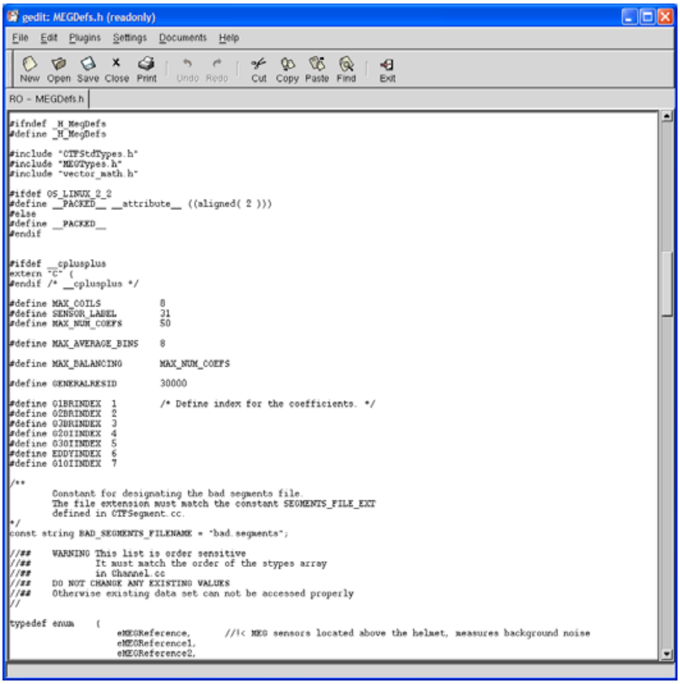
CTF MEG™ File Formats

CTF MEG™ Software



Release 5.4

CTF | MEG™

VSM MedTech Ltd.
MEDICAL ADVANCES THROUGH TECHNOLOGY™

PN900-0088
Revision 1.3, 30 November 2006
Proprietary and Confidential

CTF MEG™ File Formats

CTF MEG™ Software

Release 5.4

CTF | MEG™

VSM MedTech Ltd.
MEDICAL ADVANCES THROUGH TECHNOLOGY™

PN900-0088
Revision 1.3, 30 November 2006

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.

VSM MedTech Ltd.
MEDICAL ADVANCES THROUGH TECHNOLOGY™

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

Warnings and Cautions

CTF | MEG™

Warnings and Cautions

The following warnings and cautions appear in this guide. Please ensure you are aware of all dangers and proper procedures to prevent injury to persons or damage to equipment.

Warnings

WARNING

If you have 3rd party software reading the MEG 4.1 data format or if you have written your own code to read MEG 4.1 data format, the code must be changed to reflect the new format changes.

Cautions

None

Introduction

This guide describes a number of important files for Release 5.4 of the CTF MEG™ software.

Changes from Previous Releases

Release 5.4

Release 5.4 contains the following changes to file formats:

A method for extracting head shape points to define the cortical hull surface is now provided by MRIViewer. The older method, which extracts scalp surface points, is still available and contains the same functionality as before. Head shape files produced by either method can be used by CTF MEG software to derive a head model and analyze the data. This change has resulted in a new "SURFACE" tag that appears in the following file formats:

- "Head Shape Info File Format" on page 67.
- "Head Model File Format" on page 69
- "Dipole File Format" on page 89

Release 5.2.1

Release 5.2.1 of the CTF MEG System software introduces a new head localization procedure called Continuous Head Localization (CHL). The following additions have been made to file formats to support this feature:

New **eAngleRef**, **eExtractionRef**, and **eFitErr** sensor types and classes have been added to the RES4 file format. See "Sensor Types" on page 17 and "Sensor Classes" on page 19, respectively.

New channels (**HLC**) have been added to the CTF MEG electronics to support Continuous Head Localization. See "CTF MEG 2005 Series Channels" on page 20.

A new **USE_CONTINUOUS_HEAD_POS** tag has been added to the Fit class of the dipole file format to allow the use of CHL data in the head model when performing a dipole fit. A new **Dipole_HeadMotion** class has also been added to track the maximum head movement that occurred in the dataset for each dipole. See "Dipole File Structure (Version 6.0)" on page 89.

The new 4.1 CTF MRI file format includes additional information about the source DICOM series so that a CTF MRI file with marked points of interest (e.g., dipoles or SAM activity peak locations) can be exported to a new DICOM series with the image restored as closely as possible. See "CTF MRI File Structure (Version 4.1)" on page 79.

Other changes to this revision of the manual include the following:

A new chapter has been added to describe the ***<dataset>*.infods** file format. See "Infods 4.2 File Format" on page 37.

A new chapter has been added to describe the "**hz**" head localization datasets. See "Head Localization Datasets" on page 51.

A new appendix has been added to describe how to extract information about higher-order gradient formation from data and apply it to calculate the forward solution from a simulated dipole. See Appendix C: "Higher-order Gradients" on page 157.

Release 5.0

This release includes changes to the electronics for the CTF MEG 2005 Series. These changes are reflected in the dataset's resource file as follows:

New **eEEGBipolar**, **eEEGAflg**, **eMEGReset**, **eDipSrc**, and **eSAMSensorNorm** sensor types and classes have been added to the RES4 file. See "MEG 4.2 Resource File Format" on page 13.

Several new channels contained in the Electronic Control Console (ECC) and Head Localization Unit (HLU) have been added to the electronics. See "CTF MEG 2005 Series Channels" on page 20.

Trigger channel formats have also changed. These are described in "Trigger Channel Formats" on page 23.

Besides the addition of new sensor types and classes, the only other change in the RES4 resource file format for version 5.0 is to the **NewSensorResRec** structure in the **MegDefs.h** header file. A previously unused field now contains a stimulus polarity mask called **stimPolarity**, which is only applicable to the new stimulus channels. Each bit in **stimPolarity** corresponds to the same bit in the input word, and indicates if the bit in the data word is inverted with respect to logic true. For example, if the bit in **stimPolarity** is **1**, then the corresponding bit in the data word is active if **0** and inactive if **1**.

The **NewSensorResRec** structure for version 4.2 is listed below, with the new field displayed in **blue text**. (See “MegDefs.h Header File” on page 26 for a complete listing of this header file.)

```
typedef struct
{
 Int16                  sensorTypeIndex;
 Int16                  originalRunNum;
 CoilType               coilShape;
 SDouble                properGain;         /*may be corrected*/
 SDouble                qGain;
 SDouble                ioGain;
 SDouble                ioOffset
 Int16                  numCoils;
 Int16                  grad_order_no;
 long                   stimPolarity;
 CoilRec_extcoilTbl     [MAX_COILS];
 CoilRec_extHdcoilTbl   [MAX_COILS];
}_PACKED_NewSensorResRec;Channels
```

In version 4.1 this field was reserved, as shown below:

```
typedef struct
{
 Int16                  sensorTypeIndex;
 Int16                  originalRunNum;
 CoilType               coilShape;
 SDouble                properGain;         /*may be corrected*/
 SDouble                qGain;
 SDouble                ioGain;
 SDouble                ioOffset
 Int16                  numCoils;
 Int16                  grad_order_no;
 long                   RESERVED;
 CoilRec_extcoilTbl     [MAX_COILS];
 CoilRec_extHdcoilTbl   [MAX_COILS];
}_PACKED_NewSensorResRec;Channels
```

File Format Overview

Files may be stored in one of the following standard data formats:

text
tab-delimited text
Config Reader text
binary
CPersist Object binary

See "Standard Data Formats" on page 7 and Appendix B: "CPersist Object" on page 153 for more information about these formats.

The files described in this guide are listed in alphabetical order below:

CTF MEG File	File Version #	File Format	Software Release Compatibility[a]	Page #
BadChannels	N/A	text	4.11	53
bad.segments	N/A	text	4.17	55
ClassFile.cls	N/A	text	4.11	63
*.cov (SAM)	1	binary	5.0	135
*.dip	6.0	Config Reader text	5.4	89
*.eeg	N/A	tab-delimited text	4.14	46
*.hc	N/A	text	4.14	50
*.hdm	6.0	Config Reader text	5.4	70
hz*.ds	N/A	dataset	4.11	51
hz_<*dataset*>.txt	N/A	text	4.11	52
<*dataset*>.infods	4.2	CPersist Object binary	5.0	37

CTF MEG File	File Version #	File Format	Software Release Compatibility[a]	Page #
MarkerFile.mrk	N/A	text	4.11	59
*.meg4	4.1	binary	4.11	11
*.mri	4.1	CPersist Object binary	5.2.1	79
*.pmat	01	binary	4.17	119
*.pos	N/A	tab-delimited text	4.13	48
*.res4	4.2	binary	5.4	13
SAM time window	N/A	tab-delimited text	4.16	123
*.shape	N/A	tab-delimited text	4.11	66
*.shape_info	N/A	Config Reader text	5.4	68
*.ssv	001	binary	4.16	116
*.svl (SAM)	2	binary	5.0	126
	1	binary	4.13	131
VirtualChannels	NA	Config Reader text	4.11	42
*.wts (SAM)	2	binary	5.0	139
	1	binary	4.13	144

a. This field shows the first release using this version of the file. All later releases also use this file version.

NOTICE

All binary files, including CPersist Object files, are saved in “big endian” format.

Standard Data Formats

Text Files

CTF MEG software uses the following types of text files:

Plain

Plain text files contain information in the form of ASCII characters that can be viewed by both people and computers. (In the Linux operating system, each line is terminated with the Line Feed (LF) character not visible to the naked eye.)

Tab-delimited

Tab-delimited files are a special type of text file containing record-based data, where the columns are separated by tab characters.

Config Reader

Config Reader is a configuration file format that presents information in the form of a class followed by one or more attributes listed in tag:value pairs, as shown below:

```
Class
   {
   Tag:  Value
   Tag:  Value
   }
```

Leading white space (spaces or tabs) between the tag and value are ignored.

Binary Files

Binary files store data in the binary number system. For example, MEG sensor data is stored in ***.meg4** binary files. The CTF MEG software uses the CPersist object class for binary files describing parameters that may contain nested objects. See Appendix B: “CPersist Object” on page 153 for details.

Currency

Release 5.4 of the CTF MEG software is compatible with the following versions of the CTF MEG electronics:

CTF MEG 2005 Series

CTF MEG 2000 Series

See "Determining the Electronics Version" below for more information.

This document is: **PN900-0088** revision **1.3** (30-Nov-06). Revisions with a letter in the tag are drafts and are subject to change before final release.

This revision is current with the CTF MEG 2005 Series:

CTF MEG System Software: **Release 5.4**

CTF MEG Electronics 2005

Determining the Software Version

GUI applications: select **Help > About** to open the Release Information dialog.

Command-line applications: from the command line, enter the command with the parameter -version to report the release number.

Determining the Electronics Version

The **CTF MEG 2005 Series** (mid-2004 onwards) contains the CTF MEG Electronics 2005:

- DSQ-2010 channel units
- DSQ-2041 interface card
- Electronics Control Console 2005
- Real-time Processing Cluster 2005

The **CTF MEG 2000 Series** (2002 to mid-2004) contains the CTF MEG Electronics 2000:

- DSQ-2010 channel units
- DSQ-2041 interface card
- Peripheral Interface Unit (PIU) 2000

This series was formerly known as the OMEGA 2000.

The **DSQ 2000 Hybrid Series** (1997 to 2001) contains the DSQ 800 electronics:

- DSQ-810 channel units
- DSQ-20xx processing units
- Peripheral Interface Unit (PIU) 853 & 851

The **DSQ 800 Series** (1993 to 1996) contains the DSQ 800 electronics:

- DSQ-810 channel units
- DSQ-84x processing units
- Peripheral Interface Unit (PIU) 800

DSQ model numbers are visible on the front panel of the electronics rack.

Intended Audience

This reference is useful to programmers who wish to access files, or create datasets through their own methods.

References

This document assumes you are familiar with the UNIX file system, the C programming language, and basic machine architecture.

Document Structure

Documents are generally provided in both hardcopy and Adobe® Acrobat® PDF (Portable Document Format). For additional copies of the printed manual, contact your account manager. The PDF editions are distributed on CD with the related software, and include bookmarks and hyperlinks to assist navigating the document.

You may print copies of the PDF editions for internal use but all copies must be treated as proprietary and confidential; they are *not* to be distributed. The PDF documents are formatted to the same size as the printed editions. Use the Acrobat option to **Expand small pages to paper size** to scale up to standard printer paper.

Typical signal strengths in MEG measurements are in the order of pT (picoteslas = 10^{-12}) or fT (femtoteslas = 10^{-15}).

Sending Your Comments

We'd like to hear from you. Your comments and suggestions for improving this document are welcome and appreciated. Please e-mail your feedback to **support@vsmmedtech.com** citing document number **PN900-0088** revision **1.3**.

Thank you.

MEG4 File Format

In the CTF MEG software, a physical dataset is represented as a directory with the same name as the dataset and a **.ds** extension. Within this directory, two critical files exist — a data file called **<*dataset_name*>.meg4** and a resource file called **<*dataset_name*>.res4**. The data file contains the collected samples of data, and the resource file contains information that is essential to interpreting the data. This chapter describes the format of the dataset and resource files.

WARNING

If you have 3rd party software reading the MEG 4.1 data format or if you have written your own code to read MEG 4.1 data format, the code must be changed to reflect the new format changes.

MEG 4.1 Data File Format

The MEG 4.1 data file (***.meg4**) is a binary file consisting of a header and the raw samples collected from the CTF MEG electronics. This format is shown in Table 1 below.

Table 1: Data File Format

`8-byte header:` **`MEG41CP\0`**
`Trial 0, Channel 0`
`Trial 0, Channel 1`

Table 1: Data File Format

•
•
•
`Trial 0, Channel m-1`
`Trial 1, Channel 0`
`Trial 1, Channel 1`
•
•
•
`Trial n-1, Channel m-2`
`Trial n-2, Channel m-1`

The header is an eight-byte character sequence **MEG41CP+NULL** (i.e., the eight-byte string is terminated with a null, or zero). The data is stored as a sequence of signed four-byte integers, starting with the first trial and first channel, then the first trial and second channel, etc. The number of channels per trial and the number of samples in every trial-channel block are constant per dataset. Constants are found in the general resources stored in the resource file (see "MEG 4.2 Resource File Format" on page 13). The numbers stored in the data file are the raw numbers collected from the electronics. For these numbers to be useful, the various gains must be applied. Gains are stored in the sensor resources, also in the resource file.

RES4 File Format

MEG 4.2 Resource File Format

WARNING

If you have 3rd party software reading the MEG 4.1 data format or if you have written your own code to read MEG 4.1 data format, the code must be changed to reflect the new format changes.

The MEG 4.2 resource file (***.res4**) is a binary file consisting of a header plus the fields and records listed in Table 2 below.

Table 2: : Resource File Format

Description	Byte Offset	Size (bytes)	How Many	Type
header: MEG42RS+NULL	0	1	8	Char
general resources	8	1832	1	meg41GeneralResRec
unused	1840	4	1	Bit32
run description	1844	1	rdl	Char
number of filters	1844+rdl	2	1	Int16

Table 2: : Resource File Format

filter information filter frequency filter class filter type number of parameters filter parameters	1846+rdl b+fi*18+p b+fi*18+p+8 b+fi*18+p+12 b+fi*18+p+16 b+fi*18+p+18	variable 8 4 4 2 8 * number of parameters	number of filters	 SDouble classType filtType Int16 SDouble sequence
channel names	b+f	32	number of channels	Str32
sensor resources	b+f+nc*32	1328	number of channels	NewSensorResRec
number of coefficient records	b+f+nc*1360	2	1	Int16
sensor coefficient records	b+f+nc*1360+2	1992	number of coefficient records	SensorCoefResRec

The header is an eight-byte character sequence **MEG42RS+NULL** (i.e., the eight-byte string is terminated with a null, or zero). The byte offset is defined by the following variables:

rdl = length of run description (**meg41GeneralResRec::rdlen**)

b = 1846 + rdl

fi = current filter index (starting at 0)

p = the cumulative number of parameters in all previous filters * 8 (i.e. the sum of **filter::numParam** of each filter previous to the current one multiplied by the size of each parameter)

nf = number of filters

np = number of filter parameters (p is a running sum, whereas np is the final value of p)

f = nf*18 + np*8

nc = number of channels (**meg41GeneralResRec::gSetUp::no_channels**)

All definitions used in the resource file are contained in the **MegDefs.h** header file. A complete listing of this file can be found in “MegDefs.h Header File” on page 26.

Definitions

The following are definitions for some of the terms used in this chapter.

ADC Analog-to-Digital Converter. The CTF MEG 2005 System contains 16 ADC channels to measure signals in the range of +/-10V at a rate of 192kHz with 16-bit resolution. All sixteen channels can be accessed from the back panel connector. The first four may also be accessed through the front panel BNC connectors. Each ADC channel can also be configured to generate digital output signals using rising and falling thresholds. To simplify the addition of manual trigger buttons or photodiodes, the first four channels may be configured to have a pullup resistor.

CHL Continuous Head Localization. This is a feature of the CTF MEG 2005 System. CHL provides the ability to record and monitor the patient's head position and head movement in real time during a data recording, thereby allowing the operator to take whatever action is appropriate when excessive head motion occurs.

DAC Digital-to-Analog Converter. The CTF MEG 2005 System contains four DAC channels as a general-purpose signal generator. They output up to +/-10V signals at +/-10mA with 16-bit resolution. The output is available from individual BNC connectors on the front panel or from a single DB9 connector on the back channel. Triggers can be generated from the DAC channels by setting rising and falling thresholds.

Digital interfaces All evoked field response MEG exams require a stimulus to be presented to the subject. Simultaneously with the presentation of these stimuli, the stimulus computer also transmits trigger information to the Electronic Control Console (ECC). Three digital ports are provided for interfacing to the stimulus computer — the general-purpose I/O port, the parallel ports, and the serial port. This trigger information is recorded along with the MEG and EEG data, and is used in post-processing to categorize events or to select a point around which to average.

ECC Electronic Control Console. A desktop box in the CTF MEG 2005 System containing general purpose ADC, DAC, and digital I/O channels. The purpose of the ECC is to provide a convenient interface to a wide variety of peripheral equipment.

Fit error The fit error is an indicator of the reliability (i.e., goodness of fit) of the head localization measurement. Specifically, it is the total weighted least-squares fit error over the integration time (or update interval) and over the sensor channels (HLC) used for head coil localization. Each head coil is localized independently; thus there is one fit error per coil per measurement (i.e., per update interval).

HLC Head Localization Channel. A channel type for the CTF MEG 2005 System used to carry continuous head localization information (x, y, and z coordinates of the head position relative to the dewar, fit error, etc.)

HLU Head Localization Unit. A subsystem in the CTF MEG 2005 System used to locate the head within the volume of the dewar helmet.

Sensor Types

Sensor types for the MEG42RS datasets are listed in Table 3.

Table 3: Sensor Type Codes

Sensor Type	Code	Description
eMEGReference	0	Reference magnetometer channel
eMEGReference1	1	Reference 1st-order gradiometer channel
eMEGReference2	2	Reference 2nd-order gradiometer channel
eMEGReference3	3	Reference 3rd-order gradiometer channel
eMEGSensor	4	Sensor magnetometer channel located in head shell
eMEGSensor1	5	Sensor 1st-order gradiometer channel located in head shell
eMEGSensor2	6	Sensor 2nd-order gradiometer channel located in head shell
eMEGSensor3	7	Sensor 3rd-order gradiometer channel located in head shell
eEEGRef	8	EEG unipolar sensors *not* on the scalp
eEEGSensor	9	EEG unipolar sensors on the scalp
eADCRef	10	(see eADCAmpRef below)
eADCAmpRef	10	ADC amp channels from HLU or PIU (old electronics)
eStimRef	11	Stimulus channel for MEG41
eTimeRef	12	Time reference coming from video channel
ePositionRef	13	Measured position of head and head coils
eDACRef	14	DAC channel from ECC or HLU
eSAMSensor	15	SAM channel derived through data analysis
eVirtualSensor	16	Virtual channel derived by combining two or more physical channels
eSystemTimeRef	17	System time showing elapsed time since trial started
eADCVoltRef	18	ADC volt channels from ECC

Table 3: Sensor Type Codes

Sensor Type	Code	Description
eStimAnalog	19	Analog trigger channels
eStimDigital	20	Digital trigger channels
eEEGBipolar	21	EEG bipolar sensor not on the scalp
eEEGAflg	22	EEG ADC over range flags
eMEGReset	23	MEG resets (counts sensor jumps for crosstalk purposes)
eDipSrc	24	Dipole source
eSAMSensorNorm	25	Normalized SAM channel derived through data analysis
eAngleRef	26	Orientation of head localization field
eExtractionRef	27	Extracted signal from each sensor of field generated by each localization coil
eFitErr	28	Fit error from each head localization coil
eOtherRef	29	Any other type of sensor not mentioned but still valid
eInvalidType	30	An invalid sensor

Sensor Classes

The sensor types in the MEG42RS datasets are grouped into the sensor classes listed in Table 4.

Table 4: Sensor Classes

Sensor Class	Description
MEGRef	All MEG sensors located above the helmet
MEGSensor	All MEG sensors located in the helmet
EEGRef	All EEG sensors not on the scalp
EEGSensor	All EEG sensors on the scalp
ADCRef	All ADC channels measured in amps (current)
StimRef	All stim channels
TimeRef	All time channels
PositionRef	Measured position of head and head coils
DACRef	All DAC channels
SAMSensor	All derived SAM channels
VirtualSensor	All derived virtual channels
badMEGSensor	Not used
badEEGSensor	Not used
ADCVoltRef	All ADC channels measured in volts
SuppRef	Supplementary channels (MEGReset and EEG_AFLG)
DipoleSource	All dipole source channels
AngleRef	One of the angle components of the orientation vector of a head localization coil
ExtractionRef	Measured data at a sensor from one of the head localization coils
FitErr	Fit error of the head localization for a head localization coil
OtherRef	Other valid channel classes
InvalidClass	Invalid channel classes

CTF MEG 2005 Series Channels

The CTF MEG 2005 Series electronics contain several new channels in the Electronic Control Console (ECC) and the Head Localization Unit (HLU). The channels consist of 32-bit stimulus channels, 32-bit analog input channels (ADCs), and 32-bit analog output channels (DACs).

