

MATLAB Import/Export Routines

Version 1.3

User Manual

CTF MEG™ System



PN900-0124

Revision 1.3, 5 October 2007

Proprietary and Confidential

MATLAB Import/Export Routines

User Manual

Version 1.3

VSM MedTech Ltd.



DISCLAIMER

VSM MedTech Ltd. ("VSM") has used reasonable effort to include accurate and up-to-date information in this manual; it does not, however, make any warranties, conditions or representations as to its accuracy or completeness. VSM assumes no liability or responsibility for any errors or omissions in the content of this manual. Your use of this manual is at your own risk. Under no circumstances and under no legal theory shall VSM be liable for any indirect, direct, special, incidental, punitive, exemplary, aggravated or consequential damages arising from your use of this manual.

Features and specifications of VSM's products are subject to change without notice. This manual contains information and images about VSM, its magnetoencephalographic systems and its other products that are protected by copyright.

Copyright © VSM MedTech Ltd., 2007. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior permission by VSM MedTech Ltd..

Proprietary and Confidential

VSM MedTech Ltd.

9 Burbidge Street,

Coquitlam, BC, Canada V3K 7B2

Phone: +1 (604) 472-2300

Fax: +1 (604) 472-2301

Web: www.vsmmedtech.com



Contents

Table of Contents

1. Introduction.....	9
Revision 1.1:.....	9
Revision 1.2:.....	9
Revision 1.3:.....	9
2. Installation and Help.....	10
2.1 Installation.....	10
2.2 Help.....	10
3. Reading CTF data into MATLAB.....	11
3.1 readCTFds - reading the dataset description.....	11
3.2 getCTFdata - reading the data.....	13
3.3 setCTFDataBalance - balancing the data.....	14
3.4 getCTFBalanceCoefs - Balancing data using the table of balance coefficients.....	16
4. Writing data from MATLAB to CTF format.....	19
4.1 No clinical use.....	19
4.2 Data preparation for writeCTFds.....	19
4.3 Creating datasets with writeCTFds.....	23
4.4 Adding trials to existing datasets with addCTFtrial.....	24
5. Reading and creating MRI and head-model files.....	25
5.1 Reading CTF MRIs with readCTFMRI.....	25
5.2 Creating CTF MRIs with writeCTFMRI.....	26
5.3 Reading CTF head-model files with readCTFhdm.....	27
5.4 Creating CTF head-model files with writeCTFhdm.....	28
Appendix A : The data structure produced by readCTFds.....	31
Appendix B : Structure ds.res4.....	37
List of Tables	
Table 1	12



Warnings and Cautions



WARNING

These programs must not be used for clinical applications. If CTF MEG data are processed by this tool, they should not be later employed for clinical and/or diagnostic purposes.

Contact Information

Please send comments, questions, problems, or suggestions for improvements to:

VSM MedTech Ltd.

9 Burbidge Street

Coquitlam, BC Canada V3K 7B2

Tel: +1 604-472-2300

FAX: +1 604-472-2301

Email: CTFCustomerSupport@vsmmedtech.com



Introduction

1. Introduction

The popularity of MATLAB for data analysis has resulted in the need for a package of MATLAB programs that will allow the reading of CTF datasets into the MATLAB environment, and the preparation of CTF datasets containing the results of MATLAB processing.

To meet this need, VSM has prepared MATLAB programs for manipulating CTF datasets:

- (1) `readCTFds`: Reads the files that define a CTF dataset, but not the sensor data.
- (2) `getCTFdata`: Reads the sensor data.
- (3) `setCTFDataBalance`: Balances the data.
- (4) `getCTFBalanceCoefs`: Reads the balance tables that are included in each dataset.
- (5) `writeCTFds`: Creates a new CTF-format dataset.
- (6) `addCTFtrial`: Adds trials to a dataset created by `writeCTFds`.

The following programs have been prepared for reading and writing MRIs and head-model files :

- (6) `readCTFMRI` : Reads MRIs in CTF format (version 4)
- (7) `writeCTFMRI` : Creates MRI files that are compatible with MRIVIEWER.
- (8) `readCTFhdm`: Reads multiple-local spheres head-model files.
- (9) `writeCTFhdm`: Creates head-model files that are compatible with CTF MEG™ Analysis Software (DipoleFit and SAM beamformer programs).

This document describes the use of these programs. The formats of CTF files are described in document "CTF MEG™ File Formats", PN900-0088.

These programs have been verified to transfer CTF datasets to and from MATLAB correctly (MATLAB6.5 running under Windows XP), but within MATLAB anything can be done to the data. **Consequently, these programs must not be used for clinical applications of MEG data. Datasets created by `writeCTFds` and files created by `writeCTFMRI` and `writeCTFhdm` must not be used later in clinical applications, even with software that has received clearance for clinical applications.**

Revision 1.1:

- (1) `getCTFdata`, `writeCTFds` handle data arrays exceeding 2 GB.
- (2) `addCTFds` allows addition of trials to an existing dataset.
- (3) `ReadCTFhdm` and `writeCTFhdm` handle both v5 and v6 head-model files.

Revision 1.2:

- (1) `readCTFds` and `writeCTFds` have been changed to accomodate fMEG .hc files.
- (2) A minor change is made to handle text strings when writing CPerist objects.
- (3) `readCPersist` has been changed so it will not give an error when a CPerist file has extra bytes after the final 'EndofParameters' field.

Revision 1.3:

- (1) Changes in `readHc`, `readClassFile`, `readMarkerFile` to make it work in MATLAB 7.2.
- (2) `readHc` no longer fails when it finds physically improbable coil positions.



Installation

2. Installation and Help

2.1 Installation

The codes are provided as a .zip file, on a floppy disk, or on a CD (PN530-0036 MATLAB Import/Export Routines: Program Files). To make them available for calls from MATLAB, follow the standard procedure:

- (1) Create a folder on the computer that is running MATLAB. For example, in Windows the folder could be **C:\CTF-MATLAB** or **C:\MATLAB\toolbox\CTFDataSet**. On a Linux system, you might create **/usr/local/matlab/toolbox/CTFDataSet**.
- (2) Copy the MATLAB programs to this folder.
- (3) Start MATLAB and use the Set Path command available in the Files Menu to add the folder to the path.

2.2 Help

Typing the name of any of the programs documented here gives a brief summary of the program and a list of input and output arguments. For example typing

```
getCTFdata;
```

produces the response

```
getCTFData: Version 1.1 13 April 2007  Reads a CTF dataset into array data.
Call : data=getCTFdata(ds,dataList,chanList,unit,prec);
      - dataList: <=0, [] or missing.  Read the complete data set.
              size(dataList)=[1 N].  Read the N trials listed.
              dataList=[pt1 pt2 Tr]'.  Read points pt1:pt2 of trial Tr.
```



Reading Data

3. Reading CTF data into MATLAB

3.1 readCTFds - reading the dataset description

The user issues the MATLAB command

```
ds=readCTFds(datasetname) ;
```

where **datasetname** is the name of the dataset (i.e. the directory containing the dataset). **readCTFds** creates structure **ds** which holds the information that describes the dataset. Table 1 (page 12) lists the fields of **ds** and the corresponding dataset files. For analysis in MATLAB, usually only the **.res4**, and possibly the **.hc**, fields are used, but the additional fields are present to supply information for dataset files when subsequently creating a dataset using **writeCTFds**.

ds.res4 holds the parameters that describe the data including sample rate, trial size, channel names, and sensor parameters (sensor positions, gains and gradiometer balancing). Appendix A describes the fields of structure **ds** in detail and Appendix B describes **ds.res4** in detail.

For a 275-channel CTF system, the size of structure **ds** is about 4 MB, of which 3.6 MB is the **.res4** field.

