

Degum

LPM Data Processing Software

Algorithm Description and Programmer's Guide

Document Version

Version 1.0 – Original

Version 1.1 – Rewrite following code restructure

Version 2.0 – First formal draft

Version 2.1 – Added DGRange –s option, added short block status bit information

Version 2.2 – Added DGHeader application

Version 2.3 – Changed ADCZero to fix all components and added Despike option to Degum2

Version 2.4 – Added output path option and use of quotation marks to DGRange

Version 2.5 – Modified output file naming for DGRange –s option

Version 2.6 – Removed DGHeader, added DGMod

Author

David Judge, PSD BAS, 24/05/06

Abbreviations

1. LPM Low Power Magnetometer
2. EM Event Manager

Associated Documents

1. LPM Processing Software specification

Associated CD

Release 1.1 – 11/07/06



**British
Antarctic Survey**

NATURAL ENVIRONMENT RESEARCH COUNCIL

Table of Contents

Degum	1
Software Overview.....	3
Processing Parameters.....	3
Output File Name Convention.....	7
Processing History.....	7
Degum1	8
Overview.....	8
Output Files and Formats.....	8
Processing Overview.....	10
Processing Algorithm.....	10
Degum2	19
Overview.....	19
Instrumental artefacts.....	19
Cleaning Algorithms.....	19
Processing Overview.....	20
Files and Formats.....	20
Processing Algorithm.....	21
Degum3	24
Overview.....	24
Quiet Period Determination.....	24
Rotation Angle Calculation.....	24
Processing Algorithm.....	24
Degum4	26
Overview.....	26
Conversion to nT units.....	26
Rotation to the Standard Reference Frame.....	26
Output File Format.....	26
Processing Algorithm.....	27
DGRange	28
Overview.....	28
Command Line Parameters.....	28
Output File Name.....	29
Output File Format.....	30
File Feedback.....	30
Further Processing.....	30
Processing Algorithm.....	31
DGMod	33
Overview.....	33
Command Line Parameters.....	33
Output File Name.....	33
Output File Format.....	33
Further Processing.....	33
Processing Algorithm.....	34
Appendix A	36

Software Overview

The Degum processing package is a suite of applications designed to reformat raw LPM data files into a scientifically usable form. To achieve this end, the data must undergo a number of processing tasks. The tasks have been divided into six groups and a separate application is provided to perform the processing for each group. Intermediate files are output to transfer the data between applications. Table DG.1 lists the applications, details the processing tasks they perform and lists the input / output files.

Application	Processing Tasks	Input	Output
Degum1	Validates raw file information blocks Separates housekeeping data and writes to O/P file Separates engineering data and writes to O/P file Computes a time stamp for each data sample and housekeeping block Writes all valid, time stamped data, associated time uncertainty and status information to O/P file Provides optional additional QC printout	Raw LPM file (V2 or later) Optional INI file	Housekeeping file (text) Engineering file (text) Data (.dg1) file Processing information is also passed to the user's console which may be redirected to a further file
Degum2	Implements data cleaning algorithms Records new values and applied corrections Optional data QC processing	dg1 file Optional INI file	Data (.dg2) file Optional QC file
Degum3	Determines rotation angle required to rotate the data to the standard reference frame	dg2 file Optional INI file	A report of the rotation angle to the user's console
Degum4	Rotates the data and applied corrections by supplied angle	dg2 file Rotation angle Optional INI file	Data (.dg4) file
DGRange	Extracts a supplied range of data from a DG1, DG2 or DG4 file and exports to binary or text format	dg1, dg2 or dg4 file	Binary (.dg#) or text file
DGMod	Appends text information to an existing DG1, DG2 or DG4 file or Replaces data in an existing DG4 file	dg1, dg2 or dg4 file	Binary (.dg#) file

Table DG.1 Application overview

Processing Parameters

Each Degum application requires a number of parameters to be available before processing can commence. Table DG.2 describes the processing parameters for each application and gives their default values. A more detailed description of the parameter use is given later in this document.

Application	Parameter	Description	Default Value
All		<i>These are general parameters that may be used by several applications</i>	
	ShowHistory	A boolean controlling the output of a file's processing history to the user's console	True
	TXTDelimiter	A delimiting character to separate fields in an output text file	,
	CommentDelimiter	A character to denote comment lines in an output text file – placed at the beginning of each comment line	;
	DAC2nT1 DAC2nT2 ADC2nT	Three coefficients of equation to convert raw LPM data units to nT	128.0 70000.0 0.77213
Degum1	MaxLineLength	The character buffer size required to read the longest raw LPM file text line	256
	NumSatMin	The minimum number of satellites contributing to a GPS fix for it to be considered a good fix	1
	Intervals	A comma delimited list of all possible sampling intervals contained within the input LPM file	1, 3, 5, 9, 10, 15, 60

Application	Parameter	Description	Default Value
	FixingLoopsMin FixingLoopsMax NormalLoopsMin NormalLoopsMax	The minimum and maximum range of possible loop counts for Fixing and Normal loops within the input LPM file	10 600 240 21300
	SwapBytes	A boolean controlling the application of byte swapping during reading of the input LPM data samples	False
	RateAverage	The number of prior clock rates to average to provide the comparison clock rate during time stamp calculations	10
	GPSQC	A boolean controlling the output of additional GPS fix QC information	False
	BlockQC	A boolean controlling the output of additional LPM file block QC information	False
	TimingQC	A boolean controlling the output of additional time stamp QC information	False
Degum2	ADCZero	A comma delimited list of sample indices to be tested for ADC Zero filtering	18759
	Despike	A comma delimited list of sample indices whose data is to be interpolated from neighbouring values	None
	ADCQC	A boolean controlling the output of an additional ADC Zero QC file	False
	ADCQCMin ADCQCMax	The minimum and maximum extent of sample values to be used in the ADC Zero QC	-1 1
	ADCQCRange	The range of sample indices to export per ADC Zero sample detected during the ADC QC process	5
Degum3	WindowLength	The window length, in seconds, to use for the quiet period scan	86400
	Moveup	The time, in seconds, the scan window is advanced during successive quiet period scans	43200
Degum4	Rotation(Deg)	The angle, in degrees, to rotate the data <i>N.B. This parameter must be supplied as a command line parameter – it can not be supplied in an INI file</i>	None
DGRange	-t['ch'] -h['ch'] [-r Day Month Year Day Month Year]	Export a text file with optional field delimiting character Export the file processing history with optional comment line character Start and end times defining the range of data to export. Identical if only a single day's data is required <i>N.B. These parameters must be supplied as command line parameters – they can not be supplied in an INI file</i>	None
DGMod	[-h TXTFile] [-d Component D M YYYY D M YYYY Value]	A file containing the text to append to the file header Component, date range and replacement value	None None

Table DG.2 Processing Parameters

An application's parameters, unless otherwise detailed above, can be supplied by the user in three ways:

- The parameters can simply be omitted - allowing the default values to be used
- They can be supplied via an INI file
- Some may be supplied via the application's command line parameter list

The order of precedence, low to high, is: Default -> INI File -> Command Line

INI File

An INI file contains grouped processing parameters for single or multiple applications. It is passed to an application as an optional command line parameter. A single INI file is usually used for all Degum applications and is of the form:

```
[Group]
Variable = Value
Variable = Value, Value, Value,...
;Comment
```

where *[Group]* is the group ID, *Variable* is a parameter name and *Value* is its supplied value. List parameter values are supplied as a comma delimited string of values and comment lines are denoted by a ';' as the first non-space character.

The group ID is the application name to which the subsequent parameters pertain or 'General' for non-application specific parameters.

A typical INI file would be:

```
;INI file for Degum, LPM processing software

[General]
ShowHistory = True
TXTDelimiter = ,
CommentDelimiter = ;
DAC2nT1 = 128.0
DAC2nT2 = 70000.0
ADC2nT = 0.77213

[Degum1]
MaxLineLength = 256
NumSatMin = 1
Intervals = 1, 3, 5, 9, 10, 15, 60
FixingLoopsMin = 10
FixingLoopsMax = 600
NormalLoopsMin = 240
NormalLoopsMax = 21300
SwapBytes = False
RateAverage = 10
GPSQC = False
BlockQC = True
TimingQC = False

[Degum2]
ADCZero = 13690, 13694, 18759,18994, 18999
;Despike =
ADCQC = False
ADCQCMin = -1
ADCQCMax = 1
ADCQCRange = 5

[Degum3]
WindowLength = 86400
Moveup = 43200
```

If an INI file is being used, it is not necessary to supply a complete list of all parameters – any parameters omitted will take their default or command line supplied values.