Table 5: CTF MEG 2005 Channels

Channel Name	Unit	Connector	Description
HADC001	HLU	Nasion	Analog input channel
HADC002	HLU	Left ear	Analog input channel
HADC003	HLU	Right ear	Analog input channel
HADC004	HLU	Inion	Analog input channel
HADC005	HLU	Vertex	Analog input channel
HADC006	HLU	Phantom	Analog input channel
HADC007	HLU	(not connected)	Analog input channel
HADC008	HLU	(not connected)	Analog input channel
HDAC001	HLU	Nasion	Analog output channel
HDAC002	HLU	Left ear	Analog output channel
HDAC003	HLU	Right ear	Analog output channel
HDAC004	HLU	Inion	Analog output channel
HDAC005	HLU	Vertex	Analog output channel
HDAC006	HLU	Phantom	Analog output channel
HDAC007	HLU	(not connected)	Analog output channel)
HDAC008	HLU	(not connected)	Analog output channel
HTRIG001	HLU	Connected to analog outputs	Analog trigger channels for HDAC001 – HDAC004
HTRIG002	HLU	Connected to analog outputs	Analog trigger channels for HDAC005 – HDAC008

Table 5: CTF MEG 2005 Channels

Channel Name	Unit	Connector	Description
UADC001	ECC	ADC 1-16 (front panel)	Analog input channel
UADC002	ECC	ADC 1-16 (front panel)	Analog input channel
UADC003	ECC	ADC 1-16 (front panel)	Analog input channel
UADC004	ECC	ADC 1-16 (front panel)	Analog input channel
UADC005	ECC	ADC 1-16	Analog input channel
UADC006	ECC	ADC 1-16	Analog input channel
UADC007	ECC	ADC 1-16	Analog input channel
UADC008	ECC	ADC 1-16	Analog input channel
UADC009	ECC	ADC 1-16	Analog input channel
UADC010	ECC	ADC 1-16	Analog input channel
UADC011	ECC	ADC 1-16	Analog input channel
UADC012	ECC	ADC 1-16	Analog input channel
UADC013	ECC	ADC 1-16	Analog input channel
UADC014	ECC	ADC 1-16	Analog input channel
UADC015	ECC	ADC 1-16	Analog input channel
UADC016	ECC	ADC 1-16	Analog input channel
UDAC001	ECC	DAC 1-4 (front panel)	Analog output channel
UDAC002	ECC	DAC 1-4 (front panel)	Analog output channel
UDAC003	ECC	DAC 1-4 (front panel)	Analog output channel
UDAC004	ECC	DAC 1-4 (front panel)	Analog output channel

Table 5: CTF MEG 2005 Channels

Channel Name	Unit	Connector	Description
UPPT001	ECC	Parallel port A	Parallel input trigger channel
UPPT002	ECC	Parallel port B	Parallel input trigger channel
USPT001	ECC	Master	Serial input trigger channel
USPT002	ECC	Slave	Serial input trigger channel
UDIO001	ECC	General purpose I/O	General purpose digital input/output channel
UTRG001	ECC	Connected to analog inputs/outputs	Analog input/output trigger channel
HLC00*n*1	HLU	derived	X coordinate relative to the dewar (in meters) of the n^{th} head localization coil
HLC00*n*2	HLU	derived	Y coordinate relative to the dewar (in meters) of the n^{th} head localization coil
HLC00*n*3	HLU	derived	Z coordinate relative to the dewar (in meters) of the n^{th} head localization coil
HLC00*n*4	HLU	derived	Reserved
HLC00*n*5	HLU	derived	Reserved
HLC00*n*6	HLU	derived	Reserved
HLC00*n*7	HLU	derived	Synchronization channel (synchronous with SCLK channel)
HLC00*n*8	HLU	derived	Coil localization fit error

Trigger Channel Formats

Each of the trigger input channels are 32 single-bit inputs that are tied either to inputs on the digital channels or to threshold detectors on the analog channels. Their formats are illustrated below. For more information on trigger channel formats, refer to the *CTF MEG Electronics Manual* (PN900-0028).

HTRG001 and HTRG002

The HLU analog output trigger channels consist of two 32-bit binary values. Every second bit in the lower eight bits of the trigger word correspond to an HLU DAC output channel, as shown in Figure 1 and Figure 2 below. Other bits are not used.

bit positions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									4		3		2		1
																									HDAC						

channel name

Figure 1: HLU Trigger Channel HTRIG001

bit positions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									8		7		6		5
																									HDAC						

channel name

Figure 2: HLU Trigger Channel HTRIG002

UTRG001

The ECC analog input/output trigger channel consists of a 32-bit binary value. The least significant four bits correspond to the four DAC output channels, and the most-significant bits correspond to the 16 ADC input channels. The mapping of the bits in the analog trigger channel to the corresponding analog channels is shown in Figure 3. Other bits are not used.

bit positions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1													4	3	2	1
UADC																												UDAC			

channel name

Figure 3: Analog I/O Trigger Channel

UPPT001 and UPPT002

The UPI contains two 32-bit parallel digital trigger channels that are compatible with a standard parallel printer port. These ports can be monitored during collection. Their format is shown in Figure 4.

bit positions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								UPPT001 / UPT002							

channel name

Figure 4: Parallel Port Trigger Channel

UDIO001

The general purpose digital input/output channel is a 32-bit data channel, containing two 8-bit ports. Each port can be configured independently for input or output. Their format is shown in Figure 5.

bit positions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																UDIO001															

channel name

Figure 5: General Purpose I/O Trigger Channel

USPT001 and USPT002

The ECC contains two serial digital I/O ports that can either be monitored during collection or used to control peripheral devices. Their format is shown in Figure 6 below.

bit positions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								USPT001 / USPT002							

channel name

Figure 6: Serial Input Trigger Channel

MegDefs.h Header File

The following file has been reformatted for this document.

```
/*===================================================================
* $Header: MEGDefs.h,v 24.0 2005/01/14 23:42:41 $
* ===================================================================
*
* MEGDefs.h
*
* Definitions used by the dataset and its clients.
*
* Copyright (c) CTF Systems Inc., 1995-1996. All Rights Reserved.
* Commercially Confidential Information
*
* ===================================================================
*
*/
#ifndef _H_MegDefs
#define _H_MegDefs
#include "CTFStdTypes.h"
#include "MEGTypes.h"
#include "vector_math.h"
#ifdef OS_LINUX_2_2
#define __PACKED__ __attribute__ ((aligned( 2 )))
#else
#define __PACKED__
#endif

#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */
#define MAX_COILS 8
#define SENSOR_LABEL 31
#define MAX_NUM_COEFS 50
#define MAX_AVERAGE_BINS 8
#define MAX_BALANCING MAX_NUM_COEFS
#define GENERALRESID 30000
#define G1BRINDEX 1 /* Define index for the coefficients. */
#define G2BRINDEX 2
#define G3BRINDEX 3
#define G2OIINDEX 4
#define G3OIINDEX 5
#define EDDYINDEX 6
#define G1OIINDEX 7
```

```
/**
Constant for designating the bad segments file.
The file extension must match the constant SEGMENTS_FILE_EXT
defined in CTFSegment.cc.
*/
const string BAD_SEGMENTS_FILENAME = "bad.segments";
//## WARNING This list is order sensitive
//## It must match the order of the stypes array
//## in Channel.cc
//## DO NOT CHANGE ANY EXISTING VALUES
//## Otherwise existing data set can not be accessed properly
//
typedef enum {
    eMEGReference, //!< MEG sensors located above the helmet, measures background noise
    eMEGReference1,
    eMEGReference2,
    eMEGReference3,
    eMEGSensor, //!< MEG sensors located in the helmet, measures head and background noise
    eMEGSensor1,
    eMEGSensor2,
    eMEGSensor3,
    eEEGRef, //!< EEG unipolar sensors that are not on the scalp
    eEEGSensor, //!< EEG unipolar sensors that are on the scalp
    eADCRef,
    eADCAmpRef = eADCRef, //!< ADC amp channels from UPI/PIU
    eStimRef, //!< Stimulus channel for MEG41
    eTimeRef, //!< Time ref coming from video channel
    ePositionRef, //!< Measured position of head and head coils
    eDACRef, //!< DAC channel from UPI or HLU
    eSAMSensor, //!< SAM channel derived through data analysis
    eVirtualSensor, //!< Virtual channel derived by combining 2 or more physical channels
    eSystemTimeRef, //!< System time showing elapsed time since trial started
    eADCVoltRef, //!< ADC volt channels from UPI
    eStimAnalog, //!< Analog trigger channels
    eStimDigital, //!< Digital trigger channels
    eEEGBipolar, //!< EEG bipolar sensor, that will not be on the scalp
    eEEGAflg, //!< EEG ADC Over range flags
    eMEGReset, //!< MEG resets - counts sensor jumps/resets for cross talk purposes
    eDipSrc, //!< Dipole source
    eSAMSensorNorm, //!< Normalized SAM channel derived through data analysis
    eAngleRef, //!< Orientation of head localization field
    eExtractionRef, //!< Extracted signal from each sensor of field generated by each localization coil
    eFitErr, //!< Fit error from each head localization head coil
    eOtherRef, //!< Any other type of sensor not mentioned, but is still valid
    eInvalidType //!< An invalid sensor
} SensorType;
```

```
typedef enum {
    MEGRef, //!< All MEG sensors located above the helmet
    MEGSensor, //!< All MEG sensors located in the helmet
    EEGRef, //!< All EEG sensors that are not on the scalp
    EEGSensor, //!< All EEG sensors that are on the scalp
    ADCRef, //!< All ADC channels measured in amps
    StimRef, //!< All stim channels
    TimeRef, //!< All time channels
    PositionRef, //!< Measured position of head and head coils
    DACRef, //!< All DAC channels
    SAMSensor, //!< All derived SAM channels
    VirtualSensor, //!< All derived virtual channels
    badMEGSensor,
    badEEGSensor,
    ADCVoltRef, //!< All ADC channels measured in volts
    SuppRef, //!< Supplementary channels, MEGReset and EEG_AFLG
    DipoleSource, //!< All dipole source channels
    AngleRef, //!< Orientation vector of a head localization coil
    ExtractionRef, //!< Measured data at a sensor of the head localization coils
    FitErr, //!< Fit error from head localization coils
    OtherRef,
    InvalidClass
} eChType;

typedef eChType SensorClass;
typedef enum { CIRCULAR,
    SQUARE,
    VOLTMETER = CIRCULAR,
    AMMETER = SQUARE
} coiltype;
typedef coiltype CoilType;

typedef union d3_point
{
    struct { DDouble x,y,z, junk;} c;
    struct { DDouble r,theta,phi, junk ;} s;
    DDouble point[4];
} d3_point;

typedef union d2_point
{
    struct { DDouble x,y; } c;
    struct { DDouble r,theta; } p;
    DDouble point[2];
} d2_point;
typedef union d3_point_ext /* Externally store points as SDouble */
{
    struct { SDouble x,y,z, junk ;} c;
```

```
    struct { SDouble r,theta,phi, junk ;} s;
    SDouble point[4];
} d3_point_ext;




typedef struct CoilRec_ext
{
    d3_point_ext position; /* position of coil */
    d3_point_ext orient; /* orientation of coil */
    Int16 numturns; /* number of turns making up the coil */
    short reserved1; /* pad out to the next 8 byte boundary */
    short reserved2;
    short reserved3;
    SDouble area; /* area of coil */
} CoilRec_ext __PACKED__ ;

typedef struct CoilRec
{
    d3_point position; /* position of coil */
    d3_point orient; /* orientation of coil */
    Int16 numturns; /* number of turns making up the coil */
    DDouble area; /* area of coil */
} CoilRec __PACKED__ ;

typedef struct coef_List
{
    Int16 index;
    CChar name[SENSOR_LABEL];
} Coef_List __PACKED__ ;
typedef struct
{
    Int16 sensorTypeIndex;
    Int16 originalRunNum;
    CoilType coilShape;
    SDouble properGain; /* may be corrected */
    SDouble qGain;
    SDouble ioGain;
    SDouble ioOffset;
    Int16 numCoils;
    Int16 grad_order_no;
    long stimPolarity;
    CoilRec_ext coilTbl[MAX_COILS];
    CoilRec_ext HdcoilTbl[MAX_COILS];
} __PACKED__ NewSensorResRec;
```

```
typedef struct CoefResRec /* Making generic resource for coefficients. */
{
    Int16 num_of_coefs;
    CChar sensor_list[MAX_BALANCING][SENSOR_LABEL];
    SDouble coefs_list[MAX_BALANCING];
} __PACKED__ CoefResRec, *CoefResRecP, **CoefResRecH;

typedef struct
{
    CChar nf_run_name[32],
    nf_run_title[256],
    nf_instruments[32],
    nf_collect_descriptor[32],
    nf_subject_id[32],
    nf_operator[32],
    nf_sensorFileName[56];
    Int32 size; /* length of following array */
    long reserved1; /* pad out to the next 8 byte boundary */
    CStrPtr nf_run_descriptor;
} __PACKED__ meg4FileSetup ;
typedef enum { CLASSERROR, BUTTERWORTH } classType;
typedef enum { TYPERROR, LOWPASS, HIGHPASS, NOTCH } filtType;
typedef struct
{
    SDouble freq;
    classType fClass;
    filtType fType;
    Int16 numParam;
    SDoubleArr params;
}__PACKED__ filter;
/*
* This enum is used by GeneralRsrc to keep track of which
* part of the trigger format union it will be reading
*/
enum TriggerStructFormat { MEG40_TRIG_FMT, MEG41_TRIG_FMT, MEG42_TRIG_FMT };
/*
* Trigger structure for the meg4 dataset
*/
typedef struct
{
    UCChar primaryTrigger;
```

```
    UCChar secondaryTrigger[MAX_AVERAGE_BINS];
    UCChar triggerPolarityMask;
}__PACKED__ meg40TriggerData;


/*
* Trigger structure for the meg5 dataset
*/
typedef struct
{
    Bit32 primaryTrigger;
    Bit32 triggerPolarityMask;
}__PACKED__ meg41TriggerData;

typedef struct
{
    Int32 no_samples;
    Int16 no_channels;
    short reserved1; /* pad out to the next 8 byte boundary */
    SDouble sample_rate;
    SDouble epoch_time;
    Int16 no_trials;
    short reserved2; /* pad out to the next 8 byte boundary */
    Int32 preTrigPts;
    Int16 no_trials_done;
    Int16 no_trials_display;
    CTFBoolean save_trials;
union
{
    meg40TriggerData meg40trig;
    meg41TriggerData meg41trig;
};
    short reserved3; /* pad out to the next 8 byte boundary */
    Int16 trigger_mode;
    short reserved4; /* pad out to the next 8 byte boundary */
    CTFBoolean accept_reject_Flag;
    Int16 run_time_display;
    short reserved5; /* pad out to the next 8 byte boundary */
    CTFBoolean zero_Head_Flag;
    CTFBoolean artifact_mode;
}__PACKED__ new_general_setup_rec_ext;

struct meg41GeneralResRec
{
    CStr256 appName;
    CStr256 dataOrigin;
    CStr256 dataDescription;
    Int16 no_trials_avgd;
```

```
    CChar data_time[255];
    CChar data_date[255];
    new_general_setup_rec_ext gSetUp;
    meg4FileSetup nfSetUp;
} __PACKED__; // so we don't have alignment problems
// padding is by explicit declarations
typedef struct meg41GeneralResRec meg41GeneralResRec;
typedef struct
{
    CStr32 sensorName;
    Bit32 coefType;
    long reserved1; /* pad out to the next 8 byte boundary */
    CoefResRec coefRec;
} SensorCoefResRec;
/// fiducials struct
    struct FidPoints_t
{
    Point_3D nasion;
    Point_3D leftEar;
    Point_3D rightEar;
};
/// List of MEG system types
static const string CTFMEGSystemType[] = { "cmeg",
"fmeg",
"Untitled" };
/**
Enumerator for MEG system type

This values can not be change as they are stored in the dataset info file. enum can be only extended.
*/
enum CTFMEGSystemType_t
{
    CTFMEGSystemType_CMEG = 0,
    CTFMEGSystemType_FMEG,
    CTFMEGSystemType_UNKNOWN
};

#ifdef __cplusplus
}
#endif /* __cplusplus */
#endif /* _H_MegDefs */
```

Sample Program

The following code demonstrates how to use the declarations and definitions in the **MegDefs.h** header file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "MegDefs.h"

typedef struct
{
    meg41GeneralResRec          genres;
    Char*                       run_description;
    Int16                       num_filters;
    filter*                     filters;
    Str32*                      chanNames;
    NewSensorResRec*            senres;
    Int16                       numcoef;
    SensorCoefResRec*           scrr;
} resource;

void makeResources(FILE* fp, resource* rez)
{
    Bit32 padding;
    Int16 numChannels;
    Int16 i;

    fread(&rez->genres,1832,1,fp);
    fread(&padding,4,1,fp);
    rez->run_description = (Char*)calloc(rez->genres.rdlen,1);
    fread(rez->run_description,1,rez->genres.rdlen,fp);
    fread(&rez->num_filters,2,1,fp);
    rez->filters = (filter*)calloc(rez->num_filters,24);
    for(i = 0; i < rez->num_filters; i++)
    {
                Int16 numParam;
                fread(&rez->filters[i].freq,8,1,fp);
                fread(&rez->filters[i].fClass,4,1,fp);
                fread(&rez->filters[i].fType,4,1,fp);
                fread(&rez->filters[i].numParam,2,1,fp);
                numParam = rez->filters[i].numParam;
                rez->filters[i].params=(SDouble*)calloc(numParam,8);
                fread(rez->filters[i].params,8,numParam,fp);
    }
```

```
    numChannels = rez->genres.gSetUp.no_channels;
    rez->chanNames = (Str32*)calloc(numChannels,32);
    fread(rez->chanNames,32,numChannels,fp);
    rez->senres = (NewSensorResRec*)calloc(numChannels,1328);
    fread(rez->senres,1328,numChannels,fp);
    fread(&rez->numcoef,2,1,fp);
    rez->scrr = (SensorCoefResRec*)calloc(rez->numcoef,1992);
    fread(rez->scrr,1992,rez->numcoef,fp);
}

void deleteResources(resource* rez)
{
    Int16 i;
    free(rez->run_description);
    for(i = 0; i < rez->num_filters; i++)
                    free(rez->filters[i].params);
    free(rez->filters);
    free(rez->chanNames);
    free(rez->senres);
    free(rez->scrr);
}

main(int argc, char* argv[])
{
    Char header[8];
    FILE* resFile;
    resource rez;

    if(argc != 2)
    {
                    fprintf(stderr,"usage:dstest MEG/41-resource file\n");
                    exit(1);
    }
    if(!(resFile = fopen(argv[1],"rb")))
    {
                    fprintf(stderr,"dsdump: cannot open %s\n",argv[1]);
                    exit(1);
    }
    fread(header,1,8,resFile);
    if(memcmp(header,"MEG41RE",8))
    {
                    fprintf(stderr,"dsdump: %s is the wrong format\n",argv[1]);
                    exit(1);
    }
```

```
    makeResources(resFile,&rez);
    /* do stuff with the resources */
    deleteResources(&rez);

    fclose(resFile);

    exit(0);
}
```

Infods Format

Infods 4.2 File Format

The **<*dataset_name*>.infods** file is a CPersist file that is stored in the dataset directory. It contains information about the procedure, dataset, head coil positions, and head localization associated with the dataset.

Table 6: Infods File Tags

Tag (Attribute)	Type	Description
_PATIENT_INFO	CPersist	Patient information group header.
_PATIENT_UID	CString	Automatically generated DICOM unique patient identifier (UID).
_PATIENT_NAME_FIRST	CPersist	The first name of the patient .
_PATIENT_NAME_MIDDLE	CString	The middle name of the patient.
_PATIENT_NAME_LAST	CString	The last name of the patient.
_PATIENT_ID	CString	The patient’s identification reference.
_PATIENT_BIRTHDATE	CString	The patient’s birthdate (yyyymmdd).
_PATIENT_SEX	integer	The patient’s gender: 0=Male 1=Female 2=Other
_PATIENT_PACS_NAME	CString	The patient’s name in the hospital’s PACS system.
_PATIENT_PACS_UID	CString	The patient’s identifier in the hospital’s PACS system

Table 6: Infods File Tags

Tag (Attribute)	Type	Description
_PATIENT_INSTITUTE	CString	The institute associated with the patient.
_PROCEDURE_INFO	CPersist	Procedure information group header.
_PROCEDURE_VERSION	integer	Procedure version (currently 1).
_PROCEDURE_UID	CString	Unique performed procedure identification number.
_PROCEDURE_ACCESSIONNUMBER	CString	The assigned procedure accession number .
_PROCEDURE_TITLE	CString	The procedure title (currently a copy of the procedure accession number).
_PROCEDURE_SITE	CString	Site identifier where the procedure was performed.
_PROCEDURE_STATUS	integer	Procedure status: • 1=Open • 2=Closed
_PROCEDURE_TYPE	integer	Procedure type: • 1=Clinical • 2=Research • 3=Unknown
_PROCEDURE_STARTEDDATETIME	CString	Date and time the performed procedure was started.
_PROCEDURE_CLOSEDDATETIME	CString	Date and time the performed procedure was closed.
_PROCEDURE_COMMENTS	CString	User-entered procedure comments.
_PROCEDURE_LOCATION	CString	Location of the procedures files in the file system.
_PROCEDURE_ISINDB	integer	Indicates whether the dataset is in the database: • 0=No • 1=Yes
_DATASET_INFO	CPersist	Dataset information group header.
_DATASET_VERSION	integer	Dataset version number.

Table 6: Infods File Tags

Tag (Attribute)	Type	Description
_DATASET_UID	CString	Automatically generated DICOM unique dataset identifier.
_DATASET_PATIENTUID	CString	Automatically generated DICOM unique patient identifier (should be the same as _PATIENT_UID).
_DATASET_PROCEDUREUID	CString	Unique performed procedure identifier (should be the same as _PROCEDURE_UID).
_DATASET_STATUS	CString	Dataset status: • Not reviewed • Accepted • Rejected • Unknown
_DATASET_RPFILE	CString	The name of the recording protocol used to record the data.
_DATASET_PROCSTEPTITLE	CString	The procedure title.
_DATASET_PROCSTEPPROTOCOL	CString	The name of the defined procedure (study).
_DATASET_PROCSTEPDESCRIPTION	CString	Defined procedure description.
_DATASET_COLLECTIONDATETIME	CString	Collection date and time of original data. (This value is carried through to all processed data.)
_DATASET_COLLECTIONSOFTWARE	CString	The name and version of the recording program. (This value is carried through to all processed data.)
_DATASET_CREATORDATETIME	CString	Creation date and time.
_DATASET_CREATORSOFTWARE	CString	Creation application and version.
_DATASET_KEYWORDS	CString	Keywords entered by user.
_DATASET_COMMENTS	CString	Comments entered by user.
_DATASET_OPERATORNAME	CString	Operator name entered during the recording.
_DATASET_LASTMODIFIEDDATETIME	CString	Last modified date and time.

Table 6: Infods File Tags

Tag (Attribute)	Type	Description
_DATASET_NOMINALHCPOSITIONS	integer	Head position: • 0=Real (measured) • 1=Nominal
_DATASET_COEFSFILENAME	CString	Name of balancing coefficients file.
_DATASET_SENSORSFILENAME	CString	Name of sensor description file.
_DATASET_SYSTEM	CString	System type name.
_DATASET_SYSTEMTYPE	CString	System type: • cMEG • fMEG
_DATASET_LOWERBANDWIDTH	double	Lower limit of bandwidth of data (Hz).
_DATASET_UPPERBANDWIDTH	double	Upper limit of bandwidth of data (Hz).
_DATASET_ISINDB	integer	Indicates whether the dataset is in the database: • 0=No • 1=Yes
_DATASET_HZ_MODE	integer	Head localization mode: • 0=None • 1=Before only • 2=After only • 3=Before and after • 4=Between trials • 5=Continuous • -1=Unknown
_DATASET_MOTIONTOLERANCE	double	Excessive head motion threshold.
_DATASET_MAXHEADMOTION	double	Maximum head motion during measurement.
_DATASET_MAXHEADMOTIONTRIAL	unsigned short	Trial containing the maximum head motion.
_DATASET_MAXHEADMOTIONCOIL	CString	Head localization coil with maximum head motion.

Virtual Channels File Format

Virtual channels are weighted linear combinations of real channels collected by the CTF MEG System. They can be created by a text editor, or in Acq or DataEditor using the Virtual Channel Editor dialog then saved to a file called **VirtualChannels** in the dataset directory. **VirtualChannels** files use the Config Reader text format (see “Text Files” on page 7 for details).

For each virtual channel, the file contains the following information:

Variable	Description
Name	The name assigned to the virtual channel.
Unit	Not used. The virtual channel unit will always be the unit of the first channel referenced by the virtual channel definition.
Ref	Consists of reference channel names and corresponding weights. The reference channels multiplied by their weights are summed to create a virtual channel. The number of channels that can be referenced is limited by the number of channels in the dataset.

Sample VirtualChannels File

NOTICE

White space between tags and values can be either spaces or tabs.

