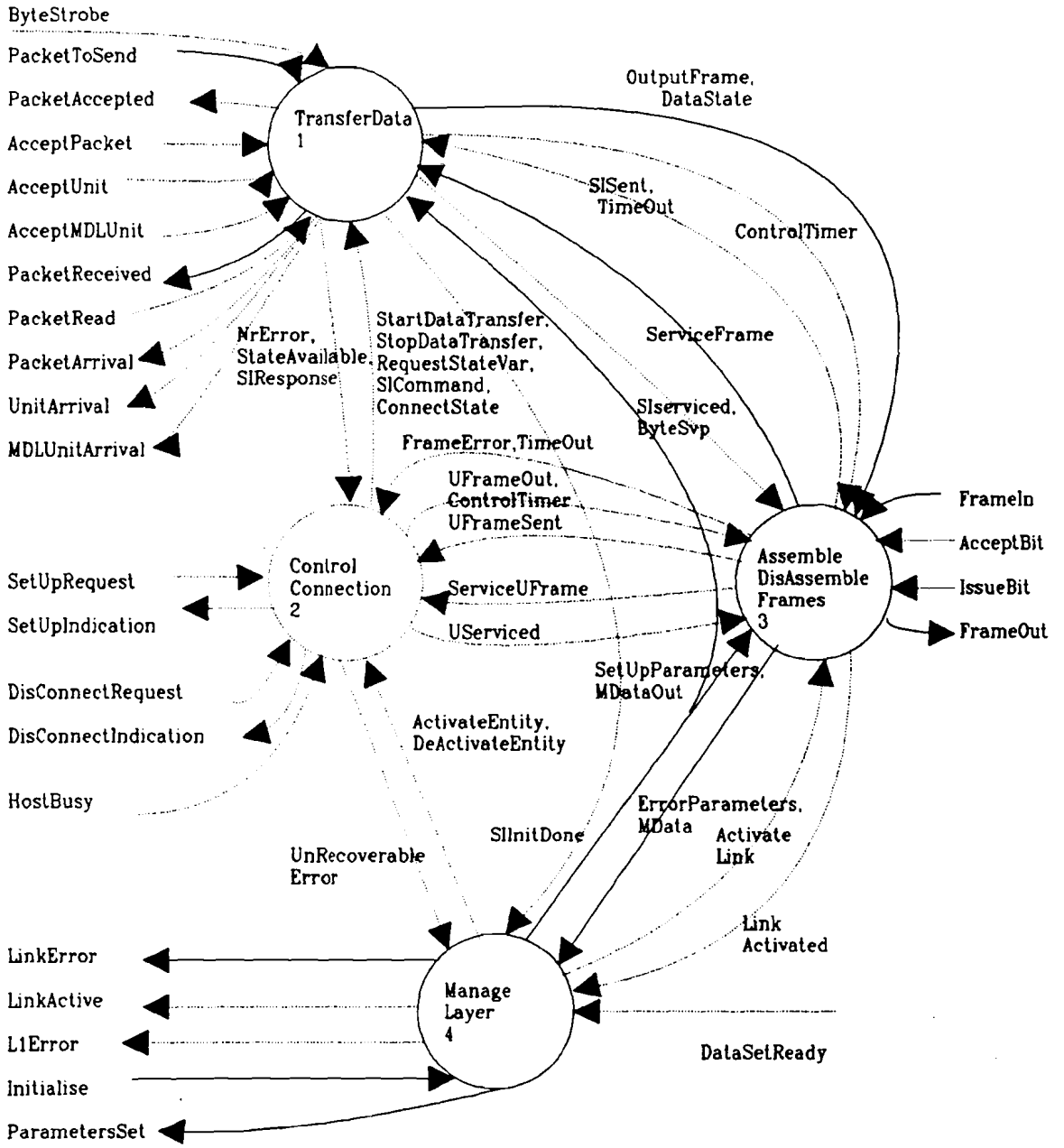
StateAvailable
: StateAvailable geeft aan dat de met RequestStateVar aangevraagde toestand (DataState) klaar staat.

4.4.2. Flows tussen ControlConnection-AssembleDisAssembleFrames

De volgende drie flows worden gebruikt bij het verzenden van unnumbered frames bij het opzetten, verbreken en herstarten van verbindingen.

UFrameOut
: UframeOut is de flow die aangeeft welk frame verzonden moet worden. Dit kan zijn: SABMOut, DMOut, DISCOut, UAOut en FRMROut. Bij deze frames is reeds impliciet vastgelegd of het command of response frames zijn. Tevens is de waarde van het Poll of Final (P/F) bit ook vast gelegd. Deze twee items hoeven dus niet expliciet in de flow opgenomen te worden.

UFrameSent
: UFrameOut bevestigt dat een frame waarvan de verzending met UFramOut aangevraagd was verzonden is.



figuur 4.4 : FD0

ControlTimer

ControlTimer wordt gebruikt om de timer en de her-transmissie teller te starten als een command frame verzonden is. Dit gebeurt nadat met SISent gemeld is dat een frame voor de eerste keer verzonden is. Tevens wordt de timer gestopt met deze flow.

Bij de uitwisseling van informatie over de ontvangen frames spelen de volgende twee flows een rol.

ServiceUFrame

ServiceUframe bevat het frametype dat ontvangen is door AssembleDisAssembleFrames. Dit kunnen de volgende frames zijn: SABM, DISC, DM, UA en FRMR. Tevens wordt aangegeven of het ontvangen frame een command of response frame was. Dit gebeurt ook als het geen van de frames is die door ControlConnection verwerkt kan worden.

UServiced

UServiced geeft aan dat het aangekomen frame verwerkt is.

De twee overblijvende flows FrameError en TimeOut zijn fout meldingen.

FrameError

FrameError geeft aan dat het frame niet voldoet aan de eisen gesteld aan dit type frame.

TimeOut

TimeOut geeft aan dat het maximaal aantal her-transmissie gedaan is.

4.4.3.Flows tussen ControlConnection en ManageLayer

Hier zijn de volgende flows te herkennen:

ActivateEntity

Met deze flow wordt de controller van de Idle toestand die hij heeft tijdens de initialisatie naar de toestand TeiAssigned gebracht.

DeActivateEntity

DeActivateEntity brengt de controller weer in de Idle toestand.

UnRecoverableError

UnRecoverableError meldt aan ManageLayer dat een fout is opgetreden. Dit kan een lokale fout zijn. Het is echter ook mogelijk dat een FRMR frame geariveerd is.

4.4.4.Flows tussen TransferData en AssembleDisAssembleFrames

Flows die betrekking hebben op de verwerking van ontvangen frames zijn:

ServiceFrame

ServiceFrame bevat het control field van het ontvangen frame. Tevens wordt indien nodig aangegeven of het een command of een respons frame betreft. De waarde van het P/F bit wordt ook mede gedeeld. Tevens wordt aangegeven of het om management data gaat. Tenslotte is er ook nog pakket data in deze flow aanwezig.

SIServiced

SIServiced geeft aan dat de met ServiceFrame geleverde informatie verwerkt is.

ByteSvp

ByteSvp geeft aan dat het volgende byte verwacht wordt.

Voor het verzenden van informatie zijn de volgende drie flows van belang:

OutputFrame

OutputFrame bevat het control field van het te verzenden frame. Tevens wordt indien nodig aangegeven dat het een command of een response frame is. Management data wordt apart aangeduid. Het dataveld kan ook nog packet

informatie bevatten.

SISent

SISent geeft aan dat de met OutputFrame aangeboden informatie verzonden is.

Controltimer

ControlTimer wordt bij het verzenden van command frames gebruikt om de timer en eventueel de her-transmissie teller te starten. Dit gebeurt nadat SISent ontvangen is.

De twee over blijvende flows spelen een rol bij de fout afhandeling.

DataState

DataState geeft de toestand van transferData weer op het moment dat hier naar met RequestState gevraagd werd.

TimeOut

TimeOut geeft aan dat de timer is afgelopen.

4.4.5.Flows tussen TransferData en ManageLayer

SetUpParameters

Tijdens de initialisatie wordt met deze flow TransferData op de hoogte gesteld van de maximale window grootte en het frame type (BasicExtended). Tevens wordt de initialisatie van het single frame data transfer proces hier door getriggerd.

SIInitDone

SIInitDone geeft aan welk deel van de initialisatie van het single frame data transfer proces gedaan is.

4.4.6.Flows tussen ManageLayer en AssembleDisAssembleFrames

Met uitzondering van de flow ErrorParameters zijn deze flows allemaal van belang bij de initialisatie van de controller. Ze worden nu genoemd in de volgorde waarin ze optreden.

ActivateLink

Met ActivateLink krijgt AssembleDisAssembleFrames de opdracht om een continue reeks van sync karakters te verzenden.

LinkActivated

Met LinkActivated meldt AssembleDisAssembleFrames dat ook een continue reeks van sync karakters ontvangen wordt. De link is dus te gebruiken voor de overdracht van bits.

SetUpParameters

SetUpParameters geeft een aantal parameters door die van belang zijn bij het gebruik van de controller. Dit zijn: MaxNumberOfBytes, MaxNumberOfRetries, MaxAcnowledgeTime en MaxIdleTime.

MDataOut

MData bevat het adres van de controller, dit moet nog geverifieerd worden met de peer controller.

MData

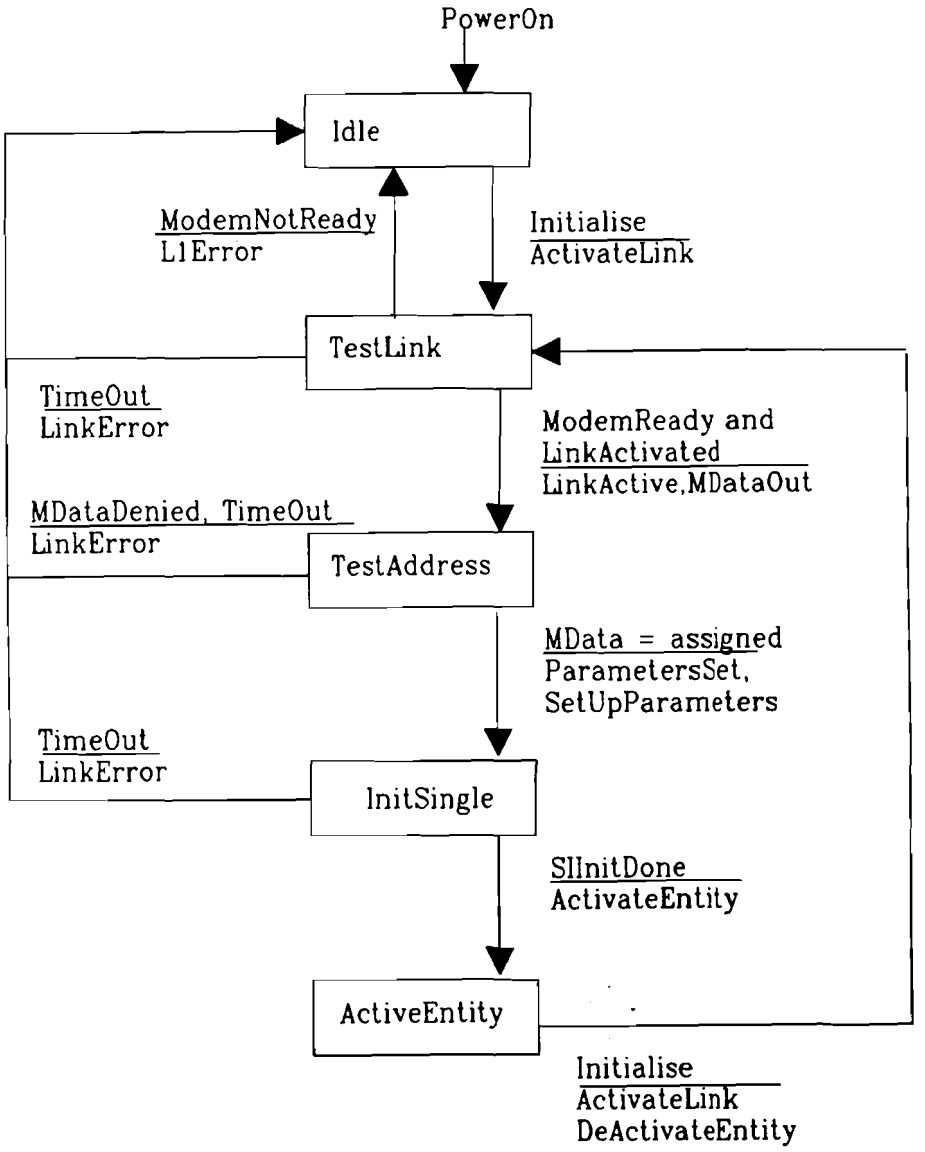
MData bevat het geverifieerde adres van de controller.

ErrorParameters

ErrorParameters geeft de toestand van de controller op het ogenblik dat een fout optrad.

4.5. ManageLayer

ManageLayer zal zorgen voor het initialiseren van de controller in de initialisatie fase en voor het doorgeven van error informatie aan de host.
Bij de initialisatie zal achtereenvolgens het volgende moeten gebeuren (figuur 4.5):



Figuur 4.5 : Gedrag Managelayer tijdens de initialisatie

-testen of de laag 1 entiteit klaar is.
-setten van timers binnen de controller.
-testen of laag 1 bits kan transporteren.
-adres verificatie met het netwerk.
-initialisatie van het datatransfer proces.
-controller voor gebruik vrijgeven.

Initialisatie begint na ontvangst van Initialise. Indien DataSetReady = false, dan zal de host hiervan op de hoogte gesteld worden door een L1Error. Gelijktijdig worden de waarden van MaxAcknowledgeTime, MaxNumberOfRetries en MaxNumberOfBytes in SetUpParameters aan AssembleDisAssembleFrames doorgegeven. Indien DataSetReady = true dan zal ActivateLink geactiveerd worden. Als AssembleDisAssembleFrames een active link detecteert dan wordt dit gesignaleerd met LinkActivated. De host wordt hiervan op de hoogte gesteld met LinkActive. De adres verificatie met de andere controller vindt nu plaats. MData en MDataOut zijn de flows die de uit te wisselen informatie voor adres verificatie bevatten. Deze informatie was met de flow Initialise aan de controller aangeboden. De bevestiging aan de host gebeurt met de flow ParametersSet. Nu volgt de initialisatie van TransferData met de flow SetUpParameters = MaxWindowSize. Na terug melding met SIInitDone wordt de controller in de TeiAssigned toestand gebracht door ActivateEntity.

De hier beschreven gang van zaken is gegeven voor het geval de initialisatie succesvol verloopt. De volgende situaties kunnen echter ook optreden. De lijn wordt niet active binnen MaxAcknowledgeTime. De adres verificatie is niet succesvol of is niet gebeurt binnen vier maal MaxAcknowledgeTime. De initialisatie van het TransferData proces is niet succesvol. Al deze gebeurtenissen worden met LinkError gemeld aan de host. Het ManageLayer proces komt nu weer in de Reset toestand die het had voor de initialisatie begon.

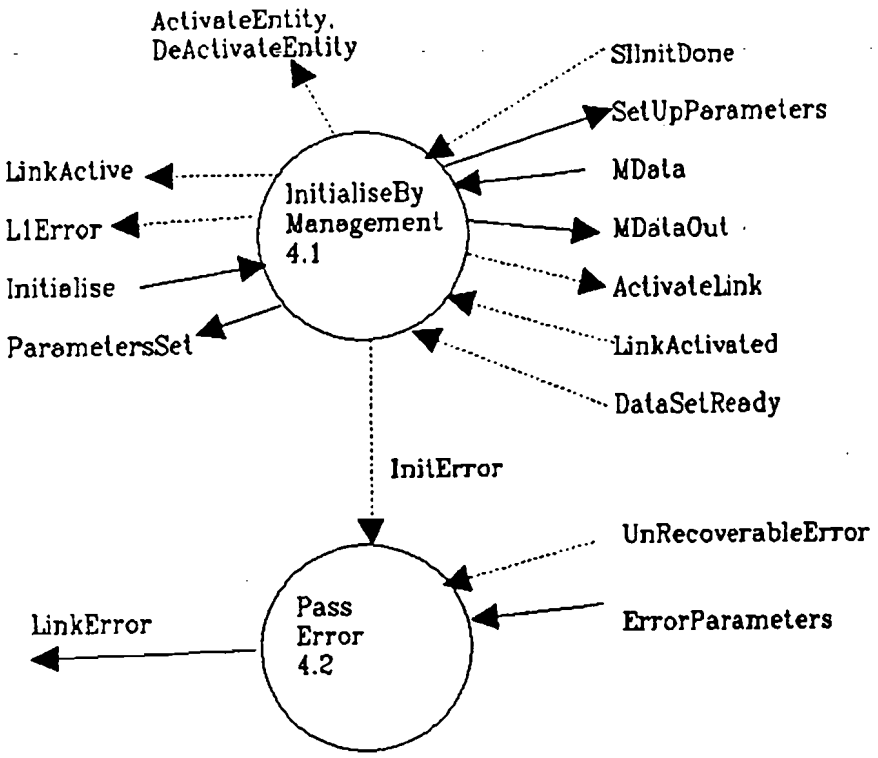
Het optreden van fouten tijdens de operationele fase wordt als UnRecoverableError gesignaleerd. Informatie over de aard van de fout wordt aangeboden als ErrorParameters. De fout wordt aan de host gemeld als LinkError. Met DeActivateEntitiy kan gesignaleerd worden dat de controller in de Idle toestand gebracht moet worden.

Figuur 4.6 toont hoe ManageLayer in subprocessen is opgedeeld. De verdere decompositie en de procesbeschrijvingen worden in appendix A gegeven.

4.6. ControlConnection

Een belangrijke functie die ControlConnection vervult is het bij houden van de toestand van de controller. Tevens zal ControlConnection er voor zorgen dat de toestand van de controller bij het opzetten en verbreken van de verbinding op de gewenste wijze veranderd wordt. De besturing van de fouten afhandeling hoort ook bij de taak van ControlConnection.

De toestanden waarin de controller zich kan bevinden zijn: Idle, TeiAssigned, Connecting, Connected, DisConnecting, Restarting en SingleFrame. Naar buiten toe wordt deze toestand door de flow ConnectState aangegeven (figuur 4.7). Tijdens de

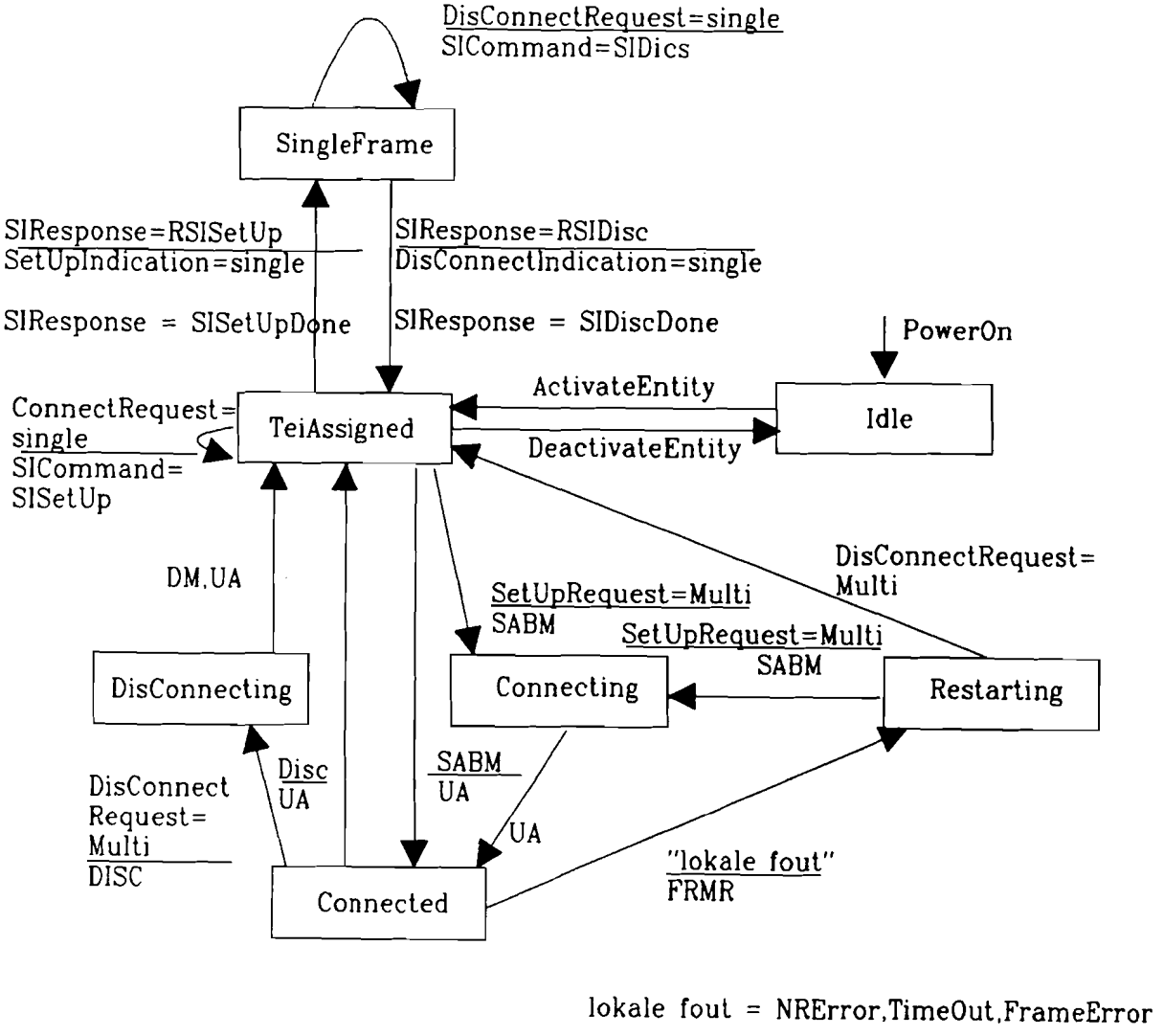


figuur 4.6 : ManageLayer

initialisatie fase is de controller in de Idle toestand. Als de initialisatie is afgelopen wordt de toestand met ActivateEntity veranderd in TeiAssigned. In deze toestand kunnen aanvragen voor het opzetten van een link gehonoreerd worden. Tevens is nu unacknowledged datatransfer mogelijk. Zowel aanvragen voor het opzetten van een link op single frame basis als aanvragen voor het opzetten van een link op multiple frame basis kunnen nu plaatsvinden. Als een aanvraag succesvol afgewerkt is komt de controller in de toestand Connected of SingleFrame voor respectievelijk multiple frame overdracht en single frame overdracht. Aanvragen voor het verbreken van de verbinding brengen de controller weer terug in TeiAssigned toestand. Hoe de controller reageert op het optreden van fouten die niet te herstellen zijn door hertransmissie wordt ook door ControlConnection bepaald. De controller is dan in de toestand Restarting.

Nadat ActivateEntitiy is ontvangen kan ControlConnection gebruikt worden voor het opzetten van een Datalink. Het initiatief tot het opzetten van een verbinding kan bij de lokale host of bij de andere host liggen.

Nu zal een korte beschrijving volgen van de functies van de flows die zijn verbonden met ControlConnection. Hierbij zal eerst de beschrijving volgen voor het opzetten van single frame datatransfer. Daarna komen achtereenvolgens aan de orde: het opzetten van een multiple frame link en fout afhandeling. Bij het opzetten en verbreken van single frame datatransfer wordt gebruik gemaakt van de zelfde frames (SI0, SI1) die ook bij de datatransfer gebruikt worden. Daarom is een deel van de processen die voor het opzetten en verbreken van de link zorgen in TransferData gesitueerd. Op deze wijze



figuur 4.7 : ConnectState toestand diagram

wordt voorkomen dat veel informatie omtrent de toestand van de processen uitgewisseld moet worden. De processen in TransferData worden vanuit ControlConnection bestuurd met de flow SICommand. De flow SIResponse zorgt voor de bevestiging van deze processen. Het opzetten van een verbinding door de lokale host gebeurt door SetUpRequest = Single. Als ConnectState = TeiAssigned heeft dit tot gevolg dat SICommand = SISetUp wordt. Als het frame verzonden is wordt SIReponse = SISetUpDone. ConnectState wordt nu SingleFrame. Verbreken van een single frame verbinding door de lokale host gebeurt na DisConnectRequest = Single. SICommand wordt nu SIDisc. Nadat SIResponse = SIDiscDone is geworden is wordt ConnectState = TeiAssigned. Als de remote host een single frame verbinding wil opzetten wordt dit met SIResponse = RSISetUp gemeld. SetUpIndication = Single geeft dit door aan de host. ConnectState wordt SingleFrame. Bij verbreken van de verbinding zal SIResponse = RSIDisc zijn. Dit resulteert in DisConnectIndication = Single. Tevens zal ConnectState TeiAssigned worden.

De lokale host kan het opzetten van een multiple frame verbinding beginnen door een SetUpRequest = multi. ServiceUFrame is SABM betekent dat de andere host een verbinding op wil zetten. Bij het uitvoeren van opdrachten van ServiceUFrame zal de acceptatie van de opdracht door UServiced terug gemeld worden aan AssembleDisAssembleFrame.

Verbreken van verbindingen gebeurt door: DisConnectRequest van de lokale host, ServiceUFrame is DISC of DM. Indien de host een SetUpRequest geeft zal dit resulteren in een SabmOut opdracht voor AssembleDisAssembleFrames. Als het frame is verzonden zal dit door USent terug gemeld worden. In de normale situatie zal de SABM bevestigd worden door een UA frame. Als dit het gebeurt is kan het datatransfer proces geactiveerd worden(StartTransfer). De host wordt van het beschikbaar zijn van een data verbinding op de hoogte gesteld met SetUpIndication.

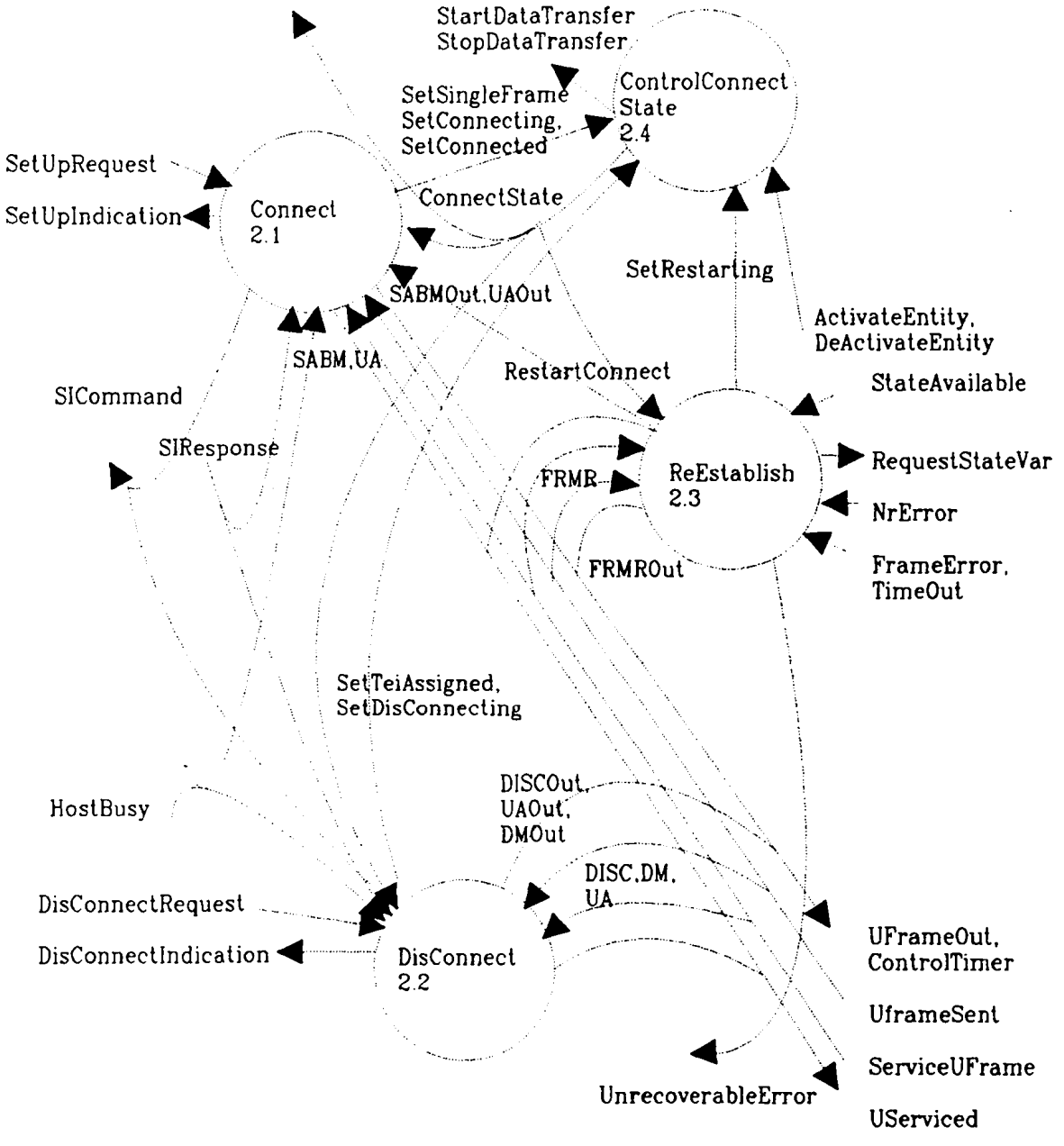
De fouten afhandeling zal bepaald worden door de plaats waar de fout opgemerkt wordt. Als de peer controller een fout conditie detecteerd dan zal dit met een FRMR frame gemeld worden. De lokale controller moet nu de host op de hoogte stellen (UnRecoverableError) en de verbinding nieuw opzetten. Lokale fouten die kunnen optreden zijn: NRError, FrameError, TimeOut. Na het lokaal detecteren van een fout moet de peer controller en de lokale host van deze fout op de hoogte gesteld worden. Om dit te doen moet eerst informatie omtrent de toestand van de controller vastgesteld worden. Dit gebeurt door het vragen van de toestand van TransferData (RequestStateVar). Als deze informatie beschikbaar is (DataState) wordt dit met StateAvailable gemeld. De informatie kan nu in een FRMR frame naar de peer controller gestuurd worden. Tevens wordt de lokale host van de fout oorzaak op de hoogte gesteld (UnRecoverableError = true). In afwachting van de te nemen actie komt de controller nu in ConnectState = Restarting toestand.

Om het gedrag dat hier boven is beschreven te realiseren wordt het ControlConnection op gedeeld in vier processen (figuur 4.8). De verdere decompositie en de proces beschrijvingen zijn te vinden in appendix A.

4.7. AssembleDisAssembleFrame

De twee belangrijkste functies die AssembleDisAssembleFrames te vervullen heeft zijn: het ontvangen van frames en het verzenden van frames. Daarnaast moet AssembleDisAssembleFrames ook nog fout informatie doorgeven en de hertransmissie teller bij houden.

Bij de initialisatie worden eerst de parameters MaxAcknowledgeTimer, MaxNumberOfBytes, MaxNumberOfRetries en BasicExtended met SetUpParameters aan het proces doorgegeven. Daarna zal het proces na ActivateLink een continue sync stroom op FrameOut genereren. Als een zelfde stroom op FrameIn wordt gedetecteerd zal LinkActivated gesignaleerd worden aan ManageLayer. Hierna vindt adres verificatie plaats. De te verifieren parameters worden met MDataOut aangeboden. MData vormt de terugmelding. De Tei waarde wordt daarna via SetUpParameters doorgegeven. MData kan ook een foutmelding zijn indien de verificatie niet tijdig gebeurt is. AcceptBit aan op welk moment een bit in FrameIn geldig is. Wanneer een bit op FrameOut uitgegeven moet worden wordt bepaald door IssueBit.



figuur 4.8 : ControlConnection

Voor het verzenden van frames bij het opzetten, verbreken en gebruiken van een verbinding worden er aan dit proces een aantal parameters aangeboden. Tijdens het opzetten en verbreken van een multiple frame verbinding zal het proces alleen communiceren met ControlConnection. ControlConnection zal met UFrameOut aangeven welk frame gezonden moet worden. UFrameSent geeft aan dat het frame verzonden is. Tevens kan ControlConnection een timer starten (ControlTimer = StartTimer) die indien er na MaxAcknowledgeTime geen ontvangstbevestiging is, het zelfde frame nog eens zendt. Dit kan maximaal MaxNumberOfRetries keren gebeuren.

Is er dan nog geen bevestiging, dan zal TimeOut aan ControlConnection gemeld worden. Wordt een frame wel bevestigd, dan kan de timer met ControlTimer = StopTimer gestopt worden.

