National Center for Atmospheric Research
NCAR
Earth Observing Laboratory
EOL

DORADE

Doppler Radar Exchange Format DORADE

Originally:

Wen-Chau Lee, Craig Walther, Richard Oye

Atmospheric Technology Division (ATD)

P. O. Box 3000, Boulder, CO 80307

October 1994, Revised 2003

Extensively revised by Mike Dixon
EOL
July 2010

1 Introduction

The Common Doppler Radar Exchange Format, generally referred to as Universal Format (UF) and introduced in 1980 (Barnes 1980), has been used extensively in the research community in the past decade as a medium to exchange Doppler radar data. The UF was designed primarily to accommodate data for a single ground-based radar.

The advent of airborne Doppler radar (NOAA WP-3D tail Doppler radar) in the early 1980s opened a new arena in mesoscale research. These data have been stored in NOAA internal format because of the different geometry and data characteristics between an airborne and ground-based radar. Attempts have been made to convert NOAA internal format into UF by introducing additional entries in the local use header that accommodate the navigation information unique to a moving platform. Due to the geometry of an airborne Doppler radar, about 70% of the total data are either below ground or above troposphere, having little or no meteorological interest. However, the redundant header information at the beginning of each ray and inflexibility in implementing the data reduction scheme create a huge overhead in the amount of data created during the transformation process (Wakimoto, personal communication). Also, the differences in terminology between airborne and ground-based radar (for example, azimuth and elevation angle) create additional problems for the user.

The development of the ELDORA/ASTRAIA airborne dual-Doppler radar system poses another problem for the UF. When ELDORA/ASTRAIA runs at its full design capability, it can transmit five different frequencies and multiple PRFs by the fore and aft radars. In addition, the antennas will rotate at a maximum rate of 144 degrees per second, which is about 10 times faster than a typical ground-based radar and 3 times faster than the NOAA WP-3D airborne Doppler radar. The data rate will be 6 to 20 times of that generated by a typical airborne or ground-based radars, even before taking into account the multiple frequencies and PRFs. It became clear that a new exchange format was needed to accommodate ELDORA/ASTRAIA data.

The design goals of the new Doppler radar exchange format are:

(1) To be a general purpose radar exchange format used not only for radars on a fixed platform (e.g. ground-based radar), but also for radars on a moving platform (airborne and ship-borne radars).

(2) To accommodate data from multiple radars or even data from multiple instruments (e.g. radar data and aircraft data).

(3) To be efficient at keeping redundant header information to a minimum, and to implement a data compression scheme.

(4)To be flexible for future expansion, without changing its basic structure.

(5) To be flexible with recording media (e.g. 9-track or Exabyte).

A proposal of the Doppler Radar Data Exchange (DORADE) format was discussed in April 1991, in Miami at NOAA/AOML/HRD, among representatives from potential data producers—CRPE (France), NCAR, and NOAA. In addition to discussing the structure of the format, we agreed upon the coordinate systems and terminologies that will be used in the DORADE format. (All these will be defined in the format description section.) The structure and contents of the DORADE format were further discussed during the 25th AMS Radar Conference held June 1991 in Paris, France, by the same group of people who attended the April meeting.

The first draft of the format was distributed in July 1991, to gather comments from an expanded group including scientists and programmers who would use the data. Many comments on the format from five different groups were received. The second draft was distributed for comments in November 1991. In the meantime, the ELDORA development group at NCAR started to integrate this format into the data system. Data collected from the ELDORA test-bed radar were recorded in DORADE. In addition, a routine was written to convert data from UF to DORADE format and vice versa. These efforts were taken to prevent practical difficulties in implementing the DORADE format.

The purpose of this article is to document the DORADE format version 1, which will be used to exchange data collected by the ELDORA/ASTRAIA airborne Doppler radar and the NCAR ground-based radars. The physical description and definitions of the DORADE format are in Section 2. Section 3 describes the data structure. Appendix A provides the schematics of the terminologies used in Section 3. For the purposes of this document, only detailed formats for radar data recording are discussed. Exchanged data should always be corrected as best as the facility making the file can do, i.e., aircraft motion removed, range delay corrected, etc.

2 Definitions

Below are definitions for the various data elements (gate, cell, ray, sweep, volume etc.) and format components (block, descriptor, header, etc.).

2.1 Data elements

Gate: A sampling element of the radar itself.

Cell: A data point recorded on the recording media. It can be a gate or an average of several gates.

Ray: A logical unit of data which contains all data cells (from a single "radar") taken during a single dwell time.

Sweep: A number of rays with similar characteristics, i.e. rays within a 360° rotation, PPI, or RHI

Volume: a number of sweeps with similar characteristics.

2.2 Format elements

Comment Block: contains any number of ASCII characters describing or commenting on anything that the generator of the data set feels is appropriate.

Volume Header: contains a number of *descriptors* to define the characteristic of the instrumentation(s) and the corresponding parameters.

Volume Descriptor: contains information unique to the volume described.

Sensor Descriptor: contains information defining the particular sensor and its operational parameters.

Parameter Descriptor: contains information describing each parameter.

Correction Factor Descriptor: contains the correction factors needed to be applied to various parameters before using them.

Cell Range Vector: contains the distance from the radar to the center of each recorded data cell.

Sweep Info Block: contains information unique to this sweep.

Data Ray: contains information and data of a ray.

Ray Info Block: contains unique information about this ray.

Platform Info Block: contains the navigation information for this ray.

Parameter Data Block: contains the actual data for a single parameter corresponding to those described in the *parameter descriptor*.

Net Info Block: contains information necessary for transferring data over a network in real time.

3 General guidelines

A file may begin with one or more *comment blocks*. The first comment block should contain the ASCII 6-character file identifier “DORADE” starting in character one or two of the data part of the comment block (the first character can be a new line). This comment block or subsequent comment blocks should also contain a description of the file format (i.e., something very similar to what you are now reading). If a compression scheme is used, it should be described.

We recommend that a flag of -999 (for 2- and 4-byte integer entries), -256 (for 1-byte integer entries), or -999.0 (for real entries) be used to denote all entries not applicable, missing data, bad data or deleted data. Positive Doppler velocity is defined as velocity away from the radar.

All blocks (user defined or defined here) begin with four ASCII characters describing the block and then a 32-bit integer giving the length of the block in bytes. The lengths of all blocks or descriptors should be evenly divisible by 4.

All floating point numbers follow the IEEE 32 bit floating point standard. All 4 byte numbers (floats or 32 bit integers) must begin on a boundary that is evenly divisible by 4. The use of place holders should be adopted to accomplish this. All integers should be in "Big Endian" notation (the first byte is the most significant byte).

In general a *volume* is a file. A volume should always begin with a *volume header*. A volume contains many sweeps of data. It can be a leg (in airborne radars), a sector scan (in ground based radars) or any other user selected block of data.

A volume may contain data from multiple sensors: for example, the fore and aft radars in ELDORA/ASTRAIA. To define a volume is relatively straight forward for ground based radar, since a well defined scan sequence (either from bottom to top in the PPI mode or from left to right in the RHI mode) is usually in place. This is not the case for an airborne Doppler radar where a typical straight-line leg can last 30 minutes or more. In other flight patterns such as a circular pattern around a hurricane eye wall, it is very difficult to define a “leg” or volume. Therefore, it is the data producer's responsibility to define a block of data which is suitable to work with as a volume.

A volume header is written whenever there is a change in the radar parameters (e.g. PRT, cell spacing, or calibration). A sensor descriptor is a logical grouping of descriptors that define specifically how the data from a particular sensor is recorded on the media. Many of these sensors will be "radars," and hence a specific sensor descriptor for radars is defined in this document. It is possible, however, that the media may also contain data from other sensors. For example, the ELDORA/ASTRAIA field files will contain data from the in-situ measurement system. The data format for each of these other sensors should be described with a unique *sensor descriptor*.

A *sweep record* should only be considered as a logical means of assembling data. However, a ray of data is a discrete stand-alone entity. The physical records inside of the sweep record should always contain an integer number of data rays. In this way, there is a guarantee that the first four characters of all physical records on the media will have an identifier. The sweep number in the *Sweep Info Block* should be the sequence number from the beginning of the volume.

The correction factors should be applied to the various parameters before using them. This permits adjustments to the correction factors without having to regenerate the whole dataset. The numbers in this descriptor should be added (mathematically) to the data that is written in the exchange format. Some of the correction factors in this descriptor may not be applicable to all radars.

A data ray will always contain a *ray info block* and the data for every parameter associated with the "radar" that created the ray. For moving platform radars, a *platform info block* has been defined that contains all parameters associated with the calculation of antenna location and pointing angles.

4 Organization of data into blocks

When DORADE was first introduced, the primary long-term storage medium was magnetic tape. The early documentation therefore dealt with tape blocks etc.

This is no longer relevant, since the data is stored in ‘file’ units.

Nevertheless, the basic block-type structure of DORADE remains. The following figures shows the logical organization and relationships of some of the main data blocks.

