Het bewerken van fysiologische signalen : met uitbreiding op het European Data Format

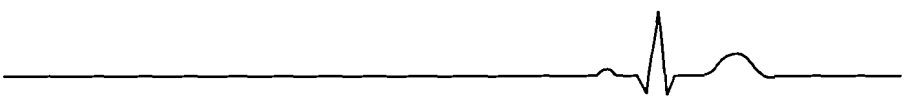
Souabi, Brahim

Award date:
1996

TECHNISCHE UNIVERSITEIT EINDHOVEN
FACULTEIT ELEKTROTECHNIEK
VAKGROEP MEET & BESTURINGSSYSTEMEN
SECTIE MEDISCHE ELEKTROTECHNIEK

Het bewerken van Fysiologische signalen

Met uitbreiding op het European Data Format



Door: Brahim Souabi

In opdracht van:	Vakgroep Psychologie, sectie Psychonomie aan de Katholieke Universiteit Brabant & vakgroep Meet en Besturingssystemen, sectie Medische Elektrotechniek aan de Technische Universiteit Eindhoven
Begeleiders TUE:	Dr.ir. P.J.M. Cluitmans ir. M. van de Velde
Begeleiders KUB:	ir. M.M.C. van den Berg-Lenssen Dr. G.J.M. van Boxtel
Datum:	September 1996

Summary

The Psychonomy section at Tilburg University and the Medical Electrical Engeneering section at the Eindhoven University of Technology cooperate in many fields for quite some time.
The section Psychonomy is going to process its data on a new platform. This change of platforms causes some problems. The software needs to be redeveloped for the new platform. This graduation project consists of the development of the software for the new platform. Next to a new operating system, the section has chosen to use a different data format (European Data Format). This format though, is not suited for data storage of event related data commonly used by the section Psychonomy.
In cooperation with Eindhoven University of Technology an extension on this data format is designed to make it suitable for storage of event related data/information. This is done without changing the standard definition of the EDF format.

After deciding a new platform with a fully elaborated protocol on how to store our data, the next step will be the development of a conversion program. This program converts the data in the old data format (Fysian) to the new data format ('extended' EDF). With this conversion we can process 'old' data on the new platform.

The final step of this graduation project is the development of a software library, which will be used by future programmers to develop processing software that uses the (extended) EDF standard. This library consists of numerous functions and routines that handle the input and output to and from an EDF data file. Instead of spending lots of time examining the structure of an EDF data file, a programmer can use these functions to read the data from or write it to the EDF data file. During the development of this library simplicity and structure were important key words, because in future different programmers will use this source and need to understand it with ease.

The software developed is tested with different methods. The conversion program is tested by processing its converted data with other software. One of these tests is averaging a data file that is converted with software from Eindhoven University and compare it with the average of the original data file which is processed at Tilburg University. This test showed that the converted data give the same results. Software from other institutes is used to display the content of the converted data files. Since the extended EDF does not deviate from the standard EDF, these programs had no problems reading the converted data.

Some functions and routines are already being used in software of other programmers at Tilburg University and the Eindhoven University of Technology.

The next step is the development of the processing software for the Psychonomy section so it can switch to the new platform. The library can be extended with other function that take 'low-level' input/output handling out of the programmers hands. This is mainly needed for processing the Event and the Info channel.

Samenvatting

De sectie Psychonomie aan de Katholieke Universiteit Brabant werkt al enige tijd samen met de sectie Medische Elektrotechniek aan de Technische Universiteit Eindhoven.
De sectie Psychonomie gaat de bewerking van experimentele data op een nieuw platform uitvoeren. Deze wisseling van platformen met verschillende besturings-systemen brengt de nodige problemen met zich mee. De software moet opnieuw ontwikkeld worden voor het nieuwe besturings-systeem.
Dit afstuderen bestaat uit het ontwikkelen van de software op het nieuwe platform.
Naast een nieuw besturings-systeem heeft men ook gekozen voor een ander data formaat (European Data Format). Het nieuwe data formaat is echter niet geschikt voor opslag van de (event gerelateerde) data zoals deze op de KUB gebruikt wordt.
In samenwerking met de TUE is hier een extensie voor ontworpen, die het mogelijk maakt de data (met event-informatie) op te slaan in de nieuwe standaard, zonder deze fundamenteel te veranderen.

Na de keuze van een nieuw besturings-systeem en een uitgewerkt protocol om de data op te slaan is de volgende stap in het ontwikkel-project het ontwerpen van een conversie programma dat de oude experimentele data, die nog in het oude (Fysian) formaat is opgeslagen om kan zetten in het nieuwe 'extended' EDF. Hiermee kan in de toekomst de oude data ook op het nieuwe platform verwerkt worden.

De laatste stap in mijn afstuderen is het ontwikkelen van een software bibliotheek die door toekomstige programmeurs gebruikt gaat worden bij het ontwikkelen van bewerkings-software voor EDF data. Deze bibliotheek bevat een aantal functies die voornamelijk de input en output routines van/naar EDF data files voor derden vereenvoudigt. Een programeur hoeft zich niet te verdiepen in de 'echte' structuur van een EDF data file, maar kan met een eenvoudige functie aanroep de data uit een data file lezen of er naar toe schrijven.
Bij het ontwikkelen van een bibliotheek zijn eenvoud en structuur zeer belangrijk, daar anderen met deze software moeten werken en moeten kunnen begrijpen.

De geprogrammeerde software is met verschillende methoden getest. Het conversie programma is getest door de geconverteerde data met behulp van software van derden te bewerken en deze bewerkingen te vergelijken. Ook is software van derden gebruikt om de geconverteerde data te bekijken. Daar het nieuwe formaat niet mag afwijken van de standaard mag deze software daar geen probleem mee hebben.

Enkele routines uit de bibliotheek zijn al in gebruik in software van anderen, zowel op de TUE als op de KUB.

De basis is gelegd. Nu moet de bewerkingssoftware geschreven worden die op het oude besturings-systeem al voorhanden is, zodat men aan de sectie Psychonomie kan overstappen naar het nieuwe systeem. De bibliotheek kan nog uitgebreid worden met andere functies die 'low-level' bewerkingen op de data van de programmeur afnemen, voornamelijk voor het bewerken van het Event en het Info kanaal.

Inhoud

1. Inleiding

Computers zijn een ideaal hulpmiddel bij het bewerken van grote hoeveelheden data. Een computer kan wiskundige berekeningen op veel data vele malen sneller dan het menselijke brein. Voor het bewerken van de data moet speciale software geschreven worden, welke de gewenste bewerking op de data uit moet voeren.
Hoe slaan we de data op zodat de software deze goed kan interpreteren? Hoe gaan we de data bewerken?
Dit zijn vragen waar bijna elke instituut, dat zich bezig houdt met data bewerking met behulp van computers, een ander antwoord voor heeft.
Zo heeft de sectie Psychonomie aan de Universiteit in Tilburg een eigen antwoord of met andere woorden een eigen data formaat, genaamd Fysian, voor het opslaan en bewerken van de experimentele data.
Het data formaat is geheel ontwikkeld aan de sectie Psychonomie en is specifiek gericht op het soort experimenten waar men zich aan de sectie mee bezig houdt. De bewerkingssoftware loopt op VAX/VMS-systemen en is geschreven in de programeer-taal Fortran.

De VAX/VMS-systemen zullen echter gaan verdwijnen aan de Universiteit. Een groot probleem voor de sectie Psychonomie, die alle data bewerking op de VAX/VMS uitvoert. Men zal over moeten gaan naar een ander platform. Door de snelle groei van de PC met zijn gunstige prijs/prestatie verhouding is gekozen toekomstige bewerkingen op de PC, met MS-DOS als besturingssysteem, uit te voeren. Een nadeel hiervan is dat de software die voor de VAX/VMS-systemen geschreven was, moet worden vertaald of herschreven voor de PC.

Mijn afstuderen op de sectie Medische Elektrotechniek (EME) van de vakgroep Meet en Besturingssystemen (MBS) aan de Technische Universiteit Eindhoven (TUE) is verricht in samenwerking met de sectie Psychonomie aan de Katholieke Universiteit Brabant (KUB). Het doel van dit afstudeer project is een basis te maken voor het (software-) ontwikkel project.

Om redenen die in de hoofdstukken 2 en 3 zullen worden beschreven, is bij de sectie Psychonomie tevens gekozen voor een ander data formaat, genaamd European Data Format (EDF).
Dit formaat heeft naast enkele voordelen ten opzichte van het eigen formaat Fysian de nadeel dat het 'eigenlijk niet' ontwikkeld is voor het soort experimenten welke aan de KUB gehouden worden. Er zal een uitbreiding van dit formaat worden beschreven, waarbij het formaat niet fundamenteel verandert. De basis van deze uitbreiding is op de TUE/EME gelegd door ir. M van de Velde en Dr.ir. P.J.M.Cluitmans en als zodanig ook al geïmplementeerd in een meetopstelling. Tijdens dit afstudeerwerk is dit concept verder uitgewerkt. Het data-formaat en de uitbreiding zullen in hoofdstuk 3 nader toegelicht worden.

We weten nu met welk besturings-systeem we willen werken en hoe we de data gaan opslaan. De volgende stap in het project is het ontwikkelen van een conversie programma dat data in het eigen formaat omzet naar het nieuwe (aangepaste) EDF formaat. Dit is nodig om in de toekomst ook oude data op de nieuwe systemen te kunnen bewerken. Dit zal in hoofdstuk 4 uitgebreid behandeld worden.

De derde en voor mijn afstuderen de laatste fase in het project is het ontwikkelen van een basis bibliotheek met routines voor het bewerken van EDF data. Het gaat hier voornamelijk om input/output routines welke het toekomstige programmeurs eenvoudiger moet maken software te schrijven voor EDF data bewerkingsalgoritmen. Het doel van deze basis bibliotheek is tevens een basis structuur te ontwerpen waar programmeurs zich aan moeten houden, daar een verscheidenheid aan programmeurs met deze routines en data moet kunnen werken. Om elkaars software te begrijpen en uniformiteit aan te brengen is een goede basis structuur een zeer belangrijk punt in het software-ontwikkel traject. Hoofdstuk 5 geeft een samenvatting van deze bibliotheek.

In hoofdstuk 6 zijn enkele methoden beschreven die gebruikt zijn voor het testen van de geprogrammeerde software.

2. Event gerelateerde neurofysiologische metingen

2.1 Inleiding

Bij veel experimenten die op de secties Psychonomie en Medische Elektrotechniek gehouden worden meten we potentialen. Dit zijn potentialen die door het zenuwstelsel of hersenen gegenereerd worden. Deze potentialen worden *evoked potentials (EP)* genoemd.

Wat zijn evoked potentials ?
Evoked potentials worden gedefinieerd als elektrische reacties van het zenuwstelsel op zintuiglijke stimulaties. De evoked potentials worden aan de oppervlakte van de menselijke huid met behulp van elektroden gemeten.
Een enkele reactie op een stimulus heeft vaak een kleine amplitude en kan geheel weg vallen in het spontane EEG. Om deze evoked potentials toch uit een EEG te kunnen filteren worden meerdere reacties (op een gelijke stimulus) gemiddeld met behulp van een computer. Deze methode reduceert de amplitude van het achtergrond EEG die niet gecorreleerd is met de stimulus en verbetert de contouren van de reactie-componenten die tijdgebonden zijn met de stimulus.

In dit verslag noemen we de signaalperiode waarin een combinatie van een of meerdere stimuli en reactie(s) plaats vinden: een *trial*. Een gemiddelde van deze trials noemen we een *trial-gemiddelde*.

De term *event-related potential* (ERP) wordt vaak gebruikt om zowel EP's als ook andere soorten potentialen aan te geven die het resultaat zijn van cognitieve processen die op een stimulus volgen of eraan voorafgaan.

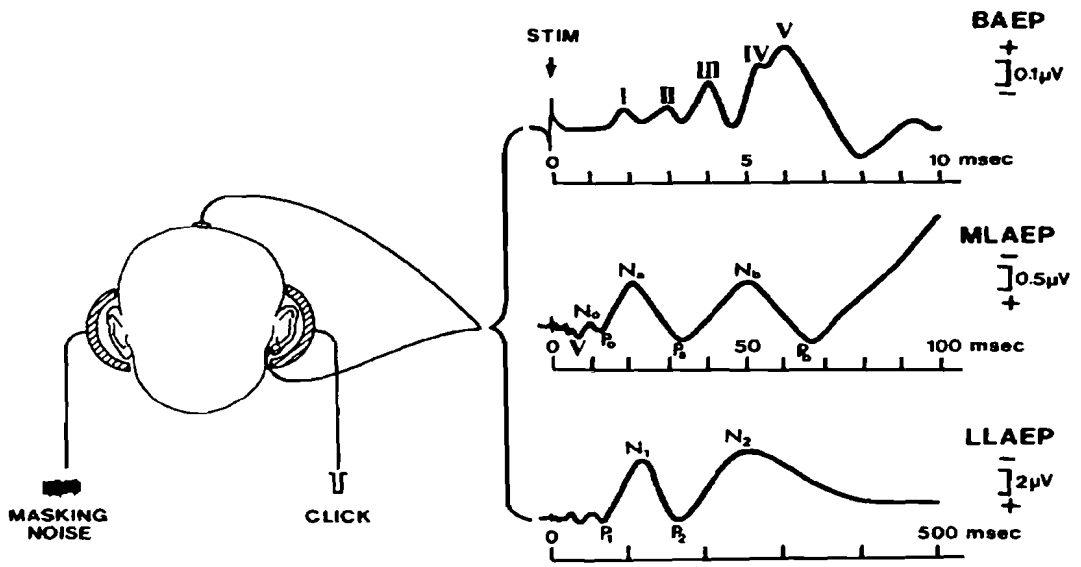
Men kan veel typen EP's onderscheiden afhankelijk van stimulatie- en opname methoden. Enkele neurofysiologische metingen waarin wij geïnteresseerd zijn, zijn van het type *auditory evoked potential* (AEP, auditieve modaliteit), *visual evoked potentials* (VEP, visuele modaliteit) en *somatosensory evoked potentials* (SEP, somato-sensorische modaliteit). In paragraaf 2.2 worden enkele voorbeelden beschreven.