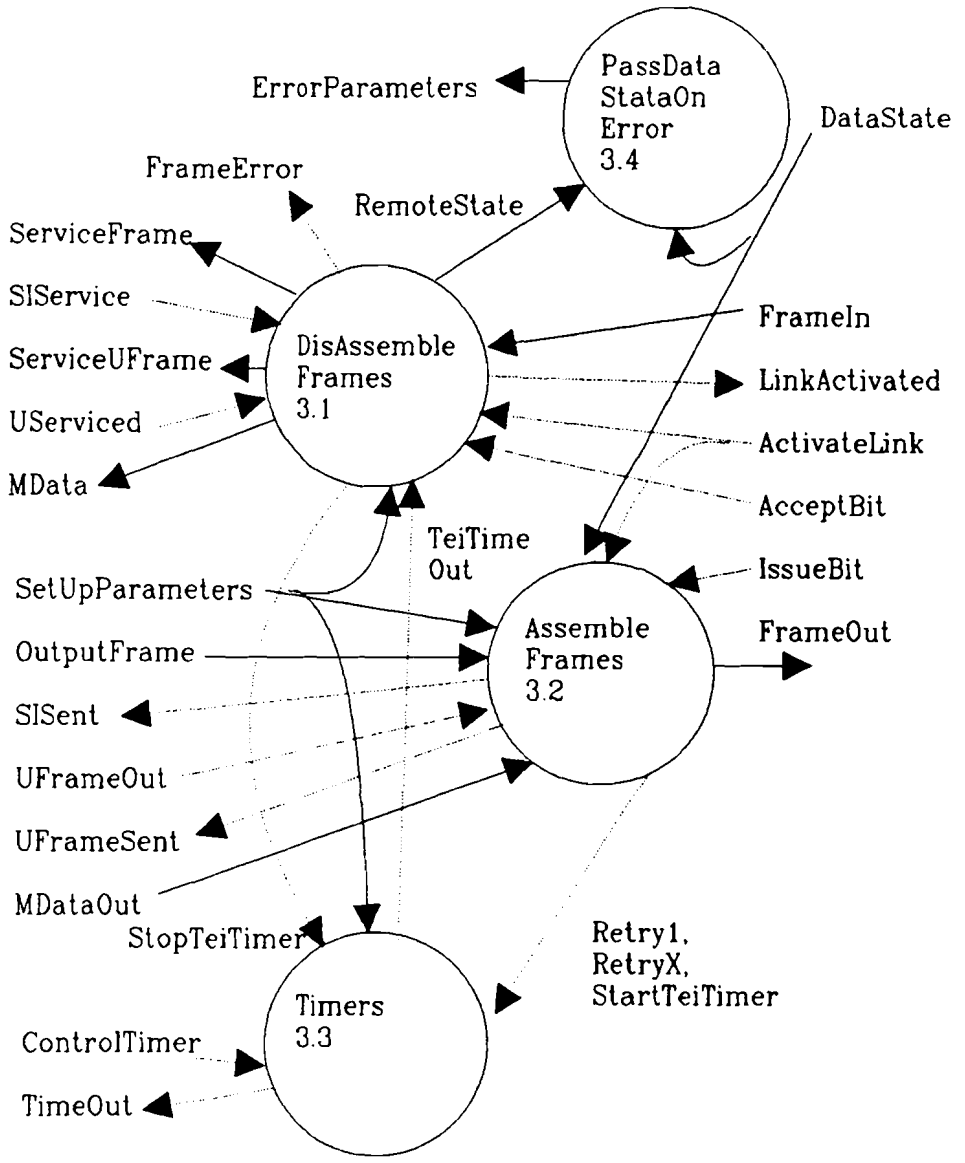
Indien een aankomend frame (FrameIn) geen fouten bevat zal het aan het verwerkende proces doorgegeven worden(ServiceUFrame, ServiceFrame, MData). Als deze informatie verwerkt is wordt dit door UServiced of SIServiced gemeld.
Als een frame wel fouten bevat dan wordt dit aan ControlConnection mee gedeeld (FrameError). ControlConnection vraagt aan TransferData om zijn toestand bekent te maken aan AssembleDisAssembleFrames. Als dit gebeurt is wordt dit gesignaleerd door StateAvailable. Nu wordt UFrameOut = FRMROut. Dit frame wordt nu samengesteld uit DataState en de informatie in AssembleDisAssembleFrames. Na USent wordt TimerControl = StartTimer. De Error informatie wordt tevens door gegeven aan ManageLayer(ErrorParameters).
Dit heeft geleid tot de in FD 3 (figuur 4.9) gegeven verdeling in subprocessen.

4.8. TransferData

TransferData is het proces dat er voor zorgt dat packet data die door de host aangeboden wordt bij de peer host wordt bezorgd en visa versa. De controller doet dit door de pakketten data in HDLC frames te zetten en als bits te verzenden. De data kan op de volgende twee manieren verzonden worden: acknowledged, unacknowledged. Bij acknowledged datatransfer zorgt de controller samen met de peer controller er voor dat de packet data correct en in de juiste volgorde bij de peer host wordt afgeleverd. Bij unacknowledged overdracht wordt er geen garantie gegeven dat een packet correct bij de peer aankomt.

Zoals reeds vermeld zullen twee soorten pakketten met unacknowledged datatransfer verzonden worden. Dit zijn pakketten die DL-UNIT DATA vormen en bestemd zijn voor de laag drie entiteit en pakketten die de MDL-UNIT DATA vormen en bestemd zijn voor het management binnen de host. Welke flows deze primitieven realiseren is reeds beschreven. De pakket data voor unacknowledged datatransfer zal in de vorm van UI frames overgedragen moeten worden. Het onderscheid tussen laag drie data en management data wordt bij de frames gemaakt door een ander adres te geven aan de verschillende frames. Voor DL-UNIT DATA zal het bij de initialisatie vastgestelde adres gebruikt worden. Bij MDL-UNIT DATA zal het adres voor management functies gebruikt worden.

Voordat acknowledged datatransfer mogelijk is zal eerst een verbinding opgezet moeten worden. Als dit gebeurt is kan de data transfer plaatsvinden. Of dit gebeurt in single frame of in multiple frame mode is tijdens het opzetten van de verbinding bepaald. Uitwisseling van pakketten met de host gebeurt op de reeds beschreven wijze. Deze pakketten worden in het geval van single frame overdracht met SI0 en SI1 frames overgedragen. In het geval van multiple frame overdracht worden I frames gebruikt. Bij multiple frame overdracht kunnen meerdere verzonden frames op bevestiging staan wachten, bij single frame overdacht is dit telkens maar een frame.



figuur 4.9 : AssembleDisAssembleFrames

De flows noodzakelijk voor de werking van TransferData zullen nu kort beschreven worden. SetParameters en SIInitDone zijn twee flows die tijdens de initialisatie fase gebruikt worden. SetUpParameters bevat de window grootte die bij multiple frame overdracht gebruikt wordt. Tevens zorgt SetParameters er voor dat de initialisatie van het single frame data transfer proces gestart wordt. Deze initialisatie bestaat uit twee fasen. SIInitDone zorgt voor de terug melding van de realisatie van elk van de twee fases. ServiceFrame en FrameServiced worden gebruikt bij de overdracht van ontvangen en reeds gedecodeerde frames. FrameServiced geeft aan dat de met ServiceFrame gevraagde actie uitgevoerd is. ServiceFrame bevat informatie omtrent de acties die TransferData moet uit voeren. ServiceFrame bestaat uit:
-Controlfield : dit is het controlfield uit het ontvangen frame en bepaalt welke

actie er ondernomen moet worden.

-C/R : geeft aan of het ontvangen frame een command (C) of een response (R) frame is.

-Data : dit item is slechts aanwezig als het ontvangen frame pakket data bevatte.

-M : wordt slechts gebruikt als we met een UI frame van het management te maken hebben.

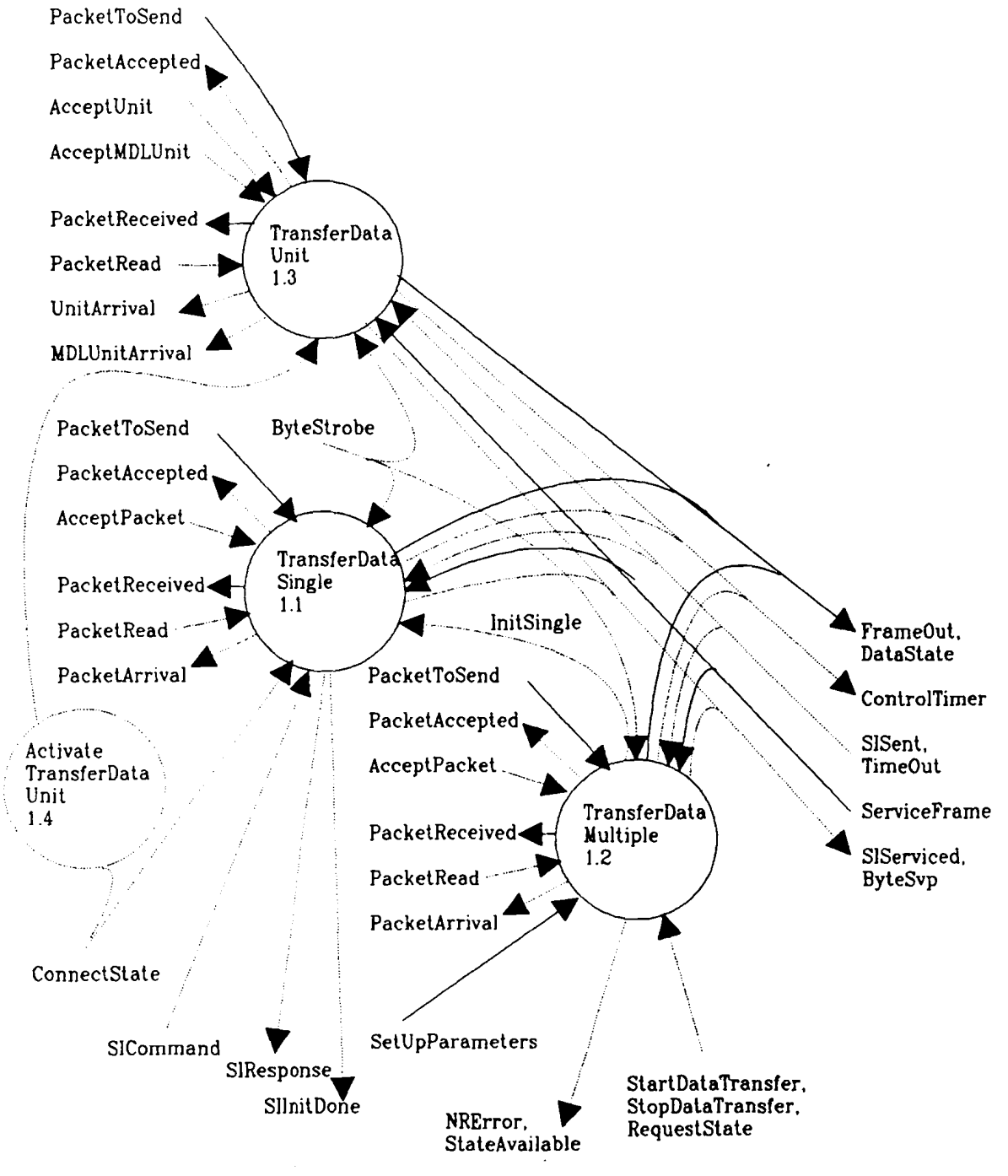
OutputFrame bevat de zelfde componenten, ze zijn dan echter alle uitgebreid met het achtervoegsel Out. Met deze flow wordt aan AssembleDisAssembleFrames de opdracht gegeven om een frame met deze componenten te verzenden. Als dit gebeurt is zal dit met FrameSent bevestigd worden. Als de bevestiging van een verzonden frame te lang op zich laat wachten, moet en frame soms opnieuw verzonden worden. Om de wachttijd bij te houden zit een timer in AssembleDisAssembleFrames. Deze timer kan gestart en gestopt worden met Controltimer, deze flow wordt tevens gebruikt voor het starten en stoppen van de hertransmissie teller. Als de tijd te lang is wordt dit met TimeOut gemeld. Als een timeout optreedt is er sprake van een error conditie. Behalve timeout kunnen er meer error condities optreden, onder andere NRError om aan te duiden dat het ontvangen acknowledge nummer niet binnen het window ligt. Deze errors moet aan het management gemeld worden, hiervoor is het onder andere noodzakelijk om de toestand van het datatransfer proces te kennen. Deze toestand wordt aangevraagd met RequestStateVar. De Toestand wordt dan klaar gezet als DataState. StateAvailable signaleert daarna dat dit gebeurt is.

Enkele van de bovengenoemde data transfer processen zullen pas geactiveerd kunnen worden nadat de link is opgezet. Dit opzetten van een link wordt gedaan door ControlConnection. Als dit proces een link voor multiple frame overdracht heeft opgezet dan zal dit met StartDataTransfer gemeld worden. StopDataTransfer stopt het datatransfer proces in multiple frame mode. Voor data overdracht in single frame mode is een andere opzet gebruikt. Het opzetten van een verbinding gebeurt hierbij namelijk met de zelfde soort frames als de dat transfer (SI0, SI1). Daarom is een deel van het proces voor het opzetten van de verbinding is TransferData geplaatst. Het single frame data transfer proces wordt bestuurd met de flow SICommand. Bevestiging gebeurt met SIresponse. Tevens speelt ConnectState nog een belangrijke rol. Deze flow geeft de momentane toestand van de controller aan. Deze toestand bepaalt tevens of unacknowledged data transfer mogelijk is. Deze transfer is namelijk mogelijk zo gauw ConnectState = not Idle. De overige flows van TransferData zijn voor de realisatie van data primitieven en werden reeds eerder beschreven.

Om deze functies te realiseren is TransferData opgedeeld zoals te zien is in FD 1 (figuur 4.10). Hierin worden sommige flows meerdere keren genoemd omdat het tekenen van deze flows zou leiden tot een onoverzichtelijk netwerk van flows.

4.8.1.TransferDataMultiple

In TransferDataMultiple moeten die functies uit gevoerd worden die nodig zijn voor acknowledged multiple frame data transfer. Deze data transfer kan pas plaatsvinden nadat StartDataTransfer mee gedeeld heeft dat de controller in de transfer data toestand is gekomen. Tijdens de initialisatie fase is TransferData reeds op de hoogte gebracht van



figuur 4.10 : TransferData

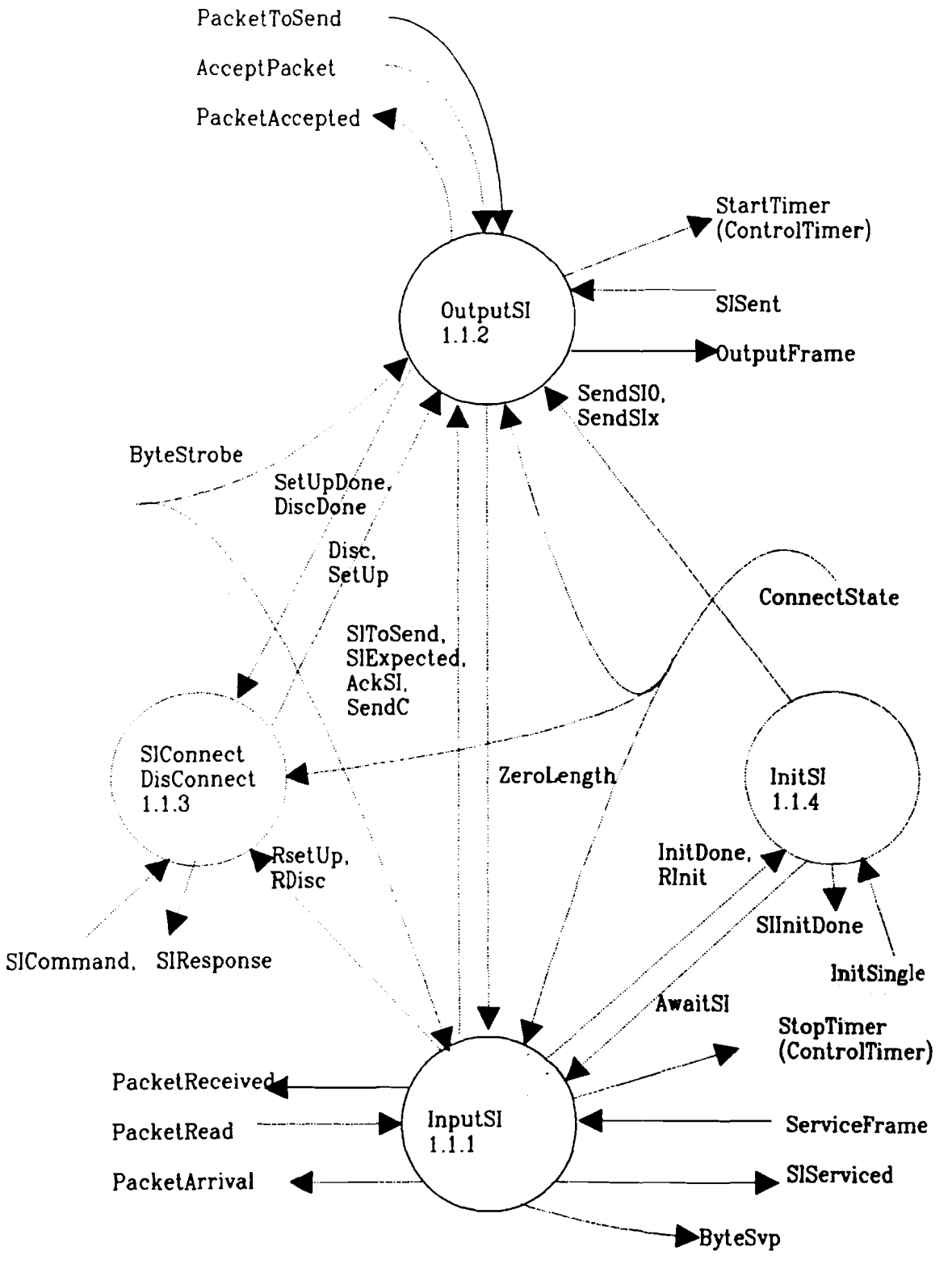
de maximale window grootte bij multiple frame operatie(MaxWindowSize). Van de frames die ontvangen zijn wordt de informatie (Data, Controlfield) aangeboden. Als blijkt dat het ontvangen frame nummer overeen komt met het verwachte framenummer, dan zal het frame aan de host door gegeven worden. De host wordt van de aankomst van een frame op de hoogte gebracht door PacketArrival. Als de host het packet geaccepteerd heeft wordt dit door PacketRead gesignaleerd. Het packet kan nu uit het ontvanger buffer verwijderd worden. Tevens wordt AssembleDisAssembleFrames er van op de hoogte gesteld dat het packet verwerkt is (SIServiced). Behalve het framenummer vinden we in Controlfield ook een acknowledge framenummer. Als dit nummer binnen het zend window valt zal het window aan gepast worden. De timer wordt gestopt. Valt het acknowledge nummer buiten het window, dan zal er een NrError gegeven worden. ConnectControl zal dan om de status van DataTransfer vragen (RequestStateVar). Deze status wordt aan AssembleDisAssembleFrame aangeboden. Als dit gebeurt is wordt ConnectControl hiervan op de hoogte gebracht (StateAvailable). Als de host een packet te verzenden heeft dan signaleert hij dit met AcceptPacket. Als de controller het packet geaccepteerd heeft wordt dit terug gemeld met PacketAccepted. Het packet zal vergezelt van een ControlfieldOut waarin het framenummer en het acknowledge framenummer zitten aangeboden aan AssembleDisAssembleFrames. Tevens zal nadat SISent ontvangen is, indien de timer nog niet gestopt is de timer gestart worden. In geval van TimeOut zullen de nog niet bevestigde frames opnieuw verzonden worden. In multiple frame mode zal TransferData er tevens voor zorgen dat er geen overloop van het ontvanger buffer kan optreden. Hiervoor wordt gebruik gemaakt van RR,RNR commands. Met RNR kan de controller aan de andere kant er van op de hoogte gesteld worden dat er geen frames meer gezonden mogen worden. Als de lokale ontvanger wel weer in staat is om frames te ontvangen, dan zal een RR frame verzonden worden.

4.8.2.TransferDataSingle

Bij single frame data transfer is het aantal op bevestiging wachtende frames maximaal 1. Voordat single frame datatransfer plaats kan vinden moet eerst initialisatie van de controller plaatsvinden. Tijdens deze initialisatie worden de toestand variabelen van de controllers gelijk gezet. Na de initialisatie moet de datalink nu eerst nog opgezet worden alvorens data transfer kan plaatsvinden. Als de datatransfer gedaan is kan de datalink weer verbroken worden. De processen die voor de realisatie hiervan zorgen worden weergegeven in FD 1.1 (figuur 4.11).

4.8.2.1.InitSI

Als door SetParameters aangegeven wordt dat het Singleframe transfer proces geïnitialiseerd moet worden dan zal dit tot gevolg hebben dat OutPutSI de opdracht krijgt om een SI0 te verzenden (SendSI0). InputSI krijgt de opdrachten om op het response frame (AwaitSI)te wachten. Bij aankomst van dit response frame wordt SIToSend gelijk aan het sequence nummer van het frame. De aankomst wordt tevens gemeld aan InitSI (InitDone). AwaitSI wordt nu false. Aankomst van een SI0 command frame tijdens de initialisatie fase wordt met RInit gemeld aan InitSI. Dit heeft tot gevolg dat een SI response frame (SendSIx) met het volgnummer SIAckExp naar de peer controller wordt gezonden. Als deze uitwisseling van frames plaats gevonden heeft wordt het einde van de initialisatie met SIInitDone gemeld aan ManageLayer en aan



figuur 4.11 : TransferDataSingle

ControlConnection. Deze processen zullen met respectievelijk ParametersSet en SetUpIndication de host hier van op de hoogte stellen.

4.8.2.2.SISetup

Een single frame datalink kan worden opgezet als de controller zich in TeiAssigned toestand bevindt. Een verbinding kan worden opgezet na een SetUpRequest = Single of na ontvangst van een Commandframe zonder informatie deel en met P = 1.
In TeiAssigned state zal SetUpRequest = Single een SISetUpReq tot gevolg hebben. Deze signaleert aan Output SI dat een gepast command zonder data deel verzonden moet worden. Tevens wordt de timer gestart. Ontvangst van SetUpDone signaleert dat de peer controller de verbinding kan opzetten. Dit wordt door SetUpDone terug gemeld aan ControlConnection. Dit heeft tot gevolg dat ConnectState = SingleFrame wordt. De timer wordt gestopt.
Als de controller in TeiAssignedState is dan zal de aankomst van een Command frame met juiste volgorde nummer en zonder data veld en met P=1 aanleiding geven tot het opzetten van een verbinding. Dit gebeurt door ControlConnection met RSISetUp op de hoogte te stellen. ConnectState wordt nu single frame. Tevens zal ControlConnection de host op de hoogte stellen met SetUpIndication = single.

In single frame toestand zal de aankomst van packet data (PacketToSend, AcceptPacket) tot gevolg hebben dat een SI command frame met P=1 verzonden moet worden. Tevens zal dit frame het packet als data veld hebben. Bij het verzenden wordt tevens de hertransmissie teller gestart (StartTimer). Als ontvangst van het frame gemeld is kan het een volgende frame door de host aangeboden worden (PacketAccepted). Bij een ontvangst van een bovengenoemd command frame zal een response terug gezonden worden. Een I frame dat aankomt (Controlfield,Data) wordt aknowledged (AckSI) met een response frame. Doorgeven van informatie aan de host vind plaats met de flows PacketReceived, PacketArrival en PacketRead.

4.8.2.3.OutPutSI

Output SI geeft de opdrachten (SIOut) aan AssembleDisAssembleFrames om frames te verzenden. Als dit gebeurt is zal dit met SISent bevestigd worden. Voor het geval er command frames zijn gezonden zal dit resulteren is een StartTimer opdracht aan AssembleDisAssembleFrames. Command frames kunnen om twee redenen verzonden worden. In de eerste plaats om data over de datalink te vervoeren. Deze frames worden verzonden na aanvragen met PacketToSend en AcceptPacket. Als het frame bevestigd is zal de host met PacketAccepted op de hoogte gesteld worden. Waarna hij eventueel een volgende frame kan aan bieden. Een tweede aanleiding om command frames te verzenden is een SetUpRequest. Deze wordt als SISetUp aangeboden. De waarde van SIToSend wordt mee gebruikt om het frame te genereren. Een command frame waarbij geen timer gebruikt wordt is het SI0 frame dat tijdens de initialisatie verzonden wordt.

4.8.2.4.InputSI

Een aankomend SI frame zal een bepaalde service moeten krijgen. Deze is afhankelijk

van het frame en van de toestand waarin de controller zich bevind. Tijdens de initialisatie fase kunnen twee types frames aankomen. In de eerste plaats SI0, dit frame geeft aan dat de peer controller om de waarde van SIToSend vraagt. RInit zal er voor zorgen dat SIInit een frame laat verzenden waarin SIToSend bevat is. Tevens is het mogelijk dat er een SI0 of een SI1 response frame aankomt. Dit frame is een antwoord op het SI0 frame dat de controller naar de peer controller heeft gestuurd. Het frame wordt gedetecteerd als AwaitSI = true. Het bevat de waarde van SIToSend van de peer controller. De waarde die het frame heeft moet in SIExpected gezet worden. Als dit gebeurt is de initialisatie afgelopen. Dit wordt met InitDone gemeld.
Na de initialisatie fase is de controller in TeiAssigned toestand. Als een peer controller in deze toestand een SI command frame stuurt met het juiste volgorde nummer en zonder data deel, dan betekend dit dat de peer controller een single frame verbinding wil opzetten. Dit wordt aan SIConnectDisConnect gemeld met RSetUp. Dit zal tot gevolg hebben dat ConnectState = SingleFrame wordt.
In deze toestand kan de data transfer plaats vinden. De frames die kunnen aankomen en de service die ze verlangen wordt in het volgende beschreven. Een command frame met een juist volgorde nummer en data zal tot gevolg hebben dat: de data doorgegeven wordt aan de host (PacketReceived, PacketArrival), SIExpected := SIExpected +1 en na overname van de data door de host PacketRead zal een response frame met waarde SIExpected terug gezonden worden. Tevens wordt AssembleDisAssembleFrame op de hoogte gesteld dat de gewenste actie uitgevoerd is (SIServiced). Als het volgorde nummer van het ontvangen frame niet juist is zal: geen data aan de host doorgegeven worden en SIExpected zal niet verhoogd worden. Wel zal een response frame (F=1) verzonden moeten worden. Dit frame zal dus de oude waarde van SIExpected bevatten. Als er een ander command frame wordt ontvangen zal ook een response frame met de waarde van SIExpected verzonden worden (F=0).
Aankomst van een response frame met een goed volgorde nummer en met F=1 zal tot gevolg hebben dat: SIToSend verhoogd wordt en dat de timer en de hertransmissie teller gereset worden. (StopTimer).
Als F=0 dan wordt ook de timer/teller gereset en zal SIToSend verhoogt worden. Tevens zal als het vorige command frame geen data bevatte het zelfde command frame nogmaals verzonden worden met een nieuw volgorde nummer. Als het vorige command frame wel data bevatte dan is het response frame een aanvraag om de verbinding te verbreken. Dus zal er een DisConnectIndication aan de host gegeven moeten worden. Response frames die niet in volgorde zijn of niet juist zijn worden niet beschouwd en hebben geen verdere actie tot gevolg.

4.9. Overwegingen met betrekking tot implementatie

Bij de overweging hoe de verdeling tussen hardware en software gemaakt moet worden, moet de snelheid waarmee processen moeten werken bekeken worden. Processen die niet snel afgewerkt moeten worden kunnen in software gerealiseerd worden. Processen waarbij wel snelle verwerking van belang is moeten in hardware gerealiseerd worden. Hier kan het parallel verwerken van informatie voor extra snelheid zorgen. Een ander aspect dat bij de implementatie beschouwd moet worden is de overdracht van informatie. Welke realisatie hier gekozen wordt is onder andere afhankelijk van de snelheid waarmee de informatie tussen de processen overgedragen moet worden. Voor langzame kanalen is het gebruik van een serieel- of een gemultiplext kanaal mogelijk. Als er van uit wordt gegaan dat de controller een ingaande en een uitgaande host data

stroom van minimaal 64k bits moet kunnen verwerken, dan zal een volledige software realisatie niet mogelijk zijn. De verwerkings tijd per bit is dan namelijk 16 microseconde. Indien echter AssembleDisAssembleFrames in hardware uit gevoerd wordt, dan zal de rest van de controller wel in software uit te voeren zijn.

5. CONCLUSIES

Het ontwerpen van een controller met de methode van Hatley en Pirbai blijkt mogelijk te zijn. Hier bij is gebleken dat niet de opdeling in processen de grootste zorg behoeft, maar dat de beschrijving van de betekenis van de flows van groter belang is.

Een facet waaraan echter vooraf aandacht moet worden geschonken is: waar wordt de analyse gestopt omdat de decompositie ver genoeg is door gevoerd. Hiermee wordt namelijk vastgelegd tot op welk diepte processen uit geplozen moeten worden. Voor dit ontwerp werd deze grens op een zeer vergaand niveau van decompositie gelegd. Dit is echter afhankelijk van de basis elementen die men voor de realisatie terbeschikking heeft.

Een ander belangrijk aspect is het vastleggen van de context. Dit dient voordat de analyse begint exact te gebeuren.

De decompositie van processen in subprocessen verloopt daarna zonder veel problemen. Hierbij bleek de opsplitsing in data verwerkende processen en control processen, die deze data processen besturen, een makkelijke realisatie mogelijk te maken. Daar decontroller sterk control georienteerde beschrijving noodzakelijk maakt is deze dan ook gebruikt, ondanks de bedenkingen die Hatley en Pirbai hiertegen hebben. Control georienteerde processen blijken met de methode namelijk goed te beschrijven. Voor de duidelijkheid is daarnaast het geven van een state diagram zeer nuttig.

De weergave van de flows en hun betekenis is een punt dat meer aandacht vraagt. Metname de betekenis van hoog niveau data flows en control flows die meerdere waardes hebben. Deze data flows kunnen het beste weergegeven worden door een tijd continue data flow voorzien van handshake signalen. De onderliggende overdracht methode is nu nog niet van belang. Hierbij verdient de control strobed realisatie de voorkeur boven een data strobed realisatie. Op een lager niveau van decompositie kan hier nog invulling aan gegeven worden. Een uitzondering hierop vormen flows welke in de context voorkomen. Deze zullen totaal uitgesplitst gegeven moeten worden. Voor meerwaardige controlflows geldt dat ze opgevat moeten worden als vectoren die door het ontvangende proces nog gedecodeerd moeten worden. Elementaire control en data flows leveren geen probleem op.
Als met de bovenstaande zaken rekening wordt gehouden is het gebruik van structured analysis voor het ontwerp van hardware bruikbaar. Er zullen echter wel strictere afspraken gemaakt moeten worden over de betekenis van flows.