3.1.1 No clinical use of readCTFds

On the first call to **readCTFds** in a MATLAB session (or on the first call since issuing the commands **clear all** or **clear readCTFds**), the following message is displayed on the monitor:

```
readCTFds: You are reading CTF data for use with a software-application tool  
that is not manufactured by VSM MedTech Ltd. and has not received marketing  
clearance for clinical applications. If CTF MEG data are processed by this tool,  
they should not be later employed for clinical and/or diagnostic purposes.
```

MATLAB Import/Export Routines

Table 1. List of fields of structure `ds` produced by `readCTFDs`

`readCTFDs` always produces fields `.baseName`, `.path`, `.res4` and `.meg4`. The other fields are created only if the files listed are actually present in the dataset. `getCTFDdata` uses fields `baseName`, `.path` and `.res4`, and file `baseName.meg4`. The Table shows which fields are required by `writeCTFDs`. Appendix A describes the fields.

ds field	Dataset file	Required by <code>writeCTFDs</code>	Information/Description
<code>.baseName</code>		x	Dataset is directory [<code>path</code> , <code>baseName</code> , ' <code>.ds</code> ']
<code>.path</code>		x	
<code>.res4</code>	<code>baseName.res4</code>	x	Description of the data including sensor gains, balancing data, and trial structure. Central to any analysis of the dataset.
<code>.meg4</code>	<code>baseName.meg4</code>	x	Data size (bytes) and 8-character header
<code>.infods</code>	<code>baseName.infods</code>	x	Dataset information. If <code>.infods</code> is not present, <code>writeCTFDs</code> will create one.
<code>.newds</code>	<code>baseName.newds</code>		Obsolete dataset information file. Included for compatibility with older CTF software.
<code>.acq</code>	<code>baseName.acq</code>		Acquisition parameters.
<code>.hist</code>	<code>baseName.hist</code>		Text file describing acquisition and processing history.
<code>.hc</code>	<code>baseName.hc</code>		Head coil positions
<code>.eeg</code>	<code>baseName.eeg</code>		EEG electrode channels, names and positions.
<code>.mrk</code>	<code>MarkerFile.mrk</code>		Event markers. Generated by Acq or by DataEditor.
<code>.TrialClass</code>	<code>ClassFile.cls</code>		Trial classes. Generated by Acq or by DataEditor.
<code>.badSegments</code>	<code>bad.segments</code>		Marks bad segments of data. Generated by DataEditor.
<code>.Virtual</code>	<code>VirtualChannels</code>		Virtual channels specified by DataEditor.
<code>.BadChannels</code>	<code>BadChannels</code>		List of channels known to be bad.
<code>.processing</code>	<code>processing.cfg</code>		DataEditor processing parameters (filtering and balancing).

MATLAB Import/Export Routines

3.2 getCTFdata - reading the data

Data are read from the dataset's **.meg4** file and converted to user-specified units by **getCTFdata**. The command is

```
data=getCTFdata(ds,data_list,chan_list,unit,prec);
```

where the input arguments are:

ds: The structure produced by **readCTFds**. (Only the first three fields in Table 1 are used by **getCTFdata**.)

data_list: Tells **getCTFdata** which data to read.

- (1) **data_list** ≤ 0 or [] or not defined. Read the complete dataset.
- (2) **size(data_list)=[1 N]**. Read the **N** trials listed in row vector **data_list**.
- (3) **size(data_list)=[3 1]**. Read points **data_list(1):data_list(2)** from trial **data_list(3)**.

chan_list: Optional list of channels to read. If not specified, or if **chan_list=[]**, then all channels are read. **chan_list** may be specified as

- (1) A list of channel numbers (i.e. a vector of doubles).
- (2) As a list of character channel names (i.e. a character array).
chan_list(n,:) is the name of the **n**th channel to be returned in array data.
getCTFdata compares the characters preceding the first blank or '-' character to the rows of **ds.res4.chanNames**.

unit: Controls the conversion from data in file baseName.**.meg4** to user-specified units.

- **'ft'**, [] or not defined: SQUIDs in fT, EEGs in μ V, ADC's and DACS in V, HLC channels in m, clock channels in s.
- **'T'**: SQUIDs in T, EEGs in V, ADC's and DACS in V, HLC channels in m.
- **'phi0'**: SQUIDs in ϕ_0 , EEGs in μ V, ADC's and DACS in V, HLC channels in m.
- **'int'**: No conversion. All channels read as signed 32-bit integers.

prec: Allows the user to specify 'single' in order to reduce memory requirements. If not specified or if set to anything other than 'double' or 'single', it returns an error message and sets **data=[]**.

All except the first input argument are optional.

getCTFdata reads the **.meg4** file of a dataset, and closes it. The data are produced as a 3-index array: **size(data) = [no_samples no_channels no_trials]**.

getCTFdata does not alter the balance of the signals recorded in the dataset. Field **grad_order_no** of structure array **ds.res4.senres** gives the balance of the recorded data. Function **setCTFDataBalance** described below is used to change data balance.

MATLAB Import/Export Routines

3.2.1 Examples of the application of `getCTFdata`:

- (1) Read the entire `meg4` file into a double array, converting the SQUID channels to fT, use the command

```
data=getCTFdata(ds);
```

- (2) Read points 1:10000 of trial 1 of the MEG channels in ϕ_0 units. Usually, MEG channels are distinguished by starting with 'M'.

```
MEGlist=strmatch('M',ds.res4.chanNames);  
MEGdata=getCTFdata(ds,[1 10000 1'],'MEGlist','phi0');
```

3.3 `setCTFDataBalance` - balancing the data

External interference in the MEG gradiometers is reduced by applying balancing to the raw MEG signals. Before applying balancing to a dataset, check the balance of the recorded data by examining `ds.res4.senres(k).grad_order_no` where `k` is a MEG channel. If `grad_order_no=3`, then the MEG signals are already balanced.

The complete set of balance coefficients for a MEG sensor is contained in structure array `ds.res4.scrs` which is produced by `readCTFds`. `setCTFDataBalance` takes an array of data, reads the balance table which is stored as structure array `ds.res4`, and changes the data balance.

The possible balance states of the MEG sensors are

- 'none' : Unbalanced data (`grad_order_no=0`)
- 'G1BR' : 1st order balancing (`grad_order_no=1`)
- 'G2BR' : 2nd order balancing (`grad_order_no=2`)
- 'G3BR' : 3rd order balancing (`grad_order_no=3`)

The possible balance states of the reference gradiometers are

- 'none' : Unbalanced data,
- 'G1BR' : 1st order balancing

There are no situations where balanced reference gradiometers are needed in MEG analysis and CTF software assumes unbalanced references. Therefore, to prevent passing erroneous reference gradiometer information to other datasets, **`writeCTFds` will not create datasets with balanced reference gradiometers.**

MATLAB Import/Export Routines

The MATLAB command to change data balance is

```
[data2,ds2]=setCTFDataBalance(data1,ds1,bal2,unit,chan_list,messages);
```

Inputs: **data1**: The current data (e.g. the array produced by `getCTFdata`).

ds1: **ds1** is a structure produced by `readCTFds` that provides the balance coefficients for rebalancing the data and also what is the balance state of **data1**.

bal2: Character array specifying the new MEG and reference gradiometer balance.
Example: If the user wants order-3 MEG balance and unbalanced reference gradiometers, then `bal2=strvcat('G3BR','none')`.

If the reference-gradiometer balance is omitted from **bal2** (e.g. `bal2='G2BR'`), then `setCTFDataBalance` assumes reference-gradiometer balance='none'.

If `bal2=[]`, then it is set to `strvcat('none','none')` (i.e. **data2** will be unbalanced).

unit: Optional input. Options: 'fT', 'T', 'phi0', 'int'. If unit is omitted from the input arguments, or `unit=[]`, it will be set to the default: `unit='fT'`.

chan_list: Optional input. List of dataset channels included in array **data1** (numbering referred to the channel numbering in structure **ds**). If `chan_list=[]` or `<0` or **chan_list** is omitted from the list of inputs, it is assumed that array **data1** has dataset channels `1:size(data,2)` (i.e. channel numbered consecutively from 1). If **chan_list** is included, `length(chan_list)=Nchan` where `size(data1)=[Npt Nchan Ntrial]`. **chan_list** must include the reference sensors required for balancing.

messages: Optional input. If omitted or `messages=[]`, set `messages=1`.

`messages=0`: Do not print tracing message.

`=1`: Print a message when the requested balance changes.

`=2`: Always print a message.

Outputs: **data2**: Rebalanced data.

ds2 **ds** structure for the rebalanced data. It is the same as **ds1**, but `grad_order_no` in `ds.res4.senres` is set to the new balance order.