Bij de sectie Psychonomie worden tijdens de experimenten verschillende signalen gemeten: EEG (hersen activiteit), ECG (hartslag), EMG (spier-activiteit), ademhaling en gedragsmaten (reactie tijden en kracht gemeten met drukopnemers). Een aantal van deze signalen worden in diverse combinaties gemeten. Als voorbeeld zullen in paragraaf 2.3 enkele veel voorkomende typen experimenten van de sectie Psychonomie (Readiness Potential, Cognitive Negative Variation en Stimulus Preceding Negativity) verder worden uitgewerkt.

2.2 Voorbeelden van evoked potentials

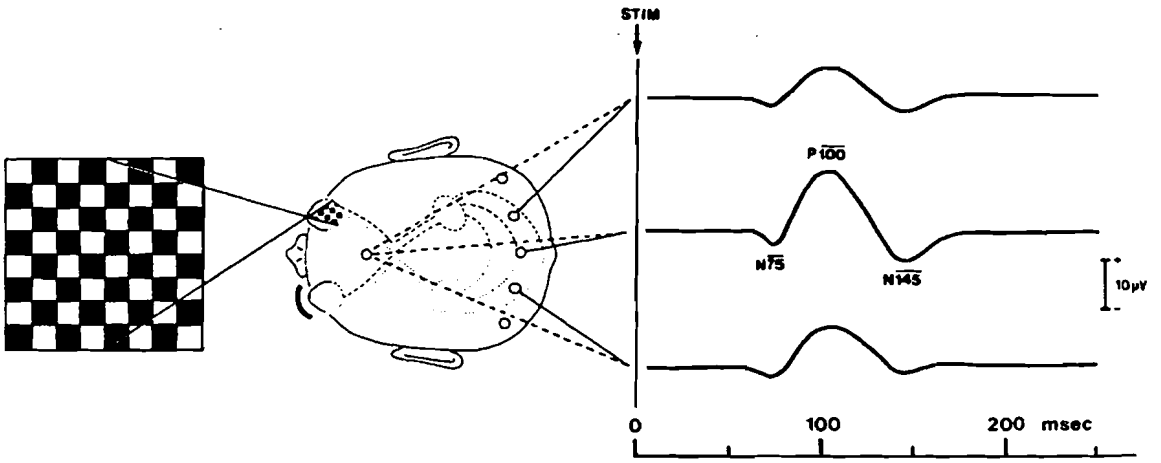
Deze paragraaf zal een aantal voorbeelden laten zien van EP's. Het is niet de bedoeling van dit verslag om deze uitgebreid uit te leggen, maar ter illustratie.

Bij *auditory evoked potential (AEP)* experimenten krijgt de proefpersoon een auditieve stimulus (bijvoorbeeld via een koptelefoon, waar aan een zijde toonpulsen gegenereerd worden). De patiënt is vaak in een ontspannen houding of in lichte slaap. De EP wordt gemeten tussen de vertex en de oorlel. Figuur 2.1 laat een voorbeeld van zo'n experiment zien. Het signaal (een aantal pieken) kan met 3 verschillende opname snelheden en versterkingsfactoren gemeten worden. *Brain stem AEP* (BAEP) is een meting met een korte tijd basis en een hoge versterking. Nemen we op met een grotere tijd basis en een iets kleinere versterking spreken we over *middle latency AEP* (MLAEP). Bij een grote tijd basis en een nog lagere versterkingsfactor spreken we over *long-latency AEP* (LLAEP). Signalen, die ook wel event related potentials (ERP's) genoemd worden.



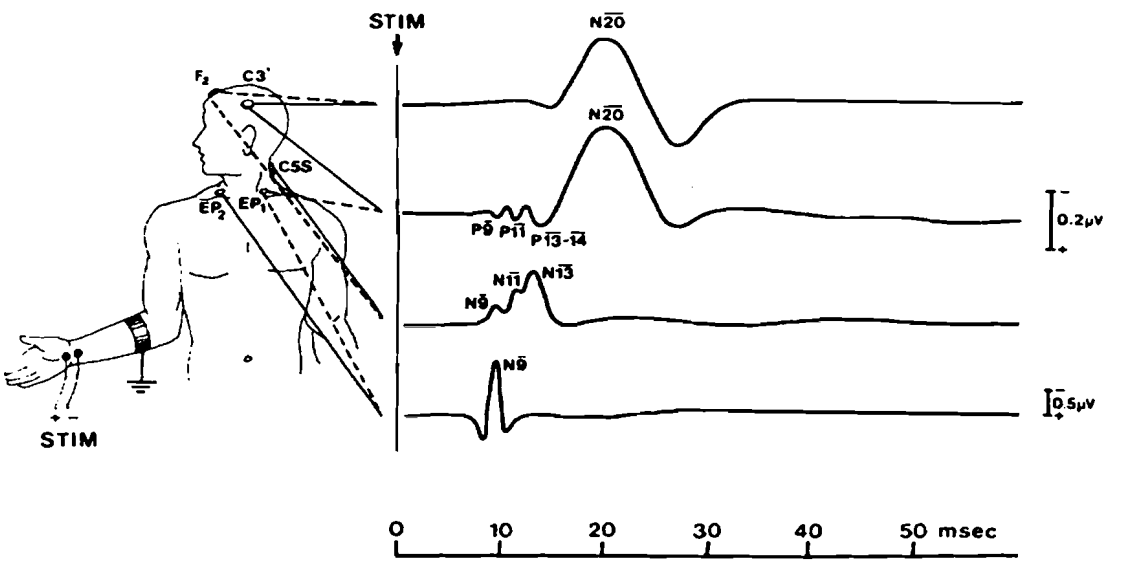
Figuur 2.1 Voorbeeld van een AEP

Bij *visual evoked potential (VEP)* experimenten krijgt de proefpersoon een visuele stimulus. Zonder al te diep in te gaan op de soorten VEP's laat figuur 2.2 een schematisch diagram zien van een VEP bij een monoculaire "full-field" stimulatie (ref. [4]).



Figuur 2.2 Voorbeeld van een VEP

Somatosensory evoked potential (SEP) worden gekenmerkt door de plaats van de stimulus en de opname elektroden op het lichaam. De meeste SEP's worden geproduceerd door de proefpersoon's zenuw in arm of been een kleine elektrische stimulatie te geven. Figuur 2.3 geeft een voorbeeld van een normale SEP, waarbij de stimulus op de arm gegeven wordt. De gemeten signalen stellen de verschillende soorten SEP's voor welke bij dit experiment gemeten worden. Dit verslag zal hier niet verder op in gaan.



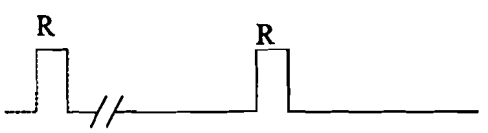
Figuur 2.3 Voorbeeld van een SEP met een arm stimulus.

Voor een uitgebreidere uitleg van EP's verwijs ik u naar Spehlmann's 'evoked potential primer' (ref. [4]).

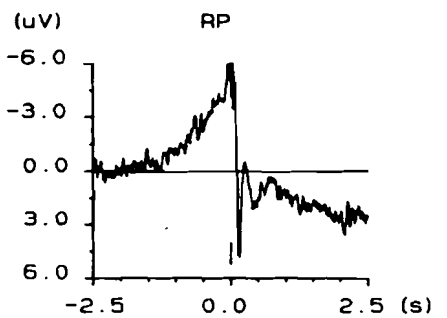
2.3 Enkele experimenten binnen de sectie Psychonomie

In deze paragraaf wordt een indicatie gegeven van de opbouw van enkele experimenten die aan de sectie Psychonomie uitgevoerd worden.

Bij 'readiness potential' is men geïnteresseerd in de hersenactiviteit die plaats vindt voordat iemand een actie uitvoert. Tijdens dit experiment wordt de proefpersoon geïnstrueerd om op 'vrijwillige basis' een beweging uit te voeren (vaak een knop indrukken).
De potentiaal veranderingen voorafgaand aan de actie (druk op de knop) worden geregistreerd. In figuur 2.4a is een grafische weergave gegeven van dit experiment. De puls stelt de reactie van de proefpersoon voor. De signalen worden bemonsterd en vanaf 2,5 seconden voor tot 2,5 seconden na de reactie opgeslagen.
In figuur 2.4b is een voorbeeld gegeven van zo'n gemeten potentiaal verandering. Men kan duidelijk waarnemen dat de hersenen een negatieve potentiaal-verandering genereren voordat de actie (tijdstip 0.0) door de proefpersoon uitgevoerd wordt.

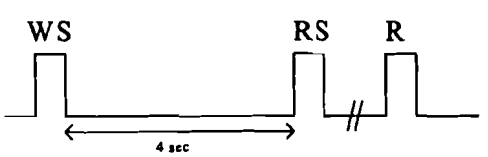


Figuur 2.4a Readiness Potential experiment

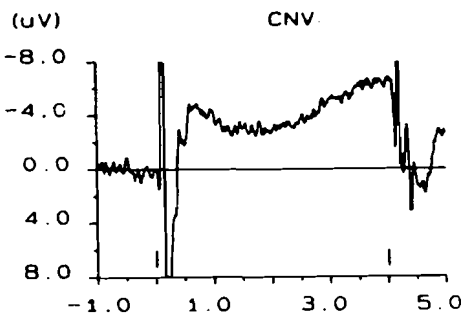


Figuur 2.4b Een gemeten RP-signaal

Bij 'contingente negatieve variatie' experimenten is men tevens geïnteresseerd in de mate waarin anticipatie door de hersenen verwerkt wordt. De proefpersoon krijgt eerst een waarschuwingssignaal (bijvoorbeeld led) en na 4 seconden krijgt hij het reactie signaal (toon) waarop hij met een actie (knop indrukken) moet reageren. De proefpersoon zal in die 4 seconden zich voorbereiden op het geven van een reactie. Dit wordt in figuur 2.5a grafisch weergegeven. Figuur 2.5b laat een voorbeeld zien van een registratie van een CNV experiment.



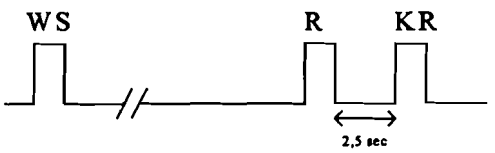
Figuur 2.5a Contingente negatieve variatie experiment



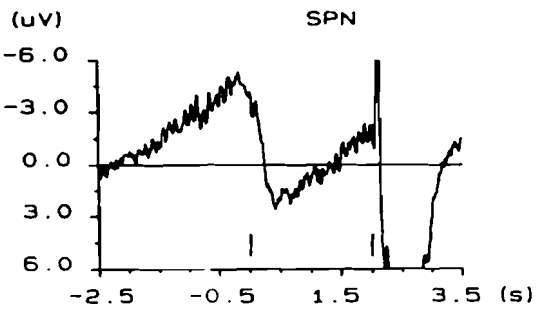
Figuur 2.5b Een gemeten CNV

De registratie is hier eenvoudiger dan bij RP experimenten omdat sturing vastligt. Bemonsterde data vanaf een seconde voor dat het waarschuwingssignaal gegeven wordt tot een seconde na het respons signaal worden opgeslagen.

Nadeel van een CNV experiment is echter dat het anticiperen op de respons stimulus en het voorbereiden van de respons in de tijd samen vallen.
'Stimulus preceading negativity'(SPN) experimenten hebben dit nadeel niet. In deze experimenten krijgt de proefpersoon een waarschuwingssignaal waarna hij naar eigen inzicht een actie moet uitvoeren (knop drukken). De proefpersoon wordt verteld zelf een schatting te maken van 3 seconden en dan de actie uit te voeren. Twee seconden na de reactie van de proefpersoon krijgt deze een feedback. De proefpersoon wordt medegedeeld dat hij/zij te laat, goed of te snel gereageerd heeft. Figuur 2.6a geeft dit grafisch weer en figuur 2.6b laat een voorbeeld zien van een registratie uit een SPN experiment. Men kan duidelijk de negatieve potentiaal-verandering voor de actie waarnemen, maar ook de anticipatie op de feedback. De registratie begint vanaf een seconde vóór het geven van een waarschuwingssignaal tot 6 seconden ná dit signaal.



Figuur 2.6a Stimulus Preceading Negativity experiment



Figuur 2.6b Een gemeten SPN-signaal

2.4 Encodering van de experimenten

De fysiologische signalen worden 'bemonsterd' zodat deze digitaal opgeslagen kunnen worden. Dit vereenvoudigt verdere verwerking van de data met behulp van computers.
De data die bij een experiment geaccumuleerd worden zullen in een file opgeslagen worden. Naast de data zelf is het om administratieve en voor verwerkings-redenen nodig extra informatie aan deze data toe te voegen. Men kan zich voorstellen dat bij een experiment met 20 proefpersonen men graag een onderscheid wil kunnen maken tussen de verschillende files. Deze extra informatie wordt 'meegeleverd' door middel van het toevoegen van een header aan de 'ruwe' data. De naam header zegt het in principe al, deze extra informatie wordt aan het begin van de file geschreven. Bij de sectie Psychonomie maakt men gebruik van het data formaat Fysian. Dit data formaat is specifiek voor en door de sectie Psychonomie ontworpen, voor het opslaan van trialsgewijze data. De precieze opbouw van dit formaat valt buiten het bestek van dit verslag. De beschrijving is eventueel in te zien bij de sectie Psychonomie aan

de KUB (documentatie, zie ref. [2]). Wel zullen enkele nadelen en voordelen van dit formaat genoemd worden.

De extra informatie die meegeleverd wordt aan een datafile in het Fysian formaat zijn variabelen zoals: aantal kanalen, aantal samples in elk kanaal, soort kanalen (EEG, EOG, EMG enz.), bemonsteringsfrequentie, heeft een calibratie plaats gevonden, datum, informatie (tekst), met welke hand/voet heeft men gereageerd, de reactie tijd(en), versterkingsfactoren van de opnemers, aantal stimuli, tijden van de stimuli e.d.
Deze informatie is opgeslagen in de header van de Fysian datafile.