LITERATUUR

[H&P]
D.J. Hatley & I.A. Pirbai,
Strategies for Real-Time system specification.
New York: Dorset House Publishing Co., 1987

[W&M]
P.T. Ward & S.J. Mellor,
Structured development for Real-Time systems.
New Jersey: Yourdon Press, 1986

[Win]
M.R.M. Winter,
Design of a universal protocol subsystem architecture. Specification of functions and services.
Eindhoven: TU-Eindhoven, 1989

[Tan]
A.S. Tanenbaum,
Computer Networks.
Englewood Cliffs: Prentice Hall, 1981

[Sta]
W. Stallings,
Data and computer communications.
New York: Macmillan Publishing company, 1985

[X200]
CCITT,
Red Book, Volume VIII - Fascicle VIII.5, Datacommunication networks open systems interconnection (OSI), System description techniques, Recommedation X.200-X.250.
Geneva: CCITT, 1985

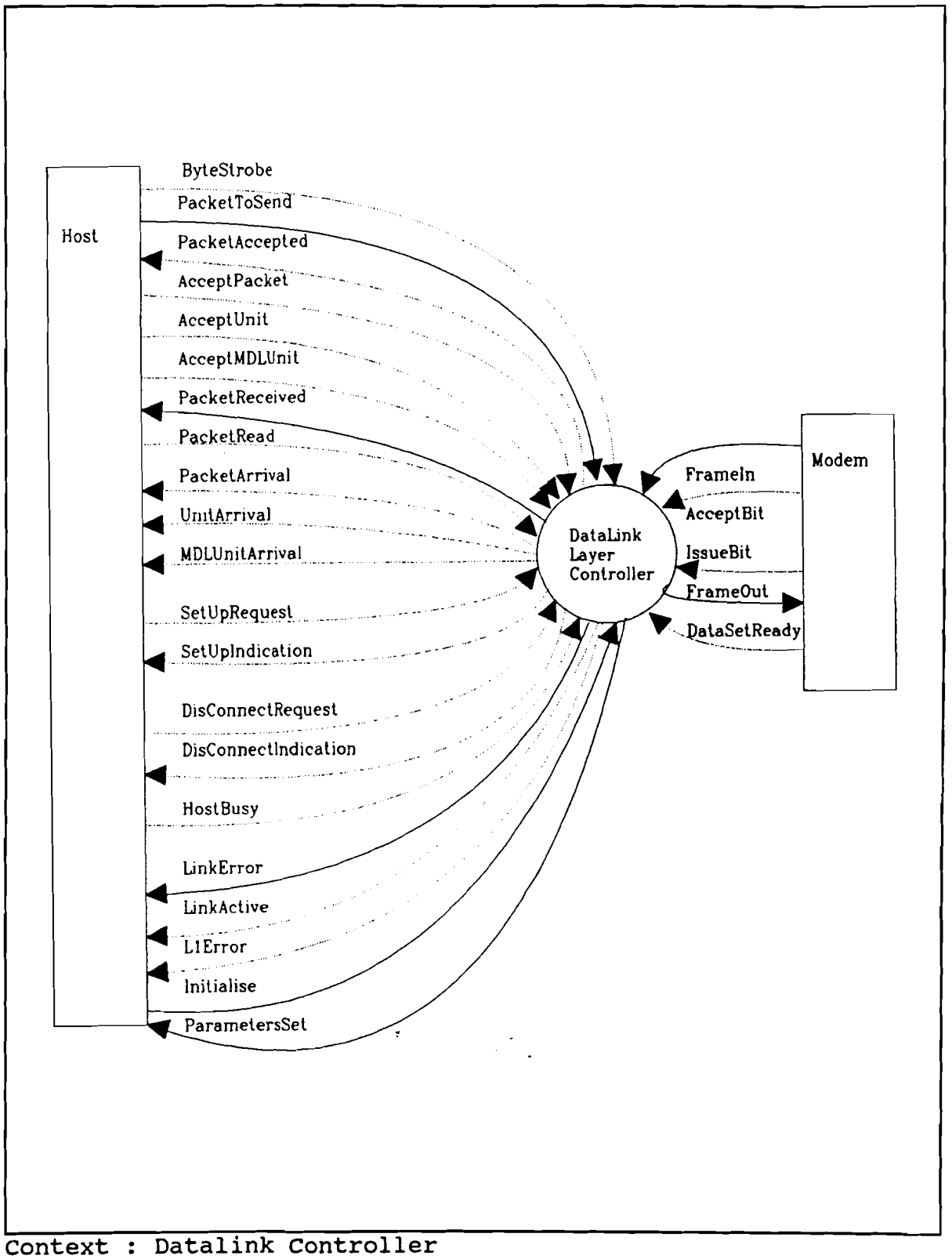
[I440]
CCITT,
Red Book, Volume III - Fascicle III.5, Integrated services digital network (ISDN), Recommedations of the series I.
Geneva: CCITT, 1985

[Svo]
L. Svobodva
Implementing OSI Systems
IEEE journal of selected areas of communication Vol.7 No.7 september 1989 pp. 1115-1130

[V&L]
C.A. Vissers & L. Logrippo
The importance of the service concept in the design of communication protocols.
Protocols specification, testing and verification.V 1986 pp.1-17

APPENDIX A : PROCESBESCHRIJVINGEN EN FLOWDIAGRAMMEN

Inhoud appendix A



Context : Datalink Controller

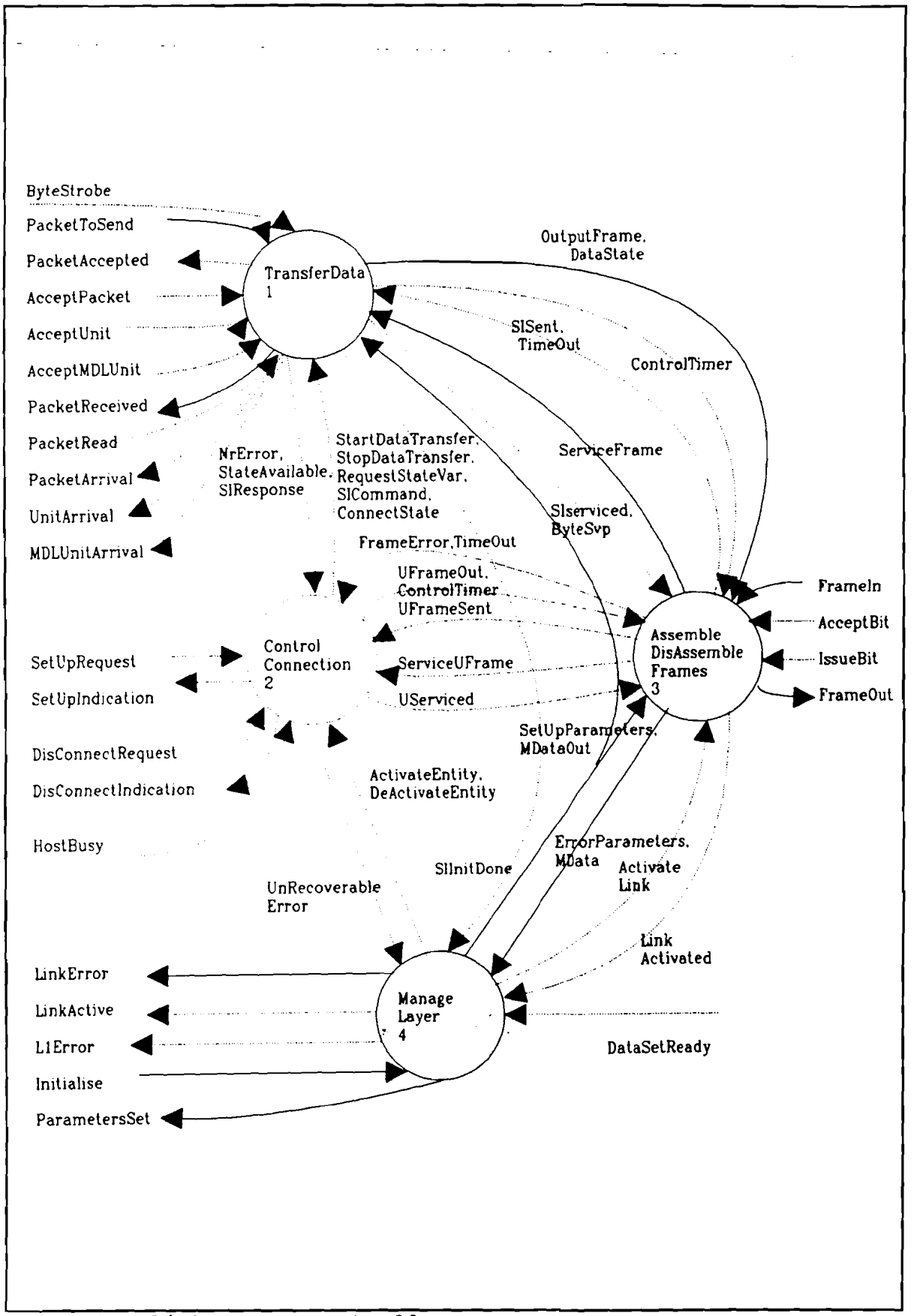
PSPEC Context : Datalink Controller

*
De Datalink controller bevindt zich in een omgeving waarin nog twee systemen te vinden zijn. De host bezit een laag 3 functionaliteit en heeft tevens een functie met betrekking tot management functies. Het laag 1 device (Modem) zorgt voor de overdracht van bits die door de datalink controller worden aangeleverd.
De Datalink controller zorgt er in deze omgeving voor dat de door de host aangegeven pakketten omgezet worden in HDLC frames. Deze frames worden op hun beurt aangeboden aan het modem. Ook de omzetting van aankomende HDLC frames naar pakketten aan de host worden door de controller gedaan. Tussen de controller en de host worden de volgende types pakketten uitgewisseld: data pakketten, commando's voor opzetten en verbreken van links, management informatie en initialisatie parameters. De overdracht van de bytes uit een pakket wordt aangegeven door de flows PacketToSend en PacketReceived. ByteStrobe geeft het tijdstip aan waarop de bytes overgedragen worden. De flows PacketAccepted en AcceptPacket zorgen voor de handshake bij te verzenden pakketten. De flows PacketArrival en PacketRead zijn de handshake signalen bij ontvangst van een pakket. Bij ontvangst van UnitData pakketten zorgen UnitArrival en PacketRead voor de handshake. MDLUnitArrival en PacketRead zijn de handshake signalen bij management pakketten. AcceptUnit en PacketAccepted zorgen voor de handshake bij het verzenden van UnitData pakketten. AcceptMDLUnit en PacketAccepted doen dit bij het verzenden van Management data. SetUpRequest, SetUpIndication, DisconnectRequest, DisconnectIndication en HostBusy hebben een functie bij het opzetten en verbreken van datalinks. Ze geven tevens aan of de link op single frame of op multiple frame basis moet werken. De overige flows tussen controller en host (LinkError, LinkActive, L1Error, Initialise, ParametersSet) zorgen voor de overdracht van management informatie en initialisatie parameters. FrameIn en FrameOut geven de bits aan die respectievelijk van het modem komen of naar het modem gaan. Clock geeft het tijdstip aan waarop de bits geldig zijn of verzonden worden. DataSetReady geeft aan dat het modem bits kan doorgeven.
*

PSPEC 0 : Datalink Controller

*
De vier processen waarin de controller is onder te verdelen zijn te zien in FD0. TransferData zorgt voor de datatransfer van : Data pakketten, UnitData Pakketten en Management pakketten. StartDataTransfer en StopDataTransfer besturen hierbij de datatransfer op multiple frame basis. SiCommand en SIResponse besturen de data overdracht op single frame basis. Hierbij speelt ook de toestand van de controller (ConnectState) een rol. NRError, RequestStateVar, StateAvailable en DataState spelen een rol bij het verwerken van niet herstelbare fouten. ControlTimer en TimeOut zorgen voor respectievelijk besturen van de timer en

terug melden van de timer toestand. Output frame en SISent zorgen voor het doorgeven van de te verzenden data en voor de bijbehorende handshake. ByteSvp geeft aan wanneer de bytes verwacht worden. SetUpParameters en SIInitDone spelen een rol bij de initialisatie van TransferData. ControlConnection zorgt voor het opzetten en verbreken van de verbindingen. Frames die nodig zijn voor het opzetten en verbreken van een multiple frame datalink worden aangevraagd met UFrameOut. Bevestiging van het verzenden gebeurt door UFrameSent. Bij het verzenden van deze informatie wordt ook gebruik gemaakt van de flows ControlTimer en TimeOut. De informatie omtrent ontvangen frames arriveert als ServiceUFrame en wordt met UServiced bevestigd. ActivateEntity en DeactivateEntity geven aan dat de TEI bepaald is en dat er nu ook andere frames dan management frames verzonden kunnen worden. De toestand Idle is dus verlaten. UnrecoverableError geeft aan dat er een fout opgetreden is die met FrameError, NrError of TimeOut gemeld is.
ManageLayer heeft een aantal functies. In de eerste plaats wordt na Initialise getest of het modem klaar is (DataSetReady). Zo nee, dat volgt een L1Error melding aan de host. Als het modem wel klaar is wordt de link in active state gebracht (ActivateLink). Als de peer dit ook doet (LinkActivated) dan zal de host met LinkActive op de hoogte gesteld worden. Anders treedt er een LinkError op. Hierna volgt het vaststellen van de TEI met MdataOut en MData. Bevestiging aan de host gebeurt met ParametersSet. Na een geslaagde poging kan TransferData en AssembleDisassembleFrames met SetUpParameters genitialiseerd worden. Met ActivateEntity wordt nu het opzetten van links mogelijk gemaakt.
*



FD0 : Datalink Layer Controller

PSPEC 1 : TransferData

*
Dit proces zorgt voor de datatransfer. Dit kan unacknowledged datatransfer zijn in de vorm van MDL UNIT Data en UNIT DATA. Het kan echter ook acknowledged datatransfer zijn. Deze transfer kan plaatsvinden op single en op multiple frame basis. In het eerste geval zal de flow control op start-stop basis werken. Bij multiple frame data transfer wordt een sliding window mechanisme toegepast. MDL Unit Data overdracht is altijd mogelijk. Unit Data overdracht en acknowledged data transfer is slechts mogelijk nadat de TEI bepaald is. Voor acknowledged data overdracht moet eerst een link op gezet worden voor het betreffende type data transfer. Data en informatie omtrent het type frame wordt met AssembleDisAssembleFrames uitgewisseld.
*

CSPEC 2 : ControlConnection

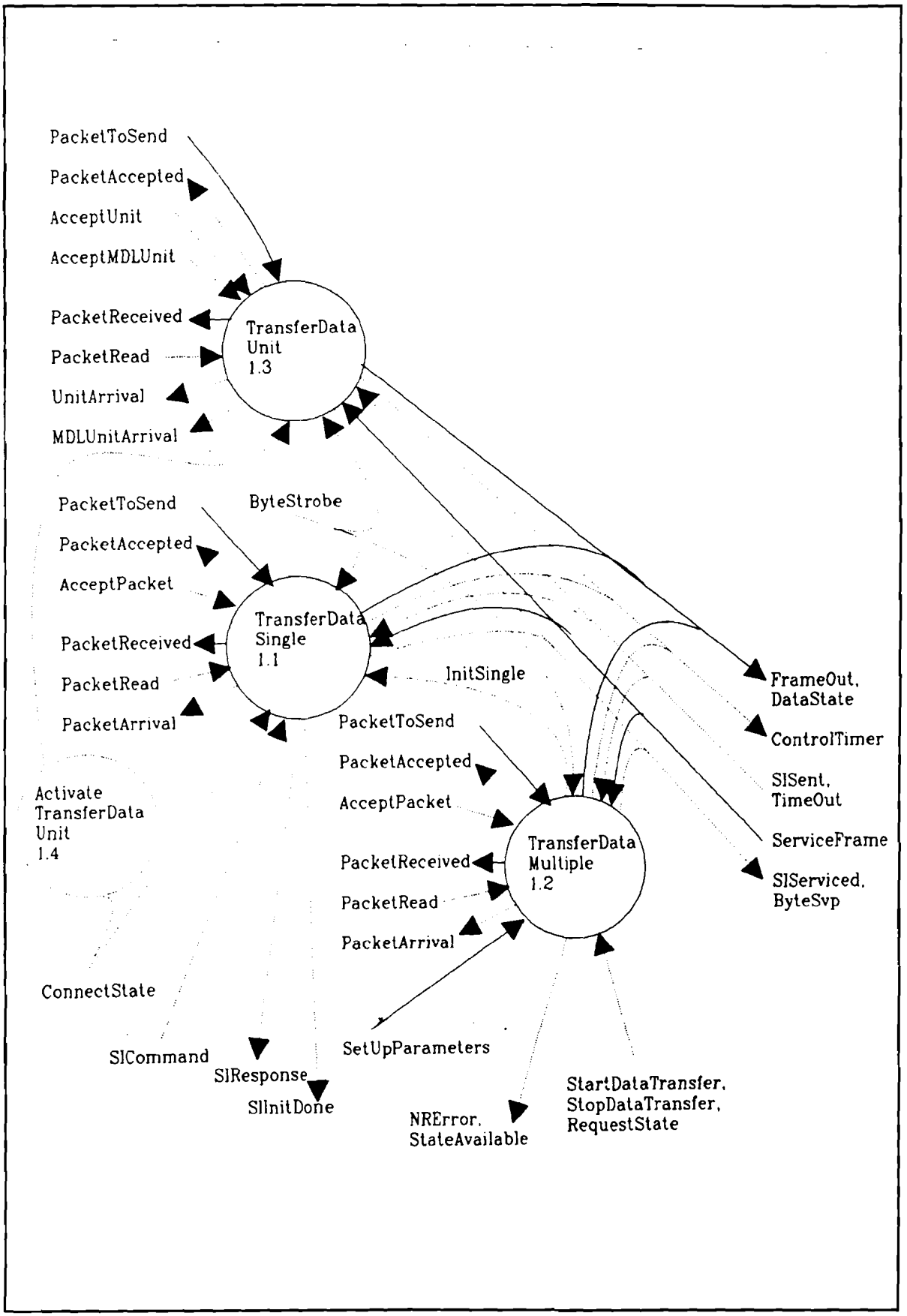
*
Dit proces zorgt voor het opzetten en verbreken van links voor acknowledged datatransfer. De aanvraag die door de host wordt gedaan wordt verwerkt. Hiervoor wordt informatie omtrent te verzenden en te ontvangen frames uitgewisseld met AssembleDisAssembleFrames. Voor het opzetten van een link op single frame basis worden tevens commands en responses uitgewisseld met TransferData. Het opzetten van een multiple frame link resulteert in het mogelijk maken van datatransfer met StartDataTransfer. De besturing van de afhandeling van fouten wordt ook door dit proces verzorgt.
*

PSPEC 3 : AssembleDisassembleFrames

*
Dit proces zorgt voor het samenstellen en decoderen van de HDLC frames. De laag niveau functies zoals bitstuffing, CRC testen en sync detectie vinden hier plaats. Tevens worden de timers hier bij gehouden.
*

PSPEC 4 : ManageLayer

*
Dit proces zorgt voor de initialisatie van de controller en voor de uitwisseling van management informatie met de host. De initialisatie begint met het doorgeven van parameters aan andere processen. Tevens wordt getest op DataSetReady en ActiveLink. De TEI bepaling wordt voor een deel ook door dit proces gedaan. Het door geven van fout informatie hoort ook bij de functies van dit proces.
*



FD 1 : TransferData

PSPEC 1.1 : TransferDataSingle

*
InitSI zal na aanvraag met InitSi zorgen voor de besturing van de initialisatie procedure. Het einde van de initialisatie wordt met SIInitDone gemeld. SIConnectDisConnect zorgt voor de besturing van het opzetten van een link op single frame basis. InputSI en OutputSI zorgen respectievelijk voor het ontvangen en verzenden van pakketten op single frame basis.
*

PSPEC 1.2 : TransferDataMultiple

*
De data transfer op multiple frame basis wordt door dit proces uitgevoerd.
*

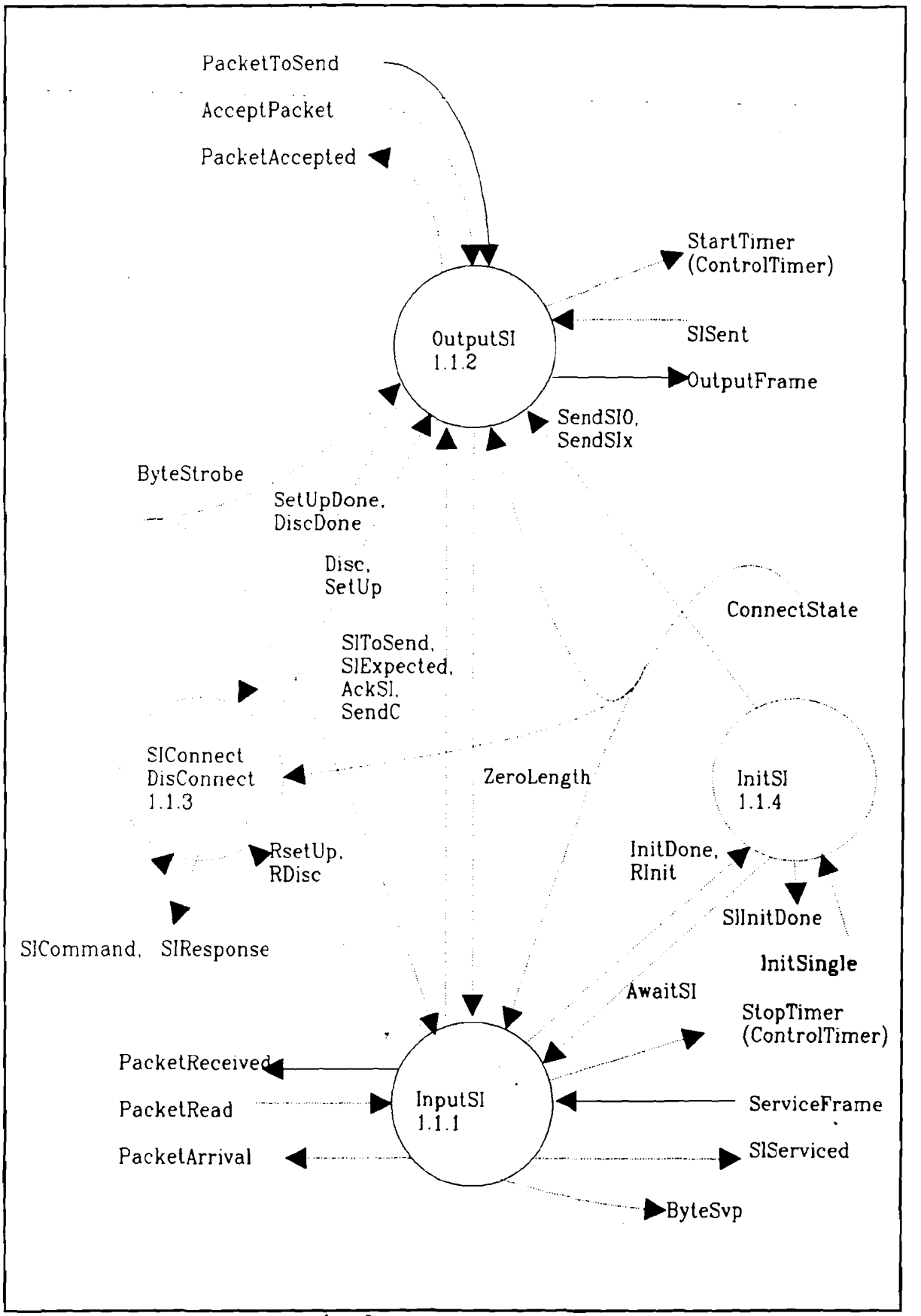
PSPEC 1.3 : TransferDataUnit

*
De unacknowledged data transfer wordt door dit proces uitgevoerd.
*

CSPEC 1.4 : ActivateTransferDataUnit

*
Als ConnectState aangeeft dat de TEI bekend is, dan zal unacknowledged data transfer mogelijk gemaakt worden.
*

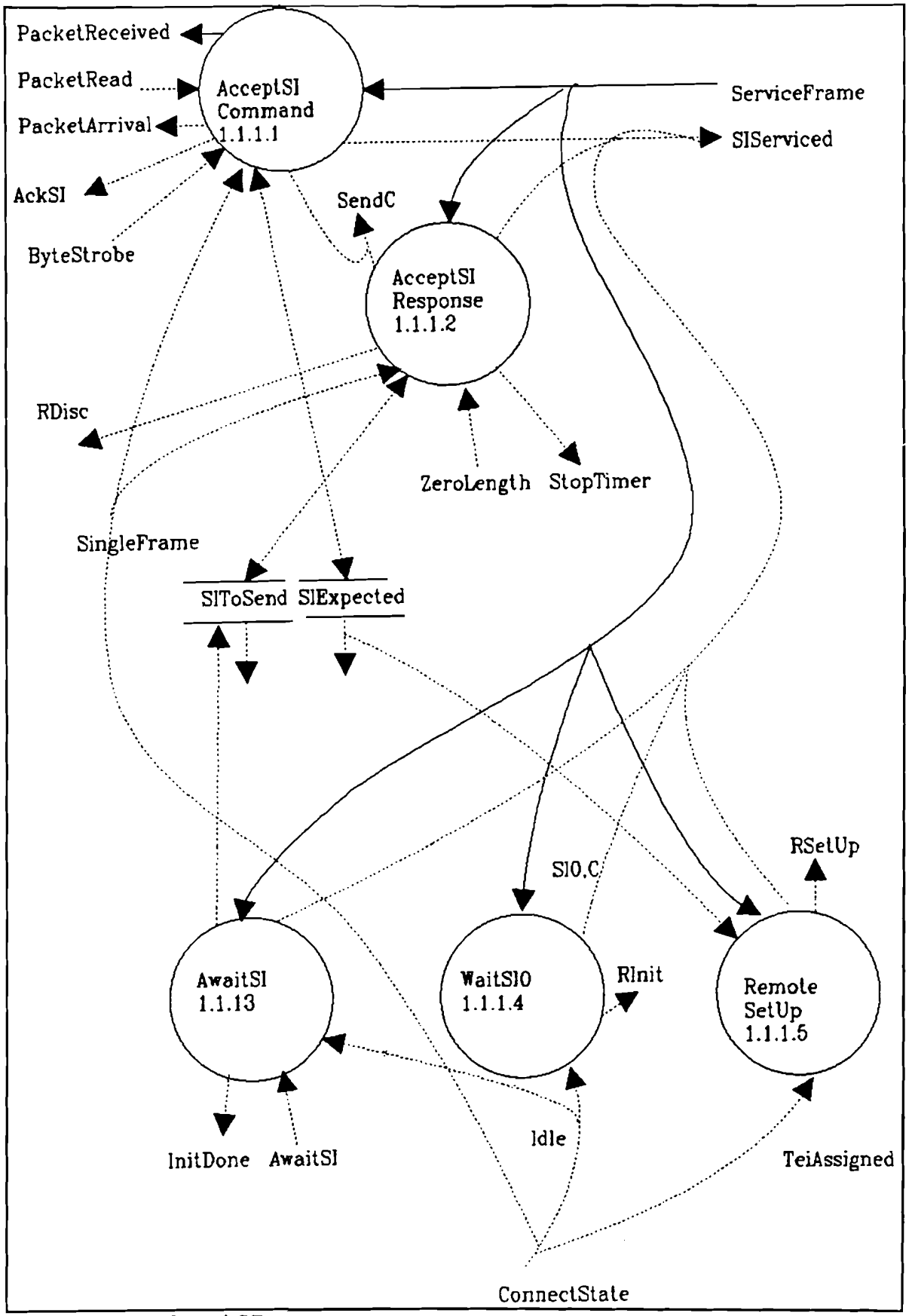
```
if ConnectState<>Idle
then activate TransferDataUnit
```



FD 1.1 : TransferDataSingle

```
PSPEC 1.1.1 : InputSI

*
In ConnectState is Single zullen de SI command en response frames
die gebruikt worden bij de single frame data transfer afgehandeld
worden. Bij ConnectState=Idle wordt het frame dat tijdens de
initialisatie verwacht wordt gedetecteerd.
*

PSPEC 1.1.2 : OutputSI

*
Afhankelijk van de toestand en de opdracht zal dit proces met
OutputFrame aangeven dat een SI frame verzonden moet worden.
*


CSPEC 1.1.3 : SIConnectDisConnect

*
Zorg voor het opzetten en verbreken van een single frame als dit
door de lokale host of door de remote host wordt aangevraagd.
*

if SICommand = SISetUp
then issue SetUp = true

if SetUpDone
then issue SIResponse = SISetUpDone

if SICommand = SIDisc
then issue Disc

if DiscDone
then SIResponse = SIDiscDone

if RSetUp and TeiAssigned
then issue SIResponse = RSISetUp

if RDisc and SingleFrame
then issue SIResponse = RSIDisc


PSPEC 1.1.4 : InitSI

*
Bestuur de initialisatie van het single frame data transfer
proces.
*

if SetParameters
then issue SendSI0
     set AwaitSI = true

if RInit
then issue SendSIx
```

```
     issue SIInitDone = 2

if InitDone
then set AwaitSI = false
     issue SIInitDone = 1
```



FD 1.1.1 : InputSI

```
PSPEC 1.1.1.1 : Accept SI Command

*
Zorg voor de handshake met de host en voor het doorgeven van
bytes totdat het buffer leeg is. Dit wordt met ControlField in
ServiceFrame aangegeven.
*

If SingleFrame and SIExpected = 0 and Controlfield = SI0
then set PacketArrival = true
     issue SIExpected = 1

If SingleFrame and SIExpected = 1 and Controlfield = SI1
then set PacketArrival = true
     issue SIExpected = 0

If PacketReceived
then issue AckSI = Fis1

If (SingleFrame and SIExpeced = 0 and Controlfield = SI1) or
(SingleFrame and SIExpected = 1 and Controlfield = SI0)
then issue SISent = true
     issue AckSI = Fis1

If SingleFrame and C/R = C
then issue SISEnd = true
     issue AckSI = Fis0

if PacketRead
then set SISent = true
     ByteSvp = ByteStrobe

if PacketRead  and ByteStrobe
then set PacketReceived = Data

if not Controlfield = SI0 and not Controlfield = SI1
then set PacketArrival = false
     set ByteSvp = false
```

PSPEC 1.1.1.2 : Accept SI Response

*
Zorg voor de afhandeling van SI Response frames.
*

C/R = R

In : ZeroLength, Controlfield, SIToSend, F
Uit : StopTimer, RDisc, SIToSend, SIServiced, SendC

In:				Uit:				
ZL ee rn og t h	CF oi ne tl rd o l	SS Ie Tn od S e n d	F	iS st so up eT i m e r	iR sD si us ec	iS sI sT uo eS e n d	iS sI sS ue er v i c e d	iS se sn ud eC
-	SI0	1	1	true	false	0	true	false
true	SI0	1	0	true	false	0	true	true
true	SI0	1	0	true	true	0	true	false
-	SI1	0	1	true	false	1	true	false
true	SI1	0	0	true	false	1	true	true
true	SI1	0	0	true	true	1	true	false
-	SI1	1	-	-	-	-	true	-
-	SI0	0	-	-	-	-	true	-
-	other response	-	-	-	-	-	true	-

PSPEC 1.1.1.3 : AwaitSIx

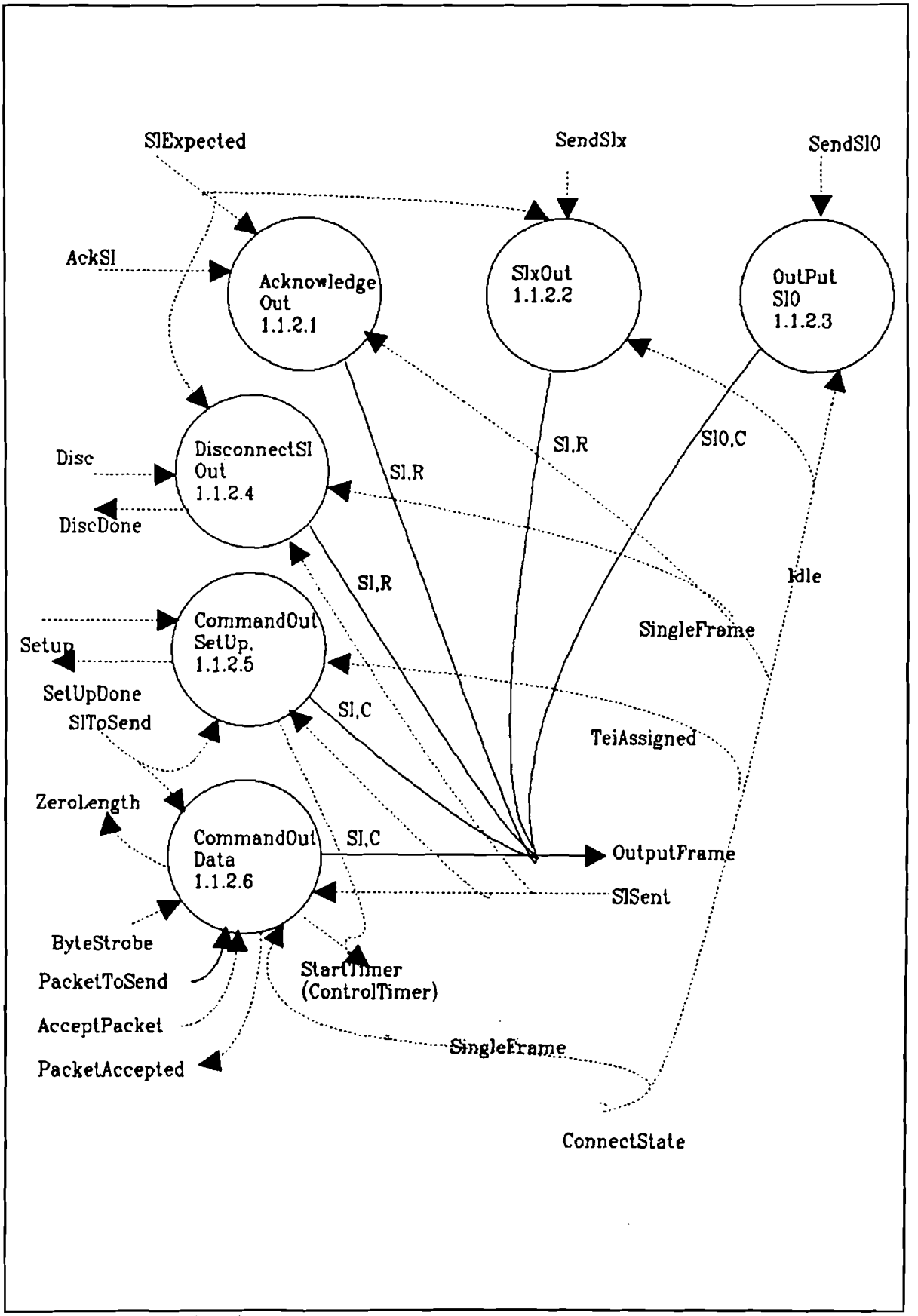
*
Wacht tijdens de initialisatie op een frame met het juiste volgorde nummer. Geeft bij aankomst door dat de initialisatie gebeurt is (InitDone).
*

```
if AwaitSI and ControlField = SI0,R
then issue SIToSend = 0
     issue InitDone = true
     issue SISServiced = true

if AwaitSI and ControlField = SI1,R
then issue SIToSend = 1
     issue InitDone = true
     issue SISServiced = true
```