Command Line

A limited subset of the processing parameters may be supplied as optional parameters on an application's command line. As detailed in Table DG.2, some processing parameters can only be supplied via the command line. Table DG.3 lists the parameters that are available as command line options:

Application	Command Line Parameter	Processing Parameters	Description
Degum1	-s# -a# -g -b -t	NumSatMin RateAverage GPSQC BlockQC TimingQC	'#' is the minimum number of satellites '#' is the number of rates to average GPC QC is enabled Block QC is enabled Timing QC is enabled
Degum2	-z # #... -d # #... -q min max range	ADCZero Despike ADCQC, ADCQCMin, ADCQCMax, ADCQCRange	'#' are sample indices to be tested '#' are sample indices to be interpolated ADC QC is enabled using supplied min, max and range
Degum3	-w length moveup	WindowLength, Moveup	Length and moveup, both in seconds, of quiet time scan window
Degum4	Rotation(Deg)	Rotation(Deg)	The rotation angle, in degrees
DGRange	-t['ch'] -h['ch'] [-rDay Month Year Day Month Year]	-t['ch'] -h['ch'] [-rDay Month Year Day Month Year]	Export a text file with optional field delimiting character Export the file processing history with optional comment line character Start and end times defining the range of data to export. These will be identical if only a single day's data is required
DGMod	[-h TXTFile] [-d Component D M YYYY D M YYYY Value]	[-h TXTFile] [-d Component D M YYYY D M YYYY Value]	A file containing the text to append to the file header Component, date range and replacement value

Table DG.3 Command line parameters

If a Degum application is run without any parameters of input file names then the application *usage* will be displayed to the user:

```
Usage: Degum1 [INIfile] [-s#] [-a#] [-g] [-b] [-t] Datafile
```

```
Usage: Degum2 [INIfile] [-z # #...] [-d # #...] [-q min max range] DG1File
```

```
Usage: Degum3 [INIfile] [-w length moveup] DG2File
```

```
Usage: Degum4 [INIfile] Rotation(Deg) DG2File
```

```
Usage: DGRange [-t['ch']] [-h['ch']] [-s] [-rDay Month Year Day Month Year] DGFile
```

```
Usage: DGMod [-h TXTFile] [-d Component D M YYYY D M YYYY Value] DGFile
```

Output File Name Convention

Output file names are not supplied by the user but are constructed from the concatenation of elements of the input file name with an application specific postfix. Table DG.4 details output file naming for each application.

Application	Output file type	Output file names
Degum1	Housekeeping file Engineering file Data file	[Input file name] + [.HK.txt] [Input file name] + [.EG.txt] [Input file name] + [.dg1]
Degum2	Data file QC file	[Input file name – stripped of '.dg1'] + [.dg2] [Input file name – stripped of '.dg1'] + [.dg2.AdcQC.txt]
Degum4	Data file	[Input file name – stripped of '.dg2'] + [.dg4]
DGRRange	Binary data file Text data file	[Input file name] + [ddmmyy-ddmmyy] [Input file name] + [ddmmyy-ddmmyy] + [.txt]
DGMod	Binary data file	[Input file name] + [.mod]

Table DG.4 Output file names

Processing History

Degum output files store, in a header at the beginning of the file, a list of all the applications, processing parameters and software versions used to process the data to that stage.

software versions are of the form:

```
vDG = 1.1.0
```

where the first digit is the application identifier:

1 = Degum1, 2 = Degum2, 3 = Degum3, 4 = Degum4, 5 = DGRRange and 6 = DGMod

and the remaining digits are the application's source code version (1.0 in this case)

The initial task performed by all Degum applications prior to data processing is to read the input file's header for its processing history, append the information and parameters for the processing that is about to be performed and write it as the header of the output file.

If the *ShowHistory* processing parameter is set to *True* then the processing history will be displayed to the user during processing.

The header consists of a 4 byte integer field specifying the number of header text strings followed by a list of text strings containing the processing history information. Each string is terminated by a newline (\n) character.

A file's processing history may be viewed and/or expanded using the DGMod application – see the DGMod section below for further information.

Degum1

Overview

The main task of the Degum1 application is to derive an accurate time stamp for all valid data samples. It also extracts ancillary information stored in the raw LPM files and exports it to additional files independent of the main magnetometer data output file. During processing the input file's data blocks and GPS fix information are validated and a time uncertainty and status information is assigned to each output sample. Optional QC tasks can be requested to provide further information about the performance of the processing algorithm.

This document assumes that the reader is familiar with the LPM data file format (see Associated Documents 1).

Output Files and Formats

Degum1 writes three files, one for housekeeping data, one for engineering data and one main output for the magnetometer data (see the section above for output file naming). Housekeeping and engineering files will contain processing history information if the *ShowHistory* processing parameter is set to *True*. Information is also presented to the user during processing.

Housekeeping files are text files and contain all the housekeeping data extracted from the input data blocks and time stamped during processing. The first line is always a comment line supplying a field description.

An example housekeeping file:

```
;Fields: Year Month Day Daynumber Hour Minute Second SupplyV Temp Temp Temp Temp Temp  
Temp Temp Temp WV SV  
2001, 12, 13, 347, 18, 42, 11.4, 751, 547, 525, 516, 568, 499, 2, 3, 2, 2, 386  
2001, 12, 13, 347, 19, 46, 50.2, 745, 539, 526, 516, 566, 498, 2, 2, 1, 1, 335  
2001, 12, 13, 347, 20, 51, 27.5, 734, 535, 526, 516, 566, 498, 2, 2, 1, 1, 313  
2001, 12, 13, 347, 21, 56, 4.7, 709, 533, 526, 513, 566, 498, 2, 2, 1, 1, 311
```

Engineering files are text files and primarily contain time information gathered during processing of GPS pairs (see below for further details). The first line is always a comment line supplying a field description but further explanation is given here:

Fields	Description
UTSeconds Year Month Day Hour Minute Second	Time of interval start GPS fix (fix0)
UTSeconds Year Month Day Hour Minute Second	Time of interval end GPS fix (fix1)
dUTTime	(UT Time of fix 1) – (UT Time of fix 0)
dLocalTime	(Local Time of fix 1) – (Local Time of fix 0)
dUT/dLT	Clock drift rate (dUTTime / dLocalTime)
dUT-dLT	Clock difference (dUTTime - dLocalTime)
Gap	Length of data gap in seconds
XVar	Variance of X sample over interval
YVar	Variance of Y sample over interval
ZVar	Variance of Z sample over interval
TotalVar	Total variance over interval (XVar + YVar + ZVar)
Interval...	Sample interval of all events within interval

Table DG1.1 Engineering file field description

In addition the engineering file also contains statistics gathered during processing about the input file (presented to the user during processing). For example:

File Summary:

Average Latitude(Deg) = 84.354 S

Average Longitude(Deg) = 23.858 W

1 Header Block Events, 8091 Data Block Events, 0 Skip Block Events

8091 HK Events, 8091 Fixing Events, 8091 Normal Events

GPS Fixes: 8091 Successful, 0 Unsuccessful

0 Events not processed

22257250 Samples exported

Clock Drift Rates:

All GPS pairs :

Min Drift Rate = 0.993249, Max Drift Rate = 1.012718, Avg Drift Rate = 0.995686

GPS pairs without gaps:

Min Drift Rate = 0.993249, Max Drift Rate = 0.994039, Avg Drift Rate = 0.993579

The main output file, .dg1, is a binary file. It contains a record of the processing history and all the validated samples of magnetometer data. The data samples are exported in blocks each preceded by a block header. The file is terminated by 12 bytes of record keeping information. The full .dg1 format description is given here:

Structure	Variable	Size(bytes)	Description
.dg1	History DataHeader Sample Sample... DataHeader Sample Sample... ... RecordKeeping	[Variable]	Degum1 output.
History	NumStrings String(1)\n String(2)\n ... String(NumStrings)\n	4 [Variable]	Number of history strings History character string terminated by '\n' character
DataHeader	Type NumSam Interval	4 4 4	Enumerated variable 0 = DAC values 1 = Fixing samples 2 = Normal samples 3 = Info sample Number of samples in following data structure Sampling interval of following data structure
Sample	Time Uncertainty Status Data	8 2 2 6	UT Time encoded to seconds * 10 Uncertainty in time (seconds * 10) Status flags XYZ components of magnetometer data. 2 bytes each in ADC Units
RecordKeeping	NumFixing NumNormal NumSamples	4 4 4	Number of Fixing blocks Number of Normal blocks Total number of samples

Table DG1.2 .dg1 file format description

A sample status variable is a bit field (16 bits) containing information about the sample and its applied processing. All bits are initially set to zero and the following bits may be set by Degum1:

Bit	Description
0	Set if sample was acquired during the GPS Fixing loop
1	Set if sample was acquired during the Normal loop
6	Set if sample is within an interval that is anomalously short
7	Set if sample is within an interval affected by a Data Gap
8	Set if the Uncertainty overflows 16 bits

Table DG1.3 Sample status flag bits

Processing Overview

The data file is processed in one pass.

Data is extracted from the input file and validated on a block-by-block basis. If a block passes validation, it is divided into a number of Events. Each Event is passed to an Event Manager object which, depending on the Event type, acts on the data immediately or stores it for future processing. Flow continues until all block Events have been passed to the Event Manager and have been processed.

Processing Algorithm

Initialisation

The following steps initialise the application in preparation for data processing:

- Command line arguments are validated
- The INI file, if supplied, is read and validated
- The input file name and location are validated
- The input file version is determined by a scan for the first text line beginning with the word, *Program* in the file. The program version number (PV#) is extracted from the line and the file format is determined by:
 - File format is V2 if PV# is between 0 and 3.9
 - File format is V3 if PV# is greater than 3.9

Only LPM file versions at V2 or above can be processed by the Degum software
- An EventManager object is created to control processing events

Processing

The input file is opened and scanned for all blocks contained therein. The block type, header or data, is determined by the first character of the line following a START line ie. 'B' or 'H'. A Warning is issued to the user if neither is detected and a fresh START line scan is initiated.

Block Validation and Extraction

Each Header and Data block encountered is examined and the data within is extracted and validated. Two levels of validation are performed, Warning and Error. Warnings are issued for invalid non-critical data but will not terminate block processing. Whenever an Error is issued, the block processing is immediately terminated and all subsequent processing steps are skipped.

