Data Set File Format

Omega 2000 Whole-Cortex MEG System

Technical Note #1

CTF Systems Inc.
A SUBISIDIARY OF VSM MEDTECH LTD.™

PN900-0012 (1.0)
Revision Date: September 13, 2001
Confidential and Proprietary

Release v4.10.4

DISCLAIMER

CTF Systems Inc. (A subsidiary of VSM MedTech Ltd.) (“CTF”) has used reasonable effort to include accurate and up-to-date information in this manual; it does not, however, make any warranties or representations as to its accuracy or completeness. CTF 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 CTF be liable for any indirect, direct, special, incidental, or consequential damages arising from your use of this manual.

Features and specifications of CTF’s products are subject to change without notice.

This manual contains information and images about CTF, its magnetoencephalographic systems and its other products that are protected by copyright.

Commercially Confidential Information

CTF Systems Inc.
A SUBSIDIARY OF VSM MEDTECH LTD.™

#15 - 1750 McLean Avenue
Port Coquitlam, BC Canada
V3C 1M9
Ph: (604) 941-8561
Fax: (604) 941-8565
Web: www.ctf.com

Table of Contents

Table of Contents

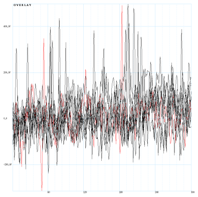
New Features

MEG 4.1 Data Format

Appendix A

Appendix B

New Features



New Features In Release v4.10.4.

- No additional changes have been made to the data format since release version 4.10.3

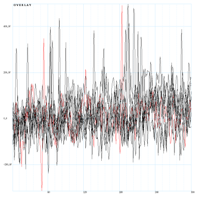
New Features In Release v4.10.3

- Uses New MEG4.1 format. This format change was required due to the increase in number of trigger and DAC channels introduced in the new DSQ-2000 electronics. The new trigger channel has 32 bits. The 8 bit trigger from DSQ-800 Stimulus is in the lower 8 bits.

New Features In Release v4.4.0

- Threshold detection on ADC channels
- The Data Analysis Window

MEG 4.1 Data Format



New to MEG 4.1 Data Format

Version 4.1 of the MEG Data Format introduces a new 8 byte header to both the Data File and Resource File associated with each data set.

All new versions of CTF software can read both 4.0 and 4.1 MEG Data Formats.. However, old versions of CTF software can only read data of the MEG4.0 format.

WARNING!
If you have 3rd party software reading the MEG 4.0 or if you have written your own code to read MEG 4.0 format, the code must be changed to reflect the new format changes.

Introduction

This section describes the format of the MEG/4.1 physical dataset files. The intended readers are programmers who wish to access or create datasets through their own methods. A familiarity with the UNIX file system, the C programming language, and basic machine architecture is assumed

This document only discusses those files that are essential to all physical datasets. Peripheral files, such as marker files, are discussed in other documents.

The Dataset Files

Every dataset has a name associated with it. A physical dataset is represented as a directory with the same name as the dataset and a `.ds` extension. Thus, a dataset named *8-megt* would reside in a directory called `8-megt.ds`. Within this directory, there are two critical files: a data file and a resource file. The data file contains the collected samples of data. The resource file contains everything essential to the data. Following the above example, the data file would be named `8-megt.meg4` and the resource file would be name `8-megt.res4`.

Datasets can also have peripheral files associated with them. For example, marker files are not critical to the basic functionality of datasets, but are often useful. Marker files must end in `.mrk`. The default marker file generated for a dataset is `MarkerFile.mrk`.

Idiosyncracies

The Hewlett Packard 9000/7xx series is the native platform for the MEG/4.1 software, and the dataset files are stored appropriately. There are some concerns that the programmer must be aware of if they are to program the dataset on a non-native platform:

- all multi-byte types are stored with the least significant byte stored in the highest memory byte (i.e. little endian format)
- the types in the `MegDefs.h` header file (see “Appendix A” on page9) must be defined for the new platform. Currently, many fundamental types are defined only when the symbol `__hp9000s800` is defined in the preprocessor. The types reflect their definitions in an obvious manner: for example, `Int32` should be a 4-byte (32-bit) integral type.
- all enumerated types must be 4 bytes in length.
- all structures are padded such that the beginning address of a multi-byte type is a multiple of that type's size. For example, objects of type `Int16` always begin on 2-byte boundaries; objects of type `Int32` always begin on 4-byte boundaries, and objects of type `SDouble` always begin on 8-byte boundaries. Appropriate padding is added by the compiler.