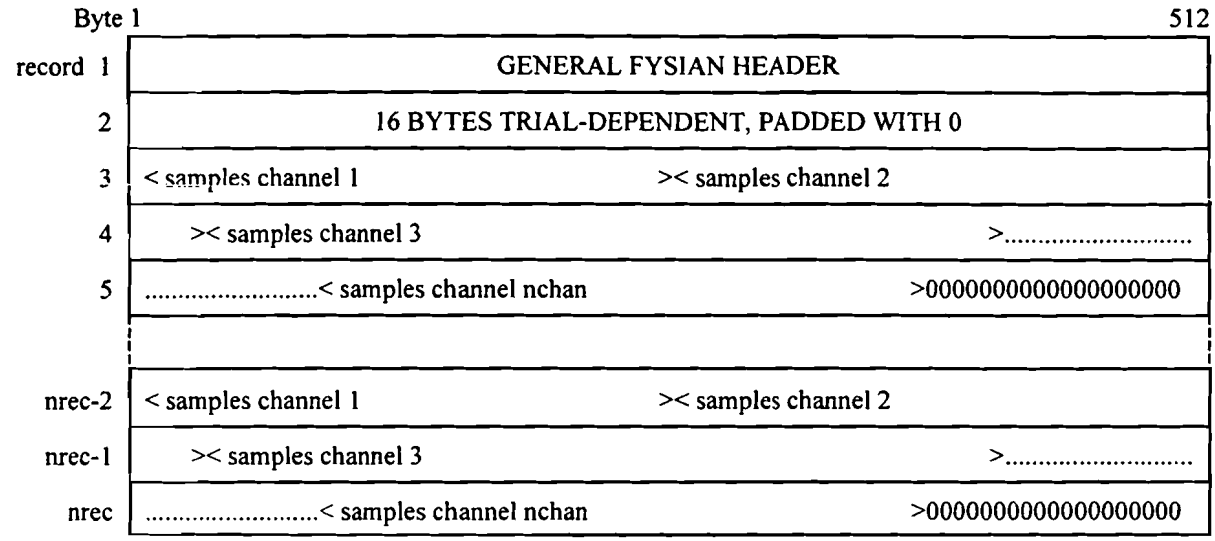
Het registreren van fysiologische signalen kan gepaard gaan met veel storingen. De gemeten voltages zijn vaak in de orde van enkele micro volts en de (materiaal-) omgevingsruis en de achtergrond EEG zijn van dezelfde orde. Een manier om de zogenaamde signaal-ruis verhouding te verbeteren is de experimenten zoals deze in de vorige paragraaf beschreven zijn meermaals te herhalen. Eén keer een experiment uitvoeren noemen we een "trial". Door deze trials dan te middelen kan de ruis er uit gemiddeld worden.
Tijdens het experiment kunnen er trials zijn waar artefacten optreden. Voor zover hier geen correctie kan worden uitgevoerd moeten deze trials weggelaten worden.
Dit verklaart waarom men aan de KUB met veel trials werkt. Waarom men de data trialsgewijs opslaat heeft te maken met de beperkingen van de hard-ware. De acquisitie van het data geschiedt met behulp van PC's. Deze hadden ten tijde van de ontwikkeling van Fysian een gering geheugen (klein buffer), waren langzaam met schrijven en lezen van de data en beschikten over relatief weinig opslag capaciteit. De data van een trial moet dus eerst 'weggeschreven' worden voordat men kon beginnen met de 'acquisitie' van de volgende trial en om ruimte te besparen worden tussenliggende intervallen niet opgeslagen.

Om later te kunnen beschikken over de individuele trials moet de trial-informatie om deze te karakteriseren mee opgeslagen kunnen worden. In Fysian gebeurt dit door in de header extra informatie toe te voegen zoals: aantal trials, welke trials zijn calibratie of eog trials, reactie tijden per trial. (Calibratie trials zijn trials die gebruikt worden om de versterkingsfactoren tussen de elektroden te bepalen en eog trials zijn trials die gebruikt worden om de effecten van artefact uit de gemeten signalen te kunnen verwijderen).

Hebben we een data file verwerkt door bijvoorbeeld alle trials te middelen, dan blijft er een gemiddelde trial over. Bij deze trial hebben we ook nog verdere afgeleide informatie zoals het aantal trials in het gemiddelde en eventueel hoeveel proefpersonen zijn meegenomen bij het middelen en gemiddelde reactie tijd.
Ook deze extra informatie is opgenomen in de header van het Fysian data formaat.

Naast het opslaan van de header moet men afspreken hoe de data opgeslagen worden.
Alle gegevens per kanaal, per trial of per sample opslaan?
Bij Fysian worden de data in trials opgeslagen. Van de eerste trial worden eerst alle samples van kanaal 1 dan het volgende kanaal, enz. opgeslagen voordat men aan de volgende trial begint. Elke trial begint bij een nieuw record (512 bytes). Als een trial niet past in een geheel

aantal records, wordt het record opgevuld met nullen.
Figuur 2.4 geeft een voorbeeld van de opbouw van een data file in het Fysian formaat.
We hebben nu een datafile met de experimentele data waarin alle informatie zit om deze te kunnen verwerken.



Figuur 2.4 De opbouw van een Fysian data file. In dit voorbeeld is een trial 3 records groot en bevat het nchan kanalen. De file is nrec*512 bytes groot en bestaat uit (nrec-2)/3 trials.

2.5 Fysian versus EDF, oud formaat versus nieuw formaat

Zoals in de vorige paragraaf vermeld is heet het data formaat dat men gebruikt binnen de sectie Psychonomie Fysian. Dit formaat is ontwikkeld door de sectie Psychonomie en is specifiek toegespitst op het soort experimenten die door deze sectie uitgevoerd worden. Fysian heeft als voordelen dat het uiterst geschikt is voor trialsgewijze data opslag. De verwerkings-software voor Fysian data files is geschreven in Fortran op VAX computersystemen. De error-afhandeling van de software voor dit type data formaat is iets waar men bij de sectie Psychonomie zeer tevreden mee is en zeker als voordeel te noemen is. Edoch heeft Fysian enkele nadelen waar men steeds meer mee te maken krijgt. Zo is het aantal kanalen dat opgeslagen kan worden beperkt tot 32. Men voert tegenwoordig experimenten uit waarbij dit aantal overschreden wordt.
Een ander nadeel is het feit dat alle kanalen dezelfde bemonsteringsfrequentie moeten hebben. Men kan zich voorstellen dat tijdens een experiment ook de hartslag van de proefpersoon geregistreerd wordt. Deze metingen moeten vaak om een goede resolutie te krijgen op een hogere frequentie bemonsterd worden dan de EEG signalen (bijvoorbeeld EEG op 128Hz en

ECG op 1000 Hz), waardoor alle signalen met de hoogste voorkomende bemonsteringsfrequentie opgeslagen moeten worden. Dit kost veel opslag ruimte.

In de toekomst wil men aan de sectie Psychonomie steeds meer continu gaan registreren, aangezien men tegenwoordig over snelle computers met meer geheugen en diskruimte beschikt en het 'buffer probleem' hier vervalt.

De KUB gaat van mainframe veranderen. Men wil van VAX/VMS naar UNIX systemen overstappen. De sectie Psychonomie is hierdoor genoodzaakt naar een ander platform te gaan. Men heeft gekozen voor de PC. Deze keuze brengt enkele nadelen met zich mee.
De software in Fortran voor de VAX/VMS machines kan niet zomaar vertaald worden naar MS Fortran voor de PC, omdat de bestaande software op de VAX/VMS systemen met behulp van systeem afhankelijke routines werkt. Die routines zouden vervangen moeten worden. Verder zou dan Fysian het file-formaat blijven, met zijn beperkingen van één frequentie voor alle signalen en alleen trialsgewijze data opslag. Dus is met de keuze voor een nieuw platform ook gekozen voor een andere programeer taal (ANSI C) en een ander file-formaat.

In samenwerking met de sectie EME aan de TUE is voor het European Data Format gekozen. Dit formaat is ontwikkeld om uitwisseling van data van slaap registraties tussen verschillende instituten eenvoudiger te maken. Het is ontworpen voor het opslaan van continue data.
Dit formaat heeft als voordeel dat het aantal kanalen niet beperkt is. Tevens kunnen de kanalen met verschillende bemonsteringsfrequenties opgeslagen worden.
Een nadeel van EDF kan zijn dat het formaat in zijn oorspronkelijke opzet niet geschikt is voor trialsgewijze data opslag. Een oplossing hierop zal in het volgende hoofdstuk behandeld worden.

3. European Data Format

3.1 Inleiding

Een grote hoeveelheid en verscheidenheid van computer gebaseerde systemen worden gebruikt voor het analyseren en archiveren van polygrafische opnames. De mate van verscheidenheid tussen deze systemen is een mate van verschil tussen de verschillende formaten welke gebruikt worden voor het digitaal opslaan van de polygrafische signalen. Daarbij heeft elke instituut een andere benadering voor het handmatig of gecomputeriseerd analyseren van hun data. Dit alles maakt vergelijking van de verschillende resultaten erg ingewikkeld.
Om deze redenen heeft de Task Group on Signal Analysis (geleid door B. Kemp, zie [1]) in een Europees onderzoeks-programma een standaard ontwikkeld (genaamd European Data Format) om het opslaan en de uitwisseling van data eenvoudiger te maken. Dit geeft als voordeel dat verschillende analyse methoden van verschillende instituten eenvoudiger met elkaar vergeleken kunnen worden. Ook is het mogelijk een gezamenlijk archief te maken waar verschillende instituten hun gegevens kunnen bewaren. Het maakt toegang tot grotere hoeveelheid informatie mogelijk voor de instituten, die ze zelf moeilijker zouden kunnen creëren.
Dit hoofdstuk zal in het kort de EDF standaard en de uitbreiding van de standaard beschrijven.

3.2 De opbouw van EDF

Zoals eerder vermeld is EDF een data formaat dat ontwikkeld is voor het opslaan van digitale polygrafische signalen.
Een datafile van het EDF formaat bestaat uit een header record gevolgd door de (digitale) data in verschillende datarecords. Het header record moet als ASCII geïnterpreteerd worden terwijl de data als 2-byte integers is gedefinieerd..
Het variabele header record identificeert de proefpersoon en bevat de technische specificaties van de opgenomen signalen. De eerste 256 bytes van het header record specificeren de versie van het data formaat, lokale proefpersoon identificatie, identificatie van de opname en meer statische gegevens (zie figuur 3.1).
Dit wordt gevolgd door de specificatie van alle opgenomen kanalen. De beschikbare opslagruimte per kanaal is 256 bytes, d.w.z de header neemt 256 bytes aan grootte toe per extra kanaal. Daar elk kanaal apart gekarakteriseerd wordt is het mogelijk verschillende bemonsteringsfrequenties te gebruiken. Dit is een van de voordelen van het EDF. De variabelen van de kanalen worden sequentieel voor alle kanalen opgeslagen. Dat wil zeggen dat eerst de eerste variabele van alle kanalen (label) wordt opgeslagen daarna de volgende variabele van alle kanalen (transducertype) enz. en niet zoals men misschien zou verwachten, alle variabelen van kanaal 1 gevolgd door alle variabelen van kanaal 2 enz.

HEADER RECORD

8 ascii	:	version of this data format (0)
80 ascii	:	local patient identification
80 ascii	:	local record identification
8 ascii	:	start date of recording (dd.mm.yy)
8 ascii	:	start time of recording (hh.mm.ss)
8 ascii	:	number of bytes in header record
44 ascii	:	reserved *(used by the Tilburg University for local variables)*
8 ascii	:	number of data records (-1 if unknown)
8 ascii	:	duration of a data record, in seconds
4 ascii	:	number of signals *(ns)* in data record
ns * 16 ascii	:	ns * label (e.g. EEG FpzCz or Body temp)
ns * 80 ascii	:	ns * transducer type (e.g. AgAgCl electrode)
ns * 8 ascii	:	ns * physical dimension (e.g. μV or °C)
ns * 8 ascii	:	ns * physical minimum (e.g. -500 or 34)
ns * 8 ascii	:	ns * physical maximum (e.g. 500 or 40)
ns * 8 ascii	:	ns * digital minimum DIGMIN (e.g. -2048)
ns * 8 ascii	:	ns * digital maximum DIGMAX (e.g. 2048)
ns * 80 ascii	:	ns * prefiltering PREFILTER (e.g. HP:0.1Hz)
ns * 8 ascii	:	ns * *nr* of samples in each data record
ns * 32 ascii	:	ns * reserved *(used by the Tilburg University for local variables)*

NB: De variabelen van de kanalen worden sequentieel opgeslagen d.w.z eerst alle labels van de verschillende kanalen vervolgens de transducer typen, enz.

DATA RECORD

nr of samples[1] * integer : first signal in the data record
nr of samples[2] * integer: second signal in data record
....
nr of samples[ns] * integer : last signal

Figuur 3.1 Gedetailleerd digitaal formaat van het header record (bovenste blok, alleen ASCII) en data record (onderste blok, alleen 2-byte integers). Elk kanaal van het aantal (ns) kanalen is gespecificeerd in de header.

Figuur 3.2 laat een voorbeeld zien van de opbouw van een header record. Dit voorbeeld is ontleend aan het artikel van Bob Kemp et al. (ref. [1]).

0	Free local patient identification: take care of privacy regulations !					
		Free local recording identification.				
					16.09.87	20.35
.00	768	Reserved				2
880	30	2	EEG FpzCz	Body temperature		Ag-Ag Cl
cup electrodes						
	Rectal thermistor					
		uV	Degree C	-440	34.4	
510	40.2	-2048	-2048	2047	2047	Time cons
tant 1s, First order lowpass at 75 Hz						
	DC to 0.1Hz (first-order)					
			15000	3	Reserved	
		Reserved				

Figuur 3.2
Header record van een 24 uurs EEG en lichaamstemperatuur opname, respectievelijk bemonsterd op 500 en 0.1 Hz. De offsets van de EEG en lichaamstemperatuur zijn 35μV en 37.3°C, terwijl de versterkingsfactoren 4.31 /μV en 706.2 /°C zijn. In dit voorbeeld bevat elke 30 seconden lange data record 15000 samples van de EEG gevolgd door 3 samples van de lichaamstemperatuur.

3.3 EDF en trialsgewijze dataopslag

Waarom is EDF niet geschikt voor trials?

EDF is voornamelijk ontwikkeld voor het uitwisselen van slaapregistraties. Dit zijn experimenten waarbij men normaal niet met trials werkt. Het begin van de data is het begin van het meting en het einde van de data is het einde van het meting.
In de EDF header ontbreekt ruimte om trial-informatie en variabelen die voor elke trial veranderen (bijvoorbeeld hand-voet codes, rejection-codes) op te slaan. De reserved fields zijn te klein voor deze trial-informatie.
Eenvoudig gezegd, men kan de informatie over de specifieke opbouw (in tijd) van de data niet met een standaard EDF data file mee geven.

Waarom dan toch EDF als data formaat gebruiken?