```
// Virtual channel configuration

VirtualChannel
{
	Name:		Fp1F3
	Unit:
	Ref:		Fp1,1
	Ref:		F3,-1
}

VirtualChannel
{
	Name:		Fp2F4
	Unit:
	Ref:		Fp2,1
	Ref:		F4,-1
}

VirtualChannel
{
	Name:		F3C3
	Unit:
	Ref:		F3,1
	Ref:		C3,-1
}

VirtualChannel
{
	Name:		F4C4
	Unit:
	Ref:		F4,1
	Ref:		C4,-1
}

VirtualChannel
{
	Name:		C3P3
```

```
    Unit:
    Ref:          C3,1
    Ref:          P3,-1
}

VirtualChannel
{
    Name:         C4P4
    Unit:
    Ref:          C4,1
    Ref:          P4,-1
}

VirtualChannel
{
    Name:         P3O1
    Unit:
    Ref:          P3,1
    Ref:          O1,-1
}

VirtualChannel
{
    Name:         P4O2
    Unit:
    Ref:          P4,1
    Ref:          O2,-1
}

VirtualChannel
{
    Name:         Fp1F7
    Unit:
    Ref:          Fp1,1
    Ref:          F7,-1
}
```

EEG File Format

EEG (***.eeg**) files are tab-delimited text files used by Acq at acquisition time as a convenient way to enter information. They can be created manually or by using the Acq application via the EEG Channel Parameters dialog.

When created by Acq, the file contains the following five fields:

```
EEG_Hardware_Name    Given_Name  <X>   <Y>   <Z>
```

where the X, Y, Z positions are defined in the CTF MEG head coordinate system relative to the nasion, left ear, and right ear head coils. The file therefore contains the necessary information to assign each EEG hardware name to its corresponding EEG given name and specify its position in the head coordinate system. (See Appendix A: “CTF MEG Head Coordinate System” on page 149 for details.) If the file is created manually and only channel numbers occur in the first column, the software will automatically convert the numbers to their corresponding EEG hardware name — e.g., 1 = EEG001, 2 = EEG002, etc.

Electrode positions can be digitized using the Polhemus FASTRAK Digitizer. The positions are saved to a ***.pos** file (see “EEG POS File Format” on page 47), then read into Acq’s EEG Channel Parameters dialog (or copied and pasted into the ***.eeg** file, if this file is being created manually). In Acq, the new positions can be incorporated into the ***.eeg** file via the **File > Save EEG File** menu in the EEG Channel Parameters dialog.

The mapping of pin numbers to EEG given names follows an agreed-upon convention. If necessary, this mapping (and the actual electrode positions) can be changed from the EEG Channel Parameters dialog, or by entering the information into an ***.eeg** file, then using the **changeeeginfo** command-line program to apply the changes to a specific dataset. For more information on using **changeeeginfo**, see the program's online help (**changeeeginfo -help**) or the *Command Line Programs Guide* (PN900-0016).

Sample *.eeg File

3	Fp1	8.75022	2.54613	3.4883
4	Fp2	9.27878	0	3.62034
5	F4	4.6016	6.17664	3.27641
6	C3	4.65599	5.0072	7.88773
7	C4	5.07733	0	9.13753
8	P3	4.65599	-5.0072	7.88773
9	P4	4.6016	-6.17664	3.27641
10	O1	-0.80095	6.96084	2.8638
11	O2	-2.59453	6.41479	8.26253
12	F7	-3.10295	0	9.61925
13	F8	-2.59453	-6.41479	8.26253
14	T3	-0.80095	-6.96084	2.8638
15	T4	-6.77766	6.58543	2.03905
16	T5	-8.89148	5.65214	6.28001
17	T6	-10.0489	0	7.43935
18	F3	8.75022	-2.54613	3.4883
19	EOG1	0	0	0
20	EOG2	0	0	0

EEG POS File Format

The EEG electrode positions (***.pos**) file is a tab-delimited text file containing the positions of each EEG electrode applied to the patient's scalp. The file can be created using the Polhemus FASTRAK Digitizer. See the *Electrode Digitizer Guide* (PN900-0038) for details.

The first line of the file displays the number of electrodes. It is followed by a line for each electrode (i.e., pin number), specifying its position in the CTF MEG head coordinate system (see Appendix A: "CTF MEG Head Coordinate System" on page 149). The last three lines of the file contain coordinates for the nasion, left ear, and right ear head coils. The electrode X, Y, Z coordinates are relative to these fiducial points.

The electrode position lines in the file have the following format:

```
No. of electrodes
Pin#     <X>      <Y>      <Z>
Pin#     <X>      <Y>      <Z>
...
```

The EEG electrode positions file should be saved with a ***.pos** extension and transferred to the acquisition computer. Acq can directly read a ***.pos** file to obtain the measured electrode positions. In this case, the EEG given names do not change in the EEG Channel Parameters dialog.

Sample *.pos File

157

1	0.431229791489441	-7.44955899209228	-0.218700731086407
2	0.431954688796231	-7.40470052917514	-0.212681883279488
3	0.356598799157876	-7.43237716375314	-0.200677312881805

• • •

154	-8.96758329825463	2.3480441429386	6.76925540948713
155	-8.76065813844709	1.23082967917847	-1.43413965377128
156	-0.408795826475624	6.20088129143348	-3.72299460808597
157	-1.8499892328631	5.39793550624389	-5.53352937190388
nasion	10.1834297299661	1.52655665885959E-16	1.12296240015186E-16
left	-0.45167937818978	7.56111597262122	-5.87637577487143E-17
right	0.451679378189779	-7.56111597262122	9.54911063416608E-16

Head Coil File Format

The head coil (***.hc**) file is a text file created by the Acq application during head localization at the time of data acquisition.

The file contains three sets of coordinates:

Standard, or "nominal" X, Y, Z coordinates for the nasion, left ear, and right ear head coils. These are default (unmeasured) positions in the dewar coordinate system that will be associated with the data if localization fails or is not performed.

Measured X, Y, Z coordinates of the nasion, left ear, and right ear head coils, specified in the dewar coordinate system. These positions are obtained during head localization at the time of data acquisition. If localization fails or is not performed, they are set to the pre-defined nominal positions.

The same X, Y, Z coordinates as above for the nasion, left ear, and right ear head coils, but this time specified in the head coordinate system (see Appendix A: "CTF MEG Head Coordinate System" on page 149).

Sample *.hc File Format

standard nasion coil position relative to dewar (cm):
x = 5.65685
y = 5.65685
z = -27
standard left ear coil position relative to dewar (cm):
x = -5.65685
y = 5.65685
z = -27
standard right ear coil position relative to dewar (cm):
x = 5.65685
y = -5.65685
z = -27
measured nasion coil position relative to dewar (cm):
x = 7.02597
y = 6.63714
z = -23.5313
measured left ear coil position relative to dewar (cm):
x = -5.87293
y = 6.00096
z = -24.2616
measured right ear coil position relative to dewar (cm):
x = 5.62803
y = -5.98867
z = -24.3631
measured nasion coil position relative to head (cm):
x = 9.78161
y = -3.747e-16
z = 3.55271e-15
measured left ear coil position relative to head (cm):
x = -0.134499
y = 8.30604
z = 1.42109e-14
measured right ear coil position relative to head (cm):
x = 0.134499
y = -8.30604
z = -1.42109e-14

Head Localization Datasets

If head localization has been performed during the data recording, one or more of the following head localization datasets are saved in the dataset directory.

Dataset	Description
hz.ds	Contains raw data and localization results for the pre-run localization.
hz2.ds	Contains raw data and localization results for the post-run localization.
hz_t<*n*>.ds	Contains raw data for on-demand localization (i.e., where the user clicks the Acquisition Monitor's **Head Localization** button during the recording). The <*n*> in the dataset name is replaced with the last completed trial number (or "0" if head localization is requested before the first trial). The calculated head positions are saved in the hz_<*dataset_name*>.txt file.
hz_<*dataset_name*>.ds hz_<*dataset_name*>.txt	Contains raw data for inter-trial localization. This dataset consists of one trial of head localization for each trial in the dataset, collected immediately after each trial completes. The calculated head positions are saved in the hz_<*dataset_name*>.txt file.

Sample hz_<*dataset_name*>.txt File Format

The head localization results for inter-trial head localization are saved in a tab-delimited ASCII text file of the name **hz_<*dataset_name*>.txt**.

The first column contains the trial number corresponding to the previous trial. (Inter-trial head localization for a given trial is performed immediately after the trial completes.) If no trials have been collected when a head localization is requested, then the trial number will be zero.

The next N * 3 columns list the x, y, z coordinates (relative to the dewar), where "N" is the number of head localization coils.

The last three columns are for verification, and contain the distances between the nasion and left ear, nasion and right ear, and the left and right ear, respectively. These values should be consistent for all head localization sets. (Due to space restrictions, these columns are not shown below.)

All measurements are in centimeters.

	Na x	NA y	Na z	Le x	Le y	LE z	RE x	RE y	RE z
0	4.65583	4.93378	-25.2762	-5.93873	5.36478	-23.1331	4.8415	-5.80471	-22.977
1	4.66609	4.93962	-25.267	-5.94528	5.37264	-23.1206	4.8497	-5.80901	-22.9749
2	4.66286	4.93968	-25.2688	-5.95004	5.37115	-23.1185	4.84538	-5.79992	-22.9755
3	4.65298	4.93621	-25.275	-5.94897	5.37113	-23.1217	4.84627	-5.81024	-22.9758
4	4.69751	4.97723	-25.2259	-5.96039	5.37761	-23.1064	4.86001	-5.80785	-22.9719

Bad Channels File Format

The **BadChannels** file is a text file containing a list of channels that have been classified as "bad" using a CTF MEG application, such as Acq or DataEditor. See the *Data Acquisition Guide* (PN900-0006) and *DataEditor Guide* (PN900-0007) for more information.

Sample BadChannels File

MLO33
MRF52
MZO02

Bad Segments File Format

The **bad.segments** file is a text file containing a list of segments that have been classified as "bad" using DataEditor. See the *DataEditor Guide* (PN900-0007) for more information.

The file has the following format:

```
Trial#    StartTime    EndTime
```

where the start and end times are in units of seconds relative to the beginning of the trial. Trial numbering starts at "1".

Sample bad.segments File

```
68  0.0016                 0.1536
```

Marker File Format

Markers are used to mark individual data points of interest within the dataset. Each marker belongs to a specific marker set, which is defined for a particular type of event. For example, trigger markers, which are non-editable, are applied to data points during data acquisition. Other markers can be applied to data points during a DataEditor session or by running other CTF MEG applications, such as **addMarker**. These markers can be inserted, moved, and deleted. See the *DataEditor Guide* (PN900-0007) and *Command Line Programs Guide* (PN900-0016) for more information.

All marker set definitions and marker locations are stored in a text file called **MarkerFile.mrk** in the dataset directory. The marker information in this file is applied to the data when the data is viewed, processed, or analyzed.

NOTICE

The MarkerFile.mrk file requires exact line spacing, otherwise the software will fail to read the it properly. Two blank lines must be present between sections and three blank lines must be present at the end of the file.

Marker files contain the path to the dataset at the top of the file, followed by the number of marker sets in the file. A section for each marker set occurs next, with each section separated by exactly two lines. The sections contain the following information:

CLASSGROUPID	Describes the type of marker set: 0 = TriggerGroup 1 = TemplateGroup 2 = ConditionGroup (not supported) 3 = ManualGroup
NAME	Name of the marker set. The name must be unique within the dataset.
COMMENT	Comments can be inserted manually using DataEditor’s Marker dialog. Otherwise this field contains automatically generated comments inserted by other CTF MEG programs.
COLOR	The display color for the marker when the data is viewed in DataEditor.
EDITABLE	Indicates whether the marker is editable or not. (Trigger markers are not editable.)
CLASSID	Marker set index. This identifier starts at 1 and increments for each defined marker set.
NUMBER OF SAMPLES	The number of marked samples in the marker set.
LIST OF SAMPLES	The trial number and time point where each marker occurs, relative to the beginning of the trial. Trial numbering starts from zero and time is is seconds.

Sample MarkerFile.mrk File

```
PATH OF DATASET:
/home/meg/data/Anonymous.proc/DATASETS/Anonymous_EPI_01.ds


NUMBER OF MARKERS:
4


CLASSGROUPID:
+3
NAME:
V1
COMMENT:
(-0.500, -5.000, 5.000) cm
COLOR:
green
EDITABLE:
Yes
CLASSID:
1
NUMBER OF SAMPLES:
3
LIST OF SAMPLES:
TRIAL NUMBER    TIME FROM SYNC POINT (in seconds)
        +0                  +7.488
        +0                   +8.5
        +0                  +9.408


CLASSGROUPID:
+3
NAME:
V2
COMMENT:
(-1.000, -4.000, 7.000) cm
COLOR:
green
EDITABLE:
Yes
CLASSID:
+2
NUMBER OF SAMPLES:
4
```

LIST OF SAMPLES:

TRIAL NUMBER	TIME FROM SYNC POINT (in seconds)
+2	+0.06
+2	+8.296
+2	+11.144
+2	+11.272

CLASSGROUPID:
+3
NAME:
V3
COMMENT:
(7.000, -2.000, 7.000) cm
COLOR:
green
EDITABLE:
Yes
CLASSID:
+3
NUMBER OF SAMPLES:
2
LIST OF SAMPLES:

TRIAL NUMBER	TIME FROM SYNC POINT (in seconds)
+2	+34.716
+2	+56.82

CLASSGROUPID:
+3
NAME:
V4
COMMENT:
(4.000, -5.000, 4.500) cm
COLOR:
green
EDITABLE:
Yes
CLASSID:
+4
NUMBER OF SAMPLES:
2
LIST OF SAMPLES:

TRIAL NUMBER	TIME FROM SYNC POINT (in seconds)
+1	+21.58
+1	+38.432

Class File Format

A class file contains information about how the trials within a dataset are classified. Trials can be classified manually during a DataEditor session or by running other CTF MEG applications, such as **addTrialClass**. The CTF MEG software recognizes a predefined classification called “bad”, excluding from processing and analysis any trial with a classification beginning with these letters in any combination of upper and lower case. See the *DataEditor Guide* (PN900-0007) and *Command Line Programs Guide* (PN900-0016) for more information.

All class definitions and trials classifications are stored in a text file called **ClassFile.cls** in the dataset directory. This information in this file is applied to the data when the data is viewed, processed, or analyzed.

NOTICE

The ClassFile.cls file requires exact line spacing, otherwise the software will fail to read the it properly. Two blank lines must be present between sections and three blank lines must be present at the end of the file.

Class files contain the path to the dataset at the top of the file, followed by the number of classes in the file. A section for each class occurs next, with each section separated by exactly two lines. The sections contain the following information:

CLASSGROUPID	Describes the type of class: 0 = TriggerGroup 1 = TemplateGroup 2 = ConditionGroup (not supported) 3 = ManualGroup
NAME	Name of the class. The name must be unique within the dataset. Any class name that begins with the letter “bad” (in any combination of upper and lower case) classifies the trial as a bad trial.
COMMENT	Comments can be inserted manually using DataEditor’s Class dialog. Otherwise this field contains automatically generated comments inserted by other CTF MEG programs.
COLOR	The display color for the class when the data is viewed in DataEditor’s Single Channel Display window.
EDITABLE	Indicates whether the class is editable or not.
CLASSID	Class index. This identifier starts at 1 and increments for each defined class.
NUMBER OF TRIALS	The number of trials classified with this class name.
LIST OF TRIALS	The trial number of each classified trial. Trial numbering starts from zero.

Sample ClassFile.cls File

PATH OF DATASET:
/home/meg/data/tutorial/TerryTester/TerryTester_AEF_19980122_01.ds

NUMBER OF CLASSES:
2

CLASSGROUPID:
3
NAME:
BAD
COMMENT:
Generic bad trial classification
COLOR:
Red
EDITABLE:
Yes
CLASSID:
1
NUMBER OF TRIALS:
0
LIST OF TRIALS:
TRIAL NUMBER

CLASSGROUPID:
3
NAME:
Bad-Blink
COMMENT:
Eye blink detected
COLOR:
#c80000
EDITABLE:
Yes
CLASSID:
2
NUMBER OF TRIALS:
18
LIST OF TRIALS:
TRIAL NUMBER
+4

+6
+11
+19
+26
+28
+42
+49
+52
+58
+60
+64
+67
+70
+74
+75
+84
+93

Head Shape File Format

The head shape (***.shape**) file is a tab-delimited text file containing positions for each of the patient's head shape points. The file is created by MRIViewer during head shape extraction.

Points can be specified using the CTF MEG head coordinate system (see Appendix A: "CTF MEG Head Coordinate System" on page 149) or the MRI coordinate system (i.e., the sagittal, coronal, and axial voxel locations for each point).

The first line of the file displays the number of head shape points in the file. It is followed by a line for each point, specifying its position. The file has the following format:

```
No. of points
X      Y      Z
X      Y      Z
...
```

if the CTF MEG head coordinate system is used, or

```
No. of points
sagittal   coronal   axial
sagittal   coronal   axial
...
```

if the MRI coordinate system is used.

In order for the head shape file to be located by CTF MEG processing scripts, it must be named ***<patient_ID>*.shape**, and must reside in the MRI subdirectory of the patient's procedure folder. A ***<patient_ID>*.shape_info** file must also be present in the same directory to interpret the head shape file. See "Head Shape Info File Format" on page 67 for details.

Sample *.shape File

71571

-2.238	8.601	2.893
-1.919	8.548	3.203
-2.011	8.550	3.236
-2.103	8.551	3.269
-1.769	8.533	3.045
-1.861	8.535	3.078
-1.952	8.536	3.112
-2.044	8.538	3.145
-2.136	8.539	3.178
-2.228	8.541	3.211
-2.320	8.542	3.245
-1.802	8.521	2.954
-1.894	8.523	2.987
-1.986	8.524	3.020
-2.077	8.526	3.054
-2.169	8.527	3.087
-2.261	8.529	3.120
-2.353	8.530	3.154
-2.445	8.532	3.187
-1.927	8.511	2.896
-2.019	8.513	2.929
-2.111	8.514	2.963
-2.202	8.516	2.996
-2.294	8.517	3.029

• • •

Head Shape Info File Format

The head shape info (***.shape_info**) file contains information necessary for interpreting the patient's ***.shape** file (see "Head Shape File Format" on page 65). The ***.shape_info** file is created by MRIViewer during the head shape extraction process in the same directory as the ***.shape** file. It uses the Config Reader text format (see "Text Files" on page 7 for details).

The file first specifies its version number and the path to the patient's MRI. It then lists the locations of the patient's three fiducial points (nasion, left ear, and right ear) using the MRI coordinate system — sagittal (256 voxels moving left to right), coronal (256 voxels moving front to back), and axial (256 voxels moving top to bottom). The voxel size, or resolution, is specified next in millimeters per voxel for each direction. The **COORDINATES** tag indicates whether the corresponding head shape file (***.shape**) uses the MRI coordinate system (**MRI**), or the CTF MEG head coordinate system (**HEAD**) to specify the head shape positions (see Appendix A: "CTF MEG Head Coordinate System" on page 149 for details). The **SURFACE** tag indicates whether the shape points define the surface of the scalp or the cortical hull.

In order for the ***.shape_info** file to be located by CTF MEG processing scripts, it must be named **<*patient_ID*>.shape_info**, and must reside in the MRI subdirectory of the patient's procedure folder along with the **<*patient_ID*>.shape** file. When a ***.shape_info** file is used to create a head model, all the information is transferred to the head model file (***.hdm**), including the path and file name of the MRI. This information is also transferred to the analysis results (***.dip** and SAM ***.svl** and ***.wts** files) so they can be associated with the MRI used to create the model.

Sample *.shape_info File

NOTICE

White space between tags and values can be either spaces or tabs.