For Warning and Error message formats, see the User Messages section below

Block information is extracted on a line-by-line basis where each line is terminated by a CR/LF character pair. The internal line buffer is of unlimited size but a Warning is issued to the user if lines greater than the length supplied by the *MaxLineLength* processing parameter are detected.

The data extraction and validation performed on each block type is detailed below:

Header Block

The block is scanned for 8 target lines - not including its own START line - beginning with one of the following key words, the lines may occur in any order:

Program, Error, Last, Dac 0, Dac 1, Dac 2, END, START

General validation for all lines:

Warning	If a line is read but not at the expected position within the block
Error	If a target line is missing from the block

Specific extraction and validation performed on each line:

Key Word	Extraction	Warning	Error
Program	File format version	If version not as expected	
Error	Error count	If line read but unable to extract data	
Last	Last error	If line read but unable to extract data	
Dac 0, 1, 2	DAC value		If value missing or out of range
END	None but exits the scan		
START			If START encountered before END

Table DG1.4 Header block extraction and validation

The valid DAC value range is 0 - 255

Data Block

The block is scanned for 12 target lines - including its own START line - beginning with one of the following key words, the lines may occur in any order:

B, WV, SV, Interval, Fixing, Normal, GPS, \$GPZDA, \$GPGGA, #, END, START
 where '#' is a lone integer

General validation for all lines:

Warning If a line is read but not at the expected position within the block
 Error If a target line - except a \$GPxxx line - is missing from the block

Specific extraction and validation performed on each line:

Key Word	Extraction	Warning	Error
START (own)	TICKS value		If missing or invalid TICKS
B	House keeping data	If line read but unable to extract data	
WV	House keeping data	If line read but unable to extract data	
SV	House keeping data	If line read but unable to extract data	
Interval	Sample interval		If interval missing or out of range
Fixing	Fixing sample count		If count missing or out of range
Normal	Normal sample count		If count missing or out of range
GPS	TICKS value		If missing or invalid TICKS
\$GPZDA	GPS Time	If line read but unable to extract data	
\$GPGGA	Number of satellites tracked	If line read but unable to extract data	
#	TICKS value		If missing or invalid TICKS
END	TICKS value, exits the scan		If missing or invalid TICKS
START			If START encountered before END

Table DG1.5 Data block extraction and validation

The valid TICKS range is 0 – 1124

The valid sampling intervals are given by the processing parameter *Intervals*

The valid fixing sample count is given by the processing parameters *FixingLoopsMin* and *FixingLoopsMax*

The valid normal sample count is given by the processing parameters *NormalLoopsMin* and *NormalLoopsMax*

The minimum valid number of tracked satellites is given by the processing parameter *NumSatMin*

The fixing and normal data samples are validated by testing for the terminating CR/LF character pair. If they are not detected at the expected offset then an Error message will be issued.

When a NMEA sentence is encountered:

- A checksum value is calculated and validated against the sentence's internal checksum value.
- Required fields are tested to be present.

- The 'number of satellites tracked' value extracted from the GGA sentence is validated against a minimum value specified by the *NumSatMin* processing parameter

User Messages

Warning and Error user messages are of the form:

```
Title
Current input file offset value
Contents of the current line read buffer
Message (With indication of block type - HB or DB)
```

To make them more visible they are bounded by an upper and lower character string. Warnings are bounded by the '-' character, errors by '*':

For example:

```
-----
Warning:
Data file offset: 17306
Current Buffer: [Last Error      ]
HB: Unable to read last error
-----

*****
Error - Skipping Block!:
Data file offset: 24819
Current Buffer: [START    13.12.01 18.12.31      ]
DB: Unable to read START TICKS
*****
```

Progress reports, produced at the end of data extraction for each block, are of the form:

```
XX (##%): AA[=aa], BB=bb, CC=cc, ...
```

where XX is the block type – Header (HB) or data (DB)
##% is the percent read for the input file
BB=bb are extracted values

For example:

```
HB (58%): V2, EC=0, LE=0, Dac0=162, Dac1=129, Dac2=45
DB (65%): INT=1, FLoop=300, NLoop=3600, GPS Fix
DB (47%): INT=1, FLoop=300, NLoop=3600, No Fix
```

Events

At the completion of a block's extraction and validation process, an Event will be passed to the EventManager (EM) object. The type of event will depend upon the block type and its validation state:

Block Type	Validation Passed	Validation Failed
Header	HeaderBlock Event passed to EM	SkipBlock Event passed to EM
Data	DataBlock Event passed to EM	SkipBlock Event passed to EM

Table DG1.6 Events passed to the EM

In addition, when all blocks in the input file have been read and processed the final event, an EOFEvent will be passed to the EM.

Events contain data extracted from their associated block:

Event	Data
HeaderBlock	3 x DAC values Error count Last error
DataBlock	11 x Housekeeping values Sampling interval Fixing and Normal loop counts Input file offsets to the two sets of data samples 8 x TICKS values A boolean denoting valid GPS information GPS time GPS location
SkipBlock	A boolean denoting which type of block is being skipped, Header or Data
EOF	None

Table DG1.7 Event Data

The Event Manager Object

The EventManager's objective is to handle all incoming events, to process them and ultimately output the Event's data into a series of text and binary output files. To achieve this it maintains Event lists, and holds data values required to meet the processing requirements. Within the EM a received DataBlock Event is subdivided into a series of sub-events: QCEvents, HKEvents, GPSEvents and DataEvents (see below).

Two Event lists are required; one for timed events: QCEvents, HKEvents and DataEvents and one for GPSEvents.

Time Formats and Computation

Time information within received events must be restructured into standard formats that can be easily handled during event processing. Two sets of time information need to be considered: GPS fix time and the LPM processor local time.

GPS Fix Time

The GPS fix date and time (UT time) extracted from valid NMEA ZDA sentences and stored in a DataBlock Event must be encoded into a single number. The date/time pair is converted to seconds since 1st January 2000, accounting for an extra day during leap years. It is accurate to 0.1 seconds but is stored as an integer so is scaled accordingly (seconds * 10).

A year is considered a leap year if it is divisible by 400 or if it is divisible by 4 and not divisible by 100.

Local Time

Local times must be calculated for all Events before they can be added to an Event list

Local time is set to 0 whenever the Event lists are cleared and increases, as detailed below, with each DataBlock Event received.

A typical LPM Data block is:

```
* L1  START 13.12.01 18.42.01 29 T1
      B 751 #0 547 #1 525 #2 516 #3 568 #4 499 #5 2 #6 3 #7 2
      WV 2
      SV 386
      Interval is 1 I
      Fixing G P S loops 300 C1
      Normal loops 3600 C2
* L2  96 T2
      XXYYZZXXYYZZ...
* L3  75 T3
* L4  GPS 13.12.01 18.47.04 85 T4
* L5  $GPZDA,184713.3,13,12,2001,,*5F
      $GPGGA,184713.0,8253.949,S,01214.630,W,1,08,0.91,01976,M,-012,M,,*77
* L6  GPS 13.12.01 18.47.05 103 T5
* L7  115 T6
      XXYYZZXXYYZZ...
* L8  93 T7
* L9  END 13.12.01 19.47.04 93 T8

      NMEA char count (inc CR/LF) = 103 NC
```

The lines marked * denote events that may require a local time value. For the purposes of this explanation, a local time will be calculated for all of them.

Red text denotes values extracted from the block and stored in the DataBlock Event, blue text is a variable name for the extracted value and green text is a variable name for the required local times

The local time (L0) and TICKS (T0) values from the end of the previous block are also stored. They will both be 0 for the first block and are taken to be 0 for this explanation.

Two constants are also required:

$$K1 = \text{TICKS per Second} = 1125 / 60$$

$$K2 = \text{Serial Link Seconds per character at a baud rate of 4800} = 8 / 4800$$

To derive the 9 required local times, the following equations are used:

$$L1 = L0 + (T1 - T0)$$

$$L2 = L1 + (T2 - T1)$$

$$L3 = L2 + ((C1 - 1) * I * K1)$$

$$L4 = L3 + (T4 - T3)$$

$$L6 = L4 + (T5 - T4)$$

$$L5 = L6 - (NC * K2 * K1)$$

$$L7 = L6 + (T6 - T5)$$

$$L8 = L7 + ((C2 - 1) * I * K1)$$

$$L9 = L8 + (T8 - T7)$$

$$L0 = L9$$

$$T0 = T8$$

If adjacent TICKS values wrap then they must be unwrapped, by adding 1125, before they can be subtracted. For example, if T1 = 5 and T0 = 1120 then

$$(T1 - T0) = T1 + 1125 - T0 = 5 + 1125 - 1120 = 10$$

Event Tasks

The tasks performed by the EM upon receipt of each Event is detailed below:

HeaderBlock Event:

- All current stored Events are processed and exported, both Event lists are cleared
- The new DAC offsets are exported as a *DAC Block* and a flag is set to signify that DAC offsets have been received

DataBlock Event:

- Local times are calculated for all TICKS values
- A HKEvent, containing the HK local time and data, is constructed and added to the event list
- A DataEvent, containing the fixing samples local time and data, is constructed and added to the event list
- A DataEvent, containing the normal samples local time and data, is constructed and added to the event list
- If the DataBlock Event contains GPS time information then a GPSEvent, containing the Event's local time and UT time, is constructed and added to the GPSEvent list
- If a TimingQC has been requested, via the TimingQC processing parameter, QCEvents, containing the event local time, will be created and added to the event list – in chronological order

SkipBlockEvent:

- All stored Events are processed and exported, both Event lists are cleared
- If the SkipBlock Event was received via Header block processing then the DAC received flag is also cleared otherwise it is left unchanged

EOFEvent:

- All stored Events are processed and exported, both Event lists are cleared
- The three record keeping variables (*NumFixing*, *NumNormal* and *NumSamples*) are appended to the end of the main output file as an *Information Block*

Event Processing

Event processing can only commence if the DAC received flag is set.