`setCTFDataBalance` assumes that all of the MEG sensors also have the same balance (usually `grad_order_no=0` or `3`). However, it is possible for the coefficients for some of the MEG channels to be missing from the balance table. When this occurs, it is not possible to balance the missing channel. If `setCTFDataBalance` is requested to balance a channel missing from the table, it sets data for the channel to 0, and prints a message when `messages=1` or `2`.

If some of the MEG channels or reference gradiometers have a different `grad_order_no` than the majority of the class of channels, these channels are considered bad, and `setCTFDataBalance` sets the corresponding part of **data2** to zero, and adds the channels to `ds.BadChannels`.

MATLAB Import/Export Routines

3.3.1 Examples of the application of `setCTFDataBalance`:

- (1) Array `data3` is in fT units and has 3rd-order balanced MEGs, and unbalanced reference gradiometers. To convert to unbalanced MEG data, while keeping a copy of the balanced data,

```
[data0,ds0]=setCTFDataBalance(data3,ds3,'none');
```

- (2) Array `data` is in ϕ_0 units and has unbalanced SQUID data. To convert to 3rd-order balanced MEGs and unbalanced reference gradiometers,

```
[data,ds]=setCTFDataBalance(data,ds,'G3BR','phi0');
```

This replaces `data` with balanced MEGs, and updates structure `ds` so it shows the new balance (G3BR).

- (3) A more complicated example is calculating the balanced response of MEG sensors to modeled sources. To model the SQUID response, start with the sensor descriptions in structure array `ds.res4.senres`. `MEGlist` and `Reflist` are lists of the MEG and reference sensors referred to the dataset channel numbering. Field `grad_order_no` of `ds.res4.senres` must be 0 for all MEG sensors and reference gradiometers.

Let $L0(k,n)$ be the calculated unbalanced response of sensor `MEGlist(k)` to source n ($k=1,\dots,\text{length}(\text{MEGlist})$, `size(L0)=[nMEG nSource]`). $R(p,n)$ is the response of sensor `Reflist(p)` to source n . Then the balanced MEG response `L3` would be calculated with the following MATLAB instructions:

```
chan_list=[Reflist MEGlist];  
[L3,ds3]=setCTFDataBalance([R L0]',ds,'G3BR',[],chan_list,0);  
L3=L3(:,length(Reflist)+[1:length(MEGlist)],:);
```

It is necessary to use the `chan_list` option in this case because the forward solution `[R;L0]` does not include the non-SQUID channels of the dataset (e.g. clock, trigger, EEG, and ADC channels). Note the matrix transpositions that are used to make the channel become the second index in the argument of `setCTFDataBalance`.

3.4 `getCTFBalanceCoefs` - Balancing data using the table of balance coefficients

Example 3 of the application of `setCTFDataBalance` in Section 3.3 showed how to calculate the response of balanced MEG sensors to model sources. An alternative approach is to read the balance table from structure `ds.res4.scr` using function `getCTFBalanceCoefs` and then balance the model solution directly.

To read the balance coefficients for the MEG channels:

```
[alphaMEG,MEGlist,Refindex]=getCTFBalanceCoefs(ds,balance_type,unit);
```


MATLAB Import/Export Routines

To read the balance coefficients for the MEG channels and the reference gradiometers.

```
[alphaMEG,MEGlist,Refindex,alphaGref,Greflist,Gbalanceindex]=  
    getCTFBalanceCoefs(ds,balance_type,unit);
```

There are no situations where balanced reference gradiometers are needed in MEG analysis and CTF software assumes unbalanced references. To prevent passing erroneous reference gradiometer information to other datasets, **writeCTFDs** will not create datasets with balanced reference gradiometers.

Inputs: - **ds** is the structure that describes the dataset (output of **readCTFDs**).

- **balance_type** is a character array that specifies the balance type. The balance options are 'none', 'G1BR', 'G2BR', 'G3BR'.
 - **size(balance_type)=[1 4]**. If the reference-gradiometer balance table is requested by giving 6 output arguments, **getCTFBalanceCoefs** extends **balance_type** by adding **balance_type(2,:)= 'none'**.
 - **balance_type(1,:)**: MEG-gradiometer balance table. The options are 'none', 'G1BR', 'G2BR', 'G3BR'.
 - **balance_type(2,:)**: Reference-gradiometer balance table. The options are 'none', 'G1BR'.
- **unit**: Options: 'fT', 'T', 'phi0', 'int'. If unit is omitted from the input arguments, or **unit=[]**, it will be set to the default **unit='fT'**.

Outputs : The MATLAB command must specify either 3 or 6 output arguments.

- **alphaMEG, MEGlist, Refindex**. The MEG gradiometer balance table. **MEGlist** is the list of the MEG sensor channels that appear in the balancing table (N_{MEG} channels) referred to the dataset channel numbering (i.e. the channel numbering in structure **ds**). **Refindex** is the list of N_{BAL} reference SQUID channels that are used for balancing MEG channels. **Refindex** is a subset of the list of all reference channels ($N_{\text{BAL}} \leq N_{\text{REF}}$) and is referred to the dataset channel numbering. **alphaMEG** is an $N_{\text{BAL}} \times N_{\text{MEG}}$ array of balance coefficients. **alphaMEG(k,m)** is the coefficient of reference sensor **Refindex(k)** for balancing MEG gradiometer **MEGlist(m)**.
- **alphaGref, Greflist, Gbalanceindex**. The reference gradiometer balance table. **Greflist** is a list of all reference gradiometers (N_{GREF} channels). **Gbalanceindex** is the list of N_{GBAL} reference channels that are used for balancing reference gradiometers. **Gbalanceindex** is a subset of the list of all reference channels. **alphaGref** is the $N_{\text{GBAL}} \times N_{\text{GREF}}$ array of balance coefficients for the reference gradiometers. **alphaGref(k,m)** is the coefficient of reference sensor **Gbalanceindex(k)** for balancing reference gradiometer **Greflist(m)**. Channel numbers are referred to the channel numbering in structure **ds**.
- When the input **balance_type(k,:)= 'none'** (**k=1 or 2**) is selected, **getCTFBalanceData** returns lists of sensors, but the balance matrix and balance-reference list are empty:

MATLAB Import/Export Routines

- `balance_type(1,:)='none' : alphaMEG=zeros(0,nMEG) , Refindex=[],`
`MEGlist = list of MEG channels`
- `balance_type(2,:)='none' : alphaGref=zeros(0,nGref) , Refindex=[],`
`Greflist = list of reference gradiometers.`

3.4.1 Examples of application of `getCTFBalanceCoefs`

- (1) To perform the forward-solution balancing described in Example 3 in Section 3.3 above where **R** is the reference-sensor response, **L0** is the unbalanced MEG response, **Reflist** is a list of reference sensors referred to dataset channel numbering, and the sensor is described by structure **ds**, use the following commands:

```
[alphaMEG,MEGlist,Refindex]=getCTFBalanceCoefs(ds,'G3BR');  
[Cx,Refx]=intersect(Reflist,Refindex);  
L3=L0-alphaMEG'*R(Refx,:);
```

Note that the transpose of **alphaMEG** is used since `size(alphaMEG)=[nBal nMEG]` while `size(L0)=[nMEG nSource]`.

- (2) Suppose array **data** is a single trial of unbalanced data read by `getCTFdata` with option `unit='int'` and `chan_list=[]` (i.e. all of the dataset channels are present in array **data**). Then the MEG channels could be converted to order-3 balance by the following commands:

```
[alphaMEG,MEGlist,Refindex]=getCTFBalanceCoefs(ds,'G3BR','int');  
data(:,MEGlist)=data(:,MEGlist)-data(:,Refindex)*alphaMEG;
```



Writing Data

4. Writing data from MATLAB to CTF format

Datasets created by `writeCTFds` must not be used for clinical applications.

4.1 No clinical use

The MATLAB programs supplied by VSM have been verified for correct operation, but it is not possible to know how data have been manipulated in MATLAB before being written as a CTF dataset by `writeCTFds`. `writeCTFds` adds "NOT FOR CLINICAL USE" messages to text fields in the `.res4`, `.infods`, `.hist`, and if present `.news` files of the new dataset.

