



SUNTECH APP IMPLEMENTATION GUIDE

PROJECT NUMBER:	B20279
PROJECT TITLE:	
AUTHOR:	Martin Louis
REVISION:	1.0

DATE:	Tuesday, 27 th January 2015
--------------	--

CLIENT:	Romteck Internal Document
----------------	---------------------------

Document Revision History

Pages	Descriptions	Rev.	Chg. By	Appr. By	Date (dd/mm/yyyy)
All	First release	1.0	MGL	MGL	27/01/2015



Contents

1	Overview	4
2	Operation	5
2.1	Survey	5
2.1.1	Survey Procedure	5
2.1.2	Measurement Data	7
2.1.3	Core Orientation Procedure Illustrated	10
2.1.4	Bore Orientation Procedure Illustrated	11
2.2	Setup	13
2.2.1	About	13
2.2.2	Reset Survey	13
2.2.3	Reset Probe	14
2.2.4	Assign Probe	14
2.2.5	Preferences	15
2.3	Advanced Operation	17
2.3.1	Probe Details	18
3	Implementation	21
3.1	Probe Bluetooth LE Interface	21
3.1.1	Characteristics	21
3.1.2	Shot Format	24
3.2	Orientation Calculation and Calibration	28
3.2.1	Calibrating The Sensor Readings	28
3.2.2	Temperature dependent Calibration	29
3.2.3	Calibration Storage On Probe	29
3.2.4	Orientation Calculation	31
3.2.5	Using The ecompass.c Code To Do All The Hard Work	33
3.3	Probe Connection Sequence	36
4	File Formats	38
4.1	Configuration File	38
4.2	Survey State	39
4.3	Survey Report	40
4.4	Calibration File Format	40
4.5	Sensor Data	44
4.6	Calibration Data	45



Romteck Australia Pty Ltd Collingwood Street, Osborne Park WA 6017
Phone (08) 9244 3011 Fax (08) 9244 2649 Web www.romteck.com



1 OVERVIEW

The Suntech mobile app controls and manages orientation "probes" used to monitor drilling operations in the mining industry. A probe uses accelerometer and magnetometer sensors to measure its orientation in terms of roll, pitch and azimuth angles which can be used to determine the path of a bore hole or the orientation of a core sample. Two types of probes exist. The first is the "BoreCam" which lowered down a bore hole. Several measurements are taken at various depths which are subsequently reported to the operator and used to determine the bore profile. The second type of probe is the "CoreCam". It is attached to a core drill and records the orientation when a core sample is taken. This measurement is used to identify the position and orientation of the core sample back on the surface with respect to other core samples. CoreCam probes are usually used in pairs (a black one and a white one) where one is used to do a survey while the data from the other is being analysed.

The Suntech App runs on a mobile device (often called the "handheld"). It is used to configure probe, control the survey operation and retrieve the results.



BoreCam probe with handheld controller.



Pair of CoreCam probes and handheld controller.

This document will first cover the use of the mobile app from an operator's perspective and then go on to a more detail, describing implementation issues for the programmer.



2 OPERATION

The application has two operating modes:

- Bore Orientation Mode is used with BoreCam probes. In this mode, a single probe is assigned to the app. It is sometimes called single probe mode.
- Core Orientation Mode is used with CoreCam probes. In this mode, a pair of probes (a black one and a white one) are assigned to the app. It is sometimes called dual probe mode.

These two modes are virtually identical apart from the number of probes it must track and a slight difference in the survey procedure. A CoreCam survey will only take one measurement each time it is sent down a hole whereas a BoreCam may take multiple measurements at different depths. The probes themselves contain the same firmware and differ only in that a CoreCam measures only pitch and roll angles and therefore does not return magnetometer sensor values.

2.1 SURVEY

2.1.1 SURVEY PROCEDURE

Conducting a survey is the primary purpose of the app and the process an operator goes through may be broken down into several steps:

- The first step is to "initialise" the probe. This just starts the survey process. In Bore Orientation mode only, the operator is asked if they wish to start a new survey session or continue a previous one. Continuing a previous session requires the operator to select a "Survey State" XML file that contains the state of a survey session. These files are detailed in the implementation section of this document.
- The operator then fills in some record keeping information (i.e. Hole ID, Operator Name & Company Name). If the depth tracking option is enabled (is enabled by default) then the operator also enters the starting depth for the survey and a depth interval. For convenience, this information defaults to values carried over from previous survey. The current Hole ID and Operator name is displayed in the main page. They can be "preset" on the main page by tapping on them.
- Once the operator confirms the information, a survey session is started. This causes the app to tell the probe to begin taking measurements and storing them at some fixed rate (determined by the probe). This rate is known as the "shot interval" and is reported by the probe in a Bluetooth LE characteristic. The app timestamps the start of the survey run and can therefore calculate the measurement number (usually called the "shot number") for any particular time.
- The probe is then sent down the hole. It will lose connection with the handheld device but will keep recording measurements at the shot interval.



- The operator may at any time press the "take measurement" button. The app runs a timer and shows an associated progress bar during which the operator may not move the probe. The timer runs for 1½ times the shot interval reported by the probe and ensures that the probe is held stationary during the next scheduled measurement. The app records the time that the measurement was taken (i.e. the button was pressed) and from this calculates the "shot number" of closest measurement recoded by the probe after that time.

During the survey, if depth tracking is enabled, the "next depth" is displayed which will be the depth assigned to the next measurement taken. This will default to the "Initial Depth" specified at the start of the survey but the operator may tap on this at any time before the measurement is taken to manually change its value.

Additional functionality in Bore Orientation mode:

In Bore Orientation mode, multiple measurements may be taken during a survey session. After a measurement, "depth interval" (specified at the start of the survey) is added to "next depth" if the probe is heading into the hole and subtracted if coming out. An in/out switch allows the operator to set the direction as the survey progresses. At any time, the operator may tap on next depth, to manually change it to a specific value or to change the depth interval. "MultiShot" mode is a configurable option which is enabled by default and gives the operator the choice to proceed to the next depth after any measurement or to take another measurement at the same current depth.

- The probe is then pulled from the hole. It will automatically reconnect to the App when it comes back into range. The App will automatically fetch any measurements that have been recorded by requesting the corresponding shot numbers.
- The measurements are then analysed.

In Core Orientation mode, the probe's orientation is compared to that of the measurement in real time so that the operator can physically align core samples with each other. This is done by tapping on a measurement to activate the Orientation screen. The App instructs the probe to transmit live sensor data while this screen is visible. Both the Survey measurement roll angle and the current real-time roll of the probe are shown. The difference between these is represented graphically on an instrument dial. Red arrows indicate which direction the probe must be turned to match the measurement. When the difference in angle gets within 0.1 degrees (this threshold is configurable) the arrows indicate a match by turning green. The operator then knows that the core sample is in the same orientation that it was in the ground.

In Bore Orientation mode, the results of the survey are typically displayed to be written into a log book by the operator. However, once a survey session has ended, the app creates a report in the form of a CSV file that can be extracted from the mobile device by connecting it to a PC or by emailing it to a specified recipient. Measurements, may also be compared to real-time probe orientation as described above for Core Orientation mode. In this case, then orientation screen shows roll, dip and azimuth angles and graphically displays the difference in roll and dip angles (see screenshots in later sections).

- In Bore Orientation mode, the state of the survey is saved to a Survey State XML file as it progresses. This allows the operator to continue a previously completed survey by sending the probe down the hole again and taking more measurements at a later time if necessary. The

contents and format of the survey state file are detailed in the implementation section of this document.

2.1.2 MEASUREMNT DATA

The app receives the following information for each measurement from the probe:

- **Temperature.** The probe may report up to 3 temperatures one for the accelerometer, one for the magnetometer and one for the probe itself. If any of these are absent, the probe will use the most appropriate of the available values). This is explained in more detail in the implementation section.
- **Accelerometer** sensor values in units of g. 3 values, one for each axis (x, y & z with respect to the probe).
- **Magnetometer** sensor values in units of μT (micro Tesla). This is returned only by BoreCam probes. 3 values, one for each axis (x, y & z with respect to the probe).

The accelerometer and magnetometer values are calibrated by the app and then used to calculate the probe's orientation in terms of roll, pitch and azimuth angles. In the case of CoreCam probes that don't have magnetometers, the magnetometer sensor values will be absent and only roll and pitch can therefore be calculated. This is explained in more detail in the implementation section. Each measurement records and displays the following information:

Quantity	Displayed On	Units	Source	Notes
Name	All measurements		Generated by app	This will normally just be "Measurement 1", "Measurement 2", etc... If a measurement is repeated at a given depth, then all of the repeats share the same measurement number. If more than one measurement is taken with the same number, then letters "A", "B", "C", etc are appended. E.g. If "Measurement 2" was repeated, it would become "Measurement 2A" and "Measurement 2B".
Timestamp	All measurements		Generated by app	The time and date that the measurement was taken.
Roll angle	All measurements	Degrees	Derived from probe sensor values	Calculated from the calibrated accelerometer sensor values. Normally displayed in black but displayed in red if a motion alarm is triggered for this measurement (see below).



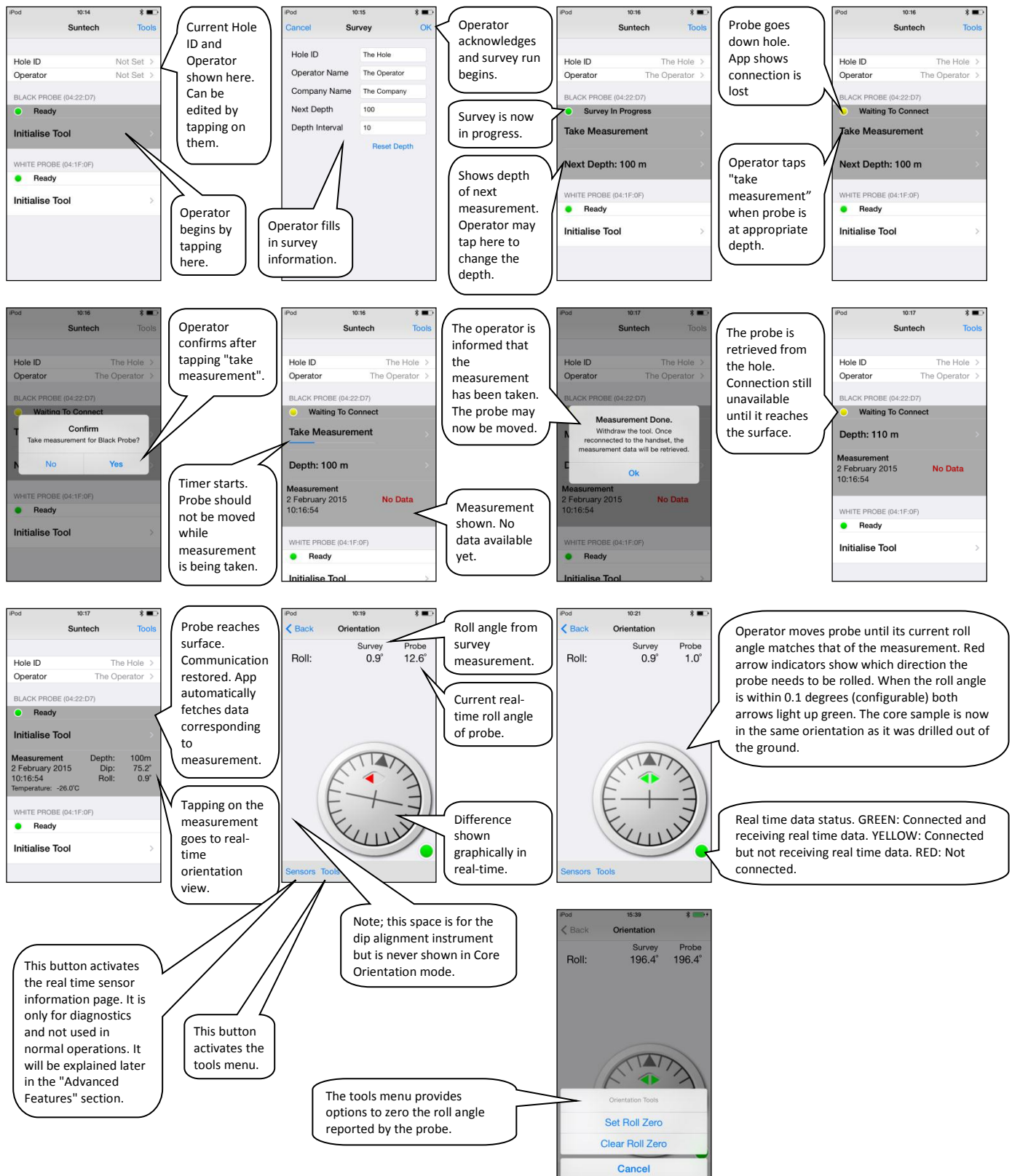
Pitch angle / Dip	All measurements	Degrees	Derived from probe sensor values	Calculated from the calibrated accelerometer sensor values. Calculated as pitch but displayed as "Dip". Pitch angle is positive up and is used for internal calculations. Operators prefer to see dip which is the same as pitch but positive down. When displayed, the negative pitch angle is shown and labelled as "Dip". Normally displayed in black but displayed in red if accelerometer stability alarm is triggered (see below).
Azimuth angle	Only BoreCam measurements	Degrees	Derived from probe sensor values	Calculated from the calibrated accelerometer and magnetometer sensor values. Normally displayed in black but displayed in red if magnetic field stability alarm is triggered (see below).
Depth	All measurements	Meters	Calculated by app during survey.	Depth tracking may be disabled in the preferences. If so, depth is not recorded in a measurement and is displayed as "N/A".
Accelerometer Magnitude	Not displayed.	g	Derived from probe sensor values	Vector sum of the 3 accelerometer sensor axial components. This value is used for the accelerometer stability alarm. In a steady state, this value should be close to 1. If it is not, the probe is in motion and the roll / pitch angles may be invalid. If this value deviates from 1 by more than a configurable amount, an accelerometer motion alarm is triggered for this measurement and the roll/ pitch values will be displayed in red.
Magnetometer Magnitude	Only BoreCam measurements	Displayed as nano Tesla (nT) but calculated as micro Tesla (μT)	Derived from probe sensor values	Vector sum of the 3 magnetometer sensor axial components. This value is used for the magnetic field alarm. normally, this value should be close to the strength of Earth's magnetic field. If it is not, the probe is close to a source of magnetic interference and the azimuth angle may be invalid. If this value deviates from a configurable nominal value by more than a configurable amount, an magnetic field alarm is triggered for this measurement and the azimuth value will be displayed in red.



Temperature	All measurements	°C	Derived from probe sensor values	The probe <u>may</u> report up to 3 temperatures, one each for the accelerometer, magnetometer and the probe itself. The values displayed for a measurement is the temperature of the probe. If the probe temperature is not reported, the most appropriate of the remaining temperatures is used instead (explained in more detail in implementation section).
-------------	------------------	----	----------------------------------	---

2.1.3 CORE ORIENTATION PROCEDURE ILLUSTRATED

Below is a typical operator sequence shown as a series of screen shots with annotation.



Current Hole ID and Operator shown here. Can be edited by tapping on them.

Operator begins by tapping here.

Operator fills in survey information.

Operator acknowledges and survey run begins.

Survey is now in progress.

Shows depth of next measurement. Operator may tap here to change the depth.

