GeoAcoustics Limited, Shuttleworth Close,
Gapton Hall Industrial Estate, Great Yarmouth,
Norfolk, NR31 0NQ, England.
National Tel: 01493 600666
Fax: 01493 651100
International Tel: +44 1493 600666
Fax: +44 1493 651100
E-Mail sales@geoacoustics.co.uk

GeoSwath Plus

Raw Data File Format
9-GS+-6061/B

Project /Product	GeoSwath Plus	Print Control (Printed 08-02-2005)
Document Title	Raw Data File Format	
Document Number	9-GS+-6061/B	
Author	Philip Carpenter	
Distribution	Master: Drawing File	

Amendment History						
Issue	Date	Status	Pages	Checked	Approved	Client
B	28-01-2005	Supercedes SWATH-6061/BA	16	A.Beck.	A.Beck.	

Amendment Details		
AMD	CTD	Details

General Description

The GeoSwath raw data file (extension **rdf**) is used to store the raw information acquired by the GeoAcoustics GeoSwath Plus wide swath bathymetry system.

Sample source code files, **rdf.c & rdf.h**, are included with this document which contain functions to read and write data stored in this format. The source code is provided 'as is' and may be used at your own risk. GeoAcoustics Limited will not be held responsible for any consequential damage resulting from the use, misuse, or inability to use the code.

Methodology

All data is stored within the raw data file on a ping by ping basis, each ping record contains all of the necessary information for that ping. The structure of the raw data file can be seen below:-

File Structure:-

File Header	Ping 1	Ping 2	Ping 3	…….	Ping n

File Header:-

Creation Time	File Header Size	Ping Header Size	Original Filename	System Frequency	Echosounder Type

Ping:-

Ping Header	Ping Data

Ping Header:-

Ping Number	Ping Time	Previous Ping Position	Ping Size	Number of Navigation Samples	Number of Attitude Samples	Number of Heading Samples	Number of Echosounder Samples

Number of miniSVS Samples	Number of Aux1 Samples	Number of Aux2 Samples	Ping Length	Transmit Pulse Length	Transmit Power	Sidescan Gain	Number of Data Samples

Side	Navigation Strings Size	Attitude Strings Size	Heading Strings Size	Echosounder Strings Size	miniSVS Strings Size	Aux1 Strings Size	Aux2 Strings Size

Binary Data Representation

All data written in the raw data file format use the structures shown below:-

Type Name	Bytes	Range Of Values
char	1	–128 to 127
unsigned char	1	0 to 255
wchar_t	2	For UNICODE characters
short	2	–32,768 to 32,767
unsigned short	2	0 to 65,535
int	4	–2,147,483,648 to 2,147,483,647
unsigned int	4	0 to 4,294,967,295
long	4	–2,147,483,648 to 2,147,483,647
unsigned long	4	0 to 4,294,967,295
float	4	3.4E +/- 38 (7 digits)
double	8	1.7E +/- 308 (15 digits)
long double	10	1.2E +/- 4932 (19 digits)

All data is written using the Intel 80x86 byte ordering (LSB to MSB). If a raw data file is to be processed on a non-Intel computer such as one from Sun Microsystems™, Inc., Silicon Graphics®, Inc., or Apple Computer®, Inc., the order of the bytes in all values must be exactly reversed. For example, a float value (4 bytes) would need to be reordered from (1,2,3,4) to (4,3,2,1) in the target machine's memory before treating the number as a floating-point value. This effectively converts the value from little-endian (least significant byte first) to big-endian (most-significant byte first).