In addition `writeCTFds` prints the following message on the monitor the first time it is called in a session, or after a `clear all` or `clear writeCTFds` command:

```
writeCTFds: The data you are writing have been processed by software not
            manufactured by VSM MedTech Ltd. and that has not received marketing clearance
            for clinical applications. These data should not be later employed for clinical
            and/or diagnostic purposes.
```

4.2 Data preparation for `writeCTFds`

It is expected that the user will start by reading a dataset using `readCTFds` and `getCTFdata` (Sections 3.1 and 3.2). This will create the `ds` structure for the dataset including the `res4`, and `meg4` and `infods` fields, plus a field for each of the other files found in the dataset as listed in Table 1.

The user will manipulate the data in MATLAB. This may include balancing the data (Section 3.3), adding test signals to existing data, adding or removing channels or trials from the dataset, and possibly filtering and subsampling the data. The output data may be either single or double precision.

Before writing the dataset, the `ds.res4` information must be updated as listed below :

- (a) **Filtering.** Increase `ds.res4.num_filters` to the correct number of stages. As an example, if a single filter was applied to the data, the `res4` structure would be changed as follows :

MATLAB Import/Export Routines

```
ds.res4.num_filters=ds.res4.num_filters+1;
```

and then field **filters** would be augmented as follows for filter frequency **f0**:

Lowpass filter: `ds.res4.filters(ds.res4.num_filters)=...
struct('freq',f0,'fClass',1,'fType',1,'numParam',0);`

Highpass filter: `ds.res4.filters(ds.res4.num_filters)=...
struct('freq',f0,'fClass',1,'fType',2,'numParam',0);`

Notch filter, width=**fwidth**:

```
ds.res4.filters(ds.res4.num_filters)=...  
struct('freq',f0,'fClass',1,'fType',3,'numParam',1,'Param',fwidth);
```

The only allowed value of **fClass** is 1. Set **fType**=1 for lowpass, 2 for highpass and 3 for notch filtering. This assumes that the filters defined in the **.res4** file have no extra parameters, and that only Butterworth filters are applied to the data.

CTF's DataEditor does not make use of this information, but it is a simple way to record the data in the **.res4** file of the dataset, and it is consistent with the way that the data-acquisition software records the filtering information. **writeCTFds** uses the information in **ds.res4.filters** to update the bandwidth parameters in the **.newds** and **.infods** files of the output dataset, and also makes a notation in the **.hist** file.

If **ds.res4.filters** is not defined, then the **.infods** file of the dataset will have bandwidth set to 0 to (sample_rate)/4.

(b) Subsampling. Adjust parameter **ds.res4.sample_rate** to match the sample rate in the data array.

(c) Truncating data. Simply remove trials or samples as required. **writeCTFds** will set

```
[no_samples no_channels no_trials]=size(data)
```

and will adjust the relevant fields of the **.res4** structure.

(d) Removing channels. If a channel is removed from the dataset, then remove the corresponding channel from **ds.res4.chanNames** and **ds.res4.senres**. For example if channels 100:200 have been removed from the dataset, make the following changes :

```
keeplist=setdiff(1:ds.res4.no_channels,[100:200]);  
ds.res4.chanNames=ds.res4.chanNames(keeplist,:);  
ds.res4.senres=ds.res4.senres(keeplist);  
ds.res4.no_channels=size(ds.res4.chanNames,1);
```

MATLAB Import/Export Routines

writeCTFds will detect the absence of these channels, and they will be removed from the balance coefficient arrays in the output dataset. If **writeCTFds** detects a **ds.res4.chanNames** or **ds.res4.senres** table that is longer than the number of channels in array data, it issues an error message since **writeCTFds** does not know which channels have been removed. It is not really necessary to adjust **ds.res4.no_channels** since **writeCTFds** reassigns it on the basis of the size of the data array.

(e) **Adding channels.** Suppose one new channel is created and that it is added at the end of the array data. Before adding channel,

```
size(data)=[Ndat Nchan Ntrial]
size(newdata)=[Ndat 1 Ntrial]
```

To add data,

```
data(:,Nchan+1,:)=newdata;
ds.res4.chanNames=[ds.res4.chanNames;(new 32-character name)];
ds.res4.senres(Nchan+1)=...
```

The new channel name must include a null byte **char(0)** to terminate the string.

If a user does not provide names for new channels or entries in the **senres** table, **writeCTFds** will notice a difference between the lengths of array **ds.res4.chanNames** and the data array and will add names **MXT001**, **MXT002**,... at the end of **ds.res4.chanNames** (i.e. it will assume that the new channels are at the end of array data). **writeCTFds** also extends the **ds.res4.senres** table for the new channels by creating channels with **sensorTypeIndex=4**, a nominal gain of $3 \times 10^9 \phi_0/T$ to the channel, and a position 21 cm above the centre of the MEG helmet (i.e. typical MEG gradiometer gains, but the system treats it as a magnetometer, and it's position clearly marks it as different).

writeCTFds does not add balancing coefficients for the new channels, so DataEditor will not adjust the new channels when different gradient-balancing orders are requested.

Generally, it will be better if users do not depend on this default approach, but rather supply a description for added channels by making entries in **ds.res4.chanNames** and **ds.res4.senres**.

MATLAB Import/Export Routines

- (f) **Saving balanced data.** The user tells CTF software (including DataEditor) the balance of channel **k** (a type=5 MEG sensor) by setting

```
ds.res4.senres(k).grad_order_no=p;
```

where **p**=0,1,2,3 is the gradiometer balancing order. All MEG channels (i.e. gradiometers with **sensorTypeIndex**=5) must have the same balance order, and a MEG channel with a different balance order is considered a bad channel and is added to the list of bad channels in **ds.BadChannels**. All of the reference gradiometers must have **grad_order_no**=0.

If balancing is done using **setCTFDataBalance** (Section 3.3) with the **ds** structure as an output argument, then **grad_order_no** will be set to the correct value by **setCTFDataBalance**.

- (g) **Changing text information.** The following text strings appear in the **.res4** file and structure **ds.res4**. If they are not changed, the strings read from the original dataset will be written into the output **.res4** file. **writeCTFds** checks the lengths of the strings before writing them in order to maintain the correct **res4** file format. If a string is too long the trailing characters will be removed except for fields marked with ***** where leading characters are removed in order to ensure that the message "NOT FOR CLINICAL USE" is not removed from the field. Four of the **.res4** fields are also transferred to the **infods** file.

res4 field	max. chars	infods tag
* ds.res4.appName	256	
* ds.res4.dataOrigin	256	
* ds.res4.dataDescription	256	
ds.res4.data_time	255	
ds.res4.data_date	255	
ds.res4.nf_run_name	32	
* ds.res4.nf_run_title	256	_DATASET_PROCSTEPTITLE
ds.res4.nf_instruments	32	
ds.res4.nf_collect_descriptor	32	_DATASET_PROCSTEPDESCRIPTION
* ds.res4.nf_subject_id	32	_DATASET_PATIENTID
ds.res4.nf_operator	32	_DATASET_OPERATORNAME
ds.res4.nf_sensorFileName	56	
ds.res4.run_description	no limit	

In all cases, **res4** strings are terminated by a null byte (**char(0)**).

In addition, text changes in the **.newsds**, **.hist** and **.infods** fields of **ds** (including the message "NOT FOR CLINICAL USE") will appear in the output dataset.

MATLAB Import/Export Routines

4.3 Creating datasets with `writeCTFds`

The MATLAB command is

```
newds=writeCTFds(datasetname,ds,data,unit);
```

Inputs :

- datasetname**: Name of dataset (i.e. directory) to be created including the complete path. The extension **.ds** is optional: if it is missing, **writeCTFds** will add it.
- ds** : The structure that describes the dataset and is created by **readCTFds**. It must include the fields shown in Appendix A. Section 4.2 describes how to change the **ds.res4** structure for a modified dataset.
- data** : The array of data to be written to the new **.meg4** file. It may be a single or double precision array.
- size(data)=[Ndat Nchan Ntrial]**
- unit** : Described in the documentation for **getCTFdata**. This controls the conversion of array **data** into 32-bit integers which are written to file **datasetname.meg4**. If not included in the call, or if **unit=[]**, **writeCTFds** assumes **unit='ft'**. The options are
- (i) **'ft'** or **'FT'**
 - (ii) **'t'** or **'T'**
 - (iii) **'int'**
 - (iv) **'phi0'**

The first three input arguments (**datasetname**, **ds**, **data**) must be included. Argument **unit** may be omitted, if the SQUID data are in ft units.