Probe goes down hole. App shows connection is lost

Operator taps "take measurement" when probe is at appropriate depth.

Operator confirms after tapping "take measurement".

Timer starts. Probe should not be moved while measurement is being taken.

The operator is informed that the measurement has been taken. The probe may now be moved.

Measurement shown. No data available yet.

The probe is retrieved from the hole. Connection still unavailable until it reaches the surface.

Probe reaches surface. Communication restored. App automatically fetches data corresponding to measurement.

Tapping on the measurement goes to real-time orientation view.

Roll angle from survey measurement.

Current real-time roll angle of probe.

Difference shown graphically in real-time.

Operator moves probe until its current roll angle matches that of the measurement. Red arrow indicators show which direction the probe needs to be rolled. When the roll angle is within 0.1 degrees (configurable) both arrows light up green. The core sample is now in the same orientation as it was drilled out of the ground.

Real time data status. GREEN: Connected and receiving real time data. YELLOW: Connected but not receiving real time data. RED: Not connected.

Note: this space is for the dip alignment instrument but is never shown in Core Orientation mode.

This button activates the real time sensor information page. It is only for diagnostics and not used in normal operations. It will be explained later in the "Advanced Features" section.

This button activates the tools menu.

The tools menu provides options to zero the roll angle reported by the probe.



2.1.4 BORE ORIENTATION PROCEDURE ILLUSTRATED

Below is a typical operator sequence for BoreCam operation.

Screenshot 1 (10:38): Operator begins by tapping here. (Callout points to the 'Initialise Tool' button)

Screenshot 2 (10:40): Operator selects whether to start new survey session or continue the previous one. (Callout points to the 'Start New' button)

Screenshot 3 (10:40): In this case a new session is started. (Callout points to the 'Start New' button)

Screenshot 4 (10:41): Selecting Resume Previous prompts the operator to select the survey state file for the desired survey. (Callout points to the 'Resume Previous' button)

Screenshot 5 (10:41): Operator fills in survey information. (Callout points to the 'Survey' form fields)

Screenshot 6 (10:41): Operator acknowledges. (Callout points to the 'Survey' form)

Screenshot 7 (10:41): Survey is ready to go. (Callout points to the 'Survey In Progress' status)

Screenshot 8 (10:42): Probe is lowered into bore hole and loses connection. (Callout points to the 'Waiting To Connect' status)

Screenshot 9 (10:42): Operator confirms measurement. (Callout points to the 'Confirm' dialog box)

Screenshot 10 (10:42): Measurement recorded. Operator must keep probe still while measurement in progress. (Callout points to the 'Confirm' dialog box)

Screenshot 11 (10:43): Measurement complete. Operator asked to either proceed to next depth or take another measurement. (Callout points to the 'Next Measurement' dialog box)

Screenshot 12 (10:43): In this case operator chooses to proceed. (Callout points to the 'Proceed' button)

Screenshot 13 (10:43): Ready for next measurement. (Callout points to the 'Waiting To Connect' status)

Screenshot 14 (10:43): Depth of previous measurement and next measurement shown. (Callout points to the 'Previous Depth' and 'Next Depth' fields)

Screenshot 15 (10:43): Operator takes another measurement. (Callout points to the 'Take Measurement 2' button)

Screenshot 16 (10:43): This time operator chooses to repeat a measurement at same depth. (Callout points to the 'Repeat' button)

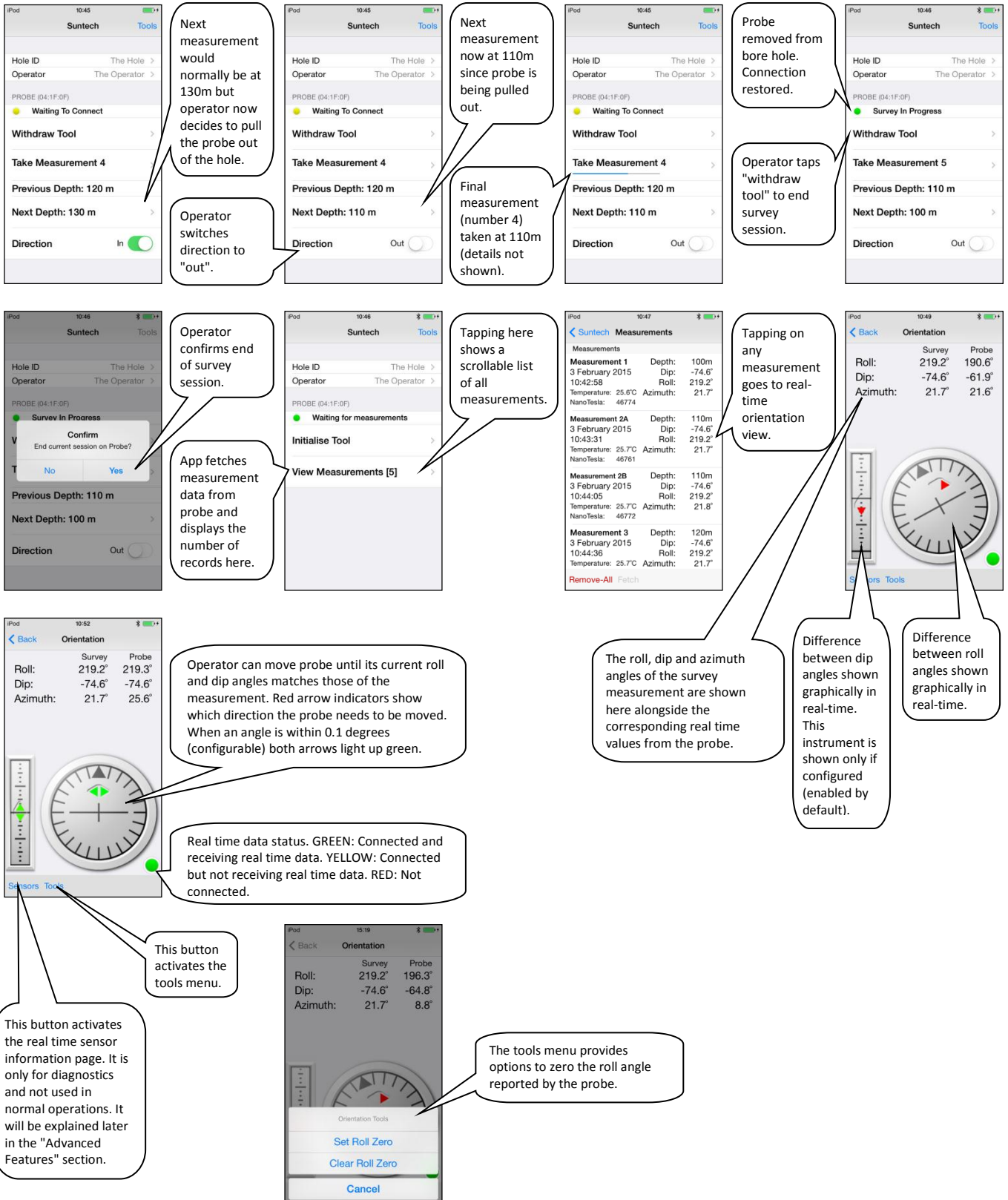
Screenshot 17 (10:43): Measurement number reflects the repeat. (2B not 3). (Callout points to the 'Take Measurement 2B' button)

Screenshot 18 (10:43): Depth of next measurement remains same as previous. (Callout points to the 'Next Depth' field)

Screenshot 19 (10:44): Measurement taken. (Callout points to the 'Take Measurement 2B' button)

Screenshot 20 (10:44): Operator proceeds. (Callout points to the 'Proceed' button)

Screenshot 21 (10:44): Operator takes one more measurement (details not shown). At this stage, they now have 4 measurements. Measurement 1 at 100m Measurements 2A & 2B at 110m Measurement 3 at 120m. (Callout points to the 'Take Measurement 3' button)



Screenshot 1: The app shows 'Waiting To Connect'. Callout: 'Next measurement would normally be at 130m but operator now decides to pull the probe out of the hole.'

Screenshot 2: The app shows 'Waiting To Connect'. Callout: 'Operator switches direction to "out".'

Screenshot 3: The app shows 'Waiting To Connect'. Callout: 'Next measurement now at 110m since probe is being pulled out.'

Screenshot 4: The app shows 'Waiting To Connect'. Callout: 'Final measurement (number 4) taken at 110m (details not shown).'

Screenshot 5: The app shows 'Waiting To Connect'. Callout: 'Probe removed from bore hole. Connection restored.'

Screenshot 6: The app shows 'Survey In Progress'. Callout: 'Operator taps "withdraw tool" to end survey session.'

Screenshot 7: The app shows 'Survey In Progress'. Callout: 'Operator confirms end of survey session.'

Screenshot 8: The app shows 'Waiting for measurements'. Callout: 'App fetches measurement data from probe and displays the number of records here.'

Screenshot 9: The app shows a scrollable list of measurements. Callout: 'Tapping here shows a scrollable list of all measurements.'

Screenshot 10: The app shows the 'Orientation' view. Callout: 'Tapping on any measurement goes to real-time orientation view.'

Screenshot 11: The app shows the 'Orientation' view. Callout: 'The roll, dip and azimuth angles of the survey measurement are shown here alongside the corresponding real time values from the probe.'

Screenshot 12: The app shows the 'Orientation' view. Callout: 'Difference between dip angles shown graphically in real-time. This instrument is shown only if configured (enabled by default).'

Screenshot 13: The app shows the 'Orientation' view. Callout: 'Difference between roll angles shown graphically in real-time.'

Screenshot 14: The app shows the 'Orientation' view. Callout: 'Operator can move probe until its current roll and dip angles matches those of the measurement. Red arrow indicators show which direction the probe needs to be moved. When an angle is within 0.1 degrees (configurable) both arrows light up green.'

Screenshot 15: The app shows the 'Orientation' view. Callout: 'Real time data status. GREEN: Connected and receiving real time data. YELLOW: Connected but not receiving real time data. RED: Not connected.'

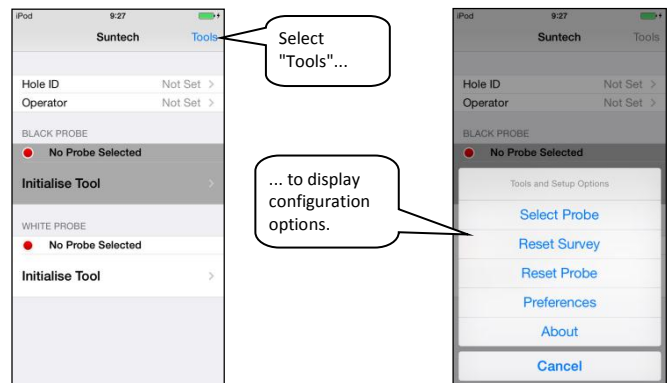
Screenshot 16: The app shows the 'Orientation' view. Callout: 'This button activates the tools menu.'

Screenshot 17: The app shows the 'Orientation' view. Callout: 'This button activates the real time sensor information page. It is only for diagnostics and not used in normal operations. It will be explained later in the "Advanced Features" section.'

Screenshot 18: The app shows the 'Orientation Tools' menu. Callout: 'The tools menu provides options to zero the roll angle reported by the probe.'

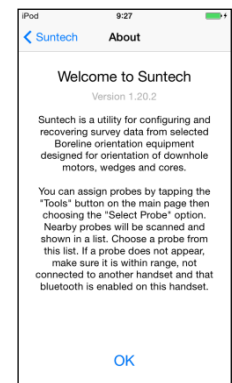
2.2 SETUP

Configuring the app is done from the "Tools" button on the main page. This pops up a menu allowing the operator to access preferences. It also provides options to assign probes to the app, to reset the state of a survey in progress and to reset a probe. An "About" option is also available showing the app's version number and brief description.



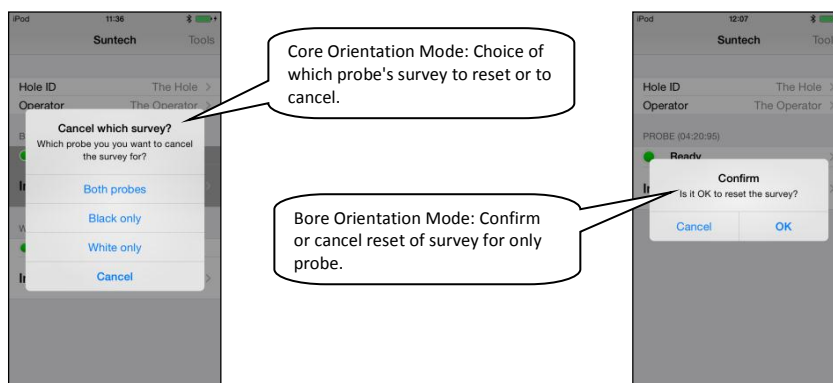
2.2.1 ABOUT

The about option shows a page with the app's version number and a short description.



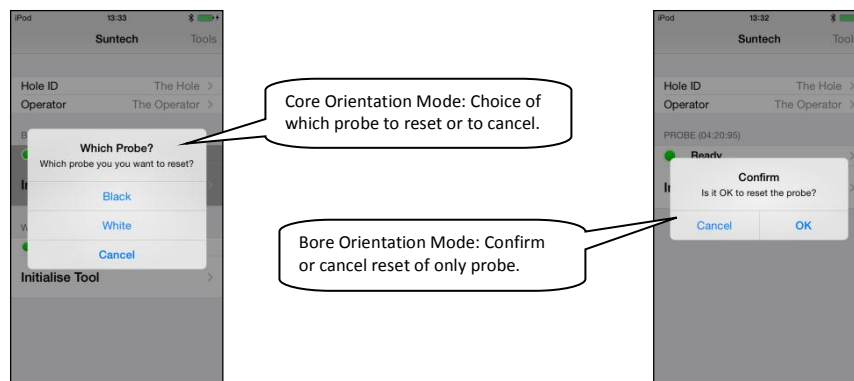
2.2.2 RESET SURVEY

The "reset survey" option effectively cancels a survey by resetting its state. The Hole ID, Operator Name, Company Name, Initial Depth and Depth Interval are retained as default values for the next survey session. In Core Orientation mode, the operator is given a choice of resetting the black probe, the white probe, both or cancelling. In Bore orientation mode the operator is asked to either confirm or cancel the survey reset.