Comments (COMM)
Super sweep descriptor (SSWB)
Volume descriptor (VOLD)

Sensor # 1 descriptor

Radar descriptor (RADD)
Parameter # 1 descriptor (PARM)
Parameter # 2 descriptor (PARM)
Parameter # N descriptor (PARM)
Cell range vector (CELV) or Cell spacing table (CSFD)
Correction Factor descriptor (CFAC)

Sensor # 2 descriptor

Radar descriptor (RADD)
Parameter # 1 descriptor (PARM)
Parameter # 2 descriptor (PARM)
Parameter # N descriptor (PARM)
Cell range vector (CELV) or Cell spacing table (CSFD)
Correction Factor descriptor (CFAC)

Sensor # N descriptor

Radar descriptor (RADD)
Parameter # 1 descriptor (PARM)
Parameter # 2 descriptor (PARM)
Parameter # N descriptor (PARM)
Cell range vector (CELV) or Cell spacing table (CSFD)
Correction Factor descriptor (CFAC)

Figure 1: Header blocks

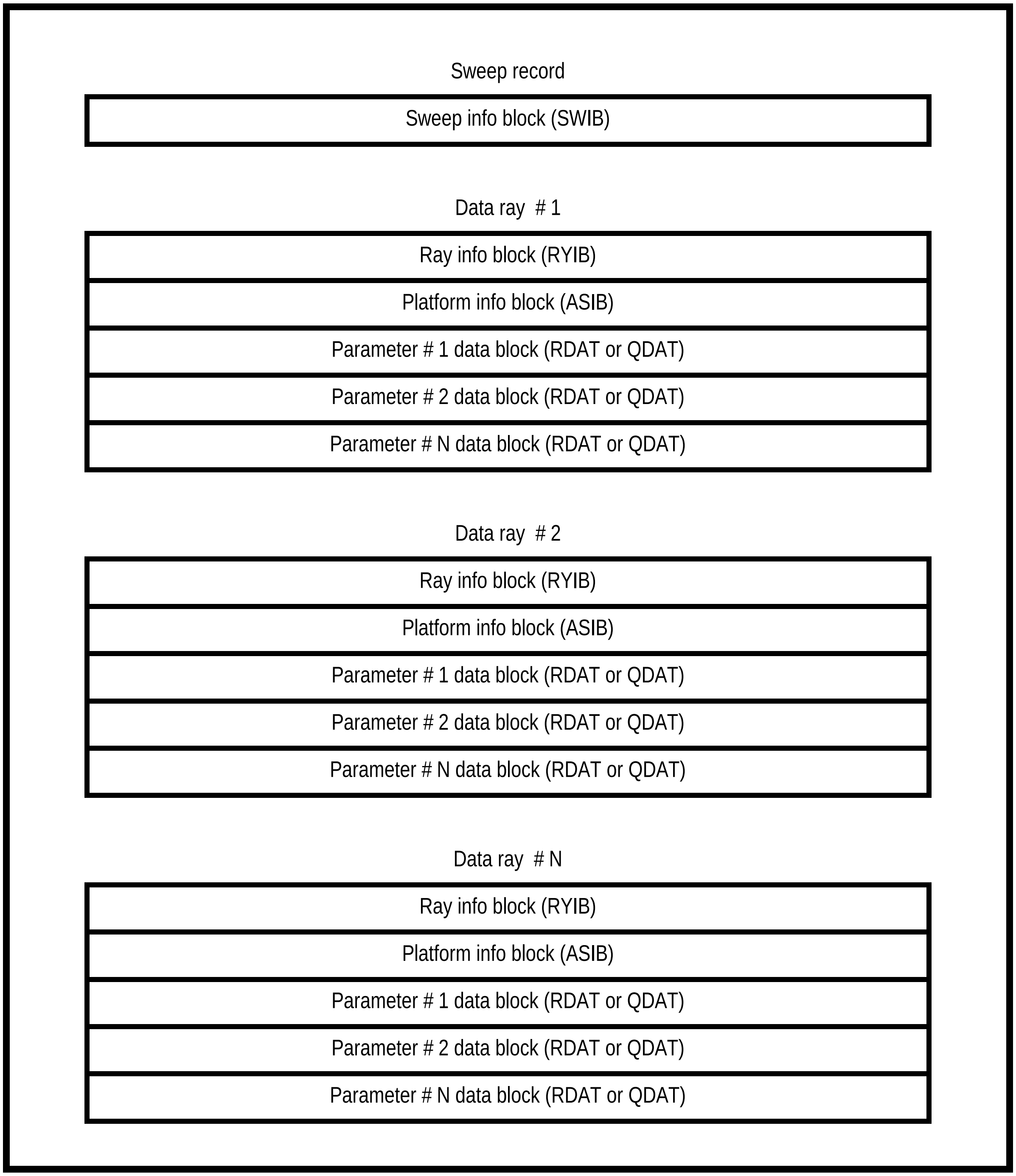

Figure 2: Sweep data blocks

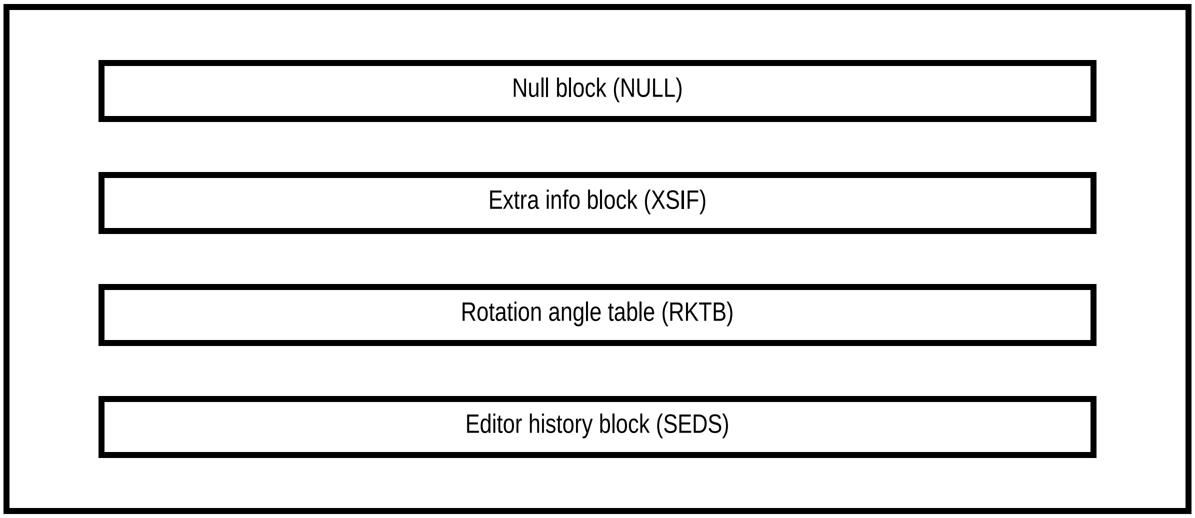

Figure 3: Trailer blocks

5 The geometry of moving platforms

5.1 Reference frame

NOTE: -see Lee et al. (1994) for further background on this topic, and on the corrections to Doppler velocity for moving platforms.

Figure 5.1 depicts the theoretical reference frame for a moving platform, such as an aircraft. This discussion also applies to water-borne platforms and land-based moving platforms, but we will use the aircraft analogy here.

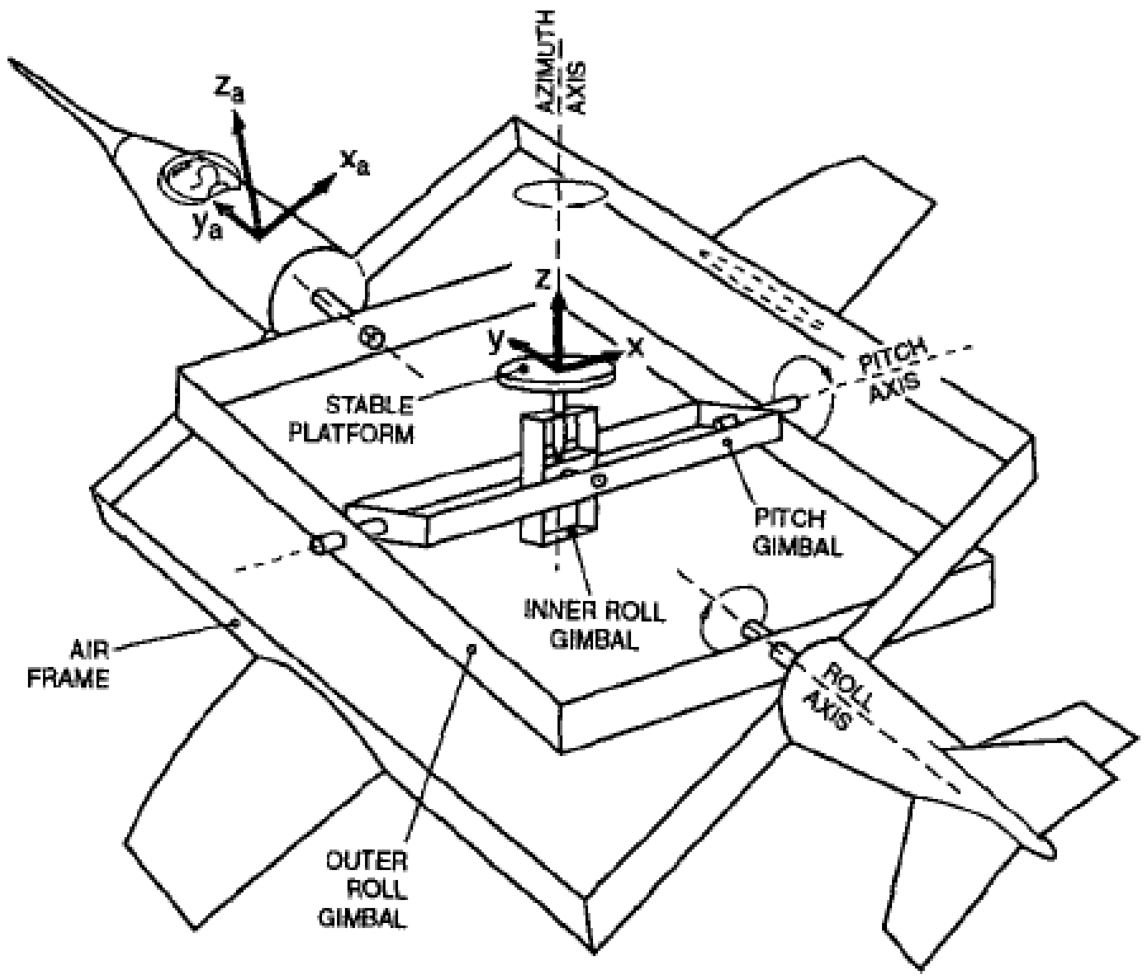

Figure 5.1: Moving platform axis definitions and reference frame
(reproduced from Lee et al., 1994,
originally from Axford, 1968)
(c)American Meteorological Society. Reprinted with permission.