```
// ************************************
/ Headshape File Information
/ ************************************

MRI_Info
{
    VERSION:            2

    FILENAME:           /home/meg/data/Anonymous.proc/MRI/Anonymous.mri

    // Fid. Points    Sag      Cor      Axi
    NASION:           130.0    24.0     160.0
    LEFT_EAR:         48.0     119.0    183.0
    RIGHT_EAR:        207.0    116.0    203.0

    MM_PER_VOXEL_SAGITTAL:     0.97656250
    MM_PER_VOXEL_CORONAL:      0.97656250
    MM_PER_VOXEL_AXIAL:        0.97656250

    COORDINATES:        HEAD

    // Available SURFACE types are: SCALP CORTICAL_HULL SKULL
    SURFACE:          CORTICAL_HULL
}
```

Head Model File Format

Head model (*.**hdm**) files can be created by three different CTF MEG programs: MRIViewer, DipoleFit, and the **localSpheres** command-line program. They use the Config Reader format (see "Text Files" on page 7), with line elements separated by tabs or spaces.

When created by MRIViewer, the head model is a single-sphere model, based on the fiducial points from the patient's MRI. When created by DipoleFit, the head model is a single-sphere model, based on the MEG fiducial points from the patient's dataset. However, both MRIViewer and DipoleFit can load other head model files and combine the information into the current head model. For example, you can create a head model using MRIViewer, open an ***.hdm** file for the same patient created by DipoleFit, then save the combined head model information to produce a single-sphere head model file containing both MRI and MEG/EEG information.

When created by **localSpheres**, the head model is a multiple local spheres model, based on both MRI data (from the patient's head shape (***.shape**), and shape info (***.shape_info**) files) and MEG information from the patient's dataset.

A head model (***.hdm**) file must be present before dipole or SAM analysis can be performed.

Head Model File Structure (Version 6.0)

A head model (***.hdm**) file consists of the following classes, tags, and line elements:

Class	Tag	Line Elements	Description
File_Info	VERSION	Version #	Head model file version.
	DATE	Date	Date the file was initially created.
	PATIENT	Patient ID	Name or ID of the patient to whom the head model file pertains.
	STATUS	Status	Status of the dipole information. Values can be the following: • **Not reviewed** • **Accepted** • **Rejected** • empty string ("") = unknown
Model	MODEL_TYPE	Model type ID	Values can be the following: • **EEG_ONLY** • **MEG_ONLY** • **EEG_AND_MEG** or **MEG_AND_EEG** • **MRI_FILE** (optional) (full path name to the MRI file used to create the head model
	MRI_File	File path	The full path and name of the MRI file associated with the dataset.
MEG_Sphere	ORIGIN_X	X	X position (in cm) of the MEG sphere origin.
	ORIGIN_Y	Y	Y position (in cm) of the MEG sphere origin.
	ORIGIN_Z	Z	Z position (in cm) of the MEG sphere origin.
	RADIUS	Radius	Radius (in cm) of the MEG sphere.

Class	Tag	Line Elements	Description
EEG_Sphere (optional)	NUM_SHELLS	Number of shells	The number of concentric shells in the model.
	ORIGIN_X	X	X position (in cm) of the EEG sphere origin.
	ORIGIN_Y	Y	Y position (in cm) of the EEG sphere origin.
	ORIGIN_Z	Z	Z position (in cm) of the EEG sphere origin.
	RADIUS1	Radius	Radius (in cm) of the shell 1 (the innermost shell). (Note that the order of shells is reversed from version 2.1 head model files.)
	RADIUS2	Radius	Radius (in cm) of the shell 2.
	RADIUS3	Radius	Radius (in cm) of the shell 3.
	RADIUS4	Radius	Radius (in cm) of the shell 4.
	CONDUCTIVITY1	Conductivity	The conductivity (in ohm - m^{-1}) of shell 1.
	CONDUCTIVITY2	Conductivity	The conductivity (in ohm - m^{-1}) of shell 2.
	CONDUCTIVITY3	Conductivity	The conductivity (in ohm - m^{-1}) of shell 3.
	CONDUCTIVITY4	Conductivity	The conductivity (in ohm - m^{-1}) of shell 4.
Voxel_Resolution (optional; necessary only if **MRI_Fid_Points** is specified.)	SAGITTAL	Resolution	Resolution (in mm per voxel) in the sagittal direction.
	CORONAL	Resolution	Resolution (in mm per voxel) in the coronal direction.
	AXIAL	Resolution	Resolution (in mm per voxel) in the axial direction.

Class	Tag	Line Elements	Description
MRI_Fid_Points (optional)	NASION	Sag Cor Axi	Voxel location of nasion fiducial point, based on MRI coordinate system (sagittal, coronal, and axial directions).
	LEFT_EAR	Sag Cor Axi	Voxel location of left ear fiducial point, based on MRI coordinate system (sagittal, coronal, and axial directions).
	RIGHT_EAR	Sag Cor Axi	Voxel location of right ear fiducial point, based on MRI coordinate system (sagittal, coronal, and axial directions).
MEG_Fid_Points (optional)	Nasion	xp(cm) yp(cm) zp(cm)	MEG nasion head coil fiducial point relative to dewar (in cm), based on the CTF MEG head coordinate system (X, Y, Z coordinates).
	LeftEar	xp(cm) yp(cm) zp(cm)	MEG left ear head coil fiducial point relative to dewar (in cm), based on the CTF MEG head coordinate system (X, Y, Z coordinates).
	RightEar	xp(cm) yp(cm) zp(cm)	MEG right ear head coil fiducial point relative to dewar (in cm), based on the CTF MEG head coordinate system (X, Y, Z coordinates).
	Nominal	Flag	Indicates whether nominal or measured values are used for the MEG fiducal points: • **TRUE** (nominal values are used) • **FALSE** (measured values are used)
Multisphere_Data (optional; this information is produced by the **localSpheres** program)	SEARCH_RADIUS	Radius	Radius used to select head shape points for fitting a local sphere for each channel.

Class	Tag	Line Elements	Description
	HEADSHAPE_FILE	File path	The full path and name of the head shape file used.
	SURFACE_TYPE	Head shape point extraction method	The head shape point extraction method used to create the head shape file. Values can be the following: • **SCALP** (head shape points define the scalp surface) • **CORTICAL_HUL**L (head shape points define the cortical hull surface) • **SKULL** (head shape points define the inner skull surface; *reserved for future use*)
	HEAD_POS	NA_x NA_y NA_z LE_x LE_y LE_z RE_x RE_y RE_z Nominals	X, Y, Z coordinates (in cm) of the nasion localization coil position. X, Y, Z coordinates (in cm) of the left ear localization coil position. X, Y, Z coordinates (in cm) of the right ear localization coil position. Indicates whether nominal or measured values are used for the position of the localization coils: • **TRUE** (nominal values are used) • **FALSE** (measured values are used)
	Channel Name (repeated for each channel)	X Y Z Radius	X, Y, Z coordinates and radius (in cm) of local sphere for the specified channel.

Class	Tag	Line Elements	Description
Marker_Data (optional)	Label (repeated for *n* labels)	X Y Z	Marker label, followed by three floating point values representing the 3-D location vector (X, Y, Z) of the marker in the head coordinate system (see Appendix A: “CTF MEG Head Coordinate System” on page 149 for details). Marker data is created using MRIViewer.

Sample *.hdm File

NOTICE

White space between tags and values can be either spaces or tabs.

```
// *************************************
//CTF Head Model:Dipole Parameter File
/ *************************************

// Codes:

//    dipole colour:
//    YELLOW = 0, GREEN = 1, RED = 2, CYAN = 3,
//    CYAN = 3, MAGENTA = 4, WHITE = 5, BLACK = 6
// dipole shape:
//    FILLED_CIRCLE = 0, FILLED_SQUARE = 1, FILLED_TRIANGLE = 2
//    HOLLOW_CIRCLE = 3, HOLLOW_SQUARE = 4, HOLLOW_TRIANGLE = 5
// display flags:
//    true = 1, false = 0
//    position/moment/orientation constraint flags:
//    FREE = 0,  FIXED = 1, RANGED = 2,  RADIAL = 3, TANGENTIAL = 4
//    CO-LOCATED = 5, MIRROR SYMMETRIC = 6, ORTHOGONAL_TO_DIPOLE = 7

File_Info
{
    VERSION:          CTF_HEAD_MODEL_FILE_VERSION_6.0
    DATE:             28-Nov-2006 14:06
    PATIENT:          Anonymous
    STATUS:
}

Model
{
    MODEL_TYPE:       MEG_ONLY
    MRI_FILE:         /home/meg/data/Anonymous.proc/MRI/Anonymous.mri
}
```

```
MEG_Sphere
{
    ORIGIN_X:       -0.360
    ORIGIN_Y:       0.695
    ORIGIN_Z:       6.267
    RADIUS:         7.142
}

Voxel_Resolution
    // Resolution in mm per voxel
    SAGITTAL:   0.97656250
    CORONAL:    0.97656250
    AXIAL:      0.97656250
}

MRI_Fid_Points
{
    // Fid. Pts      Sag     Cor     Axi
    NASION:          130     24      160
    LEFT_EAR:        48      119     183
    RIGHT_EAR:       207     116     203
}

MultiSphere_Data
{
    SEARCH_RADIUS:      4.500
    HEADSHAPE_FILE:     anonymous_hull.shape
    SURFACE_TYPE:       CORTICAL_HULL

    // 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:    7.022  7.121  -22.838  -4.871  5.809  -25.215  6.132  -4.764 -25.844  FALSE