2.2.3 RESET PROBE

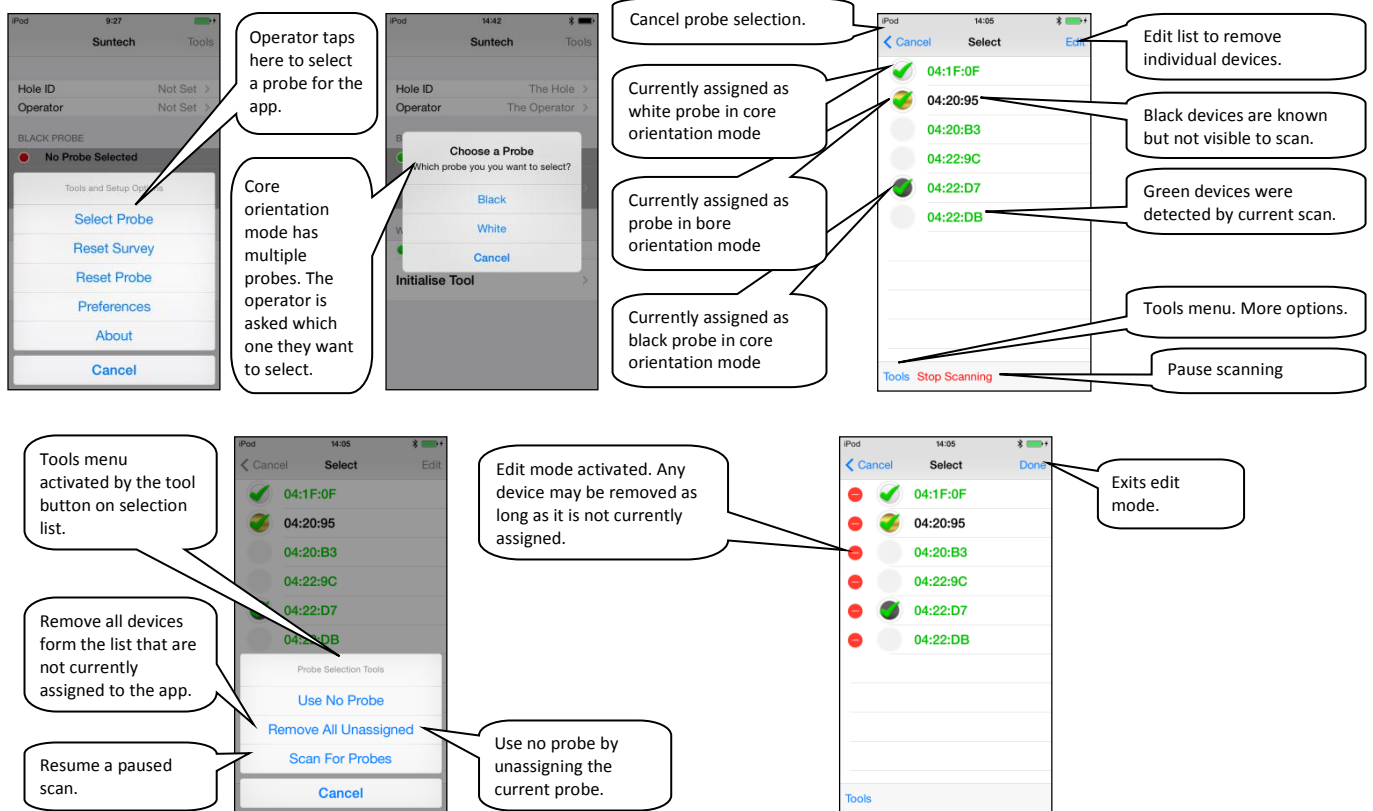
Resetting a probe just forces a re-connect. This means that the BlueTooth LE connection is closed and on success, immediately re-established. This causes the app to go through the probe connection sequence again, effectively putting it back to default state and re-reading its internal data. More detail on the probe connection sequence is given later in the implementation section. In Core Orientation mode, the operator is asked to select which probe to reset. In Bore orientation mode the operator is asked to confirm the reset.



2.2.4 ASSIGN PROBE

Before any probes can be used, they first need to be assigned to the app. This is done from the "Probe Select" screen, activated from the "Select Probe" option on the main screen's tools menu. When activated, this screen begins a scan for all nearby BlueTooth LE devices and shows them in a list. This list is non volatile, it remembers devices from previous scans. This feature allows the operator to change probes temporarily for testing purposes and quickly change back to the original even if the original is not currently within scanning range. Each time the list is displayed, all devices will appear black but then change to green as they are detected in the current scan (and added if they are not already in the list). The operator can then tell which probes are currently within range and which are not by their colour. Probes that are currently assigned to the handheld are marked with a coloured icon to indicate its role (black or white for Core Orientation mode, gold for Bore Orientation mode). Any of the devices not currently assigned may be removed by editing the list. All devices not currently assigned may also be removed using a single option in the tools menu. Scanning may be manually paused and resumed by the operator. Pausing the scan prevents the list being updated and makes it easier to select a device if there many being detected.

The operator may select any probe from the list whether or not it is currently visible to the scan (as long as it is not already assigned to the app). They may also cancel out of the select screen to leave the current probe assignment as it is or opt to assign "no probe" which returns the selected probe in an unassigned state. Before entering the probe select screen in Core Orientation mode, the operator will be asked to choose which of the probes he wants to assign (black or white).



2.2.5 PREFERENCES

Application options are set in the preferences list which can be activated from the preferences option in the tools menu of the main screen. One of the settings allows the activation of advanced mode. This enables additional options that a normal operator would not be interested in. The preferences in normal mode are detailed below. The values shown are the default for each item. In advanced mode, more preferences will appear. These are covered later in the "Advanced Operation" section.



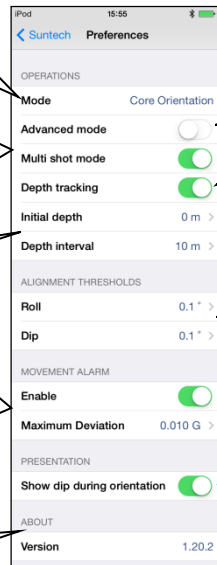
Shows operating mode. Either "Core Orientation" or "Bore Orientation". This can only be changed when advanced mode is active.

Enables or disables multi shot mode. If enabled in Bore Orientation mode, the operator is given the choice to proceed to the next depth after each measurement or to repeat a measurement at the same depth. Has no effect in Core Orientation mode. Enabled by default.

Default values for initial depth and depth interval used when starting a new survey session.

Enables movement alarm and specifies the maximum deviation. If enabled, any measurement whose calibrated accelerometer vector magnitude differs from 1g by more than this amount is deemed not stable. The dip and roll will then be displayed in red and the measurement is flagged with a movement alarm in the survey report.

Application version number.



Turns advanced mode on or off. Additional features available in advanced mode are covered later in the "Advanced Operation" section.

Enables depth tracking. When enabled, the depth is recorded for each measurement. The depth is calculated by the app based on an initial depth for the survey session and a depth interval between measurements.

Specifies the alignment thresholds for dip and roll during real-time orientation of a measurement. The probe's real-time orientation angles must be within this much of the measurements angles to be considered aligned and the arrow indicators to show green.

This option enables the display of the dip alignment instrument on the real time orientation screen for a measurement. This used in Bore Orientation mode only since the instrument is never displayed in Core Orientation mode.



Each of the numeric options above can be edited. Tapping on them opens an edit page, displaying a description and the units for the item. The table below lists the description, units and default value for each of the editable numeric options.

Description.

Units.

Value.

Item	Description	Default	Units
Initial Depth	Default Initial Depth	0	meters
Depth Interval	Default Depth Interval	10	meters
Roll	Real Time Roll Alignment Threshold	0.1	degrees
Dip	Real Time Dip Alignment Threshold	0.1	degrees
Maximum Deviation	Accelerometer Movement Alarm Maximum Deviation	0.01	g

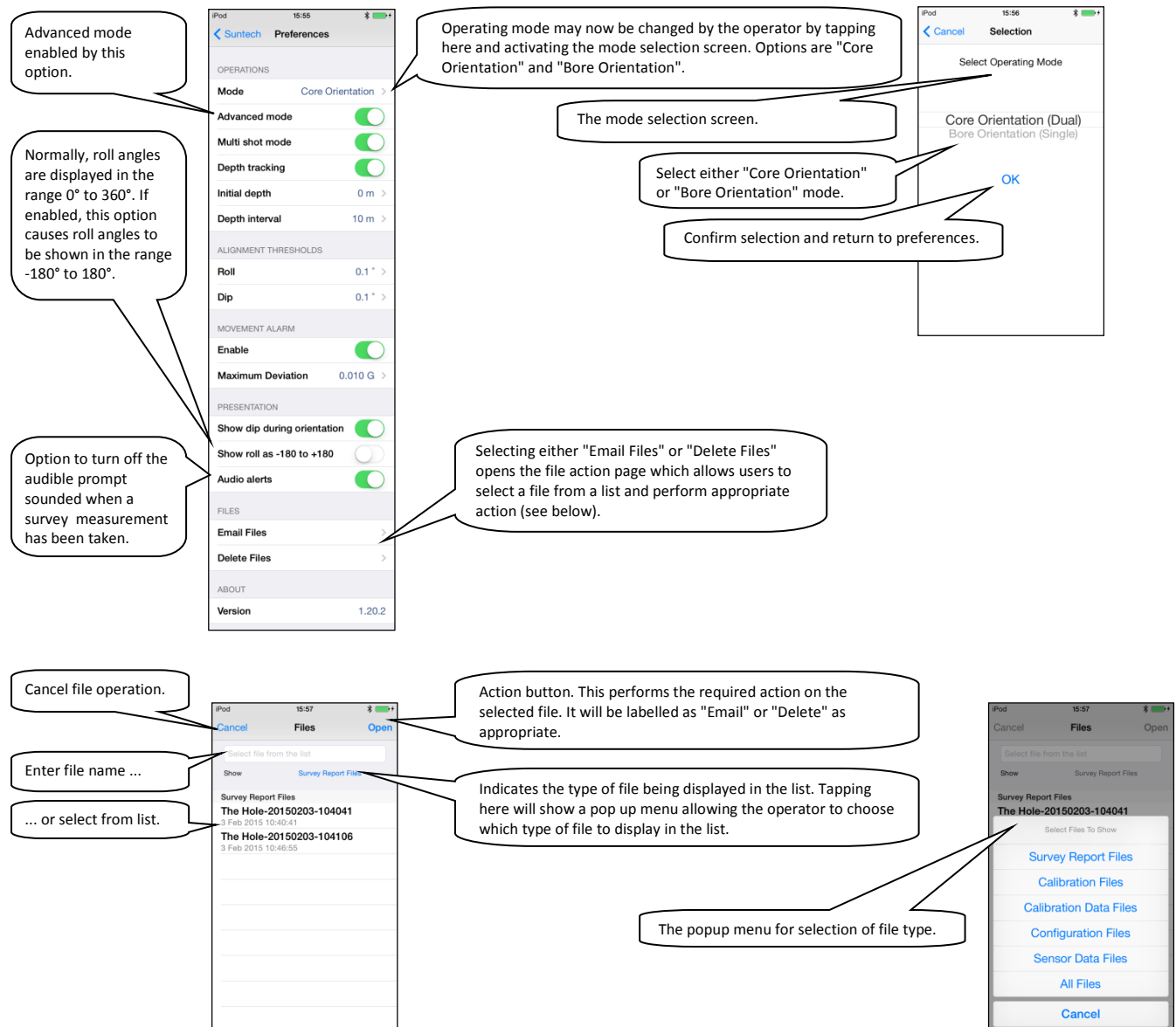
2.3 ADVANCED OPERATION

The advanced mode can be activated from the preferences screen. When enabled, the following additional features are available:

Preference option to show roll as -180° to 180° . Normally, roll angles are displayed in the range 0° to 360° . The default for this option is to be off.

Preference option to disable the audible prompt (the beep) that sounds when a survey measurement has been taken and the operator may again move the probe. Audio prompts are enabled by default. The ability to email or delete various files used or generated by the app. Details of file types and their formats can be found in the implementation section of this document.

Access to "Probe Details" page which provides detailed information about a probe and some diagnostic features.



Advanced mode enabled by this option.

Operating mode may now be changed by the operator by tapping here and activating the mode selection screen. Options are "Core Orientation" and "Bore Orientation".

The mode selection screen.

Select either "Core Orientation" or "Bore Orientation" mode.

Confirm selection and return to preferences.

Normally, roll angles are displayed in the range 0° to 360° . If enabled, this option causes roll angles to be shown in the range -180° to 180° .

Option to turn off the audible prompt sounded when a survey measurement has been taken.

Selecting either "Email Files" or "Delete Files" opens the file action page which allows users to select a file from a list and perform appropriate action (see below).

Cancel file operation.

Enter file name ...

... or select from list.

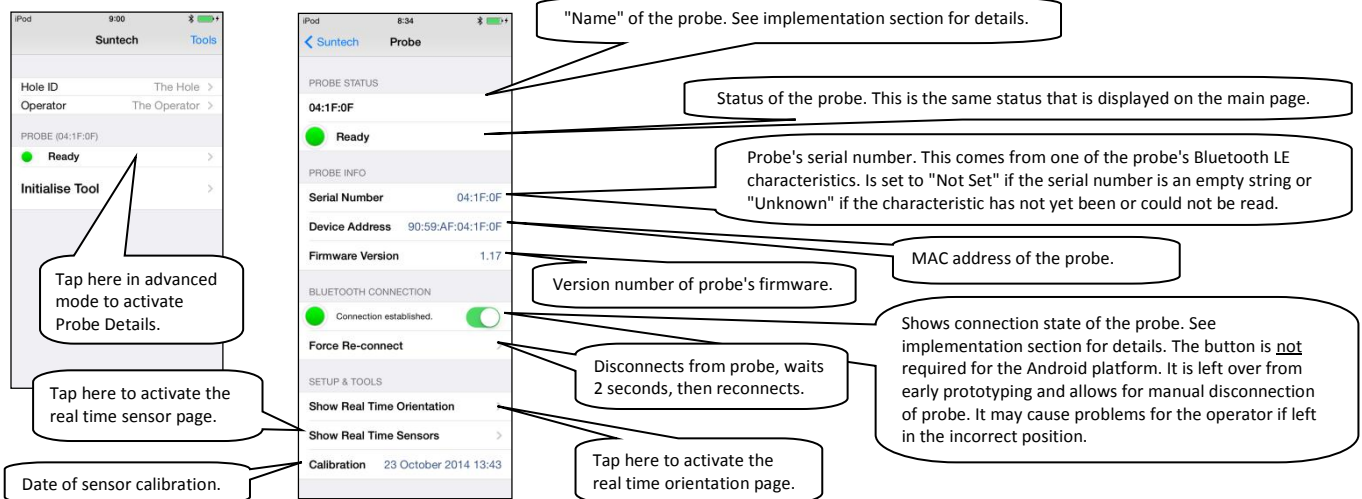
Action button. This performs the required action on the selected file. It will be labelled as "Email" or "Delete" as appropriate.

Indicates the type of file being displayed in the list. Tapping here will show a pop up menu allowing the operator to choose which type of file to display in the list.

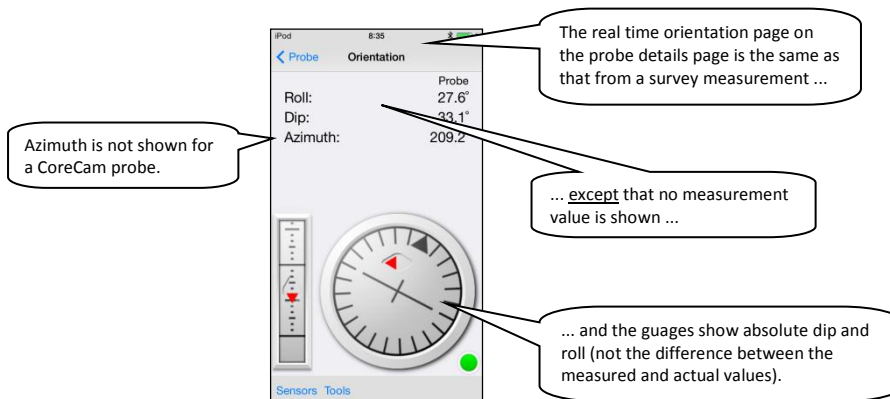
The popup menu for selection of file type.