We use right-handed coordinate systems in this discussion. Angles are positive for clockwise rotations about an axis perpendicular to the plane when looking away from the origin.

We consider three coordinate systems here:

- X_a, coordinates relative to the platform
- X_h, coordinates relative to a platform with 0 roll and 0 pitch, but with the Y axis aligned with the aircraft heading
- X, relative to the earth, +y points north and +x points east.

The radar pointing angles rotation (θ) and tilt (τ) are measured relative to X_a. The requirement is to derive the angles elevation (φ) and azimuth (λ) relative to X.

The airframe-relative coordinate system is characterized by 3 angles: pitch (P), roll (R) and heading (H). These angles are generally measured by an inertial navigation system (INS).

The platform moves relative to X, based on its heading H, and the drift D, caused by wind or current. (D is not relevant to land-based platforms). The track T is the line of movement over the ground.

Figures 6.2 a through c show the definitions of heading, drift, pitch and roll.

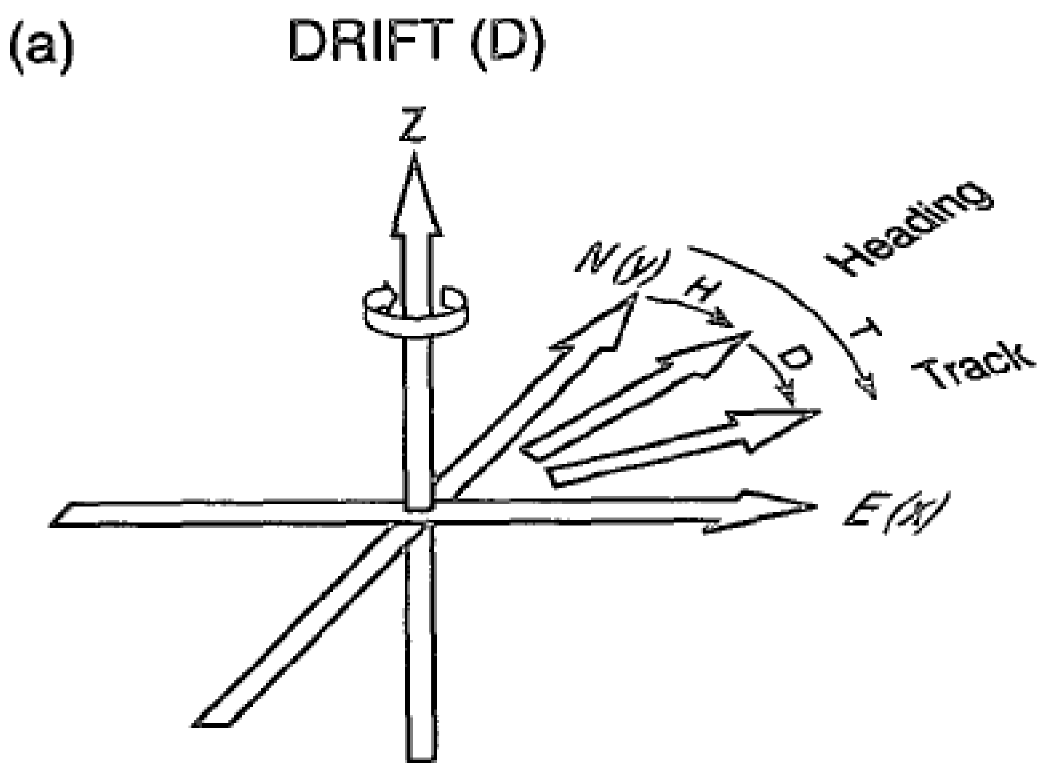

Figure 5.2(a): Definition of heading, drift and track.
Reproduced from Lee et al., 1994.

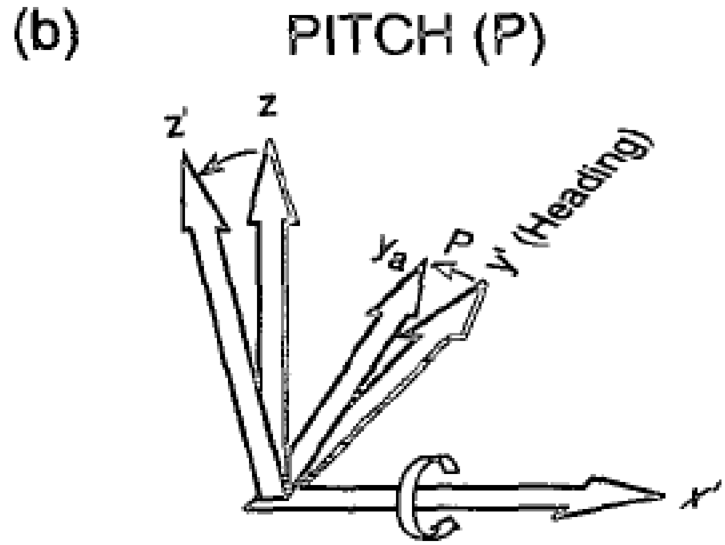

Figure 5.2(b): Definition of pitch
(Reproduced from Lee et al., 1994)

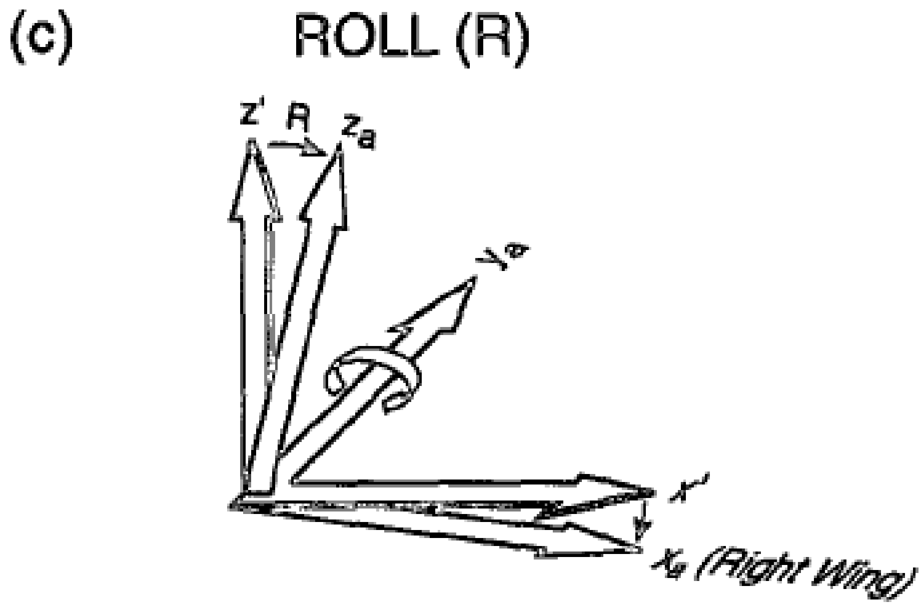

Figure 5.2(c): Definition of roll
(Reproduced from Lee et al., 1994)
(c)American Meteorological Society. Reprinted with permission.

5.2 Primary rotation axes

Weather radars and lidars rotate primarily about a principal axis. The antenna can also slew about a secondary axis, orthogonal to the primary axis.

In the case of most fixed radars, this principle axis is generally vertical. The azimuth angle (λ) is measured clockwise from true north, and the elevation angle (φ) is measured as positive above the horizontal plane and negative below it.

With moving platforms, we maintain that same definition for azimuth and elevation , i.e. that they are relative to X, the earth reference frame. Our task here is to derive azimuth and elevation from angular measurements made relative to the platform, and from the platform reference frame itself.

We define two additional terms, rotation (θ) and tilt(τ), which are angular measurements made relative to the platform.

The conversion from rotation and tilt to elevation and azimuth depends on the primary axis of rotation.

We use the global variable "primary_axis" to indicate in a CfRadial file which axis is primary.

5.2.1 Type Z radars: primary axis z_a (normal radar, nose radar)

If the primary axis is z_a, the rotation angle θ is 0 in the (y_a, z_a) plane and the tilt angle (τ) is 0 in the (x_a, y_a) plane.