Outputs :

- **newds**: The **ds** structure for the new dataset.
- CTF dataset **datasetname**.

The new CTF-format dataset contains one file for each field of structure **ds** as listed in Table 1. (p.12). If **ds** does not include a **.hist** field, it is added and indicates that the dataset was created by **writeCTFds** rather than CTF MEG™ software. The bandwidth information in files ***.infods** and ***.newds** is adjusted to reflect new filters and subsampling specified by the user in **ds.res4**. The new dataset will not include **hz.ds** or **hz2.ds** subdirectories which contain the signals recorded during pre- and post-run head localization. **writeCTFds** also adds messages indicating that the dataset was produced by **writeCTFds** and not by CTF MEG™ software and that the data must not be used for clinical purposes.

Version 1.2 of **writeCTFds** will create multiple meg4 files if the total data size exceeds 2 GB.

MATLAB Import/Export Routines

4.4 Adding trials to existing datasets with `addCTFtrial`

It is not uncommon for the size of a dataset to exceed the memory available for MATLAB in a computer running Windows. As an example, with a 2 GB machine memory, MATLAB will encounter memory problem if the dataset exceeds ~100 million points. In this case, if a user wants to analyze a dataset and then create a new output dataset of the same size, it will be necessary to process the data in segments, and then to add the results of later segments to the dataset created from the earlier ones. `addCTFtrial` adds one or more trials to a dataset previously created by `writeCTFds`. `addCTFtrial` will not add trials to datasets that were not created by `writeCTFds`.

The MATLAB command is

```
cntrl=addCTFtrial(datasetname,data,unit,mrk,cls,badSegments);
```

Inputs :

- datasetname**: Name of dataset (i.e. directory) to be created including the complete path. The extension `.ds` is optional.
- data** : The array of data to be added to the existing `.meg4` file. It may be a single or double precision array.
`size(data)=[Ndat Nchan Ntrial]`
where **Ntrial** is the number of trials being added.
- unit** : Described in the documentation for `getCTFdata` and `writeCTFds`.
- mrk** : Markers for the trials being added. Trial=1 is the first trial being added.
- cls** : Classes for the trials being added. Trial=1 is the first trial being added.
- badSegments**: Bad segment markings for the trials being added. Trial=1 is the first trial being added. See description of `ds.BadSegments` in Appendix A.15.

Output: cntrl: 0 = failure.
1 = success.



MRI and Head Model Files

5. Reading and creating MRI and head-model files

Analysis of MEG data often requires analysis of the MRI of a subject to create a head model as part of the calculation of the magnetic field of point dipole in the brain (the "forward solution"), or to establish the anatomical location of a source determined by MEG analysis. Programs have been prepared to allow users to read MRIs and head models into MATLAB and to create files that are compatible with CTF MEG™ software (e.g. **MRIVIEWER**, **DipoleFit**, and the SAM beamformer programs). These files may be especially useful for users who need to convert MRIs in other formats to CTF MRI format, or who wish to do multiple-local-spheres forward solutions in MATLAB and need access to the model parameters calculated by Linux program **localSpheres**.

The formats of MRI and head-model files are described in document "CTF MEG™ File Formats", PN900-0088.

5.1 Reading CTF MRIs with **readCTFMRI**

A CTF MRI can be read into MATLAB by the following command:

```
[MRItags,MRIdata]=readCTFMRI (MRIfile) ;
```

where

MRIfile is the name of a v4 CTF-format MRI including the full path and extension **' .mri '**. CTF MRI files (version 4) are in the CPersist format (like **.infods** and **.acq**)

MRItags is a structure array listing the names, types and contents of the tags found in the file. The tags are listed in the reference.

MRIdata is a 256x256x256 **int16** array containing the MRI. **MRIdata(j,k,n)** refers to coronal slice **j**, axial slice **k**, sagittal slice **n**. **j=1**: anterior, **j=256**: posterior, **k=1**: superior, **k=256**: inferior, **n=1**: left, **n=256**: right.

Tag **'CTFMRI_TRANSFORMMATRIX'** is a text string giving the 4x4 matrix that connects positions in CTF Head Coordinates and MRI pixel coordinates. The matrix elements are separated by a **'\'** character:

```
string='M11\M12\M13\...\M43\M44'.
```

Let d = pixel size. Then the relation between physical positions (x,y,z) in CTF Head Coordinates and MRI pixel coordinates (j,k,n) is

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x/d \\ y/d \\ z/d \\ 1 \end{bmatrix} = \begin{bmatrix} j \\ k \\ n \\ 1 \end{bmatrix}$$

MATLAB Import/Export Routines

(In this matrix $M_{41}=M_{42}=M_{43}=0$, $M_{44}=1$ and $[M_{14} \ M_{24} \ M_{34}]$ = pixel coordinates of the coordinate-system origin). CTF Head Coordinates are defined in Appendix A of the document "CTF MEG™ File Formats", PN900-0088.

5.1.1 No clinical use of `readCTFMRI`

`readCTFMRI` prints the following message on the monitor the first time it is called in a session, or after a `clear all` or `clear readCTFMRI` command:

```
readCTFMRI: You are reading a CTF-format MRI for use with a software-application
tool that is not manufactured by VSM MedTech Ltd. and has not received marketing
clearance for clinical applications. If the MRI is processed by this tool,
it should not be later employed for clinical and/or diagnostic purposes.
```

5.2 Creating CTF MRIs with `writeCTFMRI`

`writeCTFMRI` reverses the operation of `readCTFMRI` and produces an MRI file compatible with MRIVIEWER. The call is

```
writeCTFMRI (MRIfile,MRItag,MRIdata) ;
```

where

MRIfile is the name of the file to be created including the full path and extension '**.mri**'.

MRItags is a structure array in the format produced by `readCTFMRI` listing the names, types and contents of the tags found in the file. The tags are listed in the MRI format section of document "CTF MEG™ File Formats", PN900-0088.

MRIdata is a 256x256x256 array containing the MRI. **MRIdata(j,k,n)** refers to coronal slice **j**, axial slice **k**, sagittal slice **n**. **j=1**: anterior, **j=256**: posterior, **k=1**: superior, **k=256**: inferior, **n=1**: left, **n=256**: right. **MRIdata** is converted to int16 when it is written to file **MRIfile**.

This program will be useful for users who need to modify the CTF MRI (for example, by expanding or shrinking it, or by processing it to remove artifacts in the MRI), or who need the MRI to prepare plots in MATLAB. Alternatively, `writeCTFMRI` provides a way to convert MRIs in other formats (e.g. AFNI) to CTF MRI format if the other formats can be read into MATLAB.

MATLAB Import/Export Routines

5.2.1 No clinical use of `writeCTFMRI`

The message "NOT FOR CLINICAL USE" is added to the patient identification and comment tags in the output MRI file. In addition `writeCTFMRI` prints the following message on the monitor the first time it is called in a session, or after a `clear all` or `clear writeCTFMRI` command:

```
writeCTFMRI: The MRI you are creating has been processed by software not
             manufactured by VSM MedTech Ltd. and that has not received marketing clearance
             for clinical applications. This MRI should not be later employed for clinical
             and/or diagnostic purposes.
```

5.3 Reading CTF head-model files with `readCTFhdm`

CTF MEG™ program `DipoleFit` and the SAM beamformer software use single or multiple local sphere models of the subject's head. These models are based on the positions of head coils and the shape of the head determined from the MRI. Command-line program `localSpheres` produces a text file with extension `.hdm` (usually the filename is `default.hdm`) containing the centre and radius of the local sphere appropriate for each MEG sensor.