File Header Format

Each raw data file begins with a file header record and is followed by one or more ping records. The format of this record is shown below:-

Element Name	Bytes	Type	Description
Creation	4	unsigned int	Creation time of raw data file
Raw_header_size	2	short	Size of file header in bytes
Raw_ping_header_size	2	short	Size of ping header in bytes
File_name	512	wchar_t[256]	Original filename
Frequency	4	int	System frequency
Echo_type	2	short	Echosounder type
Spare	18	char[18]	Reserved

Ping Header Format

Each ping within the raw data file following the file header record starts with a ping header record, the format of this record is shown below:-

Element Name	Bytes	Type	Description
Ping_number	4	int	Ping number
Ping_time	8	double	Start time of ping
Previous_ping_position	4	int	Start position of previous ping
Ping_size	4	int	Size of ping data in bytes
Navigation_number	1	char	Number of navigation samples in ping
Attitude_number	1	char	Number of attitude samples in ping
Heading_number	1	char	Number of heading samples in ping
Echosounder_number	1	char	Number of echosounder samples in ping
MiniSVS_number	1	char	Number of MiniSVS samples in ping
Aux1_number	1	char	Number of aux1 samples in ping
Aux2_number	1	char	Number of aux2 samples in ping
Ping_length	2	short	Ping length in metres
Pulse_length	1	char	Transmit pulse length
Power	1	char	Transmit power
Sidescan_gain	1	char	Sidescan gain channel
Sample_number	4	int	Number of range/angle/amplitude samples
Side	1	char	Port (0), Starboard(1)
Navigation_strings_size	2	short	Size of raw navigation strings
Attitude_strings_size	2	short	Size of raw attitude strings
Heading_strings_size	2	short	Size of raw heading strings
Echosounder_strings_size	2	short	Size of raw echosounder strings
MiniSVS_strings_size	2	short	Size of raw miniSVS strings

Aux1_strings_size	2	short	Size of raw aux 1 strings
Aux2_strings_size	2	short	Size of raw aux 2 strings
Spare	27	char[13]	Reserved

Ping Data Format

Following the ping header each ping record contains the ping data record, the format of the ping data record is shown below:-