PSPEC 1.1.1.4 : WaitSI0

*

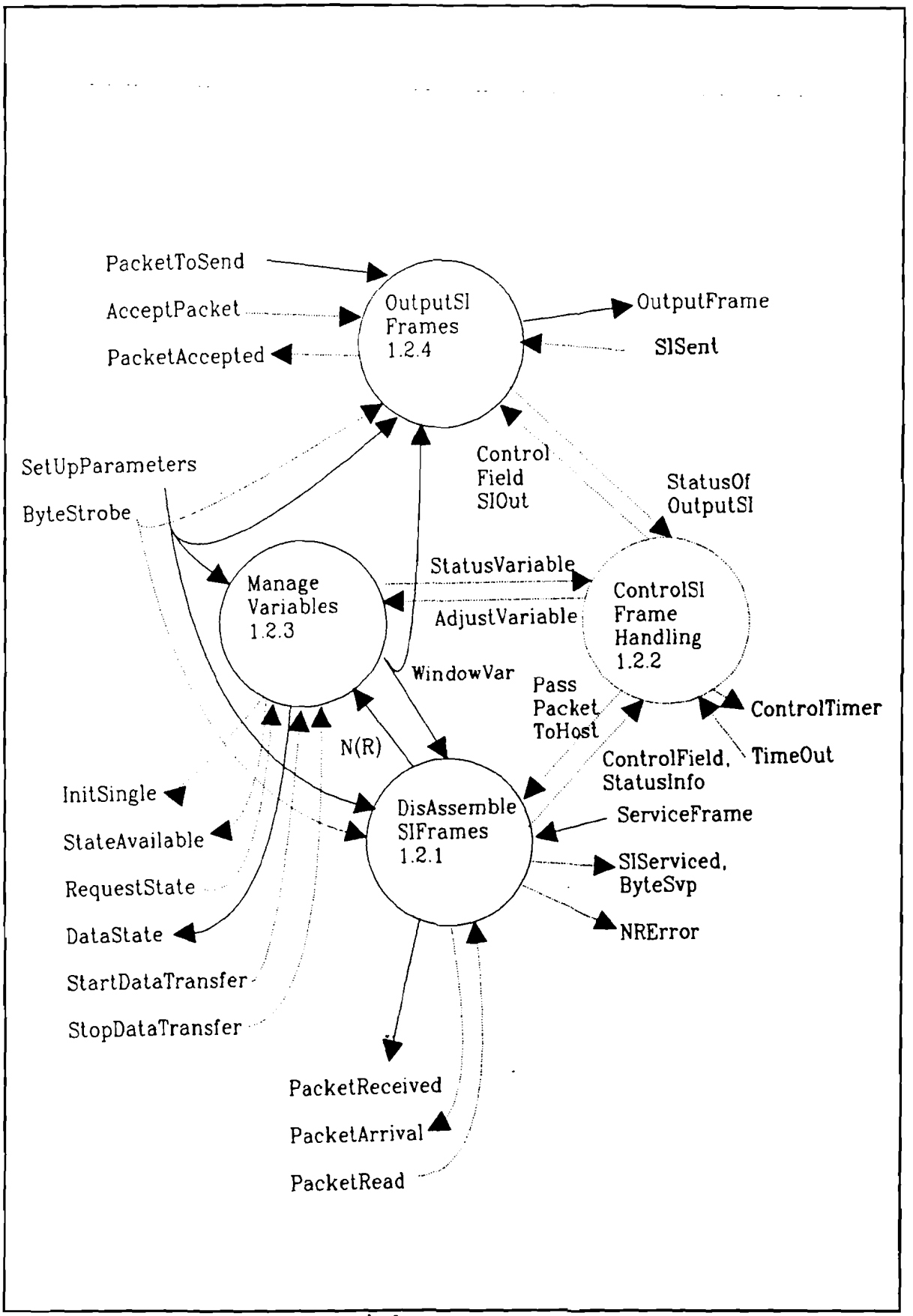
```
Het eerste frame dat moet aankomen tijdens de initialisatie is
(Idle) een SI0 command frame. Deze aankomst wordt met RInit door
gegeven aan ControlConnection
*

if Idle and ControlField = SI0,C
then issue RInit = true
     issue SIServiced = true


PSPEC 1.1.1.5 : RemoteSetUp

*
Aankomst van een frame met het juiste frame nummer in TEI
unassigned toestand betekent dat de remote controller een link
wil opzetten.
*

if TeiAssigned and ControlField = SI0,C and SIExpected = 0
then issue RSetUp = true
     issue SIServiced = true

if TeiAssigned and ControlField = SI1,C and SIExpected = 1
then issue RSetUp = true
     issue SIServiced = true
```



FD 1.1.2 : OutputSI

```
PSPEC 1.1.2.1 : AcknowledgeOut

*
Een frame met het juiste volgorde nummer wordt bevestigd.
*

if AckSI and SingleFrame
then issue SI = SIExpected,AckSI,R


PSPEC 1.1.2.2 : SIxOut

*
Zend tijdens de initialisatie de lokale variabel naar de peer.
*

if SendSIx and Idle
then issue SI = SIExpected,F = 1,R


PSPEC 1.1.2.3 : OutPutSI0

*
Vraag tijdens de initialisatie de toestand van de peer zend
variabele op.
*

if SendSI0 and Idle
then issue SI = SIExpected,F = 0,R


PSPEC 1.1.2.4 : DisConnectSIOut

*
Verzend bij een lokale disconnect aanvraag een disconnect frame
naar de peer controller.
*

if Disc and SingleFrame
then issue SI = SIExpected,F = 0,R


PSPEC 1.1.2.5 : CommandOutSetUp

*
Laat bij een SetUp een frame verzenden met het juiste volgorde
nummer en P=1. Wacht daarna op de bevestiging van het verzenden
(SISent) en meldt dit met SISetUpDone.
*

if SetUp and TeiAssigned
then issue SI = SIToSend,P = 1,C

if SISent and TeiAssigned
then issue StartTimer
     issue SetUpDone
```

```
PSPEC 1.1.2.6 : CommandOutData

*
Verzendt een single frame data pakket.
*

If AcceptPacket and SingleFrame
then issue SI = SIToSend,P = 1,C,PacketToSend
     set ZeroLength = false

If SISent and SingleFrame
then issue StartTimer
     issue PacketAccepted = true
```



FD 1.2 : TransferDataMultiple

PSPEC 1.2.1 : DisAssembleSIFrame

*
Dit proces herkent welk type frame aangekomen is en splitst het controlfield op in de samenstellende delen: type veld, sequence nummer, poll of final indicatie. De Sequence nummers worden vergeleken met de variabelen en de uitkomst van deze vergelijking wordt met de StatusInfo door gegeven aan ControlSIFrameHandling. Als het een I frame betreft wordt dit met PacketArrival aan de host gemeld. Als deze bereid is de data over te nemen dan wordt dit met PacketRead gesignaliseerd. Dit wordt met SIServiced terug gemeld. Nu kunnen de bytes in ByteStrobe tempo door gegeven worden. Als de buffers leeg zijn verdwijnt de signalisering dat een I frame aangekomen is in ServiceFrame en zal PacketReceived dit aan de host melden. Als de host weer in staat is om data op te nemen wordt dit door waarde verandering van PacketRead aangegeven. Een fout in het aankomende frame zal tot gevolg hebben dat SIServiced direct na de eerst toestand verandering weer van toestand veranderd.
*

CSPEC 1.2.2 : ControlSIFrameHandling

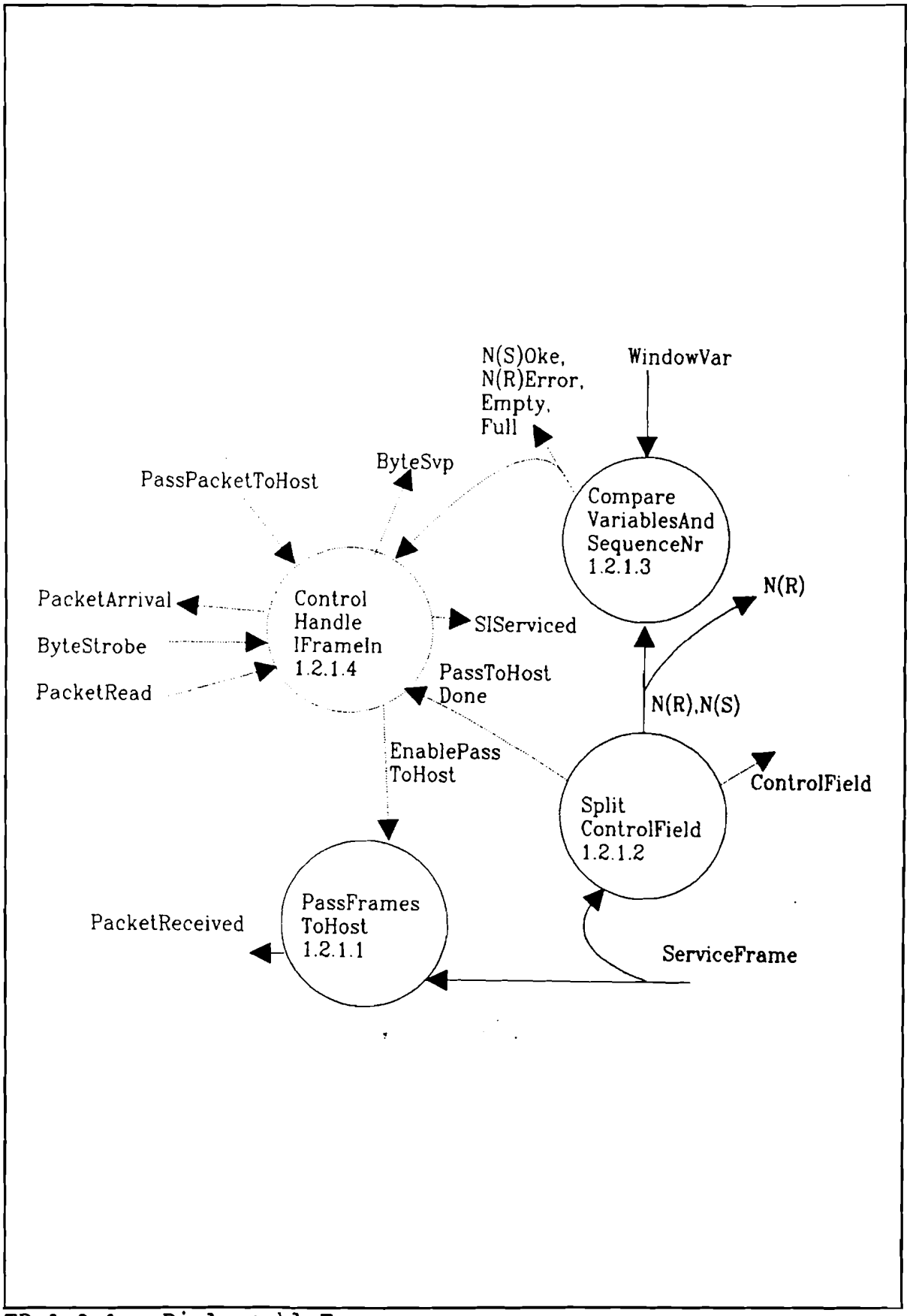
*
Dit proces bestuurt de data transfer op multiple frame basis. Hierbij wordt gebruikt gemaakt van de frame types I, RNR, RR en REJ. Het gedrag van dit proces wordt beschreven in I.441. Om dit gedrag te kunnen realiseren is het proces op de in FD 1.2.1 gegeven wijze opgedeeld. De flow ControlField bevat hier de gedecodeerde informatie van het aangekomen controlfield. EnablePassToHost geeft aan dat een I frame doorgegeven kan worden aan de host. AdjustVariables geeft opdracht om de variabelen aan te passen. ControlTimer1 bestuurt de timer. De TimeOut condities worden met TimeOut1 gemeld. SIFramesOut geeft aan welk frame verzonden moet worden. StatusOfOutputSI rapporteert de toestand van OutputSIFrames.
*

PSPEC 1.2.3 : ManageVariables

*
De window variabelen (NextFrameToSend, LastAck, FrameExpected) worden door dit proces bij gehouden en aangepast (AdjustVariables). Met StartDataTransfer worden ze op de nul waarden genitialiseerd. Met RequestDataState wordt de waarde van de variabelen aangevraagd. Als ze met DataState klaar staan wordt dit met StateAvailable gemeld. SetUpParameters geeft informatie omtrent de window grootte en het aantal hertransmissie pogingen. Tevens wordt hierdoor met InitSingle de initialisatie van TransferDataSingle gestart.
*

PSPEC 1.2.4 : OutPutSIFrames

```
*
Dit proces zorgt voor de buffering van te verzenden data. tevens
wordt de handshake met de host hier uitgevoerd. Het controlfield
wordt hier ook samengesteld.
*
```

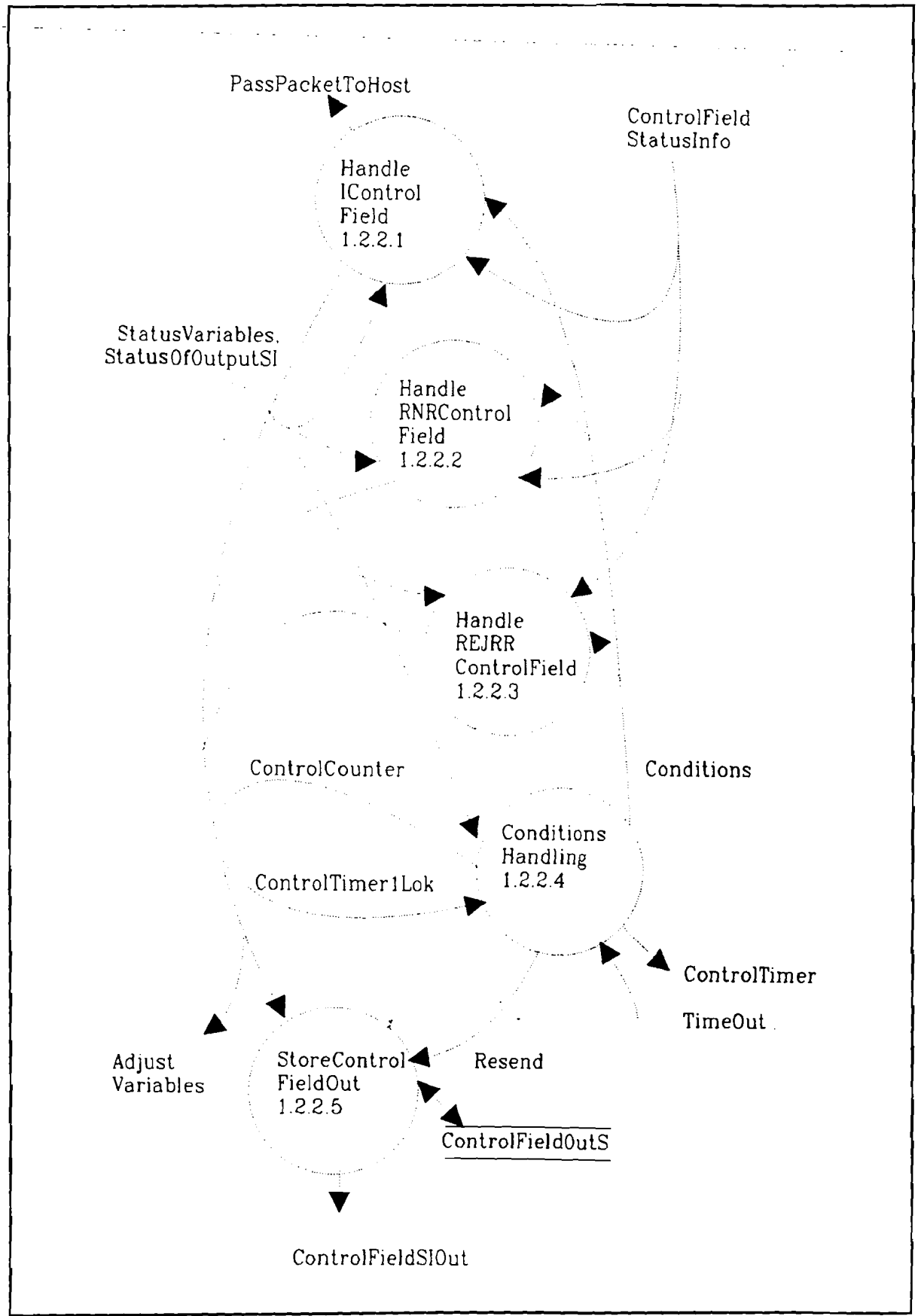


FD 1.2.1 : DisAssembleFrame

```
PSPEC 1.2.1.1 : PassFrameToHost

*
Geef de bytes die de pakket informatie vormen door.
*

if EnablePassToHost
then set PacketReceived = PacketData


PSPEC 1.2.1.2 : SplitControlField

*
Het aangeboden controlfield wordt gesplitst in zijn componenten,
dit zijn: het frame type, de waarde van het poll final bit en de
eventuele volgorde nummers N(S) en N(R). Tevens wordt door
gegeven of het een command of response frame is. StatusInfo geeft
de toestand van de buffer en van de sequence nummers aan.
*

if ServiceFrame = I
then set PassToHost = true
else set PassToHost = false


PSPEC 1.2.1.3 : CompareVariablesAndSequenceNr

*
De aankomende sequence nummers worden vergeleken met de window
variabelen. De uitkomst hiervan wordt doorgegeven aan
ControlSIFrameHandling. Tevens wordt de toestand van de buffer
bepaald aan de hand van de window variabelen.
*

if LastAck < N(R) <= NextFrameToSend (Modulo 8/128)
then issue NrError = false
else issue NRError = true

if N(S) = FrameExpected
then issue NSOke = true
else issue NSOke = false

if LastAck = NextFrameToSend - 1 (Modulo 8/128)
then set Empty = true
else set Empty = false

if LastAck + MaxWindowSize = NextFrameToSend (Modulo 8/128)
then set Full = true
else set Full = false


PSPEC 1.2.1.4 : ControlHandelIFrameIn

*
Als met PassPacketToHost aangegeven wordt dat een pakket
```

doorgegeven kan worden aan de host zal dit proces de host met PacketArrival op de hoogte stellen. Na ontvangst van PacketRead kan de host de data bytes overnemen, deze worden met ByteSvp aangevraagd. Als control informatie weg is zal hiermee gestopt worden. Dit wordt aan de host meegedeeld met PacketArrival = false. SIServiced veranderd ook van waarde. Ook PacketRead wordt nu false. Bij een fout N(R)Error of N(S)Oke = false en als SIServiced kort true wordt om aan te geven dat het frame nier doorgegeven wordt.
*

```
if PassPacketToHost
then set PacketArrival = true

if not PassPacketToHost
then set EnablepassToHost = false
     set PacketArrival = false

if PacketRead
then set ByteSvp = ByteStrobe
     set EnablePassToHost = true
```



FD 1.2.2 : ControlFrameHandling

CSPEC 1.2.2.1 : HandleIControlField

*
Bestuur de verwerking van een aangekomen I frame
*

CSPEC 1.2.2.2 : HandleRNRFrame

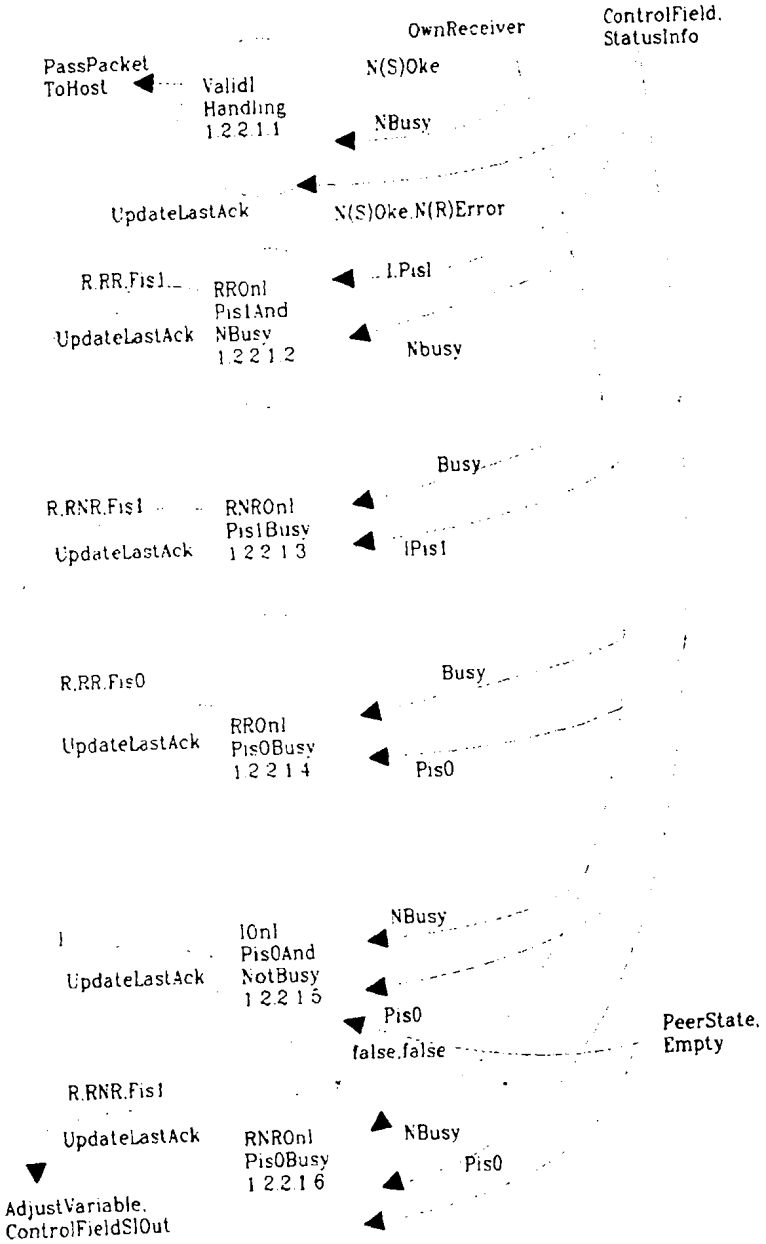
*
Bij aankomst van een RNR frame zal dit proces zorgen voor de juiste afhandeling van het frame.
*

CSPEC 1.2.2.3 : REJRRHandling

*
Zorgt er voor dat een ontvangen REJ of RR frame verwerkt wordt.
*

CSPEC 1.2.2.4 : ConditionHandling

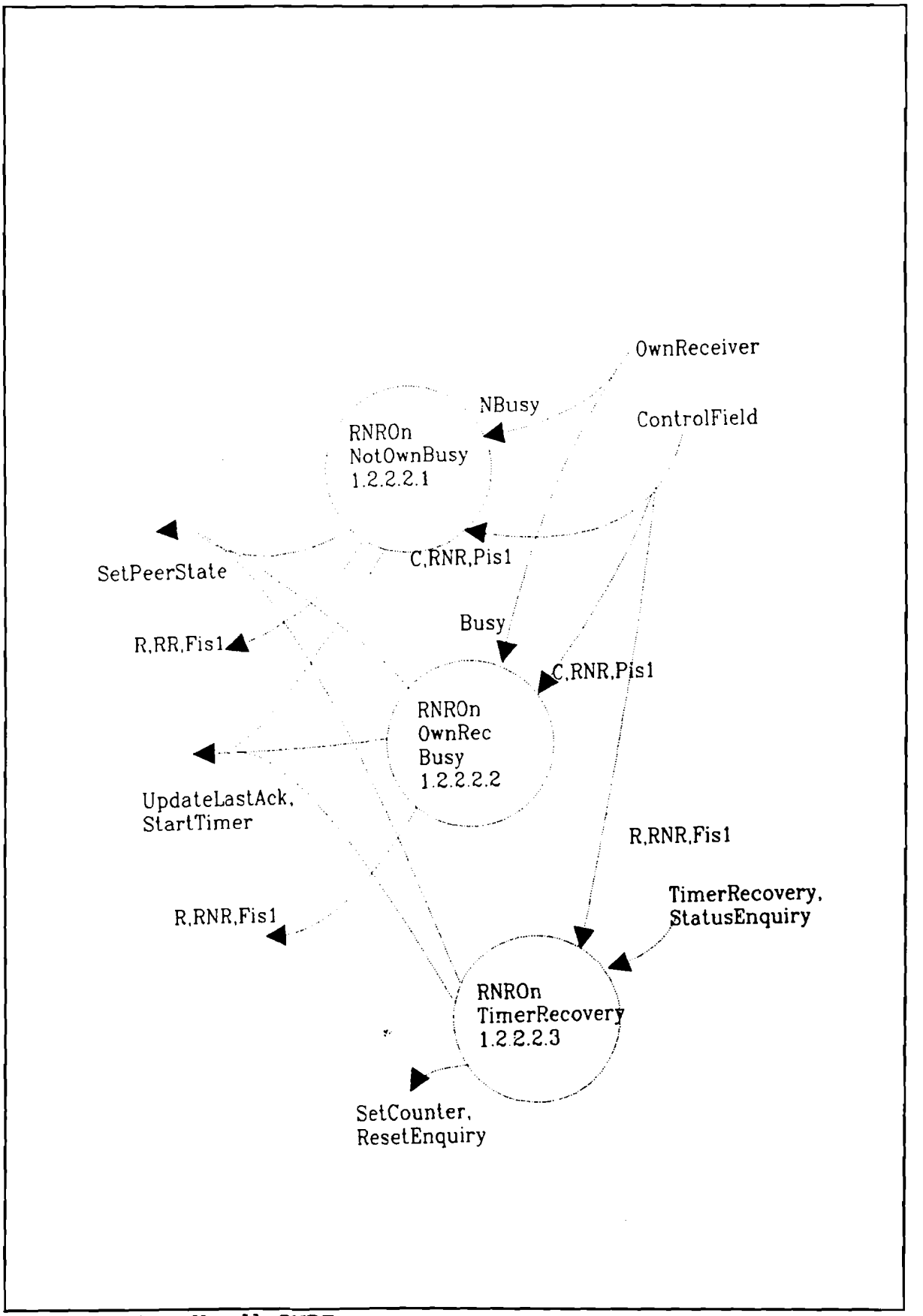
*
Zet de condities van HandleSIFrames
*



FD 1.2.2.1 : HandleArrivingFrames

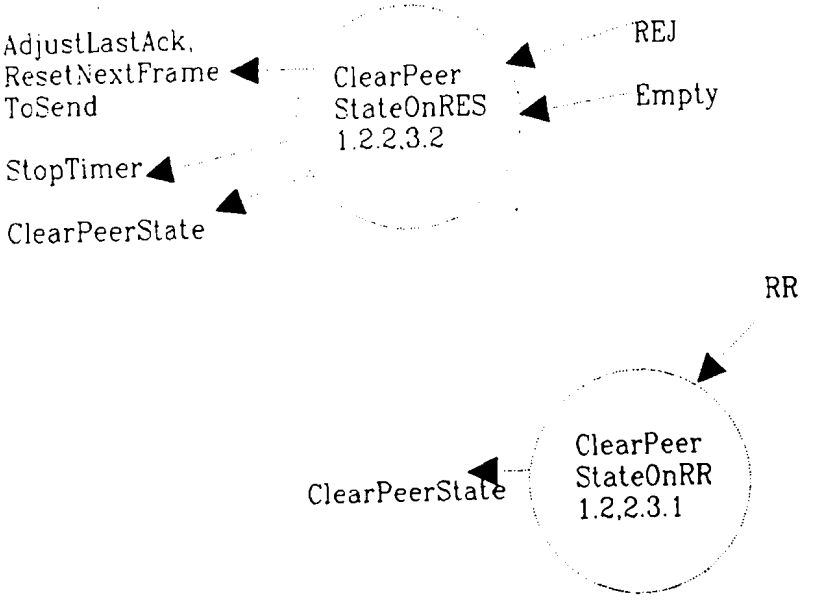
```
CSPEC 1.2.2.1.1 : ValidIHandling

*
Als een correct I frame met correct sequence nummer aangekomen
is wordt dit aan de host door gegeven. Zolang dit niet bevestigd
is wordt de toestand OwnReceiver = Busy
*

if NBusy and N(S) and not N(R)Error
then set PassPacketToHost = true
     issue UpdateLastAck = true
     issue UpdateFrameExpected = true


CSPEC 1.2.2.1.2 : RROnPis1andNBusy

*
Een I frame met P=1 komt aan als de receiver not busy is. Nu
moet een RR response frame met F=1 verzonden worden. Tevens moet
LastAck aangepast worden.
*
if I and Pis1 and NBusy
then issue ControlfieldSiOut = R,RR,Fis1
     issue UpDateLastAck = true


CSPEC 1.2.2.1.3 : RNROnIPis1Busy

*
Een I frame met P=1 komt aan terwijl OwnReceiver = Busy. Dan
moet er een RNR response met F=1 verzonden worden. Tevens moet
LastAck bij gewerkt worden.
*

if I and Pis1 and Busy
then issue ControlfieldSIOut = R,RNR,Fis1
     issue UpdateLastAck = true


CSPEC 1.2.2.1.4 : RROnIPis0Busy

*
Een Iframe met P=1 komt aan terwijl OwnReceiver = Busy. Nu moet
een RR response met Fis0 verzonden worden. Ook moet LastAck
aangepast worden.
*

if I and Pis0 and Busy
then issue ControlFieldSIOut = R,RR,Fis0
     issue UpdateLastAck = true


CSPEC 1.2.2.1.5 : IonIandNotBusy

*
Een I frame wordt veranderd als een I frame met P=0 aangekomen
```

```
is en de peer not busy is. Update van LastAck vindt plaats.
*

if I and not Empty and PeerNotBusy
then issue ControlfieldSIOut = I, Pis0
     issue UpdateLastAck = true


CSPEC 1.2.2.1.6 : RNROnIPis0Busy

*
Als een I met P=0 aankomt terwijl OwnReceiver = Busy dan moet
een RNR response met F=1 verzonden worden.
*

if I and Pis0 and Empty
then issue ControlFieldSIOut = R,RNR,F=1
     issue UpDateLastAck = true
```



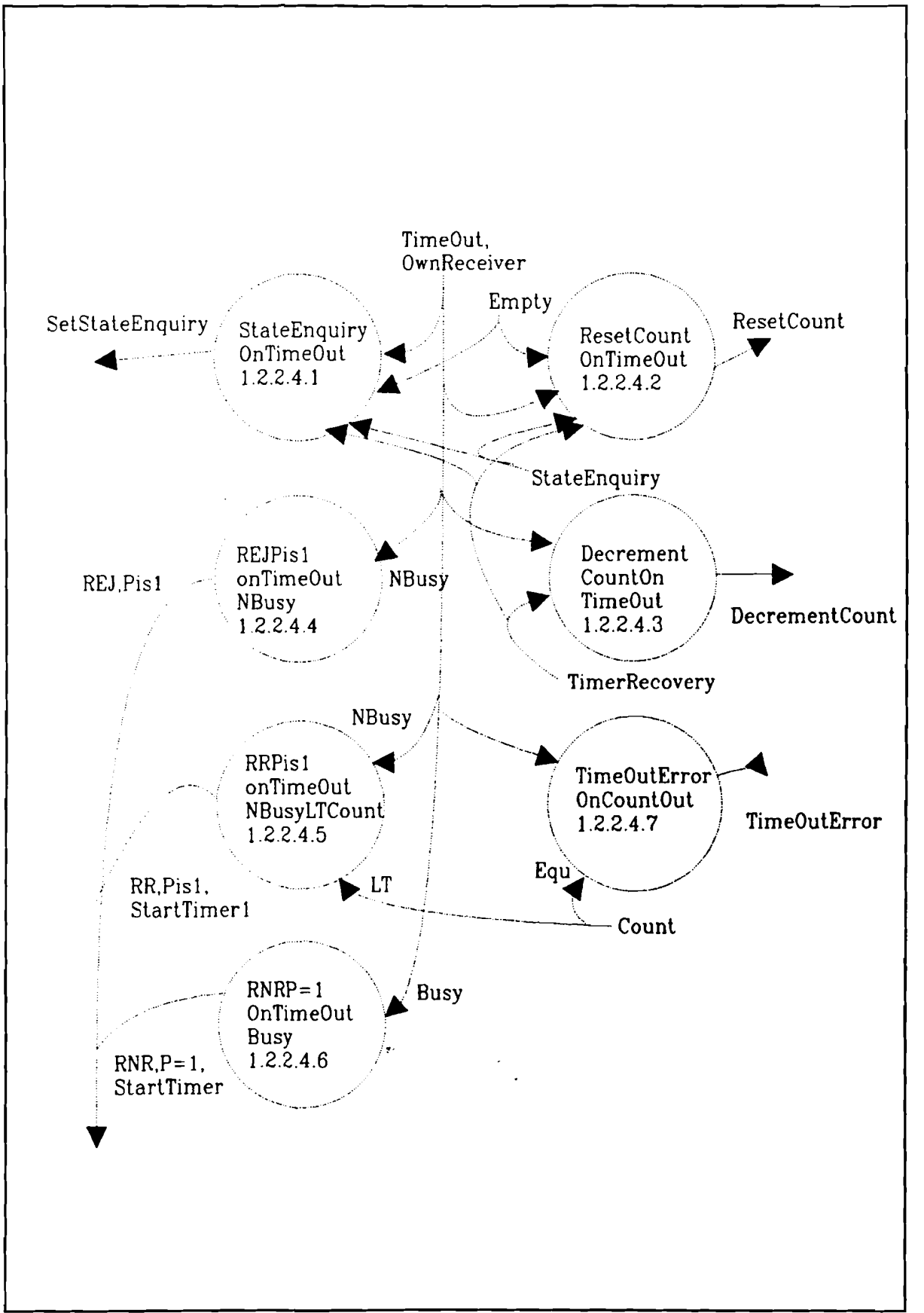
FD 1.2.2.2 : HandleRNRFrame

```
CSPEC 1.2.2.2.1 : RNROnNotOwnBusy

*
Aankomst van een RNR command met P=1 als OwnReceiver = NBusy
heeft tot gevolg dat: PeerState = PeerBusy wordt, UpdateLastAck
en dat de timer wordt gestart.
*

if NBusy and C and RR and Pis1
then issue SetPeerState = Busy
     issue UpdateLastAck = true
     issue StartTimer1 = true
     issue ControlFieldSIOut = R,RR,F=1


CSPEC 1.2.2.2.2 : RNROnOwnBusy

*
Aankomst van een RNR command net P=1 als OwnReceiver = not Busy
resulteert in een RNR response met F=1, LastAck update en
StartTimer.
*

if Busy and C and RNR en Pis1
then issue ControlFieldSIOut = R,RNR,Fis1
     issue UpDateLastAck = true
     issue StartTimer1 = true


CSPEC 1.2.2.2.3 : RNROnTimerRecovery

*
Verzend een RNR als TimerRecovery of Status enquiry plaats vindt.
*

if R and RNR and Fis1 and TimerRecovery or StatusEnquiry
then issue SetCounter = true
     issue ResetEnquiry = true
```