- If DAC offsets have been received, the following processing algorithm is implemented:
- GPS events are treated in pairs in chronological order (Fix0 and Fix1)
- For each pair in the GPSEvent list, the Local Clock Rate is calculated:
$$\text{Rate} = (\text{Fix1.UTime} - \text{Fix0.UTime}) / (\text{Fix1.LocalTime} - \text{Fix0.LocalTime})$$
- The GPS pair time interval is interrogated for a data gap or anomalously short block (see below)
- The timed Event list is then interrogated for all events with local time \leq Fix1 local time. Any events found are sent for export (see below) and then removed from the event list
- If the GPSEvent pair are the last two in the GPSEvent list then any remaining events, with local times $>$ Fix 1 local time are also sent for export and removed from the event list

If the event list still holds Events at the end of the above processing then the GPSEvent list contained less than 2 events and a GPSEvent pair could not be formed. In this case the Events will not be processed or exported and will be removed from the Event list.

Data Gaps and Anomalously Short Blocks

Occasionally, for reasons that are not completely understood, the time interval between GPS fixes cannot be fully accounted for by the total time acquiring data samples (taking into account the difference in local and GPS clock rates). In this situation the data within the time interval is considered to have a data gap error.

To detect a data gap, the current interval's clock rate is compared to the clock rate from previous intervals not affected by a data gap. Several previous clock rates, determined by the *RateAverage* processing parameter, are averaged for the comparison. If the rate change is greater than twice that suggested by the uncertainty calculation then a data gap has been detected. More formally, if the ratio:

$$((R1 - R2) * dLT) / (0.2 + (dUT / 5400))$$

where,

R1 = the current interval clock rate

R2 = the averaged rate from previous good intervals

dLT = the current interval in local time units

dUT = the current interval in GPS UT time units

is greater than 2, then the current interval has a data gap error (uncertainty in time) of magnitude: $((R1 - R2) * dLT)$.

Likewise, if the rate change has decreased by an amount greater than twice that suggested by the uncertainty calculation, i.e. the ratio above is less than -2 , then the block is marked as anomalously short. The uncertainty in time is calculated as above.

Event Export

The details of a Event's export procedure depends upon the Event type. However, a common stage for all Events is the computation of the Event's UT time by interpolation of the UT times from its associated GPSEvent pair.

The Event processing algorithm is constructed such that Events with local times within the range of a GPSEvent pair will be processed by interpolation but Events outside the GPSEvent local time range will be processed by extrapolation from the two nearest GPSEvents.

The event UT time is calculated via linear interpolation from GPS Fix0 using the rate derived in the processing stage:

$$\text{Event.UTime} = (\text{Event.localTime} - \text{Fix0.LocalTime}) * \text{Rate} + \text{Fix0.UTime}$$

The event UT Time is then un-encoded from scaled seconds back to a standard date and time format.

The Event export procedure, by Event type, is detailed below:

QCEvents

A readout displaying the Event's TICKS value, local time and calculated UT time of the form shown below is presented to the user.

```
Pre Normal TKS= 426, Local= 591097, UT= ( 3:24:12.5 14/12/2001)
```

HKEvents

A delimited string, of the form shown below, containing the Event UT time and house keeping data is written to the house keeping text file. The delimiting character is provided by the *TXTDelimiter* processing parameter.

```
2001, 12, 13, 347, 18, 42, 11.4, 751, 547, 525, 516, 568, 499, 2, 3, 2, 2, 386
```

DataEvents

Firstly, the DataHeader structure, containing the block type (Fixing or Normal), the number of samples and the sample interval is exported.

All samples within the DataEvent are then exported in the Sample structure given above. Samples are stored as two byte fields, the bytes may be swapped at this stage, via the *SwapBytes* processing parameter, to resolve big/small endian differences between the acquisition and processing environments.

A time uncertainty is calculated for each sample, this will simply be the gap magnitude if the sample is affected by a data gap error or the value from the uncertainty formula:

$$\text{Uncertainty} = 0.2 + ((\text{time from nearest GPS fix}) / 5400)$$

To process all data samples in a DataEvent, the first sample local time is taken to be the event local time. The local time for subsequent data samples is simply an increment of the sample interval. Given the local time, the UT time is interpolated as detailed above.

The DataEvent structure contains a file offset to the first sample. To access the sample values the input file pointer is relocated to the first sample offset and incremented by the sample size for each subsequent sample.

The processing statistics are updated and interval sample variance values are calculated at this point.

Event Manager User Messages

The Event Manager presents various messages throughout processing to notify the user of progress and to assist in error identification. Some examples are:

```
EM: DAC Event received [162, 129, 45]
EM: Exporting data from Event queue...
EM: 0 GPS fixes, 0 Events
EM: No DAC values, Events not exported
EM: Export complete, 0 Event(s) not processed

EM: Data Event received

EM: Skip Block Event received
EM: Exporting data from Event queue...
EM: 8091 GPS fixes, 24273 Events
EM: ( 0:44:31.2 15- 1-2003) - ( 1:49:13.2 15- 1-2003) dG/dL = 0.993537 Gap = 0
EM: ( 1:49:13.2 15- 1-2003) - ( 2:53:55.3 15- 1-2003) dG/dL = 0.993535 Gap = 0
EM: ( 2:53:55.3 15- 1-2003) - ( 3:58:37.3 15- 1-2003) dG/dL = 0.993496 Gap = 0
EM: ( 3:58:37.3 15- 1-2003) - ( 5: 3:19.3 15- 1-2003) dG/dL = 0.993496 Gap = 0
EM: ( 5: 3:19.3 15- 1-2003) - ( 6: 8: 1.3 15- 1-2003) dG/dL = 0.993510 Gap = 0
EM: Export complete, 0 Event(s) not processed

EM: EOF Event received
EM: Exporting data from Event queue...
EM: 0 GPS fixes, 0 Events
EM: Export complete, 0 Event(s) not processed
```

QC Reports

Degum1 has three processing parameters that control the output of additional QC reports: *GPSQC*, *BlockQC* and *TimingQC*.

GPSQC Report

If enabled, all GPS NMEA sentences encountered during processing will be displayed to the user. For example:

```
$GPZDA,021258.2,16,01,2002,,*5E
$GPGGA,021258.0,8421.235,S,02351.478,W,1,06,1.24,02073,M,-020,M,,*72
```

BlockQC Report

If enabled, all input file text lines encountered during processing will be displayed to the user. For example:

```
EM: Data Block Event received
START 16.01.02 04.22.07 138
B 688 #0 536 #1 526 #2 514 #3 580 #4 512 #5 2 #6 2 #7 1
WV 1
SV 308
Interval is 1
Fixing G P S loops 300
Normal loops 3600
171
150
GPS 16.01.02 04.27.08 151
$GPGGA,042713.0,8421.236,S,02351.485,W,1,08,1.01,02072,M,-020,M,,*74
$GPZDA,042714.3,16,01,2002,,*57
GPS 16.01.02 04.27.09 169
190
168
END 16.01.02 05.27.08 168
DB (10%): Int=1, FLoop=300, NLoop=3600, GPS Fix
```

TimingQC Report

If enabled, a time calculation report will be displayed to the user for every line in the input file that contains a TICKS value. For example:

```
EM: ( 1:14:57.3 14-12-2001) - ( 2:19:34.3 14-12-2001)  dG/dL = 0.993559 Gap = 0
  Post GPS TKS= 375, Local= 517916, UT= ( 2:19:34.5 14-12-2001)
  Pre Normal TKS= 396, Local= 517937, UT= ( 2:19:35.6 14-12-2001)
Post Normal TKS= 375, Local= 585418, UT= ( 3:19:11.5 14-12-2001)
  END TKS= 375, Local= 585418, UT= ( 3:19:11.5 14-12-2001)
  START TKS= 382, Local= 585425, UT= ( 3:19:11.9 14-12-2001)
  Pre Fixing TKS= 415, Local= 585458, UT= ( 3:19:13.6 14-12-2001)
Post Fixing TKS= 393, Local= 591064, UT= ( 3:24:10.7 14-12-2001)
  Pre GPS TKS= 407, Local= 591078, UT= ( 3:24:11.5 14-12-2001)
```

For V3 LPM files, in addition to the above report, extra TimingQC report lines will be produced. The report uses the extra TICKS information for each sample to test the inter-sample interval. In data with no data gap error the expected sampling interval will be stored as one of two TICKS values. For example a 1 second sample interval will have a TICKS value of 18 or 19 (1 second = 1125 / 60 TICKS = 18.75). The QC report will provide a count of samples at the expected intervals and list samples outside of the expected interval:

```
EM: ( 7:12:53.3 25- 2-2005) - (11:13: 4.3 25- 2-2005)  dG/dL = 1.000545 Gap = 100
  Post GPS TKS= 162, Local= 83725497, UT= (11:13: 4.5 25- 2-2005)
  Pre Normal TKS= 171, Local= 83725506, UT= (11:13: 4.9 25- 2-2005)
Out of expected range: Sample = 13691, TICKS = 769, Sec = 41.0133
# as expected = 13799 : 18's = 3450, 19's = 10349
Post Normal TKS= 900, Local= 83984237, UT= (15: 3:11.5 25- 2-2005)
  END TKS= 900, Local= 83984237, UT= (15: 3:11.5 25- 2-2005)
  START TKS= 920, Local= 83984257, UT= (15: 3:12.5 25- 2-2005)
  Pre Fixing TKS= 940, Local= 83984277, UT= (15: 3:13.6 25- 2-2005)
# as expected = 600 : 18's = 151, 19's = 449
Post Fixing TKS= 918, Local= 83995508, UT= (15:13:12.9 25- 2-2005)
  Pre GPS TKS= 928, Local= 83995518, UT= (15:13:13.4 25- 2-2005)
```

Degum2

Overview

The Degum2 application's primary task is to apply cleaning algorithms to the input file to remove known instrumental artefacts that may be present in the data. The three effects that are corrected for are ADC zero values, spikes and data shifts due to GPS operation. If a sample is modified, a record of the data corrections applied and the algorithms used are stored in the output sample. An optional QC step can be requested to examine the data for further unexpected ADC values.