Enkele voordelen zijn al genoemd. Verschillende kanaal frequenties, en de mogelijkheid voor continue data opslag zijn een paar van deze voordelen.
Een ander pluspunt van EDF is dat het los staat van een of ander bedrijf en dit formaat door steeds meer instituten en bedrijven gebruikt wordt. Hierdoor is het mogelijk de software van derden te gebruiken voor verwerking van data. Als bijvoorbeeld een instituut een goed 'display'-programma of een analyse-programma heeft en dit afstaat aan anderen, is het mogelijk deze software te gebruiken en hoeft men de software niet zelf ontwikkelen. Naast tijd spaart dit natuurlijk ook mankracht, kennis (deze is nu immers van buiten gehaald) en niet te vergeten geld.

We willen dus toch EDF gebruiken en deze geschikt maken voor het verwerken van data welke is opgebouwd uit verschillende trials.

Er zijn verscheidene manieren om dit op te lossen. Enkele geanalyseerde opties worden hier opgesomd:

1. Het toevoegen van een extra file met de extra informatie, die samen met de EDF data-file wordt opgeslagen. Dit heeft als nadeel dat de administratie ingewikkelder wordt wat aanleiding geeft tot het maken van onnodige fouten en verlies van data.
2. Het aanpassen van de header, door deze bijvoorbeeld nieuwe velden te geven. Als we dit doen wijken we af van de standaard. Dit was nu net een van de redenen waarom we EDF willen gebruiken. Software van derden is dan niet meer bruikbaar.
3. De extra informatie achteraan de data file toevoegen. Dit is niet wenselijk, omdat een van de opties in de header van een EDF data file is dat het aantal data records onbekend kan zijn. Dit houdt in dat de file lengte niet bekend is (kan niet uitgerekend worden) en men dus deze extra informatie niet kan lokaliseren.

Er is gekozen voor een aanpak waarbij de standaard header niet veranderd wordt. De nieuwe header voldoet geheel aan de standaard zoals deze beschreven staat in literatuur [1].

3.4 Extensie op EDF

Een van de sterke punten van EDF is dat het aantal kanalen niet beperkt is. De gebruiker kan zelf bepalen hoeveel kanalen hij voor zijn experiment gebruikt.
Door gebruik te maken van deze optie is het mogelijk om extra informatie, welke de opbouw van onze data beschrijft, eenvoudig aan een EDF data file toe te voegen.
We hebben besloten de informatie die we nodig hebben bij een bepaald experiment in twee extra kanalen te coderen.
Dit heeft tot voordeel dat men niet van de standaard afwijkt. Standaard software kan deze 'extra' kanalen negeren of interpreteren als 'gemeten data'. De gebruiker weet (moet weten) dat deze kanalen geen gemeten data representeren en kan ze meestal negeren. Software die wel gebruik maakt van deze extra informatie, kan de kanalen lezen en de informatie hieruit decoderen.
Een nadeel van het toevoegen van extra kanalen kan zijn dat de gebruikte diskruimte nogal kan toenemen. Dit mag tegenwoordig echter niet zo'n groot probleem meer zijn, omdat de opslagmedia aanzienlijk meer data kunnen bevatten dan voorheen.

Er zijn twee soorten extra informatie die aan een EDF data file toegevoegd kunnen worden. Tijdsafhankelijke informatie zoals: wanneer begint/eindigt een trial, wanneer vindt een stimulus of reactie plaats en afgeleide informatie zoals: gegevens die gegenereerd zijn door bewerkingen op de 'ruwe' data (bijvoorbeeld aantal trials in gemiddelde, gemiddelde reactietijd), extra gebruikersinformatie.

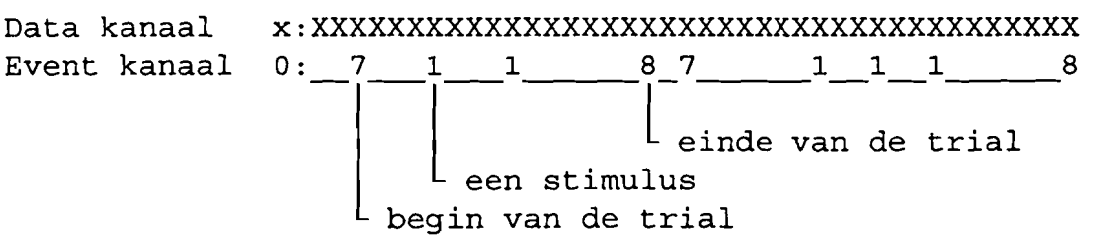
3.4.1 Het Event kanaal

Het belangrijkste kanaal dat we willen toevoegen is het *Event kanaal*. Het Event kanaal is een real-time kanaal dat parallel aan alle kanalen loopt. Men kan het beschouwen als een tijdas waar de gebeurtenissen in de tijd zijn gecodeerd. Als we het kanaal opslaan met een frequentie van bijvoorbeeld 1000 Hz, dan stelt elk sample een gebeurtenis in de tijd voor met een precisie van 1 milliseconde.
Alle tijdsafhankelijke gebeurtenissen tijdens een experiment kunnen in dit kanaal gecodeerd worden. De codes kunnen gebruikt worden om te markeren wanneer bijvoorbeeld een calibratie trial begint, een stimulus plaats vindt of een andere willekeurige tijdsafhankelijke gebeurtenis.

Door deze definitie kunnen de gemeten kanalen ook worden opgeslagen als continue data. Het Event kanaal bevat de extra tijdsinformatie nodig voor het beschrijven van de opbouw van de data.
Dit maakt het tevens mogelijk in de toekomst de 'trials' achter elkaar te registreren en continue op te nemen en op te slaan. Parallel aan het registreren wordt het Event kanaal opgebouwd en met de data in de file opgeslagen. Het Event kanaal wordt als binair 'data' kanaal mee opgeslagen waardoor we niet afwijken van de standaard zoals deze oorspronkelijk gedefinieerd is.

Het eenvoudige voorbeeld in figuur 3.3 kan deze extensie duidelijker maken.

Code voor het begin van een trial is 7, voor het einde van een trial 8 en voor een stimulus 1. (x=gemeten data)



Figuur 3.3 Voorbeeld van een Event kanaal.

We kunnen precies bepalen waar de data van een bepaalde trial begint/eindigt, wanneer er een stimulus heeft plaatsgevonden enz. door het Event kanaal te bekijken.

De data in de kanalen bestaan in het algemeen uit 2-byte integers. Met 2 bytes kunnen we 65535 codes en dus 65535 verschillende gebeurtenissen in het Event kanaal coderen.

Een onderverdeling kan worden aangebracht door de code in 2 individuele enkelvoudige bytes te splitsen. Het eerste byte representeert de 'main event code' en het tweede byte representeert een 'sub code'. We hebben nu 256 main codes en 256 sub codes welke in totaal weer de

65535 codes geven.
Het encoderen van de informatie op de hier beschreven manier zorgt ervoor dat de langste zoekroutine 256+256 vergelijkingen nodig heeft, terwijl het willekeurig toekennen van unieke codes zorgt voor een hoeveelheid vergelijkingen lineair aan het aantal codes met een maximum van 65535.
Dit wordt in deze vorm toegepast voor de experimenten op de KUB en zal duidelijk worden aan de hand van de op de sectie Psychonomie toegepaste event-codering die in paragraaf 7.1 besproken wordt.

3.4.1.1 Gelijktijdige events

Wat als nu meer dan één gebeurtenis (event) op het zelfde moment plaats vinden?
Men kan alle mogelijke combinaties van gebeurtenissen een nieuwe code geven. Deze oplossing zorgt echter voor veel onnodige codes.
Wij lossen dit op door het invoeren van een '*multiple event* code'. Dit is een gereserveerde code die de gebruiker vertelt dat er meerdere gebeurtenissen hebben plaats gevonden op een tijd-moment. De sub code (het tweede byte) wordt gebruikt om het *aantal* gelijktijdige gebeurtenissen te coderen.
Wanneer men nu een '*multiple event* code' tegenkomt in het Event kanaal kan men eenvoudig door het decoderen van de sub code het aantal gebeurtenissen bepalen.
Aangezien niet elk moment een gebeurtenis plaatsvindt, zal het grootste deel van het Event kanaal bestaan uit 'niet gebruikte' ruimte. We kunnen deze ruimte gebruiken om de codes van de gebeurtenissen die tijdens een multiple event hebben plaats gevonden in op te slaan.
Een *multiple event* code wordt dus gevolgd door de codes van de gebeurtenissen die hebben plaats gevonden. Mocht er nu een of meer gebeurtenissen plaats vinden voordat we ruimte hebben gehad om alle codes van een multiple event op te slaan zal deze met behulp van dezelfde *multiple event* code weergegeven worden. De gebruiker weet nu dat deze code niet toebehoort aan het vorige multiple event maar een nieuwe gebeurtenis voorstelt. De direct daarop volgende codes behoren altijd bij het *laatste multiple event*. Wanneer alle codes van het laatste multiple event opgeslagen zijn worden weer de codes van het eerdere multiple event weggeschreven.
Op het moment dat we alle codes die bij de multiple event(s) horen hebben opgeslagen hoeven we de *multiple event* code niet meer te gebruiken om enkelvoudige gebeurtenissen te coderen.

Deze definitie maakt het toepassen van recursie in de software voor het lezen en schrijven van het event kanaal eenvoudiger.

Het voorbeeld in tabel 3.1 kan dit verduidelijken.
De codes zijn opgebouwd uit 2 bytes: MMxx (MM=main code en xx= subcode)
De samplenummers zijn gelijk aan een moment in de tijd.

Tabel 3.1 Voorbeeld van de inhoud van een Event kanaal.

Sample nr.	Code	Toelichting
00	0000	Code voor event op moment 00.
01	FF03	FFxx= multiple event, op dit moment 3 events.
02	0001	Code voor eerste event op moment 01.
03	FF02	Voordat we alle events van moment 01 konden wegschrijven heeft weer een multiple event plaats gevonden, nu met 2 events.
04	0002	Code voor eerste event op moment 03.
05	0003	Code voor tweede event op moment 03.
06	0004	Code voor tweede event op moment 01.
07	FF01	We hebben nog steeds niet alle events van moment 01 weg kunnen schrijven en er heeft 1 event plaats gevonden. Ook deze moeten we met een multiple event coderen.
08	0005	Code voor het echte event op moment 07.
09	0006	Code voor derde event op moment 01.
10	0007	Alle multiple events zijn gecodeerd, nu kunnen enkele events weer met hun eigen code worden weergegeven.

(De codes 0000 t/m 0007 in dit voorbeeld stellen alleen volgnummers van de events voor)

In hoofdstuk 6 wordt beschreven hoe de codes aan Universiteit in Tilburg gebruikt worden. De identificatie van het kanaal gebeurt door in het variabele veld 'label' van de header de naam van dit kanaal op te slaan. Voor het Event kanaal zal dit "EVENT CHANNEL" zijn.

3.4.2 Het Info kanaal

Naast het Event kanaal willen we soms nog extra informatie toevoegen die niet als binaire data opgeslagen kan worden. Deze informatie slaan we op in een ander kanaal met de naam *Info kanaal*. De interpretatie van dit kanaal wijkt enigszins af van de standaard. In de standaard is beschreven dat alle kanaal data is opgebouwd uit 2 byte integers. Echter wij interpreteren het Info kanaal als ASCII data (1 byte). Dit mag geen problemen geven met algemenere software, daar het Info kanaal door dit soort programmatuur toch niet gebruikt wordt. Deze software zal het kanaal als 'data' interpreteren met een pseudo-bemonsteringsfrequentie die uit het kanaal-informatie deel van de header bepaald kan worden. De bemonsteringsfrequentie van het kanaal is uitgedrukt in samples per seconde en een sample is 2 bytes groot. Elk sample is dus equivalent met 2 ascii waarden. Een Info kanaal met een pseudo-bemonsteringsfrequentie van bijvoorbeeld 100 Hz (aantal samples/data record is dan 100 met een data record duur van 1 seconde), bevat dus 200 ascii waarden per data record. Software die het kanaal wel gebruikt volgens bovenstaande definitie weet dat dit ASCII data is en zal dit kanaal goed interpreteren.

In dit kanaal kunnen we de afgeleide en andere gebruikersafhankelijke data/variabelen opnemen. De inhoud van het Info kanaal is gebruiker afhankelijk en geeft deze de vrijheid om het kanaal voor specifieke gegevens te gebruiken. De gebruiker dient er wel op letten dat wanneer hij/zij de data aan derden wil geven, deze gebruikers op de hoogte moeten zijn

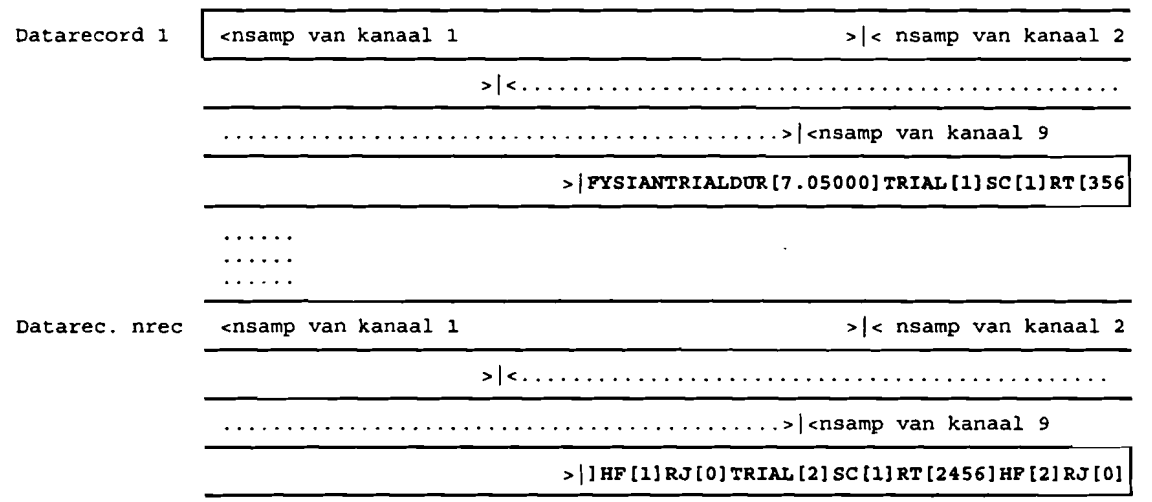
(indien nodig) van de interpretatie van de inhoud van het Info kanaal. Software van derden kan hier over het algemeen niets mee.