The location of a gate in X_a coordinates is:

$$\begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} = r \begin{pmatrix} \sin\theta\cos\tau \\ \cos\theta\cos\tau \\ \sin\tau \end{pmatrix}$$

5.2.2 Type Y radars: primary axis y_a (e.g. tail radar rotating around longitudinal axis of fuselage, ELDORA)

If the primary axis is y_a, the rotation angle θ is 0 in the (y_a, z_a) plane and the tilt angle (τ) is 0 in the (x_a, z_a) plane.

The location of a gate in X_a coordinates is:

$$\begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} = r \begin{pmatrix} \sin\theta\cos\tau \\ \sin\tau \\ \cos\theta\cos\tau \end{pmatrix}$$

5.2.3 Type X radars, primary axis x_a (e.g. belly radar scanning forward and aft, EDOP)

If the primary axis is x_a, the rotation angle θ is 0 in the (x_a, z_a) plane and the tilt angle (τ) is 0 in the (y_a, z_a) plane.

The location of a gate in X_a coordinates is:

$$\begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} = r \begin{pmatrix} \sin\tau \\ \sin\theta\cos\tau \\ \cos\theta\cos\tau \end{pmatrix}$$

5.3 Rotating X_a to X_h

Rotating X_a to X_h requires 2 steps:

- first remove the roll R, by rotating the x axis around the y axis by –R.

- then remove the pitch P, by rotating the y axis around the x axis by –P.

The transformation matrix for removing the roll component is:

$$M_R = \begin{pmatrix} \cos R & 0 & \sin R \\ 0 & 1 & 0 \\ -\sin R & 0 & \cos R \end{pmatrix}$$

The transformation matrix for removing the pitch component is:

$$M_P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos P & -\sin P \\ 0 & \sin P & \cos P \end{pmatrix}$$

We apply these transformations consecutively:

$$X_h = M_P M_R X_a$$

$$M_P M_R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos P & -\sin P \\ 0 & \sin P & \cos P \end{pmatrix} \begin{pmatrix} \cos R & 0 & \sin R \\ 0 & 1 & 0 \\ -\sin R & 0 & \cos R \end{pmatrix}$$

$$= \begin{pmatrix} \cos R & 0 & \sin R \\ \sin P \sin R & \cos P & -\sin P \cos R \\ -\cos P \sin R & \sin P & \cos P \cos R \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

For type Z radars:

$$X_h = M_P M_R X_a$$

$$\begin{pmatrix} x_h \\ y_h \\ z \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} r \begin{pmatrix} \sin\tau \\ \sin\theta\cos\tau \\ \cos\theta\cos\tau \end{pmatrix}$$

$$\begin{pmatrix} x_h \\ y_h \\ z \end{pmatrix} = r \begin{pmatrix} m_{11}\sin\tau + m_{12}\sin\theta\cos\tau + m_{13}\cos\theta\cos\tau \\ m_{21}\sin\tau + m_{22}\sin\theta\cos\tau + m_{23}\cos\theta\cos\tau \\ m_{31}\sin\tau + m_{32}\sin\theta\cos\tau + m_{33}\cos\theta\cos\tau \end{pmatrix}$$

For type Y radars:

$$X_h = M_P M_R X_a$$

$$\begin{pmatrix} x_h \\ y_h \\ z \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} r \begin{pmatrix} \sin\theta\cos\tau \\ \sin\tau \\ \cos\theta\cos\tau \end{pmatrix}$$

$$\begin{pmatrix} x_h \\ y_h \\ z \end{pmatrix} = r \begin{pmatrix} m_{11}\sin\theta\cos\tau + m_{12}\sin\tau + m_{13}\cos\theta\cos\tau \\ m_{21}\sin\theta\cos\tau + m_{22}\sin\tau + m_{23}\cos\theta\cos\tau \\ m_{31}\sin\theta\cos\tau + m_{32}\sin\tau + m_{33}\cos\theta\cos\tau \end{pmatrix}$$

For type Z radars:

$$X_h = M_P M_R X_a$$

$$\begin{pmatrix} x_h \\ y_h \\ z \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} r \begin{pmatrix} \sin\theta\cos\tau \\ \cos\theta\cos\tau \\ \sin\tau \end{pmatrix}$$

$$\begin{pmatrix} x_h \\ y_h \\ z \end{pmatrix} = r \begin{pmatrix} m_{11}\sin\theta\cos\tau + m_{12}\cos\theta\cos\tau + m_{13}\sin\tau \\ m_{21}\sin\theta\cos\tau + m_{22}\cos\theta\cos\tau + m_{23}\sin\tau \\ m_{31}\sin\theta\cos\tau + m_{32}\cos\theta\cos\tau + m_{33}\sin\tau \end{pmatrix}$$

5.4 Computing earth-relative elevation and azimuth

After we have applied the above transformations to obtain coordinates relative to X_h, we can compute the azimuth and elevation as follows:

$$\lambda_h = \tan^{-1}(x_h / y_h)$$

$$\lambda = \lambda_h + H$$

$$\phi = \sin^{-1}(z / r)$$

6 References

Axford, D. N., 1968: On the accuracy of wind measurements using an inertial platform in an aircraft, and an example of a measurement of the vertical structure of the atmosphere. *J. Appl. Meteor.*, 7, 645-666.

Barnes, S. L., 1980: Report on a meeting to establish a common Doppler radar exchange format. *Bull. Amer. Meteor. Soc.*, **61,** 1401-1404.

Lee, W., P. Dodge, F. D. Marks Jr. and P. Hildebrand, 1994: Mapping of Airborne Doppler Radar Data. *Journal of Oceanic and Atmospheric Technology*, 11, 572 – 578.

7 Data block structures

The following tables provide details on the data blocks supported by Dorade.

7.1 Comment block - COMM

Length: 508 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	“COMM.”
4	integer	4	1	nbytes	Length in bytes =508
8	char	1	500	comment	comment text

7.2 Super Sweep Identification block - SSWB

Length: 196 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	SSWB
4	integer	4	1	nbytes	parameters from the first version
8	integer	4	1	last_used	Unix time
12	integer	4	1	start_time	
16	integer	4	1	stop_time	

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
20	integer	4	1	sizeof_file	
24	integer	4	1	compression_flag	
28	integer	4	1	volume_time_stamp	to reference current volume
32	integer	4	1	num_params	number of parameters
36	char	1	8	radar_name	
44	float	8	1	start_time	
52	float	8	1	stop_time	"last_used " is an age off indicator where >0 implies Unix time of the last access and 0 implies this sweep should not be aged off
60	integer	4	1	version_num	
64	integer	4	1	num_key_tables;=1	
68	integer	4	1	status;	
72	integer	4	7	place_holder	k ey_table_info_t key_table (MAX_KEYS) ; offset and key info to a table containing key value such as the rot. angle and the offset to the corresponding ray in the disk file.
100	integer	4	1	key_table[0].offset	Offset of key table 0
104	integer	4	1	key_table[0].size	Size of key table 0
108	integer	4	1	key_table[0].type	Type of key table 0

Offset bytes	Data type	Byte width	Count	Item	Description
112	integer	4	1	key_table[1].offset	Offset of key table 1
116	integer	4	1	key_table[1].size	Size of key table 1
120	integer	4	1	key_table[1].type	Type of key table 1
…..	…..	…..	…..	…..	…..
…..	…..	…..	…..	…..	…..
172	integer	4	1	key_table[6].offset	Offset of key table 6
176	integer	4	1	key_table[6].size	Size of key table 6
180	integer	4	1	key_table[6].type	Type of key table 6
184	integer	4	1	key_table[7].offset	Offset of key table 7
188	integer	4	1	key_table[7].size	Size of key table 7
192	integer	4	1	key_table[7].type	Type of key table 7

7.3 Volume description block - VOLD

Length: 72 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Volume descriptor identifier: ASCII * characters “VOLD” stand for Volume Descriptor.

Offset bytes	Data type	Byte width	Count	Item	Description
4	integer	4	1	nbytes	Volume descriptor length in bytes.
8	integer	2	1	format_version	ELDORA/ASTRAEA field format revision number.
10	integer	2	1	volume_num	Volume Number in current operations
12	integer	4	1	maximum_bytes	Maximum number of bytes in any physical record in this volume
16	char	1	20	proj_name	Project number or name
36	integer	2	1	year	Year data taken in years
38	integer	2	1	month	Month data taken in months
40	integer	2	1	day	Day data taken in days
42	integer	2	1	data_set_hour	Hour data taken in hours
44	integer	2	1	data_set_minute	Minute data taken in minutes
46	integer	2	1	data_set_second	Second data taken in seconds
48	char	1	8	flight_number	Flight number
56	char	1	8	gen_facility	Identifier of facility that generated this recording
64	integer	2	1	gen_year	Year this recording was generated in years
66	integer	2	1	gen_month	Month this recording was generated in months

Offset bytes	Data type	Byte width	Count	Item	Description
68	integer	2	1	gen_day	Day this recording was generated in days
70	integer	2	1	number_sensor_des	Total number of sensor descriptors that follow

7.4 Radar description - RADD

Length: 300 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier for a radar descriptor block (ascii characters “RADD”)
4	integer	4	1	nbytes	Length of a radar descriptor block in bytes
8	char	1	8	radar_name	Eight character radar name
16	float	4	1	radar_const	Radar/lidar constant in??
20	float	4	1	peak_power	Typical peak power of the sensor in kw. Pulse energy is really the peak_power pulse_width
24	float	4	1	noise_power	Typical noise power of the sensor in dBm.
28	float	4	1	receiver_gain	Gain of the receiver in db
32	float	4	1	antenna_gain	Gain of the antenna in db
36	float	4	1	system_gain	System gain in db. (Ant G – WG loss)