Element Name	Bytes	Type	Description
Raa		RAA[]	Range/angle/amplitude samples
Navigation		NAV[]	Navigation samples
Attitude		MRU[]	Attitude samples
Heading		GYRO[]	Heading samples
Echosounder		ECHO[]	Echosounder samples
MiniSVS		MINISVS[]	MiniSVS samples
Aux1_times		double[]	Aux1 time stamps
Aux2_times		double[]	Aux2 time stamps
Navigation_strings		char[][256]	Navigation strings
Attitude_strings		char[][256]	Attitude strings
Heading_strings		char[][256]	Heading strings
Echosounder_strings		char[][256]	Echosounder strings
MiniSVS_strings		char[][256]	MiniSVS strings
Aux1_strings		char[][256]	Aux1 strings
Aux2_strings		char[][256]	Aux2 strings

```
/**************************************************************************
        Sample Source Code
        Filename 'rdf.c'

        (c) Copyright 2003 GeoAcoustics Limited

**************************************************************************/



#include <windows.h>
#include <winuser.h>
#include <fcntl.h>
#include <io.h>
#include <time.h>
#include <string.h>
#include <conio.h>
#include <math.h>
#include <malloc.h>
#include <stdlib.h>
#include <stdio.h>
#include <commdlg.h>
#include <commctrl.h>

#include "rdf.h"

/**************************************************************************
```

```
*****************************************************************************/
BOOL read_raw_file_header(RAWFILEHEADER *lfh, FILE *linfile)
{
        if(fread(lfh, sizeof(RAWFILEHEADER), 1, linfile) != 1)
                return(FALSE);

        return(TRUE);
}

/*****************************************************************************

*****************************************************************************/
BOOL read_raw_ping_header(RAWPINGHEADER *lph, FILE *linfile)
{
        if(fread(lph, sizeof(RAWPINGHEADER), 1, linfile) != 1)
                return(FALSE);

        return(TRUE);
}

/*****************************************************************************

*****************************************************************************/
BOOL read_raw_ping_data(char *lping_buffer, int lsize, FILE *linfile)
{
        if(fread(lping_buffer, sizeof(char), lsize, linfile) != (unsigned)lsize)
                return(FALSE);


        return(TRUE);
}

/*****************************************************************************
```

```
*******************************************************************************/
void get_ping_data(RAA **lraa, NAV **lnav, MRU **lmru, GYRO **lgyro, ECHO **lecho, MINISVS **lsvs, double **laux1, double **laux2,
RAWPINGHEADER lph, char *lping_buffer)
{

        *lraa = (RAA *)lping_buffer;
        *lnav = (NAV *)((char *)*lraa + (lph.sample_number * sizeof(RAA)));
        *lmru = (MRU *)((char *)*lnav + (lph.navigation_number * sizeof(NAV)));
        *lgyro = (GYRO *)((char *)*lmru + (lph.attitude_number * sizeof(MRU)));
        *lecho = (ECHO *)((char *)*lgyro + (lph.heading_number * sizeof(GYRO)));
        *lsvs = (MINISVS *)((char *)*lecho + (lph.echosounder_number * sizeof(ECHO)));
        *laux1 = (double *)((char *)*lsvs + (lph.miniSVS_number * sizeof(MINISVS)));
        *laux2 = (double *)((char *)*laux1 + (lph.aux1_number * sizeof(double)));


}

/*******************************************************************************

*******************************************************************************/
void get_raw_strings(char **lnav_strings, char **lmru_strings, char **lgyro_strings, char **lecho_strings, char **lsvs_strings, char **laux1_strings,
char **laux2_strings, RAWPINGHEADER lph, char *lping_buffer)
{

        *lnav_strings = (char *)(lping_buffer + (lph.sample_number * sizeof(RAA)) + (lph.navigation_number * sizeof(NAV)) + (lph.attitude_number
* sizeof(MRU)) + (lph.heading_number * sizeof(GYRO)) + (lph.echosounder_number * sizeof(ECHO)) + (lph.miniSVS_number * sizeof(MINISVS))
+ (lph.aux1_number * sizeof(double)) + (lph.aux2_number * sizeof(double)));
        *lmru_strings = (char *)(*lnav_strings + lph.navigation_strings_size);
        *lgyro_strings = (char *)(*lmru_strings + lph.attitude_strings_size);
        *lecho_strings = (char *)(*lgyro_strings + lph.heading_strings_size);
        *lsvs_strings = (char *)(*lecho_strings + lph.echosounder_strings_size);
        *laux1_strings = (char *)(*lsvs_strings + lph.miniSVS_strings_size);
```

```
        *laux2_strings = (char *)(*laux1_strings + lph.aux1_strings_size);

}
/*****************************************************************************


*****************************************************************************/
BOOL write_raw_file_header(RAWFILEHEADER *lfh, FILE *loutfile)
{
        if(fwrite(lfh, sizeof(RAWFILEHEADER), 1, loutfile) != 1)
                return(FALSE);

        fflush(loutfile);
        return(TRUE);
}

/*****************************************************************************


*****************************************************************************/
BOOL write_raw_ping_header(RAWPINGHEADER *lph, FILE *loutfile)
{
        if(fwrite(lph, sizeof(RAWPINGHEADER), 1, loutfile) != 1)
                return(FALSE);

        fflush(loutfile);
        return(TRUE);
}

/*****************************************************************************
```

```
*****************************************************************************/
BOOL write_raw_ping_data(char *lping_buffer, int lsize, FILE *loutfile)
{
        if(fwrite(lping_buffer, sizeof(char), lsize, loutfile) != (unsigned)lsize)
                return(FALSE);

        fflush(loutfile);
        return(TRUE);
}
```

```
/*****************************************************************************
        Sample Source Code
        Filename 'rdf.h'

        (c) Copyright 2003 GeoAcoustics Limited

*****************************************************************************/


//////////////////////////////////////////////////////////////////////////
//
// Structures
//
//////////////////////////////////////////////////////////////////////////
typedef struct
{
        unsigned        short           time;           //Time in wavelengths