2.3.1 PROBE DETAILS

The probe details page shows probe information and provides access to diagnostic features. It is available in advanced mode by tapping a probe's status on the main page.



The real time orientation pages shows the current real time position of the probe. It is the same as the one shown for a survey measurement except there are no survey measurement values to compare the real time orientation to.



The real time sensors page show detailed numerical values and related derived information for the probe sensors. The sections on this page are listed below:

- **Orientation:**
This shows the numeric values for current roll dip and azimuth. Any values that are unavailable are shown as "N/A" (e.g. when there is no connection to the probe or the azimuth on a CoreCam which is never available since it has no magnetometer). Also shows the probe temperature.



- **Accelerometer:**
Shows the calibrated values for each of the accelerometer's axes. Also shows the accelerometer temperature and the magnitude error of the reading. If the probe is steady then the vector sum of the axial readings should be 1.0g. The "magnitude error" is the absolute value of the difference between the vector sum of the axial readings and 1.0.
i.e.
$$e_{mag} = \left| \sqrt{a_x^2 + a_y^2 + a_z^2} - 1.0 \right|$$
- **Magnetometer:**
Shows the calibrated values for each of the magnetometer's axes and the magnetometer temperature.
- **Maximum Deviation:**
This section shows information about the history of several sensor values. Each live reading from the probe gives the app the 3 axial components from the accelerometer and 3 from the magnetometer (if appropriate). From this, the app calculates the accelerometer and magnetometer magnitudes as the vector sum of the axial components. For each of these 8 values, the following calculation is performed: The mean value over the last 5 measurements is calculated. For each of the last 5 measurements, the deviation from that mean value is calculated (the absolute value of the difference of the value from the mean). I.e.
$$\Delta_{max} = \max \left(\left| x_i - \frac{\sum_{n=1}^5 x_n}{5} \right| \right) \text{ where } i = 1..5 \text{ (each of the last 5 measurements).}$$

The largest deviation of the last 5 samples is the "maximum deviation" and gives an indication of how steady the value is.
The values shown in this section will be green if the value is stable, and black if it is not. An accelerometer value is deemed stable if its maximum deviation is less than or equal to 0.0004g and a magnetometer value is stable if less than or equal to 0.0375μT.
NOTE: The **ecompass . c** module tracks the history of sensor values as they are fed in to calculate orientation over time and will automatically generate the mean, maximum deviations and stability flag for you. See implementation section for details.

A hold toggle button at the bottom of the page can be used to hold the current values by preventing updates of the display when active.

The real time sensors page also lets the operator save the currently visible data to a CSV file for further analysis later. The first time the save button is pressed after entering the real time sensors page, it will automatically generate the file name using the following auto save file template: "SensorData-Auto-yyyyMMdd-HHmms.csv", where "yyyy" is the year, "MM" is the month, "dd" is the day, "HH" is the hour (in 24 hour format), "mm" is the minutes and "ss" the seconds of the current time. Each time the save button is pressed, the measurement is appended to this file. If a file ever grows to 1000 records, a new file is started. An auto save option can be enabled which will automatically save every measurement to file as it is displayed. The iOS version also has an option to allow the operator to specify their own file names, however this is never used and is not required in the Android version.

Orientation section shows Roll, Dip, Azimuth (if appropriate) and Probe Temperature.

Accelerometer reading. Calibrated axial components, Accelerometer temperature and Magnitude Error.

Magnetometer reading. Calibrated axial components, and Magnetometer temperature.

MEAN VALUES	
Show Mean Values	<input checked="" type="checkbox"/>
Accelerometer	0.999 g
Magnetometer	51.230 μ T
Accelerometer X	-0.545 g
Accelerometer Y	0.387 g
Accelerometer Z	0.742 g
Magnetometer X	11.832 μ T
Magnetometer Y	-14.325 μ T
Magnetometer Z	-47.742 μ T

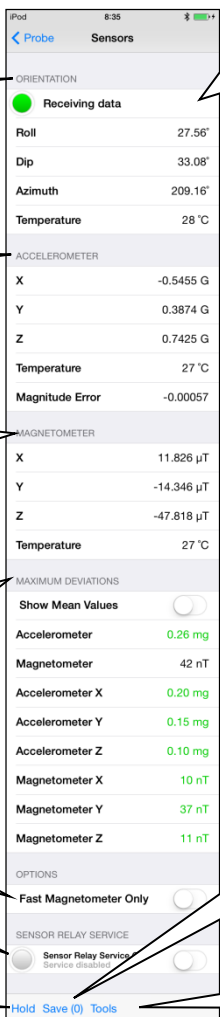
Normally, this section shows maximum deviations. When enabled, this button shows mean values ...

... like this.

Don't worry about this. This is not required for Android version.

Don't worry about this. This is not required for Android version.

Hold button toggles freezing of the display.

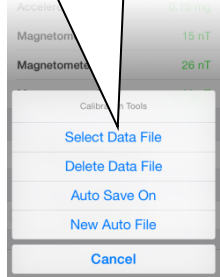


Status of real time data. Red indicator denoted "No Connection to Probe", Yellow is "Connected But No Data", Green is "Receiving Data". If a survey is in progress no real time data is shown, the indicator will be red and the message will be "No data, Survey in Progress"

Tools menu. Contains options for saving data to file. SELECT DATA FILE: Allows the operator to manually select their own file name (rather than to have it automatically generated). This option is not required in the Android version. DELETE FILE: Allows the operator to delete one or more save data files. AUTO SAVE ON/OFF: Toggles auto saving. NEW AUTO FILE: Forces the next saved measurement to begin a new file with its name determined by the auto file template explained above.

Save button saves current data to selected file. The number in brackets shows how many records are currently in the file.

Tools menu. Provides options for saving data to file.





3 IMPLEMENTATION

This section covers implementation details for the hand held application including:

- Interfacing to the probe hardware.
- Configuring a probe.
- Retrieving data from a probe.
- Orientation concepts and calculations.
- Data files and formats.

References will occasionally be made to the iOS source code.

3.1 PROBE BLUETOOTH LE INTERFACE

Communication with the probe is done via Bluetooth LE (sometimes called Bluetooth Smart) using GAP and ATT protocols to exchange GATT information. This should be handled by the Android's Bluetooth API but an overview can be found here:

<https://developer.bluetooth.org/TechnologyOverview/Pages/BLE.aspx>

3.1.1 CHARACTERISTICS

The probe's Bluetooth characteristics are grouped into two services.

Service FFF0 is the primary service and contains configuration and operational data.

Service FF00 provides calibration parameter storage services.

The probe basically functions as a data logger for accelerometer, magnetometer and temperature sensor readings. The characteristics in the primary service (FFF0) configure and control this process. The probe starts in an idle mode where it waits to begin a survey. It is then put into survey mode where it starts to continually log sensor readings called "shots" at a fixed interval. When the survey is complete the probe is put back into idle mode where it waits for the app to retrieve the specific "shots" that it is interested in. The probe has a third mode called Real Time mode whereby it will take measurements continuously and send immediately them directly to the mobile application as characteristic notifications. Real time mode cannot be entered from survey mode (only from idle mode).

The Calibration Parameter service (FF00) provides a small amount of local storage on the probe that holds sensor calibration parameters. This is in the form of 64 blocks of 16 bytes each. The data can be accessed by first writing a block select index (0-63) to the probe then reading or writing the 16 bytes of data for that selected block.

The available characteristics are summarised in the table below:

The access type may be Read and/or Write but can also be Notify (N). Notification characteristics are sent by the probe asynchronously when the associated data is available.



Service	Characteristic	Name	Access	Description
FFF0	FFF1	Device ID	R/W	n Bytes forming an Device ID string. The n bytes may <u>not</u> have a null terminator. The Device ID is normally set by the customer and is typically the probe serial number. The Device ID is used as the name of the probe if it is available.
FFF0	FFF2	Probe Mode	R/W	<p>1 Byte operating mode of the probe.</p> <p>0 = Idle (Probe not busy, ready to begin survey)</p> <p>1 = Recording Survey (Probe busy logging survey shots)</p> <p>2 = Real Time Mode (Probe is sending shots in real time)</p> <p>Can be read to get the current mode of the probe.</p> <p>Can be written to change the mode of the operating probe for example to start or end a survey or to put it into real time measurement mode (a.k.a. rolling shot).</p> <p>When in real time mode, the probe will continually send "Core Shot" or "Bore Shot" notifications containing the real time data at the "Rolling Shot Interval". To prevent excessive battery drain, the probe will drop out of mode 2 back to mode 0 after sending 1200 real time shots. To keep the probe in real time mode, the app can reset real time mode by writing 2 to the Probe Mode again before this happens.</p> <p>Two additional testing modes are available but not used with the Android application:</p> <p>3 = Real Time No Magnetometer (same as mode 2 but magnetometer sensor is not reported)</p> <p>4 = Real Time No Accelerometer (same as mode 2 but accelerometer sensor is not reported)</p>
FFF0	FFF3	Shot Interval	R/W	1 Byte shot interval. The number of seconds between measurement recording in survey mode. Although this is a writeable characteristic, the application never allows it to be modified. The shot interval is purely dictated by the probe. The Shot Interval x Survey Max Shots determines the maximum duration of a survey.
FFF0	FFF4	Record Count	R	2 byte value (low byte first). The number of measurements (a.k.a. shots) that the probe has recorded. Not used by the app. The app infers the number of measurements from the survey duration and the Shot Interval.
FFF0	FFF5	Shot Request	W	2 byte value (low byte first). Requests a survey measurement (a.k.a. shot) from the probe. Write the desired shot number (index starting from 1) to this characteristic. The probe will fetch the record from its memory then send it back to the app as a "Bore Shot" or "Core Shot" notification.



FFF0	FFF6	Survey Max Shots	R	2 byte value (low byte first). Maximum number of measurements that the probe can record when in survey mode. Multiplying this by the Shot Interval gives the maximum duration of a survey.
FFF0	FFF8	Bore Shot	N	A block of data representing a single shot (or measurement record) from a BoreCam probe, containing accelerometer axial components, magnetometer axial components and associated temperatures. The shot format is discussed in detail later.
FFF0	FFF9	Core Shot	N	A block of data representing a single shot (or measurement record) from a CoreCam probe, containing accelerometer axial components, magnetometer axial components and associated temperatures. The shot format is discussed in detail later.
FFF0	FFFB	Device Address	R	6 byte MAC address of the probe. Sent in reverse order. If Device Address characteristic is [0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF] then the MAC address is FF:EE:DD:CC:BB:AA
FFF0	FFFC	Version Major	R	1 byte major portion of the probe's firmware version number.
FFF0	FFFD	Version Minor	R	1 byte minor portion of the probe's firmware version number.
FFF0	FFFE	Rolling Shot Interval	R/W	2 byte value (low byte first). Approximate number of milliseconds between rolling shots (real time measurements when Probe Mode is set to 2). This value is used to determine a timeout for real time probe measurements by adding 40% (i.e. timeout = Rolling Shot Interval * 1.4). If more than this amount of time has elapsed since the last real time measurement was received, the application assumes that the probe has stopped sending them and updates the status on the "Real Time Orientation" and "Real Time Sensors" pages.
FF00	FF01	Calibration Index	W	1 Byte block index (0-63). Selects the storage block number that will be used by the Calibration Data characteristic.
FF00	FF02	Calibration Data	R/W	16 bytes of data corresponding to the storage block selected by the Calibration Index characteristic. Writing 16 bytes stores the data into the selected block. Reading returns the 16 bytes stored in the selected block of the probe.

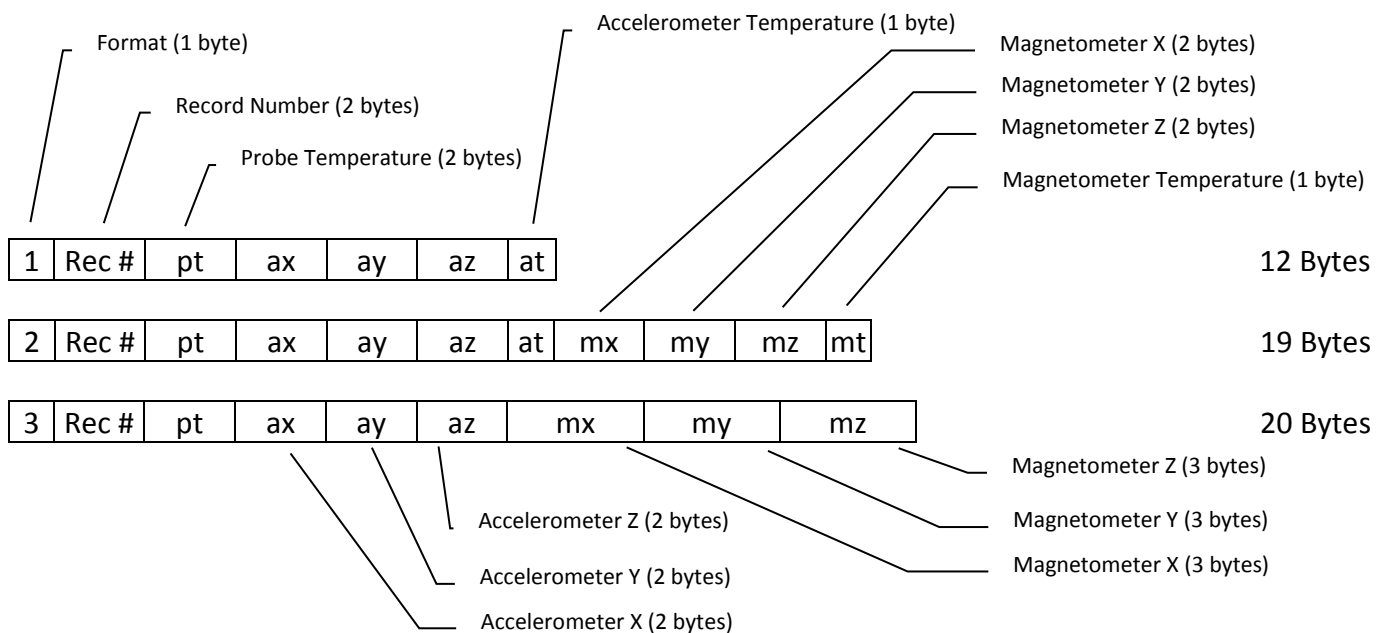
3.1.2 SHOT FORMAT

Survey measurements (sometimes called "shots") and real time sensor data are reported by the probe using its FFF8 or FFF9 Bluetooth LE characteristics. A BoreCam will send them as FFF8 and the CoreCam sends them as FFF9.

Shot records are reported by the probe as characteristic notifications in response to one of two conditions:

- The app wrote a shot number into the Shot Request characteristic (0xFF5). The probe will fetch the specified shot for the last survey session from its internal storage and send it to the app.
- The app put the probe into real-time mode by writing a value of 2 into the Probe Mode characteristic (0xFF2). While in real time mode, the probe will continually send shots containing the current sensor values to the app. The probe will remain in real time mode for 1200 shots after which it will revert back to idle mode. This is to conserve battery if communications is ever lost. To keep the probe in continuous real time mode, the app needs to periodically write 2 to the Probe Mode characteristic to reset the count.