Het aanwezig zijn van het Info kanaal sluit het Event kanaal niet uit en visa versa. In beide kanalen kan ook voor een deel dezelfde informatie voorkomen. Bijvoorbeeld het aantal trials dat een data file bevat kan uit het Event kanaal bepaald worden, maar kan ook als extra variabele in het Info kanaal opgeslagen worden.

De identificatie van de kanalen gebeurt door in het variabele veld 'label' van de header de naam van deze kanalen op te slaan. Net zoals het label "EVENT CHANNEL" wordt gebruikt voor identificatie van het Event kanaal wordt "INFO CHANNEL" gebruikt voor identificatie van het Info kanaal.

De inhoud van een Info kanaal kan er bijvoorbeeld als volgt uitzien (figuur 3.4):

(In dit voorbeeld heeft de datafile 10 kanalen, waarvan kanaal 10 het Info kanaal is)



Figuur 3.4 Voorbeeld van de inhoud van een Info kanaal.

De variabelen in figuur 3.4 zijn voorbeelden van variabelen die door de sectie Psychonomie gebruikt worden. Ze zijn specifiek voor een gebruiker (sectie Psychonomie) van de data file. Een andere gebruikersgroep kan dit kanaal naar eigen voorkeur invullen. Om deze data te kunnen interpreteren moet men dus over de informatie beschikken wat een onderzoeker specifiek voor dit experiment nodig had.

3.4.3 Overige data

Naast deze kanalen gebruikt men aan de sectie Psychonomie de "reserved fields" van de header voor het opslaan van enkele specifieke variabelen. Dit wordt gedaan om 'snel' toegang tot deze variabelen mogelijk te maken; deze informatie is als zodanig dus redundant. Een voorbeeld hiervan is het aantal trials in de data file.

Als een file van het ene medium naar het andere getransporteerd wordt kunnen er fouten optreden. Als we te maken hebben met een file waarvan het aantal data records niet bekend is en er treedt een fout op waardoor de file niet in zijn geheel getransporteerd is, kunnen we dit niet met behulp van de header informatie controleren.
Door het toevoegen van een variabele *TR[number of trials]* in het reserved veld (44 ASCII) van de header kunnen we een controle uitvoeren door het aantal trials te bepalen met behulp van het event kanaal (mits aanwezig) en deze te vergelijken met de waarde gegeven door deze variabele.
Een ander voordeel bijvoorbeeld van deze variabele is dat snel is te bepalen uit hoeveel trials de data file bestaat zonder eerst het Event kanaal door te moeten lopen. Aangezien de header in ascii is kan een gebruiker dit met behulp van een editor of met een "header-display" programma bekijken. Nogmaals, dit is dus KUB specifiek. In paragraaf 4.1.2 zijn nog enkele variabelen beschreven die in de header geschreven worden. Deze beschrijving is opgenomen in verband met het omzetten van oude Fysian data files naar het nieuwe EDF formaat[1].

[1] In het vervolg van dit verslag zal het aangepaste EDF formaat 'extended EDF' genoemd worden.

4. Het Fysian naar EDF conversie programma

4.1 Inleiding

Het data formaat is bepaald en we weten hoe we onze data willen opbouwen.
Het eerste onderdeel in het verdere ontwikkel-traject is het ontwikkelen van een conversie programma dat de oude data in het Fysian formaat om kan zetten in het 'extended' EDF formaat.
Ik heb een commandline gebaseerd programma geschreven, genaam FYS2EDF, welke de conversie van Fysian naar extended EDF kan uitvoeren.
Het programma is in standaard C geschreven. Dit is gedaan om platformonafhankelijkeid te creëren. Mocht men aan de sectie Psychonomie over willen gaan naar een UNIX systeem, kan de omzetting op deze vaak snellere machine plaats vinden, door het programma onder UNIX te compileren.
De kennis welke opgedaan is met het schrijven van dit conversie programma is gebruikt bij het ontwikkelen van een basis EDF software bibliotheek (zie hoofdstuk 5).

4.2 Beperkingen van dataconversie

De omzetting van een dataformaat naar een ander formaat brengt vaak enige beperkingen met zich mee. Zeker in het geval van twee data formaten met een fundamenteel verschillende structuur.

4.2.1 De bemonsteringsfrequentie

De Fysian bemonsterings frequentie is in het verleden vastgesteld als een reëel getal en geldt voor alle kanalen. In EDF kan de bemonsterings-frequentie van een kanaal bepaald worden door het aantal samples per datarecord van dat kanaal te delen door de tijdsduur van een datarecord.
Door afronden of aanpassingen van de EDF data record tijds-duur kan de 'nieuwe' bemonsterings-frequentie afwijken van de originele Fysian-frequentie. Dit kan complicaties geven bij het lezen van de data met behulp van het Event kanaal. Stel het Event kanaal heeft een bemonsteringsfrequentie van 1000 Hz, dus een precisie van 1 milliseconde. Als we het sample op de 135ste milliseconde van een bepaald data kanaal willen bepalen, kan dit door de tijd te vermenigvuldigen met de bemonsteringsfrequentie van dat data kanaal. Met een verkeerde bemonsterings- frequente is de kans groot dat de software er een of meer samples naast zit. Dit soort fouten kan zeer desastreus zijn voor verdere bewerking, waar nauwkeurigheid van belang is.
Door in de 'reserved fields' van de kanaal informatie in de header een variabele toe te voegen kunnen we er voor zorgen dat men altijd kan beschikken over de exacte bemonsterings-frequentie. De variabele *SF[real sample frequency]*, welke de echte bemonsteringsfrequentie van de Fysian data beschrijft wordt in het 'reserved field' geschreven als de Fysian bemonsterings-frequentie niet overeenkomt met de berekende frequentie (m.b.v de EDF

header informatie). De bewerkingssoftware kan, wanneer deze variabele in de header voorkomt, de goede bemonsteringsfrequentie hieruit decoderen. Komt de variabele niet voor in de header dan zijn de Fysian en kanaal bemonsteringsfrequenties in EDF aan elkaar gelijk en kan de berekende bemonsteringsfrequentie gebruikt worden.

4.2.2 Specifieke variabelen

Fysian bevat een aantal variabelen die per trial veranderen. Bij de conversie van Fysian naar het extended EDF formaat zijn deze variabelen in het Info kanaal opgenomen. De variabelen zijn worden als volgt vertaald:

- *TRIAL[nummer]* wordt gebruikt om aan te geven dat de variabelen die volgen gelden voor de trial met nummer '*nummer*' (deze variabele is toegevoegd om de trial-afhankelijkheid te kunnen herleiden).
- *SC[code]* staat voor de variabele '*select code*' in Fysian. Deze variabele wordt gebruikt voor het selectief middelen van trials. Voor de codes en hun betekenis verwijs ik u naar documentatie [2] (geldt voor alle hier beschreven variabelen).
- *HF[code]* staat voor de variabele '*hand-foot code*' in Fysian. Deze representeert met welke hand en/of voet er tijdens het experiment gereageerd is.
- *RT[reactietijd in msec]* geeft reactie tijd weer in milliseconden. Deze variabele is bij de aanwezigheid van een event kanaal eigenlijk overbodig, daar deze informatie ook in het event kanaal gecodeerd wordt. Hebben we te maken met een trial-gemiddelde zonder een event kanaal, wordt deze variabele gebruikt voor de gemiddelde reactie tijd.
- *RJ[code]* staat voor de variabele '*rejection code*' in Fysian. Deze wordt gebruikt om bijvoorbeeld aan te geven of een trial goed of fout is.

Veder zijn er nog enkele variabelen in de Fysian header, die aangeven of een data file een 'ruwe' data file, een gemiddelde of een gemiddelde over verschillende proefpersonen is en het aantal trials in de datafile.
Deze variabelen worden opgenomen in het reserved veld van de header informatie (44 ASCII).
De variabelen zijn:

- *TR[aantal trials]* voor een ruwe data file,
- *AV[aantal trials]* (Average) voor een gemiddelde data file,
- *SA[aantal trials]* (Subject Average) voor een gemiddelde van een aantal trial-middelingen van een proefpersonen,
- *GA[aantal trials,aantal proefpersonen]* (Grand Average) voor een gemiddelde over een aantal proef-personen.

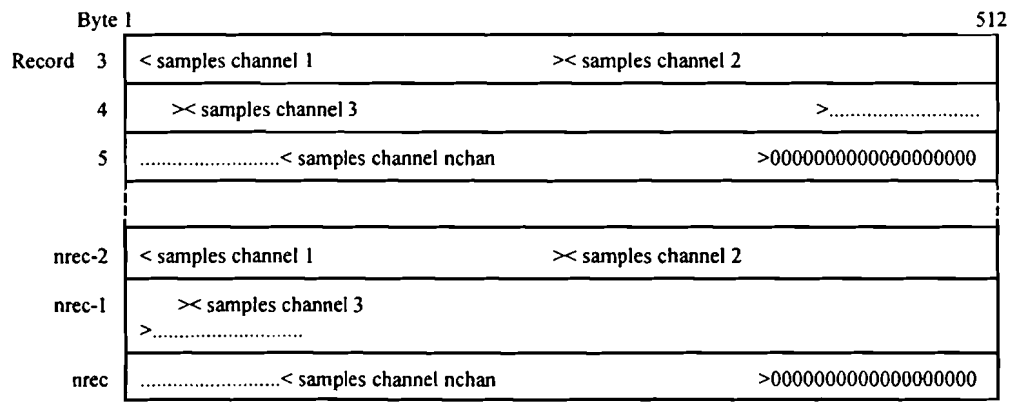
Er zijn nog enkele restricties waar men rekening mee moet houden als men de standaard software van derden wil gebruiken op de omgezette data.
De tijdsduur van een data record kan in onze declaratie (en volgens de standaard, lit [1]) een reëel getal zijn. Gebleken is echter dat sommige programma's alleen positieve gehele getallen accepteren (zie paragraaf 6.1) dat wil zeggen hele seconden en soms alleen 1 seconde.
De tijdsduur van een datarecord is ook belangrijk. Een tijdsduur van 8 seconden kan vaak te

lang zijn voor sommige bewerkingssoftware in verband met geheugenruimte (echter, door opkomst van 32-bit (en meer) besturings-systemen hoeft dit in de toekomst geen probleem meer te zijn). Men moet rekening houden met de software die men wil gebruiken voordat de data omgezet worden. De tijdsduur van het datarecord kan tijdens het converteren aangepast worden tot een gewenste waarde. De trials (die uit een of meer datarecords kunnen bestaan) zullen tot hele records aangevuld worden met nullen. Standaard zal tijdens de conversie de data record lengte gelijk zijn aan de tijds-duur van een Fysian trial.

4.3 De omzetting van Fysian naar EDF

Deze paragraaf behandelt de opbouw van de geconverteerde Fysian data file, zonder al te diep in te gaan in de opbouw van Fysian data files. De 'berekening' van de EDF header en het 'copiëren' van de data zal worden uitgelegd.

Een Fysian data file bestaat uit records van 512 bytes. In de header van een Fysian data file staat beschreven uit hoeveel records de header bestaat. Hieruit kunnen we eenvoudig bepalen waar de data begint.
De data is opgebouwd zoals weergegeven in figuur 4.1. Als alle samples van alle kanalen van een trial in een aantal records geschreven zijn, wordt het record aangevuld met nullen. Elke trial begint dus bij een nieuw record. In de header kan verder nog beschreven staan uit hoeveel records de trials, calibratie- en eogtrials bestaan (specifiek voor de experimenten zoals beschreven op pagina 11).



Figuur 4.1 De opbouw de data in een Fysian data file. In dit voorbeeld is een trial 3 records groot en bevat het nchan kanalen. De file is nrec*512 bytes groot en bestaat uit (nrec-2)/3 trials. De header informatie omvat 2 records.

De opbouw van de EDF header:

EDF HEADER

VERSION	: altijd de string *"0 "*
LOCAL PATIENT ID	: de variabele 'text' van de Fysian header
LOCAL RECORD ID	: de string *"Converted file from : <input filename>"*
START TIME	: *"12:00:00"*
START DATE	: de variabele 'date' van de Fysian header
NUMBER OF BYTES IN HEADER	: 256 + NCHAN (=aantal kanalen) * 256 bytes
RESERVED	: de variabele **"TR[***aantal Fysian trials***]"**, **AV[***aantal Fysian trials***]** of **GA[***aantal trials, aantal proefpersonenen***]**
NUMBER OF DATA RECORDS (NS)	: (Trial duur * aantal trials + Calibratie duur * aantal calibraties + Eog duur) / EDF data record duur
DURATION OF A DATA RECORD	: aantal samples in een trial / Fysian kanaal frequentie
NUMBER OF CHANNELS	: aantal Fysian kanalen (+ event kanaal + info kanaal) (mits nodig !)

EDF KANAAL INFORMATIE

LABEL	: de Fysian variabelen chancode, channame & refcode
TRANSDUCERTYPE	: *"Transducer Unknown"*
PHYSICAL DIMENSION	: *"uV"*,*"ASCII"*, or *"mSEC"*
PHYSICAL MINIMUM	: Fysian variabele factor * DIGITAL MINIMUM
PHYSICAL MAXIMUM	: Fysian variabele factor * DIGITAL MAXIMUM
DIGITAL MINIMUM	: -32768 (als PHYSICAL MINIMUM >-100000)
DIGITAL MAXIMUM	: 32767 (als PHYSICAL MAXIMUM <100000)
PREFILTERING	: *"Not Filtered"*
NUMBER OF SAMPLES	: RECORD DURATION * Fysian kanaal frequentie
RESERVED	: **"SF[***real sample frequency***]"**[2] als NUMBER OF SAMPLES / RECORD DURATION ongelijk is aan de Fysian frequentie.

(Als PHYSICAL MINIMUM en MAXIMUM respectievelijk kleiner dan -100000 of groter dan 100000 zijn zal PHYSICAL MIN. -99999 en PHYSICAL MAX. -99999 gemaakt worden en worden DIGITAL MIN. en MAX. aangepast. Samples en tijdsduur worden naar boven afgerond!)

Als de Fysian data file geen 'gemiddelde' data file is wordt altijd een Event kanaal toegevoegd. De standaard bemonsteringsfrequentie van dit kanaal is 1000 Hz, waardoor een precisie van 1 milliseconde gehaald kan worden. De gebruiker kan deze frequentie veranderen in een frequentie hoger of gelijk aan de hoogst voorkomende Fysian kanaal frequentie.