Instrumental artefacts

ADC Zero Values

The raw magnetometer data is comprised of three, X, Y and Z, component values, known as ADC values. Sometimes an ADC value for a particular sample is erroneously zero (or close to zero). To date this effect has only been observed on samples at specific indices within a block of input data - indices 13690, 13694, 18759, 18994 and 18999. If one of these samples contains an incorrect ADC value then the subsequent sample may also have an erroneous value - though not to the same extent.

Data Spikes

Sometimes an ADC value for a particular sample has an erroneous value - but one which is not close to zero. It appears as a spike in the data record for one or more components. Similar to the ADC zero values, this effect usually occurs at specific samples, 18754 for example.

Data Shift due to GPS Operation

The periodic operation of the GPS receiver to obtain a timing fix produces errors in the recorded magnetometer data. When the GPS is switched on, a spike is introduced into the next recorded sample and all subsequent samples, while the GPS is on, are shifted from their true values. When the GPS is switched off, again, the next recorded sample contains a spike but subsequent samples are now recorded at their true values.

Cleaning Algorithms

ADC Zero Values

It is unhelpful to test all input samples for ADC zero values because a zero value in itself may not be erroneous – it may be the true detected value. Hence the algorithm only examines specific samples for ADC zero values. The list of sample indices to test can be supplied by the user via the *ADCZero* processing parameter or allowed to take the single default value of 18759.

All components, X, Y and Z, of the test sample are examined and if any is within the range -1 to 1 inclusive the sample will be considered to have an ADC zero error and all components will be replaced. Its replacement value is simply the average of the data components from two samples earlier and two samples later.

Once a sample with an ADC zero error is detected, it is assumed that the next sample is also erroneous and is replaced in a similar manner, that is, its value will be the average from two samples earlier and two samples later.

So, for example, sample 18759 is to be replaced then the average of samples 18757 and 18761 will be used and sample 18760 will be replaced with average of samples 18758 and 18762.

Data Spikes

If a dataset contains spiked samples then a list of sample indices to interpolate can be supplied by the user via the *Despike* processing parameter (there is no default)

All components, X, Y and Z, of the spiked sample within all data blocks are replaced with the average of the same component value from the two neighbouring samples.

For example, if sample 18754 is identified as a spiked sample, it will be replaced by the average of samples 18753 and 18755.

Data Shift due to GPS Operation

To remove the data shift on samples acquired during GPS operation, the fixing samples, one must first derive acceptable data values for the two spiked samples – the samples recorded immediately after the GPS was switched on (S1) and immediately after it was switched off (S2). Two values for each sample are derived by linear extrapolation of the data before and after the spiked sample (E1a, E1b, E2a, E2b). The extrapolation is achieved via a linear fit of 60 seconds of data – though limited to no more than 20 samples and no less than 2 – as shown below:

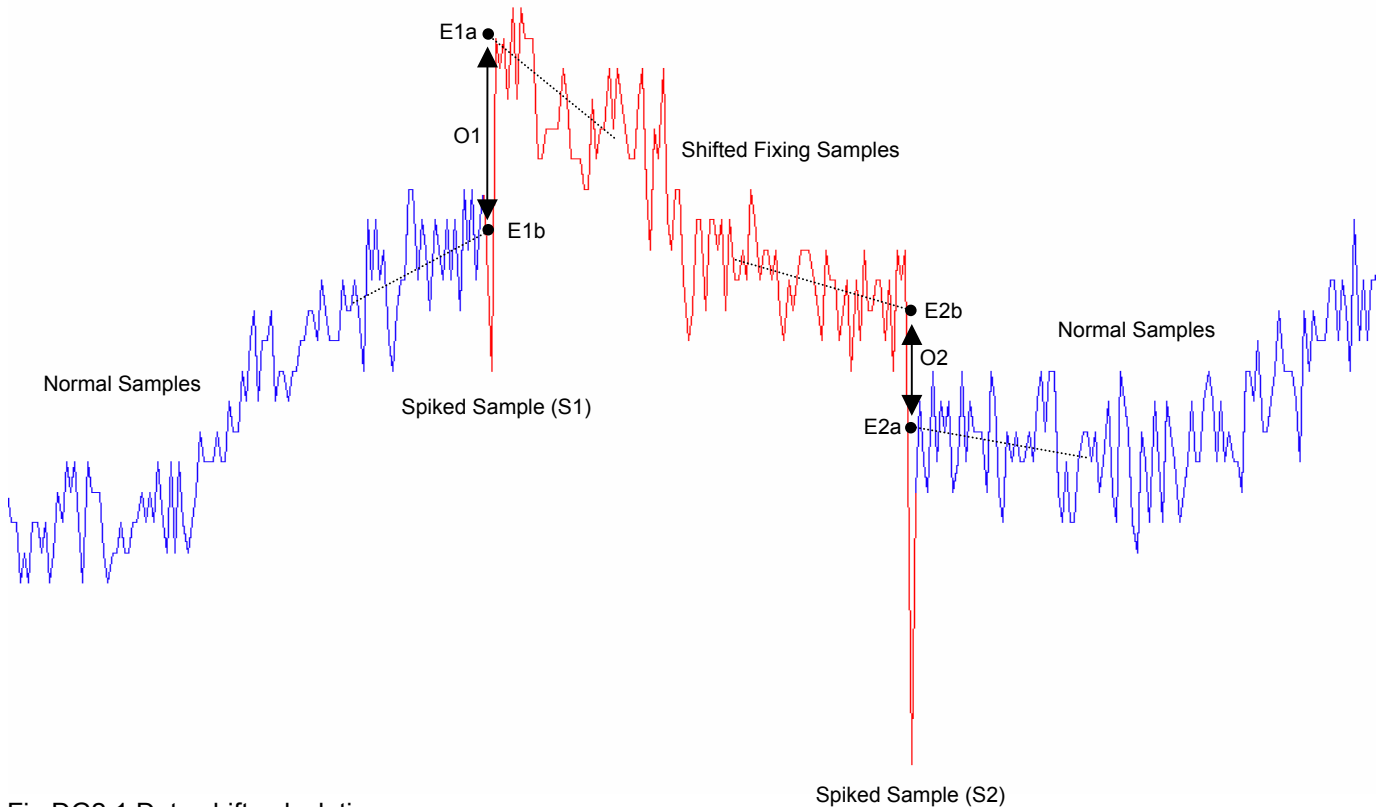


Fig DG2.1 Data shift calculation

Data values at the spiked samples are simply replaced with the values extrapolated from the normal samples (S1 = E1b, S2 = E2a). The remaining fixing samples are shifted back to their normal position by the linear interpolation of the difference between the extrapolated values at both ends. That is, the fixing sample after S1 will be shifted by the offset O1 ($E1b - E1a$), the fixing sample before S2 will be shifted by the offset O2 ($E2a - E2b$) and the samples in between will be shifted by an offset linearly interpolated from O1 to O2.

Processing Overview

Degum1 data is read in block pairs, corresponding to a Fixing block and a Normal block. An optional ADC zero QC (see below) is performed if requested. Each pair is tested for ADC zero values and fixed if required. The data shift in the fixing block sample is removed and the data for the two spiked samples replaced. A record of the data modifications is kept and stored in the sample's *Delta* and *Status* fields. Processing statistics are gathered and the data is written to output. Flow continues until all block pairs have been processed.

Files and Formats

The input .dg1 files are arranged in block structures each with a block header containing the block type, number of samples and the sampling interval. This structure makes it easy to buffer block pairs and to test for expected block types. The four possible block types are *DAC*, *Fixing*, *Normal* and *Information*.

Degum2 writes a single data file and an optional QC file.

Degum2 data file

The main output file, .dg2, is a binary file. It contains a record of the processing history followed by a single array of all the data samples. The samples are not exported in blocks and there are no header structures. The last sample exported is not a data sample but an information sample that contains the total number of data samples exported. The full .dg2 format description is given here:

Structure	Variable	Size(bytes)	Description
.dg2	History Sample(1) Sample(2) Sample(3)...Sample(N) Sample(N+1)	[Variable]	Degum2 output. Last data sample Information sample
History	NumStrings String(1)\n String(2)\n... String(NumStrings)\n	4 [Variable]	Number of history strings History character string terminated by '\n' character
Sample	Time	8	UT Time encoded to seconds * 10
	Uncertainty	2	Uncertainty in time (seconds * 10)
	Status	2	Status flags
	Data	6	XYZ data components. 2 bytes each in ADC Units
	Delta	6	Accumulated changes. 2 bytes for each component in ADC units

Table DG2.1 .dg2 file format description

The sample *Status* variable is passed from input to output with four possible modifications:

Bit	Description
2	Set if sample is a GPS transition sample (spiked sample – S1 or S2)
3	Set if sample required ADC zero value fix
4	Set if sample's data was interpolated via the <i>Despike</i> option
9	Set if the accumulated changes stored in the <i>Delta</i> variable overflow 16 bits

Table DG2.2 Sample status flag bits

ADC Zero QC file

The ADC zero QC file is a text file. Each row contains 6 fields of QC information for a single output sample:

Field	Description
1	The sample index (relative to the start of the file)
2	The sample index (relative to the start of the data block)
3	Date/Time string of format (<i>HH:MM:SS DD-MM-YY</i>)
4	X Component in DAC units
5	Y Component in DAC units
6	Z Component in DAC units

Table DG2.3 The ADC QC file format

A further description of the ADC zero QC process is given in the *Processing Algorithm* section below.