    // Multiple Sphere locations in cm
    //              X       Y       Z       Radius
    BG1:            -0.448  0.452   4.898   7.994
    BG2:            -0.448  0.452   4.898   7.994
    BG3:            -0.448  0.452   4.898   7.994
    BP1:            -0.631  0.270   4.088   8.795
    BP2:            -0.631  0.270   4.088   8.795
    BP3:            -0.631  0.270   4.088   8.795
    BR1:            -0.391  0.059   4.325   8.563
    BR2:            -0.391  0.059   4.325   8.563
    BR3:            -0.391  0.059   4.325   8.563
    G11:            -0.400  0.417   5.171   7.730
    G12:            -0.400  0.417   5.171   7.730
    G13:            -0.400  0.417   5.171   7.730
    G22:            -0.437  0.438   4.699   8.184
```

G23:	-0.437	0.438	4.699	8.184
P11:	-0.526	0.328	4.386	8.495
P12:	-0.526	0.328	4.386	8.495
P13:	-0.526	0.328	4.386	8.495
P22:	-0.648	0.295	4.449	8.454
P23:	-0.648	0.295	4.449	8.454
Q11:	0.047	0.792	4.715	8.065

...

MZF03:	0.327	0.373	4.012	8.292
MZO01:	-2.044	0.936	5.276	6.731
MZO02:	-3.098	0.748	5.472	5.756
MZP01:	-0.587	0.570	4.787	8.092
MZP02:	-0.513	0.932	4.613	8.273

}

MRI File Format

CTF MRI File Structure (Version 4.1)

The CTF MRI file created by MRIConverter (and used by MRIViewer) is a binary file with the file extension "**.mri**". The file format is based on the VSM proprietary CPersist object class described in Appendix B: "CPersist Object" on page 153. The MRI file comprises a set of header tags and their related values, followed by the actual MRI slice data. To display the tags in a CTF MRI file, use the command-line application **mrihead**. See *Command Line Programs Guide* (PN900-0016) for details.

Tag Organization

Tags in the CTF MRI file can be categorized into the following two classes:

header

slice data

Header Tags

Header tags contain general information about the patient, examination, MRI acquisition, and head model. For details on header tags see Table 7 on page 80.

Slice Data Tags

The file contains 256 slices, with slices always in the sagittal direction. Each slice is individually tagged from "**_CTFMRI_SLICE_DATA#00001**" to "**_CTFMRI_SLICE_DATA#00256**".

Slice data contains 256 x 256 pixels, starting at the top left corner and scanning downwards row by row; i.e., the coronal position changes faster than the axial position. The sagittal position is the slice number.

The slice sequence is standardized to “sagittal, left to right” whenever MRI data is ported into a CTF MRI file. On exporting, the data orientation is determined by the value of the mandatory tag “**_IMAGEPLANE_ORIENTATION**”.

Pixels are stored in two-byte words using Big Endian order.

For details on slice data, see the tag “**_CTFMRI_SLICE_DATA#XXXXX**” in Table 7 (page 87).

CTF MRI File Tags

The following table lists tags found in the CTF MRI file.

The column “**M/O**” indicates if the tag is Mandatory or Optional. Optional tags may be set to default nominal values. Numbers in this column have the following meanings:

1. Mandatory only if the modality is MR
2. Mandatory only if the modality is CT
3. Mandatory only when the CTF MRI is converted from a DICOM series

The column “**D**” indicates if the value is exported to DICOM.

Table 7: CTF MRI File Tags

Tag (Attribute)	Value Type	Description	M O	D
_CTFMRI_VERSION	string	Specifies the version of the CTF MRI file.	M	N
_CTFMRI_UID	string	Unique ID for each CTF MRI file.	M	N
_HDM_NASION	string	Location of nasion head coil in voxels.	O	N
_HDM_LEFTEAR	string	Location of left ear head coil in voxels.	O	N

Table 7: CTF MRI File Tags

Tag (Attribute)	Value Type	Description	M O	D
_HDM_RIGHTEAR	string	Location of right ear head coil in voxels.	O	N
_HDM_DEFAULTSPHERE	string	Location and diameter of default sphere in voxels.	O	N
_CTFMRI_ROTATE	string	Image plane rotation angles in degrees.	O	N
_CTFMRI_SIZE	short	Cube dimensions (Y x Y x Y) in voxels. The value is fixed at 256.	M	N
_CTFMRI_DATASIZE	short	Pixel data size in bytes. The value is fixed at 2.	M	N
_CTFMRI_MMPERPIXEL	string	Voxel size, sagittal, coronal, and axial values respectively, in mm.	M	N
_HDM_HEADORIGIN	string	Head origin in voxels.	O	N
_CTFMRI _ORTHOGONALFLAG	short	0 = not orthogonalized !0 = is orthogonalized. This tag is not used by CTF MEG software.	O	N
_CTFMRI _INTERPOLATEDFLAG	short	0 = not interpolated !0 = is interpolated. This tag is not used by CTF MEG software.	M	N
_CTFMRI _TRANSFORMMATRIX	string	Four-by four rotation matrix between the CTF MRI and CTF MEG head coordinate systems.	O	N
_CTFMRI_COMMENT	string	User comments related to the CTF MRI file.	O	N
_PATIENT_NAME	string	Identifies patient by patient name, patient ID, birthday, and sex. Patient name is stored in the same format as PACS.	M	Y

Table 7: CTF MRI File Tags

Tag (Attribute)	Value Type	Description	M O	D
_PATIENT_ID	string	Unique Patient ID within PACS stored in the same format as PACS.	M	Y
_PATIENT_BIRTHDAY	string	Patient birthday stored in the same format as PACS.	M	Y
_PATIENT_SEX	short	Sex can be one of following values: 0 = male, 1 = female, 2 = other	M	Y
_STUDY_ID	string	Unique study ID within patient studies. Study ID is stored in the same format as PACS.	O	Y
_STUDY_DATETIME	string	Date and time when original image was scanned.	O	N
_STUDY_DATE	string	Date when the original study was created. Used as study date in DICOM files created from CTF MRI file.	3	Y
_STUDY_TIME	string	Time when the original study was created. Used as study time in DICOM files created from CTF MRI file.	3	Y
_STUDY_DESCRIPTION	string	Study description from original study; may be replaced by new description.	O	Y
_STUDY_COMMENTS	string	Study comments from original study; may be replaced by new comments.	O	Y
_STUDY _ACCESSIONNUMBER	string	Study accession number from original study; must be replaced by new accession number unique within PACS.	O	Y

Table 7: CTF MRI File Tags

Tag (Attribute)	Value Type	Description	M O	D
_SERIES_MODALITY	string	Identifies modality used to collect the series. Values can be any of the following: • MRI • CT • PET • SPECT • OTHER	3	Y
_SERIES_DATE	string	Date when the original series was created; used as series date in DICOM files created from CTF MRI file.	3	Y
_ SERIES_TIME	string	Time when the original series was created; used as series time in DICOM files created from CTF MRI file.	3	Y
_SERIES_DESCRIPTION	string	Series description from original series; may be replaced by new description.	O	Y
_EQUIP_MANUFACTURER	string	Identifies manufacturer of modality used to collect the original series.	O	Y
_EQUIP_MODEL	string	Identifies model of modality used to collect the original series.	O	Y
_EQUIP_INSTITUTION	string	Identifies institution where modality is located and where the original series was collected.	O	Y
_IMAGE _REFERENCEDUID	string	UID of any of original DICOM images used in conversion back to DICOM to keep a reference to the original series.	O	Y

Table 7: CTF MRI File Tags

Tag (Attribute)	Value Type	Description	M O	D
_REFERENCE_UID	string	Unique frame of reference UID for the spatial location and orientation of each slice. Each series has a single frame of reference UID. All images in the series that share the same frame of reference UID must be spatially related to each other.	O	Y
_REFERENCE _INDICATOR	string	Position Reference Indicator specifies the part of the patient's anatomy that was used as an anatomical reference point associated with a specific Frame of Reference UID.	O	Y
_IMAGEPLANE _LOCATION	string	X, Y, Z coordinates of the upper left hand corner of the first slice in sagittal view left to right, in mm.	3	Y
_IMAGEPLANE _ORIENTATION	int	CTF MRI files are always sagittal, left-right. This is the orientation used for import/export via DICOM. 0 = Non-supported view 1 = Sagittal view, Left-Right 2 = Coronal view, Posterior-Anterior 3 = Axial view, Superior-Inferior 4 = Coronal view, Anterior-Posterior 5 = Axial view, Inferior-Superior 6 = Sagittal view, Right-Left	3	Y
_IMAGEPIXEL _INTERPRETATION	string	Must be MONOCHROME2	O	Y
_MRIMAGE _SEQUENCENAME	string	User defined sequence name from original series.	O	Y

Table 7: CTF MRI File Tags

Tag (Attribute)	Value Type	Description	M O	D
_MRIMAGE _SCANNINGSSEQUENCE	string	Scanning sequence of original series. Must be one of the values enumerated in DICOM standard (part 3, Table C.8-4).	3	Y
_MRIMAGE _SEQUENCEVARIANT	string	Sequence variant of original series. Must be one of enumerated values in DICOM standard (part 3, Table C.8-4).	1	Y
_MRIMAGE _REPETITIONTIME	string	Repetition time carried over from original series without any modification.	O	Y
_MRIMAGE_ECHOTIME	string	Echo time carried over from original series without modification.	O	Y
_MRIMAGE _INVERSIONTIME	string	Inversion time carried over from original series without modification.	O	Y
_MRIMAGE_AVERAGES	string	Number of averages carried over from original series without modification.	O	Y
_MRIMAGE_FREQUENCY	string	Scanning frequency carried over from original series without modification.	O	Y
_MRIMAGE _IMAGEDNUCLEUS	string	Imaged nucleus carried over from original series without modification	O	Y
_MRIMAGE _FIELDSTRENGTH	string	Magnetic field strength carried over from original series without modification.	O	Y
_MRIMAGE_FLIPANGLE	string	Flip angle carried over from original series without modification.	O	Y
_CTIMAGE _RESCALEINTERCEPT	string	Rescale interception carried over from original series without modification.	**2**	Y

Table 7: CTF MRI File Tags

Tag (Attribute)	Value Type	Description	M O	D
_CTIMAGE _RESCALESLOPE	string	Rescale slope carried over from original series without modification.	2	Y
_VOILUT_WINDOWWIDTH	double	Contrast of the original images.	O	Y
_VOILUT _WINDOWCENTER	double	Brightness of the original images.	O	Y
_SPECIFICCHARSET	string	Specific character set used to decode values of most of string type DICOM attributes.	3	Y
_DICOMSOURCE_HASVALIDIMAGE	int	Boolean flag to indicate whether the source image contains valid information. 0 = No 1 = Yes	3	Y
_DICOMSOURCE_NUMBER_SLICES	unsigned short	Number of slices in the source DICOM series.	3	Y
_DICOMSOURCE_NUMBER_ROWS	unsigned short	Number of rows in the source DICOM series.	3	Y
_DICOMSOURCE_NUMBER_ COLUMNS	unsigned short	Number of columns in the source DICOM series.	3	Y
_DICOMSOURCE_SLICE_SPACING	double	Slice spacing in millimeters in the source DICOM series.	3	Y
_DICOMSOURCE_SLICE_THICKNESS	double	Slice thickness in millimeters in the source DICOM series.	3	Y
_DICOMSOURCE_ROW_SPACING	double	Row spacing (resolution) in millimeters in the source DICOM series.	3	Y
_DICOMSOURCE_COLUMN_ SPACING	double	Column spacing (resolution) in millimeters in the source DICOM series.	3	Y
_DICOMSOURCE_ROW_ ORIENTATION	CString	Row orientation (slice view direction) in the source DICOM series.	3	Y

Table 7: CTF MRI File Tags

Tag (Attribute)	Value Type	Description	MO	D
_DICOMSOURCE_COLUMN_ORIENTATION	CString	Column orientation (slice view direction) in the source DICOM series.	3	Y
_DICOMSOURCE_LOCATION_GAP	CString	Used internally by the system.	3	Y
_DICOMSOURCE_SLICE_LOCATION	CString	(One for each slice in source DICOM series.) X, Y, Z location of the slice.	3	Y
_DICOMSOURCE_CTF_TO_SOURCE_SLICE	CString	(One or more for each slice read from the source DICOM series.) Maps the DICOM source slice to the CTF MRI slice as if the CTF MRI slices were resliced in the original source slice direction.	3	Y
_CTFMRI_SLICE _DATA#XXXXX	binary	XXXXX = slice number (1 to 256). E.g., _CTFMRI_SLICE _DATA#00256	M	Y

1 — Mandatory only if the modality is MR
2 — Mandatory only if the modality is CT
3 — Mandatory only when the CTF MRI is converted from a DICOM series

Dipole File Format

Dipole (***.dip**) files are used by the DipoleFit GUI application (or **dfit** command-line program) prior to a dipole fit to obtain initial dipole parameters that are typical for a particular study. These parameters are used as a starting point (i.e., an initial guess) for the dipole fit analysis. After the fit, the actual results for each dipole can be saved back to the file in the "Fit Results" section. The format of this section depends on the type of fit performed. Two main types are possible:

Moving dipole – A separate fit is performed for each time point (sample) in a time window. This option allows for a different position, orientation, and moment per dipole for each sample in the window unless constraints are explicitly applied. (Note that a moving dipole fit can also be performed over a single sample.)

Spatio-temporal fit – A separate fit is performed for each time point (sample) in a time window. This option allows only the moment to vary from sample to sample, while the dipole position and orientation remain fixed.

Dipole files use the Config Reader format (see "Text Files" on page 7), with line elements separated by tabs or spaces.

Dipole File Structure (Version 6.0)

All dipole (***.dip**) files begin with head model information identical to that stored in the head model (***.hdm**) file, followed by dipole parameter information. This information may include fit results, if the file was created (or modified) by DipoleFit after a dipole fit analysis is performed.

For a description of the head model classes, tags, and line elements in the dipole file, see “Head Model File Format” on page 69. The dipole parameters are described below.

Class	Tag	Line Elements	Description
Dipoles	Dipole Index	Numeric index of the dipole starting from 1.	
		xp yp zp	X, Y, Z coordinates (in cm) for the dipole’s 3-D vector (position).
		xo yo zo	X, Y, Z coordinates (in cm) for the dipole’s unit-length orientation vector.
		Moment	The dipole’s moment in nanoAmpere-meters (1.0 * 10^{-9} Ampere-meters).
		Label	Alphanumeric label for the dipole (maximum of 32 characters).
Dipole_Flags	Dipole Index	Numeric index of the dipole starting from 1.	
		Color code	The dipole’s color when displayed. Values can be any of the following: • **0** = Yellow • **1** = Green • **2** = Red • **3** = Cyan • **4** = Magenta • **5** = White • **6** = Black
		Shape code	The dipole’s shape when displayed. Values can be any of the following: • **0** = Filled circle • **1** = Filled square • **2** = Filled triangle • **3** = Hollow circle • **4** = Hollow square • **5** = Hollow triangle

Class	Tag	Line Elements	Description
		Direction flag	Indicates whether the dipole's orientation is displayed: • **0** = False (not displayed) • **1** = True (displayed)
		Label flag	Indicates whether the dipole's label is displayed: • **0** = False (not displayed) • **1** = True (displayed)
		Error volume flag	Indicates whether the dipole's error volume is displayed: • **0** = False (not displayed) • **1** = True (displayed)
Dipole_Constraints	Index	Numeric index of the dipole starting from 1.	
		Orientation constraint code	Constrains the dipole's orientation to restrict allowable values during the fit. Values can be any of the following: • **0** = Free (no constraints during the fit) • **1** = Fixed (orientation does not change during the fit) • **2** = Ranged (restricted to max./min. values) • **3** = Radial (orientation can contain a component that is radial to the surface of the volume conductor (**obsolete; this value is no longer used**) • **4** = Tangential (orientation must be tangential to the surface of the volume conductor (**obsolete; this value is no longer used**) • **7** = Orthogonal to paired dipole (orientation is always orthogonal to a linked dipole)

Class	Tag	Line Elements	Description
		Position constraint code	Constrains the dipole's position to restrict allowable values during the fit. Values can be any of the following: • **0** = Free (no constraints during the fit) • **1** = Fixed (position does not change during the fit) • **5** = Co-located to paired dipole (position is the same as the paired dipole; **obsolete; this value is no longer used**) • **6** = Mirror symmetric to paired dipole (position is mirror symmetric about midline to paired dipole; **obsolete; this value is no longer used**)
		Moment constraint code	Constrains the dipole's moment to restrict allowable values during the fit. Values can be any of the following: • **0** = Free (no constraints during the fit) • **1** = Fixed (moment does not change during the fit) • **2** = Ranged (restricted to max./ min. values)
		Min.radius Max. radius	Values (in cm) for the minimum and maximum radius used for the "Ranged" constraint.
		Min. moment Max. moment	Values (in nanoAmpere-meters) for the minimum and maximum moment used for the "Ranged" constraint.
		Dipole index	Index of another dipole that this dipole is paired to. Used for the "Co-located", "Mirror-symmetric", and "Orthogonal" constraints (**obsolete; this value is no longer used**).

Class	Tag	Line Elements	Description
Dipole_FitInfo	Dipole Index	Numeric index of the dipole starting from 1.	
		Trial index	Index of trial used for the fit. Numbering starts from 1.
		Start sample index	First sample of the time window relative to the start of the trial. Numbering starts from 1.
		Latency (s)	First sample of the time window specified in seconds, relative to the trial synchronization (time zero).
		Number of points	Number of samples in the time window.
		Fit error	If fit is performed over one sample only, this value is the total weighted fit error for the sample. If the fit is performed over a time window, it is the total weighted fit error for the first sample in the range. The fit error is either a percentage if a least-squares fit is performed, or the reduced chi squared value if a chi squared fit was performed.
		MEG fit error	Fit error computed for the MEG channels. If fit is performed over one sample only, this value is the total MEG fit error (over all selected MEG channels) for the sample. If the fit is performed over a time window, it is the total MEG fit error for the first sample in the range.
		EEG fit error	Fit error computed for the EEG channels. If fit is performed over one sample only, this value is the total EEG fit error (over all selected EEG channels) for the sample. If the fit is performed over a time window, it is the total EEG fit error for the first sample in the range.

Class	Tag	Line Elements	Description
		Error type	The type of error processing performed. Values can be any of the following: • **0** = LEAST_SQUARES (normalized least-squares error) • **1** = CHI_SQUARE_VARIANCE (reduced chi square error) • **2** = CHI_SQUARE_PLUS_MINUS (reduced chi square error using plus-minus average) • **3** = CHI_SQUARE_USER DEFINED
		Fit type	The type of fit performed. Values can be either of the following: • **0** = SPATIO-TEMPORAL • **1** = MOVING DIPOLE
		Classification flag	Indicates whether the dipole is classified as good or bad: • **0** = bad • **1** = good
		Dataset	Full path and name (including the ***.ds** extension) of the dataset used for the fit.
Error_Volumes	Dipole Index	Numeric index of the dipole	
		ax ay az	X, Y, Z coordinates (in cm) of the major axis vector of the dipole's confidence volume.
		bx by bz	X, Y, Z coordinates (in cm) of minor axis vector of the dipole's confidence volume.
		cx cy cz	X, Y, Z coordinates (in cm) of the intermediate axis vector of the dipole's confidence volume.

Class	Tag	Line Elements	Description
		ox oy oz	X, Y, Z coordinates (in cm) of the origin (centroid) of the dipole's confidence volume.
		Confidence level	The confidence level — i.e., the percentage of dipole positions that must fall within the ellipsoid to create the confidence (error) volume.
Dipole_Head-Motion	Dipole Index	Numeric index of the dipole starting from 1.	
		MaxMotion	The distance (in centimeters) between the actual head position and the one used in the fit. If **USE_CONTINUOUS_HEAD_POS** (see page 97) is set to TRUE, the *actual* head position for each sample is used for a moving dipole fit, and an *average* head position is used for a spatio-temporal fit. If set to FALSE, the *default* head position (i.e., the one saved with the dataset) is used for both fit types. A MaxMotion value of zero indicates that a moving dipole fit was performed using CHL data. A negative value means that head motion is unknown.
Fit	FIT_TYPE	Fit type	The type of fit performed. Values can be either of the following: • **SPATIO_TEMPORAL** • **MOVING_DIPOLE**
	WIN_START	Start sample index	Start sample for time window of the dipole fit.
	WIN_POINTS	Number of samples	Number of samples in time window of the dipole fit.
	DATA_FILE	Dataset name	Full path name of the dataset that was used in the dipole fit.

Class	Tag	Line Elements	Description
	VERSION	Version	Version of the file's fit parameters record format. Current version is 4.
	ERROR_TYPE	Error type	The type of error processing performed. Values can be any of the following: • **0** = LEAST_SQUARES (normalized least-squares error) • **1** = CHI_SQUARE_VARIANCE (reduced chi square error) • **2** = CHI_SQUARE_PLUS_MINUS (reduced chi square error using plus-minus average) • **3** = CHI_SQUARE_USER _DEFINED
	MEG_WEIGHT	MEG to EEG proportion	The proportion of MEG to EEG data to use for modeling. E.g., • **1.0** = use only MEG data • **0.5** = use 50% MEG and 50% EEG data • **0.0** = use only EEG data
	INTEGRATION_ ORDER	Integration order code	Integration order used. Values can be any of the following: • **1** = 1 point (first order) • **2** = 3 point (second order) • **3** = 4 point (third order) • **4** = 6 point (fourth order) • **5** = 7 point (fifth order) • **7** = 12 point (seventh order)
	USE_PRE-CEDING_FIT	Preceding fit flag	Indicates whether to use the initial guess dipole for each sample in the fit window or the preceding fit results: • **TRUE** = use preceding fit results • **FALSE** = use initial guess

Class	Tag	Line Elements	Description
	FLIP_NEGATIVE_DIPOLES	Flip negative dipoles flag	Indicates whether to automatically flip the orientation of a dipole's vector whenever the dipole moment is negative: • **TRUE** = automatically flip the dipole's orientation when its moment is negative • **FALSE** = leave the dipole's orientation as is when its moment is negative
	USE_CONTINUOUS_HEAD_POS	Use continuous head position flag	Indicates whether to use Continuous Head Localization (CHL) position data in the head model when performing a dipole fit. • **TRUE** = use CHL position data in the head model • **FALSE** = do not use CHL position data in the head model
	TOT_ERROR	Total error	Total error (%) of the dipole fit. This is the least-squares error of the minimization summed over all channels, normalized by the power summed over all channels, and expressed as a percentage. For spatio-temporal fits, this value is also summed over all time points in the fit window.
	Dipole Index	Numeric index of the initial guess dipole.	
		xp yp zp	X, Y, Z coordinates (in cm) for the dipole's 3-D vector (position).
		xo yo zo	X, Y, Z coordinates (in cm) for the dipole's unit-length orientation vector.
		Moment	The dipole's moment in nanoAmpere-meters (1.0 * 10^{-9} Ampere-meters).

Class	Tag	Line Elements	Description
		Label	Alphanumeric label for the dipole (maximum of 32 characters).
Fit_Results (Moving Dipole)	Dipole Index	The Numeric index of the dipole starting from 1.	
		Trial index	Index of trial used for the fit. Numbering starts from 1.
		Sample	Time point in the sample index. Index starts from 1.
		Latency (s)	Time point in seconds relative to the trial synchronization (time zero).
		xp yp zp	X, Y, Z coordinates (in cm) for the dipole's 3-D vector (position).
		xo yo zo	X, Y, Z coordinates (in cm) for the dipole's unit-length orientation vector.
		Moment	The dipole's moment in nanoAmpere-meters = 1.0 * 10^{-9} Ampere-meters.
		ax ay az	X, Y, Z coordinates (in cm) of the major axis vector of the dipole's confidence volume.
		bx by bz	X, Y, Z coordinates (in cm) of the minor axis vector of the dipole's confidence volume.
		cx cy cz	X, Y, Z coordinates (in cm) of the intermediate axis vector of the dipole's confidence volume.
		ox oy oz	X, Y, Z coordinates (in cm) of the origin (centroid) of the dipole's confidence volume.

Class	**Tag**	**Line Elements**	**Description**
		Confidence level	The percentage of dipole positions that must fall within the ellipsoid to create the confidence (error) volume.
		Fit error	Total weighted fit error for the sample. The fit error is either a percentage if a least-squares fit is performed, or the reduced chi squared value if a chi squared fit was performed.
		MEG fit error	Fit error computed for the MEG channels. If the fit is performed over a time window, it is the total MEG fit error for the first sample in the range.
		EEG fit error	Fit error computed for the EEG channels. If the fit is performed over a time window, it is the total EEG fit error for the first sample in the range.
		Label	Label for the dipole.
Fit_Results (Spatio-Temporal)	Dipole Index	Numeric index of the dipole starting from 1.	
		xp yp zp	X, Y, Z coordinates (in cm) of the dipole's 3-D vector (position) for all samples.
		xo yo zo	X, Y, Z coordinates (in cm) of the dipole's unit-length orientation vector for all samples.
		ax ay az	X, Y, Z coordinates (in cm) of the major axis vector of the confidence volume for all samples.
		bx by bz	X, Y, Z coordinates (in cm) of the minor axis vector of the confidence volume for all samples.

Class	Tag	Line Elements	Description
		cx cy cz	X, Y, Z coordinates (in cm) of the intermediate axis vector of the confidence volume for all samples.
		ox oy oz	X, Y, Z coordinates (in cm) of the origin (centroid) of the confidence volume for all samples.
		Confidence level	The percentage of dipole positions that must fall within the ellipsoid to create the confidence (error) volume.
		Fit error	Total weighted fit error for all samples. The fit error is either a percentage if a least-squares fit is performed, or the reduced chi squared value if a chi squared fit was performed.
		MEG fit error	Fit error computed for the MEG channels for all samples.
		EEG fit error	Fit error computed for the EEG channels for all samples.
		Label	Label for the dipole.
	Dipole Index	Numeric index of the dipole starting from 1.	
		Trial index	Index of trial used for the fit. Numbering starts from 1.
		Sample	The sample number over which the fit is performed. Separate fit results will be calculated per dipole for each sample in the fit.
		Latency (s)	Position of the sample specified in seconds, relative to the trial synchronization (time zero).
		Moment	The dipole's moment in nanoAmpere-meters = 1.0 * 10^{-9} Ampere-meters for the specified sample.

Class	Tag	Line Elements	Description
		Fit error	Total weighted fit error for the sample. The fit error is either a percentage if a least-squares fit is performed, or the reduced chi squared value if a chi squared fit was performed.
		MEG fit error	Fit error for the sample computed for the MEG channels.
		EEG fit error	Fit error for the sample computed for the EEG channels.
		Label	Label for the dipole/sample unit.

Sample Dipole File — No Fit Results (v6.0)

This example shows a dipole file that contains no fit results.

```
// ************************************
//   CTF Head Model:Dipole Parameter File
// ************************************

// Codes:
// dipole colour:
//   YELLOW = 0, GREEN = 1, RED = 2, CYAN = 3,
//   CYAN = 3, MAGENTA = 4, WHITE = 5, BLACK = 6
// dipole shape:
//   FILLED_CIRCLE = 0, FILLED_SQUARE = 1, FILLED_TRIANGLE = 2,
//   HOLLOW_CIRCLE = 3, HOLLOW_SQUARE = 4, HOLLOW_TRIANGLE = 5
// display flags:
//   true = 1, false = 0
// position/moment/orientation constraint flags:
//   FREE = 0,  FIXED = 1, RANGED = 2,  RADIAL = 3, TANGENTIAL = 4
//   CO-LOCATED = 5, MIRROR SYMMETRIC = 6, ORTHOGONAL_TO_DIPOLE = 7

File_Info
{
    VERSION:                CTF_HEAD_MODEL_FILE_VERSION_6.0
    DATE:                   28-Nov-2006 14:06
    PATIENT:                Anonymous
    STATUS:
}

Model
{
    MODEL_TYPE      :       MEG_ONLY
}

MEG_Sphere
{
    ORIGIN_X:               0.527
    ORIGIN_Y:               0.001
    ORIGIN_Z:               5.318
    RADIUS:                 8.834

}
```

```
Voxel_Resolution
{
     // Resolution in mm per voxel
     SAGITTAL:                    0.93750000
     CORONAL:                     0.93750000
     AXIAL:                       0.93750000

}

MRI_Fid_Points
{
     // Fid. Pts            Sag          Cor          Axi
     NASION:                128          27           159
     LEFT_EAR:              44           124          201
     RIGHT_EAR:             209          131          201

}


Dipoles
{
         // Dipole parameters ...
         // xp (cm)    yp (cm)      zp (cm)      xo        yo        zo        Mom(nAm)   Label
1:       0.803         4.708        10.120       0.737     0.260     0.624     -32.464
}

Dipole_Flags
{
         // Colour     Shape   show dir.     show label      show err
1:       0             0       1             0               0
}


Dipole_Constraints
{
    // orient.        pos.  moment  minRad (cm)  maxRad (cm)  minMom (nAm) maxMom (nAm)  pairIndex
1:  0                 0     0       0.100        50.000       1.000        100.000       2
}

Dipole_FitInfo
{
  //  Trial    Start  Latency (s)  N_Pts  Error    MEG_Error EEG_Error ErrType FitType Good Dataset
1:    1        1      0.0000       1      0.0000   0.0000    0.0000    0       0       1
}
```

Moving Dipole Fit Sample File 1 (v6.0)

This example shows a moving dipole fit over a single sample (time point).

```
// ************************************
//   CTF Head Model:Dipole Parameter File
// ************************************

// Codes:
// dipole colour:
//   YELLOW = 0, GREEN = 1, RED = 2, CYAN = 3,
//   CYAN = 3, MAGENTA = 4, WHITE = 5, BLACK = 6
// dipole shape:
//   FILLED_CIRCLE = 0, FILLED_SQUARE = 1, FILLED_TRIANGLE = 2,
//   HOLLOW_CIRCLE = 3, HOLLOW_SQUARE = 4, HOLLOW_TRIANGLE = 5
// display flags:
//   true = 1, false = 0
// position/moment/orientation constraint flags:
//   FREE = 0,  FIXED = 1, RANGED = 2,  RADIAL = 3, TANGENTIAL = 4
//   CO-LOCATED = 5, MIRROR SYMMETRIC = 6, ORTHOGONAL_TO_DIPOLE = 7

File_Info
{
    VERSION:                CTF_HEAD_MODEL_FILE_VERSION_6.0
    DATE:                   28-Nov-2006 14:06
    PATIENT:                Anonymous
    STATUS:
}

Model
{
    MODEL_TYPE      :       MEG_ONLY
    MRI_FILE:               /home/meg/data/Anonymous.proc/MRI/Anonymous.mri
}

MEG_Sphere
{
    ORIGIN_X:               0.527
    ORIGIN_Y:               0.001
    ORIGIN_Z:               5.318
    RADIUS:                 8.834

}
```

```
Voxel_Resolution
{
    // Resolution in mm per voxel
    SAGITTAL:               0.93750000
    CORONAL:                0.93750000
    AXIAL:                  0.93750000

}

MRI_Fid_Points
{
    // Fid. Pts         Sag         Cor         Axi
    NASION:             128         27          159
    LEFT_EAR:           44          124         201
    RIGHT_EAR:          209         131         201

}

MultiSphere_Data
{
    SEARCH_RADIUS:      4.500
    HEADSHAPE_FILE:     anonymous_hull.shape
    SURFACE_TYPE:       CORTICAL_HULL

    // 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: 7.022 7.121  -22.838 -4.871  5.809  -25.215  6.132  -4.764 -25.844  FALSE

    // Multiple Sphere locations in cm
    //                  X       Y       Z       Radius
    BG1:                -0.461  0.128   6.730   7.814
    BG2:                -0.461  0.128   6.730   7.814
    BP1:                -0.611  -0.066  6.796   7.748

...
    MZO02:                  -0.674  0.237   5.788   8.179
    MZO03:                  -1.087  0.207   5.580   7.847
    MZP01:                  -0.621  0.141   6.798   7.740

}

Dipoles
{
     // Dipole parameters ...
     // xp (cm)   yp (cm)    zp (cm)    xo      yo      zo      Mom(nAm)   Label
1:   0.803       4.708      10.120     0.737   0.260   0.624   -32.464
}
```

```
Dipole_Flags
{
        // Colour        Shape     show dir.       show label           show err
1:      0                0         1               0                    0
}


Dipole_Constraints
{
    // orient.          pos.  moment   minRad (cm)   maxRad (cm)   minMom (nAm) maxMom (nAm)  pairIndex
1:  0                   0     0        0.100         50.000        1.000        100.000       2
}

Dipole_FitInfo
{
  //  Trial     Start   Latency (s)   N_Pts   Error      MEG_Error  EEG_Error  ErrType  FitType  Good   Dataset
1:    1         65      0.1133        1       8.4711     8.4711     0.0000     0        1        1      /home/meg/
data/Anonymous.proc/DATASETS/Anonymous-av.ds
}

Dipole_HeadMotion
{
        // MaxMotion(cm) (negative means unknown)
        1:   -100
        2:   -100
}

Fit
{
        // Fit type and time window...
        FIT_TYPE:                    MOVING_DIPOLE
        WIN_START:                   65
        WIN_POINTS:                  1
        DATA_FILE:                   /home/meg/data/Anonymous2.proc/DATASETS/Anonymous-av.ds
        VERSION:                     4
        ERROR_TYPE:                  0  = Normalized Least-Squares
        MEG_WEIGHT:
        INTEGRATION_ORDER:           2
        USE_PRECEDING_FIT:           FALSE
        FLIP_NEGATIVE_DIPOLES:       FALSE
        USE_CONTINUOUS_HEAD_POS: FALSE
        TOT_ERROR:                   8.4711

        // Starting parameters for fit:
        // xp (cm)      yp (cm)      zp (cm)      xo        yo        zo            Mom(nAm)
1:      -1.900          3.200        6.900        0.889     0.362     0.279         10.000
}
```

```
Fit_Results
{
   // Dipole positions, error volumes and moments for moving dipole solution: (time window 0.113 sec to 0.117 sec)
   // xp yp zp          = dipole positions (cm)
   // xo yo zo          = dipole orientations
   // ax ay az          = ellipsoid major axis vector (cm), semi-axis length = |a|
   // bx by bz          = ellipsoid minor axis vector (cm), semi-axis length = |b|
   // cx cy cz          = ellipsoid intermediate axis vector (cm), semi-axis length = |c|
   // ox oy oz          = ellipsoid origin (centroid) (cm)
   // conf(%)           = confidence level
   // Err(%)            = weighted error
   // MEG Err(%)        = MEG unweighted error
   // EEG Err(%)        = EEG unweighted error

   //  Trial      Sample     Latency (s)     xp          yp          zp
1:    1          65        0.1133          0.8034      4.7084      10.1202

      xo         yo         zo             Mom(nAm)    ax          ay          az
      0.7366     0.2597     0.6244         -32.46      0.0000      0.0000      0.0000

      bx         by         bz             cx          cy          cz
      0.0000     0.0000     0.0000         0.0000      0.0000      0.0000

      ox         oy         oz             conf(%)     Err(%)      MEG Err(%)
      0.0000     0.0000     0.0000         0.0         8.4711      8.4711

      EEG Err(%             Label
      0.0000                Dip1

}
```

Moving Dipole Fit Sample File 2 (v6.0)

This example shows a moving dipole fit over a specified time window.