FD 1.2.2.3 : REJRRHandling

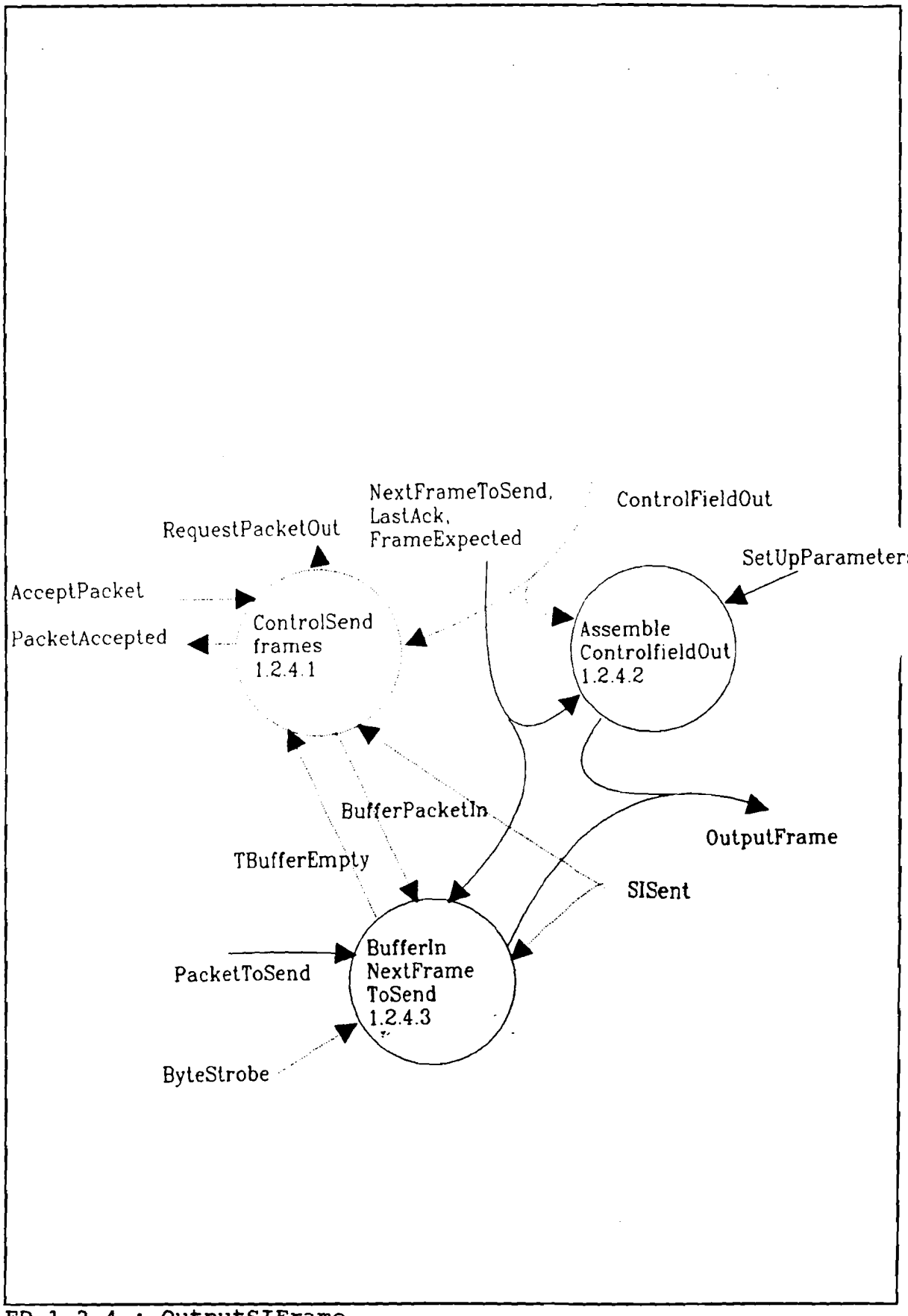
```
CSPEC 1.2.2.3.1 : ClearPeerStateOnRR

*
Als een RR frame aankomt dan moet PeerBusy false worden
*

if RR
then issue ClearPeerState = true


CSPEC 1.2.2.3.2 : ClearPeerStateOnREJ

*
Als een REJ frame aankomt en het zend buffer is leeg, dan zal
PeerBusy gereset worden. Tevens wordt de timer gestopt en zullen
de window variabelen aangepast moeten worden.
*

if REJ and Empty
then issue AdjustLastAck = true
     issue ResetNextFrameToSend = true
     issue StopTimer = true
     issue ClearPeerState = true
```



FD 1.2.2.4 : ConditionsHandling

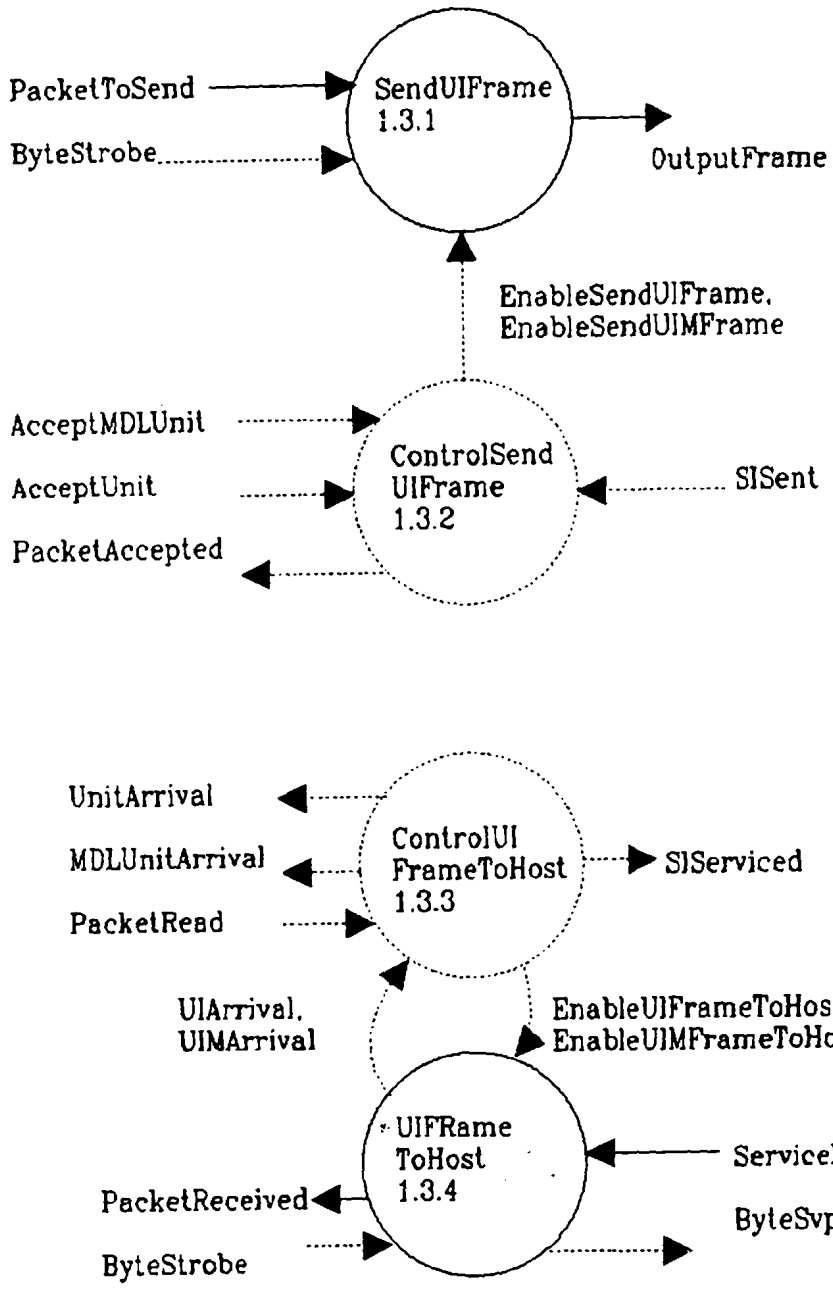
```
CSPEC 1.2.2.4.1 : StateEnquiryOnTimeOut

*
Een timeout die optreedt als de condition niet StateEnquiry of
TimerRecovery is heeft tot gevolg dat de conditie StateEnquiry
gezet wordt.
*

if  not StateEnquiry and not TimerRecovery
then set StateEnquiry = true


CSPEC 1.2.2.4.2 : ResetCountONTimeOut

*
Een TimeOut die optreedt als de buffer leeg is heeft tot gevolg
dat de counter gereset wordt.
*

if Empty and TimeOut
then issue ResetCount = true


CSPEC 1.2.2.4.3 : IncrementCountOnTimeOut

*
Een timeout die optreedt tijdens timerrecovery conditie heeft
een verlaging van de teller tot gevolg
*

if TimeOut and TimerRecovery
then issue DecrementCount = true


CSPEC 1.2.2.4.4 REJPis1OnTimeOutNBusy

*
TimeOUt als OwnReceiver = not Busy heeft het verzenden van een
REJ command met P=1 tot gevolg.
*

if TimeOut and NBusy
then issue ControlFieldSiOut = C,REJ,Pis1


CSPEC 1.2.2.4.5 : RRPis1OnTimeOutNBusyLTCount

*
Een RR Command met P=1 wordt verzonden als, een timeout optreedt
en OwnReceiver = not Busy en de hertransmissie teller is nog niet
op nul gekomen.
*

if TimeOut and NBusy and LT
then issue ControlFieldSIOut = C,RR,Pis1
```

```
CSPEC 1.2.2.4.6 : RNRP=1OnTimeOutBusy

*
Een RNR Command met P=1 wordt verzonden als een timeout optreedt
bij OwnReceiver = Busy
*

if TimeOut and Busy
then issue ControlFieldSIOut = C,RNR,Pis1


CSPEC 1.2.2.4.7 : TimeOutErrorOnCountOut

*
Als het maximale aantal hertransmissies gedaan is na een timeout,
dan moet er een timeout fout gegeven worden.
*

if TimeOut and EQU
then issue TimeOutError
```



FD 1.2.4 : OutputSIFrame

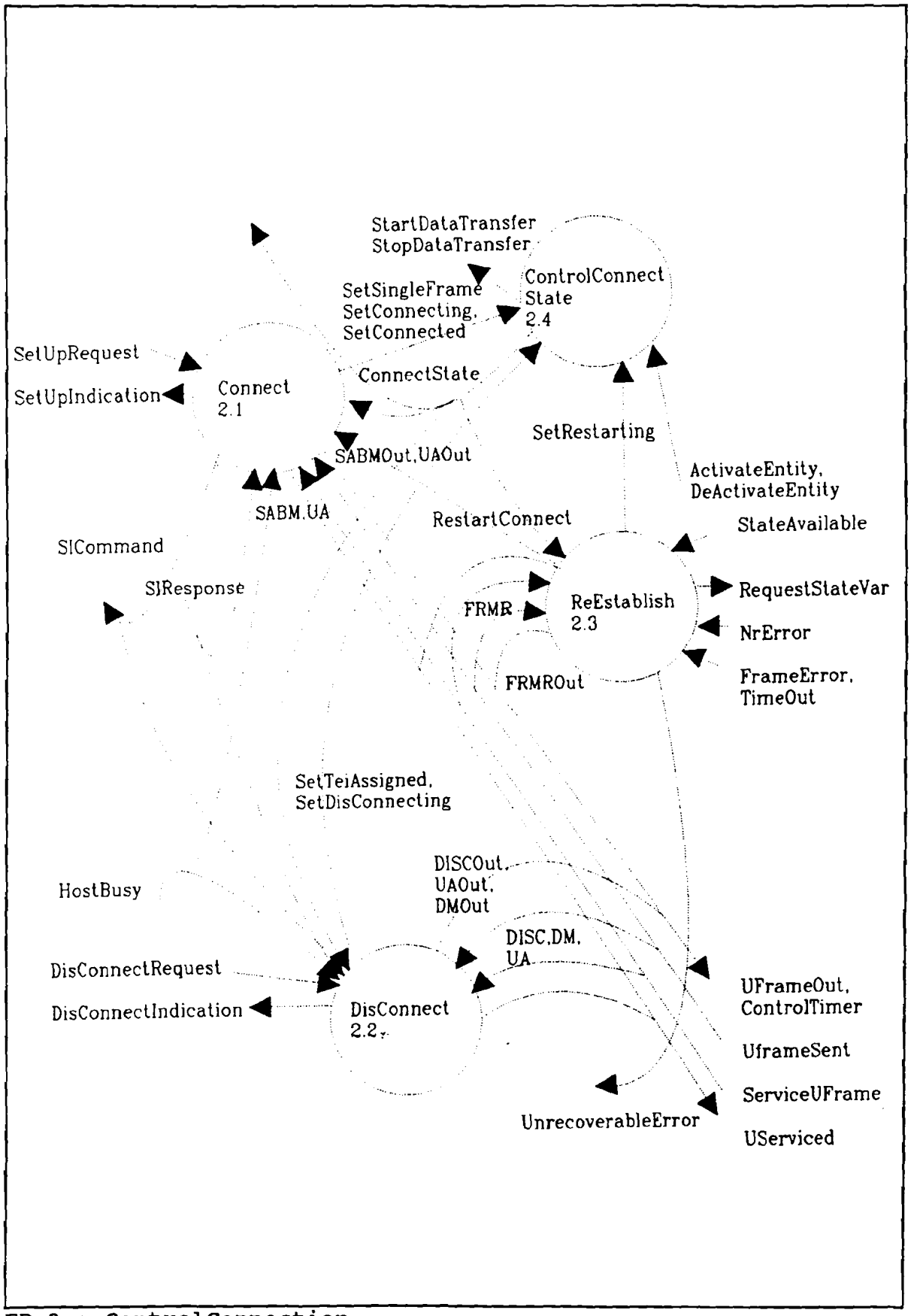
```
CSPEC 1.2.4.1 : ControlSendFrames

*
Zorg voor de handshake met de host en bestuur ook
BufferInNextFrameToSend.
*

if AcceptPacket and not SISent
then set BufferPacketIn = true
     set PacketAccepted = true
     set RequestPacketOut = true

if not AcceptPacket
then set BufferPacketIn = false

if not SISent and TBufferEmpty
then set RequestPacketOut = false
     set PacketAccepted = false


PSPEC 1.2.4.2 : AssembleControlFieldOut

*
Genereer een controlfield uit: NextFrameToSend, FrameExpected,
SetUpParameters en de besturing signalen ControlFieldOut.
*

set OutPutFrame = function off Controlfield + (FrameExpected) +
(NextFrameToSend)


PSPEC 1.2.4.3 : BufferInNextFrameToSend

*
Aangeboden bytes worden opgeslagen in het buffer met sequence
nummer NextFrameToSend. Als het frame geheel overgenomen is kan
het uitgelezen worden met OutPutFrame. Als een frame bevestigd
is veranderd LastAck en kan het buffer waarin dit frame staat
gewist worden.
*

if ByteStrobe and BufferPacketIn
then write byte to Buffer(NextFrameToSend)

if not BufferPacketIn and Buffer(NextFrameToSend) = empty
then issue TBufferEmpty = true

if LastAck
then for x=OldLastAck to LastAck
          clear Buffer(x)
     OldLastAck = LastAck

if NextByte and SISent and not BufferPacketIn
then OutPutFrame = next byte of Buffer(NextFrameToSend)
```



FD 1.3 : TransferDataUnit

```
PSPEC 1.3.1 : SendUIFrame

*
Geef de te verzenden bytes door.
*
if ByteStrobe and EnableSendUIFrame
then issue FrameOut = PacketToSend

if ByteStrobe and EnableSendUIFrameM
then issue FrameOut = PacketToSend + M


CSPEC 1.3.2 : ControlSendUIFrame

*
Zorg voor de handshake bij te verzenden pakketten en maak het
door geven van bytes mogelijk.
*

if AcceptUnit and not SISent
then set PacketAccepted = true
     set EnableSendUIFrame = true

if AcceptMDLUnit and not SISent
then set PacketAccepted = true
     set EnableSendUIMFrame = true
if SISent
then set PacketAccepted = false


CSPEC 1.3.3 : ControlUFrameToHost

*
Zorgt voor de besturing van het UISendProcess en de hiervoor
benodigde handshake.
*

if UIArrival
then set UnitArrival = true

if UIMArrival
then set MDLUnitArrival = true

if PacketRead
then set EnableUIFrame = true

if not PacketRead
then set EnableUIFrame = false

if not UIArrival
then set UnitArrival = true
     issue SIServiced = true

if not UIMArrival
then set MDLUnitArrival = true
     issue SIServiced = true
```

```
PSPEC 1.3.4 : UIFrameToHost

*
Meld de aankomst van een frame aan 1.3.3 en geef na enable de
bytes door.
*
if ServiceFrame = UI
then issue UIArrival = true

if ServiceFrame = UIM
then issue UIMArrival = true

if EnableUFrameToHost
then PacketReceived = ServiceFrame
```



FD 2 : ControlConnection

CSPEC 2.1 : Connect

*
Dit proces zorgt voor het opzetten van een verbinding als de controller in TEI assigned toestand is en een host vraagt om een link op te zetten.
*

CSPEC 2.2 : DisConnect

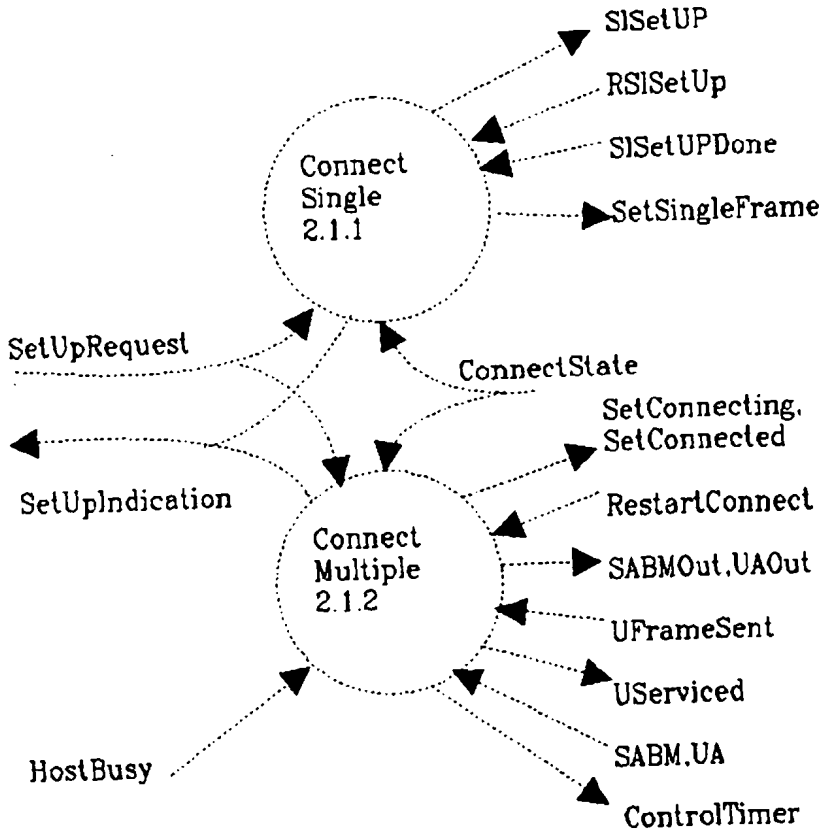
*
Dit proces verbreekt een verbinding als dit door de host of door de peer host gewenst wordt.
*

CSPEC 2.3 : ReEstablish

*
Bij het optreden van een fout zorgt dit proces voor het besturen van de fout afhandeling.
*

CSPEC 2.4 : ControlConnectState

*
Dit proces zorgt voor het op aanvraag veranderen van ConnectState.
*



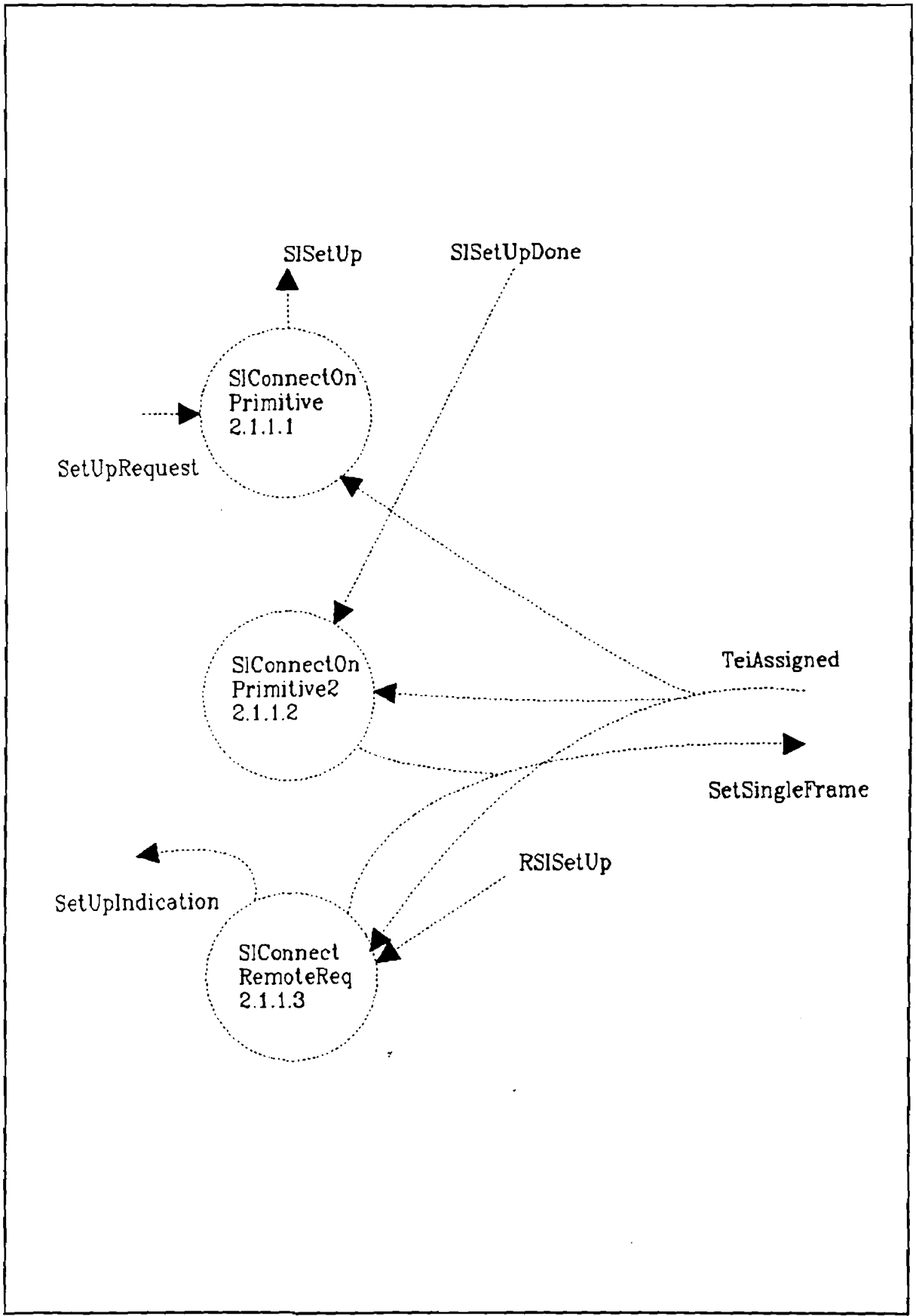
FD 2.1 : Connect

CSPEC 2.1.1 : ConnectSingle

*
Dit proces zorgt voor het opzetten van een single frame link.
*

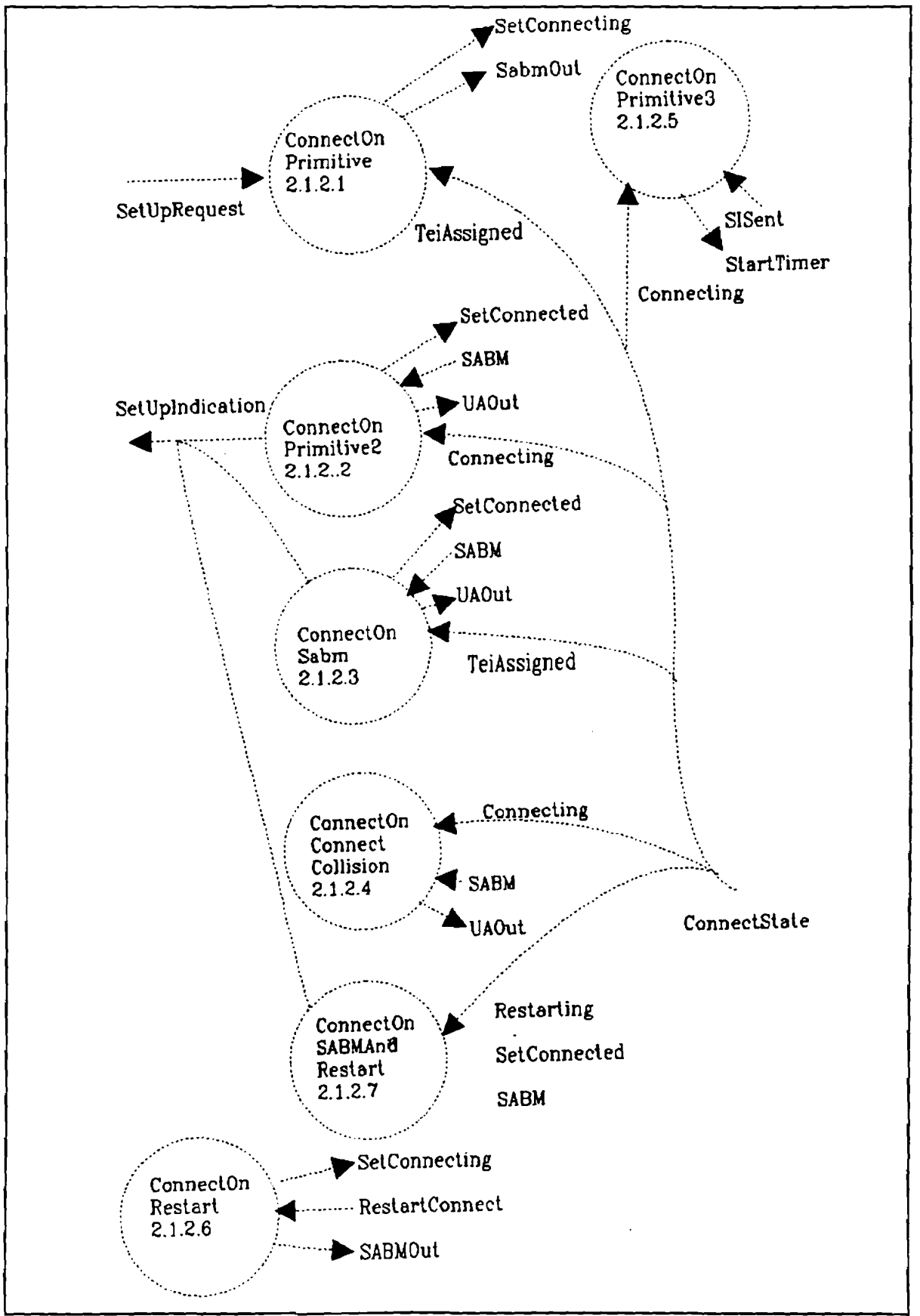
CSPEC 2.1.2 : ConnectMultiple

*
Dit proces zorgt voor het opzetten van een multiple frame link.
*



FD 2.1.1 : ConnectSingle

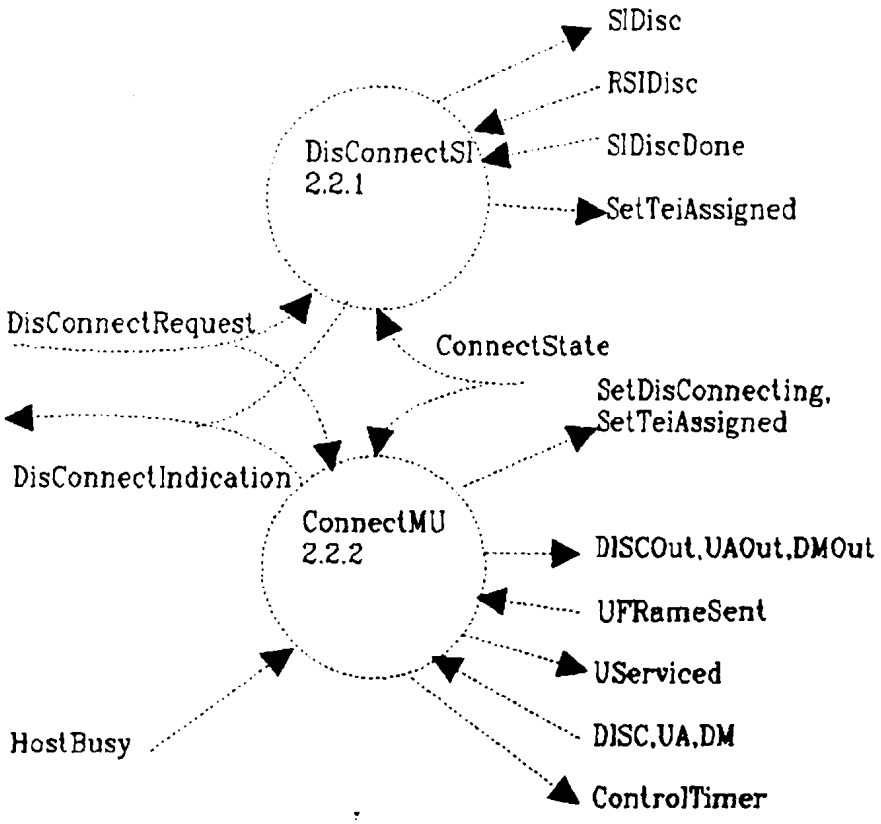
```
CSPEC 2.1.1.1 : SIConnectOnPrimitive

*
Een locale aanvraag om een single frame link op te zetten heeft
een aanvraag aan TransferData met SISetUp tot gevolg.
*

if SetUpRequest = Single
then set SISetUP = true

if not TeiAssigned
then set SISetUp = false


CSPEC 2.1.1.2 : SIConnectOnPrimitive2

*
Bij bevestiging door TransferData kan ConnectState veranderd
worden.
*

if TeiAssigned and SISetUpDone
then issue SetSingleFrame = true


CSPEC 2.1.1.3 : SIConnectRemoteRequest

*
TransferData geeft aan dat de peer controller een aanvraag voor
een single frame link gedaan heeft.
*

if TeiAssigned and RSISetUp
then issue SetSingleFrame = true
     issue SetUpIndication = true
```



FD 2.1.2 : ConnectMultiple

```
CSPEC 2.1.2.1 : ConnectOnPrimitive

*
Een locale aanvraag voor een multiple frame link moet verwerkt
worden.
*

if TeiAssigned and SetUpRequest = multiple
then issue SetConnecting = true
     issue SabmOut = true
     issue StartTimer

CSPEC 2.1.2.2 : ConnectOnPrimitive2

*
Als een UA frame ontvangen is kan de link gebruikt worden.
ConnectState wordt Connecting.
*

if Connecting and UA
then issue SetConnecting
     issue StopTimer

CSPEC 2.1.2.3 : ConnectOnSABM

*
Een frame met een aanvraag om een multiple frame link op te
zetten is aangekomen.
*

if TeiAssigned and SABM (and Available)
then issue SetUpIndication = multi
     issue StopTimer

CSPEC 2.1.2.4 : ConnectOnConnectCollision

*
Als een aanvraag om een link op te zetten aankomt terwijl reeds
een poging gedaan wordt om een link op te zetten, dan zal dit
resulteren in een normale afhandeling van de aan vraag.
*

if Connecting and SABM
then issue UAOut


CSPEC 2.1.2.5 : ConnectOnPrimitive3

*
Als een frame verzonden is wordt de timer gestart.
*

if SISent and Connecting
then issue StartTimer

CSPEC 2.1.2.6 : ConnectOnRestart
```

```
*
Een RestartConnect brengt de controller in Connecting toestand.
*

if RestartConnect
then issue SetConnecting
     issue SetUpIndication

CSPEC 2.1.2.7 : ConnectOnSABMandRestart

*
Als de toestand Restarting is en er komt een SABM frame aan, dan
zal dit resulteren in het realiseren van de link en het melden
hiervan aan de host.
*

if Restarting and SABM
then issue SetConnected
     issue ConnectIndication = multi
```