`readCTFhdm` reads the head-model file into a structure `hdm`. The head-model file is a text file in the CTF "Config Reader" format (see "CTF MEG™ File Formats", PN900-0088). The command is

```
hdm=readCTFhdm(hdmFile)
```

where

`hdmFile` is the name of the head-model file including the path and `.hdm` extension

`hdm` is a structure containing the information read from `hdmFile`. It has a field for each of the Config Reader classes identified in file `hdmFile`, and fields named `note` contain comment lines read from the file.

The most important part of structure `hdm` is `hdm.MultiSpheres_Data` which contains the list of SQUID sensor names and the local sphere for each. Version 5 head-model files have the following sub-fields :

```
hdm.MultiSphere_Data =
    note: [2x34 char]
    SEARCH_RADIUS: 7
    HEADSHAPE_FILE: 'MRI/s2_20051209_matched_fids_head.shape'
    SQUIDname: [301x5 char]
    sphereOrigin: [3x301 double]
    radius: [1x301 double]
```

Release 5.4 of the CTF software assumes version 6 head-model files in which two sub fields are added to `MultiSphere_Data`:

```
hdm.MultiSphere_Data.SURFACE_TYPE: 'SCALP' or 'BRAIN'
    .HEADPOS.HEADPOS: List of head coil positions in the dewar reference
                     frame (same information as ds.hc.dewar)
    .HEADPOS.NOMINAL: 'TRUE' or 'FALSE'
```

MATLAB Import/Export Routines

The **SEARCH_RADIUS** parameter is the radius (cm) of the head patch that is used to estimate the centre and radius of the local sphere. Character array **SQUIDname** is a list of all the SQUID sensors in the dataset that was used to create the head-model file. **sphereOrigin** and **radius** are the sphere parameters in cm for each of the SQUID sensors. The coordinate system is CTF Head Coordinates. The sensor positions in this coordinate frame are given in the **pos** field of the **ds.res4.senres** structure array (see Appendix B.2).

Command-line program **localSpheres** is described in more detail in documents "Command Line Programs Guide", PN900-0016, and SAMsuite Guide, PN900-0037.

5.3.1 No clinical use of readCTFhdm

readCTFhdm prints the following message on the monitor the first time it is called in a session, or after a **clear all** or **clear readCTFhdm** command:

```
readCTFhdm: You are reading a CTF head model for use with a software-application
tool that is not manufactured by VSM MedTech Ltd. and has not received marketing
clearance for clinical applications. If CTF MEG data are processed by this tool,
they should not be later employed for clinical and/or diagnostic purposes.
```

5.4 Creating CTF head-model files with writeCTFhdm

writeCTFhdm allows creation of a single- or multiple-local-spheres head model that can be used by **DipoleFit** and the SAM programs to generate forward solutions for MEG analysis. The command is

```
writeCTFhdm(hdmFile,hdm);
```

where the arguments have the same meaning as for the complementary program **readCTFhdm**:

hdmFile is the name of the head-model file including the path and **.hdm** extension

hdm is a structure in the form produced by **readCTFhdm** containing the information required to create a head-model file. It has a field for each of the Config Reader classes that will be created in **hdmFile**.

To create a multiple-local-sphere model, structure **hdm** must have a field **MultiSphere_Data** in the format discussed in Section 5.3 (**readCTFhdm**).

A single-sphere MEG model can be defined by field **MEG_Sphere**. An example is:

```
hdm.MEG_Sphere =
    note: ''
    ORIGIN_X: -2.0030
    ORIGIN_Y: -0.1110
    ORIGIN_Z: 4.6580
    RADIUS: 9.0850
```

This defines the origin in cm in CTF Head Coordinates.

writeCTFhdm produces both version 5 and version 6 hdm files depending on the presence of the new subfields in **hdm.MultiSphere_Data**.

MATLAB Import/Export Routines

writeCTFhdm requires fields **Model** and one or both of **MEG_Sphere** and **MultiSphere_Data** (see "CTF MEG™ File Formats", PN900-0088 for a description of the classes in .hdm files). Without these fields it is not a head-model file.

5.4.1 Converting head-model files from v5 to v6

v6 head-model files have two tags added to the **MultiSphere_Data** class and the version number changed in the **File_Info** class. This extra information is required by v5.4 Linux command **DipoleFit**, and if it is not present an error message will be printed. An example of a v5 text file (Config Reader format) is

```
File_Info
{
    VERSION:    CTF_HEAD_MODEL_FILE_VERSION_5.0
    DATE:       18-Oct-2006 16:05
    PATIENT:    none
    STATUS:
}
...
MultiSphere_Data
{
    SEARCH_RADIUS: 7.000
    HEADSHAPE_FILE:    MRI/head.shape

    // Multiple Sphere locations in cm
    //      X      Y      Z      Radius
    BG1:    -2.950 -0.038 5.544  8.365
    BG2:    -2.950 -0.038 5.544  8.365
```

The v6 file has tags for head-coil positions (in dewar coordinates) and a keyword to indicate whether this is a scalp or brain surface:

```
File_Info
{
    VERSION:    CTF_HEAD_MODEL_FILE_VERSION_6.0
    DATE:       19-Apr-2007 15:52
    PATIENT:    none
    STATUS:
}
...
MultiSphere_Data
{
    SEARCH_RADIUS: 7.000
    HEADSHAPE_FILE:    MRI/head.shape
    SURFACE_TYPE:      SCALP

    // Head Coil coordinates relative to dewar, cm
    //      NA_x  NA_y  NA_z    LE_x    LE_y    LE_z    RE_x  RE_y  RE_z    Nominals
    HEADPOS: 7.844 6.379 -22.727 -3.080  6.495  -25.079 6.766 -3.735 -24.147 FALSE
    // Multiple Sphere locations in cm
    //      X      Y      Z      Radius
    BG1:    -2.950 -0.038 5.544  8.365
    BG2:    -2.950 -0.038 5.544  8.365
```

MATLAB Import/Export Routines

The simplest way to change from v5 to v6 is to regenerate the head-model file using the v5.4 `localSpheres` command. A second way is to edit the v5 head model file with a Linux text editor to add the extra information as shown above. Note that Config Reader format separates fields by tabs. Finally, the following MATLAB script could be used where the head-model file name is

`default.hdm`:

```
ds=readCTFDs(dataset);
hdmFile=[dataset,filesep,'default.hdm'];
hdm=readCTFhdm(hdmFile);
hdm.File_Info.VERSION=' CTF_HEAD_MODEL_FILE_VERSION_6.0';
hdm.MultiSphere_Data.SURFACE_TYPE='SCALP';
hdm.MultiSphere_Data.HEADPOS=...
    struct('HEADPOS',reshape(ds.hc.dewar,9,1),'NOMINAL','FALSE');
delete(hdmFile);
writeCTFhdm(hdmFile,hdm);
```

5.4.2 No clinical use of `writeCTFhdm`

The message "NOT FOR CLINICAL USE" is added to the comment section at the start of the file. In addition `writeCTFhdm` prints the following message on the monitor the first time it is called in a session, or after a `clear all` or `clear writeCTFhdm` command:

```
writeCTFhdm: The head-model data you are writing have been processed by software
not manufactured by VSM MedTech Ltd. and that has not received marketing clearance
for clinical applications. These data should not be later employed for clinical
and/or diagnostic purposes.
```



Appendix A: Data Structure

Appendix A : The data structure produced by `readCTFds`

`ds` is the structure produced by `readCTFds`:

```
ds=readCTFds(dsName)
```

The fields of `ds` are listed in Table 1 (page 12). This Appendix describes the structure and information in each field of `ds` while Appendix B describes `ds.res4` in detail. The format and information in the dataset files are described in the document "CTF MEG™ File Formats", PN900-0088. The function of many of the fields is simply to store information that can later be used by `writeCTFds` to create a CTF-format dataset. For operations within MATLAB, only the `baseName`, `path` and `res4` fields are required.

A.1 `ds.baseName`

Character string containing the name of the dataset, without path information and without the `.ds` extension. E.g. : `ds.baseName='AbsCal_Noise_20050301_04'`.