```
// *************************************
//    CTF Head Model:Dipole Parameter File
// *************************************
// Codes:
// dipole colour:
//    YELLOW = 0, GREEN = 1, RED = 2, CYAN = 3,
//    CYAN = 3, MAGENTA = 4, WHITE = 5, BLACK = 6
// dipole shape:
//    FILLED_CIRCLE = 0, FILLED_SQUARE = 1, FILLED_TRIANGLE = 2,
//    HOLLOW_CIRCLE = 3, HOLLOW_SQUARE = 4, HOLLOW_TRIANGLE = 5
// display flags:
//    true = 1, false = 0
// position/moment/orientation constraint flags:
//    FREE = 0,  FIXED = 1, RANGED = 2,  RADIAL = 3, TANGENTIAL = 4
//    CO-LOCATED = 5, MIRROR SYMMETRIC = 6, ORTHOGONAL_TO_DIPOLE = 7

File_Info
{
    VERSION:                CTF_HEAD_MODEL_FILE_VERSION_6.0
    DATE:                   28-Nov-2006 14:06
    PATIENT                 none
    STATUS:
}

Model
{
    MODEL_TYPE      :           MEG_ONLY
    MRI_FILE:                   home/meg/data/test.proc/MRI/test.mri
}

MEG_Sphere
{
    ORIGIN_X:                   0.000
    ORIGIN_Y:                   0.000
    ORIGIN_Z        :           0.000
    RADIUS:                     7.500

}
```

```
MEG_Fid_Points
{
        //              xp (cm)         yp (cm)         zp (cm)     Head Coil coordinates relative to dewar
        Nasion:         3.694           3.129           -23.921
        LeftEar:        -6.690          4.419           -22.187
        RightEar:       2.999           -7.433          -21.803
        Nominal:        FALSE
}

Dipoles
{
        // Dipole parameters ...
        // xp (cm)      yp (cm)         zp (cm)    xo        yo        zo              Mom(nAm)Label
1:      -0.037          -3.617          3.120      -0.907    -0.270    -0.324          302.535
}

Dipole_Flags
{
    // Colour           Shape           show dir.       show label      show err
1:  0                   0               1               1               0
}

Dipole_Constraints
{
  // orient.    pos.    moment      minRad (cm)     maxRad (cm)     minMom(nAm) maxMom (nAm) pairIndex
1:    0         0       0           0.100           50.000          1.000       100.000        2
}


Dipole_FitInfo
{
  //  Trial     Start   Latency (s)  N_Pts  Error    MEG_Error  EEG_Error  ErrType  FitType  Good  Dataset
1:    1         70      0.1150       5      2.5588   2.5588     0.0000     0        1        1     /home/meg/
data/dsim2/sphere.ds
}

Dipole_HeadMotion
{
        // MaxMotion(cm) (negative means unknown)
        1:    0
        2:    0
}
```

```
Fit
{
    // Fit type and time window...
    FIT_TYPE:                       MOVING_DIPOLE
    WIN_START:                      70
    WIN_POINTS:                     5
    DATA_FILE:                      /home/meg/data/dsim2/sphere.ds
    VERSION:                        4
    ERROR_TYPE:                     0  = Normalized Least-Squares
    MEG_WEIGHT:                     1
    INTEGRATION_ORDER:              2
    USE_PRECEDING_FIT:              TRUE
    FLIP_NEGATIVE_DIPOLES:          TRUE
    USE_CONTINUOUS_HEAD_POS:        TRUE
    TOT_ERROR:                      2.0999

  // Starting parameters for fit:
  //  xp (cm)       yp (cm)      zp (cm)     xo         yo         zoMom(nAm)
1:   -0.037         -3.617       3.120       -0.907     -0.270     -0.324302.535
}

Fit_Results
{
    // Dipole positions, error volumes and moments for moving dipole solution: (time window 0.115 s to 0.123 s)
    // xp yp zp         = dipole positions (cm)
    // xo yo zo         = dipole orientations
    // ax ay az         = ellipsoid major axis vector (cm), semi-axis length = |a|
    // bx by bz         = ellipsoid minor axis vector (cm), semi-axis length = |b|
    // cx cy cz         = ellipsoid intermediate axis vector (cm), semi-axis length = |c|
    // ox oy oz         = ellipsoid origin (centroid) (cm)
    // conf(%)          = confidence level
    // Err(%)           = weighted error
    // MEG Err(%)       = MEG unweighted error
    // EEG Err(%)       = EEG unweighted error

  //  Trial     Sample    Latency (s)    xp         yp         zp
1: 1          70        0.1150         -0.0365    -3.6167    3.1196
1: 1           71       0.1167         -0.0409    -3.6194    3.1152
1: 1           72       0.1183         -0.0469    -3.6188    3.1085
1: 1           73       0.1200         -0.0535    -3.6156    3.1004
1: 1           74       0.1217         -0.0614    -3.6101    3.0911

  xo          yo        zo             Mom(nAm    ax         ay         az
  -0.9068     -0.2700   -0.3236        302.53     0.0000     0.0000     0.0000
  -0.9074     -0.2682   -0.3235        311.12     0.0000     0.0000     0.0000
  -0.9080     -0.2661   -0.3235        318.54     0.0000     0.0000     0.0000
  -0.9086     -0.2640   -0.3236        324.70     0.0000     0.0000     0.0000
  -0.9092     -0.2618   -0.3238        329.57     0.0000     0.0000     0.0000
```

bx	by	bz	cx	cy	cz
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

ox	oy	oz	conf(%)	Err(%)	MEG Err(%)
0.0000	0.0000	0.0000	0.0	2.5588	2.5588
0.0000	0.0000	0.0000	0.0	2.4229	2.4229
0.0000	0.0000	0.0000	0.0	2.3013	2.3013
0.0000	0.0000	0.0000	0.0	2.1932	2.1932
0.0000	0.0000	0.0000	0.0	2.0999	2.0999

EEG Err(%)	Label
0.0000	Dip1.1
0.0000	Dip1.2
0.0000	Dip1.3
0.0000	Dip1.4
0.000	Dip1.5

}

Spatio-temporal Fit Sample File (v6.0)

This example shows a spatio-temporal fit over a specified time window.

```
// **************************************
//   CTF Head Model:Dipole Parameter File
// **************************************

// Codes:
// dipole colour:
//   YELLOW = 0, GREEN = 1, RED = 2, CYAN = 3,
//    CYAN = 3, MAGENTA = 4, WHITE = 5, BLACK = 6
// dipole shape:
//   FILLED_CIRCLE = 0, FILLED_SQUARE = 1, FILLED_TRIANGLE = 2,
//   HOLLOW_CIRCLE = 3, HOLLOW_SQUARE = 4, HOLLOW_TRIANGLE = 5
// display flags:
//   true = 1, false = 0
// position/moment/orientation constraint flags:
//   FREE = 0,  FIXED = 1, RANGED = 2,  RADIAL = 3, TANGENTIAL = 4
//   CO-LOCATED = 5, MIRROR SYMMETRIC = 6, ORTHOGONAL_TO_DIPOLE = 7
```

```
File_Info
{
    VERSION:                CTF_HEAD_MODEL_FILE_VERSION_6.0
    DATE:                   28-Nov-2006 14:06
    PATIENT                 none
    STATUS:
}

Model
{
    MODEL_TYPE     :            MEG_ONLY
    MRI_FILE:                   home/meg/data/test.proc/MRI/test.mri
}

MEG_Sphere
{
    ORIGIN_X:                   0.000
    ORIGIN_Y:                   0.000
    ORIGIN_Z       :            0.000
    RADIUS:                     7.500

}

MEG_Fid_Points
{
      //            xp (cm)     yp (cm)     zp (cm)    Head Coil coordinates relative to dewar
      Nasion:       3.694       3.129       -23.921
      LeftEar:      -6.690      4.419       -22.187
      RightEar:     2.999       -7.433      -21.803
      Nominal:      FALSE
}

Dipoles
{
      // Dipole parameters ...
      // xp (cm)    yp (cm)  zp (cm)  xo      yo      zo          Mom(nAm)Label
1:    -0.048        -3.616   3.106    -0.908  -0.266  -0.324      303.264
}



Dipole_Flags
{
   // Colour      Shape          show dir.      show label      show err
1:      0         0              1              1               0
}
```

```
Dipole_Constraints
{
  // orient.  pos.  moment  minRad (cm)  maxRad (cm)  minMom (nAm)maxMom (nAm) pairIndex
1:   0        0      0       0.100        50.000       1.000        100.000      2
}

Dipole_FitInfo
{
  //  Trial  Start  Latency (s)  N_Pts  Error   MEG_Error  EEG_Error  ErrType  FitType  Good  Dataset
1:1          70     0.1150       5      2.3432  2.3432     0.0000     0        0        1     /home/meg/
data/dsim2/sphere.ds
}

Dipole_HeadMotion
{
      // MaxMotion(cm) (negative means unknown)
      1:   0.4
      2:   0.4
}

Fit
{
    // Fit type and time window...
    FIT_TYPE:                        SPATIO_TEMPORAL
    WIN_START:                       70
    WIN_POINTS:                      5
    DATA_FILE:                       /home/meg/data/dsim2/sphere.ds
    VERSION:                         4
    ERROR_TYPE:                      0  = Normalized Least-Squares
    MEG_WEIGHT:                      1
    INTEGRATION_ORDER:               2
    USE_PRECEDING_FIT:               TRUE
    FLIP_NEGATIVE_DIPOLES:           TRUE
    USE_CONTINUOUS_HEAD_POS:         FALSE
    TOT_ERROR:                       2.3074

  // Starting parameters for fit:
  // xp (cm)    yp (cm)    zp (cm)     xo        yo        zo        Mom(nAm)
1:   -0.048     -3.616     3.106       -0.908    -0.266    -0.324    303.240
}

Fit_Results
{
    // Dipole positions and error volumes for spatiotemporal solution:
    // xp yp zp          = dipole positions (cm)
    // xo yo zo          = dipole orientations
    // ax ay az          = ellipsoid major axis vector (cm), semi-axis length = |a|
    // bx by bz          = ellipsoid minor axis vector (cm), semi-axis length = |b|
```

```
    // cx cy cz          = ellipsoid intermediate axis vector (cm), semi-axis length = |c|
    // ox oy oz          = ellipsoid origin (centroid) (cm)
    // conf(%)           = confidence level
    // Err(%)            = maximum weighted error over range
    // MEG Err(%)     = maximum unweighted MEG error over range
    // EEG Err(%)        = maximum unweighted EEG error over range

  //xp        yp        zp          xo          yo          zo
1:-0.0484     -3.6160   3.1063      -0.9081     -0.2658     -0.3236

  ax          ay        az          bx          by          bz
  0.0000      0.0000    0.0000      0.0000      0.0000      0.0000

  cx          cy        cz          ox          oy          oz
  0.0000      0.0000    0.0000      0.0000      0.0000      0.0000

  conf(%)     Err(%)    MEG Err(%)  EEG Err(%)  Label
  0.0         2.3432    2.3432      0.0000      Dip1


  // Dipole Moments for spatiotemporal solution: (time window 0.115 sec to 0.123 sec)

  //  Trial   Sample    Latency (s)  Mom(nAm)   Err(%)      MEG Err(%)
1:  1         70        0.1150       303.26     2.34        2.34
1:  1         71        0.1167       311.79     2.34        2.34
1:  1         72        0.1183       318.82     2.32        2.32
1:  1         74        0.1217       328.36     2.24        2.24
1:  1         73        0.1200       324.35     2.29        2.29

  EEG Err(%)            Label
  0.00                  Dip1.1
  0.00                  Dip1.2
  0.00                  Dip1.3
  0.00                  Dip1.4
  0.00                  Dip1.5
}
```

SSV File Format

An SSV (***.ssv**) file contains an orthogonal set of signal-space vectors that make up the average of selected data windows (i.e., events) in a dataset. The file is created by the **getSSV** command-line program. Typically it is used in conjunction with **templateDetect** and **ssvDs** to identify and remove spatially stationary artifacts from the data by orthogonal projection in signal-space. The **templateDetect** program is used to identify artifacts, such as heartbeat signals. The **getSSV** program extracts the signal-space vectors that describe the averaged artifact, then applies the orthogonal projection to remove the artifacts from the data. For more information about these programs, see the *Command Line Programs Guide* (PN900-0016).

SSV File Structure (Version 1)

Table 8: Version 1 *.ssv File Format

Type	Variable	Bytes	How Many	Description
char	SSVFILEID1 = 'SSV_001' +NULL	8	1	Identifies the file as an SSV file. Field is null-terminated.
unsigned long	coefType	4	1	Describes the gradient order. Can have one of the following values: • **NOGRAD = 0x00000000** (no gradient formation) • **G1BR = 0x47314252** (first-order synthetic gradient) • **G2BR = 0x47324252** (second-order synthetic gradient) • **G3BR = 0x47334252** (third-order synthetic gradient) • **G0AR = 0x47304152** (adaptive balancing only) • **G1AR = 0x47314152** (first-order synthetic gradient with adaptive balancing) • **G2AR = 0x47324152** (second-order synthetic gradient with adaptive balancing) • **G3AR = 0x47334152** (third-order synthetic gradient with adaptive balancing)
long	numChannels	4	1	Number of channels.
char	name	32	nc[a]	Array of channel names. Each string is null-terminated.
long	chIdxs	4	nc	Array of channel indices.
long	numVectors	4	1	Number of signal-space vectors.
double	data	8	nc*nv[b]	Signal-space vector matrices.

a. nc = **numChannels**
b. nv = **numVectors**

```
// SSV files are structured as follows:
// Version 1
char            SSVFILEID1[8] = "SSV_001";     // identifies the file format
unsigned long   coefType                       // describes the gradient order
long            numChannels;                   // 4-byte integer for the number of channels
char            name[numChannels][32];         // array of strings containing channels names; each
                                               // string is 32-characters long and null-terminated
long            chIdxs[numChannels];           // 4-byte integer array of length numChannels for
                                               // the channel indices.
long            numVectors;                    // 4-byte integer, for the number of signal-space vectors.
double          data[numVectors][numChannels]; // Double float matrix of length numChannels for each
                                               // signal-space vector.


// define coefType gradient order:
#define         NOGRAD      0x00000000         // No gradient formation
#define         G1BR        0x47314252         // First-order synthetic gradient
#define         G2BR        0x47324252         // Second-order synthetic gradient
#define         G3BR        0x47334252         // Third-order synthetic gradient
#define         G0AR        0x47304152         // Adaptive balancing only
#define         G1AR        0x47314152         // 1st-order synthetic gradient with adaptive balancing
#define         G2AR        0x47324152         // 2nd-order synthetic gradient with adaptive balancing
#define         G3AR        0x47334152         // 3rd-order synthetic gradient with adaptive balancing
```

PMAT File Format

A PMAT (***.pmat**) file is created by the **ssvDs** command-line program. It is used by **dfit** (and the DipoleFit GUI application) to perform a signal-space projection on the data as part of the data modeling process.

PMAT File Structure (Version 1)

Table 9: Version 1 *.pmat File Format

Type	Variable	Bytes	How Many	Description
char	PMATFILEID1 = 'PMAT_01' +NULL	8	1	Identifies the file as an PMAT file. Field is null-terminated.

Table 9: Version 1 *.pmat File Format

Type	Variable	Bytes	How Many	Description
unsigned long	coefType	4	1	Describes the gradient order. Can have one of the following values: • **NOGRAD = 0x00000000** (no gradient formation) • **G1BR = 0x47314252** (first-order synthetic gradient) • **G2BR = 0x47324252** (second-order synthetic gradient) • **G3BR = 0x47334252** (third-order synthetic gradient) • **G0AR = 0x47304152** (adaptive balancing only) • **G1AR = 0x47314152** (first-order synthetic gradient with adaptive balancing) • **G2AR = 0x47324152** (second-order synthetic gradient with adaptive balancing) • **G3AR = 0x47334152** (third-order synthetic gradient with adaptive balancing)
long	numChannels	4	1	Number of channels.
long	chIdxs	4	nc[a]	Array of channel indices.
double	data	8	nc*nc	Projection matrix containing the data.

a. nc = **numChannels**

```
// SSV files are structured as follows:
// Version 1
char            PMATFILEID1[8] = "PMAT_01";      // identifies the file format
unsigned long   coefType                         // describes the gradient order
long            numChannels;                     // 4-byte integer for the number of channels
long            chIdxs[numChannels];             // 4-byte integer array of length numChannels for
                                                 // the channel indices
double          data[numChannels][numChannels];  // Projection matrix containing the data
```

```
// define coefType gradient order:
#define    NOGRAD    0x00000000    // No gradient formation
#define    G1BR      0x47314252    // First-order synthetic gradient
#define    G2BR      0x47324252    // Second-order synthetic gradient
#define    G3BR      0x47334252    // Third-order synthetic gradient
#define    G0AR      0x47304152    // Adaptive balancing only
#define    G1AR      0x47314152    // 1st-order synthetic gradient with adaptive balancing
#define    G2AR      0x47324152    // 2nd-order synthetic gradient with adaptive balancing
#define    G3AR      0x47334152    // 3rd-order synthetic gradient with adaptive balancing
```

SAM Time Window File Format

The SAM time window parameter file is a tab-delimited text file containing specifications for active-and control-state time windows, relative to named markers or to the trial sync (time zero). It is used by **SAMcov** and **SAMJcov** when the **-m** parameter is specified. For more information, see the *SAM-suite Guide* (PN900-0037).

Sample Time Window Parameter File

```
3                                   #number of active-state time windows
Tr1        -0.955    -0.250         #1st marker name, active start, end (s)
_time_     -0.750    0.385          #trial sync, active start, end (s)
Vox        0.250     0.875          #2nd marker name, active start, end (s)
2                                   #number of control-state time windows
TR2        0.600     1.250          #1st marker name, active start, end (s)
_time_     0.245     1.875          #trial sync, active start, end (s)
```

Note: The comment field shown above is present only for explanatory purposes and is not permitted in the file.

In this example, Line 1 indicates three active-state windows (with their parameters on the next three lines).

Line 2 shows the first window definition, which specifies that the active-state covariance is to be integrated over the time -0.955 to -0.250 seconds, relative to the marker “Tr1”. (Time is always specified in seconds.) If the same marker name appears more than once within a dataset, then the MEG data relative to each instance of this marker will be combined in the integration for that state. The same marker name can appear in both active and control states.

Line 3 shows the second active-state window defined by a time range relative to the trial sync, or time zero.

Note: The **_time_** marker name is reserved for the trial sync at the beginning of the trial.

Line 5 indicates that there are two control-state time windows, with their parameters on the following lines.

SAM SVL File Format

SAM static image (***.svl**) files are generated by the **SAMsrc**, **LINsrc**, **SAMJsrc**, and **LINJsrc** command-line programs. SAM ***.svl** files computed by **SAMsrc** and **LINsrc** contain pseudo-statistics of sources using the covariance files generated by **SAMcov**. SAM ***.svl** files computed by **SAMJsrc** and **LINJsrc** contain exact statistics of sources, using covariance files computed by **SAMJcov**. For more information about these programs, see the *SAMsuite Guide* (PN900-0037).

SAM static image files can be loaded into MRIViewer and overlaid on individual MRI slices in the three MRI views where they show the image intensity in each slice. These intensities can then be analyzed for peaks and marked on the SAM display. For more information about displaying SAM ***.svl** files in MRIViewer, see the *MRIViewer Guide* (PN900-0015).

SAM *.svl File Structure (Version 2)

Table 10: Version 2 SAM *.svl File Format

Type	Variable	Bytes	How Many	Description
char	Identity = 'SAMIMAGE'	1	8	Identifies the file as a SAM image file.
SAM_HDR_v2	SAMHeader	768	1	SAM image header (version 2).
double	Voxel	8	V[a]	Number of SAM voxels (units = A-m, (A-m)^2, Z, T, F, or P).

a. V = **(XEnd – XStart / StepSize) * (YEnd – YStart / StepSize) * (ZEnd – ZStart / StepSize)** with each term rounded up to the closest larger integer. See the SAM image header (below) for a description of these variables.

Table 11: Version 2 SAM Image Header

Type	Variable	Bytes	How Many	Description
int	Version	4	1	File version number.
char	SetName	256	1	Name of parent dataset.
int	NumChans	4	1	Number of channels used by SAM.
int	NumWeights	4	1	Number of SAM virtual channels. This field has a value of zero for SAM static image (***.svl**) files.
int	pad_bytes1	4	1	Used to align the next double on an eight-byte boundary.
double	XStart	8	1	XStart coordinate (m). **Note**: If a point falls on a start or end boundary, it is included in the voxel. This is true for all X, Y, Z start and end positions.
double	XEnd	8	1	XEnd coordinate (m). (See note for XStart.)
double	YStart	8	1	YStart coordinate (m). (See note for XStart.)

Table 11: Version 2 SAM Image Header

Type	Variable	Bytes	How Many	Description
double	YEnd	8	1	YEnd coordinate (m). (See note for XStart.)
double	ZStart	8	1	ZStart coordinate (m). (See note for XStart.)
double	ZEnd	8	1	ZEnd coordinate (m). (See note for XStart.)
double	StepSize	8	1	Voxel step size (m). (See note for XStart.)
double	HPFreq	8	1	High-pass frequency (Hz).
double	LPFreq	8	1	Low-pass frequency (Hz).
double	BWFreq	8	1	Bandwidth of filters (Hz).
double	MeanNoise	8	1	Mean primary sensor noise (T).
char	MriName	256	1	MRI image file name.
int	Nasion	4	3	MRI voxel index for nasion.
int	RightPA	4	3	MRI voxel index for right pre-auricular.
int	LeftPA	4	3	MRI voxel index for left pre-auricular.
int	SAMType	4	1	SAM file type. This field can have the following values: • **SAM_TYPE_IMAGE = 0** (SAM static image file) • **SAM_TYPE_WT_ARRAY = 1** (SAM coefficients file for regular target array) • **SAM_TYPE_WT_LIST = 2** (SAM coefficients file for target list)

Table 11: Version 2 SAM Image Header

Type	Variable	Bytes	How Many	Description
int	SAMUnit	4	1	SAM units. This field can have the following values: • **SAM_UNIT_COEFF = 0** (SAM coefficients A-m/T) • **SAM_UNIT_MOMENT = 1** (SAM source (or noise) strength A-m) • **SAM_UNIT_POWER = 2** (SAM source (or noise) power (A-m)^2) • **SAM_UNIT_SPMZ = 3** (SAM z-deviate) • **SAM_UNIT_SPMF = 4** (SAM F-statistic) • **SAM_UNIT_SPMT = 5** (SAM T-statistic) • **SAM_UNIT_SPMP = 6** (SAM probability) • **SAM_UNIT_MUSIC = 7** (MUSIC metric) • **SAM_UNIT_G2 = 8** (SAM kurtosis) • **SAM_UNIT_NORM = 9** (SAM normalized signal-to-noise ratio)
int	pad_bytes2	4	1	Used to align the next double on an eight-byte boundary.
double	MegNasion	8	3	MEG dewar coordinates for nasion (m).
double	MegRightPA	8	3	MEG dewar coordinates for right pre-auricular.
double	MegLeftPA	8	3	MEG dewar coordinates for left pre-auricular.
char	SAMUnitName	32	1	Name for the SAM units.