The data returned in these characteristics comes in 3 possible formats, all starting with a single byte format identifier (1, 2 or 3) followed by a 2 byte record number and a 2 byte probe temperature. Format 1 follows this with accelerometer X, Y and Z components (each 2 bytes) and a 1 byte accelerometer temperature. Format 2 is the same as format 1 but adds magnetometer X, Y and Z components (each 2 bytes) and a 1 byte magnetometer temperature. Format 3 is similar to format 2 except that it sacrifices the accelerometer and magnetometer temperature fields and expands the magnetometer X, Y and Z components to 3 bytes each. The formats are detailed below. All multi byte values are big-endian (i.e. most significant byte first).





Format	Can be 1, 2, or 3. Determines the structure of the rest of the data.
Rec #	<p>If the shot was received in response to a shot request then this is the requested shot number or 0xFFFF if the requested shot number does not exist.</p> <p>If the shot was received because the probe in real-time mode then the record number is incremented for every shot sent from the probe (starting at 1). To prevent excessive battery drain after a connection is lost, the probe will revert back to idle mode (mode 0) after 1200 shots have been sent. The app therefore needs to regularly reset real time mode by periodically writing 2 to the Probe Mode characteristic (0xFF2) to keep the probe in real time mode. A good time to do this is after receiving shot number 1000 in real time mode.</p>
pt	<p>Probe temperature is a signed 16 bit fixed point value with 8 binary places and is converted into floating point. A value of 0x8000 (-32768) indicates that the probe temperature is invalid. A valid probe temperature is therefore calculated as:</p> $\text{float ProbeTemp} = (\text{float})\text{pt} / 256.0$
ax, ay, az	<p>These are 16 bit signed values from the accelerometer, one for each axis. They are raw sensor values (integers) that need to be scaled to generate uncalibrated values (floating point) in units of g. These uncalibrated values need to be calibrated before they can be used in orientation calculations. This is covered in later sections.</p> $\text{Uncalibrated Acc}_x = 2.0 * ((\text{double})\text{ax} / (\text{double})0x7FFF)$ $\text{Uncalibrated Acc}_y = 2.0 * ((\text{double})\text{ay} / (\text{double})0x7FFF)$ $\text{Uncalibrated Acc}_z = 2.0 * ((\text{double})\text{az} / (\text{double})0x7FFF)$
at	<p>Accelerometer temperature is a signed 8 bit integer that is converted into floating point. A value of 0x80 (-128) indicates that the accelerometer temperature is unavailable and the probe temperature should be used instead. I.e.</p> $\text{float AccTemp} = (\text{at} \neq 0x80) ? (\text{float})\text{at} : ((\text{float})\text{pt} / 256.0)$
mx, my, mz	<p>These are either 16 bit or 24 bit signed values from the magnetometer depending on the shot format, one for each axis. They are raw sensor values (integers) that need to be scaled to generate uncalibrated values (floating point) in units of μT (micro Tesla). These uncalibrated values need to be calibrated before they can be used in orientation calculations. This is covered in later sections.</p> $\text{Uncalibrated Mag}_x = \text{ScaleFactor} * (\text{double})\text{mx}$ $\text{Uncalibrated Mag}_y = \text{ScaleFactor} * (\text{double})\text{my}$ $\text{Uncalibrated Mag}_z = \text{ScaleFactor} * (\text{double})\text{mz}$ <p>where, ScaleFactor=0.001 in format 3 shots (24 bit values) or ScaleFactor=0.004 in format 2 shots (16 bit values)</p> <p>The iOS code has a method called <code>magnetometersScaleFactorForShotFormat</code>: in <code>probe.m</code> (around line 2978) which calculates the scaling factor for a given shot format. It contains an additional scaling factor that is used only for probes with old firmware and does not need to be supported in the Android app (only the two above are required).</p>
mt	<p>Magnetometer temperature is a signed 8 bit integer that is converted into floating point. A value of 0x80 (-128) indicates that the magnetometer temperature is unavailable and the probe temperature should be used instead. I.e.</p> $\text{float MagTemp} = (\text{mt} \neq 0x80) ? (\text{float})\text{mt} : ((\text{float})\text{pt} / 256.0)$

Using existing code:



The iOS app uses the module **ecompass.c** to handle the calibration of sensor values and orientation calculations. Using that code will save a lot of work. The uncalibrated accelerometer and magnetometer axial components are fed into **ecompass.c** along with the floating point temperatures. Calibrated sensor values and orientation (roll, pitch & azimuth) can then be read out. The **ecompass.c** code is covered in more detail later.

The iOS app parses the shot data using the **processShotData:** method of the **Probe** class in **Probe.m** (at around line 2625). It is suggested to use this function as a reference. Both the FFF8 and FFF9 characteristics are sent to this function for decoding. A brief commentary of **processShotData:** is given below:

- [≈ Line 2637] The function first checks to see if it needs to read the Rolling Shot Interval again from the probe. This is not needed for the android version. Additional probe modes used only during probe development and testing would have different rolling shot intervals that would need to be re-read if the mode changed. These modes are not used in production probes and this check is no longer needed.
- [≈ Line 2646] This check is also left over from development testing. If no shot data was passed to the function, it would try to load it from a characteristic. This function is always called with the data as an argument so this is not needed.
- [≈ Line 2658] Bail out if no shot data.
- [≈ Line 2664] Print hex dump of shot data if debugging enabled. Not enabled in production builds.
- [≈ Line 2683] Parses old shot format for prototype probes only if PROBE_SUPPORT_OLD_SHOT_FORMAT is defined. This is no longer defined so this code is not needed.
- [≈ Line 2733] Gets format from shot data (first byte) and checks if data length is correct for the format. It then parses out and scales each field according to the format, and flags which fields are available.
- [≈ Line 2838] Calculates the temperatures according to the calculations outlined above. If any of the temperatures are reported as invalid by the probe it will fall back to one of the other valid temperatures. This accounts for hardware builds that may not have all temperature sensors available.
- [≈ Line 2867] Does final validation of the record. Only process records whose number matches what was asked for (in real time mode, it will accept any record). If record is ok, it pushes the fields into the ecompass module for calibration and orientation calculation.
- [≈ Line 2893] Performs extra actions for real time mode if appropriate. Flags that real time data is available, restarts the timeout so it can detect if the probe stops sending data and reports the new data to be displayed on the real-time orientation or real-time sensors pages. Finally it will re-write the Probe Mode with 2 after a certain number of records to keep it from dropping back to idle mode.



- [≈ Line 2927] Performs extra actions if not in real time mode (fetching specific survey measurements) if appropriate. Calculates the magnitude (vector sum) for the accelerometer and magnetometer readings then stores the data to the survey. If the probe responded with record number 0xFFFF (no such record) then tells the survey that the request shot number is not in the probe. Finally, it checks to see if there are any further shot number that need to be retrieved (which triggers the fetch).

3.2 ORIENTATION CALCULATION AND CALIBRATION

The accelerometer and magnetometer sensors on the probe are not perfect, their mounting on the circuit board may not be perfectly aligned and the sensors will be subject to electrical and magnetic influence of other components of the probe hardware. Therefore before the sensor values reported by the probe can be used, they need to be calibrated. Calibration of values is handled by the **ecompass.c** code module. Much work can be saved by using this code. What follows is an explanation of how it works.

3.2.1 CALIBRATING THE SENSOR READINGS

Both the accelerometer and magnetometer measurements are represented as a 3 element vector with each component being an axial reading from the sensor:

$$\vec{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \vec{m} = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}$$

Each of these needs to be passed through its own calibration function $f()$:

$$\vec{a}_{calibrated} = f_a(\vec{a}_{uncalibrated}), \quad \vec{m}_{calibrated} = f_m(\vec{m}_{uncalibrated})$$

The accelerometer calibration function $f_a()$ and the magnetometer calibration function $f_m()$ both have the same form and so the general case is discussed:

$$\vec{v} = f(\vec{u})$$

Where \vec{u} is the uncalibrated sensor value and \vec{v} is the corresponding calibrated sensor value (both are 3 element vectors). $f(\vec{u})$ is a polynomial function in \mathbb{R}_3 as follows:

$$\vec{v} = f(\vec{u}) = \sum_{i=0}^n P_i \begin{bmatrix} u_x^i \\ u_y^i \\ u_z^i \end{bmatrix}$$

Where P_i is the 3 x 3 coefficient matrix for the i^{th} polynomial term.

The order and coefficients of the calibration function are constrained to fit an error model. The zero and first order coefficients will account for linear and rotational offsets, non-orthogonality, hard iron and soft iron effects. A cubic term is sufficient to remove sensor non-linearities and since these have negligible cross-axis effects, the third order coefficient only needs to be a diagonal matrix. The calibration function is therefore a 3rd order polynomial (n=3) determined by 15 parameters as follows:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \vec{v} = f(\vec{u}) = \begin{bmatrix} a_{00} \\ a_{11} \\ a_{22} \end{bmatrix} + \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} + \begin{bmatrix} c_{00} & 0 & 0 \\ 0 & c_{11} & 0 \\ 0 & 0 & c_{22} \end{bmatrix} \begin{bmatrix} u_x^3 \\ u_y^3 \\ u_z^3 \end{bmatrix}$$

Each component of a calibrated sensor measurement can therefore be calculated as follows:

$$\begin{aligned}v_x &= a_{00} + b_{00}u_x + b_{01}u_y + b_{02}u_z + c_{00}u_x^3 \\v_y &= a_{11} + b_{10}u_x + b_{11}u_y + b_{12}u_z + c_{11}u_y^3 \\v_z &= a_{22} + b_{20}u_x + b_{21}u_y + b_{22}u_z + c_{22}u_z^3\end{aligned}$$

An accelerometer and a magnetometer will each have their own set of 15 parameters which together make up the probe's calibration. These parameters are generated by an external application and loaded into the mobile app in the form of an XML file.

3.2.2 TEMPERATURE DEPENDENT CALIBRATION

Unfortunately, the sources of error in a sensor, may change with temperature. The calibration therefore needs to deal with the fact that each of its 15 parameters may be a function of temperature:

$$\begin{aligned}v_x &= a_{00}(t) + b_{00}(t)u_x + b_{01}(t)u_y + b_{02}(t)u_z + c_{00}(t)u_x^3 \\v_y &= a_{11}(t) + b_{10}(t)u_x + b_{11}(t)u_y + b_{12}(t)u_z + c_{11}(t)u_y^3 \\v_z &= a_{22}(t) + b_{20}(t)u_x + b_{21}(t)u_y + b_{22}(t)u_z + c_{22}(t)u_z^3\end{aligned}$$

A linear function of temperature is sufficient to model errors over a reasonable operating range such that each parameter becomes $p(t) = p + p' \cdot t$ and each parameter is now represented by a constant term and a slope with respect to temperature (the prime version of the parameter). The calibration equations therefore become:

$$\begin{aligned}v_x &= (a_{00} + a'_{00} \cdot t) + (b_{00} + b'_{00} \cdot t)u_x + (b_{01} + b'_{01} \cdot t)u_y + (b_{02} + b'_{02} \cdot t)u_z + (c_{00} + c'_{00} \cdot t)u_x^3 \\v_y &= (a_{11} + a'_{11} \cdot t) + (b_{10} + b'_{10} \cdot t)u_x + (b_{11} + b'_{11} \cdot t)u_y + (b_{12} + b'_{12} \cdot t)u_z + (c_{11} + c'_{11} \cdot t)u_y^3 \\v_z &= (a_{22} + a'_{22} \cdot t) + (b_{20} + b'_{20} \cdot t)u_x + (b_{21} + b'_{21} \cdot t)u_y + (b_{22} + b'_{22} \cdot t)u_z + (c_{22} + c'_{22} \cdot t)u_z^3\end{aligned}$$

We now have 2 sets of 15 parameters for each sensor, one set for the temperature dependency constant terms and one for the slopes (the primed version of the parameter).

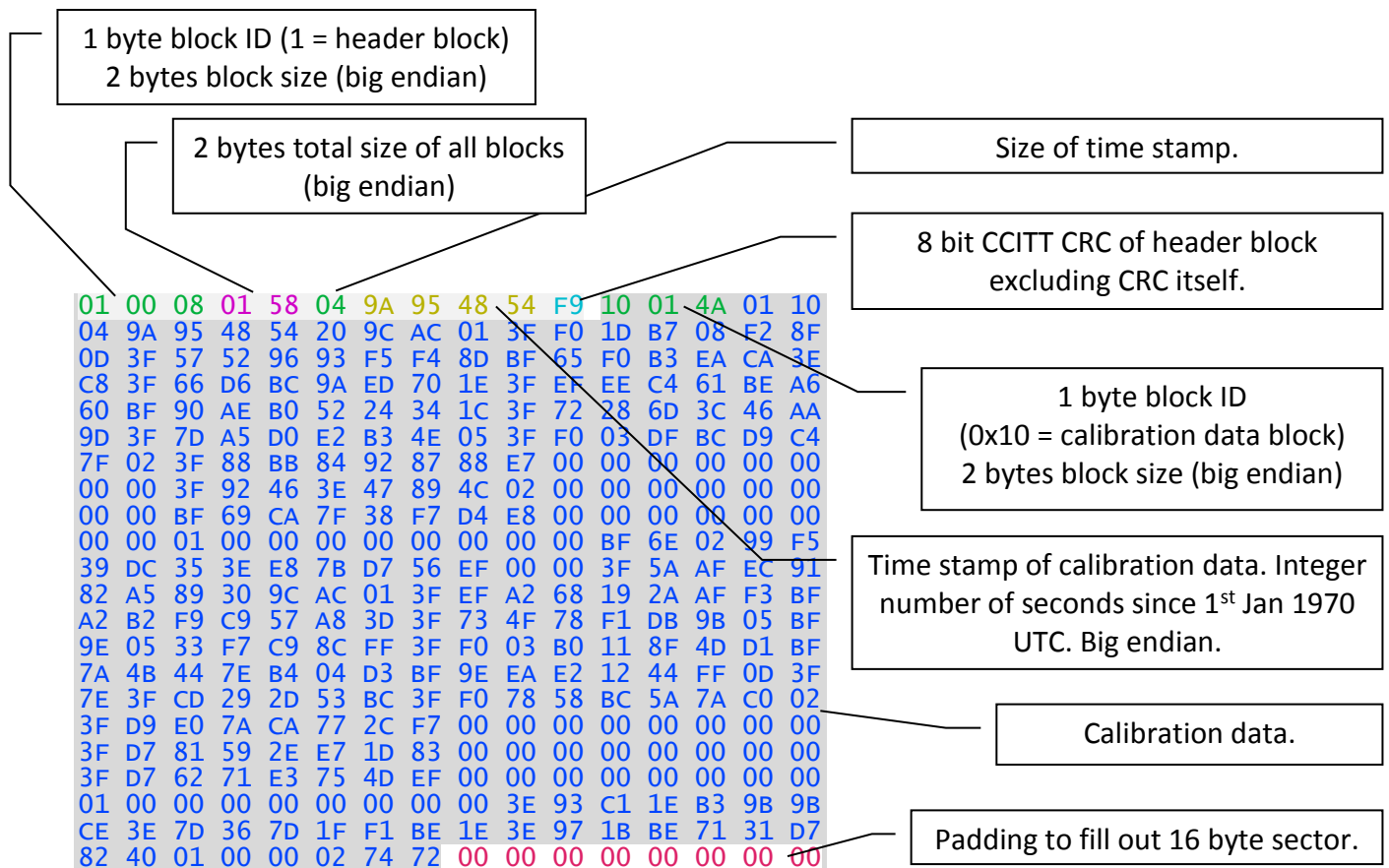
In the iOS version of the app, these calculations are handled by **ecompass.c**. See following sections for details.