[2] Zie paragraaf 4.2

Het Event kanaal bevat de codes van tijdsafhankelijke gebeurtenissen tijdens het experiment. Bij de conversie van Fysian naar EDF zijn dit:

-begin van een trial, calibratie- of eog-trial
-einde van een trial, calibratie- of eog-trial
-begin van de baseline (voor baseline correcties)
-einde van de baseline (voor baseline correcties)
-de reactietijden (wordt ook als variabele RT[reaction time] in Info kanaal opgeslagen)
-de stimuli (er zijn in totaal maximaal 5 stimuli per trial in een Fysian data file)

(De laatste 2 tijdsafhankelijke variabelen zijn in Fysian opgegeven in milliseconden t.o.v. het begin van de trial.)
Aangezien het begin van een baseline van een gewone Fysian trial gelijk is aan het begin van de trial, zal in het event kanaal een gewone trial beginnen met een 'multiple event code', gevolgd door een 'begin of trial code' en een 'begin of base line code'.

De encodering van de tijdsafhankelijke informatie die aan de sectie Psychonomie gebruikt wordt voor het converteren van Fysian data files naar EDF data files is weergegeven in tabel 4.1.

Tabel 4.1 Codes (Hex) voor de gebeurtenissen:

Event	Main Code	Extra informatie over event	Sub Code
None	00	-	00
Begin of Trial	01	Normal trial Calibration trial EOG trial	01 02 03
End of trial	02	Normal trial Calibration trial EOG trial	01 02 03
Begin of baseline	03	voor alle kanalen	FF
End of baseline	04	voor alle kanalen	FF
Stimulus on	05	nummer van de stimulus	xx
Reaction on	07	*hand-foot* code uit Fysian	xx
Multiple Events	FF	aantal events	xx

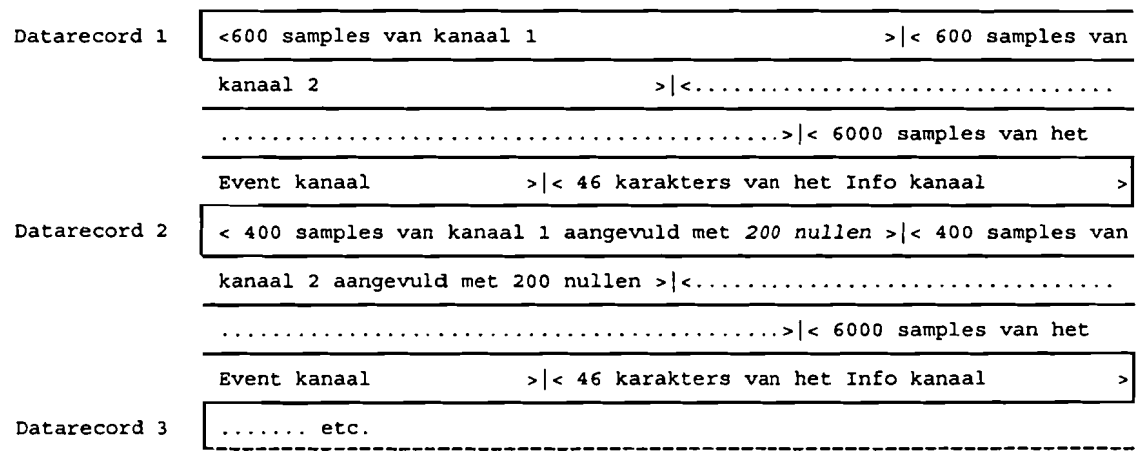
Een Info kanaal wordt altijd toegevoegd aan een geconverteerde Fysian data file.
Dit kanaal moet als ASCII geïnterpreteerd worden. De variabelen die hierin gecodeerd worden zijn al in paragraaf 4.3 beschreven. Hier wordt ook een voorbeeld gegeven van de inhoud van een Info kanaal. Als alle informatie uit het Info kanaal naar de EDF data file is geschreven, zal het Info kanaal, mocht dit nodig zijn, aangevuld worden met spaties.

Het aantal samples in het Info kanaal per data record wordt bepaald door het aantal karakters dat nodig is om alle variabelen te beschrijven te delen door het aantal data records in de data file en dit resultaat (aantal karakters per data record) nogmaals te delen door 2, daar een sample (integer) bestaat uit 2 bytes en een karakter maar 1 byte groot is.

De data van de Fysian records wordt gecopieerd naar de EDF data records. Nadat het aantal samples van alle kanalen in een trial naar een EDF data record gecopieerd zijn wordt de data van het Info kanaal en Event kanaal (mits nodig) toegevoegd.

Het voorbeeld in figuur 4.2 laat zien hoe een EDF data record opgebouwd is:

In dit voorbeeld geldt:
Duur van een (Fysian) trial : 10 seconden
Duur van een data record: 6 seconden
Aantal samples in de totale (Fysian) trial : 1000
Aantal samples per kanaal per data record : 600
Aantal samples in Event kanaal per data record: 6000 (6 seconden bij 1000 Hz)
Aantal samples in Info kanaal per data record: 23 (komt overeen met 46 ASCII)



Figuur 4.2 Voorbeeld van een EDF data record.

5. Ontwerp van een basis EDF software bibliotheek

5.1 Inleiding

Het ontwikkelen van software gebeurt zelden door een persoon. Er wordt altijd wel in teamverband gewerkt. Bij het ontwikkelen van de software voor het verwerken van de experimentele data van de sectie Psychonomie zal dit ook zeker het geval zijn. Het is de bedoeling dat in de toekomst verscheidene programmeurs verschillende software moeten gaan ontwikkelen voor het verwerken van de data.
Als elke programmeur telkens de opbouw van de 'extended' EDF data files zou moeten analyseren en eigen lees en schrijf functies moet ontwikkelen voor het schrijven en lezen van de in een EDF data file opgeslagen data, kost dit onnodig veel tijd.
Het is hetzelfde als een programmeur eigen routines zou moeten maken voor het aansturen van een harddisk om data te lezen en te schrijven. Voor zulke routines bestaan er standaard bibliotheken in C welke door alle programmeurs gebruikt kunnen worden.

Het belangrijkste onderdeel van dit afstuderen is het ontwikkelen van een 'standaard' bibliotheek voor IO-bewerkingen op data files van het extended EDF-formaat.
Toekomstige programmeurs van EDF programmatuur kunnen de routines uit deze bibliotheek gebruiken om data te lezen van en te schrijven naar een file. De programmeur hoeft dan alleen een functie aan te roepen met 'lees van trial 3 kanaal 5'en de functie zorgt voor de verdere afhandeling van het lezen van de data. De programmeur hoeft zich alleen te concentreren op de bewerking/analyse van de gelezen data.

Het ontwikkelen van een bibliotheek vraagt om structuur en eenvoud, daar in de toekomst anderen deze software moeten gaan gebruiken.
De functies zijn geprogrammeerd in ANSI C om platformonafhankelijkheid te verkrijgen.
De data kan in verschillende structuren worden opgeslagen. Deze structuren worden beschreven in 5.2. Deze structuren kunnen (en moeten) worden geïnitialiseerd en kunnen ook weer vrij gemaakt worden.
Op deze manier blijft de programmeur gevrijwaard van de 'low-level' operaties op de data files. Hij krijgt alleen te maken met lezen/schrijven van header informatie, kanaal-data, trial-data en (EDF)record-data. De geprogrammeerde basis routines worden in paragraaf 5.4 behandeld.

5.2 De structuren voor de header

Om de programmeur af te schermen van het direct manipuleren van de ruwe data zijn naast een paar noodzakelijke structuren voor de header informatie, nieuwe structuren ontworpen voor het verwerken van de data.
Naast structuren zijn een aantal type-definities geïntroduceerd met de syntax **EDF_*type***, omdat voornamelijk integers platform afhankelijk zijn. De EDF data is gedeclareerd als 2 byte integers, daarom moet een EDF_int een integertype zijn van 2 byte. Dit is opgelost door de EDF_int als short te declareren, daar een short vrijwel altijd 2 bytes groot is. Dit moet door de

programmeur bij het overzetten naar een nieuw besturings-systeem gecontroleerd worden. Het geeft ons ook de mogelijkheid verscheidene typen met behulp van een correctie te kunnen veranderen voor alle source. In de file EDF.H worden deze structuren gedeclareerd. Deze structuren worden hieronder een voor een behandeld.

5.2.1 De hoofdstructuur : EDF_file

Voor het vereenvoudigen van routine aanroepen is de structuur **EDF_file** ontworpen.
Deze structuur bundelt een aantal variabelen die altijd nodig zijn bij het verwerken van een EDF datafile. De structuur bevat pointers naar de data file (type EDF_stream), de header informatie (type EDF_header), de kanaal informatie lijst (type EDF_channel_node) en de event-tabel (type Event_table_node).

De syntax van deze structuur is:

```
typedef struct
{
  EDF_stream        *Pstream;
  EDF_header        *Pedfhead;
  EDF_channel_node  *Pedfchannels;   (zie figuur 5.1)
  Event_table_node  *Peventtable;    (zie figuur 5.2)
} EDF_file;
```

5.2.2 De header structuren : EDF_header & EDF_channel

Twee structuren beschrijven zichzelf. Dit zijn de structuur **EDF_header** voor de statische header informatie en **EDF_channel** voor de kanaal informatie. Dit zijn 2 structuren waarin de verschillende variabelen kunnen worden opgeslagen.
Syntax voor de EDF_header structuur (in C-syntax):

```
typedef struct
{
  EDF_char version[LVERSION+1];      /* version of this data format (0) */
  EDF_char local_pat_id[LPAT_ID+1];      /* local patient identification    */
  EDF_char local_rec_id[LREC_ID+1];      /* local record identification     */
  EDF_char startdate[LCDATE+1];      /* startdate of recording (dd.mm.yy)*/
  EDF_char starttime[LCTIME+1];      /* starttime of recording (hh.mm.ss)*/
  EDF_ulong nbyte;                   /* number of bytes in header       */
  EDF_char reserved[LHRESER+1];      /* reserved                        */
  EDF_long nrec;                     /* number of datarecords (-1 if unknown)*/
  EDF_float rec_nsec;                /* duration of a datarecord in seconds */
  EDF_int  nchan;                    /* number of channels (signals) in */
  } EDF_header;
```

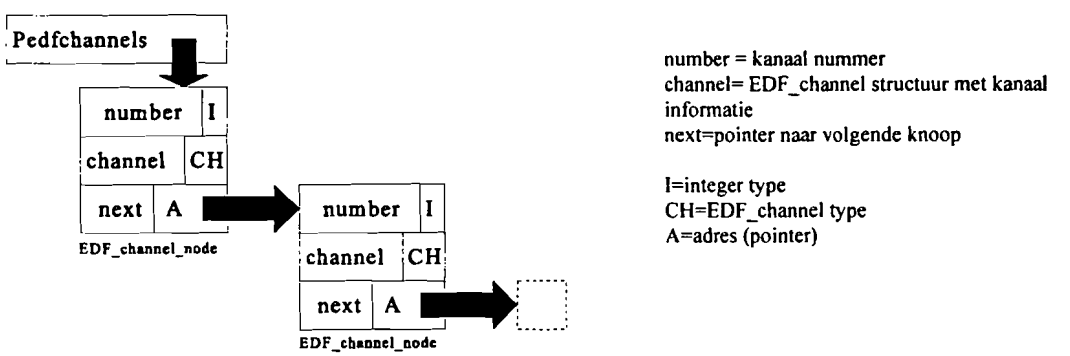
Syntax voor de EDF_channel structuur:

```
typedef struct
{
  EDF_char  label[LLABEL+1];      /* label (e.g. EEG FpzCz or Bodytemp  */
  EDF_char  trans[LTRANS+1];      /* transducer type                    */
  EDF_char  physdim[LPHDIM+1];    /* physical dimension                 */
  EDF_float physmin;              /* physical minimum                   */
  EDF_float physmax;              /* physical maximum                   */
  EDF_int   digmin;               /* digital minimum */
  EDF_int   digmax;               /* digital maximum */
  EDF_char  prefilter[LPFILT+1];  /* prefiltering */
  EDF_ulong nsamp;                /* number of samples in each data record */
  EDF_char  reserved[LCRESER+1];  /* reserved */
} EDF_channel;
```

De definities zoals LLABEL zijn constanten die de grootte van de ASCII velden representeren. Dit maakt het aanpassen van de source eenvoudiger. Door deze constante te gebruiken in plaats van de integer waarde kan een simpele verandering in de header file genoeg zijn om een heel programma aan te passen.
(Al deze constanten beginnen met hoofdletter L.)

5.2.3 Structuur voor de kanaal informatie : EDF_channel_node

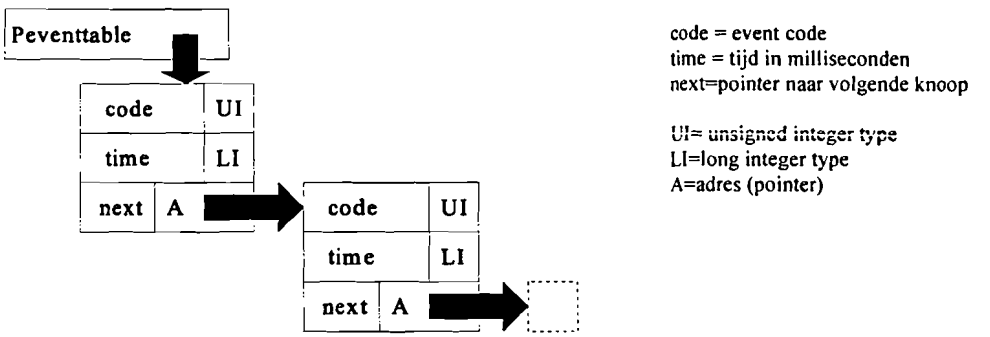
Volgens de standaard is het aantal kanalen vrij aan de gebruiker. Dit betekent dat het maximum aantal kanalen niet vast ligt. Door gebruik te maken van dynamische geheugen allocatie en pointers is een structuur ontworpen om de informatie van de kanalen in op te slaan. Dit heet ook wel een 'linked list'. De knopen van deze lijst zijn opgebouwd met de structuur **EDF_channel_node**.
Figuur 5.1 geeft een grafische representatie van de opbouw van zo'n linked list met de kanaal informatie. Pedfchannels is de variabele welke het begin van de lijst onthoudt.