```
// SAM static image files are structured as follows:
// Version 2
char            Identity[8] = "SAMIMAGE";   uniquely identifies image file
SAM_HDR_v2      SAMHeader;                  SAM image header
double          Voxel[V];                   SAM voxels (units = A-m, (A-m)^2, Z, T, F, or P)
//
//   Coefficients & image voxels are ordered in X,Y,Z sequence, with Z the least
//              significant index (most rapidly changing), Y is next, and then X.
//              Coordinate indices always advance in the positive direction. This implies
//              that Voxel[0] is in the right, posterior, inferior position relative to
//              the region of interest (bounding box of image).


// SAM_HDR_v2 is used for both SAM coefficients (weights) and SAM static images.
// Version 2
typedef struct {
    int         Version;            // file version number
    char        SetName[256];       // name of parent dataset
    int         NumChans;           // number of channels used by SAM
    int         NumWeights;         // number of SAM virtual channels (0=static image)
    int         pad_bytes1;         // ** align next double on 8 byte boundary
    double      XStart;             // x-start coordinate (m)
    double      XEnd;               // x-end coordinate (m)
    double      YStart;             // y-start coordinate (m)
    double      YEnd;               // y-end coordinate (m)
    double      ZStart;             // z-start coordinate (m)
    double      ZEnd;               // z-end coordinate (m)
    double      StepSize;           // voxel step size (m)
    double      HPFreq;             // highpass frequency (Hz)
    double      LPFreq;             // lowpass frequency (Hz)
    double      BWFreq;             // bandwidth of filters (Hz)
    double      MeanNoise;          // mean primary sensor noise (T)
    char        MriName[256];       // MRI image file name
    int         Nasion[3];          // MRI voxel index for nasion
    int         RightPA[3];         // MRI voxel index for right pre-auricular
    int         LeftPA[3];          // MRI voxel index for left pre-auricular
    int         SAMType;            // SAM file type
    int         SAMUnit;            // SAM units
    int         pad_bytes2;         // ** align end of structure on 8 byte boundary
    double      MegNasion[3];       // MEG dewar coordinates for nasion (m)
    double      MegRightPA[3];      // MEG dewar coordinates for right pre-auricular (m)
    double      MegLeftPA[3];       // MEG dewar coordinates for left pre-auricular (m)
    char        SAMUnitName[32];    // SAM unit name
} SAM_HDR_v2;
```

```
// define SAM file types
#define SAM_TYPE_IMAGE          0       //SAM static image file
#define SAM_TYPE_WT_ARRAY       1       //SAM coefficients for regular target array
#define SAM_TYPE_WT_LIST        2       //SAM coefficients for target list


// define SAM unit types
enum {SAM_UNIT_COEFF = 0,               // SAM coefficients A-m/T
    SAM_UNIT_MOMENT,                    // SAM source (or noise) strength A-m
    SAM_UNIT_POWER,                     // SAM source (or noise) power (A-m)^2
    SAM_UNIT_SPMZ,                      // SAM z-deviate
    SAM_UNIT_SPMF,                      // SAM F-statistic
    SAM_UNIT_SPMT,                      // SAM T-statistic
    SAM_UNIT_SPMP,                      // SAM probability
    SAM_UNIT_MUSIC,                     // MUSIC metric
    SAM_UNIT_G2,                        // SAM kurtosis
    SAM_UNIT_NORM };                    // SAM normalized (signal-to-noise ratio)
```

SAM *.svl File Structure (Version 1)

Table 12: Version 1 SAM *.svl File Format

Type	Variable	Bytes	How Many	Description
char	Identity = ‘SAMIMAGE’	1	8	Identifies the file as a SAM image file.
SAM_HDR_v1	SAMHeader	664	1	SAM image header (version 1).
double	Voxel	8	V[a]	Number of SAM voxels (units = A-m, (A-m)^2, Z, T, F, or P).

a. V = **(XEnd – XStart / StepSize) * (YEnd – YStart / StepSize) * (ZEnd – ZStart / StepSize)** with each term rounded up to the closest larger integer. See the SAM image header (below) for a description of these variables.

Table 13: Version 1 SAM Image Header

Type	Variable	Bytes	How Many	Description
int	Version	4	1	File version number.
char	SetName	256	1	Name of parent dataset.
int	NumChans	4	1	Number of channels used by SAM.
int	NumWeights	4	1	Number of SAM virtual channels. This field has a value of zero for SAM static image (***.svl**) files.
int	pad_bytes1	4	1	Used to align the next double on an eight-byte boundary.
double	XStart	8	1	XStart coordinate (m). **Note**: If a point falls on a start or end boundary, it is included in the voxel. This is true for all X, Y, Z start and end positions.
double	XEnd	8	1	XEnd coordinate (m). (See note for XStart.)
double	YStart	8	1	YStart coordinate (m). (See note for XStart.)

Table 13: Version 1 SAM Image Header

Type	Variable	Bytes	How Many	Description
double	YEnd	8	1	YEnd coordinate (m). (See note for XStart.)
double	ZStart	8	1	ZStart coordinate (m). (See note for XStart.)
double	ZEnd	8	1	ZEnd coordinate (m). (See note for XStart.)
double	StepSize	8	1	Voxel step size (m). (See note for XStart.)
double	HPFreq	8	1	High-pass frequency (Hz).
double	LPFreq	8	1	Low-pass frequency (Hz).
double	BWFreq	8	1	Bandwidth of filters (Hz).
double	MeanNoise	8	1	Mean primary sensor noise (T).
char	MriName	256	1	MRI image file name.
int	Nasion	4	3	MRI voxel index for nasion.
int	RightPA	4	3	MRI voxel index for right pre-auricular.
int	LeftPA	4	3	MRI voxel index for left pre-auricular.
int	SAMType	4	1	SAM file type. This field can have the following values: • **SAM_TYPE_IMAGE = 0** (SAM static image file) • **SAM_TYPE_WT_ARRAY = 1** (SAM coefficients file for regular target array) • **SAM_TYPE_WT_LIST = 2** (SAM coefficients file for target list)

Table 13: Version 1 SAM Image Header

Type	Variable	Bytes	How Many	Description
int	SAMUnit	4	1	SAM units. This field can have the following values: • **SAM_UNIT_COEFF = 0** (SAM coefficients A-m/T) • **SAM_UNIT_MOMENT = 1** (SAM source (or noise) strength A-m) • **SAM_UNIT_POWER = 2** (SAM source (or noise) power (A-m)^2) • **SAM_UNIT_SPMZ = 3** (SAM z-deviate) • **SAM_UNIT_SPMF = 4** (SAM F-statistic) • **SAM_UNIT_SPMT = 5** (SAM T-statistic) • **SAM_UNIT_SPMP = 6** (SAM probability) • **SAM_UNIT_MUSIC = 7** (MUSIC metric) • **SAM_UNIT_G2 = 8** (SAM kurtosis) • **SAM_UNIT_NORM = 9** (SAM normalized signal-to-noise ratio)
int	pad_bytes2	4	1	Used to align the next double on an eight-byte boundary.

```
// SAM static image files are structured as follows:
// Version 1
char            Identity[8] = "SAMIMAGE";    uniquely identifies image file
SAM_HDR_v1      SAMHeader;                   SAM image header
double          Voxel[V];                    SAM voxels (units = A-m, (A-m)^2, Z, T, F, or P)
//
//   Coefficients & image voxels are ordered in X,Y,Z sequence, with Z the least
//              significant index (most rapidly changing), Y is next, and then X.
//              Coordinate indices always advance in the positive direction. This implies
//              that Voxel[0] is in the right, posterior, inferior position relative to
//              the region of interest (bounding box of image).
```

```
// SAM_HDR_v1 is used for both SAM coefficients (weights) and SAM static images.
/ /Version 1
typedef struct {
    int         Version;          // file version number
    char        SetName[256];     // name of parent dataset
    int         NumChans;         // number of channels used by SAM
    int         NumWeights;       // number of SAM virtual channels (0=static image)
    int         pad_bytes1;       // ** align next double on 8 byte boundary
    double      XStart;           // x-start coordinate (m)
    double      XEnd;             // x-end coordinate (m)
    double      YStart;           // y-start coordinate (m)
    double      YEnd;             // y-end coordinate (m)
    double      ZStart;           // z-start coordinate (m)
    double      ZEnd;             // z-end coordinate (m)
    double      StepSize;         // voxel step size (m)
    double      HPFreq;           // highpass frequency (Hz)
    double      LPFreq;           // lowpass frequency (Hz)
    double      BWFreq;           // bandwidth of filters (Hz)
    double      MeanNoise;        // mean primary sensor noise (T)
    char        MriName[256];     // MRI image file name
    int         Nasion[3];        // MRI voxel index for nasion
    int         RightPA[3];       // MRI voxel index for right pre-auricular
    int         LeftPA[3];        // MRI voxel index for left pre-auricular
    int         SAMType;          // SAM file type
    int         SAMUnit;          // SAM units
    int         pad_bytes2;       // ** align end of structure on 8 byte boundary
} SAM_HDR_v1;


// define SAM file types
#define SAM_TYPE_IMAGE        0        //SAM static image file
#define SAM_TYPE_WT_ARRAY     1        //SAM coefficients for regular target array
#define SAM_TYPE_WT_LIST      2        //SAM coefficients for target list


// define SAM unit types
enum {SAM_UNIT_COEFF = 0,              // SAM coefficients A-m/T
    SAM_UNIT_MOMENT,                   // SAM source (or noise) strength A-m
    SAM_UNIT_POWER,                    // SAM source (or noise) power (A-m)^2
    SAM_UNIT_SPMZ,                     // SAM z-deviate
    SAM_UNIT_SPMF,                     // SAM F-statistic
    SAM_UNIT_SPMT,                     // SAM T-statistic
    SAM_UNIT_SPMP,                     // SAM probability
    SAM_UNIT_MUSIC,                    // MUSIC metric
    SAM_UNIT_G2,                       // SAM kurtosis
    SAM_UNIT_NORM };                   // SAM normalized (signal-to-noise ratio)
```

SAM COV File Format

SAM covariance (***.cov**) files are generated by the **SAMcov** and **SAMJcov** command-line programs. They are used by the **SAMsrc**, **LINsrc**, **SAMJsrc**, and **LINJsrc** programs to compute SAM static image (***.svl**) files. For more information about SAM programs and the files they use, see the *SAMsuite Guide* (PN900-0037).

SAM *.cov File Structure (Version 1)

Table 14: Version 1 SAM *.cov File Format

Type	Variable	Bytes	How Many	Description
char	Identity = 'SAMCOVAR'	1	8	Identifies the file as a SAM covariance file.
COV_HDR	CovHeader	560	1	SAM covariance header.
int	ChannelIndex	4	nc[a]	Index of used primary sensor channels.
double	Cov	8	nc*nc[b]	Matrix of covariance elements.

a. nc = **NumChans**. This variable is defined in the SAM covariance header (see below).
b. **NumChans** * **NumChans**.

Table 15: Version 1 SAM *.cov Header

Type	Variable	Bytes	How Many	Description
int	Version	4	1	File version number.
char	SetName	256	1	Name of parent dataset.
int	NumChans	4	1	Number of channels used by SAM.
char	SpecName	256	1	Name of covariance specification file.
int	NumChans	4	1	Number of channels used by SAM.
double	HPFreq	8	1	High-pass frequency (Hz).
double	LPFreq	8	1	Low-pass frequency (Hz).
double	BWFreq	8	1	Bandwidth of filters (Hz).
int	NumSegments	4	1	Number of time segments.
int	NumSamples	4	1	Number of samples.
int	CovType	4	1	SAM covariance type. This field can have the following values: • **ALL = 0** (single-state covariance) • **ACT = 1** (active-state covariance) • **CTL = 2** (control-state covariance) • **UNI = 1** (single-state covariance matrix) • **DUO = 3** (multi-state covariance matrices)
int	pad_bytes1	4	1	Used to align the next double on an eight-byte boundary.

```
// SAM covariance files are structured as follows:
// Version 1
charIdentity[8] = "SAMCOVAR";                 // uniquely identifies covariance file
COV_HDRCovHeader;                             // SAM covariance header
int         ChannelIndex[M];                  // index of used primary sensor channels
double      Cov[0][0];                        // 1st covariance element
double      Cov[0][1];                        // 2nd covariance element
            "
            "
double      Cov[0][M];
double      Cov[1][0];
double      Cov[1][1];
            "
            "
double      Cov[M][M];                             // last covariance element
```

```
// COV_HDR
// Version 1
typedef struct
{
int         Version;              // file version number
char        SetName[256];         // name of parent dataset
char        SpecName[256];        // name of covariance specification file
int         NumChans;             // number of channels used by SAM
double      HPFreq;               // highpass frequency (Hz)
double      LPFreq;               // lowpass frequency (Hz)
double      BWFreq;               // bandwidth of filters (Hz)
in  t       NumSegments;          // number of time-segments
int         NumSamples;           // total number of samples
in  t       CovType;              // covariance type
int         pad_bytes1;           // ** align end of structure on 8 byte boundary
} COV_HDR;

// covariance type definitions
#define ALL     0                 // single-state covariance
#define ACT     1                 // active-state covariance
#define CTL     2                 // control-state covariance
#define UNI     1                 // single-state covariance matrix
#define DUO     3                 // multi-state covariance matrices
```

SAM WTS File Format

SAM coefficients (***.wts**) files are generated by the **SAMsrc** command-line program when the **-W** parameter is specified (see the *SAMsuite Guide* (PN900-0037) for details). A SAM coefficients file can be loaded into DataEditor to display SAM channel data. For more information, see the *DataEditor Guide* (PN900-0007).

SAM *.wts File Structure (Version 2)

Table 16: Version 2 SAM *.wts File Format

Type	Variable	Bytes	How Many	Description
char	Identity = 'SAMCOEFF'	1	8	Identifies the file as a SAM coefficients file.
SAM_HDR_v2	SAMHeader	768	1	SAM coefficients header (version 2).
char	Channel-Names	32	nc[a]	Names of used primary sensor channels.
char	SAMChannel-Names	32	nw[b]	Names of SAM channels (V1, V2, etc.).
double	Locations	8	nc*3	Locations of SAM channels for each X, Y, Z coordinate.
double	SAMCoeffs	8	nc*nw	SAM coefficient sets (units = A-m/T).

a. nc = **NumChans**. This variable is defined in the SAM coefficients header (see below)
b. nw = **NumWeights**. This variable is defined in the SAM coefficients header (see below).

Table 17: Version 2 SAM *.wts Header

Type	Variable	Bytes	How Many	Description
int	Version	4	1	File version number.
char	SetName	256	1	Name of parent dataset.
int	NumChans	4	1	Number of channels used by SAM.
int	NumWeights	4	1	Number of SAM virtual channels. This field has a non-zero value for SAM coefficients (***.wts**) files.
int	pad_bytes1	4	1	Used to align the next double on an eight-byte boundary.
double	XStart	8	1	XStart coordinate (m). **Note**: If a point falls on a start or end boundary, it is included in the voxel. This is true for all X, Y, Z start and end positions.
double	XEnd	8	1	XEnd coordinate (m). (See note for XStart.)
double	YStart	8	1	YStart coordinate (m). (See note for XStart.)
double	YEnd	8	1	YEnd coordinate (m). (See note for XStart.)
double	ZStart	8	1	ZStart coordinate (m). (See note for XStart.)
double	ZEnd	8	1	ZEnd coordinate (m). (See note for XStart.)
double	StepSize	8	1	Voxel step size (m). (See note for XStart.)
double	HPFreq	8	1	High-pass frequency (Hz).
double	LPFreq	8	1	Low-pass frequency (Hz).
double	BWFreq	8	1	Bandwidth of filters (Hz).
double	MeanNoise	8	1	Mean primary sensor noise (T).
char	MriName	256	1	MRI image file name.

Table 17: Version 2 SAM *.wts Header

Type	Variable	Bytes	How Many	Description
int	Nasion	4	3	MRI voxel index for nasion.
int	RightPA	4	3	MRI voxel index for right pre-auricular.
int	LeftPA	4	3	MRI voxel index for left pre-auricular.
int	SAMType	4	1	SAM file type. This field can have the following values: • **SAM_TYPE_IMAGE = 0** (SAM static image file) • **SAM_TYPE_WT_ARRAY = 1** (SAM coefficients file for regular target array) • **SAM_TYPE_WT_LIST = 2** (SAM coefficients file for target list)
int	SAMUnit	4	1	SAM units. This field can have the following values: • **SAM_UNIT_COEFF = 0** (SAM coefficients A-m/T) • **SAM_UNIT_MOMENT = 1** (SAM source (or noise) strength A-m) • **SAM_UNIT_POWER = 2** (SAM source (or noise) power (A-m)^2) • **SAM_UNIT_SPMZ = 3** (SAM z-deviate) • **SAM_UNIT_SPMF = 4** (SAM F-statistic) • **SAM_UNIT_SPMT = 5** (SAM T-statistic) • **SAM_UNIT_SPMP = 6** (SAM probability) • **SAM_UNIT_MUSIC = 7** (MUSIC metric) • **SAM_UNIT_G2 = 8** (SAM kurtosis) • **SAM_UNIT_NORM = 9** (SAM normalized signal-to-noise ratio)
int	pad_bytes2	4	1	Used to align the next double on an eight-byte boundary.
double	MegNasion	8	3	MEG dewar coordinates for nasion (m).
double	MegRightPA	8	3	MEG dewar coordinates for right pre-auricular.

Table 17: Version 2 SAM *.wts Header

Type	Variable	Bytes	How Many	Description
double	MegLeftPA	8	3	MEG dewar coordinates for left pre-auricular.
char	SAMUnitName	32	1	Name for the SAM units.