The Data File Format

The data file consists of a header and the raw samples as collected from the electronics. The header is the 8-byte character sequence: `MEG41CP+NULL`. The data is stored as a sequence of (signed) 4-byte integers, starting with the first trial and first channel, then the first trial and second channel, etc, see Table 1. The number of channels per trial and the number of samples in every trial-channel block are constant per dataset. The constants are found in the general resources stored in the resource file, see “The Resource File Format”. The numbers stored in the data file are the raw numbers collected from the electronics. For these numbers to be useful, the various gains, stored in the sensor resources, must be applied.

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

The Resource File Format

Refer to “Appendix A” on page9 for definitions of the types in this section. The following table outlines the format of the resource file. The byte offset is defined with respect to 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)

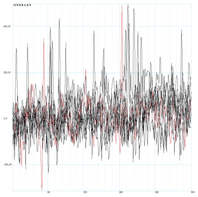
f = nf*18 + np*8

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

Table 2: Resource File Format

Description	**Byte Offset**	**Size (in bytes)**	**How many**	**Type**
header: MEG41RE+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
filter information	1846+rdl	variable	number of filters	
filter frequency	b+fi*18+p	8		SDouble
filter class	b+fi*18+p+8	4		classType
filter type	b+fi*18+p+12	4		filtType
number of parameters	b+fi*18+p+16	2		Int16
filter parameters	b+fi*18+p+18	8 * number of parameters		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

Appendix A



The `MegDefs.h` Header File

The following file has been reformatted for this document.