Processing Algorithm

Initialisation

The following steps initialise the application in preparation for data processing:

- Command line arguments are validated
- The INI file, if supplied, is read and validated
- The input file name and location are validated
- A DG1File object is created and processing is initiated

Processing

The Processing History data block is read from the input file, the current Degum2 processing parameters are appended and the new history block is written to output.

Data is then read in a block-by-block basis and acted upon according to the block type as detailed below. If the block type is not recognised then an error is thrown. The Fixing Block is presented first as it initiates the primary data processing algorithm:

Fixing Block

- All block samples are read into memory and the ADC zero QC is performed if requested (see below)
- The header for the next block is read and its samples are read and stored in a second memory buffer.

A check is performed to establish that this block is indeed a Normal block as expected. If not, an error message is issued to the user, the data is bypassed and flow returns to the next block read.

- The ADC Zero value and Despiking cleaning algorithms are implemented for both blocks in memory
- Extrapolated values: $E1a$, $E2a$ and $E2b$ are extracted from the data and offset $O2$ is calculated:

$$O2 = E2a - E2b$$

- If this is not the first block pair to be processed then the extrapolation coefficients required to derive the value $E1b$ will have been saved from the previous block's processing and are used to calculate here. If this is the first block pair then $E1b$ is calculated from $E1a$ and $O2$:

$$E1b = E1a + O2$$

- Offset $O1$ is calculated:

$$O1 = E1b - E1a$$

and the interpolation coefficients, $O1$ to $O2$, are derived

- The data for the first Fixing sample, $S1$, is replaced with the value of $E1b$ and the remaining samples are modified by the interpolated offset value. The applied modifications are stored and each sample's *Status* field modified accordingly
- The Fixing samples are exported to output
- The data for the first Normal sample, $S2$, is replaced with the value $E2a$ and the modification stored. Each sample's status field is modified as necessary
- The Normal samples are exported to output
- The extrapolation coefficients for the tail of the Normal samples are calculated and stored for the next block pair's processing
- Processing statistics are updated

DAC Block

Any DAC values received will be required in subsequent processing as parameters in the ADC units to nT conversion. They are exported to output as a special sample in which its *Time* field is set to -1, its *Status* field is set to zero and the three DAC values are stored in the sample's *Data* fields.

Normal Block

A Normal block should not be read during the main processing loop because all Normal blocks should be preceded by a Fixing block and should be read during the standard Fixing block processing routine. Hence if a Normal block is read at this stage an error message is posted to the user and the block is bypassed from further processing.

Information Block

An Information block signifies the end of the input file and the completion of processing. The information, counts for Fixing and Normal blocks and total number of samples, are read and compared to the corresponding totals accumulated during Degum2 processing. If they do not match then samples have been lost during Degum2 processing and a warning message is issued to the user.

A report of additional statistics (see below), also accumulated during processing, are presented to the user.

The total number of samples processed in Degum2 is stored in a special Information sample, in which its *Time* field is set to -1, its *Status* field is set to 1 and the sample count is stored in two of the sample's *Data* fields, and exported to the end of the output file.

User Messages

Errors and Warnings

The following error and warning messages may be presented to the user during Degum2 processing:

```
DG2: DAC Values: 97, 126, 43
DG2(100%): Blocks processed: 1 DAC, 8091 F/N pairs, 0 Normal, 1 Info
```

```
DG2: Error processing DAC block
DG2: Error processing Fixing block
DG2: Error processing Normal block
DG2: Error processing Info block
```

```
DG2: Error, Fixing block not followed by a Normal block. Skipping data
DG2: Error, Unexpected Normal block. Skipping data
```

Statistics

At the end of processing the user is presented with input and output block and sample counts:

```
DG1 File: # Fixing = 8091, # Normal = 8091, # Samples = 22257250
DG2 O/P : # Fixing = 8091, # Normal = 8091, # Samples = 22257250
```

If there is a mismatch a warning message will also be produced:

```
DG2: # Fixing blocks not as expected
```

Statistics are also presented for each sample interval processed:

```
Samples with 1 second interval:
# Samples = 17616300
# Samples with gap = 980100
# Samples without gap = 16636200
# Samples with ADC Zero Fix = 0
# Samples with Despiking Fix = 0
# Samples with GPS Transition Fix = 9034
```

```
Samples with 3 second interval:
# Samples = 4639700
# Samples with gap = 4639300
# Samples without gap = 400
# Samples with ADC Zero Fix = 0
# Samples with Despiking Fix = 0
# Samples with GPS Transition Fix = 7138
```

```
Samples with 60 second interval:
# Samples = 1250
# Samples with gap = 1250
# Samples without gap = 0
# Samples with ADC Zero Fix = 0
# Samples with Despiking Fix = 0
# Samples with GPS Transition Fix = 10
```

ADC Zero QC Report

The Degum2 processing parameters: *ADCQC*, *ADCQCMin*, *ADCQCMax* and *ADCQCRange* control the output of an optional ADC zero QC report. If the *ADCQC* parameter is set to *True* then if any data component of a sample is within the range *ADCQCMin* to *ADCQCMax* inclusive then information, as detailed above, for that sample and *ADCQCRange* samples before and after it will be exported to the QC file. For example if the min, max and range values were set to -10, 10 and 3 respectively, the following may be seen:

```
457 157 (18:49:50.1 13-12-2001) 189 221 1142
458 158 (18:49:51.1 13-12-2001) 187 222 1142
459 159 (18:49:52.1 13-12-2001) 188 221 1140
460 160 (18:49:53.1 13-12-2001) 188 -2 1142
461 161 (18:49:54.1 13-12-2001) 189 222 1141
462 162 (18:49:55.0 13-12-2001) 187 222 1141
463 163 (18:49:56.0 13-12-2001) 188 221 1141
```

Degum3

Overview

The Degum3 application's sole task is to scan the whole input data set over a series of time windows to determine the most inactive period in the data – the quiet period. Once the quiet period has been determined it is assumed that the horizontal vector formed by the average of the X and Y data components within the quiet period window is a good estimate of the orientation of the instrument. The vector is used to calculate the rotation angle required to rotate the data to the standard reference frame.

Only the rotation angle is calculated and reported to the user, the data is not rotated and no file is exported.

Quiet Period Determination

The variance of each data component, converted to nT units, for all samples within a sliding time window are calculated for the whole data set. The time period of the window with the smallest X and Y variance product is the required quiet period. The window length used is given by the processing parameter *WindowLength* and the move-up of the window between successive calculations is specified by the *Moveup* processing parameter – both supplied in seconds.

For full details of ADC units to nT conversion, refer to the Degum4 documentation.

Rotation Angle Calculation

Taking the vector formed by the average X and Y data components to be (X, Y), the rotation angle, to rotate the data to the standard reference frame, is calculated by computing the arc tangent of the ratio Y / X. Care must be taken to return the angle in the correct quadrant.

The rotation derived is a counter-clockwise rotation.

Processing Algorithm

Initialisation

The following steps initialise the application in preparation for data processing:

- Command line arguments are validated
- The INI file, if supplied, is read and validated
- The input file name and location are validated
- A DG2File object is created and processing is initiated

Processing

The Processing History data block is read from the input file, the current Degum3 processing parameters are appended and, if the *ShowHistory* processing parameter is enabled, the new history block is presented to the user.

The processing window period is initialised to the time of the first data sample and the following algorithm is implemented until all complete windows have been processed:

- Read and store all samples within the processing window period. If a special DAC sample is encountered, extract the DAC values and store separately
- Convert the sample data to nT units – using the stored DAC values
- Compute the variance for each data component:

$$\text{Variance}(x) = 1 / N * \Sigma x^2 - 1 / N^2 * (\Sigma x)^2$$

Where N is the number of samples in the processing window

- If the product of the X and Y variance is the minimum variance computed so far, update the stored minimum variance values and period
- Advance the processing window period by *Moveup* seconds

At the end of the input file the *Info* sample will be read, the number of samples is extracted and compared to the actual number of samples read. A mismatch will throw a warning to the user.