Offset bytes	Data type	Byte width	Count	Item	Description
40	float	4	1	horz_beam_width	Horizontal beam width in degrees. Beam divergence in milliradians is equivalent to beamwidth
44	float	4	1	vert_beam_width	Vertical beam width in degrees
48	integer	2	1	radar_type	RadarType (0) Ground, 1)Airborne Fore, 2)Airborne Aft, 3) airborne tail, 4)Airborne lower fuselage, 5)Shipborne
50	integer	2	1	scan_mode	Scan Mode (0)Calibration, 1)PPI (constant elevation) 2)Coplane, 3)RHI (Constant azimuth), 4)Vertical Pointing, 5)Target (Stationary), 6)Manual, 7)Idle (out of control)
52	float	4	1	req_rotat_vel	Requested rotational velocity of the antenna in degrees / sec
56	float	4	1	scan_mode_pram0	Scan mode specific parameter #0 (Has different meaning for different scan modes
60	float	4	1	scan_mode_pram1	Scan mode specific parameter #1
64	integer	2	1	num_parameter_des	Total number of additional descriptor block for this radar
66	integer	2	1	total_num_des	Total number of additional descriptor block for this radar
68	integer	2	1	data_compress	Data compression. 0 =none, 1 = HRD scheme
70	integer	2	1	data_reduction	Data reduction algorithm: 1 = none, 2 = between 2 angles, 3 = between concentric circles, 4 = above / below certain altitudes
72	float	4	1	data_red_parm0	1 = smallest positive angle in degrees, 2 = inner circle diameter, km, 4 = minimum altitude, km
76	float	4	1	data_red_parm1	1 = largest positive angle, degress, 2 = outer circle diameter, km, 4 = maximum altitude
80	float	4	1	radar_longitude	Longitude of radar in degrees

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
84	float	4	1	radar_latitude	Latitude of radar in degrees
88	float	4	1	radar_altitude	Altitude of radar above msl in km
92	float	4	1	eff_unamb_vel	Effective unambiguous range, km
96	float	4	1	eff_unamb_range	Effective unambiguous range, km
100	integer	2	1	num_freq_trans	Number of frequencies transmitted
102	integer	2	1	num_ipps_trans	Number of different inter-pulse periods transmitted
104	float	4	1	freq1	Frequency 1
108	float	4	1	freq2	Frequency 2
112	float	4	1	freq3	Frequency 3
116	float	4	1	freq4	Frequency 4
120	float	4	1	freq5	Frequency 5
124	float	4	1	interpulse_per1	Interpulse period 1
128	float	4	1	interpulse_per2	Interpulse period 2
132	float	4	1	interpulse_per3	Interpulse period 3
136	float	4	1	Interpulse_per4	Interpulse period 4
140	float	4	1	interpulse_per5	Interpulse period 5

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
144	integer	4	1	extension_num	1995 extension #1
148	char	1	8	config_name	Used to identify this set of unique radar characteristics
156	integer	4	1	config_num	Facilitates a quick lookup of radar characteristics for each ray. Extend the radar descriptor to include unique lidar parameters
160	float	4	1	aperture_size	Diameter of the lidar aperature in cm
164	float	4	1	field_of_view	Field of view of the receiver.mra;
168	float	4	1	aperture_eff	Aperature efficiency in %.
172	float	4	11	freq	Make space for a total of 16 freqs
216	float	4	11	interpulse_per	And ipps other extensions to the radar descriptor
260	float	4	1	pulse_width	Typical pulse width in microseconds. Pulse width is inverse of the band width.
264	float	4	1	primary_cop_baseln	Coplane baselines
268	float	4	1	secondary_cop_baseln	
272	float	4	1	pc_xmtr_bandwidth	Pulse compression transmitter bandwidth
276	integer	4	1	pc_waveform_type	Pulse compression waveform type
280	char	1	20	site_name	

7.5 Correction factor – CFAC

Length: 72 bytes

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
0	char	1	4	id	Correction descriptor identifier: ASCII characters "CFAC" stand for Volume Descriptor
4	integer	4	1	nbytes	Correction descriptor length in bytes
8	float	4	1	azimuth_corr	Correction added to azimuth(deg)
12	float	4	1	elevation_corr	Correction added to elevation (deg)
16	float	4	1	range_delay_corr	Correction used for range delay(m)
20	float	4	1	longitude_corr	Correction added to radar longitude
24	float	4	1	latitude_corr	Correction added to radar latitude
28	float	4	1	pressure_alt_corr	Correction added to pressure altitude (msl) (km)
32	float	4	1	Radar_alt_corr	Correction added to radar altitude above ground level (agl) (km)
36	float	4	1	ew_gndspd_corr	Correction added to radar platform ground speed (E-W) (m/s)
40	float	4	1	ns_gndspd_corr	Correction added to radar platform ground speed (N-S) (m/s)

Offset bytes	Data type	Byte width	Count	Item	Description
44	float	4	1	vert_vel_corr	Correction added to radar platform vertical velocity (m/s)
48	float	4	1	heading_corr	Correction added to radar platform heading (deg)
52	float	4	1	roll_corr	Correction added to radar platform roll (deg)
56	float	4	1	pitch_corr	Correction added to radar platform picth (deg)
60	float	4	1	drift_corr	Correction added to radar platform drift (deg)
64	float	4	1	rot_angle_corr	Correction added to radar rotation angle (deg)
68	float	4	1	tilt_corr	Correction added to radar tilt angle

7.6 Parameter (data field) description – PARM

Length: 216 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Parameter descriptor identifier (ascii characters “PARM”)
4	integer	4	1	Nbytes	Parameter descriptor length in bytes
8	char	1	8	parameter_name	Name of parameter being described

Offset bytes	Data type	Byte width	Count	Item	Description
16	char	1	40	param_description	Detailed descri9ption of this parameter
56	char	1	8	param_units	Units parameter is written in
64	integer	2	1	interpulse_time	Inter-pulse periods used. Bits 0-1 = frequencies 1-2
66	integer	2	1	xmitted_freq	Frequencies used for this parameter
68	float	4	1	recvr_bandwidth	Effective receiver bandwidth for this parameter in MHz
72	integer	2	1	pulse_width	Effective pulse width of parameter in m
74	integer	2	1	polarization	Polarization of the radar beam for this parameter (0 Horizontal, 1 vertical, 2 circular, 3 elliptical) in na
76	integer	2	1	num_samples	Number of samples used in estimate for this parameter
78	integer	2	1	binary_format	Binary format of radar data
80	char	1	8	threshold_field	Name of parameter upon which this parameter is thresholded (ascii characters NONE if not thresholded)
88	float	4	1	threshold_value	Value of threshold in ?
92	float	4	1	parameter_ scale	Scale factor for parameter
96	float	4	1	parameter_bias	Bias factor for parameter
100	integer	4	1	bad_data	Bad data flag.

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
104	integer	4	1	extension_num	1995 extension #1
108	char	1	8	config_name	Used to identify this set of unique radar characteristics
116	integer	4	1	config_num	
120	integer	4	1	offset_to_data	Bytes added to the data struct pointer to point to the first datum whether it’s an RDAT or a QDAT
124	float	4	1	mks_conversion	
128	integer	4	1	num_qnames	
132	char	1	32	qdata_names	Each of 4 names occupies 8 characters of this space and is blank filled. Each name identifies some interesting segment of data in a particular ray for this parameter.
164	integer	4	1	num_criteria	
168	char	1	32	criteria_names	Each of 4 names occupies 8 characters and is blank filled. These names identify a single interesting fl32ing point value that is associated with a particular ray for this parameter. Examples might be a brightness temperature or the percentage of cells above or below a certain value
200	integer	4	1	number_cells	
204	float	4	1	meters_to_first_cell	center
208	float	4	1	meters_between_cells	

Offset bytes	Data type	Byte width	Count	Item	Description
212	float	4	1	eff_unamb_vel	Effective unambiguous velocity, m/s

Note: some files have a smaller size of 104 bytes, with only the first part filled in

7.7 Cell vector block – CELV

Length: 6012 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Cell descriptor identifier: ASCII characters “CELV” stand for cell vector
4	integer	4	1	nbytes	Comment descriptor length in bytes
8	integer	4	1	number_cells	Number of sample volumes
12	float	4	1500	dist_cells	Distance from the radar to cell n in meters

7.8 Cell spacing table – CSFD

Length: 64 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier for a cell spaceing descriptor (ascii characters CSFD)
4	integer	4	1	nbytes	Cell spacing descriptor length in bytes
8	integer	4	1	num_segments	Number of segments that contain cells of

Offset bytes	Data type	Byte width	Count	Item	Description
12	float	4	1	dist_to_first	Distance to first gate in meters
16	float	4	8	spacing	Width of cells in each segment in m
48	integer	2	8	num_cells	Number of cells in each segment. Equal widths

7.9 Sweep information table – SWIB

Length: 40 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Comment descriptor identifier: ASCII characters “SWIB” stand for sweep info block descriptor
4	integer	4	1	nbytes	Sweep descriptor length in bytes
8	char	1	8	radar_name	
16	integer	4	1	sweep_num	Sweep number from the beginning of the volume
20	integer	4	1	num_rays	Number of rays recorded in this sweep
24	float	4	1	start_angle	True start angle (deg)
28	float	4	1	stop_angle	True stop angle (deg)
32	float	4	1	fixed_angle	
36	integer	4	1	filter_flag	