                                short           sine;           //Sine of return angle * 32000
        unsigned        short           amplitude;      //16bit amplitude value
} RAA;



typedef struct
{
                                double          x;                      //Easting in metres
                                double          y;                      //Northing in metres
                                float           z;                      //Antenna height in metres
                                double          time;           //GPS time, number of seconds elapsed since midnight (00:00:00), January 1, 1970
                                double          time_stamp;     //Time stamp, number of seconds elapsed since midnight (00:00:00), January 1, 1970
unsigned                char            quality;        //GPS quality indicator
} NAV;
```

```
typedef struct
{
                    float       roll;       //Roll in degrees
                    float       pitch;      //Pitch in degrees
                    float       heave;      //Heave in metres
                    double      time_stamp; //Time stamp, number of seconds elapsed since midnight (00:00:00), January 1, 1970
} MRU;

typedef struct
{
                    float       heading;    //Heading in degrees
                    double      time_stamp; //Time stamp, number of seconds elapsed since midnight (00:00:00), January 1, 1970
} GYRO;

typedef struct
{
                    float       depth1;     //Depth1 in metres
                    float       depth2;     //Depth2 in metres
                    double      time_stamp; //Time stamp, number of seconds elapsed since midnight (00:00:00), January 1, 1970
} ECHO;

typedef struct
{
                    float       velocity;   //MiniSVS velocity in metres per second
                    double      time_stamp; //Time stamp, number of seconds elapsed since midnight (00:00:00), January 1, 1970
} MINISVS;
```

```
typedef struct
{
        unsigned        int                     creation;                       //File creation time, number of seconds elapsed since
midnight (00:00:00), January 1, 1970
                        short           raw_header_size;                //Size of this structure in bytes
                        short           raw_ping_header_size;           //Size of ping header structure in bytes
                        char            filename[512];                  //Original filename
                        int                     frequency;                              //system frequency, in hertz
                        short           echo_type;                      //echosounder type
                        char            spare[18];
} RAWFILEHEADER;

typedef struct
{
                        int                     ping_number;                    //Ping number
                        double          ping_time;                      //Ping time stamp, number of seconds elapsed since midnight
(00:00:00), January 1, 1970
                        int                     previous_ping_position;         //File position of previous ping
                        int                     ping_size;                              //size of ping in bytes
        unsigned        char            navigation_number;              //number of navigation samples in ping
        unsigned        char            attitude_number;                //number of attitude samples in ping
        unsigned        char            heading_number;                         //number of heading samples in ping
        unsigned        char            echosounder_number;             //number of echosunder samples in ping
        unsigned        char            miniSVS_number;                         //number of miniSVS samples in ping
        unsigned        char            aux1_number;                    //number of aux1 strings in ping
        unsigned        char            aux2_number;                    //number of aux2 strings in ping
                        short           ping_length;                            //ping length in metres
        unsigned        char            pulse_length;                   //transmit pulse length (1 - 7)
        unsigned        char            power;                                  //transmit power (0 - 9)
        unsigned        char            sidescan_gain;                  //sidscan gain (0 - 3)
                        int                     sample_number;                          //number of RAA samples in ping
        unsigned        char            side;                                   //side, port = 0, starboard = 1
                        short           navigation_strings_size;        //total size of navigation strings in bytes
```

```
                        short       attitude_strings_size;      //total size of attitude strings in bytes
                        short       heading_strings_size;       //total size of heading strings in bytes
                        short       echosounder_strings_size;   //total size of echosounder strings in bytes
                        short       miniSVS_strings_size;           //total size of miniSVS strings in bytes
                        short       aux1_strings_size;              //total size of aux1 strings in bytes
                        short       aux2_strings_size;              //total size of aux2 strings in bytes
                        char        spare[13];
}RAWPINGHEADER;

//////////////////////////////////////////////////////////////////////////
// function prototypes

BOOL read_raw_file_header(RAWFILEHEADER *, FILE *);
BOOL read_raw_ping_header(RAWPINGHEADER *, FILE *);
BOOL read_raw_ping_data(char *, int, FILE *);
void    get_ping_data(RAA **, NAV **, MRU **, GYRO **, ECHO **, MINISVS **, double **, double **, RAWPINGHEADER, char *);
void    get_raw_strings(char **, char **, char **, char **, char **, char **, char **, RAWPINGHEADER, char *);
BOOL write_raw_file_header(RAWFILEHEADER *, FILE *);
BOOL write_raw_ping_header(RAWPINGHEADER *, FILE *);
BOOL write_raw_ping_data(char *, int, FILE *);
```