Figuur 5.1 De lijst met kanaal informatie

5.2.4 Structuur voor de *event tabel*: Event_table_node

Om het event kanaal eenvoudig te kunnen gebruiken, introduceren we een andere structuur. De structuur **Event_table_node** wordt gebruikt om de code en de tijd dat deze code voorkomt op te slaan. Met behulp van de structuur kunnen we een linked list maken van alle codes die in het Event kanaal voorkomen. Op deze manier staat de tijdsafhankelijke informatie in het geheugen en is deze snel te doorlopen. In figuur 5.2 wordt een grafische representatie gegeven van deze lijst. Peventtable is de pointer naar het begin van de lijst met event data.



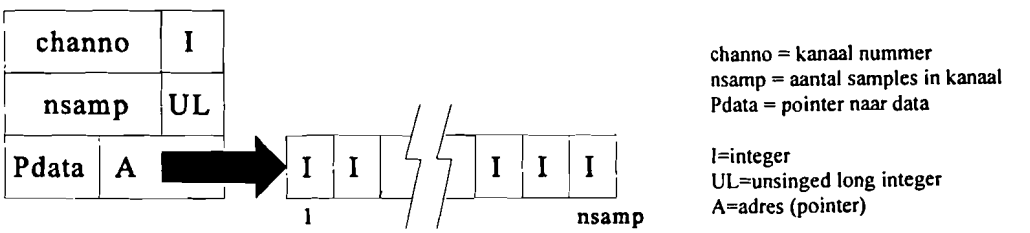
Figuur 5.2 De event tabel

5.3 De structuren voor de data

Er zijn verschillende soorten hoeveelheden data die we willen lezen en schrijven. Het lezen en schrijven van een kanaal, het lezen en schrijven van een hele trial of het lezen en schrijven van de data record. Om het de programmeur eenvoudiger en duidelijker te maken zijn hier een aantal structuren voor gedefinieerd. Deze structuren worden met behulp van grafische representaties weergegeven (figuur 5.3 t/m 5.6).

5.3.1 Structuur voor de kanaal data: EDF_channeldata

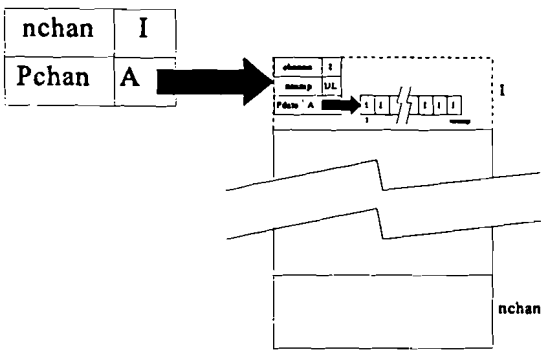
Willen we de data van een kanaal lezen of opslaan, dan gebruiken we de structuur **EDF_channeldata.**



Figuur 5.3 De EDF_channeldata structuur

5.3.3 Structuur voor een data record: EDF_recorddata

Willen we een data record lezen of schrijven kunnen we gebruik maken van de structuur **EDF_recorddata.**



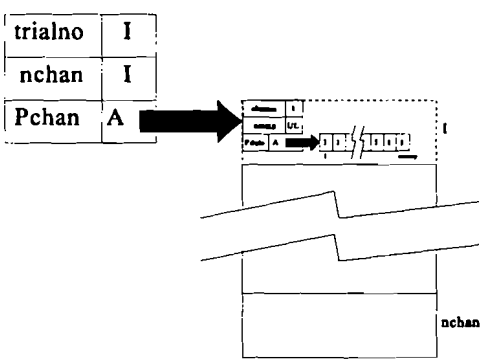
nchan = aantal kanalen
Pchan = pointer naar array van EDF_channeldata structuren

I=integer
A=adres (pointer)

Figuur 5.4 De EDF_recorddata structuur

5.3.2 Structuur voor de trial data: EDF_trialdata

Voor het opslaan van een hele trial (alle kanalen in een trial) gebruiken we de **EDF_trialdata** structuur. De programmeur moet rekening houden met het geheugen dat nodig is voor zo'n structuur.



trialno = trial nummer
nchan = aantal kanalen in structuur
Pchan = pointer naar array van EDF_channeldata structuren

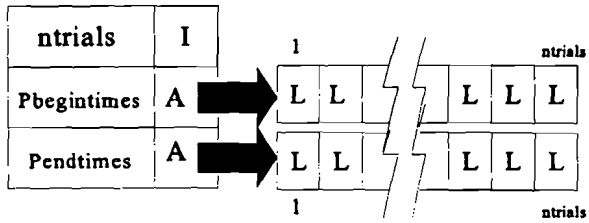
I=integer
A=adres

Figuur 5.5 De EDF_trialdata structuur

Deze structuur lijkt veel op de EDF_recorddata structuur. Het wezenlijk verschil is dat de data van de kanalen in de EDF_trial structuur verdeeld kan zijn over verschillende data records, daar een trial uit meerdere data records kan bestaan. De gefragmenteerde kanaal data in de data file is nu als 'een geheel' in deze structuur opgeslagen.

5.3.4 Structuur voor trial informatie: EDF_trialinfo

Tot slot wordt een structuur geïntroduceerd welke het ons vereenvoudigt om te beschikken over de begin en eindtijden van de verschillende trials. Tevens staat in deze structuur het aantal trials dat in de datafile voorkomt. Het betreft de structuur **EDF_trialinfo** (zie figuur 4.7 voor een grafische weergave).



Figuur 5.6 De EDF_trialinfo structuur

5.4 De basis routines

De basis EDF software bibliotheek bestaat uit een aantal routines die voornamelijk input output bewerkingen uitvoeren. De bibliotheek moet een basis vormen voor toekomstige programmeurs van EDF bewerkingssoftware.

In deze paragraaf wordt een opsomming gegeven van alle basis routines en hun functie. Voor een uitgebreide beschrijving van deze routines verwijs ik naar de handleiding die beschikbaar is op de sectie Medische Elektrotechniek (TUE) en de sectie Psychonomie (KUB). Hierin staat uitvoerig beschreven hoe alle routines aangeroepen moeten worden en waar de programmeurs rekening mee moeten houden. Ook de error-handeling van de routines wordt hier beschreven.

De structuren die we gebruiken voor het opslaan van informatie en data kunnen worden geïnitialiseerd met behulp van de volgende routines:

- **EDF_open**
  Deze functie initialiseert een EDF_file structuur. Het opent de gewenste data file voor lezen of schrijven en alloceert geheugen voor het opslaan van de header, kanaal en event-tabel informatie. Alle variabelen krijgen hun initiële waarde (getallen worden nul en karakters worden met spaties geïnitialiseerd).
  De header- en kanaal informatie van de betreffende file worden ingelezen en opgeslagen in de EDF_file structuur.
- **EDF_init_datarecord**
  Deze functie initialiseert een EDF_datarecord structuur. Het alloceert geheugen voor het opslaan van de data van een data record.

- **EDF_init_trial**
  Deze functie initialiseert een EDF_trialdata structuur. Het alloceert geheugen voor het opslaan van de data van een hele trial. De functie maakt gebruik van de data welke staat in een EDF_trialinfo structuur voor het bepalen van de trial duur. Men moet dus voor het initialiseren van een variabele met de EDF_trialdata structuur de functie 'get_trialinfo' aanroepen, welke de variabele met de EDF_trialinfo structuur initialiseert en de nodige data uit het event kanaal haalt.
- **EDF_get_trialinfo**
  Deze functie initialiseert een EDF_trialinfo structuur en bepaalt meteen de begin en eind tijden van de trials uit het event kanaal. Deze tijden worden in de structuur opgeslagen voor verdere processen.
- **EDF_init_channel**
  Deze functie initialiseert een EDF_channeldata structuur. Het alloceert geheugen voor het opslaan van de data van een kanaal.

De structuren dienen altijd eerst geïnitialiseerd te worden voordat ze gebruikt worden in de aanroepen van andere functies/routines.

Naast het initialiseren van de variabelen (structuren) moeten we het geheugen dat de variabelen gebruiken vrij kunnen maken als deze variabelen niet meer nodig zijn. (Eenvoudig te gebruiken bij een toekomstige implementatie in C++, als C++ constructor/deconstructor functie(s)). De functies spreken verder voor zich.
Het vrij maken van dit geheugen gebeurt met de volgende functies:

- **EDF_close**
  Naast het vrij maken van het gebruikte geheugen, kan de header als dit gewenst is naar de data file geschreven worden en wordt de datafile afgesloten. In opdracht van de programmeur kan de functie na het afsluiten van de file, deze wissen.
- **EDF_free_datarecord**
  Vrij maken van EDF_recorddata structuur.
- **EDF_free_trial**
  Vrij maken van EDF_trialdata structuur.
- **EDF_free_trialinfo**
  Vrij maken van EDF_trialinfo structuur.
- **EDF_free_channel**
  Vrij maken van EDF_channeldata structuur.

Er is een aantal routines geschreven die door andere functies/routines gebruikt worden. Sommige van deze functies zijn echter ook handig voor programmeurs. Daarom zijn een aantal van deze routines als externe functie aan te roepen, en zo dus vrij gegeven voor gebruik.
Het betreft de routines:

- **EDF_det_rec_nsamp**
  Deze routine berekent het aantal samples dat in een data record worden opgeslagen.

- **EDF_get_time_from_table**
  Deze routine bepaalt de tijd (in milliseconden), uit het event kanaal, welke bij een bepaalde code hoort. Deze routine gebruikt de 'linked list' met de event informatie welke aangeduidt wordt met een (pointer)variabele van het Event_table_node structuur.
- **EDF_get_real_freq**
  Deze functie bepaalt de 'echte' bemonsteringsfrequentie van een kanaal. Als de variabele SF[bemonsteringsfrequentie] in de kanaal informatie voorkomt, zal de functie de frequentie uit de ASCII variabele decoderen. (Deze variabele wordt specifiek gebruikt door de KUB)

De voornaamste functies in de bibliotheek zijn uiteraard de input/output functies. Deze functies worden gebruikt voor het lezen en het schrijven van de header informatie en de file data.
Bij elke lees-functie hoort een schrijf-functie die een complementaire implementatie bewerkstelligt.
De leesroutines (en *schrijfroutines*) zijn:

- **EDF_read_channel_samp** / ***EDF_write_channel_samp***
  Deze functie leest de data tussen twee absolute sample posities van een kanaal en slaat deze op in een geheugen gebied in de vorm van een array (alle data achter elkaar).
  De functie is de basis voor de hierna volgende functies. Deze functies rekenen de gewenste informatie om in de absolute sample posities en geven deze door aan de read_channel_samp functie welke de echte lees functie uitvoert.
- **EDF_read_channel** / ***EDF_write_channel***
  Deze functie leest de data van een kanaal uit de gewenste trial en slaat deze op in een EDF_channeldata structuur. Men kan een interval opgeven of de hele trial laten lezen. Mocht het gealloceerde geïnitialiseerde geheugen van de EDF_channeldata structuur niet voldoende zijn voor het opslaan van alle data, zal de functie de EDF_channeldata structuur zelf aanpassen . Bij write_channel is dit uiteraard niet nodig, daar deze functie het aantal samples welke in deze structuur zijn aangegeven wegschrijft.
- **EDF_read_trial** / ***EDF_write_trial***
  Deze functie leest een heel trial (alle kanalen in die trial) in en slaat deze op in een EDF_trialdata structuur.
- **EDF_read_between_events** / ***EDF_write_between_events***
  Deze functie leest de data tussen 2 codes (events) van een bepaald kanaal. De programmeur bepaalt welke codes. Deze data wordt opgeslagen in een dynamische array in het geheugen.

De write_.... functies schrijven de data naar een file in plaats van lezen. Ze maken gebruik van dezelfde structuren als de read_... functies.

De laatste functie in de bibliotheek in zijn huidige vorm is **EDF_read_event_channel**. Deze functie scant het Event kanaal voor 'bekende' *codes* en slaat deze en de *tijd* (in ***milliseconden***) dat ze voorkomen op in een tabel (linked list met Event_table_node structuur). Een van de variabelen (Peventtable) in de hoofdstructuur EDF_file is een pointer naar deze tabel (aangezien elke data file over eigen event-data kan beschikken).

6. Testen van de ontworpen software

6.1 CHECKREC en QD van Alpo Värri

Een goede manier om te testen of het conversie programma FYS2EDF de Fysian data omzet in een geldig EDF formaat, is software van derden te gebruiken.
Als de geconverteerde data aan de standaard voldoet mag de software er geen problemen mee hebben, ondanks het feit dat deze software niet het 'extended' EDF formaat gebruikt. Maar zoals eerder besproken voldoet het uitgebreide EDF formaat nog steeds aan de originele afspraken van EDF.

Alpo Värri van de Tampere University of Technology te Finland, een van de onderzoekers in het eerder genoemde Task Group of Signal Analysis, heeft een aantal programma's ontwikkeld voor het verwerken van EDF data. Deze zijn voornamelijk bedoeld voor het bekijken van slaap registraties.
Het programma CHECKREC wordt gebruikt om de EDF header van een datafile op scherm weer te geven. Het programma QD is een grafisch 'display'-programma waarmee men de kanalen grafisch op scherm weer kan geven.

De programma's zijn gedurende het hele ontwikkelproces van FYS2EDF gebruikt voor verificatie van de geconverteerde data. We kunnen hier enkele conclusies uit trekken met betrekking tot FYS2EDF , conclusies over 'extended' EDF in het algemeen, en over beperkingen in het gebruik van de software van derden.

Het programma CHECKREC accepteert alleen 'integers' (natuurlijke gehele getallen) voor data record lengtes, terwijl onze software deze als 'float' (reële getallen) interpreteert. De standaard volgens referentie [1] specificeert geen specifiek data-type voor deze variabele. Er wordt alleen aangeraden record lengtes van hele seconden te gebruiken, maar dat geldt voornamelijk voor het voorbeeld waar men EDF gebruikt voor slaapregistraties. Alleen als een record groter zou worden dan 64 Kbyte, wordt aangeraden kleinere record lengtes te nemen.

Bij het gebruik van het programma QD zijn we er achter gekomen dat dit programma problemen heeft met grote record lengtes. Bij sommige geconverteerde Fysian data bedroegen deze 8 seconden. Dit gaf problemen voor het programma.
Een ander probleem is het beperkt aantal kanalen van 32 welke het programma aankan. Er zijn data files waarbij (door toevoeging van Info- en Event kanaal) er meer dan 32 kanalen gebruikt worden. Het EDF formaat daarentegen legt geen grens aan het aantal kanalen dat gebruikt kan worden. Dit is een restrictie van het programma QD.