7.10 Platform geo-reference block – ASIB

Length: 80 bytes

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
0	char	1	4	id	Identifier for the aircraft/ shop parameters block
4	integer	4	1	nbytes	Length in bytes of the aircraft/ship arameters block
8	float	4	1	longitude	Antenna longitude (Eastern hemisphere is positive, West negative) in degrees
12	float	4	1	latitude	Antenna latitude (northern hemisphere is positive, south negative) in degrees
16	float	4	1	altitude_msl	Antenna altitude above mean sea level (MSL) in km
20	float	4	1	altitude_agl	Antenna altitude above ground level (AGL) in km
24	float	4	1	ew_velocity	Antenna east-west ground speed (towards East is positive) in m/sec
28	float	4	1	ns_velocity	Antenna north-south ground speed (towards North is positive) in m/sec
32	float	4	1	vert_velocity	Antenna vertical velocity (up is positive) in degrees
36	float	4	1	heading	Antenna heading (angle between rotodome rotational axis and true North, clockwise (looking down positive) in degrees
40	float	4	1	roll	Roll angle of aircraft tail section (horizontal zero, positive left wing up) in degrees
44	float	4	1	pitch	Pitch angle of rotodome (horizontal is zero positive front up) in degrees
48	float	4	1	drift_angle	Antenna drift angle. (angle between platform true velocity and heading, positive is a drift more clockwise looking down) in degrees

Offset bytes	Data type	Byte width	Count	Item	Description
52	float	4	1	rotation_angle	Angle of the radar beam with respect to the airframe 9zero is along vertical stabilizer, positive is clockwise) in deg
56	float	4	1	tilt	Angle of radar beam and line normal to longitudinal axis of aircraft, positive is towards nose of aircraft in degrees
60	float	4	1	ew_horiz_wind	East-west wind velocity at the platform (towards East is positive) in m/sec
64	float	4	1	ns_horiz_wind	North-south wind velocity at the platform (towards North is positive) in m/sec
68	float	4	1	vert_wind	Vertical wind velocity at the platform (up is positive) in m/sec
72	float	4	1	heading_change	Heading change rate in degrees/second
76	float	4	1	pitch_change	Pitch change rate in degrees/second

7.11 Ray information block – RYIB

Length: 44 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier for a data ray info block (ascii characters “RYIB”
4	integer	4	1	nbytes	Length of a data ray info block in bytes
8	integer	4	1	sweep_num	Sweep number for this radar
12	integer	4	1	julian_day	guess
16	integer	2	1	hour	Hour in hours

Offset bytes	Data type	Byte width	Count	Item	Description
18	integer	2	1	minute	Minute in minutes
20	integer	2	1	second	Second in seconds
22	integer	2	1	millisecond	Millisecond in milliseconds
24	float	4	1	azimuth	Azimuth in degrees
28	float	4	1	elevation	Elevation in degrees
32	float	4	1	peak_power	Last measured peak transmitted power in kw
36	float	4	1	true_scan_rate	Actual scan rate in degrees/second
40	integer	4	1	ray_status	0 = normal, 1 = transition, 2 = bad

7.12 Field data block – RDAT

Length: 16 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Parameter data de3scriptor identifier: ASCII characters "RDAT" stand for parameter data block Descriptor
4	integer	4	1	nbytes	Parameter data descriptor length in bytes
8	char	1	8	pdata_name	Name of parameter

7.13 Extended field data block – QDAT

Length: 56 bytes

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
0	char	1	4	id	Parameter data descriptor identifier: ASCII characters “QDAT” for a block that contains the data plus some supplemental and identifying information
4	integer	4	1	nbytes	Parameter data descriptor length in bytes. This represents the size of this header information plus the data. For this data block the start of the data is determined by using “offset_to_data” in the corresponding parameter descriptor “struct parameter _d” the offset is from the beginning of this descriptor block
8	char	1	8	pdata_name	Name of parameter
16	integer	4	1	extension_num	
20	integer	4	1	config_num	Facilitates indexing into an array of radar descriptors where the radar characteristics of each ray and each parameter might be unique such as phased array antennas
24	integer	2	4	first_cell	
32	integer	2	4	num_cells	First cell and num cells demark some feature in the data and it’s relation to the cell vector first_cell [n] = 0 implies the first datum present corresponds to “dist_cells [0] in “struct cell_d” for TRMM data this would be the nominal sample where the cell vector is at 125 meter resolution instead of 250 meters and identified segments might be the rain echo oversample “RAIN_ECH” and the surface oversample “SURFACE”

Offset bytes	Data type	Byte width	Count	Item	Description
40	float	4	4	criteria_value	Criteria value associatged with a criteria name in “struct parameter_d”

7.14 Extra stuff block – XSTF

Length: 24 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	“XSTF”
4	integer	4	1	nbytes	Size in bytes
8	integer	4	1	one	Always set to one (endian flag)
12	integer	4	1	source_format	As per .. /include/dd_defines.h
16	integer	4	1	offset_to_first_item	Bytes from start of struct
20	integer	4	1	transition_flag	Is antenna in transition?

7.15 NULL block – NULL

Length: 8 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	NULL
4	integer	4	1	nbytes	Size in bytes

7.16 Rotation angle table block – RKTB

Length: 28 bytes

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
0	char	1	4	id	RKTB
4	integer	4	1	nbytes	Size in bytes
8	integer	4	1	angle2ndx	360 / ndx_que_size
12	integer	4	1	ndx_que_size	Number of indices
16	integer	4	1	first_key_offset	Offset, in bytes, to first key
20	integer	4	1	angle_table_offset	Offset, in bytes, to angle table
24	integer	4	1	num_rays	Number of rays in ppi

7.17 Radar parameter block – FRAD

Length: 52 bytes

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
0	char	1	4	id	Field parameter data identifier (ascii characters FRAD)
4	integer	4	1	nbytes	Length of the field parameter data block in bytes
8	integer	4	1	data_sys_status	Status word, bits will be assigned particular status when needed
12	char	1	8	radar_name	Name of radar from which this data ray came from

Offset bytes	Data type	Byte width	Count	Item	Description
20	float	4	1	test_pulse_level	Test pulse power level as measured by the power meter in dbm
24	float	4	1	test_pulse_dist	Distance from antenna to middle of test pulse im km
28	float	4	1	test_pulse_width	Test pulse width in m
32	float	4	1	test_pulse_freq	Test pulse frequency in Ghz
36	integer	2	1	test_pulse_atten	Test pulse attenuation in db
38	integer	2	1	test_pulse_fnum	Frequency number being calibrated with the test pulse (what mux on timing module is set to)
40	float	4	1	noise_power	Total estimated noise poser in dbm
44	integer	4	1	ray_count	Data ray counter for this particular type of data ray
48	integer	2	1	first_rec_gate	First recorded gate number (N)
50	integer	2	1	last_rec_gate	Last recorded gate number (M)

7.18 Field radar block – FRIB

Length: 264 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier for a field written radar information block (ascii characters FRIB)
4	integer	4	1	nbytes	Length of this field written radar information block in bytes

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
8	integer	4	1	Data_sys_id	Data system identification
12	float	4	1	loss_out	Waveguide losses between transmitter and antenna in db
16	float	4	1	loss_in	Waveguide losses between antenna and low noise amplifier in db
20	float	4	1	loss_rjoint	Losses in the rotary joint in db
24	float	4	1	ant_v_dim	Antenna Vertical Dimension in m
28	float	4	1	ant_h-dim	Antenna horizontal dimension in m
32	float	4	1	ant_noise_temp	Antenna noise temperature in degrees k
36	float	4	1	r_noise_figure	Receiver noise figure in dB
40	float	4	5	xmit_power	Nominal peak transmitted power in dBm by channel
60	float	4	1	x_band_gain	x band gain in dB
64	float	4	5	receiver_gain	Measured receiver gain in db (by channel)
84	float	4	5	if_gain	Measured IF gain in db (by channel)
104	float	4	1	conversion_gain	A to D conversion gain in dB
108	float	4	5	scale_factor	Scale factor to account for differences in the individual channels, and the inherent gain due to summing over the dwell time
128	float	4	1	processor_const	Constant used to scale dBz to units the display processors understand

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
132	integer	4	1	dly_tube_antenna	time delay from RF being applied to tube and energy leaving antenna in ns
136	integer	4	1	dly_rndtrip_chip_atod	Time delay from a chip generated in the yiming module and the RF pulse entering the A to D converters. Need to take the RF input to the HPA and inject it into the waveguide back at the LNA to make this measurement in ns
140	integer	4	1	dly_timmod_testpulse	Time delay from timing module test pulse edge and test pulse arriving at the A/D converter in ns
144	integer	4	1	dly_modulator_on	Modulator rise time (time between video on into HPA and modulator full up in the high power amplifier) in ns
148	integer	4	1	dly_modulator_off	Modulator fall time (time between video off into the HPA)
152	float	4	1	peak_power_offset	Added to the power meter reading of the peak output power this yields actual peak output power (in dB)
156	float	4	1	test_pulse_offset	Added to the power meter reading of the test pulse this yields actual injected test pulse power (dB)
160	float	4	1	E_plane_angle	E-plane angle (tilt) this is the angle in the horizontal plane (when antennas are vertical) between a line normal to the aircraft’s longitudinal axis and the radar beam in degrees. Positive is in direction of motion (fore)
164	float	4	1	H_plane_angle	H plane angle in degrees – this follows the sign convention described in the