A.2 `ds.path`

Character string containing the complete path for the dataset, including the final file-separator character. E.g.:

Windows : `ds.path= S:\CHL\NIMH_20060606\`

Linux: `ds.path=/home/hwilson/data/CHL/NIMH_20060606/`

A.3 `ds.res4`

The contents of the file [`baseName`, '`.res4`']. This field contains the description of the system sensors in substructure `ds.res4.senres`, and the balance tables in `ds.res4.scorr`. `ds.res4` is a complicated field, and Appendix B discusses `ds.res4` in detail. It is the most important part of `ds`. For a 275-channel CTF MEG system, structure `ds` has a size of about 4 MB, of which 3.6 MB is the `.res4` field.

MATLAB Import/Export Routines

A.4 ds.meg4

This gives the 8-bytes MEG4 file header and the size of the MEG4 file in bytes

E.g.: `ds.meg4=`

```
    fileSize: 961920008
    header:    'MEG41CP '
```

A.5 ds.infods

This is a structure array containing all of the information listed in the file `baseName.infods`. The file structure (CPersist format) is listed in Appendix B of "CTF MEG™ File Formats", PN900-0088. Each element of `ds.infods` contains the name, type and data of one of the fields in the `infods` file:

```
ds.infods = 1x63 struct array with fields:
    name
    type
    data
```

An example of one element of the structure array is

```
ds.infods(8) =
    name: '_PATIENT_ID'
    type: 10
    data: 'Anonymous subject'
```

The CPersist reading function also creates two data types in addition to the types listed in the reference:

`type=0` : The string `'WS1_'` that marks the start of a CPersist object.

`type=-1`: The string `'EndOfParameters'` that marks the end.

```
ds.infods(1) =          ds.infods(63) =
    name: 'WS1_'          name: 'EndOfParameters'
    type: 0               type: -1
    data: []              data: []
```

For data processing, the important `.infods` fields are the bandwidth fields

`__DATASET_LOWERBANDWIDTH` and `__DATASET_UPPERBANDWIDTH`

and the fields that describe head motion when Continuous Head Localization is applied:

```
__DATASET_HZ_MODE,
__DATASET_MOTIONTOLERANCE,
__DATASET_MAXHEADMOTION,
__DATASET_MAXHEADMOTIONTRIAL,
__DATASET_MAXHEADMOTIONCOIL.
```


MATLAB Import/Export Routines

A.6 ds.newds

ds.newds is a simple character image of the file **baseName.newds**. **writeCTFDs** updates the bandwidth information in **ds.newds**, indicates that the dataset was created by **writeCTFDs**, and adds the no-clinical-use message "NOT FOR CLINICAL USE" in the **appName** field.

ds.newds will have no application within MATLAB. If **newds** is not a field of **ds**, **writeCTFDs** will not indicate that **ds.newds** is missing. The **newds** file is obsolete, and is now replaced by the **infods** file.

A.7 ds.acq

This is a structure array containing all of the information listed in the file **baseName.acq** which describes the electronics and acquisition setup. The file structure (CPersist) format is listed in Appendix B of "CTF MEG™ File Formats", PN900-0088. Each element of **ds.acq** contains the name, type and data of one of the fields in the **baseName.acq**.

```
ds.acq=
    1x1319 struct array with fields:
        name
        type
        data
```

The **acq** file is read simply to enable **writeCTFDs** to create an **.acq** file in the output dataset. It is unlikely that MATLAB users will need to access **ds.acq**.

A.8 ds.hist

ds.hist is a simple character image of file **baseName.hist**. **writeCTFDs** adds a record of filter information from **ds.res4**, indicates that the dataset was created by **writeCTFDs**, and adds the message "NOT FOR CLINICAL USE".

ds.hist will have no application within MATLAB. If **hist** is not a field of **ds**, **writeCTFDs** will create it.

A.9 ds.hc

ds.hc contains the information from the head-coil position file **baseName.hc**. It records the head coil positions in three frames : "standard", "measured" and "head". "measured" gives the head coil positions in cm in the dewar coordinates (origin at the centre of the reference-sensor array, 21 cm above the centre of the MEG helmet. "head" gives head coil positions in CTF head coordinates. If there are 3 head coils in the **.hc** file then

```
ds.hc =
    names: [3x6 char]
    standard: [3x3 double]
    dewar: [3x3 double]
    head: [3x3 double]
```

MATLAB Import/Export Routines

`ds.hc.names` lists the head coils: `ds.hc.names =nasion`
left
right

`ds.hc.dewar(:,k)` is the position of coil `k` (name=`ds.hc.names(k,:)`), in the dewar reference frame.

When reading fMEG datasets, the field `hc.head` will be replaced by `hc.abdomen`.
`readCTFDs` and `writeCTFDs` will also handle `.hc` files with 4 head coils.

A.10 ds.eeg

If file `baseName.eeg` exists, `ds.res4.eeg` is a list of EEG channels (channel number in the dataset), channel names and electrode positions in cm in CTF Head Coordinates. If 32 EEG sensors were defined, then

```
ds.eeg = 1x32 struct array with fields:
    chanNum
    name
    pos
```

A sample entry is

```
ds.eeg(4) =
    chanNum: 190
    name: 'EEG004'
    pos: [4.6016 6.1766 3.2764]'
```

A.11 ds.mrk

`ds.mrk` is a structure array with one element for each marker set found in `MarkerFile.mrk`. An example of an element of `ds.mrk` is

```
ds.mrk(2) =
    ClassGroupId: 0
    Name: 'Tr18'
    Comment: 'Trigger Marker corresponding to bit 18'
    Color: 'Red'
    Editable: 'No'
    ClassId: 2
    BitNumber: 18
    Polarity: '-'
    Source: 'Source is not defined'
    Threshold: 0
    trial: [1x83 double]
    time: [1x83 double]
```

These field names are taken directly from the keywords in `MarkerFile.mrk` (see document "CTF MEG™ File Formats", PN900-0088, for the file format), but note the addition of `BitNumber`, `Polarity`, `Source` and `Threshold` for `ClassGroupID=0` markers. These four fields are empty for `ClassGroupID=3` markers.

Inside `MarkerFile.mrk`, trial numbering starts at 0, but in `ds.mrk` it starts at 1. This means it will be easy to match `ds.mrk(k).trial` to `data(:, :, k)` when the whole dataset is read, but if only a few

MATLAB Import/Export Routines

trials are read by `getCTFdata` (e.g. `data_list=2`), then it will be necessary to edit the trial and time arrays within `ds.mrk`.

A.12 `ds.TrialClass`

`readCTFds` forms `ds.TrialClass` from `ClassFile.cls`, if it exists. `ds.TrialClass` is a structure array with one element for each class set found in `ClassFile.cls`. `ds.TrialClass` has 7 fields:

```
ds.TrialClass = 1x2 struct array with fields:
    ClassGroupId
    Name
    Comment
    Color
    Editable
    ClassId
    trial
```

In `ClassFile.cls`, trial numbering starts at 0, but in `ds.TrialClass` it starts at 1. This means it will be easy to match `ds.TrialClass(k).trial` to `data(:, :, k)` when the whole dataset is read, but if only a few trials are read by `getCTFdata` (e.g. `data_list=2`), then it will be necessary to edit the trial arrays within `ds.TrialClass`.

A.13 `ds.badSegments`

If file `bad.segments` exists, `readCTFds` stores information about segments of data marked BAD in `ds.badSegments`. `bad.segments` is usually generated by DataEditor. An example is

```
ds.badSegments =
    trial: [2 3 5]
    StartTime: [0.5633 2.6400 1.2000]
    EndTime: [1.1400 3.7333 2.3167]
```

Trial numbering starts at 1. If `writeCTFds` finds this field in structure `ds`, it will create file `bad.segments`. This field can be used to mark artefacts or bad segments found during MATLAB analysis of the data, and then to pass the information to programs such as DataEditor or the SAM beamformer programs.

MATLAB Import/Export Routines

A.14 **ds.Virtual**

DataEditor allows the creation of virtual channels as weighted sums of real channels, and stores the definitions in file **VirtualChannels**. **readCTFds** creates **ds.Virtual** as a way of passing the definitions to **writeCTFds** for creation of an output dataset. It is a structure array with one element for each virtual channel. An example is

```
ds.Virtual(3) =  
    Name: 'VCi'  
    Unit: ''  
    chan: [4x5 char]    (MLF41, MLF42, MLP11, MLP12)  
    wt: [4x1 double]    [0.5 0.5 -0.5 -0.5]
```

It is not likely that **ds.Virtual** will be used within MATLAB.