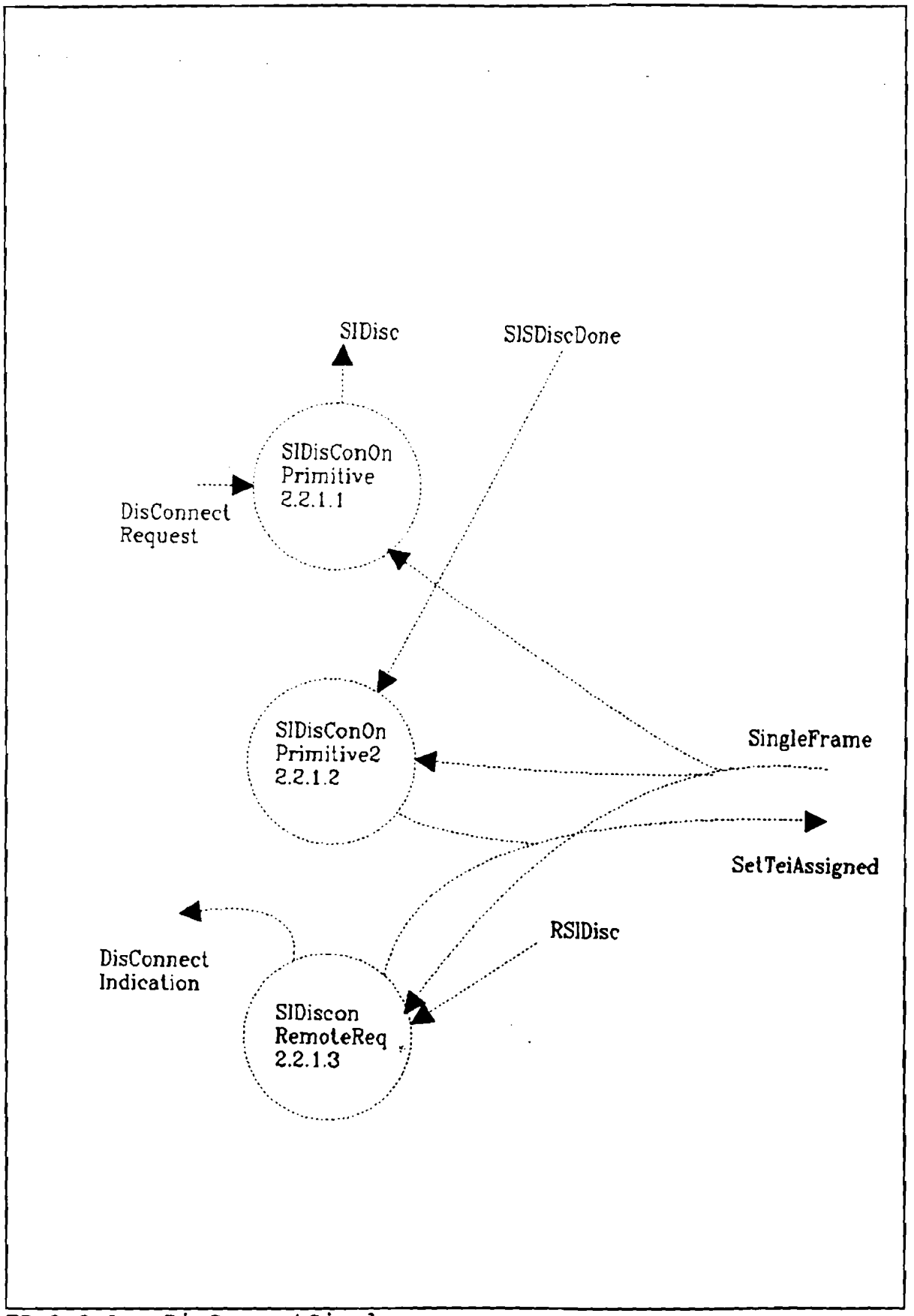
FD 2.2 : DisConnect

CSPEC 2.2.1 : DisConnectSingle

*
Dit proces zorgt voor het afbreken van een single frame link.
*

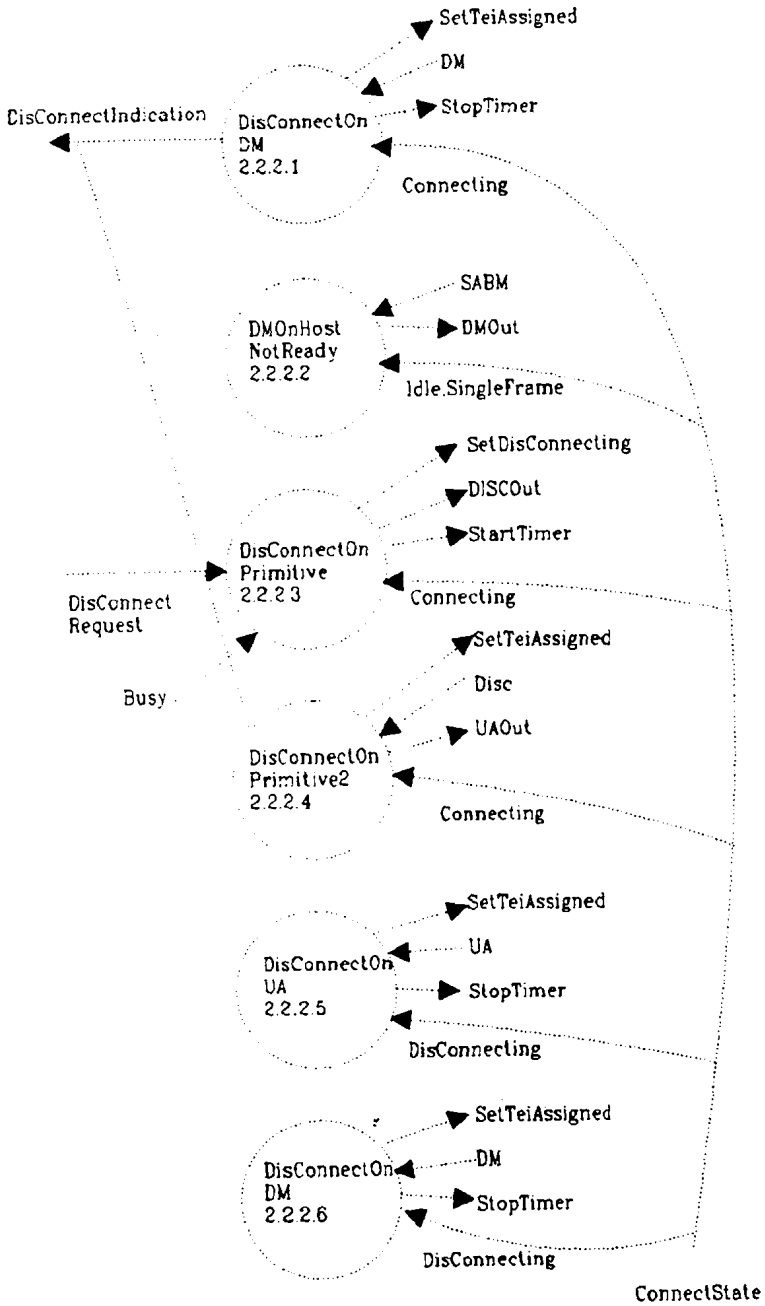
CSPEC 2.2.2 : DisConnectMultiple

*
Dit proces zorgt voor het afbreken van een multiple frame link.
*

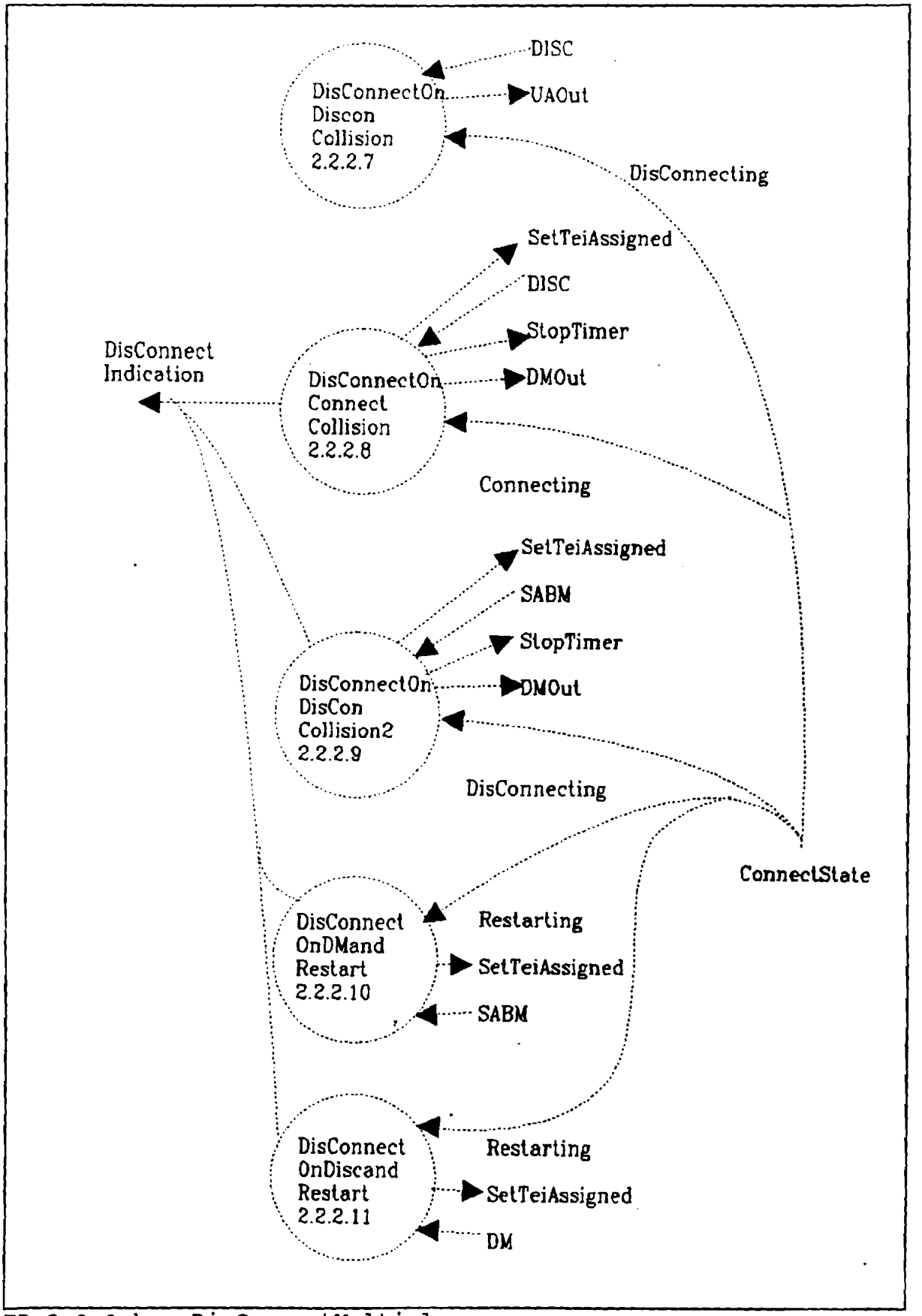


FD 2.2.1 : DisConnectSingle

```
CSPEC 2.2.1.1 : SIDisConOnPrimitive

*
Verbreek een single frame link als dit met een primitieve
gevraagd wordt.
*

if DisConnectRequest = Single
then set SIDisc = true

if not SingleFrame
then set SIDisc = false


CSPEC 2.2.1.2 : SIDisConOnPrimitive2

*
Verander ConnectState als DataTransfer meldt dat de verbinding
verbroken is.
*

if SingleFrame and SIDiscDone
then issue SetTeiAssigned = true


CSPEC 2.2.1.3 : SIDisConRemoteRequest

*
Verander ConnectState als de remote controller de verbinding wil
verbreken.
*

if SingleFrame and RSIDisc
then issue SetTeiAssigned = true
     issue DisConnectIndication = true
```



FD 2.2.2.a : DisConnectMultiple

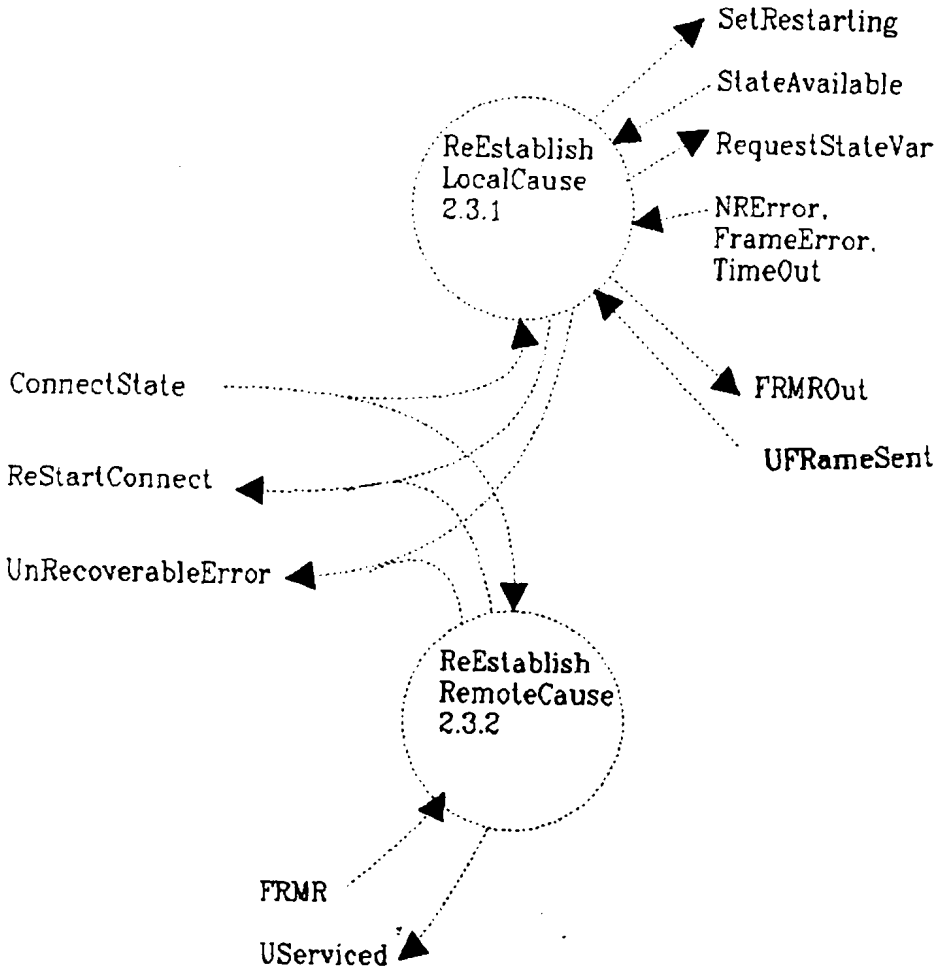


FD 2.2.2.b : DisConnectMultiple

```
CSPEC 2.2.2.1 : DisConnectOnDM

*
Aankomst van een DM frame heeft tot gevolg dat de timer gestopt
wordt en een indication aan de host gegeven wordt.
*

if Connecting and DM
then issue StopTimer = true
     issue DisConnectIndication = multi


CSPEC 2.2.2.2 : DMOnHostNotReady

*
Een SABM in de toestand busy, Idle of SingleFrame heeft het
verzenden van een DM frame tot gevolg.
*

if (busy or Idle or SingleFrame) and SABM
then issue DM


CSPEC 2.2.2.3 : DisConnectOnPrimitive

*
Een host request in on Connected toestand heet het verzenden van
een DISC frame tot gevolg.
*

if Connected and DisConnectRequest = multi
then issue DISCOut
     issue StartTimer
     issue SetDisConnecting


CSPEC 2.2.2.4 : DisConnectOnPrimitive2

*
Een aankomend DISC frame in Connected toestand resulteert in een
toestand verandering naar Disconnecting en het verzenden van een
UAFrame.
*

if Connecting and DISC
then issue StopTimer
     issue DisConnectIndication = multi
     issue UAOut


CSPEC 2.2.2.5 : DisConnectOnUA

*
Aankomst van een UA frame in Disconnecting toestand resulteert
in de toestand TEIAssigned.
*
```

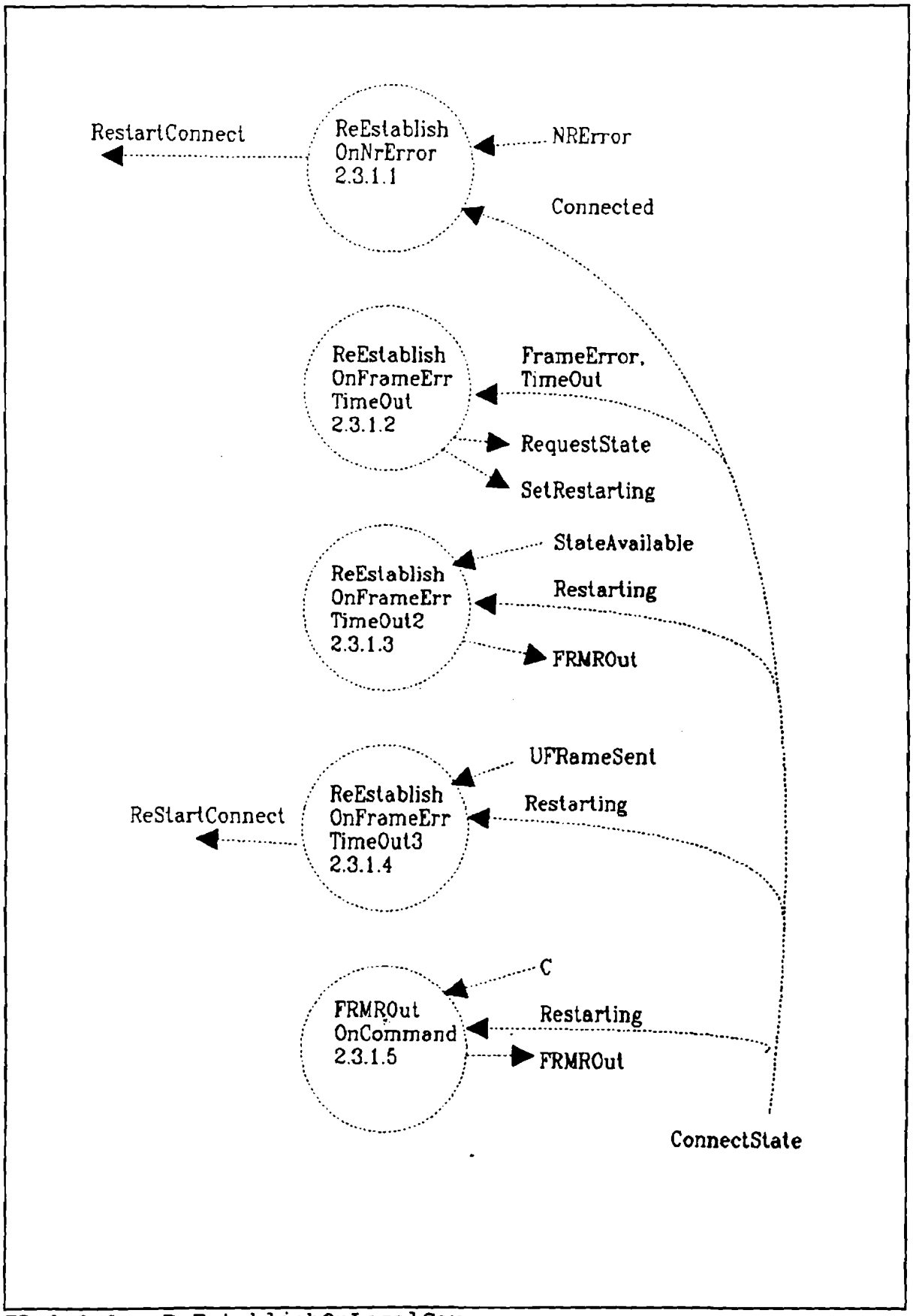
```
if DisConnecting and UA
then issue SetTeiAssigned
     issue StopTimer


CSPEC 2.2.2.6 : DisConnectOnDM

*
Disconnecting en de aankomst van een DM frame resulteert in de
toestand TEIAssigned.
*

if DisConnecting and DM
then issue SetTeiAssigned
     issue StopTimer


CSPEC 2.2.2.7 : DisConnectOnDiscCollision

*
Disconnecting en DISC resulteren in het verzenden van een UA
frame.
*

if DisConnecting and DISC
then issue UAOut


CSPEC 2.2.2.8 : DisConnectOnConnectCollision

*
DISC in Connecting toestand resulteert in de toestand TEIAssigned
en een DM frame wordt naar de peer gestuurd.
*

if Connecting and DISC
then issue StopTimer
     issue SetTeiAssigned
     issue DMOut

CSPEC 2.2.2.9 : DisConnectOnDiscCollision2

*
DisConnecting en aankomst van een SABM resulteert in TEIAssigned
toestand en een DM frame wordt naar de peer verstuurd.
*

if DisConnecting and SABM
then issue SetteiAssigned
     issue StopTimer
     issue DMOut

CSPEC 2.2.2.10 : DisConnectOnDMandRestart

*
```

```
Aankomst van een DM frame in Restarting toestand resulteert in
:TEIAssigned toestand en een indication aan de host.
*

if Restarting and DM
then issue SetTeiAssigned
     issue DisConnectIndication = multi

CSPEC 2.2.2.11 : DisConnectOnDiscandRestart

*
De toestand Restarting en de aankomst van een DISC frame
resulteert in: toestand TEIAssigned en een Disconnect indication
naar de host.
*

if Restarting and DISC
then issue SetTeiAssigned
     issue DisConnectIndication = multi
```



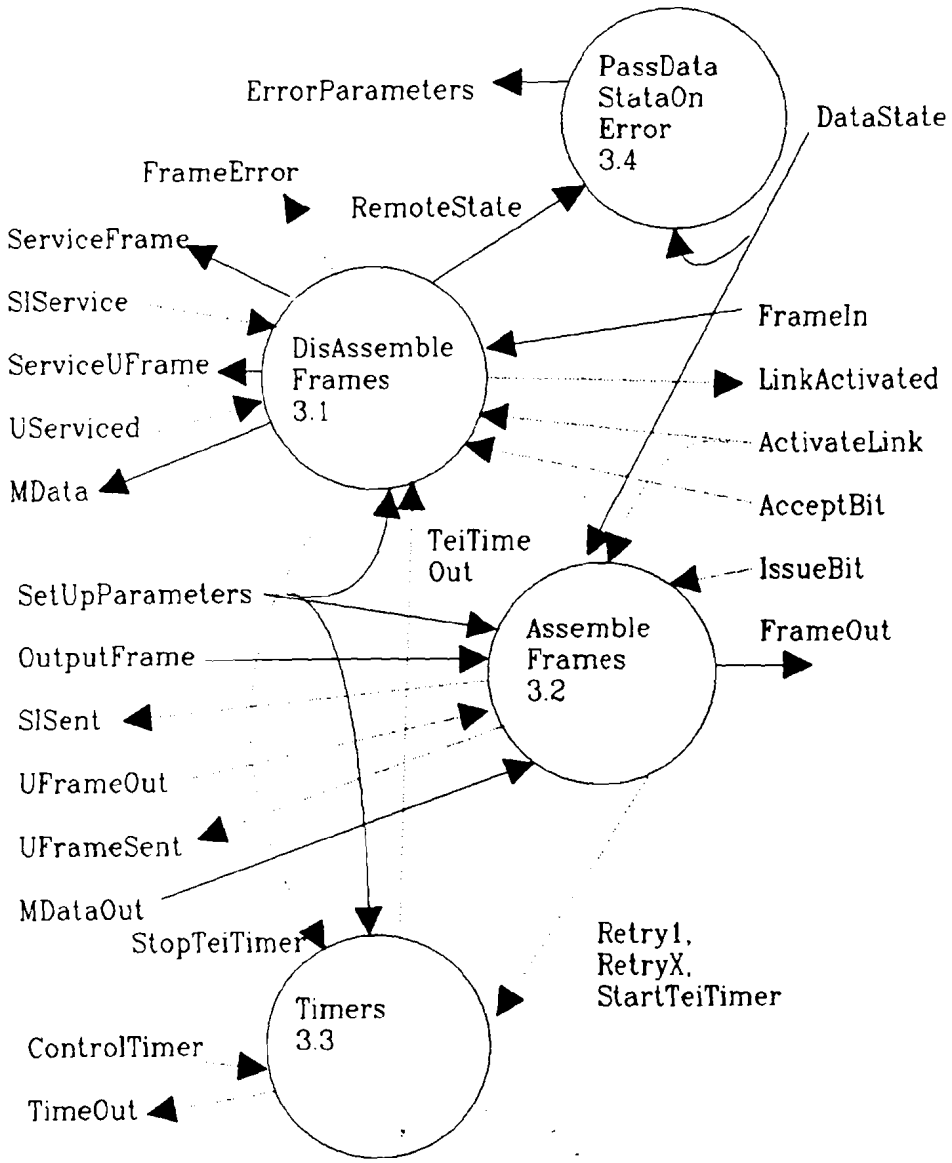
FD 2.3 : ReEstablish

```
CSPEC 2.3.1 : ReEstablishLocalCause

*
Bestuur de afhandeling van een locale fout.
*


CSPEC 2.3.2 : ReEstablishRemoteCause

*
Afhandelen van een fout die door de peer gemeld wordt.
*

if FRMR
then issue RestartConnect
     issue SIServiced
     issue UnRecoverableError
```



FD 2.3.1 : ReEstablishOnLocalCause

```
CSPEC 2.3.1.1 : ReEstablishOnError

*
Verander de toestand naar Restarting als een fout gemeld wordt.
*

if NRError and Connected
then issue RestartConnect = true

CSPEC 2.3.1.2 : ReEstablishOnFrameErrTimeOut

*
Verander de toestand naar Restarting als een FrameError gemeld
wordt.
*

if FrameError or TimeOut
then issue RequestStateVar = true
     issue SetRestarting

CSPEC 2.3.1.3 : ReEstablishOnFrameErrTimeOut2

*
Wacht tot het frame verzonden is.
*

if SISent and Restarting
then issue RestartConnect

CSPEC 2.3.1.4 : ReEstablishOnFrameErrTimeOut3

*
Verzend een frame met fout informatie.
*

if StateAvailable and Restarting
then issue FRMROut

CSPEC 2.3.1.5 : FRMROutOnCommand

*
Als een command frame ontvangen wordt dan wordt het FRMR frame
herhaald.
*

if Restarting and Command and not (SABM or DISC)
then issue FRMROut
```



FD 3 : AssembleDisAssembleFrames

PSPEC 3.1 : DisAssembleFrames

*
Aan de bitstroom die via FrameIn wordt ontvangen worden de verschillende onderdelen waaruit een frame is samengesteld onttrokken. Deze onderdelen zijn: addressfield, controlfield en eventueel datafield. Verificatie van het adres deel zal eerst plaatsvinden. Als het adres klopt wordt het frame verder gevalueerd. Of een frame een command of een response frame is kan ook uit het adres deel worden afgeleid. Indien er geen fouten gedetecteerd worden bij de verdere evaluatie van de binnenkomende bits dan zullen die door gegeven worden. Met ServiceFrame gebeurt dit als het een S, I, SIx, of een UI data frame betreft. UI frames voor Tei verificatie worden door gegeven als MData. De andere U frames worden door gegeven met ServiceUFrame. Gedetecteerde fouten worden door FrameError gesignaliseerd. Bij het ontvangen van frames worden de volgende functies uitgevoerd: aannemen van bits, destuffing van bits, opdelen frame in delen, CRC test, adres test en bufferen van de ontvangen bytes.
*

PSPEC 3.2 : AssembleFrames

*
De informatie door de andere processen aangeleverd wordt door dit proces omgezet in de bitstroom FrameOut. OutputFrame levert de data frames aan die verzonden moeten worden. Als dit gebeurt is wordt dit bevestigd met SIServiced. MData is de flow die de informatie voor de Tei vaststelling aanlevert. De informatie voor het verbreken en opzetten van de link zit in UFrameOut. Als dit gebeurt is wordt dit met UFrameSend bevestigd. Het proces geeft aan hoe vaak een frame herzonden moet worden voor TimeOut. Herzenden wordt ingezet door ReSend.
*

PSPEC 3.3 : Timers

*
Verzorg de timers en houdt de hertransmissie teller bij.
*

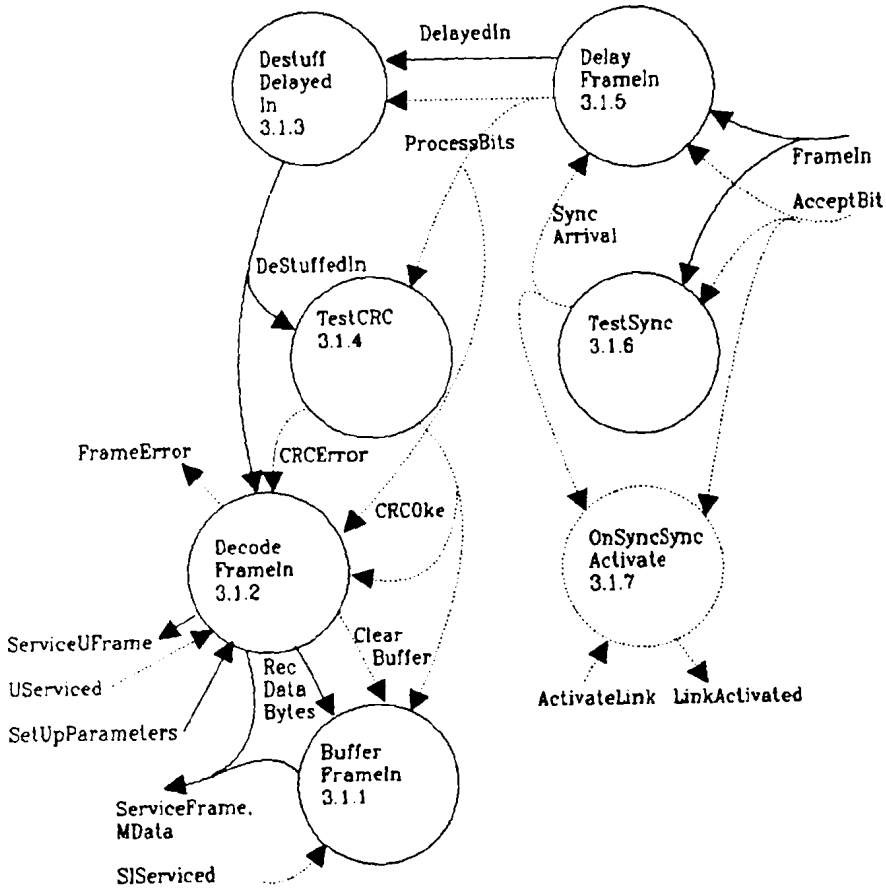
PSPEC 3.4 : PassDataStateOnError

*
Geef bij een lokale fout de toestand van het data transfer process door. Geef de toestand van het datatransfer proces door bij een fout opgemerkt door de peer controller.
*

if DataState then
set ErrorParameters = DataState

if RemoteError

```
then set ErrorParameters = RemoteError
```



FD 3.1 : DisAssembleFrames

PSPEC 3.1.1 : BufferFrameIn

*
Dit proces beheert twee buffers hiervan kan het ene gebruikt worden om data uit RecDataBytes op te slaan terwijl uit het andere via ServiceFrame of Mdata data uit te lezen is. Als met CRCOke aangeduid wordt dat een buffer vol is kan het uit gelezen worden. Met SiServiced wordt aangeduid dat de data gelezen is. ClearBuffer betekent dat de aangeboden data incorrect was en dus niet geldig is.
*

PSPEC 3.1.2 : DecodeFrameIn

*
Dit proces krijgt in met de flow DeStuffedIn de bits binnen die de informatie vormen uit het aangekomen frame. Deze informatie moet uit de bitstroom gehaald worden. Het betreft hier: Sapi, Tei, Controlfield en eventueel Data. Hiervoor vindt eerst de omzetting plaats van bitstroom naar bytes. Als het adres dat als eerste arriveert klopt zal de rest van het frame ook gevalueerd worden. Het ControlField en de eventuele data worden afgesplitst. Data gaat naar BufferFrameIn (DataBytesIn). Control informatie wordt intern bewaard. Als CRCOke is zal Controlfield door gegeven worden. Voor I, SI, RNR, RR, REJ en UI frames met management data zal dit met ServiceFrame gebeuren. De UI frames die gebruikt worden voor de Tei bepaling worden met MData door gegeven. CRCError heeft tot gevolg dat het buffer weer gewist moet worden. SIServiced geeft aan dat ClearBufferRec gegeven moet worden. UServiced geeft aan dat het een unnumbered frame door ControlConnection verwerkt is.
*