3.2.3 CALIBRATION STORAGE ON PROBE

A probe's calibration parameters are normally stored on the probe in a dedicated block of non-volatile memory. The probe will not apply these parameters itself. Instead, the mobile app must fetch the parameters from the probe and apply them to the raw uncalibrated sensor readings that the probe sends to obtain calibrated measurements. The probe memory is arranged into a series of 16 byte sectors. Information is stored in the 3 sectors as a sequence of variable sized blocks. Each block consists of a single byte ID, 2 bytes of size (big endian) followed by data bytes (there are as many data bytes as specified by the size field). The meaning of the data bytes for each block depends on the block ID data (see below). The first block is the header block with ID=0x01. It contains meta data about the

calibration and is designed to fit entirely within the first sector. Checking if the calibration needs to be synced to or from the probe can therefore be done by reading only sector 0 of the probe memory. The header block is followed by the calibration data block with ID=0x10. If the probe memory does not contain a valid header block with a valid CRC, the entire probe memory is considered invalid. When writing configuration data into probe memory, sector 0 is written first as all zeros, then sectors 1 onwards are written and finally sector 0 is written with its correct data. This keeps the probe memory invalidated until all sectors have been successfully written.

A typical example is show below with block fields annotated. Each line is a successive sector of memory. A header block appears first in the memory and is followed by a calibration data block (shaded).



The calibration data (all the dark blue bytes above) are extracted from the probe memory into a buffer which is decoded by the source file named **`ecompass_store.c`** directly into a **`ecompass_t`** structure (**`ecompass_store.c`** can also encode calibration parameters from a **`ecompass_t`** structure into a buffer for storage to probe memory). It is recommended that this module be used to encode and decode calibration data. The source is reasonably well documented with doxygen mark up and can be used as a format reference.

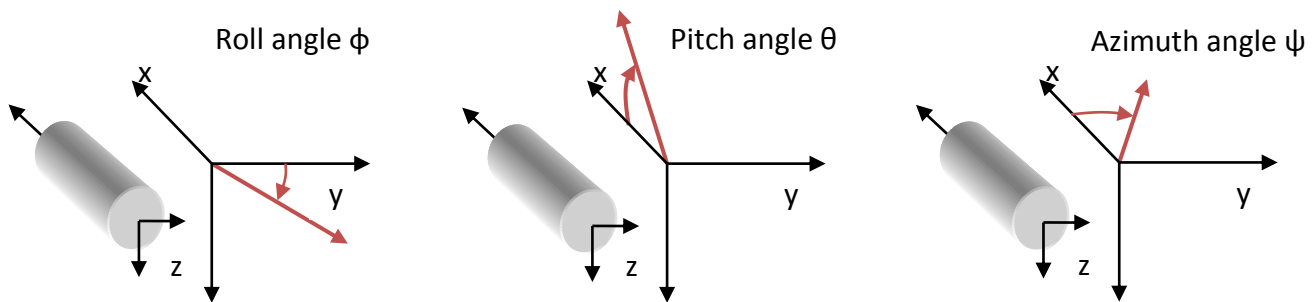
3.2.4 ORIENTATION CALCULATION

The probe follows the "NED" (North, East, Down) co-ordinate convention commonly used in aeronautical applications for. The "absolute" (fixed) reference frame has the X axis pointing North, the Y axis pointing East and the Z axis pointing Down. The Probe's orientation is measured with respect to this absolute reference frame. The accelerometer and magnetometer measurements are with respect to the sensor's reference frame which move as the probe moves. The Probe's sensors are calibrated such that their the X axis points forward along the length of the probe, the Y axis to the right and the Z axis down. This is the Probe's "local" (moving) reference frame.

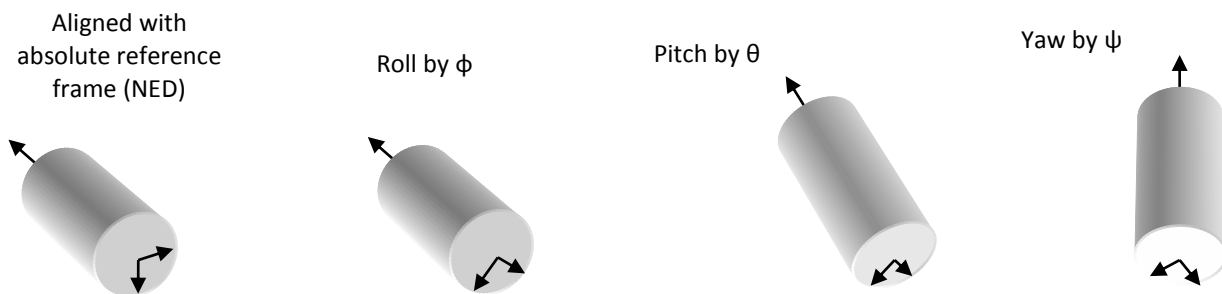
We know what the direction of the Earth's gravitational and magnetic fields are known in the absolute frame and we know what the accelerometer and magnetometer are measuring in the Probe's local frame so we can calculate the Probe's orientation. This is done in terms of three angles:

- Roll ϕ : Rotation around the X axis (Y axis towards Z axis).
- Pitch θ : Rotation around the Y axis (X axis away from Z axis).
- Azimuth ψ : Rotation around the Z axis (X axis towards Y axis). Sometimes called yaw.

These are illustrated below:



The order that these angles are applied is important (i.e. they don't commute). To obtain an orientation of (ϕ, θ, ψ) , you start aligned with the absolute reference frame (i.e. probe lying flat, pointing north). You then roll by ϕ , pitch by θ and finally, yaw by ψ . All the rotations are done in the absolute reference frame as illustrated below:



Given calibrated accelerometer and magnetometer readings:

$$\vec{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \vec{m} = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}$$

The three orientation angles can be calculated as follows.

$$\begin{aligned} \phi &= \tan^{-1} \left(\frac{a_y}{a_z} \right) \\ \theta &= \tan^{-1} \left(\frac{-a_x}{\sqrt{a_y^2 + a_z^2}} \right) \\ \psi &= \tan^{-1} \left(\frac{-m_y \cos \phi + m_z \sin \phi}{m_x \cos \theta + m_y \sin \theta \sin \phi + m_z \sin \theta \cos \phi} \right) \end{aligned}$$

A CoreCam Probe has no magnetometer so it can only calculate and report roll and pitch angles.



3.2.5 USING THE ECOMPASS.C CODE TO DO ALL THE HARD WORK

The source code for the iOS version has a file called **ecompass.c** (and associated **ecompass.h**) which implements most of the functionality described so far in this section. A companion file called **ecompass_store.c** provides functions to encode and decode calibration parameters to and from compact binary forms. These are used to package up calibrations for storage in the probe's limited memory and to unpack them when retrieved. These files are reasonably well commented (with doxygen markup) and are written in standard C so should be easily portable to other languages. Much work can be saved by incorporating this code.

Using **ecompass.c** requires the following steps:

1. Declare a variable of type **ecompass_t** and initialise it using **ecompass_init(ecompass_t *)**. **ecompass_t** is a structure that defines the context of an electronic compass and is a required parameter to most functions in **ecompass.c**. It must be initialised before it is used. Obviously, each active probe will need its own context.
2. Configure the **ecompass_t** structure by setting the calibration parameters for the accelerometer and magnetometer if applicable. If no magnetometer is available (e.g. when using a CoreCam) then no magnetometer calibration parameters are needed.
3. Pass sensor readings on to the **ecompass_t** structure as they become available from the probe. Any or all of the following may be given to **ecompass_t**.
Uncalibrated accelerometer (x, y & z along each axis).
Uncalibrated accelerometer temperature.
Uncalibrated magnetometer (x, y & z along each axis).
Uncalibrated magnetometer temperature.
Uncalibrated probe temperature.
The uncalibrated sensor values will be immediately calibrated if the associated calibration parameters have been set. If at this stage, a calibrated accelerometer measurement is available, the roll and pitch angles are calculated. Furthermore, if a calibrated magnetometer measurement is available, the azimuth angle is also calculated. All of the uncalibrated and calibrated values along with the orientation angles are stored in the **ecompass_t** structure for later retrieval.

Two roll angles are calculated for each orientation. The first is the "real" roll and is simply called **roll**. It is the roll measured with respect to the calibrated reference frame and is used in all calculations. Since the probe is rotationally symmetric about its X axis, the orientation of zero roll does not matter to the operator, as long as it is consistent. A second roll is therefore provided called **roll_for_display**. The operator may position the probe arbitrarily and zero the roll to that orientation. A zero offset is then stored in the **ecompass_t** structure and **roll_for_display** is calculated as **roll** subtract this offset. Furthermore, **roll_for_display** can be converted from the usual range of [0, 360) to (-180, +180]. **roll_for_display** is used as the displayed value in measurements, on the real time pages and in report files.



An **ecompass_t** structure can also report on the stability of sensor measurements. It does this by buffering the last few readings for each sensor. For each sensor (accelerometer and magnetometer) it analyses each of the 3 axial components (x, y and z) and the magnitude (the vector sum of x, y and z components) giving a total of 8 components. For each of these components, an **ecompass_t** structure will calculate the mean and variance over the last few samples and identify the value that deviates most from the mean. This is called the maximum deviation. Some of these values are displayed on the real time sensor page. Any of these calculated values can be obtained from a **ecompass_t** structure using the **ecompass_noise_metric(...)** function. This function can also return a flag indicating that the specified value is stable (i.e. its maximum deviation is within acceptable limits). The calculations can be configured to be performed over a recent number of measurements or a time interval using:

ecompass_noise_metric_enable_number(...)

The number of measurements or duration can be configured using:

ecompass_noise_metric_number(...)

ecompass_noise_metric_duration(...)

By default these calculations are done over the most recent 5 measurements.

Setting calibration parameters in the **ecompass_t** structure:

Normally, the calibration values are retrieved from the probe's memory as a single block of encoded data. This is decoded using the function **ecompass_store_get(ecompass_t *, const void *, size_t)** which extracts the calibration parameters from the data buffer and puts them into the **ecompass_t** structure. An associated **ecompass_store_put(ecompass_t *, void *, size_t)** function will encode an **ecompass_t** structure's calibration parameters into a data buffer. The function **ecompass_store_space_required(ecompass_t *)** will tell you how big the buffer needs to be to encode the parameters.

Alternatively, the parameters may be set directly by manually populating a **ecompass_calibration_t** structure then passing a pointer to

ecompass_calibration_set_accelerometer(ecompass_t *, const ecompass_calibration_t *) or
ecompass_calibration_set_magnetometer(ecompass_t *, const ecompass_calibration_t *)

which will copy the parameters to the **ecompass_t** structure for the specified sensor, ready to be used. The associated "get" functions can be used to copy the parameters out of the **ecompass_t** structure if required. This is the method used by the iOS version of the app when loading a calibration from XML file.

Several additional configuration options exist for an **ecompass_t** structure (specifically, roll stabilisation and uncompensated azimuth enable/disable) but these were used only during testing and are not used in production software. These should be left at their default values.

Passing sensor readings on to the **ecompass_t** structure:

As each "shot" is received from the probe (either in response to a request for a particular measurement after a survey or as a spontaneous notification of a real-time reading), the uncalibrated sensor values are extracted from the shot data and passed onto the probe's **ecompass_t** structure. This in turn, calibrates them with its current parameters (if possible), and calculates which ever orientation angles it can based on what calibrated measurements are available. The results are stored in the structure and



flagged as to what is valid and what is not. This is all handled in the **processShotData:** method of the **Probe** class in **Probe.m** (at around line 2625). Basically it does the following:

- Invalidates previous values stored in the structure using:
`ecompass_accelerometer_and_magnetometer_invalidate(...)` and
`ecompass_accelerometer_and_magnetometer_invalidate(...)` and
- Passes the uncalibrated sensor data using one or more of the following functions depending on what was available in the shot data:
`ecompass_accelerometer_and_magnetometer_reading(...)`
`ecompass_accelerometer_reading(...)`
`ecompass_magnetometer_reading(...)`
`ecompass_temperature_reading(...)`

The results will be immediately available as the following structure elements:

<code>ecompass.accelerometer.is_valid</code>	Has an accelerometer reading
<code>ecompass.accelerometer.is_calibrated</code>	The reading is calibrated
<code>ecompass.accelerometer.t_is_valid</code>	The temperature for the reading was available.
<code>ecompass.accelerometer.x</code>	Calibrated X axis value (if <code>is_valid</code> and <code>is_calibrated</code>)
<code>ecompass.accelerometer.y</code>	Calibrated Y axis value (if <code>is_valid</code> and <code>is_calibrated</code>)
<code>ecompass.accelerometer.z</code>	Calibrated Z axis value (if <code>is_valid</code> and <code>is_calibrated</code>)
<code>ecompass.accelerometer.uncalibrated_x</code>	Uncalibrated X axis value (if <code>is_valid</code>)
<code>ecompass.accelerometer.uncalibrated_y</code>	Uncalibrated Y axis value (if <code>is_valid</code>)
<code>ecompass.accelerometer.uncalibrated_z</code>	Uncalibrated Z axis value (if <code>is_valid</code>)
<code>ecompass.accelerometer.t</code>	Uncalibrated sensor temperature (if <code>t_is_valid</code>)
<code>ecompass.magnetometer.is_valid</code>	Has a magnetometer reading
<code>ecompass.magnetometer.is_calibrated</code>	The reading is calibrated
<code>ecompass.magnetometer.t_is_valid</code>	The temperature for the reading was available.
<code>ecompass.magnetometer.x</code>	Calibrated X axis value (if <code>is_valid</code> and <code>is_calibrated</code>)
<code>ecompass.magnetometer.y</code>	Calibrated Y axis value (if <code>is_valid</code> and <code>is_calibrated</code>)
<code>ecompass.magnetometer.z</code>	Calibrated Z axis value (if <code>is_valid</code> and <code>is_calibrated</code>)
<code>ecompass.magnetometer.uncalibrated_x</code>	Uncalibrated X axis value (if <code>is_valid</code>)
<code>ecompass.magnetometer.uncalibrated_y</code>	Uncalibrated Y axis value (if <code>is_valid</code>)
<code>ecompass.magnetometer.uncalibrated_z</code>	Uncalibrated Z axis value (if <code>is_valid</code>)
<code>ecompass.magnetometer.t</code>	Uncalibrated sensor temperature (if <code>t_is_valid</code>)
<code>ecompass.temperature.is_valid</code>	Has a temperature reading
<code>ecompass.temperature.is_calibrated</code>	The reading is calibrated
<code>ecompass.temperature.value</code>	Calibrated probe temperature (if <code>is_valid</code> and <code>s_calibrated</code>)
<code>ecompass.temperature.uncalibrated_value</code>	Uncalibrated probe temperature (if <code>is_valid</code>)
<code>ecompass.orientation.roll_is_valid</code>	The roll angle has been calculated
<code>ecompass.orientation.pitch_is_valid</code>	The pitch angle has been calculated
<code>ecompass.orientation.azimuth_is_valid</code>	The azimuth angle has been calculated
<code>ecompass.orientation.roll</code>	The roll angle in radians (if <code>roll_is_valid</code>)
<code>ecompass.orientation.pitch</code>	The pitch angle in radians (if <code>pitch_is_valid</code>)
<code>ecompass.orientation.azimuth</code>	The azimuth angle in radians (if <code>azimuth_is_valid</code>)
<code>ecompass.orientation.roll_for_display</code>	The roll angle with the manual zero offset applied.