Offset bytes	Data type	Byte width	Count	Item	Description
					DORADE documentation for ROLL angle
168	float	4	1	encoder_antenna_up	Encoder reading minus IRU roll angle when antenna is up and horizontal
172	float	4	1	pitch_antenna_up	Antenna pitch angle (measured with transit) minus IRU pitch angle when antenna is pointing up
176	integer	2	1	indepf_times_flg	0= neither recorded, 1 = independent frequency data only, 3 = independent frequency and time series data recorded
178	integer	2	1	indep_freq_gate	Gate number where the independent frequency data comes from
180	integer	2	1	time_series_gate	Gate number where the time series data come from
182	integer	2	1	num_base_params	Number of base parameters
184	char	1	80	file_name	Name of this header file

7.19 LIDAR description block – LIDR

Length: 148 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier a lidar descriptor block (four ASCII characters “LIDR”)
4	integer	4	1	nbytes	Length of a lidar descriptor block
8	char	1	8	lidar_name	Eight character lidar name. (characters SABL)

Offset bytes	Data type	Byte width	Count	Item	Description
16	float	4	1	lidar_const	Lidar constant
20	float	4	1	pulse_energy	Typical pulse energy of the lidar
24	float	4	1	peak_power	Typical peak power of the lidar
28	float	4	1	pulsewidth	Typical pulse width
32	float	4	1	aperature_size	Diameter of the lidar aperature
36	float	4	1	field_of_view	Field of view of the receiver. mra;
40	float	4	1	aperature_eff	Aperature efficiency
44	float	4	1	beam_divergence	Beam divergence
48	integer	2	1	lidar_type	Lidar type: 0) Ground, 1) Airborne fore, 2) Airborne aft, 3) Airborne tail, 4) Airborne lower fuselage, 5) Shipborne, 6) Airborne Fixed
50	integer	2	1	scan_mode	Scan mode: 0) Calibration, 1) PPI (constant elevation), 2) Co-plane, 3) RHI (Constant azimuth), 4) Vertical pointing up, 5) Target (stationary), 6) Manual, 7) Idle (out of control), 8) Surveillance, 9) Vertical sweep, 10) Vertical scan, 11) Vertical pointing down, 12) Horizontal pointing right, 13) Horizontal pointing left
52	float	4	1	req_rotat_vel	Requested rotational velocity of the scan mirror
56	float	4	1	scan_mode_pram0	Scan mode specific parameter #0 (Has different meanings for different scan modes) (Start angle for vertical scanning)

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
60	float	4	1	scan_mode_pram1	Scan mode specific parameter #1 (Has different meaning for different scan modes) Stop angle for vertical scanning)
64	integer	2	1	num_parameter_des	Total number of parameter descriptor blocks for this lidar
66	integer	2	1	total_number_des	Total number of all descriptor blocks for this lidar
68	integer	2	1	data_compress	Data compression scheme in use: 0) no data compression 1) using HRD compression scheme
70	integer	2	1	data_reduction	Data reduction algorithm in use: 0)None, 1) Between two angles, 2) Between concentric circles, 3) Above and below certain altitudes
72	float	4	1	data_red_parm0	Data reduction algorithm specific parameter #0: 0) Unused, 1) Smallest positive angle in degrees, 2) Inner circle diameter in km, 3) Minimum altitude in km
76	float	4	1	data_red_parm1	Data reduction algorithm specific parameter #1 0) unused, 1) Largest positive angle in degrees, 2) Outer circle diameter in km, 3) Maximum altitude in km
80	float	4	1	lidar_longitude	Longitude of airport from which aircraft took off. Northern hemisphere is positive, southern negative
84	float	4	1	lidar_latitude	Latitude of airport from which aircraft took off. Eastern hemisphere is positive, western negative
88	float	4	1	lidar_altitude	Altitude of airport from which aircraft took off. Up is positive, above mean sea level.
92	float	4	1	eff_unamb_vel	Effective unambiguous velocity

Offset bytes	Data type	Byte width	Count	Item	Description
96	float	4	1	eff_unamb_range	Effective unambiguous range
100	integer	4	1	num_wvlen_trans	Number of different wave lengths transmitted
104	float	4	1	prf	Pulse repetition frequency
108	float	4	10	wavelength	Wavelengths of all the different transmitted light

7.20 Field LIDAR information block – FLIB

Length: 748 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier for a field written lidar information block (ascii characters FLIB)
4	integer	4	1	nbytes	Length of this field written lidar information block in bytes
8	integer	4	1	data_sys_id	Data systems identification number
12	float	4	10	transmit_beam_div	Transmitter beam divergence. Entry [0] is for wavelength #1 etc.
52	float	4	10	xmit_power	Nominal peak transmitted power (by channel) Entry [0] is for wavelength #1 etc.
92	float	4	10	receiver_fov	Receiver field of view
132	integer	4	10	receiver_type	0 = direct detection, no polarization, 1 = direct detection polarized parallel to transmitted beam, 2 = direct detection, polarized perpendicular to transmitted beam, 3 = photon counting no polarization, 4 = photon counting polarized parallel to transmitted beam,

Offset bytes	Data type	Byte width	Count	Item	Description
					5 = photon counting, polarized perpendicular to transmitted beam
172	float	4	10	r_noise_floor	Receiver noise floor
212	float	4	10	receiver_spec_bw	Receiver spectral bandwidth
252	float	4	10	receiver_elec_bw	Receiver electronic bandwidth
292	float	4	10	calibration	0 = linear receiver, non zero log receiver
332	integer	4	1	range_delay	Delay between indication of transmitted pulse in the data system and the pulse actually leaving the telescope (can be negative)
336	float	4	10	peak_power_multi	When the measured peak transmit power is multiplied by this number it yields the actual peak transmit power
376	float	4	1	encoder_mirror_up	Encoder reading minus IRU roll angle when scan mirror is pointing directly vertically up in the roll axes
380	float	4	1	pitch_mirror_up	Scan mirror pointing angle in pitch axes, minus IRU pitch angle, when mirror is pointing directly vertically up in the roll axes
384	integer	4	1	max_digitizer_count	Maximum value (count) out of the digitizer
388	float	4	1	max_digitizer_volt	Volgage that causes the maximum count out of the digitizer
392	float	4	1	digitizer_rate	Sample rate of the digitizer
396	integer	4	1	total_num_samples	Total number of A/D samples to take
400	integer	4	1	samples_per_cell	Number of samples average in range per data cell

Offset bytes	Data type	Byte width	Count	Item	Description
404	integer	4	1	cells_per_ray	Number of data cells averaged per data ray
408	float	4	1	pmt_temp	PMT temperature
412	float	4	1	pmt_gain	D/A setting for PMT power supply
416	float	4	1	apd_temp	APD temperature
420	float	4	1	apd_gain	D/A setting for APD power supply
424	integer	4	1	transect	Transect number
428	char	1	[10] [12]	derived _names	Derived parameter names
548	char	1	[10][8]	derived_units	Derived parameter units
628	char	1	[10] [12]	temp_names	Names of the logged temperatures

7.21 In-situ description – SITU

Length: 4108 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier = SITU
4	integer	4	1	nbytes	Block size in bytes
8	integer	4	1	number_params	Number of parameters
12	char	1	8	name 1	Name 1

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
20	char	1	8	units1	Units 1
28	char	1	8	name 2	Name 2
36	char	1	8	units 2	Units 2
44	char	1	8	name 3	Name 3
52	char	1	8	units 3	Units 3
60	char	1	8	name 4	Name 4
68	char	1	8	units 4	Units 4
76	char	1	8	name 5	Name 5
84	char	1	8	units 5	Units 5
….	….	….	….	….	….
….	….	….	…..	….	….
….	….	….	….	….	….
4076	char	1	8	name 243	Name 243
4084	char	1	8	units 243	Units 243
4092	char	1	8	name 244	Name 244
4100	char	1	8	units 244	Units 244

7.22 In-situ parameters – ISIT

Length: 16 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier = ISIT
4	integer	4	1	nbytes	Block size in bytes
8	integer	2	1	julian_day	
10	integer	2	1	hours	
12	integer	2	1	minutes	
14	integer	2	1	seconds	

7.23 Independent frequency block – INDF

Length: 8 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier = INDF
4	integer	4	1	nbytes	Block size in bytes

7.24 Mini RIMS block – MINI

Length: 4112 bytes

Offset bytes	Data type	Byte width	Count	Item	Description
0	char	1	4	id	Identifier = MINI
4	integer	4	1	nbytes	Block size in bytes
8	integer	2	1	command	Current command latch setting
10	integer	2	1	status	Current status
12	float	4	1	temperature	Degrees c
16	float	4	128	x_axis_gyro	Roll axis gyro position
528	float	4	128	y_axis_gyro	Pitch axis gyro position
1040	float	4	128	z_axis_gyro	Yaw axis gyro position
1552	float	4	128	xr_axis_gyro	Roll axis reduntante gyro position
2064	float	4	128	x_axis_vel	Longitudinal axis velocity
2576	float	4	128	y_axis_vel	Lateral axis velocity
3088	float	4	128	z_axis_vel	Vertical axis velocity
3600	float	4	128	x_axis_pos	Roll axis gimbal