Men kan de conclusie maken dat men voor het gebruik van software van derden op eigen data, goed dient weten waar de grenzen van deze software liggen. Aannemen dat er geen grenzen zijn omdat deze niet in de standaard gedefinieerd zijn is natuurlijk verkeerd. Een programmeur kan om verscheidene redenen kiezen voor bijvoorbeeld een maximum aantal

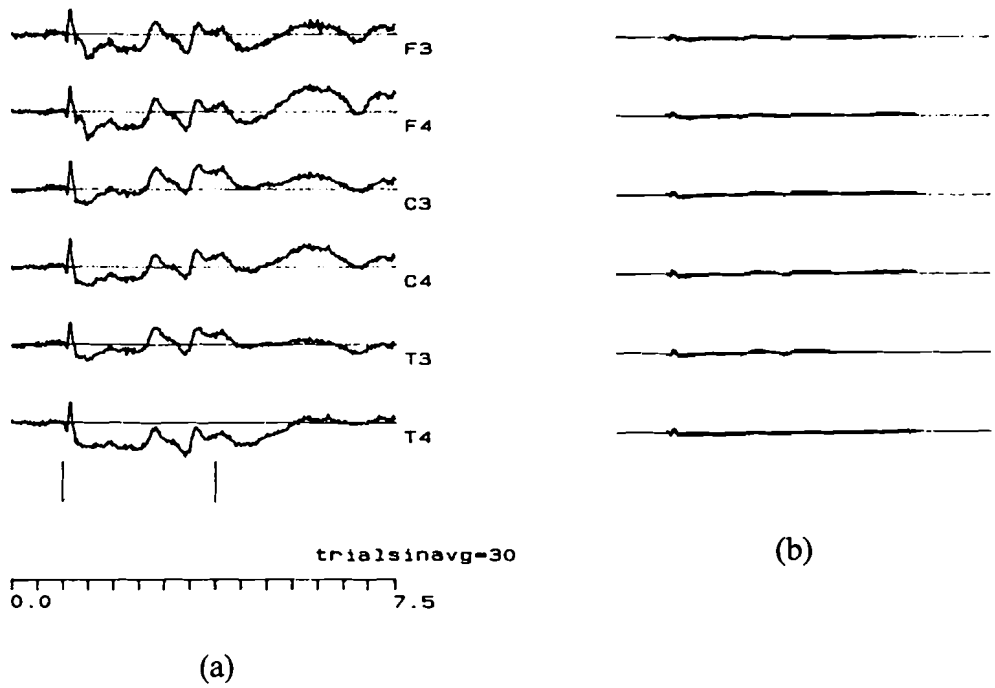
kanalen of een maximum data record lengte vanwege apparatuur en de grenzen van eigen onderzoek.

6.2 EDF conversie naar NLA

Op de sectie EME aan de TUE heeft ir. M van de Velde een conversie programma geschreven dat 'extended' EDF data omzet in een specifiek formaat genaamd NLA. Hoe dit formaat opgebouwd is zal niet behandeld worden en is voor deze test ook niet noodzakelijk.
Met behulp van mijn conversie programma is een Fysian data file omgezet naar extended EDF data file. Deze file is verder geconverteerd naar het NLA formaat. Met behulp van verwerkingssoftware voor data van het NLA formaat is hier een gemiddelde data file mee gemaakt.

Bij de sectie Psychonomie aan de KUB is van dezelfde Fysian data file een gemiddelde gemaakt met behulp van Fysian verwerkingssoftware.
Als het conversie programma de Fysian data file goed heeft geconverteerd moeten de gemiddelden identiek zijn.

Figuur 6.1a geeft grafisch de gemiddelden (in dezelfde grafiek) weer en figuur 6.1b geeft het verschil van deze gemiddelden (de kleine verschillen worden veroorzaakt doordat men bij Fysian met 16 bits getallen rekent en bij NLA met 12 bits getallen, waardoor afrond-verschillen ontstaan tussen de gemiddelden).



Figuur 6.1 a De resultaten van een trial-gemiddelde van een Fysian data file in Fysian en NLA formaat (10 µV/cm). De twee gemiddelden van de 6 kanalen vallen in de figuur samen.
b Het verschil van de gemiddelden (1 µV/cm).

6.3 EDF conversie naar Fysian

Een derde methode die gebruikt gaat worden voor het testen van de software is door de geconverteerde EDF data terug te converteren naar Fysian en deze te vergelijken met de originele Fysian data file. Als de conversies goed zijn mag er geen verschil tussen de data files zijn.

De conversie van EDF naar Fysian wordt gerealiseerd om in gevallen dat men een bewerking op nieuwe experimentele data in het EDF formaat wil uitvoeren en deze bewerking nog niet voorhanden is op het nieuwe platform, terug te kunnen vallen op de VAX.
Het conversie programma wordt geschreven door ir. M.M.C. van den Berg-Lenssen (mede ontwikkelaar van het Fysian formaat) aan de KUB. Het wordt in Fortran geschreven en zal op de VAX gedraaid worden.
Tijdens het schrijven van dit bestek is het programma nog in de maak en kunnen nog geen conclusies getrokken worden over het gebruik van de terug-conversie.

Deze conversie is tevens een goede manier om andere bewerkingssoftware te testen die in de toekomst geschreven moet worden. Een verwerkings-programma dat op de PC werkt moet dezelfde resultaten geven als een soortgelijk verwerkings-programma op de VAX.

6.4 Gebruik van enkele functies

De output write-functies worden al gedeeltelijk gebruikt aan de Technische Universiteit Eindhoven. Het gaat voornamelijk om de header functie. Deze wordt gebruikt bij een conversie programma om OSG BrainlaB file om te zetten in Extended EDF. Dit voorbeeld is puur ter illustratie en zal hier niet behandeld worden.

Bij de sectie Psychonomie is J.P.C. van Wieringhen Borski bezig (afstudeerder op vakgroep MBS/Sectie Medische Elektrotechniek aan de TUE) met een programma dat de header van een Extended EDF data file kan displayen en data naar ascii converteren. Voor deze software zijn ook al enige functies uit EDFIO.C gebruikt. Deze afstudeerder is de eerste programmeur die gebruik maakt van de bibliotheek.

7. Inventarisatie van 'event' bewerkingsalgoritmen

7.1 De codes die momenteel aan de KUB gebruikt worden

Tijdens de experimenten slaan we tijdsafhankelijke informatie op in het Event kanaal. Deze informatie geeft naast een tijdindicatie ook aan wat voor een soort gebeurtenis (event) plaats heeft gevonden. De gebeurtenissen worden gecodeerd opgeslagen, waarbij de code de gebeurtenis weergeeft en het sample-nummer de tijd representeert.

De sectie Psychonomie van de Vakgroep Psychologie aan de KUB maakt momenteel gebruik van de volgende codes voor het encoderen van gebeurtenissen tijdens experimenten.
In de toekomst kan deze lijst uitgebreid worden met codes voor nieuwe gebeurtenissen.

Codes(Hexadecimaal) voor de gebeurtenissen:

Event	Main Code	Extra informatie over event	Sub Code
None	0x00	-	00
Begin of Trial	0x01	Normal trial Calibration trial EOG trial	01 02 03
End of trial	0x02	Normal trial Calibration trial EOG trial	01 02 03
Begin of baseline	0x03	voor welk kanaal (alle=FF)	FF
End of baseline	0x04	voor welk kanaal (alle=FF)	FF
Stimulus on	0x05	nummer of soort stimulus	xx
Stimulus off	0x06	nummer of soort stimulus	xx
Reaction on	0x07	nummer/code van reactie die aangeeft op welke stimulus is gereageerd	xx
Reaction off	0x08	nummer/code van reactie die aangeeft op welke stimulus is gereageerd	xx
Multiple Events	0xFF	aantal gebeurtenissen die plaats hebben gevonden op dit moment	xx

Bijvoorbeeld het begin van een calibratie trial wordt gecodeerd door 0102 (Hex.) wat overeenkomt met een decimale waarde van 258.

7.2 Mogelijke programmatuur en hun gebruik van de codes

Enkele routines die gebruik maken van de codes in het Event kanaal hebben we al kunnen zien in paragraaf 5.4. Deze routines gebruiken de 'begin of trial' en 'end of trial' event-codes om te bepalen waar een bepaalde trial begint of eindigt.

Deze paragraaf zal enkele voorbeelden van mogelijke bewerkingssoftware beschrijven en de manier waarop deze software gebruik kan maken van de event-codes in het Event kanaal. Dit wordt gedaan door eerst de werking van de routine of programma te beschrijven gevolgd door de event-codes welke gebruikt kunnen worden en hoe deze gebruikt moeten worden.

Een functie welke standaard tot de Fysian functies behoort is de *functie baseline*. Deze functie berekent het gemiddelde van een interval welke in fysian gegeven wordt door 2 sample waarden. Dit gemiddelde wordt gebruikt om een baseline correctie (offset correctie) uit te voeren op de data. Dit gebeurt door het berekende gemiddelde van alle data waarden af te trekken.
Bij een EDF data file wordt het begin en einde van de baseline aangegeven door de codes 'begin of baseline' en 'end of baseline'. De functie kan het gemiddelde bepalen door de data tussen deze event-codes te middelen. Als men onderscheid wil maken tussen een baseline gecorrigeerde en een ongecorrigeerde data file, kan dat door een variabele toe te voegen. Dit kan bijvoorbeeld door in het Info kanaal de variabele BASCOR[TRUE] of BASCOR[FALSE] te implementeren (Dit is maar een voorbeeld en is momenteel niet geïmplementeerd door de sectie Psychonomie).

Een van de belangrijkste bewerkingen van de data is *middelen van de trials*. De software moet het *'trial-gemiddelde'* berekenen door de data van de verschillende trials te middelen.
De event-codes 'begin of trial' en 'end of trial' zullen hier gebruikt worden. We willen dan bijvoorbeeld alleen normale trials (geen calibratie- en eog-trials) middelen. Met behulp van de event-codes in het Event kanaal kunnen we dit onderscheid maken en de goede trials over elkaar middelen.
Een optie hier is middelen met een andere uitgangspositie. Men wil bijvoorbeeld middelen vanaf de eerste stimulus, omdat deze niet bij elke trial op precies hetzelfde moment (t.o.v. begin van een trial) plaats vindt. De middeling zal dan plaats vinden tussen de event-codes 'stimulus on' of 'stimulus off' en 'end of trial'.

Reactietijden moeten ook bepaald kunnen worden. Ook hier zal een functie voor geschreven kunnen worden. De reactietijd is de tijd tussen 'stimulus on' en 'reaction on' of bijvoorbeeld als de proefpersoon een toets los moet laten en een andere toets moet indrukken, tussen 'reaction off' van toets 1 en 'reaction on' van toets 2. De toetsen kunnen gecodeerd worden in de tweede byte van de event-code (subcode).
Stel je hebt 10 soorten stimuli en 10 toetsen welke ingedrukt moeten worden afhankelijk van de soort stimulus, dan kan men in de subcode het nummer van de stimulus en reactie coderen. Reactietijden van verschillende stimuli kunnen op deze manier eenvoudig bepaald worden.

8. Conclusie & discussie

Er is een uitbreiding op het al bestaande EDF uitgewerkt om deze geschikt te maken voor het opslaan van 'event gerelateerde' data. De ontwikkelde software is getest door mij zelf en mijn begeleiders aan de KUB en aan de TUE.

Het Fysian naar EDF conversie programma werkt goed. Het converteert de Fysian data files naar extended EDF data files. De trial en tijdsafhankelijke informatie die bij Fysian in de header is opgeslagen wordt naar het Event kanaal van de EDF data file geconverteerd. Tijdens een test is de informatie in het Event kanaal gebruikt bij het middelen van een data file die van extended EDF naar het NLA formaat (paragraaf 6.2) is geconverteerd.

Tijdens het afronden van mijn afstuderen is er nog weinig bewerkingssoftware geschreven die de functies in de bibliotheek gebuikt. Dit maakt het moeilijk de functies in 'crisis' situaties te testen. Er is echter wel veel werk gemaakt van de errorafhandeling, dus het achterhalen van kleine bugs in de software zal geen groot probleem zijn en relatief weinig tijd kosten.
De meeste functies zijn getest door eenvoudige programma's die de functies hebben gebruikt om een EDF data file te copiëren. Op deze manier werden de complementaire lees en schrijf functies getest.

De basis structuur van de functies en de variabelen moet toekomstige programmeurs veel werk uit handen nemen. Een goede basis zorgt er ook voor dat men elkaars sources kan lezen en begrijpen. Hier horen natuurlijk ook een aantal afspraken bij wat betreft het gebruik van variabelen en de functies. Op de secties Psychonomie (KUB) en Medische Elektrotechniek (TUE) is documentatie aanwezig dat een aantal richtlijnen geeft voor programmeurs die met deze bibliotheek willen werken.
Hierin is ook de ervaring die opgedaan is met de gebruikte compiler opgenomen om toekomstige programmeurs op voorhand te waarschuwen voor eventuele obstakels die men tegen kan komen bij het ontwikkelen van software met het gekozen software-pakket (Microsoft Visual 1.0).

Er is nu een basis bibliotheek en het echte werk kan beginnen. De nodige bewerkings-software voor het nieuwe besturings-systeem kan geprogrammeerd worden.
Er zijn nog enkele functies die ontwikkeld moeten worden. Het gaat hier voornamelijk om functies die het Info kanaal kunnen interpreteren. Daar is nog geen software voor geschreven. Voor het Event kanaal zijn al de nodige functies voorhanden in de bibliotheek.

Literatuur

[1] Bob Kemp et al.
A SIMPLE FORMAT FOR EXCHANGE OF DIGITIZED POLYGRAPHIC RECORDINGS
Electroencephalography and clinical neurophsiology 1992, Vol 82, p.391-391

[2] M.M.C. van den Berg-Lenssen et al.
FYSIAN SOFTWARE DEVELOPMENT GUIDE
University of Tilburg, Psychonomy section, march 1991

[3] Brian W. Kernighan & Dennis M. Ritchie
THE C PROGRAMMING LANGUAGE
Prentice Hall, USA, 1978

[4] M.D. Rainer Spehlmann
ENVOKED POTENTIAL PRIMER
Butterworth Publishers, USA, 1985