Manually zeroing the roll:

The `ecompass_set_zero_offsets(...)` function can be used to set the zero offset for roll either automatically based on current orientation or explicitly to a specified value. Setting to zero effectively removes the offset so that `roll` and `roll_for_display` once again become aligned with each other. The `ecompass_get_zero_offsets(...)` function returns the current value of the roll zero offset.

To set the range of `roll_for_display` to $(-180, +180]$, set the global variable `recompass_roll_for_display_range_is_symmetric` to 1. Its default value of 0 ensures that `roll_for_display` is restricted to the range $[0, 360)$.

3.3 PROBE CONNECTION SEQUENCE

After a connection is established to a probe, the mobile app goes through a sequence of tasks before the probe can be used. These are listed below:

- **Interrogating (features)**
This is the Bluetooth LE service and characteristic discovery phase. All the characteristics of the connected device are identified. If the connected device does not have all of the expected services and characteristics of an orientation probe, the operator is notified that the device is not an orientation probe.
- **Interrogating (configuration)**
If the previous phase succeeds, the app will attempt to read the value of the following characteristics. These values are used during the operation of the probe.
 - Device ID (0xFFF1)
 - Probe Mode (0xFFF2)
 - Shot Interval (0xFFF3)
 - Survey Max Shots (0xFFF6)
 - Device Address (0xFFFB)
 - Firmware Version Major (0xFFFC)
 - Firmware Version Minor (0xFFFD)
 - Rolling Shot Interval (0xFFFE)
- **Checking calibration.**
The first sector (16 bytes) of the probe memory are read and inspected for a valid header. This header contains the time stamp of the calibration data saved in the probe. This is checked against the calibration data that the app may have stored locally for the probe. The calibration data is then synced to or from the probe based on the validity and timestamp of the calibration data in the probe and app as follows.
 - Neither probe nor app are valid:** No further action is taken. Probe is not calibrated.
 - Probe is valid and app is not valid:** Probe's calibration is fetched, saved and used by the app.
 - Probe is not valid and app is valid:** The probe's calibration memory is updated from the app.
 - Both probe and app are valid and time stamps are the same :** No further action is taken. Probe is calibrated and app has correct calibration parameters.



Both probe and app are valid but probe has more recent time stamp: App's copy of the calibration parameters are out of date. Probe's calibration is fetched and saved to app.

Both probe and app are valid but app has more recent time stamp: Probe's copy of the calibration parameters are out of date. Probe's calibration is updated from the app.

- **Fetching calibration.**

After checking calibration, this state signifies that calibration parameters are being fetched from the probe to be saved by the app. Sectors 1 onwards are read since sector 0 has already been fetched during then "Checking Calibration" phase. Sector 0 contains the total size of the calibration data as saved in the probe so the number of sectors that must be read can be easily calculated.

- **Updating calibration.**

After checking calibration, this state signifies that the app's calibration parameters are being written to probe memory. To prevent corruption due to incomplete or interrupted write sequence, this is done as follows:

Sector 0 of probe memory is first written with all zeros.

Sectors 1 onwards are then written with their correct data.

Finally sector 0 is written with correct data.

This effectively invalidates the probe memory at the start of the update and re-validates it when the update completes. If an update is interrupted, any subsequent connect will re-attempt the update as a part of the "Checking Calibration" step.



4 FILE FORMATS

Several different file types are used by the application. File names start with a prefix that identifies the type. The remainder of the file name can be anything but default names usually include the date and some other identifying information depending on the file type such as probe ID or hole number. On the iOS version of the App, files are usually only displayed a single type at a time with the prefix and extension stripped off. For example, **SurveyReport-Site123-20140822.csv** is usually displayed in a survey report file list as "**Site123-20140822**" (it saves space on a narrow screen).

Type	Prefix	Format	Extension	Description
Configuration	Configuration-	Property List	.plist	Stores the app's persistent state.
Survey State	SurveyState-	XML	.xml	Stores the state of each survey so that it may be resumed at a later time.
Survey Report	SurveyReport-	CSV	.csv	Stores information about a survey (the measurements in particular) in a human readable format that is easy to import into a spreadsheet.
Calibration	Calibration-	XML	.xml	Stores the probe's calibration parameters.
Sensor Data	SensorData-	CSV	.csv	Stores on-demand sensor values (when operator presses "save" button on real-time sensor values page) for diagnostic purposes. Not used in normal operation.
Calibration Data	CalibrationData-	CSV	.csv	This file is not used in the android version of the app. It is a by-product of older calibration procedures and stored data used for calibrating a probe. It no longer serves any purpose.

Details for each of these file types are given in the following sections.

4.1 CONFIGURATION FILE

The configuration file stores the current state of the application and all of its settings. It allows the application to continue where it left off if it is ever stopped and re-started. This file is only used internally by the app and therefore its format and contents are entirely up to the app. The iOS app uses a configuration file called **Configuration-Probes.plist** which contains the values of all the options from the setup page and information about the three possible probes associated with the app (the probe when used in bore orientation mode, the black probe and the white probe when in core orientation mode). Each of these probes saves the following information to the configuration file:

- Name
- Device Address (MAC address)
- Serial number (as a string).
- Colour. This identifies how the probe is used by the app. Its value may be Unassigned (the probe is not assigned for any purpose), Black (the probe is the black probe in core orientation mode), White (the probe is the white probe in core orientation mode), Gold (the probe used in bore orientation mode).
- Bluetooth peripheral UUID. This is an iOS specific value that uniquely associates a Bluetooth LE peripheral as defined by the iOS Bluetooth API with a probe object. This will probably be different for the Android app.



- The state of the current survey for the probe. The survey state contains the same information as outlined in the Survey State file below.
- The calibration parameters for the probe.

Refer to the iOS app source code in the file **Suntech.m**. The methods **saveToFile:** and **loadFromFile:** illustrate the exact data saved by the iOS app. These indirectly call **initWithCoder:** and **encodeWithCoder:** in **Suntech.m** to handle data saved to the configuration file specifically for each probe.

The iOS app also has a secondary configuration file called **Configuration-Probes.plist** which stores a list of all Bluetooth LE peripheral devices known to the app. This is used when selecting probes associated with the app. A scan list of devices is displayed to the operator based on Bluetooth LE scan results. This list is initially populated with any known devices that may already be known to the app (the names appear black). As devices are seen by the scan, their names are turned green if they are already in the list or added to the list (in green) if they are not. This feature allows an operator to assign a probe that may not be currently in range.

4.2 SURVEY STATE

The survey state file stores the state of a survey so that it may be resumed at a later time. This file is only used internally by the application and so its contents and format are entirely up to the application. The iOS version of the app, uses an XML format for the survey state file with the following data:

- 1) Hole ID as input by the operator.
- 2) Company name as input by the operator.
- 3) Operator name as input by the operator.
- 4) Start time and date of survey.
- 5) A list of measurements so far taken. The information saved for each measurement consists of:
 - a) The measurement name/number/sub-number (e.g. "Measurement 1", "Measurement 2A", "Measurement 2B", etc...)
 - b) The time stamp.
 - c) The roll, pitch and azimuth orientation angles as well as the probe temperature. This information is present only for measurements that have been fetched from the probe. Without the probe's sensor reading for a measurement the orientation cannot be known.
 - d) The depth of the measurement.
 - e) The magnitudes of the sensor reading from the accelerometer and magnetometer (if available).
- 6) A list of internal state variables required to track the progress of the survey. These are implementation dependent but would typically include things such as:
 - a) The current depth interval
 - b) How to calculate the next depth. Was it explicitly set by the operator or deduced by the depth interval? Is the operator repeating a measurement at the current depth? What direction is the probe going down the hole (in or out)?
 - c) The next measurement number and sub-number.

Refer to the implementation for the iOS app in the file **ProbeSurvey.m**. The methods **saveStateToXMLFile:** and **appendSurveyToXMLWriter:** illustrate the exact state data saved by the iOS app.



4.3 SURVEY REPORT

The survey report file is generated each time the survey state changes. It contains a subset of the data in the survey state file in a more human readable format (CSV). It may be copied from the mobile device onto a PC or emailed to a recipient. The file consists of a title, some header information identifying the company and the operator followed by a list of survey measurements. Each survey measurement has the following fields: Probe ID, hole, measurement number, depth, date, time, data integrity indicators, azimuth, dip, roll, total magnetometer magnitude (in nano Tesla) and temperature. The data integrity indicators are a set of symbols that reflect the quality of measurement. These are listed in the legend that makes up the final part of the survey report file. Creating survey report files is handled in the **saveReportToCSVFile**: method in the **ProbeSurvey.m** file of the source code. A sample file is shown below:

Camteq - Precision Instrumentation

"Company Name", My Company
"Operator Name", Driller Dan

```
Probe ID,Hole,Measurement,Depth,Date,Time,DIntegrity,Azimuth,Dip,Roll,TotMag,Temperature
04:1F:0F,Sample Hole,1,100,2015-03-04,11:41:02,,180.88,69.49,1.54,48423.1,21.5
04:1F:0F,Sample Hole,2A,110,2015-03-04,11:41:31,,178.59,65.76,7.44,48932.7,21.8
04:1F:0F,Sample Hole,2B,110,2015-03-04,11:41:54,,178.61,65.75,7.43,48920.2,22.0
04:1F:0F,Sample Hole,3,120,2015-03-04,11:42:23,,174.29,63.18,11.45,49178.6,22.0
```

DIntegrity Legend:

D*,Dip outside expected range.
M*,Total magnetic field outside expected range.
A*,Azimuth outside expected range.
#,Probe outside operating temperature.
!,Probe moving during survey shot.

4.4 CALIBRATION FILE FORMAT

Calibration files are normally only used during factory set up after manufacturing. The calibration parameters are calculated on a PC. The resulting calibration can then be copied onto the mobile device which then uploads the calibration parameters to the probe using the connect sequence to check the calibration and update the probe memory. Any other mobile device running the same app can then retrieve the calibration parameters from the probe in order to process its sensor values.

The calibration file is in XML format and the parsing in the iOS application is done by the file **XMLParseCalibration.m**. It implements the **XMLParseCalibration** class which uses a native **NSXMLParser** to do all the hard work by setting itself up as the **NSXMLParser** delegate. This file may be used as a reference.

The file has a root node of **<probe-calibration>**. This may have an optional "record keeping" attribute called **probe-id** that identifies the probe that the calibration belongs to. Such record keeping attributes do not affect the calibration and exist primarily as information for technicians. The probe id can be set to the device address of the probe. As a safety mechanism, if a probe id attribute exists in a calibration, and the Suntech app is asked to load it for a probe that has a different device address, the operator will

be informed and asked if they want to proceed anyway. The calibration file will be loaded, no questions asked, if it has no probe id attribute. For example:

```
<probe-calibration probe-id="90:59:AF:04:20:88">
```

Under the **<probe-calibration>** node are multiple **<sensor>** sub-nodes, one each for the accelerometer and the magnetometer. The **<sensor>** nodes contain the actual calibration parameters for a sensor. The only required attribute is **type** which should be either **accelerometer** or **magnetometer**. An optional record keeping attribute **timestamp** may be added to record the time the calibration was performed. For example:

```
<sensor type="accelerometer">
```

The file needs to store the calibration parameters for each sensor. Recall that the calibration function of each sensor is a third order polynomial in \mathbb{R}_3 which means that each polynomial coefficient is in general a 3x3 matrix. A sensor node will therefore contain a list of **<matrix>** nodes, each associated with a specific power of the polynomial and containing the coefficient matrix of that power term. There will usually be 3 such matrices under a **<sensor>** node, one for the zero order term, one for the first order term and one for the third order term. If a matrix is omitted it is assumed not used (i.e. set to the identity matrix in the case of the first order term or the zero matrix for other terms). When temperature compensation is used, each calibration parameter is no longer a constant, but a polynomial function of temperature. In practice, this will be a first order polynomial (a linear function) as illustrated below:

$$\begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} \xrightarrow{\text{with temperature compensation}} \begin{bmatrix} b_{00}(t) & b_{01}(t) & b_{02}(t) \\ b_{10}(t) & b_{11}(t) & b_{12}(t) \\ b_{20}(t) & b_{21}(t) & b_{22}(t) \end{bmatrix} = \begin{bmatrix} b_{00} + b'_{00}t & b_{01} + b'_{01}t & b_{02} + b'_{02}t \\ b_{10} + b'_{10}t & b_{11} + b'_{11}t & b_{12} + b'_{12}t \\ b_{20} + b'_{20}t & b_{21} + b'_{21}t & b_{22} + b'_{22}t \end{bmatrix}$$

When a calibration function is dependent on temperature then each coefficient is stored as multiple matrices, each containing a power of temperature:

$$\begin{bmatrix} b_{00} + b'_{00}t & b_{01} + b'_{01}t & b_{02} + b'_{02}t \\ b_{10} + b'_{10}t & b_{11} + b'_{11}t & b_{12} + b'_{12}t \\ b_{20} + b'_{20}t & b_{21} + b'_{21}t & b_{22} + b'_{22}t \end{bmatrix} \xrightarrow{\text{stored as}} \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}, \begin{bmatrix} b'_{00}t & b'_{01}t & b'_{02}t \\ b'_{10}t & b'_{11}t & b'_{12}t \\ b'_{20}t & b'_{21}t & b'_{22}t \end{bmatrix}$$

Therefore, in the calibration file, each matrix is a 3x3 array of constants associated with a particular power of the measurement and a particular power of temperature.



A **<matrix>** has the following attributes:

Attribute	Description
name	The name of the matrix. Usually "0th-order", "1st-order" or "3rd-order"
rows	The number of rows in the matrix. This is usually 3.
columns	The number of columns in the matrix. This is usually either 3 or 1 (see explanation below).
measurement-power	This is the polynomial power of the calibration function that the matrix corresponds to.
temperature-power	This is the power of temperature (0 or 1) that the matrix corresponds to. If not specified, the power is assumed to be 0 (i.e. the constant term). If there is no temperature compensation in the calibration, then this is omitted (or set to 0)

A matrix contains a list of **<matrix-element>** nodes, specifying the values appearing as elements in the matrix. Each **<matrix-element>** has a row and a column attribute that identify the element and a value attribute that specifies the value. For example:

```
<matrix-element row="0" column="1" value="0.000123"/>
```

Strictly speaking, a coefficient matrix of the calibration function will be 3x3. In practice, the zero order and third order matrices are diagonal and can be specified more compactly with 3 parameters, instead of 9. A coefficient matrix specified as a column vector (a 3x1 matrix) is interpreted as a diagonal matrix with the vector elements along the diagonal. This saves a little space in the calibration file and makes it slightly easier to read. For example, the following two matrix encodings are identical (either is acceptable):

```
<matrix name="3rd-order" rows="3" columns="3" measurement-power="3">
  <matrix-element row="0" column="0" value="8.61142717484949e-08"/>
  <matrix-element row="0" column="1" value="0"/>
  <matrix-element row="0" column="2" value="0"/>
  <matrix-element row="1" column="0" value="0"/>
  <matrix-element row="1" column="1" value="2.53132971002569e-07"/>
  <matrix-element row="1" column="2" value="0"/>
  <matrix-element row="2" column="0" value="0"/>
  <matrix-element row="2" column="1" value="0"/>
  <matrix-element row="2" column="2" value="1.27570641475147e-07"/>
</matrix>
```

and

```
<matrix name="3rd-order" rows="3" columns="1" measurement-power="3">
  <matrix-element row="0" column="0" value="8.61142717484949e-08"/>
  <matrix-element row="1" column="0" value="2.53132971002569e-07"/>
  <matrix-element row="2" column="0" value="1.27570641475147e-07"/>
</matrix>
```



Several “record keeping” nodes and attributes may appear throughout the XML file. They can safely be ignored by the app and serve only to document the conditions of the calibration for possible future diagnostics. These include such things as:

- A **<calibration-options>** node under the **<probe-calibration>** root node, recording all the calibration options used.
- A **<temperature>** node under each **<sensor>** node recording the temperature statistics for the calibration.
- A **<calibration-errors>** node under each **<sensor>** node recording the accuracy of the calibration.
- A **pre-subtracted** attribute in some **<matrix>** nodes. This was used during prototyping/testing and should be ignored.