Once the minimum variance period for the whole file has been determined, calculate the average X and Y values over that period, AvgX and AvgY. The required rotation angle can then be computed using:

$$\text{Rotation Angle} = \arctan (\text{AvgY} / \text{AvgX})$$

User Messages

During the quiet period determination, Degum3 will periodically report the number of windows scanned and the current time period with the minimum variance:

```
DG3 (5%): 99 Windows scanned. Min Var at: (1:47:1.8 14-12-2001) - (2:47:9.8 14-12-2001)
```

When the *Info* sample is read, a report of the expected and processed sample counts is made. A warning message is displayed if there is a mismatch:

```
DG2 File: # Samples = 22257250  
DG3 O/P : # Samples = 22257244
```

```
DG3: # Samples not as expected
```

Finally, a report of the quiet period information statistics and computed rotation angle, in radians and degrees, is made:

```
DG3: Quiet period information:  
DG3: Window range = ( 1:47:1.8 14-12-2001) - ( 2:47:9.8 14-12-2001)  
DG3: Average X = 18495.9, Average Y = 340.483, Average Z = -46240.7  
DG3: Variance X = 3.68514, Variance Y = 3.86874, Variance Z = 10.3505  
  
DG3: CCW Rotation Angle = 0.266422(rad), 15.2649(deg)
```

Degum4

Overview

Degum4 is the final processing application in the Degum suite. Its tasks are to convert the data to nT units, rotate it to the standard reference frame and write it in the final output format. The rotation angle is passed as a command line parameter.

Conversion to nT units

The following equation is used to convert a component (X, Y or Z) of the raw LPM magnetometer readings (ADC units) to nT:

$$nT_x = ((DAC_x - K1) * (K2 / K1)) - (ADC_x * K3)$$

Where K1, K2 and K3 are constants supplied via the *DAC2nT1*, *DAC2nT2* and *ADC2nT* processing parameters respectively.

DAC_x is the latest DAC value for the component and ADC_x is the ADC value.

The default constant values are:

$$DAC2nT1 = 128.0, \quad DAC2nT2 = 70000.0, \quad ADC2nT = 0.77213$$

Rotation to the Standard Reference Frame

The following equations are used to apply a counter-clockwise rotation to the X and Y components of the data to convert them to the standard reference frame components H and D:

$$H = (X * \cos(\bullet)) + (Y * \sin(\bullet))$$

$$D = (Y * \cos(\bullet)) - (X * \sin(\bullet))$$

Where \bullet is the rotation angle.

Output File Format

The output, .dg4, file is a binary file. It contains a record of the processing history followed by a single array of all the data samples. The data is stored in nT units hence each component is allocated 4 bytes. The data is stored to an accuracy of 0.1 nT but as integers so they are scaled accordingly ($* 10$). There is no information sample at the end of the file. The full .dg4 format description is given here:

Structure	Variable	Size(bytes)	Description
.dg4	History Sample(1)...Sample(N)	[Variable]	Degum4 output.
History	NumStrings String(1)\n String(2)\n ... String(NumStrings)\n	4 [Variable]	Number of history strings History character string terminated by '\n' character
Sample	Time Uncertainty Status Data Delta	8 2 2 12 6	UT Time encoded to seconds * 10 Uncertainty in time (seconds * 10) Status flags XYZ components of magnetometer data. 4 bytes each in (nT * 10) Units Accumulated changes. 2 bytes for each component each in (nT * 10) Units

Table DG4.1 .dg4 file format description

Degum4 does not add any status bits but may modify bit-9 if the accumulated changes overflow 16 bits during the rotation and conversion to nT.

Processing Algorithm

Initialisation

The following steps initialise the application in preparation for data processing:

- Command line arguments are validated
- The INI file, if supplied, is read and validated
- The input file name and location are validated
- A DG2File object is created and processing is initiated

Processing

The Processing History data block is read from the input file, the current Degum4 processing parameters are appended and the new history block is written to output.

The file is processed on a sample-by-sample basis and the following algorithm is implemented until all input samples have been read:

- Proceed according to sample type:
- If the sample is a *DAC* sample, extract the DAC values and store
- If the sample is a data sample,
 - Convert all components of the data and *Deltas* to nT
 - Rotate the X and Y components of the data and Deltas to the standard reference frame
 - Write the sample to output
 - Update processing statistics
- If the sample is an *Info* sample, it signifies the end of the input file. The number of samples is extracted and compared to the actual number of samples read. A mismatch will throw a warning to the user.

User Messages

Periodically during processing, Degum4 will report the number of samples processed:

```
DG4(42%): 9236733 Samples processed
```

When the info sample is read, a report of the expected and processed sample counts is made. A warning message is displayed if there is a mismatch:

```
DG2 File: # Samples = 22257250  
DG4 O/P : # Samples = 22257244
```

```
DG4: # Samples not as expected
```

Finally, a report of the statistics, the average and variance of the data, gathered during processing is made:

```
DG4: Average X = 17749.5, Average Y = 15.1973, Average Z = -47485.7  
DG4: Variance X = 9562.08, Variance Y = 2445.08, Variance Z = 7181.76
```

DGRange

Overview

DGRange is a non-processing utility application that allows the user to extract whole-day portions of data from a Degum file and export it to a new file. Degum1, Degum2 or Degum4 data may be used and the output can be in binary or text format. It is purely command line driven – no INI file is used.

For text output files, the field delimiting character can be specified and the export of the file processing history is optional.

The utility can also be used to just report the time range and/or processing history of a supplied Degum file.

Command Line Parameters

DGRange usage is:

```
DGRange [-t['ch']] [-h['ch']] [-s] [-pOutput Path] [-rDay Month Year Day Month Year]] DGFile
```

where *DGFile* is the name of the input file.

The options are:

- t ['ch'] Export in text format.
The number and content of fields will vary with the type of input file (see below)
The optional *ch* character will be used as the field delimiter. Default is ' ' (space)
- h ['ch'] Export processing history information.
This option only applies if exporting in text format or reporting the input file range
The optional *ch* character will be used as the comment character. Default is ' ' (space)
- s Split a multi-day export range into separate output files.
- pOutput Path Supply alternative path for the output file. Output files will be placed in the input directory if this option is not supplied. Do not use the '~' character for /home paths.
- rDay Month Year Day Month Year The range of data to export. If not specified, the whole input file will be exported.
First *Day*, *Month* and *Year* (DMY) specify the start date, the second triplet specify the end date, both inclusive
Only whole days can be exported. Duplicate DMY triplets will export a single day's data
DMY parameters are numerical and integers. Years are 4 digit.
All 6 DMY values must be specified

A command with no parameters, except the input file name, will report the range of the input file only.

If the -t option is used without the -r option then the whole input file will be exported as a text file.

For some parameters, such as single characters or path names containing spaces, it may be necessary to wrap the string or character in quotation characters. Single or double quotes should be used – depending on operating system – but in general, use single quotes for linux/unix systems and double quotes for Windows/DOS systems.

Some examples:

DGRange -h M66-294.dg1	Reports the range of file <i>M66-294.dg1</i> and displays its processing history
DGRange -t -r14 3 2003 15 3 2003 M84-336.dg4	Exports data from 14/3/03 to 15/3/03 of <i>M84-336.dg4</i> file in text format with space delimiter and no processing history
DGRange -r2 8 2005 11 8 2005 M83-348.dg3	Exports data from 2/8/05 to 11/8/05 of <i>M83-348.dg3</i> file in binary format
DGRange -t, -h; -r1 6 2002 1 6 2002 M78-337.dg4	Exports a comma delimited text file with colon commented processing history for data from the single day 1/6/02 of file <i>M78-337.dg4</i>
DGRange -t -s -r1 6 2002 3 6 2002 M78-337.dg4	Exports, to separate text files, data from each day between 1/6/02 and 3/6/02 of file <i>M78-337.dg4</i>
DGRange -t -p/home/lpm -r1 6 2002 3 6 2002 M78-337.dg4	Exports data from 1/6/2002 to 3/6/2002 of <i>M78-337.dg4</i> in text format. The output is directed to the folder: <i>/home/lpm</i>
DGRange -t -p"C:\lpm data" -r1 6 2002 3 6 2002 M78-337.dg4	Exports data from 1/6/2002 to 3/6/2002 of <i>M78-337.dg4</i> in text format. The output is directed to the folder: <i>C:\lpm data</i>

Output File Name

The name of the output file from DGRange is not user supplied, but derived from within the application. The actual name used is always the name of the input file concatenated with a multi-component suffix that is dependant upon the output options selected. Table DGR.1 details the component of the suffix associated with each DGRange command line option.

Option	Output File Suffix Component
-t	.txt
-r	.DMY-DMY where DMY are the supplied range triplets (each 6 digit)
-s	.Y.D where Y is the two digit year and D is the three digit day number

Table DGR.1 DGRange output file name suffix componens by option

Some examples:

Command Line	Output File Name(s)
DGRange -t M66-294.dg4	M66-294.dg4.txt
DGRange -t -r1 6 2002 3 6 2002 M66-294.dg4	M66-294.dg4.010602-030602.txt
DGRange -r1 6 2002 3 6 2002 M66-294.dg4	M66-294.dg4.010602-030602
DGRange -t -s -r1 6 2002 3 6 2002 M66-294.dg4	M66-294.dg4.152.txt M66-294.dg4.153.txt M66-294.dg4.154.txt
DGRange -s -r1 6 2002 3 6 2002 M66-294.dg4	M66-294.dg4.152 M66-294.dg4.153 M66-294.dg4.154

Output File Format

Binary

If binary output format is specified, the extracted data and processing history will be of the same binary format as the input file. That is, either *.dg1*, *.dg2* or *.dg4* format.