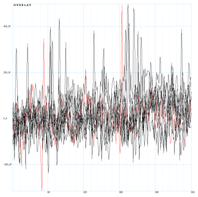
```
/*
===========================================================================
 * $Header: MEGDefs.h,v 10.5 98/03/13 12:33:40 sixtus Exp $
 *
===========================================================================
 *
 * MEGDefs.h
 *
 * Definitions used by the dataset and its clients.
 *
 * Copyright (c) CTF Systems Inc., 1995-1996.  All Rights Reserved.
 * Commercially Confidential Information
 *
 *
===========================================================================
*/

#ifndef _H_MegDefs
#define _H_MegDefs


#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */

#define MAX_COILS8
#define SENSOR_LABEL 31
#define MAX_NUM_COEFS50

#defineMAX_AVERAGE_BINS 8

#define MAX_BALANCINGMAX_NUM_COEFS

#defineGENERALRESID     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
```

```
/*
*     Some CTF defined types
*/

typedef enum { false, true } CTFBoolean;

typedef shortInt16;
typedeflong Int32;
typedefunsigned longBit32;

typedefchar Char;
typedefchar CChar;
typedefchar *CStrPtr;

typedefunsigned char UCChar;

typedefChar CStr32[ 32 ];
typedefChar CStr60[ 60 ];
typedefChar CStr255[ 255 ];
typedefChar CStr256[ 256 ];

typedefenum {           eMEGReference,
                        eMEGReference1,
                        eMEGReference2,
                        eMEGReference3,
                        eMEGSensor,
                        eMEGSensor1,
                        eMEGSensor2,
                        eMEGSensor3,
                        eEEGRef,
                        eEEGSensor,
                        eADCRef,
                        eStimRef,
                        eTimeRef,
                        ePositionRef,
                        eDACRef,
                        eSAMSensor,
                        eOtherRef,
                        eInvalidType

            } SensorType;


typedefenum {           MEGRef,
                        MEGSensor,
                        EEGRef,
                        EEGSensor,
                        ADCRef,
                        StimRef,
                        TimeRef,
                        PositionRef,
                        DACRef,
                        SAMSensor,
                        badMEGSensor,
                        badEEGSensor,
                        OtherRef,
                        InvalidClass

            } eChType;


typedef eChType SensorClass;

typedef enum { CIRCULAR, SQUARE } coiltype;

typedef coiltype CoilType;


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


typedef union d2_point
{
      struct                        { double x,y; } c;
      struct                        { double r,theta; } p;
      double                        point[2];

} d2_point;


typedef union d3_point_ext /* Externally store points as double */
{
      struct                        { double x,y,z, junk ;} c;
      struct                        { double r,theta,phi, junk ;} s;
      double            point[4];
} d3_point_ext;


typedef struct CoilRec_ext
{
      d3_point_extposition;/* position of coil */
      d3_point_extorient;           /* orientation of coil */
      Int16             numturns;   /* number of turns making up the coil
*/
      double            area;                   /* area of coil*/
} CoilRec_ext;


typedef struct CoilRec
```

```
{
      d3_pointposition; /* position of coil */
      d3_pointorient;             /* orientation of coil */
      Int16             numturns;   /* number of turns making up the coil
*/
      double            area;                   /* area of coil */
} CoilRec;


typedef struct coef_List
{
      Int16                         index;
      CChar                         name[SENSOR_LABEL];
} Coef_List;

typedef struct
{
      Int16             sensorTypeIndex;
      Int16             originalRunNum;
      CoilTypecoilShape;
      double            properGain;             /* may be corrected */
      double            qGain;
      double            ioGain;
      double            ioOffset;
      Int16             numCoils;
      Int16             grad_order_no;
      CoilRec_extcoilTbl[MAX_COILS];
      CoilRec_extHdcoilTbl[MAX_COILS];

} NewSensorResRec;


typedef struct CoefResRec/* Making generic resource for coefficients. */
{
      Int16                         num_of_coefs;
      CChar
sensor_list[MAX_BALANCING][SENSOR_LABEL];
      double                        coefs_list[MAX_BALANCING];
} 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[60];
      Int32 size; /* length of following array */
      CStrPtrnf_run_descriptor;
```

```
} meg4FileSetup;


typedef enum { CLASSERROR, BUTTERWORTH } classType;
typedef enum { TYPERROR, LOWPASS, HIGHPASS, NOTCH } filtType;

typedef struct
{
            double freq;
            classType fClass;
            filtType fType;
            Int16 numParam;
            double* params;
} 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 };

/*
 * Trigger structure for the meg4 dataset
 */
typedef struct
{
            UCChar primaryTrigger;
            UCChar secondaryTrigger[MAX_AVERAGE_BINS];
            UCChar triggerPolarityMask;
} meg40TriggerData;

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


typedef struct
{
            Int32 no_samples;
            Int16 no_channels;
            double sample_rate;
            double epoch_time;
            Int16 no_trials;
            Int32 preTrigPts;
            Int16 no_trials_done;
            Int16 no_trials_display;
            CTFBoolean save_trials;
            union
            {
                        meg40TriggerData meg40trig;
                        meg41TriggerData meg41trig;
            };
            Int16 trigger_mode;
            CTFBoolean accept_reject_Flag;
            Int16 run_time_display;
            CTFBoolean zero_Head_Flag;
            CTFBoolean artifact_mode;

} new_general_setup_rec_ext;


typedef struct
{
      CStr256                                   appName;
      CStr256                                   dataOrigin;
      CStr256                                   dataDescription;
      Int16                                     no_trials_avgd;
      CChar                                     data_time[255];
      CChar                                     data_date[255];
      new_general_setup_rec_extgSetUp;
      meg4FileSetup                nfSetUp;

} meg41GeneralResRec;

typedef struct
{
      CStr32           sensorName;
      Bit32            coefType;
      CoefResReccoefRec;
} SensorCoefResRec;

#ifdef __cplusplus
}
#endif /* __cplusplus */


#endif /* _H_MegDefs */
```

Appendix B



A Sample Program

The following code demonstrates how to use the declarations and definitions in appendix A.

```
#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);
}
```