PSPEC 3.1.3 : DestuffDelayedIn

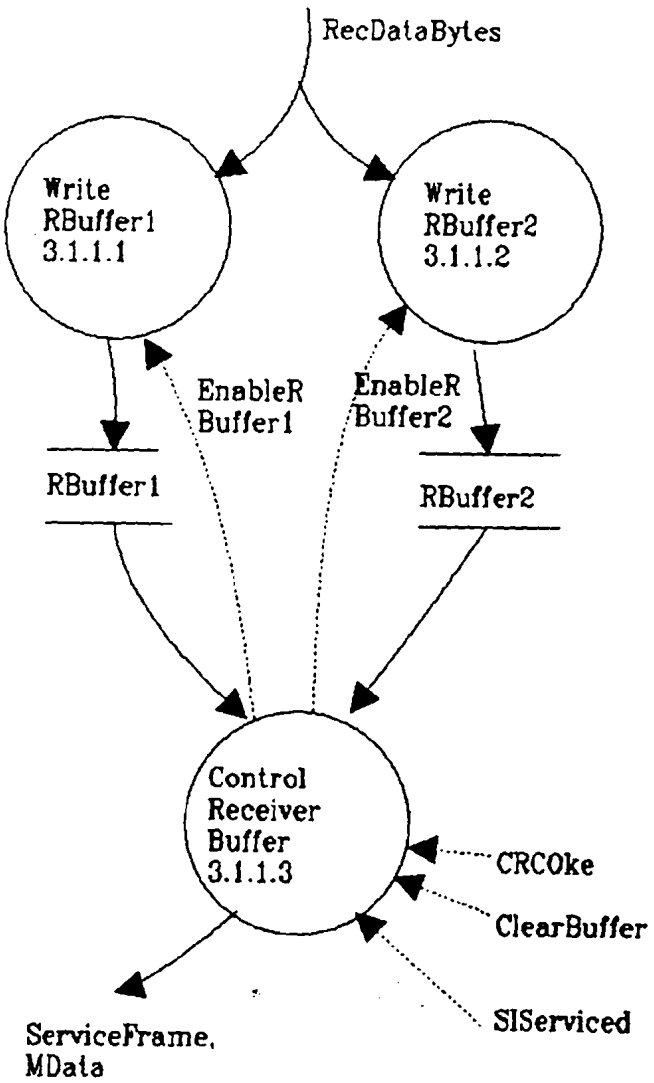
*
Geef de bits van DelayedIn door aan DestuffedIn tenzij het de 0 is in de string 111110, deze 0 wordt niet doorgegeven.
*

```
if DelayedIn = 1 and ProcessBits
then OneCount = OneCount +1
     issue DestuffedIn = DelayedIn

if DelayedIn = 0 and ProcessBits and OneCount <> 6
then OneCount = 0
     issue DestuffedIn = DelayedIn

if not ProcessBits
then OneCount = 0
```

PSPEC 3.1.4 : TestCRC

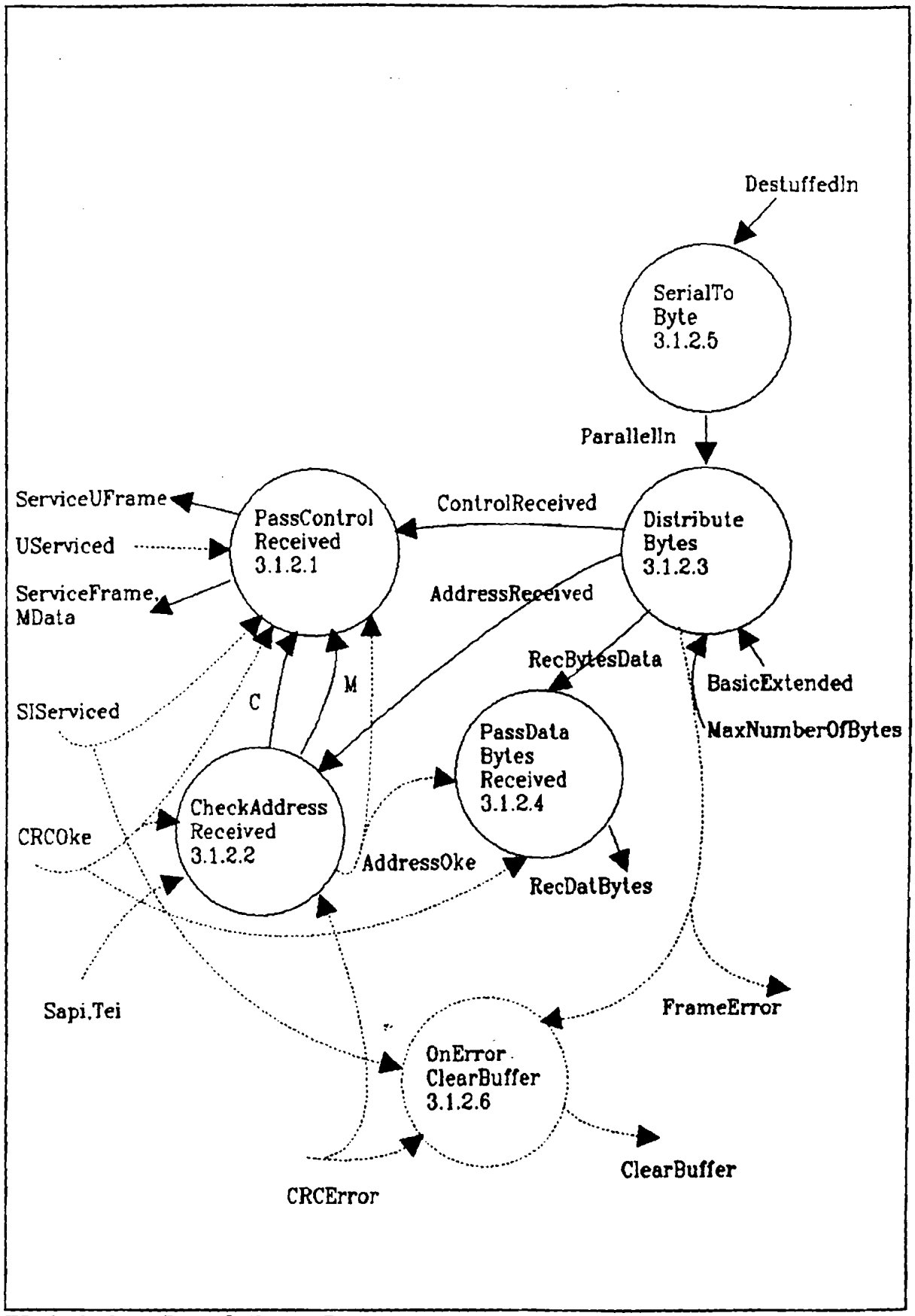
```
*
Als ProcessBits = true berekent dit proces de CRC-16 van
DestuffedIn. Als ProcessBits = false wordt aangegeven of de CRC
nul was of niet, dit gebeurt met respectievelijk CRCOke en
CRCError.
*

if ProcessBits
then "calculate the CRC of DestuffedIn"

if not ProcessBits and CRC = 0
then issue CRCOke = true

if not ProcessBits and CRC <> 0
then issue CRCError = true


PSPEC 3.1.5 : DelayFrameIn

*
Als een sync gedetecteerd is moet dit process de bits met acht
bits vertraging doorgeven via DelayedIn.
*

if SyncArrival
then Wacht := 1
     set ProcessBits = false

if Wacht = 8
then DelayedIn = EightBitDelayed(FrameIn)
     set ProcessBits = true


PSPEC 3.1.6 : TestSync

*
Test of de bitstring 01111110 in FrameIn voorkomt, indien ja dan
maak SyncArrival = true
*

if FrameIn = 0 and AcceptBit and BitTeller <> 8
then BitTeller = 1

if FrameIn = 1 and AcceptBit
then BitTeller = BitTeller +1

if FrameIn = 0 and AcceptBit and BitTeller = 8
then issue SyncArrival = true


PSPEC 3.1.7 : OnSyncSyncActivate

*
Als tijdens het wachten op een active link twee op elkaar
volgende syncs ontvangen worden moet dit met LinkActivated gemeld
worden.
```

```
*

if SyncArrival and ActivateLink and BitTeller2 = 0
then issue LinkActivated = true

if SyncArrival and ActivateLink and BitTeller2 <> 0
then BitTeller2 = 0

if AcceptBit and ActivateLink
then BitTeller2 = BitTeller2 +1
```



FD 3.1.1 : BufferFrameIn

```
PSPEC 3.1.1.1 : WriteRBuffer1

*
Schrijf in buffer 1.
*

if EnableRBuffer1
then issue RBuffer1 = RecDataBytes


PSPEC 3.1.1.2 : WriteRBuffer2

*
Schrijf in buffer 2.
*

if EnableRBuffer2
then issue RBuffer2 = RecDataBytes


PSPEC 3.1.1.3 : ControlReceiverBuffers

*
Wissel de buffers als beschreven in PSPEC 3.1.1
*

if RBuffer1 = free
then set EnableBuffer1 = true

if RBuffer2 = free and not RBuffer1 = free
then set EnableRBuffer2 = true

if CRCOke and RBuffer1 = free and not ClearBuffer
then RBuffer1 = full
     set EnableRBuffer1 = false

if SIServiced and RBuffer1 = full
then RBuffer1 = free

if CRCOke and RBuffer2 = free and not ClearBuffer
then RBuffer2 = full
     set EnableRBuffer2 = false

if SIServiced and RBuffer2 = full
then RBuffer2 = free
```



FD 3.1.2 : DecodeFrameIn

PSPEC 3.1.2.1 : PassControlReceived

*
Bij het ontvangen van CRCOke kan de informatie uit het control veld van het frame door gegeven worden. Voor Unnumbered frames met uitzondering van UI en SI frames gebeurt dit via ServiceUFrame. Voor I, S, UI, SI gebeurt dit met ServiceFrame mits de UI frames niet voor Tei onderhandeling gebruikt worden.
*

PSPEC 3.1.2.2 : CheckAddressReceived

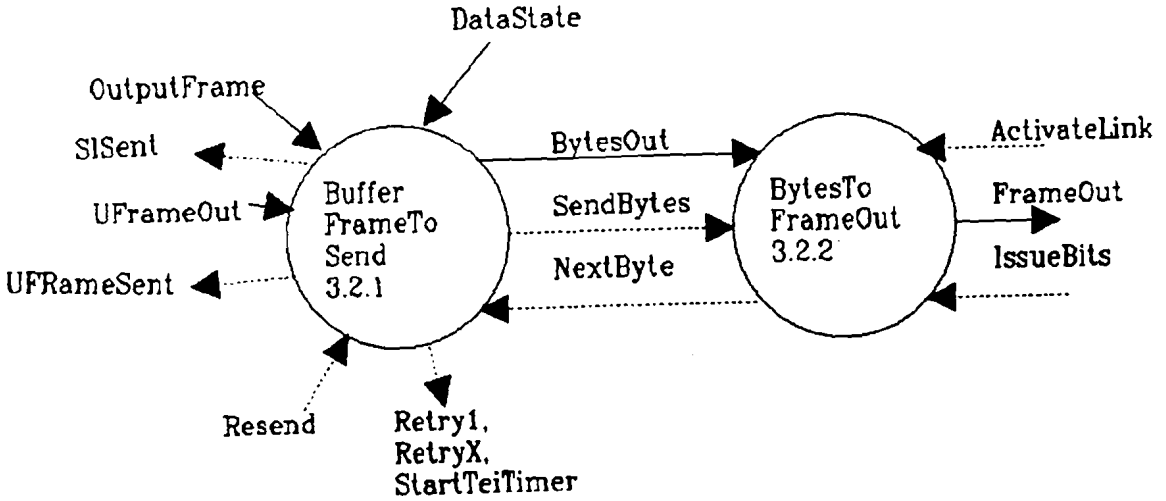
*
De twee bytes die het adres bevatten worden geanalyseerd. Als het adres juist is wordt dit met AddressOke = true aangegeven. M geeft aan dat een frame met management adres ontvangen is. C geeft aan dat we te maken hebben met een command frame.
*

```
if AddressReceived = Sapi,Tei
then set AddressOke = true
     issue C = bit 1 of AddressReceived
     issue M = false

if AddressReceived = 63,127
then set AddressOke = true
     issue M = true
     issue C = bit 1 of AddressReceived

if AddressReceived = Sapi,Tei
then set AddressOke = true
     issue M = false
     issue C = bit 1 of AddressReceived

if CRCOke
then set AddressOke = false

if CRCError
then set AddressOke = false
```

PSPEC 3.1.2.3 : DistributeBytes

*
De eerste twee bytes die ontvangen worden na CRCOke of CRCError worden doorgegeven aan CheckAddressReceived. Het hierna volgende byte wordt doorgegeven aan PassControlReceived. Als het frame formaat extended is zal indien dit laatste byte een S of I control frame was het volgende byte ook aan PassControlReceived worden doorgegeven. Dit byte zal aan PassDataBytesRec doorgegeven worden als het frame formaat Basic is of als het een U Frame betreft. Alle volgende bytes gaan ook naar PassDataBytesRec. Als het frame afwijkt van het verwachte, dan zal een FrameError optreden. Dit gebeurt als een S of U frame met data ontvangen wordt of indien ParallelIn MaxNumberOfBytes + 3/4 bytes of meer

```
levert.
*

if Extended
then ByteLimit = MaxNumberOfBytes +4
     HeaderLimit = 4

if Basic
then ByteLimit = MaxNumberOfBytes +3
     HeaderLimit = 3

if ProcessBit = false
then ByteCount = 1

if ByteCount >= ByteLimit
then issue FrameError = true

if ParallelIn and ByteCount < 2 and ProcessBits
then issue AddressReceived = ParallelIn
     ByteCount = ByteCount +1

if ParallelIn and 2 <= ByteCount <= HeaderLimit and ProcessBits
then issue ControlReceived = ParallelIn
     ByteCount = ByteCount +1

if ParallelIn and ByteCount > HeaderLimit and ProcessBits
then issue RecBytesData = ParallelIn
     ByteCount = ByteCount +1

PSPEC 3.1.2.4 : PassDataBytesRec

*
Als het adres oke is kunnen de bytes RecBytesData door gegeven
worden aan RecDataBytes
*

if AddressOke
then RecDataBytes = RecBytesData


PSPEC 3.1.2.5 : SerialToByte

*
De bitstroom DestuffedIn wordt omgevormd in de byte stroom
ParallelIn.
*
```



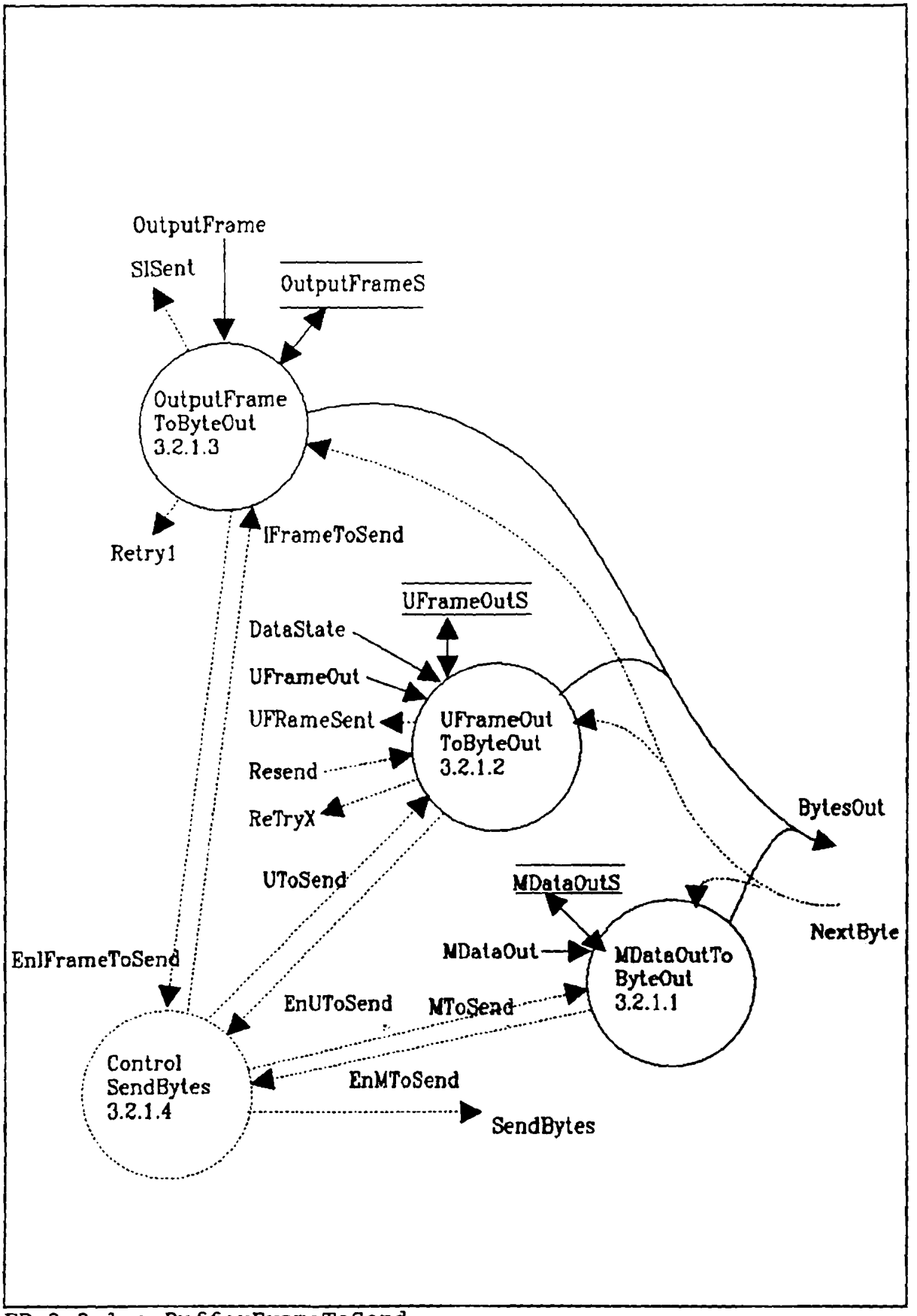
FD 3.2 : AssembleFrames

PSPEC 3.2.1 : BufferFrameToSend

*
Buffer het frame dat verzonden moet worden.
*

PSPEC 3.2.2 : BytesToFrameOut

*
Als geen data bytes verzonden hoeven worden zal dit proces een continue reeks van sync karakters op FrameOut verzenden. Als er wel bytes te verzenden zijn dan wordt dit door SendBytes gesignaleerd. Na het beeindigen van het momentele sync karakter worden de bytes verzonden die met BytesOut aangeboden worden. NextByte vraagt nieuwe bytes aan zolang SendBytes aangeeft dat bytes verzonden moeten worden.
*

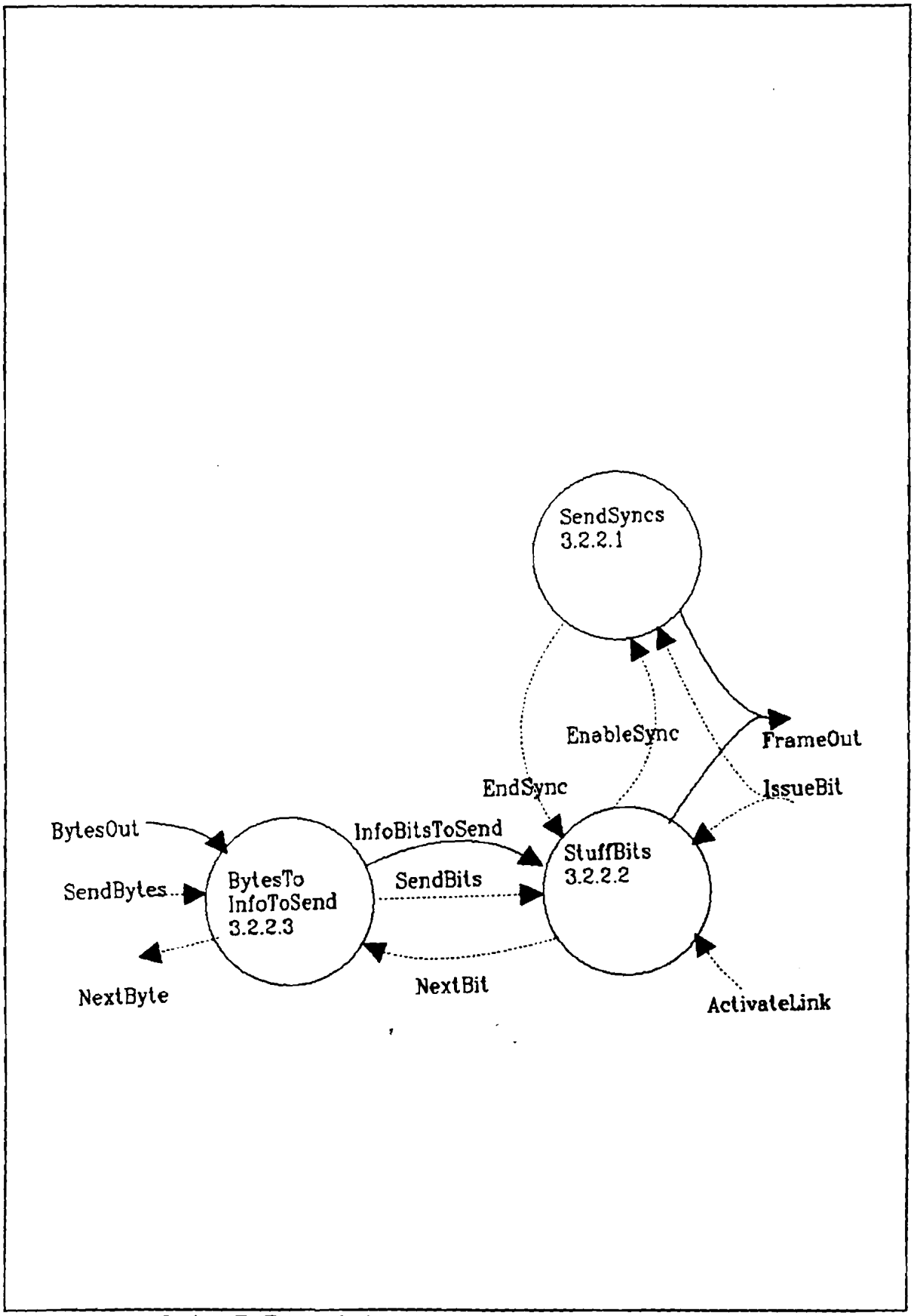


FD 3.2.1 : BufferFrameToSend

```
PSPEC 3.2.1.1 : MDataOutToByteOut

*
De zes bytes die voor de Tei vaststelling moet worden verzonden
worden opgeslagen in MDataOutS. Als EnMToSend het aangeeft kunnen
de bytes voorafgegaan door twee bytes met het management adres
op aanvraag van NextByte doorgegeven worden met BytesOut.
*

if MDataOut
then set MDataToSend = true
     MDataByte = 0
     issue MDataOuts = MDataOut

if EnMToSend and NextByte and not MDataByte = 8
then set ByteOut = byte MDataByte of MDataOutS
     MDataByte = MDataByte +1

if MDataByte = 8
then set MDataToSend = false
     issue StartTeiTimer = true


PSPEC 3.2.1.2 : UFrameOutToByteOut

*
U frames worden aangeboden aan BytesOut. Als het een FRMR frame
betreft wordt ook de informatie uit DataState door gegeven. Het
doorgeven gebeurt als EnUToSend en NextByte hierom vragen. Als
het frame verzonden is wordt dit met UFrameSend en UToSend
kenbaar gemaakt. Command frames moeten als ze niet tijdig
bevestigd zijn herzonden worden. Met RetryX wordt dit door
gegeven. Met Resend wordt aangegeven dat een frame herzonden moet
worden.
*

if UFrame
then set UFrameToSend = true
     UFrameOutS = UFrameOut
     UFToSend = 1
     ULimit = 1
     set RetryX = false

if UFrameOut = FRMROut
then set UFrameToSend = true
     UFrameOutS = UFrameOut + DataState
     UFToSend = 1
     ULimit = 5
     set RetryX = false

if UFToSend = ULimit
then set UToSend = false
     issue UFrameSend = true

if NextByte and EnUToSend
then set BytesOut = byte UToSend of UFrameOutS
```

```
     UFToSend = UFToSend = 1

if Resend
then set UFToSend = true
     UFToSEnd =1

if BytesOut = SabmOut
then set RetryX = true

if BytesOut = DiscOut
then set RetryX = true


PSPEC 3.2.1.3 : OutPutFrameToByteOut

*
De informatie in de OutputFrame flow wordt opgeslagen in
OutPutFrameS hier wordt de informatie vooraf gegaan door
AddressOut. Als het proces met EnIToSend op de hoogte gesteld
wordt dat bytes gezonden kunnen worden, dan gebeurt dit na
ontvangst van NextByte. Retry 1 geeft aan dat een TimeOut na de
eerste keer zenden moet worden gegeven. Als alle bytes verzonden
zijn wordt dit met SISent en IToSend aan gegeven.
*

if OutputFrame and M
then AddressOut = 63,127
     set Retry1 = true

if OutPutFrame
then OutputFrameS = OutputFrame
     AddressOut = Tei,Sapi
     set Retry1 = true

if EnIToSend and NextByte and not PacketEnd
then set ByteOut = next byte of OutputFrameS

if PacketEnd
then set IFrameToSend = false

if PacketEnd
then issue SISent = true
     set IFrameToSEnd = false


CSPEC 3.2.1.4 : ControlSendByte

*
Als een proces bytes wil zenden wordt dit aan gevraagd met
xxToSend. Als dit mogelijk is wordt dit gemeld door zowel
EnxxToSend als SendBytes.
*

if IFrameToSend and NextByte
then set SendBytes = true
```

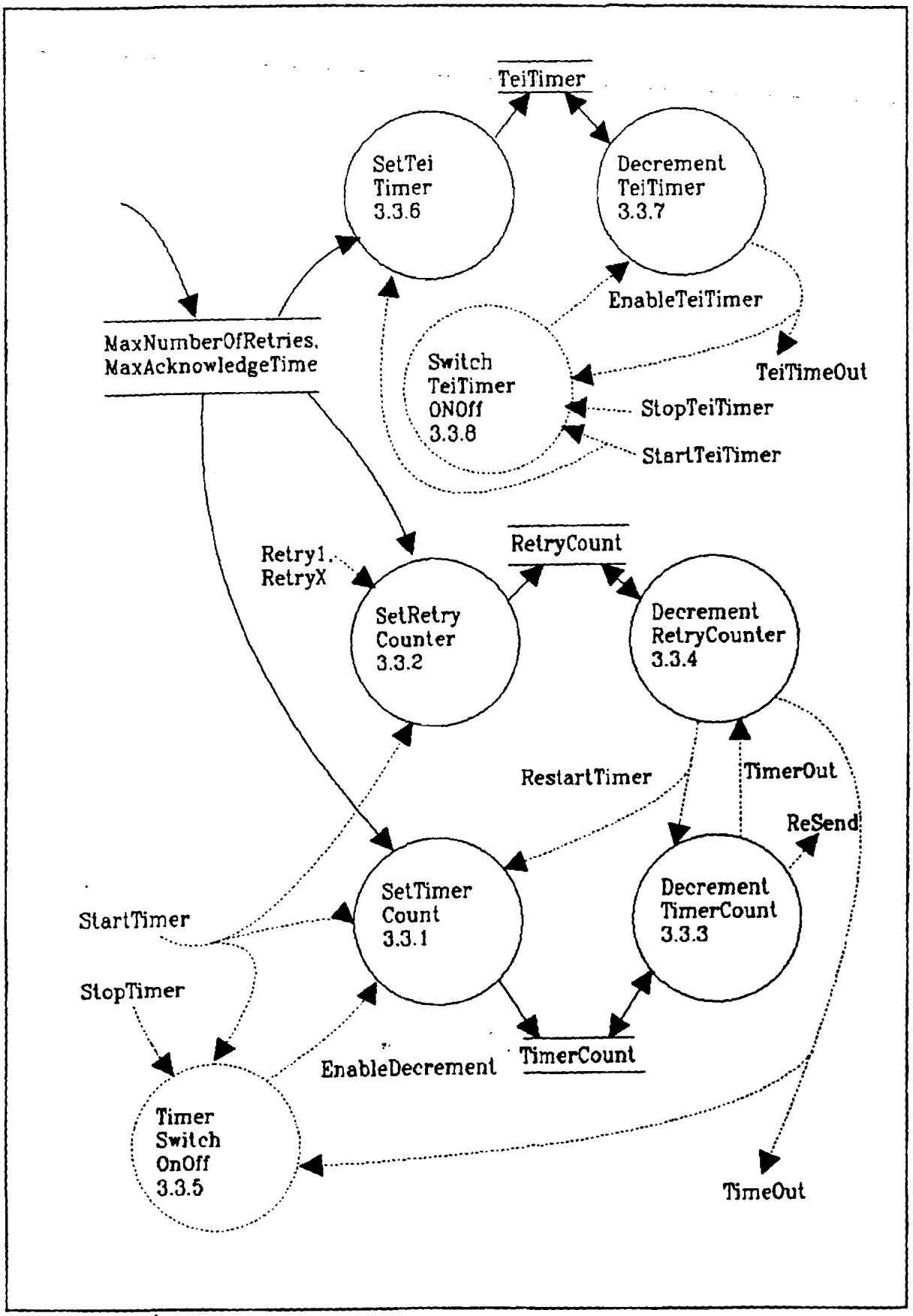
```
     set EnIToSend = true

if UToSend and NextByte
then set SendBytes = true
     set EnUToSend = true

if MDataToSend and NextByte
then set MDataToSend = true
     set SendBytes = true

if not IFrameToSend and NextByte
then set SendBytes = false
     set EnIToSend = false

if not UToSend and NextByte
then set SendBytes = false
     set EnUToSend = false

if not MDataToSend and NextByte
then set MDataToSend = false
     set SendBytes = false
```



FD 3.2.2 : BytesToFrameOut

```
PSPEC 3.2.2.1 : SendSyncs

*
Als het door EnableSync toegestaan wordt zendt dit process een
continue reeks van sync karakters.
*

if EnableSync and IssueBit and SyncBit = 1
then Set FrameOut = 0
     SyncBit = SyncBit + 1

if EnableSync and IssueBit and 1 < SyncBit < 8
then set FrameOut = 1
     SyncBit = SyncBit + 1

if EnableSync and IssueBit and SyncBit = 8
then set FrameOut = 0
     issue EndSync = true
     SyncBit = 1

If not EnableSync
then SyncBit = 1


PSPEC 3.2.2.2 : StuffBits

*
Vijf achter op een volgende 1 bits in InfoBitsToSend worden
gevolgd door een 0 bit.
*

if ActivateLink
then set EnableSync = true

if SendBits and EndSync
then set EnableSync = false
     EnableBits = true
     Onebitcount = 0

if EnableBits and IssueBit and OneBitCount < 5 and InfoBitToSend
= 1
then set FrameOut = InfoBitsToSend
     issue NextBit = true
     OnebitCount = OnebitCount + 1

if EnableBits and IssueBit and OneBitCount < 5 and InfoBitToSend
= 0
then set FrameOut = InfoBitsToSend
     issue NextBit = true
     OnebitCount = 0

if EnableBits = true and Issuebit and OneBitCount = 5
then set FrameOut = 0
     OneBitCount = 0

if not SendBits
```

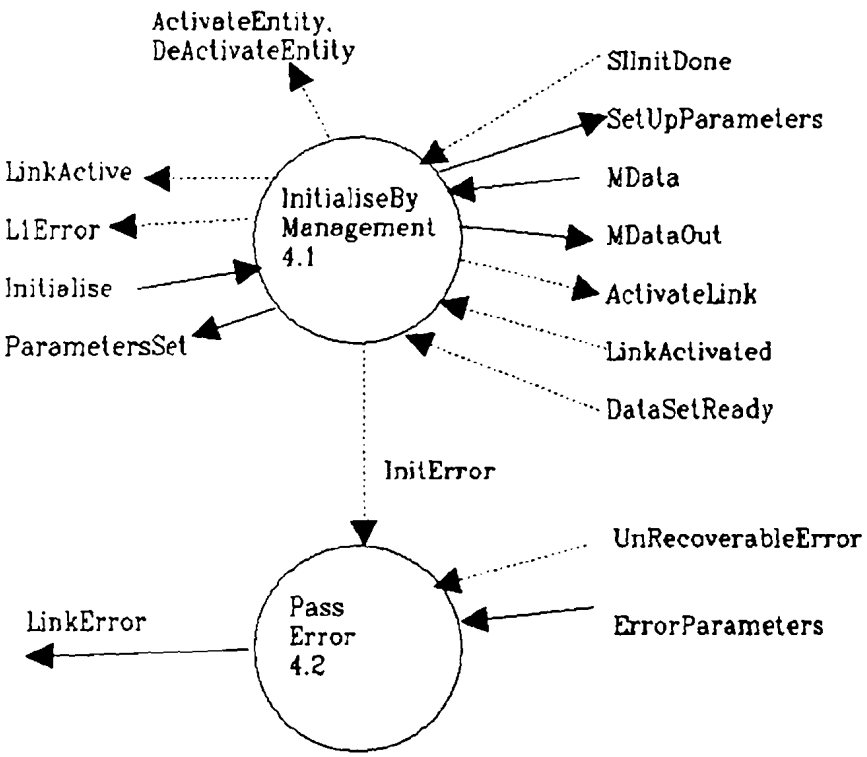
```
then set EnableSync = true
     EnableBits = false


PSPEC 3.2.2.3 : ByteToInfoToSend

*
Aangeboden bytes (BytesOut) worden in bits door gegeven aan
StuffBits als dit proces om een nieuw bit vraagt met NextBit.
Wanneer er een nieuw byte aangegeven moet worden wordt bepaald
door NextByte. Een en ander wordt gedaan zolang SendBytes dit
signaleert.
*

if not SendBytes
then set BitPointer = 1

if SendByte
then set SendBits = true

if SendByte and BitPointer < 8 and NextBit
then set InfoBitToSend = bit BitPointer of BytesOut
     BitPointer = BitPointer + 1

if SendByte and BitPointer = 8 and NextBit
then set InfoBitToSend = bit BitPointer of BytesOut
     BitPointer = 1
     issue NextByte = true
```