A.15 **ds.BadChannels**

The **BadChannels** file is a text file produced by the data-acquisition program or by DataEditor that lists SQUID channels that should not be used in data analysis. **readCTFds** simply copies this file. **ds.BadChannels** is a character array with one line for each bad channel. An example is

```
ds.BadChannels =                                size(ds.BadChannels)=[2 5]  
    Q11  
    MRO21
```

This provides a simple way to pass back to DataEditor a list of bad channels. An alternative would be to remove the bad channels from **data**, **ds.res4.chanNames**, and **ds.res4.senres**.

A.16 **ds.processing**

ds.processing is a simple character image of the file **processing.cfg** which records filter and balancing commands that have been executed by DataEditor. **writeCTFds** copies **ds.processing** to file **processing.cfg** in the output dataset.

ds.processing will have no application within MATLAB. If **processing** is not a field of **ds**, **writeCTFds** will not indicate that **ds.processing** is missing.



Appendix B: Structure ds.res4

Appendix B : Structure ds.res4

Reference : "CTF MEG™ File Formats", PN900-0088.

B.1 Fields of structure ds.res4.

Here is an example of a MATLAB listing of **ds.res4** for a 275-channel CTF MEG system. The names of most fields indicate their meaning.

```
ds.res4 =
    header: 'MEG42RS '
    appName: ''
    dataOrigin: ''
    dataDescription: ''
    no_trials_avgd: 0
    data_time: '16:37'
    data_date: '06-Jun-2006'
    no_samples: 720000
    no_channels: 334
    sample_rate: 2400
    epoch_time: 300
    no_trials: 1
    preTrigPts: 0
    no_trials_done: 0
    no_trials_display: 0
    save_trials: 0
    primaryTrigger: -65536
    triggerPolarityMask: 0
    trigger_mode: 0
    accept_reject_Flag: 0
    run_time_display: 0
    zero_Head_Flag: 0
    artifact_mode: 0
    nf_run_name: '
    nf_run_title: 'CHL test'
    nf_instruments: '
    nf_collect_descriptor: 'SEF no motion'
    nf_subject_id: 'Anonymous'
    nf_operator: 'Staff'
    nf_sensorFileName: ''
    rdlen: 14
    run_description: 'SEF no motion '
    num_filters: 0
    filters: []
    chanNames: [334x32 char]
    senres: [1x334 struct]
    numcoef: 1393
    scrr: [1x1393 struct]
```

MATLAB Import/Export Routines

B.2 ds.res4.senres

ds.res4.senres is a structure array containing the channel descriptions and sensor calibration data (it contains the same information as the **.sens** file). For a typical SQUID channel (a 3rd-order balanced MEG gradiometer in this case) the structure is

```
sensorTypeIndex: 5
originalRunNum: 0
coilShape: 0
properGain: -2.8982e+009
qGain: 1048576
ioGain: 1
ioOffset: 0
numCoils: 2
grad_order_no: 3
gain: -0.3291
pos0: [3x2 double]
ori0: [3x2 double]
area: [2.5472 2.5472]
numturns: [2 2]
pos: [3x2 double]
ori: [3x2 double]
```

All SQUID channels look like this except that **sensorTypeIndex** is 0 for reference magnetometers and 1 for reference gradiometers, and **numCoils** is 1 for magnetometers. Field **area** is the loop area in cm^2 . The possible values of **sensorTypeIndex** are listed in the RES4 File Format Section of the document "CTF MEG™ File Formats", PN900-0088.

B2.1 Gain parameters in ds.res4.senres

To convert SQUID data from the 32-bit integer format of the dataset to ϕ_0 units, divide the raw data by **qGain** ($= 2^{20}$ for the current generation of SQUID electronics). The physical unit of **qGain** is ϕ_0 .

To convert data from the 32-bit integer format of the dataset to physical MKS units (T), divide by **ioGain**, **properGain** and **qGain** :

$$\text{data in physical units} = (\text{raw integer data}) / (\text{ioGain} * \text{properGain} * \text{qGain})$$

The physical unit of **properGain** is ϕ_0/T .

To convert to fT, multiply by an extra factor 10^{15} . **ds.res4.senres** field **gain** is given by

$$\text{gain} = 10^{15} / (\text{ioGain} * \text{properGain} * \text{qGain}) \quad (\text{SQUIDs})$$

$$\text{gain} = 1 / (\text{ioGain} * \text{properGain} * \text{qGain}) \quad (\text{non-SQUID channels})$$

For SQUID channels multiplying by **gain** converts 32-bit integer data to fT and the physical unit of **gain** is fT. Field **gain** is calculated by **readCTFDs** and is not part of the **res4** file.

If **getCTFdata** encounters a channel with **ioGain*properGain*qGain=0**, it sets the channel to 0. This allows **properGain=0** to be used to mark bad channels.

MATLAB Import/Export Routines

B.2.2 Sensor position and orientation information

Arrays **pos0** and **ori0** (size=[3 **numCoils**]) are the Cartesian vectors for the pickup-loop positions (cm) and orientations (unit vector) in “dewar” coordinates where the reference sensors are located at height $z=0$ and the coordinate frame is rotated 45° so \hat{x} is 45° to the subject's right. These are the values that would be read from the **.sens** file. Arrays **pos** and **ori** are the positions and orientations after conversion to the CTF Head Coordinates (or to room coordinates in the case of fMEG sensors). If a Head Localization is done as part of a dataset, then **pos** and **ori** are referred to coordinates established by the head coils: the origin ([0 0 0]) is the midpoint of the left and right coils, and the \hat{x} direction is the line from the origin to the nasion coil. If no Head Localization is done, then **pos** and **ori** refer to an origin 21 cm below the reference sensors, where \hat{z} is the dewar axis, and \hat{x} points forward (toward the subject's nose).

Fields **pos0**, **ori0**, **area**, **numturns**, **pos** and **ori** appear in the structure array **ds.res4.senres**.

	pos0, ori0	pos, ori (no head zeroing)	pos, ori (with head zeroing)
\hat{x} (coord 1)	45° right of nose	nose	Line from [0 0 0] through the nasion coil
\hat{y} (coord 2)	45° left of nose	left	$\hat{z} \times \hat{x}$
\hat{z} (coord 3)	up (dewar axis)	up (dewar axis)	\perp to the head-coil plane
Origin	Centre of the reference sensors	21 cm below references	Mid point between left and right head coils

B.2.3 **grad_order_no** (field of **ds.res4.senres**)

Field **grad_order_no** indicates what is the gradient balancing of data read from the dataset. Function **setCTFDataBalance** updates **grad_order_no** when an output data structure is specified in the output-argument list.

It is important that all of the MEG channels have the same **grad_order_no** because channels with different values of **grad_order_no** may indicate bad or disabled channels. **setCTFDataBalance** sets these channels to zero and adds them to the list of bad channels in **ds.BadChannels**.

B.3 Structure `ds.res4.scurr`

This structure array contains all of the balancing data for the sensor. The number of records in this structure is much greater than the number of SQUID channels because `ds.res4.scurr` contains 5 sets of MEG balancing coefficients (G1BR, G2BR, G3BR, G1OI, G2OI; the last two types are not true balance tables), plus G1BR coefficients for balancing the reference gradiometers. A typical element of the structure array is

```
ds.res4.scurr(101) =  
    sensorName: [1x32 uint8]    32-character sensor name  
    coefType: [1x4 uint8]      4-character balance type  
    numcoefs: 17                Number of reference sensors  
    sensor: [31x50 uint8]      List of 31-character names of reference sensors  
    coefs: [50x1 double]       The balance coefficients
```

There is room for coefficients for 50-term balancing in `ds.res4.scurr`, but only `numcoefs` terms are used (the remaining elements are zeros). `numcoefs` is 8 for order-1 and order-2 balancing, and 15-18 for order-3.

The documentation for `getCTFBalanceCoefs` (Section 3.4) shows how to use the balance coefficients.