Text

The text output format will contain, if requested, the processing history as a series of character strings each starting with the comment character, if specified, followed by multiple rows of data fields – one row per sample. The number and content of each field will depend upon the type of input file and is summarised in table DGR.2 below. However, the first ten fields are common to all text output files and are described separately:

Input File type	Field #	Description
All	1	UT Time (seconds)
	2	Seconds since start of first day in file
	3	Year
	4	Month
	5	Day
	6	Hour
	7	Minute
	8	Second
	9	Uncertainty in time (seconds)
	10	Status
Degum1 (.dg1)	11	Data X component (ADC units)
	12	Data Y component (ADC units)
	13	Data Z component (ADC units)
Degum2 (.dg2)	11	Input data X component (ADC units)
	12	Input data Y component (ADC units)
	13	Input data Z component (ADC units)
	14	Delta X (ADC units)
	15	Delta Y (ADC units)
	16	Delta Z (ADC units)
	17	Output data X component (ADC units)
	18	Output data Y component (ADC units)
	19	Output data Z component (ADC units)
Degum4 (.dg4)	11	Data X component (nT units)
	12	Data Y component (nT units)
	13	Data Z component (nT units)
	14	Delta X (nT units)
	15	Delta Y (nT units)
	16	Delta Z (nT units)

Table DGR.2 DGRRange text output format description

File Feedback

It is possible to apply DGRRange to a file that was itself exported from DGRRange provided it was originally exported as a binary file.

Further Processing

An output file, of any file type, from DGRRange should not be used for further Degum processing – even if it is a binary file. For example, if a portion of a *.dg1* file is exported to a binary file it should not be used as an input to the Degum2 application. This is because not all required file information will be passed from the input to output file. Most notably, DAC values that are present in an input file prior to the export range will not be present in the output.

Processing Algorithm

Initialisation

The following steps initialise the application in preparation for data processing:

- Command line arguments are validated
- The input file name and location are validated
- The input file is scanned for type
 - The file's processing history is read and reverse searched for the string "vDG ="
 - The integer found after the search string if the Degum file type
- A Matching DGXFile object is created and range processing is initiated

Processing

The Processing History data block is read from the input file and the current DGRange application version is appended. If the user has requested a binary output file, then the new history block is written to output. If a text output file was requested and the -h history parameter was supplied the new history block is displayed to the user.

Flow continues on a sample-by-sample basis depending on the four possible types of processing that can be requested by the user:

Report file range

- Read the first and last samples in the file
- Convert internal time to UT Time components
- Report the time of both samples

Output complete input file to text file

- Read a sample
- Convert internal time to UT Time components
- Write in binary format

Output range of data to text or binary file

- Compute internal time values for the requested output time range
- Perform binary search to skip to the first sample within range
- Read remaining samples
- If within range, export in binary or text format. If not, exit

User Messages

If the input file type could not be determined an error report is made:

```
DGRRange: Unable to extract file type
```

Periodically during processing, DGRRange will report on the processing progress:

```
DGRRange: 31% of file processed, sample time = (22: 1: 3.6 13-12-2001)
```

If the input file range is requested:

```
DGRRange: Time range is (18:42:14.9 01-12-2001) - ( 5:28:25.9 14-11-2002)
```

If the input file range is requested with history information:

```
+++++++ Header ++++++
LPM File = m83_348_data.fil
vLPM = 2
vDG = 1.1.0
INI: General / ADC2nT = 0.77213
INI: General / CommentDelimiter = ;
INI: General / DAC2nT1 = 128.0
INI: General / DAC2nT2 = 70000.0
INI: General / ShowHistory = True
INI: General / TXTDelimiter = ,
INI: Degum1 / BlockQC = False
INI: Degum1 / FixingLoopsMax = 600
INI: Degum1 / FixingLoopsMin = 10
INI: Degum1 / GPSQC = False
INI: Degum1 / Intervals = 1,3,5,9,10,15,60
INI: Degum1 / MaxLineLength = 256
INI: Degum1 / NormalLoopsMax = 21300
INI: Degum1 / NormalLoopsMin = 240
INI: Degum1 / NumSatMin = 1
INI: Degum1 / RateAverage = 10
INI: Degum1 / SwapBytes = False
INI: Degum1 / TimingQC = True
vRange = 5.1.0
+++++++
DGRRange: Time range is (18:42:14.9 01-12-2001) - ( 5:28:25.9 14-11-2002)
```

DGMod

Overview

DGMod is a non-processing utility application that allows the user to view or modify any Degum file's header (processing history) or replace the data of a Degum4 file. If the input file is modified, a new binary file will be created. The status byte, bit 5, is set for all replaced data samples.

DGMod is purely command line driven – no INI file is used.

Command Line Parameters

DGMod usage is:

```
DGMod [-h TXTFile] [-d Component D M YYYY D M YYYY Value] DGFile
```

where *DGFile* is the name of the input degum file.

The options are:

- h *TXTFile* Append the text information in file *TXTFile* to the input file's header.
- d Component D M YYYY D M YYYY Value
Replaces the data values of a Degum4 file.
Component specifies which components to replace where 1 = X, 2 = Y and 4 = Z. These can be combined to replace multiple components simultaneously.
D M YYYY D M YYYY are the start and end dates to apply the modification.
Value is the replacement value to use

A command with no parameters, except the input file name, will report the header of the input file only.

Some examples:

- | | |
|---|---|
| DGMod M66-294.dg1 | Displays the processing history of file
<i>M66-294.dg1</i> |
| DGMod -h info.txt M84-336.dg4 | Appends the text contained within the file
<i>info.txt</i> to the header of file <i>M84-336.dg4</i> .
A separate output file is created |
| DGMod -d 5 28 12 2004 31 1 2005 99999.9 M84-336.dg4 | Replaces the X and Z components of the
data within file <i>M84-336.dg4</i> with the
value 99999.9 over the date range
28/12/2004 to 31/01/2005.
A separate output file is created |

Output File Name

The name of the output file from DGMod is not user supplied, but derived from within the application. The file name used is the name of the input file concatenated with the '.mod' suffix.

For example, if the input file was M66-294.dg4 the output, after header modification, would be M66-294.dg4.mod

Output File Format

The output file will be in the same binary format as the input file. That is, either *.dg1*, *.dg2* or *.dg4* format.

Further Processing

An output file, of any file type, from DGMod can be used by any of the Degum applications as normal – it may even be used as an input file to DGMod.

Processing Algorithm

Initialisation

The following steps initialise the application in preparation for data processing:

- Command line arguments are validated
- The file name and location of all input files are validated
- A DGMFile object is created and header processing is initiated

Processing

If the header update, -h, option is supplied:

- The TXTFile is opened and all text lines within are read and appended to the processing history string list
- The updated file history is exported to the output file
- The remainder of the input degum file is copied to the output file as an exact byte for byte copy

If the data replace, -d, option is supplied:

- The file history is exported to the output file.
- The input file is read, by sample, and if the sample time falls within the replacement range, the data for the specified components are replaced with the replacement value.
- The sample is exported to the output file.

User Messages

The input file header is always displayed. If the header was modified during processing, then the updated header is also displayed:

```
+++++++ Header ++++++
LPM File = ../M67-292/raw/2002/m67-292_data1.fil
vLPM = 2
vDG = 1.1.0
INI: General / ADC2nT = 0.77213
INI: General / CommentDelimiter = ;
INI: General / DAC2nT1 = 128.0
INI: General / DAC2nT2 = 70000.0
INI: General / ShowHistory = True
INI: General / TXTDelimiter = ,
INI: Degum1 / BlockQC = False
INI: Degum1 / FixingLoopsMax = 600
INI: Degum1 / FixingLoopsMin = 10
INI: Degum1 / GPSQC = False
INI: Degum1 / Intervals = 1,3,5,9,10,15,60
INI: Degum1 / MaxLineLength = 256
INI: Degum1 / NormalLoopsMax = 21300
INI: Degum1 / NormalLoopsMin = 240
INI: Degum1 / NumSatMin = 1
INI: Degum1 / RateAverage = 10
INI: Degum1 / SwapBytes = False
INI: Degum1 / TimingQC = False
+++++++
```

```
+++++++ Header ++++++
LPM File = ../M67-292/raw/2002/m67-292_data1.fil
vLPM = 2
vDG = 1.1.0
INI: General / ADC2nT = 0.77213
INI: General / CommentDelimiter = ;
INI: General / DAC2nT1 = 128.0
INI: General / DAC2nT2 = 70000.0
INI: General / ShowHistory = True
INI: General / TXTDelimiter = ,
INI: Degum1 / BlockQC = False
INI: Degum1 / FixingLoopsMax = 600
INI: Degum1 / FixingLoopsMin = 10
INI: Degum1 / GPSQC = False
INI: Degum1 / Intervals = 1,3,5,9,10,15,60
INI: Degum1 / MaxLineLength = 256
INI: Degum1 / NormalLoopsMax = 21300
INI: Degum1 / NormalLoopsMin = 240
INI: Degum1 / NumSatMin = 1
INI: Degum1 / RateAverage = 10
INI: Degum1 / SwapBytes = False
INI: Degum1 / TimingQC = False
vHeader = 6.1.0
This is the header update
+++++++
```

Appendix A

Status Bits

The two status bytes store additional information about a data sample in 16 bit fields:

Bit	Description
0	Sample acquired during a Normal loop
1	Sample acquired during a GPS Fixing loop
2	A GPS transition sample (first sample acquired after GPS switched on or off)
3	Sample required an ADC Zero fix during processing
4	Sample required a Despiking fix during processing
5	One or more components have been replaced post-processing
6	Sample is within an anomalously short block
7	Sample is within a block affected by a data gap
8	Set if the uncertainty in time overflows 16 bits
9	Set if the accumulated changes overflow 16 bits
10	Unassigned
11	Unassigned
12	Unassigned
13	Unassigned
14	Unassigned
15	Unassigned

Table A.1 Status bit description