TeiTimer
SetTei
Timer
3.3.6
Decrement
TeiTimer
3.3.7
EnableTeiTimer
MaxNumberOfRetries,
MaxAcknowledgeTime
Switch
TeiTimer
ONOff
3.3.8
TeiTimeOut
StopTeiTimer
StartTeiTimer
RetryCount
Retry1,
RetryX
SetRetry
Counter
3.3.2
Decrement
RetryCounter
3.3.4
RestartTimer
TimerOut
ReSend
SetTimer
Count
3.3.1
Decrement
TimerCount
3.3.3
StartTimer
StopTimer
EnableDecrement
TimerCount
Timer
Switch
OnOff
3.3.5
TimeOut

FD 3.3 : Timers

```
PSPEC 3.3.1 : SetTimer

*
Zet bij StartTimer of RestartTimer de timer waarde op
MaxAcknowledgeTime
*

if StartTimer
then issue TimerCount = MaxAcknowledgeTime

if RestartTimer
then issue TimerCount = MaxAcknowledgeTime


PSPEC 3.3.2 : SetRetryCounter

*
Zet de hertransmissie teller op de gewenste waarde (1 /
MaxNumberOfRetries).
*

if StartTimer and Retry1
then issue RetryCount = 1
if StartTimer and RetryX
then issue RetryCount = MaxNumberOfRetries


PSPEC 3.3.4 : DecrementRetryCount

*
Verlaag de hertransmissie teller en geef een timeout als het
aantal hertransmissies gedaan is. Indien niet gedaan, herstart
de timer.
*

if TimerOut
then issue RetryCount = RetryCount -1

if RetryCount = 0
then issue TimeOut = true


PSPEC 3.3.5 : TimerSwitchOnOff

*
Enable / disable het verlagen van de timer
*

if StartTimer
then issue EnableDecrementTimerCount = true

if StopTimer
then issue EnableDecrementTimerCount = false

if TimeOut
then issue EnableDecrementTimerCount = false
```

```
PSPEC 3.3.6 : SetTeiTimer

*
Zet de TeiTimer bij het aanvragen van de Tei parameters op vier
maal MaxAcknowledgeTime.
*

if StartTeiTimer
then issue TeiTimer = 4 * MaxAcknowledgeTime


PSPEC 3.3.7 : DecrementTeiTimer

*
Verlaag de timer tijdens TEI aanvragen.
*

if EnableTeiTimer
then TeiTimer = TeiTimer -1

if TeiTimer = 0
then issue MData = TeiTimeOut


PSPEC 3.3.8 : SwitchTeiTimerOnOff

*
Enable / disable verlagen van de timer
*

if StartTeiTimer
then set EnableTeiTimer = true

if StopTeiTimer
then set EnableTeiTimer = false

if TeiTimeOut
then set EnableTeiTimer = false
```



FD 4 : ManageLayer

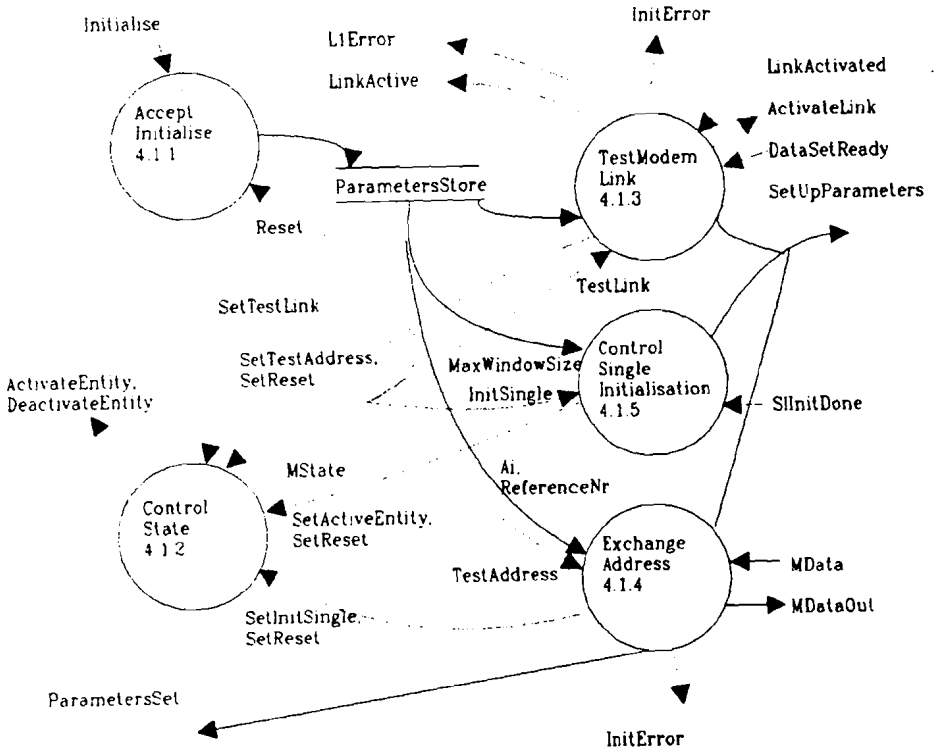
PSPEC 4.1 : InitialiseByManagement

*
Zorg voor de initialisatie van de controller.
*

PSPEC 4.2 : PassError

*
Als UnRecoverableError aangeeft dat een fout is opgetreden wordt de informatie die door ErrorParameters wordt aangeboden als LinkError aan de host doorgegeven. Een fout tijdens de initialisatie (InitError) wordt ook met LinkError doorgegeven.
*

```
if UnRecoverableError
then issue LinkError = ErrorParameters

If InitError
then issue LinkError = InitError
```



FD 4.1 : InitialiseByManagement

PSPEC 4.1.1 : AcceptInitialise

*
Na Initialise moet de link getest worden. Tevens worden de parameters die in Initialise zitten opgeslagen in ParametersStore. De toestand moet nu met SetTestLink veranderd worden.
*

```
if Initialise and Reset
then issue ParametersStore = Initialise
     issue SetTestLink
```

PSPEC 4.1.2 : ControlState

*
Hoe ver de initialisatie van de controller gevorderd is wordt door dit proces met de flow MState aangegeven. De toestanden zijn: Reset, TestLink, TestAddress, InitSingle, ActiveEntity. De toestand wordt veranderd door de volgende flows: SetReset, SetTestLink, SetTestAddress, SetInitSingle, SetActiveEntity. Bij het aannemen van de toestand Reset wordt tevens de DeActivateEntity flow geactiveerd. Het aannemen van de ActiveState toestand gaat gepaard met het signaal ActivateEntity
*

```
if SetReset
then set MState = Reset
     issue DeActivateEntity

if SetTestLink
then set MState = TestLink

if SetTestAddress
then set MState = TestAddress

if SetInitSingle
then set MState = InitSingle

if SetActiveEntity
then set Mstate = ActiveEntity
     issue ActivateEntity
```

PSPEC 4.1.3 : TestModemLink

*
Bij TestLink wordt eerst gekeken of ModemReady is. Gelijktijdig worden de parameters MaxAcknowledgeTimer, MaxNumberOfBytes, Sapi en MaxNumberOfRetries doorgegeven aan AssembleDisAssembleFrames. Met ActivateLink wordt dan de continue sync stroom op de lijn gezet. Als ook een sync stroom op de inkomende lijn gedetecteerd wordt zal dit met LinkActivated gemeld worden. Dit wordt nu met LinkActive aan de host door gegeven. De toestand wordt met SetTestAddress gezet. Als ModemNotReady dan wordt dit met L1Error

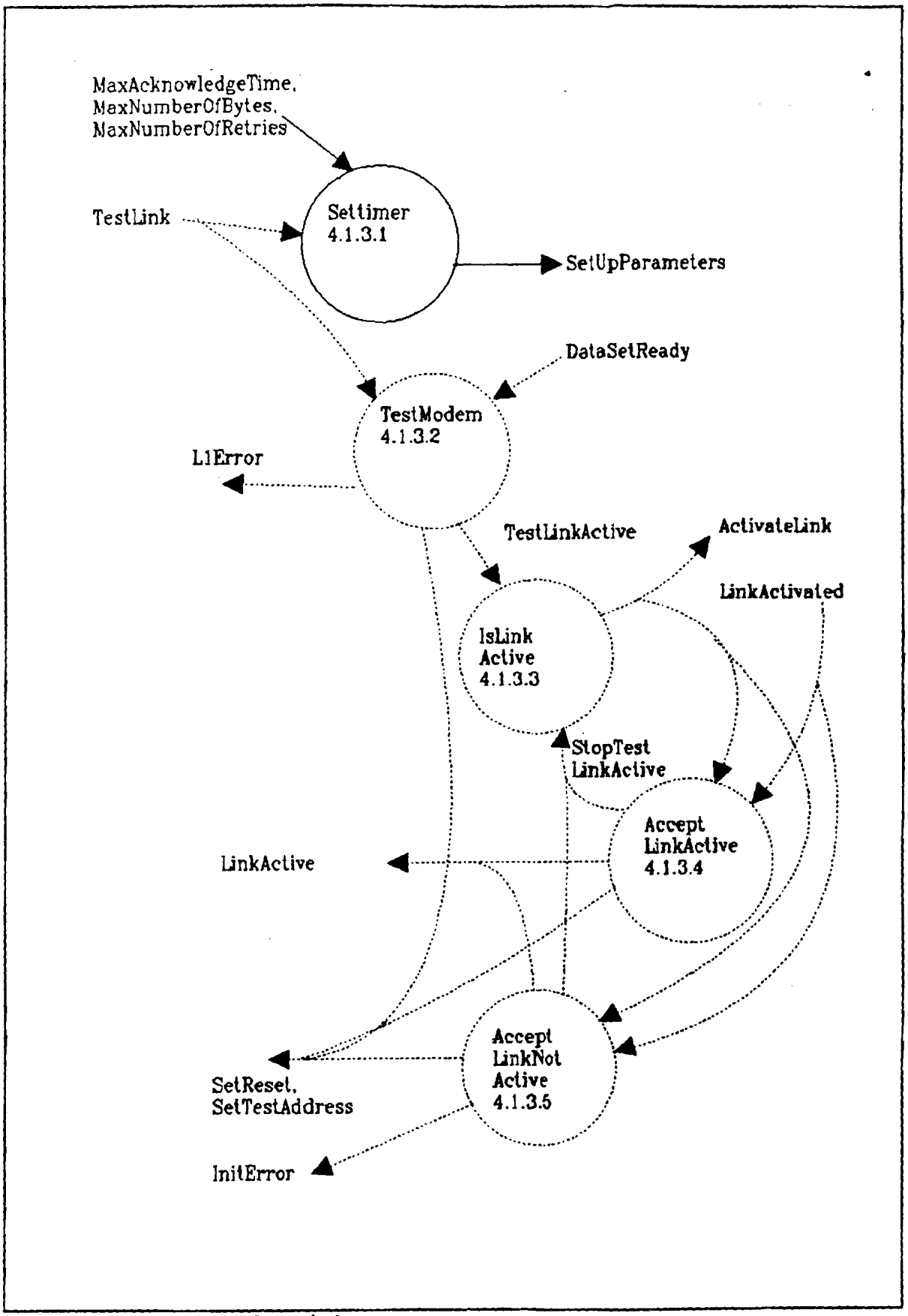
aan de host door gegeven en wordt de toestand met SetReset veranderd. LinkActivated = LinkTimeOut moet als InitError = LinkTimeOut gemeld worden.
*

PSPEC 4.1.4 : ExchangeAddress

*
TestAddress heeft tot gevolg dat via MDataOut een frame naar de peer gestuurd wordt met de aanvraag om de gewenste Tei te checken (IDRequest) met een referentienummer en de gewenste Tei (ReferenceNR, Ai). Als de gevraagde Tei toegewezen is wordt dit met MData = IDResponse, ReferenceNr, Ai gemeld. Dit wordt dan door gegeven aan de host met ParametersSet. De toestand wordt dan met SetActiveEntity gezet. Als de gevraagde Tei niet gebruikt kan worden is MData = IDDenied, ReferenceNr, Ai. Dit wordt ook aan de host doorgegeven met ParametersSet. Met SetReset wordt de toestand nieuw gezet. Tevens wordt InitError = DeniedError. Als de aanvraag om een Tei niet binnen vier maal MaxAcknowledgeTime aangekomen is wordt dit gemeld met MData = TeiTimeOut. Dit wordt via InitError = TeiTimeOut aan de host doorgegeven. Tevens wordt SetReset naar ControlState gestuurd.
*

PSPEC 4.1.5 : ControlSingleInitialisation

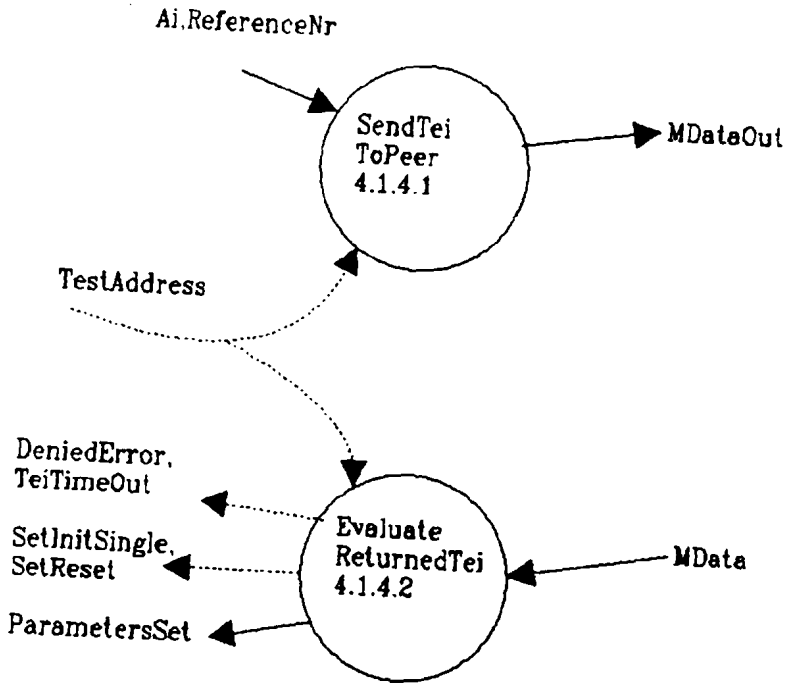
*
Bij ontvangen van InitSingle wordt SetUpParameters gelijk aan MaxWindowSize. Als de beide gereed meldingen van TransferData (InitSIDone) aangekomen zijn wordt de toestand met SetActiveEntity veranderd.
*



FD 4.1.3 : TestModemLink

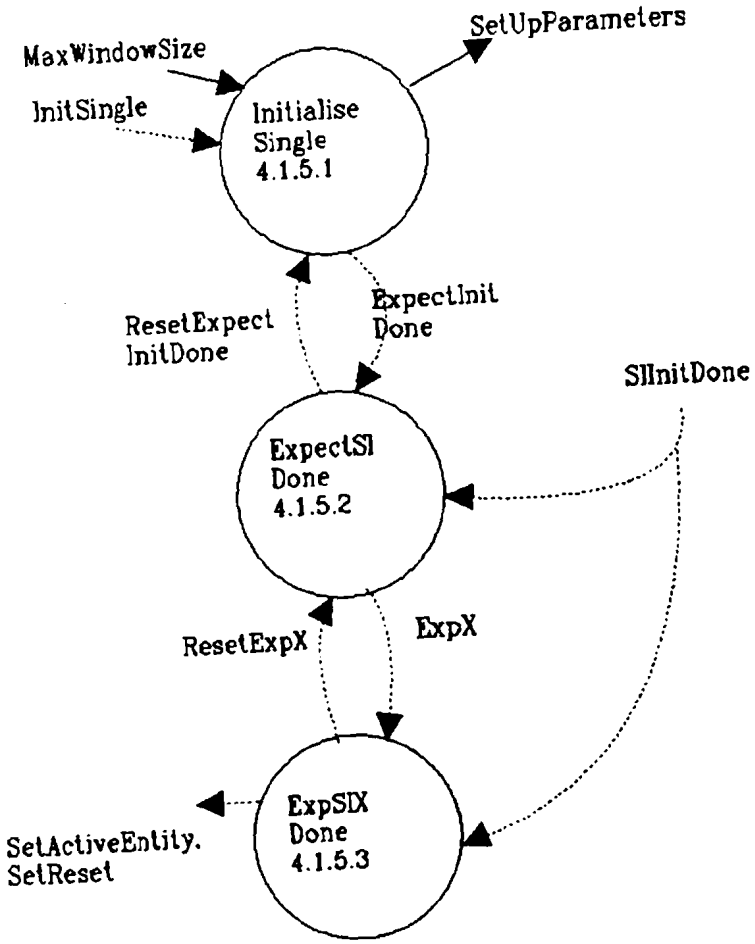
```
PSPEC 4.1.3.1 : SetTimer

*
Geeft parameters voor de timer en hertansmissie door.
*

If TestLink
then    issue    SetUpParameters    =    MaxAcknowledgeTime,
MaxNumberOfBytes, MaxNumberOfRetries, Sapi


CSPEC 4.1.3.2 : TestModem

*
Test of DataSetReady true is. Zo ja, ga verder met de
initialisatie.
*

if TestLink and ModemNotReady
then issue L1Error = true
     issue SetReset = true

if TestLink and ModemReady
then issue TestLinkActive

CSPEC 4.1.3.3 : IsLinkActive

*
Test of link Active is.
*

if TestLinkActive
then set ActivateLink = true

if StopTestLinkActive
then set ActivateLink = false

CSPEC 4.1.3.4 : AcceptLinkActive

*
LinkActive gedetecteerd, ga verder met adres verificatie.
*

if LinkActivated and ActivateLink
then issue LinkActive = true
     issue TestAddress = true
     issue StopTestLinkActive = true

CSPEC 4.1.3.5 : AcceptLinkNotActive

*
De link is niet tijdig active geworden. Geef dit als fout door
aan de host.
*

if LinkActivated = LinkTimeOut and ActivateLink
```

```
then issue InitError = LinkTimeOut
     issue SetReset = true
```



FD 4.1.4 : ExchangeAddress

```
PSPEC 4.1.4.1 : SendTeiToPeer

*
Zend TEI voor verificatie naar de peer controller.
*

if TestAddress
then issue MDataOut = IDRequest, ReferenceNr, Ai


PSPEC 4.1.4.2 : EvaluateReturnedTei

*
Als de TEI oke is kan de initialisatie doorgaan, anders wordt
een fout gemeld.
*

if TestAddress and IDResponse
then issue SetSingleInit = true
     issue ParameterSet = IDResponse, ReferenceNr, Ai


if TestAddress and IDDenied
then issue SetReset = true
     issue InitError = DeniedError

if MData = TeiTimeOut
then issue InitError = TeiTimeOut
```



FD 4.1.5 : ControlSingleInitialisation

```
PSPEC 4.1.5.1 : InitialiseSingle

*
Initialiseer TransferData.
*

if InitSingle
then issue SetParameters = MaxWindowSize
     set ExpectInitDone = true

if SingleInit and ResetExpectInitDone
then set ExpectedInitDone = false

PSPEC 4.1.5.2 : ExpectSIDone

*
Wacht tot het single frame data transfer proces zich gereed meld.
*

if SIInitDone = 0 and ExpectInitDone
then set ExpX = 1
     issue ResetExpectInitDone = true

if SIInitDone = 1 and ExpectInitDone
then set ExpX = 0
     issue ResetExpectInitDone = true
if SITimeOut and ExpectInitDone
then issue ResetExpectInitDone = true

PSPEC 4.1.5.3 : ExpSIXDone

*
Wacht op de tweede gereedmelding.
*

if ExpX = SIInitDone
then issue SetActiveEntity = true
     issue ResetExpX = true

if SITimeOut and ExpX
then issue SetReset = true
     issue ResetExpX = true
```

APPENDIX B : REQUIREMENTS DICTIONARY

Tabel B.1 : Data Dictionary notaties

Symbool	(Betekenis) Beschrijving
=	(bestaat uit) geeft aan hoe een flow samengesteld is of welke waardes hij heeft.
+	(samen met) geeft aan dat flows samengevoegd worden
{ }	(iteraties) hiermee wordt het aantal keren dat een flow voorkomt aangegeven
[Î]	(keuze) slechts een van de flows kan voor komen
( )	(optioneel) de flow kan maar hoeft niet voor te komen
" "	(letterlijk) de ingesloten tekst komt letterlijk in de flow voor
* *	commentaar
\ \	(primitieve) geeft aan dat een flow een primitieve flow is

```
AcceptBit =
\Accept a bit from Modem in FrameIn\
2 values :true, false ,C

AcceptMDLUnit =
\Order the controller to send management data in PacketToSend
using unacknowledged datatransfer\
2 values :true, false ,C

AcceptPacket =
\Order the controller to send the data in PacketToSend\
2 values :true, false ,C

AcceptUnit =
\Order the controller to send the data in PacketToSend using
unacknowledged datatransfer\
2 values :true, false ,C

AckSI =
*Send an acknowledge frame having an F bit set to this value*
2 values : Fis0, Fis1 , C

ActivateEntity =
\Put controller in TEI assigned state after initialisation\
2 values: true, false ,C

ActivateLink =
\Put link in active state\
2 values : true, false ,C

AdjustVariable =
*Opdracht om window variablene aan te passen.*
AdjustLastAck, ResetNextFrameToSend, AdjustFrameExpected , C


AwaitSI =
\Await the sequence number that will be used by te next command
frame\
2 values : 0, 1 , C

BufferPacketIn =
\Geeft aan dat data van de host gebufferd moet worden\
2 values : true, false , C

ByteStrobe =
\Pass next byte\
2 values : true, false ,C

ClearBuffer =
\Buffered bytes are not valid, thus dispose of them\
2 values : true, false , D

ConnectState =
\Current State of the controller\
7 values : Idle, TeiAssigned, Connecting, DisConnecting,
Restarting, SingleFrame , C
```

```
ControlField =
*Type van aangekomen frame*
[IÎRNRÎRRÎREJÎ],([C/R]),(C)

ControlField =
\A    S or I frame is received and must be serviced by
TransferData\
7 values : RRService, RNRService, REJService, IService
,SI0Service, SI1Service, UIService ,C

ControlFieldOut =
\Order AssembleDisAssembleFrames to send a S or I frame using
this controlfield\
7 values : Iout, RROut, RNROut, REJOut, UIOut, SI0Out, SI1Out
,C

ControlFieldSIOut =
*ControlField van het frame dat verzonden moet worden.*
[IOutÎRNROutÎRROutÎREJOut],([C/R]),(C) , C

CRCError =
\CRC error detected in DestuffedIn\
2 values : true, false ,D

CRCOke =
\No CRC error detected in DestuffedIn\
2 values : true, false ,D

Data =
\Packet information contained in the received information frame\
1 ( bytes ) MaxNumberOfBytes ,D

DataOut =
\Packet information that has to be included in an information
frame\
1 ( bytes ) MaxNumberOfBytes ,D


DataSetReady =
[ ModemReady Î ModemNotReady] ,C

DataState =
\State of Transfer data process\
NextFrameToSend + FrameExpected + ControlField , D

DeActivateEntity =
\Put controller in IdleState\
2 values : true, false ,C

DelayedIn =
\Delayed version of FrameIn. This flows contains no syncs
anymore\
2 values : 0, 1 ,D
```

```
DestuffedBits =
\Stuffed zero are removed from DelayedIn\
2 values : 0, 1 ,D

DisConnectIndication =
\Peer request to disconnect a datalink\
2 values : single, multiple ,C

DisConnectRequest =
\Disconnect a datalink\
2 values : single, multiple ,C

EnablePassToHost =
\Maak het doorgeven van ServiceFrame informatie aan de hos
mogelijk\
2 values : true, false , C

EnableSendUIFrame =
\Maak doorgeven van packet informatie mogelijk\
2 values : true, false , C

EnableTransferDataUnit =
\Met dit signaal wordt de overdracht van UI frames mogelijk
gemaakt. Dit gebeurt direct nadat de Tei is vastgesteld.\
2 values : true, false , C

EnableUIFrameToHost =
\Maak het doorgeven van de data in een UIFrame naar de host
mogelijk\
2 values : true, false ,C

ErrorParameters =
[DataState + \undefined control field\ + \not permitted frame\
+ \I field to long\  + \NrError\ Î \received reset info\ ] ,D

ExpectInitDone =
\Expect an SIInitDone from TransferDataSingle\
2 values : true, false , C

ExpX =
\Expect then other acknowledge frame from TransferDataSingle\
2 values : true, false , C

FrameError =
\Frame error occured\
2 values : true, false ,C

FrameIn =
\bit from modem\
2 values : 0, 1 ,D

FrameOut =
\bit to modem\
2 values : 0, 1 ,D

HostBusy =
```

```
\Host not able to setup or disconnect a datalink\
2 values : true, false ,C

InitDone =
\The expected sequence number of the peer controller is received
and stored\
2 values : 0, 1 ,C

InitError =
\Signal to PassError which error ocurred during initialisation\
3 values : LinkTimeOut, TeiTimeOut, DeniedError , C

Initialise =
*SetUp link Mode for multiple frame mode and pass operational
parameters:
BasicîExtended : set length of control field
AI : desired TEI
SAPI desired SAPI
ReferenceNr : random number used in TEI negotiation
*
[ Basic Î Extended ] + Ai + ReferenceNrs + Sapi +
MaxNumberOfBytes + MaxNumberOfRetries + MaxAcknowledgeTime +
MaxWindowSize ,D

IssueBit =
\Indicates when a bit sould be offered to Modem\
2 values :true, false ,C

L1Error =
\Dataset not ready\
2 values : true, false ,C

LinkActivated =
\Report link in active state\
2 values : true, false ,C

LinkActive =
\Link in active state\
2 values : true, false ,C

LinkError =
*signal an error to the host*
[ LinkIdle î ErrorParameters î InitError ] ,D

M =
\het betreft hier een frame voor het management\
2 values : true, false ,C

MData =
*Management data received for address negotiation*
1 { bytes } 5

MData =
*Reponse from network on tei request*
[IDResponse + ReferenceNr + Ai î IDDenied + ReferenceNr + Ai ]
```

```
MDataOut =
*Request to network for tei*
[IDResponse + ReferenceNr + Ai ĩ IDDenied + ReferenceNr + Ai ]

MDataOut =
*Management data to send for address negotiation*
1 ( bytes ) 5

MDLUnitArrival =
\A management packet is received in unacknowledged transfer
mode\
2 values :true, false ,C

MState =
\MState reflects the state of the initialisation proces. This
state determines which action will be performed next\
5 values : Reset, TestLink, TestAddress, InitSingle,
ActiveEntity , C

N(R) =
\Acknowledge sequence nummer uit ontvangen frame\
0( )MaxWindowSize

NrError =
\Report NrError to control connection\
2 values : true, false

PacketAccepted =
\Signal a packet is sent to the peer host\
2 values :true, false ,C

PacketArrival =
\A packet is received in acknowledged transfer mode\
2 values :true, false ,C

PacketRead =
\A received packet is accepted\
2 values :true, false ,C


PacketReceived =
\A packet which was received via the link\
1 ( Bytes ) MaxNumberOfBytes ,D

PacketToSend =
\Data which the controler has to send to the peer host\
1 ( bytes ) MaxNumberOfBytes ,D

ParametersSet =
*Setup parameters used for intialisation*
[ Basic ĩ Extended ] + Ai + ReferenceNrs + Sapi ,D

ParametersStore =
\Stored parameters for the initialisation\
MaxNumberOfBytes, MaxAcknowledgeTime, MaxNumberOfRetries,
MaxWindowSize, Ai, Sapi, ReferenceNr , C
```

```
PassPacketToHost =
\Geeft aan dat frame data door gegeven moet woren aan de host\
2 values : true, false , C

PassToHost =
\Geef  aan dat de peer frames naar de host doorgegeven kunnen
worden.\
2 values : true,false , C

ProcessBits =
\Bits to be processed are arriving while this signal is true\
2 values : true, false ,D

RecDataBytes =
\The information of DestuffedIn is passed here in byte form\
bytes ,D

RemoteState =
*The state of the remote controler at te moment an error was
detected. This information was contained in FRMR.*
DataState ,D

RequestStateVar =
\Request State variables for error message\
2 values : true, false ,C

Resend =
\Resend frames after timeout\
2 values : true, false , C

ResetExpectInitDone =
\InitDone from TransferDataSingle received\
2 values : true, false , C

ResetExpX =
\Other acknowledge from TransferDataSingle received\
2 values : true, false , C

RestartConnect =
\Connect after severe error\
2 values : true, false ,C

Retry1 =
\Number of retries before a timeout is issued is one\
2 values : true, false , C

RetryX =
\MaxNumberOfRetries will be executed before a timeout is
issued\
2 values : true, false , C

RInit =
\The remote host send a SI0 frame to enquire for the seuence
number to be used in data exchange\
2 values : 0, 1 ,C
```

```
SendC =
\Send Command\
2 values : true, false , C

SendSI0 =
\Send a SI0 frame to inquire for the next sequence number\
2 values : true, false , C

SendSIx =
*Send sequence number of next command tat will be send\
2 values : 0, 1 ,C

ServiceUFrame =
\Accept an U frame for connecting/disconnecting/error
notification\
5 values : SABM, DISC, DM, UA, FRMR ,C

SetActiveEntity =
\Order ControlState to set MState to ActiveEntity\
2 values : true, false , C

SetConnected =
\Set ConnectState to Connected\
2 values : true, false , C

SetConnecting =
\Set ConnectState to Connecting\
2 values : true, false , C

SetDisConnecting =
\Set ConnectState to DisConnecting\
2 values : true, false , C

SetInitSingle =
\Order ControlState to set MState to InitSingle\
2 values : true, false , C

SetReset =
\Order ControlState to set MState to Reset\
2 values : true, false , C

SetRestarting =
\Set ConnectState to Restarting\
2 values : true, false , C

SetSingleFrame =
\Set ConnectState to SingleFrame\
2 values : true, false , C

SetTeiAssigned =
\Set ConnectState to TeiAssigned\
2 values : true, false , C

SetTestAddress =
\Order ControlState to set MState to TestAddress\
```

```
2 values : true, false , C

SetTestLink =
\Order ControlState to set MState to TestLink\
2 values : true, false , C

SetUpIndication =
\Peer request to setup a datalink\
2 values : single, multiple ,C

SetUpParameters=
*parameters for AssembleDisAssembleFrames en TransferData:
BasicÎExtended : set length of control field
AI : desired TEI
SAPI desired SAPI
ReferenceNr : random number used in TEI negotiation
*
[  Basic  Î  Extended  ]  +  Ai  +  ReferenceNrs  +  Sapi  +
MaxNumberOfBytes + MaxNumberOfRetries + MaxAcknowledgeTime +
MaxWindowSize ,D

SetUpRequest =
\Setup a datalink for single or multiple frame acknowledged
datatransfer\
2 values : single, multiple ,C

SICommand =
*Pass command to the single frame data transfer process*
2 values : SISetUp, SIDisc ,C

SIExpected =
*The sequence number of the next SIx frame expected*
2 values : 0,1 , C

SIInitDone =
\Signal to ManageLayer the end of the initialisation\
2 values : 1, 2 , C

SIResponse =
*Pass responses from the single frame data transfer process*
4 values : RSISetUp, RSIDisc, SIDiscDone, SISetUpDone, C

SISent =
\S or I frame send\
2 values : true, false ,C

SIServiced =
\A recieved S or I service request is performed\
2 values : true, false ,C

SIToSend =
*The frame sequence number of the next SIx frame that will be
send*
2 values : 0,1 , C

StartDataTransfer =
```

```
\Start data transfer\
2 values : true, false

StartTeiTimer =
\Start the Tei timer during address verification\
2 values : true, false , C

StateAvailable =
\The state of the data transfer process is available on
DataState\
2 values : true, false ,C

StatusInfo =
* Uitkomst van de vergelijking van de aangekomen sequence
nummers met de toestand variabelen.*
Full, Empty , C

StatusVariable =
*Toestand van de buffers, afgeleidt uit de window variablen*
Empty, Full , C

StopDataTransfer =
\Stop data transfer\
2 values : true, false

StopTestLinkActive =
\ Stop waiting for an active link\
2 values : true, false , C

StopTieTimer =
\Stop the Tei timer after succesfull address verification\
2 values : true, false , C

TBufferEmpty =
\Geeft aan dat alle bytes uit het packet door gegeven zijn aan
AssembleDisAssembleFrames\
2 values : true, false , C

TestLinkActive =
\Start the detection of an active link during initialisation\
2 values : true, false , C

UFrameOut =
\Output an U frame for connecting/disconnecting/Error
notification\
5 values : SABMOut, DISCOut, DMOut, UAOut, FRMROut ,D

UFrameSent =
\U frame is sent\
2 values : true, false ,C

UIArrival =
\UIFrame voor laag 3 is aangekomen\
2 values : true, false , C

UIMArrival =
```

```
\UI frame voor management is aangekomen\
2 values : true, false , C

UnitArrival =
\A packet is received in unacknowledged transfer mode\
2 values :true, false ,C

UnrecoverableError =
\Signale error condition to management\
2 values : true, false ,C

UServiced =
\U frame is serviced\
2 values : true, false ,C

WindowVar =
*Window variabelen van het windowprocess*
LastAck, NextFrameToSend, FrameExpected , D

ZeroLength =
\Inform InputSI about the length of the information part of the
last command frame\
2 values : true, false , C
```