A sample calibration file is shown below. Its accelerometer is temperature compensated but the magnetometer is not illustrating each case (the only difference is the presence of **temperature-power** attributes in the **<matrix>**).

```
<probe-calibration>
  <sensor type="accelerometer">
    <temperature min="-33.00" max="-1.00" mean="-13.26" std-dev="90.2409" est-ambient="-7.96"/>
    <matrix name="0th-order" rows="3" columns="1" measurement-power="0" temperature-power="0" pre-subtracted="0">
      <matrix-element row="0" column="0" value="0.00922436778135957"/>
      <matrix-element row="1" column="0" value="0.0138839410781416"/>
      <matrix-element row="2" column="0" value="-0.00334982888883995"/>
    </matrix>
    <matrix name="0th-order" rows="3" columns="1" measurement-power="0" temperature-power="1" pre-subtracted="0">
      <matrix-element row="0" column="0" value="-2.84702833120381e-05"/>
      <matrix-element row="1" column="0" value="-4.00931154711463e-05"/>
      <matrix-element row="2" column="0" value="4.82685966037705e-05"/>
    </matrix>
    <matrix name="1st-order" rows="3" columns="3" measurement-power="1" temperature-power="0">
      <matrix-element row="0" column="0" value="1.01102439065558"/>
      <matrix-element row="0" column="1" value="-0.0041781952176493"/>
      <matrix-element row="0" column="2" value="-0.0056954543135021"/>
      <matrix-element row="1" column="0" value="0.00322561974718827"/>
      <matrix-element row="1" column="1" value="1.00800697817515"/>
      <matrix-element row="1" column="2" value="-0.0421659140248752"/>
      <matrix-element row="2" column="0" value="0.00263380869761858"/>
      <matrix-element row="2" column="1" value="0.0415035855950001"/>
      <matrix-element row="2" column="2" value="1.00753645079371"/>
    </matrix>
    <matrix name="1st-order" rows="3" columns="3" measurement-power="1" temperature-power="1">
      <matrix-element row="0" column="0" value="0.000158605502525686"/>
      <matrix-element row="0" column="1" value="-1.93591955912653e-06"/>
      <matrix-element row="0" column="2" value="-5.65315321796496e-06"/>
      <matrix-element row="1" column="0" value="1.00518551213619e-05"/>
      <matrix-element row="1" column="1" value="-0.000190114867488019"/>
      <matrix-element row="1" column="2" value="2.27762971744859e-05"/>
      <matrix-element row="2" column="0" value="6.05136414208003e-06"/>
      <matrix-element row="2" column="1" value="-3.5100388846495e-05"/>
      <matrix-element row="2" column="2" value="0.000165288027619523"/>
    </matrix>
    <matrix name="3rd-order" rows="3" columns="3" measurement-power="3" temperature-power="0">
      <matrix-element row="0" column="0" value="0.00690178715196384"/>
      <matrix-element row="0" column="1" value="0"/>
      <matrix-element row="0" column="2" value="0"/>
      <matrix-element row="1" column="0" value="0"/>
      <matrix-element row="1" column="1" value="0.00046264982116143"/>
      <matrix-element row="1" column="2" value="0"/>
      <matrix-element row="2" column="0" value="0"/>
      <matrix-element row="2" column="1" value="0"/>
      <matrix-element row="2" column="2" value="-0.000945181459904015"/>
    </matrix>
    <matrix name="3rd-order" rows="3" columns="3" measurement-power="3" temperature-power="1">
      <matrix-element row="0" column="0" value="-0.00017816179990777"/>
      <matrix-element row="0" column="1" value="0"/>
      <matrix-element row="0" column="2" value="0"/>
      <matrix-element row="1" column="0" value="0"/>
      <matrix-element row="1" column="1" value="0.000162543959086292"/>
      <matrix-element row="1" column="2" value="0"/>
      <matrix-element row="2" column="0" value="0"/>
      <matrix-element row="2" column="1" value="0"/>
      <matrix-element row="2" column="2" value="-0.000135631357748656"/>
    </matrix>
  </sensor>
  <calibration-errors>
    <calibration-error name="magnitude" rms="0.00108138506441144" max="0.00309104124611803"/>
    <calibration-error name="roll" rms="0.191761903078201" max="0.554025991992091"/>
  </calibration-errors>
</probe-calibration>
```



```
<calibration-error name="pitch" rms="0.0643603176222847" max="0.256476299658047"/>
<calibration-error name="azimuth" rms="0.241238119323651" max="0.851444771340226"/>
</calibration-errors>
</sensor>
<sensor type="magnetometer">
  <temperature min="-8.00" max="-7.00" mean="-7.96" std-dev="0.0393"/>
  <matrix name="0th-order" rows="3" columns="1" measurement-power="0" pre-subtracted="0">
    <matrix-element row="0" column="0" value="0.578478315166017"/>
    <matrix-element row="1" column="0" value="0.282434254295653"/>
    <matrix-element row="2" column="0" value="0.0419574113042802"/>
  </matrix>
  <matrix name="1st-order" rows="3" columns="3" measurement-power="1">
    <matrix-element row="0" column="0" value="0.958263284215712"/>
    <matrix-element row="0" column="1" value="0.0175101043794852"/>
    <matrix-element row="0" column="2" value="-0.00216565260485913"/>
    <matrix-element row="1" column="0" value="-0.0282548127313769"/>
    <matrix-element row="1" column="1" value="0.902631554683952"/>
    <matrix-element row="1" column="2" value="-0.0544318169822898"/>
    <matrix-element row="2" column="0" value="0.0643341962382862"/>
    <matrix-element row="2" column="1" value="-0.0181522403626641"/>
    <matrix-element row="2" column="2" value="0.914235489256867"/>
  </matrix>
  <matrix name="3rd-order" rows="3" columns="3" measurement-power="3">
    <matrix-element row="0" column="0" value="7.25335784541496e-08"/>
    <matrix-element row="0" column="1" value="0"/>
    <matrix-element row="0" column="2" value="0"/>
    <matrix-element row="1" column="0" value="0"/>
    <matrix-element row="1" column="1" value="-1.27167957792254e-07"/>
    <matrix-element row="1" column="2" value="0"/>
    <matrix-element row="2" column="0" value="0"/>
    <matrix-element row="2" column="1" value="0"/>
    <matrix-element row="2" column="2" value="5.72398595764903e-07"/>
  </matrix>
  <calibration-errors>
    <calibration-error name="magnitude" rms="0.0333086942331575" max="0.0800739061353255"/>
    <calibration-error name="roll" rms="0.191761903078201" max="0.554025991992091"/>
    <calibration-error name="pitch" rms="0.0643603176222847" max="0.256476299658047"/>
    <calibration-error name="azimuth" rms="0.241238119323651" max="0.851444771340226"/>
  </calibration-errors>
</sensor>
<calibration-options>
  <option name="UsePreSubtractedAccelerometerOffset" type="boolean" value="1"/>
  <option name="AccelerometerAlgorithm" type="enum(1s-full-auto,1s-full-auto-0az,1s-6-aa,1s-14-aa,ellipsoid-fit)"
value="1s-full-auto"/>
  <option name="AccelerometerLeastSquaresPowerTerms" type="enum(simple,cubic,cubic-diagonal,square-cubic,square-
cubic-diagonal)" value="cubic-diagonal"/>
  <option name="AccelerometerTemperaturePolyTerms" type="integer" value="2"/>
  <option name="UsePreSubtractedMagnetometerOffset" type="boolean" value="1"/>
  <option name="MagnetometerAlgorithm" type="enum(1s-full-auto,1s-full-auto-0az,1s-6-aa,1s-14-aa,ellipsoid-fit)"
value="1s-full-auto"/>
  <option name="MagnetometerLeastSquaresPowerTerms" type="enum(simple,cubic,cubic-diagonal,square-cubic,square-
cubic-diagonal)" value="cubic-diagonal"/>
  <option name="MagnetometerAlignmentType" type="enum(none,embedded,independent,implicit)" value="embedded"/>
  <option name="MagnetometerTemperaturePolyTerms" type="integer" value="2"/>
  <option name="MagnetometerCalibrationFieldMagnitude" type="float" value="58.2778"/>
  <option name="MagnetometerCalibrationFieldInclination" type="float" value="-65.73"/>
  <option name="MagnetometerMinTemperatureRange" type="float" value="10"/>
  <option name="OrientationToleranceRadians" type="float" value="0.0174532925199433"/>
  <option name="PitchLimitForErrorCalculation" type="float" value="85"/>
  <option name="MagnetometerTemperatureCompensationMinSamples" type="integer" value="100"/>
  <option name="CalibrateOldSkool" type="boolean" value="1"/>
</calibration-options>
</probe-calibration>
```

4.5 SENSOR DATA

The sensor data file is not normally used in operational situations. It stores a list of real-time sensor values sent by the probe in response to the operator tapping the "Save" button on the real-time sensor page. The sensor data file is used primarily in the factory for diagnostic or testing purposes. It takes the form of a CSV file with each record containing the following fields:

- **Num:** The record number in the file.
- **Time:** The Date & time in ISO 8601 format.
- **Roll:** The roll angle of the orientation.
- **Pitch:** The pitch angle of the orientation.
- **Azimuth:** The azimuth angle of the orientation.
- **Acc-X:** Calibrated accelerometer X axis reading.
- **Acc-Y:** Calibrated accelerometer Y axis reading.



- **Acc-Z:** Calibrated accelerometer Z axis reading.
- **Acc-T:** Accelerometer temperature.
- **Mag-X:** Calibrated magnetometer X axis reading.
- **Mag-Y:** Calibrated magnetometer Y axis reading.
- **Mag-Z:** Calibrated magnetometer Z axis reading.
- **Mag-T:** Magnetometer temperature.
- **Probe-T:** Probe temperature.
- **UncAcc-X:** Uncalibrated accelerometer X axis reading.
- **UncAcc-Y:** Uncalibrated accelerometer Y axis reading.
- **UncAcc-Z:** Uncalibrated accelerometer Z axis reading.
- **UncMag-X:** Uncalibrated magnetometer X axis reading.
- **UncMag-Y:** Uncalibrated magnetometer Y axis reading.
- **UncMag-Z:** Uncalibrated magnetometer Z axis reading.

Saving sensor data to file is handled in the **saveBeasurementInitiatedAutomatically:** method in the **SensorViewController.m** file of the source code. An example of a sensor data file is shown below:

```
Num,Time,Roll,Pitch,Azimuth,Acc-X,Acc-Y,Acc-Z,Acc-T,Acc-Err,Mag-X,Mag-Y,Mag-Z,Mag-T,Probe-T,AccUnc-X,AccUnc-Y,AccUnc-Z,MagUnc-X,MagUnc-Y,MagUnc-Z
1,2015-02-16 11:39:02,206.901,89.8973,176.034,-0.998284,-0.000809566,-0.0015957,-15,-0.00171401,67.873,19.091,31.958,-15,-14.25,-0.994415,-0.00683615,-0.00463881,67.873,19.091,31.958
2,2015-02-16 11:39:26,359.733,83.9937,331.612,-0.991604,-0.000486354,0.104331,-15,-0.00292263,70.245,18.916,28.137,-15,-14.5,-0.987579,-0.00512711,0.101321,70.245,18.916,28.137
3,2015-02-16 11:39:37,359.835,77.9825,334.053,-0.976108,-0.000597936,0.207789,-15,-0.00202003,71.954,18.736,24.25,-15,-14.25,-0.971933,-0.00390637,0.20484,71.954,18.736,24.25
4,2015-02-16 11:39:46,359.879,71.9923,336.039,-0.950566,-0.000652473,0.308999,-15,-0.000471519,73.022,18.561,20.32,-15,-14.25,-0.946318,-0.00268563,0.306162,73.022,18.561,20.32
5,2015-02-16 11:39:54,359.875,65.9118,337.705,-0.913003,-0.000893695,0.408179,-15,9.24694e-05,73.438,18.381,16.408,-15,-14.25,-0.908719,-0.00170904,0.40553,73.438,18.381,16.408
6,2015-02-16 11:40:04,359.884,60.0699,338.828,-0.867553,-0.00100887,0.499471,-15,0.00105951,73.162,18.313,12.551,-15,-14.5,-0.863308,-0.000732444,0.497085,73.162,18.313,12.551
```

4.6 CALIBRATION DATA

The calibration data file is used during calibration. Since the Android version of the app does not need to calibrate probes, this file type does not need to be supported.

Each record of the file contains fields for axial sensor readings and temperature of a single sensor. The columns are named according to the type of sensor (either Accelerometer-X, Accelerometer-Y, Accelerometer-Z or Magnetometer-X, Magnetometer-Y, Magnetometer-Z and Temperature). Each record also contains the orientation angles that the measurement was taken at (named Orientation-Roll, Orientation-Pitch and Orientation-Azimuth).

A sample calibration data file is shown below:

```
Accelerometer-X,Accelerometer-Y,Accelerometer-Z,Temperature,Orientation-Roll,Orientation-Pitch,Orientation-Azimuth
0.985626,-0.013184,-0.0061037,-15,0,-90,0
0.936552,0.28956,-0.0734886,-15,102,-72,0
0.852077,-0.0192877,-0.504166,-15,180,-60,0
0.796167,-0.518815,-0.29078,-15,-120,-54,0
0.730003,-0.614521,0.275887,-15,-66,-48,0
0.731468,-0.208747,0.635762,-15,-18,-48,0
0.661397,0.368419,0.636494,-15,30,-42,0
0.577898,0.75808,0.238044,-15,72,-36,0
0.491226,0.772485,-0.364513,-15,114,-30,0
0.489517,0.410413,-0.761742,-15,150,-30,0
0.394543,-0.1189,-0.914823,-15,-174,-24,0
0.394299,-0.69216,-0.608417,-15,-132,-24,0
0.297861,-0.953642,-0.0952177,-15,-96,-18,0
0.200201,-0.846217,0.494644,-15,-60,-12,0
0.201666,-0.482925,0.850124,-15,-30,-12,0
0.09888,0.106937,0.989776,-15,6,-6,0
```