```
// SAM coefficient files are structured as follows:
// Version 2
char          Identity[8] = "SAMCOEFF";     // uniquely identifies coefficient file
SAM_HDR_v2    SAMHeader;                    // SAM header
char          ChannelNames[M][32];          // names of used primary sensor channels
char          SAMChannelNames[V][32];       // names of SAM channels (eg. 'V1', 'V2' ...)
double        Locations[V][3];              // locations of SAM channels (head coordinates)
double        SAMCoeffs[V][M];              // SAM coefficient sets (units = A-m/T)
```

```
// SAM_HDR_v2 is used for both SAM coefficients (weights) and SAM static images.
// Version 2
typedef struct {
    int         Version;          // file version number
    char        SetName[256];     // name of parent dataset
    int         NumChans;         // number of channels used by SAM
    int         NumWeights;       // number of SAM virtual channels (0=static image)
    int         pad_bytes1;       // ** align next double on 8 byte boundary
    double      XStart;           // x-start coordinate (m)
    double      XEnd;             // x-end coordinate (m)
    double      YStart;           // y-start coordinate (m)
    double      YEnd;             // y-end coordinate (m)
    double      ZStart;           // z-start coordinate (m)
    double      ZEnd;             // z-end coordinate (m)
    double      StepSize;         // voxel step size (m)
    double      HPFreq;           // highpass frequency (Hz)
    double      LPFreq;           // lowpass frequency (Hz)
    double      BWFreq;           // bandwidth of filters (Hz)
    double      MeanNoise;        // mean primary sensor noise (T)
    char        MriName[256];     // MRI image file name
    int         Nasion[3];        // MRI voxel index for nasion
    int         RightPA[3];       // MRI voxel index for right pre-auricular
    int         LeftPA[3];        // MRI voxel index for left pre-auricular
    int         SAMType;          // SAM file type
```

```
    int             SAMUnit;            // SAM units
    int             pad_bytes2;         // ** align end of structure on 8 byte boundary
    double          MegNasion[3];       // MEG dewar coordinates for nasion (m)
    double          MegRightPA[3];      // MEG dewar coordinates for right pre-auricular (m)
    double          MegLeftPA[3];       // MEG dewar coordinates for left pre-auricular (m)
    char            SAMUnitName[32];    // SAM unit name
} SAM_HDR_v2;
```

```
// define SAM file types
#define SAM_TYPE_IMAGE        0         //SAM static image file
#define SAM_TYPE_WT_ARRAY     1         //SAM coefficients for regular target array
#define SAM_TYPE_WT_LIST      2         //SAM coefficients for target list
```

```
// define SAM unit types
enum {SAM_UNIT_COEFF = 0,               // SAM coefficients A-m/T
    SAM_UNIT_MOMENT,                    // SAM source (or noise) strength A-m
    SAM_UNIT_POWER,                     // SAM source (or noise) power (A-m)^2
    SAM_UNIT_SPMZ,                      // SAM z-deviate
    SAM_UNIT_SPMF,                      // SAM F-statistic
    SAM_UNIT_SPMT,                      // SAM T-statistic
    SAM_UNIT_SPMP,                      // SAM probability
    SAM_UNIT_MUSIC,                     // MUSIC metric
    SAM_UNIT_G2,                        // SAM kurtosis
    SAM_UNIT_NORM };                    // SAM normalized (signal-to-noise ratio)
```

SAM *.wts File Structure (Version 1)

Table 18: Version 1 SAM *.wts File Format

Type	Variable	Bytes	How Many	Description
char	Identity = 'SAMCOEFF'	1	8	Identifies the file as a SAM coefficients file.
SAM_HDR_v1	SAMHeader	664	1	SAM coefficients header (version 2).
int	ChannelIndex	4	nc[a]	Index of used primary sensor channel numbers.
double	SAMCoeffs	8	nc	SAM coefficient sets (units = A-m/T).

a. nc = **NumChans**. This variable is defined in the SAM coefficients header (see below)

Table 19: Version 1 SAM *.wts Header

Type	Variable	Bytes	How Many	Description
int	Version	4	1	File version number.
char	SetName	256	1	Name of parent dataset.
int	NumChans	4	1	Number of channels used by SAM.
int	NumWeights	4	1	Number of SAM virtual channels. This field has a non-zero value for SAM coefficients (***.wts**) files.
int	pad_bytes1	4	1	Used to align the next double on an eight-byte boundary.
double	XStart	8	1	XStart coordinate (m). **Note**: If a point falls on a start or end boundary, it is included in the voxel. This is true for all X, Y, Z start and end positions.
double	XEnd	8	1	XEnd coordinate (m). (See note for XStart.)
double	YStart	8	1	YStart coordinate (m). (See note for XStart.)

Table 19: Version 1 SAM *.wts Header

Type	Variable	Bytes	How Many	Description
double	YEnd	8	1	YEnd coordinate (m). (See note for XStart.)
double	ZStart	8	1	ZStart coordinate (m). (See note for XStart.)
double	ZEnd	8	1	ZEnd coordinate (m). (See note for XStart.)
double	StepSize	8	1	Voxel step size (m). (See note for XStart.)
double	HPFreq	8	1	High-pass frequency (Hz).
double	LPFreq	8	1	Low-pass frequency (Hz).
double	BWFreq	8	1	Bandwidth of filters (Hz).
double	MeanNoise	8	1	Mean primary sensor noise (T).
char	MriName	256	1	MRI image file name.
int	Nasion	4	3	MRI voxel index for nasion.
int	RightPA	4	3	MRI voxel index for right pre-auricular.
int	LeftPA	4	3	MRI voxel index for left pre-auricular.
int	SAMType	4	1	SAM file type. This field can have the following values: • **SAM_TYPE_IMAGE = 0** (SAM static image file) • **SAM_TYPE_WT_ARRAY = 1** (SAM coefficients file for regular target array) • **SAM_TYPE_WT_LIST = 2** (SAM coefficients file for target list)

Table 19: Version 1 SAM *.wts Header

Type	Variable	Bytes	How Many	Description
int	SAMUnit	4	1	SAM units. This field can have the following values: • **SAM_UNIT_COEFF = 0** (SAM coefficients A-m/T) • **SAM_UNIT_MOMENT = 1** (SAM source (or noise) strength A-m) • **SAM_UNIT_POWER = 2** (SAM source (or noise) power (A-m)^2) • **SAM_UNIT_SPMZ = 3** (SAM z-deviate) • **SAM_UNIT_SPMF = 4** (SAM F-statistic) • **SAM_UNIT_SPMT = 5** (SAM T-statistic) • **SAM_UNIT_SPMP = 6** (SAM probability) • **SAM_UNIT_MUSIC = 7** (MUSIC metric) • **SAM_UNIT_G2 = 8** (SAM kurtosis) • **SAM_UNIT_NORM = 9** (SAM normalized signal-to-noise ratio)
int	pad_bytes2	4	1	Used to align the next double on an eight-byte boundary.

```
// SAM coefficient files are structured as follows:
// Version 1
char          Identity[8] = "SAMCOEFF";     // uniquely identifies coefficient file
SAM_HDR_v1    SAMHeader;                    // SAM header
int           ChannelIndex[M];              // index of used primary sensor channel numbers
double        SAMCoeffs[0][M];              // 1st SAM coefficient set (units = A-m/T)
double        SAMCoeffs[1][M];              // 2nd SAM coefficient set
                 "
                 "
double        SAMCoeffs[V][M];              // last SAM coefficient set
```

```
// SAM_HDR_v1 is used for both SAM coefficients (weights) and SAM static images.
/ /Version 1
typedef struct {
    int         Version;          // file version number
    char        SetName[256];     // name of parent dataset
    int         NumChans;         // number of channels used by SAM
    int         NumWeights;       // number of SAM virtual channels (0=static image)
    int         pad_bytes1;       // ** align next double on 8 byte boundary
    double      XStart;           // x-start coordinate (m)
    double      XEnd;             // x-end coordinate (m)
    double      YStart;           // y-start coordinate (m)
    double      YEnd;             // y-end coordinate (m)
    double      ZStart;           // z-start coordinate (m)
    double      ZEnd;             // z-end coordinate (m)
    double      StepSize;         // voxel step size (m)
    double      HPFreq;           // highpass frequency (Hz)
    double      LPFreq;           // lowpass frequency (Hz)
    double      BWFreq;           // bandwidth of filters (Hz)
    double      MeanNoise;        // mean primary sensor noise (T)
    char        MriName[256];     // MRI image file name
    int         Nasion[3];        // MRI voxel index for nasion
    int         RightPA[3];       // MRI voxel index for right pre-auricular
    int         LeftPA[3];        // MRI voxel index for left pre-auricular
    int         SAMType;          // SAM file type
    int         SAMUnit;          // SAM units
    int         pad_bytes2;       // ** align end of structure on 8 byte boundary
} SAM_HDR_v1;


// define SAM file types
#define SAM_TYPE_IMAGE          0       //SAM static image file
#define SAM_TYPE_WT_ARRAY       1       //SAM coefficients for regular target array
#define SAM_TYPE_WT_LIST        2       //SAM coefficients for target list


// define SAM unit types
enum {SAM_UNIT_COEFF = 0,               // SAM coefficients A-m/T
    SAM_UNIT_MOMENT,                    // SAM source (or noise) strength A-m
    SAM_UNIT_POWER,                     // SAM source (or noise) power (A-m)^2
    SAM_UNIT_SPMZ,                      // SAM z-deviate
    SAM_UNIT_SPMF,                      // SAM F-statistic
    SAM_UNIT_SPMT,                      // SAM T-statistic
    SAM_UNIT_SPMP,                      // SAM probability
    SAM_UNIT_MUSIC,                     // MUSIC metric
    SAM_UNIT_G2,                        // SAM kurtosis
    SAM_UNIT_NORM };                    // SAM normalized (signal-to-noise ratio)
```

Appendix A: CTF MEG Head Coordinate System

Introduction to 3D Coordinate Systems

The spatial volume of the head can be represented in a variety of three-dimensional coordinate systems. All forms can be expressed as points in three axes: (x, y, z). The issues are the location of the origin and the orientation of the axes.

At various times, the CTF MEG System may need to work with data from any of the following systems:

MEG system, based on the fiducial points.

MRI system, based on the image slice orientation.

Polhemus “raw” system based on the digitizer’s transmitter location and orientation.

Each have their own distinct coordinate systems. Coregistration is needed for the system to translate locations to a common reference. The common reference used is the CTF MEG head coordinate system.

CTF MEG Head Coordinate System

The CTF MEG System sets the origin and axes orientation on a reference determined by the three head localization coils, shown in Figure 7 (on the next page) as the green (left), red (right), and blue (nasion) dots.

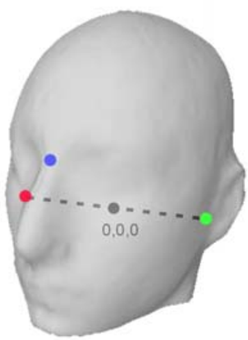


Figure 7: MEG References and Origin

The origin is defined as the midpoint between the left and right preauricular fiducial points. This means the CTF MEG head coordinate system uses both positive (to the subject's left) and negative values (to the subject's right).

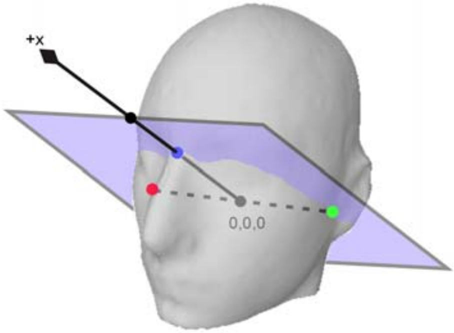


Figure 8: MEG X-Y Plane

The orientation of the axes is determined by setting the x-axis on the line from the origin through the nasion fiducial point. The x-y plane (violet) is defined by the three fiducial points.

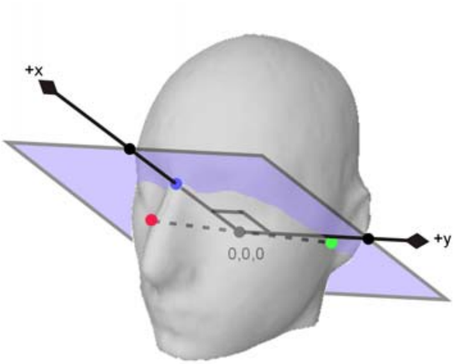


Figure 9: MEG Y-Axis Offset From Fiducial

The y-axis is perpendicular to the x-axis on the x-y plane. Since the human head rarely has perfect symmetry, the y-axis is not likely through a fiducial point, but could be slightly ahead (as in this sample) or behind it.

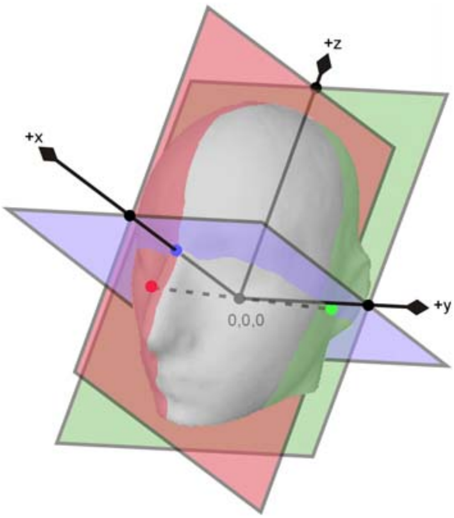


Figure 10: CTF MEG Head Coordinate System

The z-axis is perpendicular to both the x- and y-axes.

Coregistration of Coordinate Systems

Coregistration is the process of defining a common reference so that objects can be mapped from one system to the other. The fiducial points are used as the common reference for all 3D data in the CTF MEG System. When the system knows their location in each coordinate system, it can translate points between the two.

Coregistering with MRI Data

The CTF MEG MRIViewer application is used to mark the fiducial points on the image to coregister with the head localization coil positions stored with the MEG dataset (see *MRIViewer Guide* (PN900-0015) for details). Having the fiducial points indicated in the MRI requires the placement of MRI (radiological) contrast markers on the fiducial points prior to recording the MRI. These markers appear as distinctive bright ovals or rings on the image.

Coregistering with Polhemus Digitized Data

The CTF MEG Electrode Digitizer application requires that you digitize the head coordinates (fiducial points) before collecting electrode location or head shape data (see *Electrode Digitizer Guide* (PN900-0038) for details). This same application performs the necessary coordinate translation to yield data files in the CTF MEG head coordinate system.

Appendix B: CPersist Object

CTF | MEG™

CPersist Object Description

The CTF MEG software uses the CPersist object class for binary files describing parameters that may contain nested objects. This format is illustrated in Figure 11.

4 bytes					15 bytes
WS1_	Tag 1	Tag 2	. . .	Tag n	EndOfParameters

Figure 11: CPersist Object Format

The CPersist object has the following properties:

- As a file, it is a single stream of data.
- Values are stored in Big Endian format.
- The file always starts with the four characters “WS1_”.
- The file always ends with the 15 characters “EndOfParameters”.
- Between these delimiters are the tags described below.

Tag Format in CPersist

Every tag consists of the following elements:

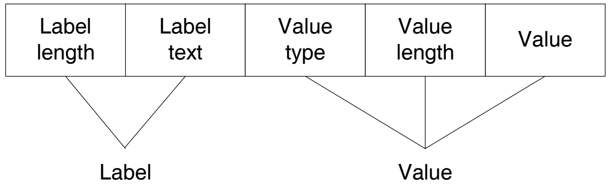


Figure 12: CPersist Tag Format

Table 20: CPersist Tag Format

Field	Type	Description
Label length	32-bit int	Length of the label text in bytes.
Label text	string	Tag name in ASCII
Value type	32-bit int	One of the following: 1 = custom 2 = CPersist; embedded (nested) object 3 = binary 4 = double 5 = integer (int) 6 = short 7 = unsigned short 8 = Boolean (char; 1=TRUE, 0=FALSE) 9 = CStr32 (null-terminated string consisting of 31 meaningful chars) 10 = CString (null-terminated array of chars of arbitrary length) 11 = list of CStrings (the first item in the list is an int specifying the number of items) 12 = list of CStr32s (the first item in the list is an int specifying the number of items) 13 = list of SensorClass objects (each SensorClass is an int; the first item in the list is an int specifying the number of items) 14 = long 15 = unsigned long 16 = unsigned integer (unsigned int) 17 = ctfboolean (int; 1=TRUE, 0=FALSE)

Table 20: CPersist Tag Format (cont.)

Field	Type	Description
Value length (optional)	32-bit int	Length of the value field in bytes. This field is present *only* for following value types: Cstring binary
Value	Value type	The data content of the attribute

If the value type is CPersist, another CPersist object is embedded (nested) in the file. Process the nested object as a separate CPersist class file. Note that the length of this tag value is omitted for this value type.

Appendix C: Higher-order Gradients

Introduction

This document describes how to extract information concerning higher-order gradient formation from data collected using the CTF MEG System and how to apply these data to calculate the forward solution from a simulated dipole.

Higher-order gradiometer formation is a noise-cancellation technique exclusive to the CTF MEG System. It permits the MEG detectors to be sensitive to the weak signals of the brain, yet impervious to the much stronger sources from the environment. This process, which is carried out in real time, allows the system to be run without the expense of magnetic shielding, or in combination with a standard shielded room for enhanced noise reduction.

In order to calculate the forward solution, the same procedure must be performed on any field data calculated from a simulated dipole. The procedure consists of calculating the field at each coil in the sensor as well as each coil in a set of reference channels. The field values at the reference channels are then multiplied by the appropriate weight and subtracted from the MEG sensor channels.

This document covers the formation of the higher-order gradients from either MEG data or simulated dipole data. It is assumed that the user already has access to the data and is familiar with the procedures outlined in this manual and the *Dataset File Library (Technical Note #2)* (PN900-0013).

Reading and Using Coefficients

Reading Coefficients

The coefficients for each sensor channel are read in from the dataset's **.res4** file as described in "RES4 File Format" on page 13. After reading in all of the sensor information, the next field in the **.res4** file is the number of coefficient records. Following this is the indicated number of coefficient records.

Each record consists of the sensor name, coefficient type (1st, 2nd or 3rd, ideal or real) and the list of reference/weight pairs as defined in

```
typedef struct
{
   char           sensorName[ 32 ];
   unsigned long  coefType;
   CoefResRec     coefRec;
} SensorCoefResRec
```

where

```
typedef struct CoefResRec
{
   short          numCoefs;
   char           sensorList[ 50 ][ 31 ];
   double         coefsList[ 50 ];
} CoefResRec
```

The values associated with **coefType** are listed in Table 21.

Table 21: Coefficient Type Values

1st gradient real	0x47314252	'G1BR'
2nd gradient real	0x47324252	'G2BR'
3rd gradient real	0x47334252	'G3BR'
2nd gradient ideal	0x47324f49	'G2OI'
3rd gradient ideal	0x47334f49	'G3OI'

Table 21: Coefficient Type Values

Adaptive only	0x47304152	‘G0AR’
3rd gradient +adaptive	0x47314152	‘G1AR’
2nd gradient +adaptive	0x47324152	‘G2AR’
3rd gradient + adaptive	0x47334152	‘G3AR’

As each coefficient record is read in, it is associated with the corresponding sensor. The **sensorList** field contains the channel names on the references used to form the higher-order gradients, and the **coefList** contains the corresponding list of coefficients. There are **numCoefs** values in each of the **sensorList** and **coefsList** arrays.

Using Coefficients

The coefficients from the dataset's **.res4** file are Phi0 ($\Phi_{0)}$ data, whereas the data are in units of Tesla. The coefficients must therefore be converted to Teslas to be relevant. To do this, use the following formula:

CoefOfRefInTesla = CoefOfRefInPhi0 * gainOfRef / gainOfSensor

where “gainOfRef” and “gainOfSensor” represent the total gain of the reference channels and sensor channels, respectively. The total gain of a channel is computed as follows:

gainOf<*Channel*> = QGain x IOGain x properGain

These gains can be retrieved from the resource file (see “MegDefs.h Header File” on page 26).

Example

If the 3rd gradient coefficients for sensor MLC11 are (for Phi0 data)

```
BG1: cBG1
BG2: cBG2
BG3: cBG3
G11: cG11
...
```

and the gains of the references and sensors are

```
BG1: gBG1
BG2: gBG2
...
MLC11: gMLC11
...
```

Then the coefficients for MLC11 which correspond to Tesla data are

```
BG1: cBG1 * gBG1 / gMLC11
BG2: cBG2 * gBG2 / gMLC11
BG3: cBG3 * gBG3 / gMLC11
G11: cG11 * gG11 / gMLC11
...
```

E.g.,

G3OI coefficient for data in phi0 for MLC12:

```
G11 (G11-1105):   0.143393
G12 (G12-1105):   -0.00166001
G13 (G13-1105):   -0.129106
G22 (G22-1105):   -0.136122
G23 (G23-1105):   0.0653427
P11 (P11-1105):   0.00202835
Q11 (Q11-1105):   -0.00211531
Q13 (Q13-1105):   0.265981
R11 (R11-1105):   -0.127228
R12 (R12-1105):   -0.00188768
R13 (R13-1105):   0.13007
R22 (R22-1105):   0.0777415
R23 (R23-1105):   0.330706
```

G3OI coefficient for data in Tesla for MLC12:

```
G11  (G11-1105):   -0.285225
G12  (G12-1105):   0.00313698
G13  (G13-1105):   -0.241933
G22  (G22-1105):   -0.270226
G23  (G23-1105):   -0.125866
P11  (P11-1105):   0.00393464
Q11  (Q11-1105):   -0.0039391
Q13  (Q13-1105):   0.477763
R11  (R11-1105):   -0.238921
R12  (R12-1105):   -0.00330739
R13  (R13-1105):   -0.234992
R22  (R22-1105):   -0.145796
R23  (R23-1105):   0.602411
```

The coefficients are applied by *subtracting* the linear combination of reference signals from the sensor signal.

For example, if the data are in Teslas

$$M_{j(3rd)} = M_{j(raw)} - \sum_{i=1}^{N} R_i \times cR_{ij} \times gR_i / gM_j$$

If the data is in Φ_0

$$M_{j(3rd)} = M_{j(raw)} - \sum_{i=1}^{N} R_i \times cR_{ij}$$

where

j	Sensor number
M_j	Field at sensor j
i	Index of reference
R_i	Field at reference i
cR_{ij}	Coefficient for reference i, sensor j
gR_i	Gain of reference i
gM_j	Gain of sensor j

Ideal vs. Real Coefficients (G3OI vs. G3BR)

The ideal coefficients are based on the geometry of the sensor configuration. The real coefficients account for geometrical and common mode errors which are determined experimentally.

NOTICE

***For DipoleFit and dfit prior to version 4.12*:**

When processing real data, we used the real coefficients. For forward solution computation (e.g., in DipoleFit and dfit), we used the ideal coefficients since our sensor descriptions are always ideal.

NOTICE

***For DipoleFit and dfit version 4.12 and up*:**

We use the real coefficients for both processing real data and forward solution computation. Recent studies have shown that using the real coefficients for the forward solution computation gives a more accurate match with the real data when compared to using the ideal coefficients.

Simulating Data/Forward Solution Computation

In order to calculate the forward solution, the field/gradients generated by the simulated source must be calculated at each sensor and reference.

Each sensor is described by the following:

- one or more coils
- zero or more baselines
- signed gain

Each coil is described by the following:

- position of center of coil (use the position relative to head coordinate system)
- number of turns (N)
- coil area (A)
- unit normal vector (p) (use the position relative to head coordinate system)
- baseline from previous coil (for coils other than the first)

To compute the "field" picked up by a sensor:

1. Compute and sum the flux picked up by each coil.
2. Convert the flux to "field" by dividing the total flux by the effective area (N * A) of the sensing coil (i.e., the first coil).

3. Our internal polarity definition is opposite of the general convention. Thus in order to have the polarity of the simulated data consistent with the measured data, you have to apply the opposite polarity (of the signed gain) to the simulated data. Therefore, if the sensor's gain is positive, you have to invert the simulated data; if the sensor's gain is negative, leave the simulated data alone.

NOTICE

If you are working with data in 'Tesla', you do not need to use the gain value, just its sign (as per Step 3. above).

To compute the flux picked up by a coil, take the projection of the field vector onto the coil's unit normal vector.

That is,

$$flux_i = N_i \times A_i \times \bar{B} \bullet \hat{p}_i$$

where

i	Coil number
$\bar{B}$	Field vector of your simulated signal source at the center of coil i
N_i	Number of turns of coil i
A_i	Area of coil i
$\hat{p}_i$	Unit vector normal to area of coil i.

The total flux of the sensor is then given by

$$flux_{total} = \sum_{i=1}^{numberofcoils} flux_i$$

and the field is then given by

$$field = -sign(gM) \times flux_{total} / (N_i \times A_i)$$

where

sign(gM) = +1 if gain of sensor is positive
-1 if gain of sensor is negative

If you are integrating over the coil area, the procedure is similar.

Further Reading

The following papers are available at http://www.vsmmedtech.com/:

Synthetic Higher-Order Gradiometers Reduce Environmental Noise, Not the Measured Brain Signals

Baseline Optimization for Noise Cancellation Systems

CTF | MEG™

VSM MedTech Ltd.
MEDICAL ADVANCES THROUGH TECHNOLOGY™

PN900-0088
Revision 1.3, 30 November 2006