7.25 Nav description block – NDDS

Length: 16 bytes

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
0	char	1	4	id	Identifier = NDDS
4	integer	4	1	nbytes	Block size in bytes
8	integer	2	1	ins_flag	0 = no INS data, 1 = data recorded
10	integer	2	1	gps_flag	0 = no GPS data, 1 = data recorded
12	integer	2	1	minirims_flag	0 = no MiniRIMS data, 1 = data recorded
14	integer	2	1	kalman_flag	0 = no kalman data, 1 = data recorded

7.26 Time series header – TIME

Length: 8 bytes

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
0	char	1	4	id	Identifier = TIME
4	integer	4	1	nbytes	Block size in bytes

7.27 Waveform block – WAVE

Length: 364 bytes

Offset bytes	**Data type**	**Byte width**	**Count**	**Item**	**Description**
0	char	1	4	id	Identifier for the waveform descriptor (ascii characters “WAVE”).
4	integer	4	1	nbytes	Length of the waveform descriptor in bytes
8	char	1	16	ps_file_name	Pulsing scheme file name
24	integer	2	6	num_chips	Number of chips in a repeat sequence for each frequency
36	char	1	256	blank_chip	Blanking RAM sequence
292	float	4	1	repeat_seq	Number of milliseconds in a repeat sequence in ms
296	integer	2	1	repeat_seq_dwel	Number of repeat sequences in a dwell time
298	integer	2	1	total_pcp	Total number of PCP in a repeat sequence
300	integer	2	6	chip_offset	Number of 60 Mhz clock cycles to wait before starting a particular chip in 60 MHz counts
312	integer	2	6	chip_width	Number of 60 MHz clock cycles in each chip in 60 MHz counts
324	float	4	1	ur_pcp	Number of PCP that set the unambiguous range, after real time unfolding
328	float	4	1	uv_pcp	Number of PCP that set the unambiguous velocity after real time unfolding
332	integer	2	6	num_gates	Total number of gates sampled

Offset bytes	Data type	Byte width	Count	Item	Description
344	integer	2	2	gate_dist1	Distance from radar to data cel #1 in 60 MHz counts in 0, subsequent spacing in 1 for freq 1
348	integer	2	2	gate_dist2	Ditto for freq 2
352	integer	2	2	gate_dist3	Ditto for freq 3
356	integer	2	2	gate_dist4	Ditto for freq 4
360	integer	2	2	gate_dist5	Ditto for freq 5

8 C enumeration types

The following are the enumerated types from the DoradeData.hh header file, which defines the Dorade data format for use with C and C++ code.

```
///////////////////////////////////////////////////
// enumarated types

/// binary format

typedef enum {
  BINARY_FORMAT_INT8 = 1, /**< signed 8-bit integer */
  BINARY_FORMAT_INT16 = 2, /**< signed 16-bit integer */
  BINARY_FORMAT_INT32 = 3, /**< signed32-bit integer */
  BINARY_FORMAT_FLOAT32 = 4 /**< IEEE 32-bit float */
} binary_format_t;

/// code word bits

typedef enum {
  REFLECTIVITY_BIT  = 0x8000, /**< bit 15 */
  VELOCITY_BIT  = 0x4000, /**< bit 14 */
  WIDTH_BIT  = 0x2000, /**< bit 13 */
  TA_DATA_BIT  = 0x1000, /**< bit 12 */
  LF_DATA_BIT  = 0x0800, /**< bit 11 */
  TIME_SERIES_BIT  = 0x0400 /**< bit 10 */
} code_word_bits_t;

/// radar types

typedef enum {
  RADAR_GROUND = 0, /**< ground-based radar */
  RADAR_AIR_FORE = 1, /**< aircraft forward-looking radar (Eldora) */
  RADAR_AIR_AFT = 2, /**< aircraft aft-looking radar (Eldora) */
  RADAR_AIR_TAIL = 3, /**< aircraft tail radar */
  RADAR_AIR_LF = 4, /**< aircraft lower fuselage radar */
  RADAR_SHIP = 5, /**< ship-borne radar */
  RADAR_AIR_NOSE = 6, /**< aircraft nose radar */
  RADAR_SATELLITE = 7, /**< satellite-based radar */
  LIDAR_MOVING = 8, /**< mobile lidar - deprecated */
  LIDAR_FIXED = 9 /**< fixed lidar - deprecated */
} radar_type_t;

/// lidar types

typedef enum {
  LIDAR_GROUND = 0, /**< ground-based lidar */
  LIDAR_AIR_FORE = 1, /**< aircraft forward-looking lidar */
  LIDAR_AIR_AFT = 2, /**< aircraft aft-looking lidar */
  LIDAR_AIR_TAIL = 3, /**< aircraft tail lidar */
  LIDAR_AIR_LF = 4, /**< aircraft lower fuselage lidar */
  LIDAR_SHIP = 5, /**< ship-borne lidar */
  LIDAR_AIR_FIXED = 6, /**< fixed lidar */
  LIDAR_SATELLITE = 7 /**< satellite-based lidar */
} lidar_type_t;
```

```
/// field IDs

typedef enum {
  SW_ID_NUM = 1, /**< ID for spectrum width */
  VR_ID_NUM = 2, /**< ID for radial velocity */
  NCP_ID_NUM = 3, /**< ID for normalized coherent power */
  DBZ_ID_NUM = 4, /**< ID for dbz */
  DZ_ID_NUM = 5, /**< ID for ZDR */
  VE_ID_NUM = 6, /**< ID for radial velocity */
  VG_ID_NUM = 7, /**< ID for combined radial velocity */
  VU_ID_NUM = 8, /**< ID for radial velocity */
  VD_ID_NUM = 9, /**< ID for radial velocity */
  DBM_ID_NUM = 10 /**< ID for dbm - power */
} field_id_t;

/// polarization types

typedef enum {
  POLARIZATION_HORIZONTAL = 0, /**< horizontal polarization */
  POLARIZATION_VERTICAL = 1, /**< vertical polarization */
  POLARIZATION_CIRCULAR_RIGHT = 2, /**< right circular polarization */
  POLARIZATION_ELLIPTICAL = 3, /**< elliptical polarization */
  POLARIZATION_HV_ALT = 4, /**< dual-pol alternating polarization */
  POLARIZATION_HV_SIM = 5 /**< dual-pol simultaneous polarization */
} polarization_t;

/// scan modes

typedef enum {
  SCAN_MODE_CAL = 0, /**< calibration scan */
  SCAN_MODE_PPI = 1, /**< PPI sector scan */
  SCAN_MODE_COP = 2, /**< coplane scan */
  SCAN_MODE_RHI = 3, /**< RHI scan */
  SCAN_MODE_VER = 4, /**< vertically pointing scan */
  SCAN_MODE_TAR = 5, /**< follow target scan */
  SCAN_MODE_MAN = 6, /**< manual scan */
  SCAN_MODE_IDL = 7, /**< idle scan */
  SCAN_MODE_SUR = 8, /**< 360-deg surveillance scan */
  SCAN_MODE_AIR = 9, /**< airborne scan (e.g. eldora) */
  SCAN_MODE_HOR = 10 /**< horizontal scan */
} scan_mode_t;

/// compression types

typedef enum {
  COMPRESSION_NONE = 0, /**< no compression */
  COMPRESSION_HRD = 1, /**< run-length encoding compression */
  COMPRESSION_RDP_8_BIT = 8 /**< deprecated */
} compression_t;
```

9 C structures for data blocks

The following are the structures from the DoradeData.hh header file, which defines the Dorade data format for use with C and C++ code.

```
////////////////////////////////////////////////////////////////////////
/// DORADE DATA FORMAT
///
/// Defines the header structures used in DORADE format radar
/// files.
///
/// Dorade data files are made up of a sequence of data structures,
/// each of which has at the start a 4-character ID, followed by a
/// length in bytes.
///
/// For 2 of the structure types, id RDAT and QDAT, the structure is
/// followed by field data. The length in bytes is therefore longer
/// than the structure itself, since it includes the bytes in the
/// following data fields.
///
/// - SSWB, struct super_SWIB_t,
///   super sweep indenitifcation block
///
/// - VOLD, struct volume_t,
///   volume description block
///
/// - RADD, struct radar_t,
///   radar description block
///
/// - CFAC, struct correction_t,
///   correction factors block
///
/// - PARM, struct parameter_t,
///   parameter (data field) description block
///
/// - CELV, struct cell_t,
///   cell (gate) spacing array block
///
/// - CSFD, struct cell_spacing_fp_t,
///   cell spacing table block
///
/// - SWIB, struct sweepinfo_t,
///   sweep information description block
///
/// - ASIB, struct platform_t,
///   platform description block
///
/// - RYIB, struct ray_t,
///   ray information block
///
/// - RDAT, struct paramdata_t,
///   field parameter data block
///
/// - QDAT, struct qparamdata_t,
///   field parameter data block, extended version
///
```

```
/// - XSTF, struct extra_stuff_t,
///   miscellaneous block
///
/// - NULL, struct null_block_t,
///   null block, end of main data
///
/// - RKTB, struct rot_ang_table_t,
///   rotation angle table, after null block
///
/// - SEDS, ASCII
///   Edit history block
///
/// - FRAD, struct radar_test_status_t,
///   RADAR test pulse and status block
///
/// - FRIB, struct field_radar_t,
///   field RADAR information block
///
/// - LIDR, struct lidar_t,
///   LIDAR description
///
/// - FLIB, struct field_lidar_t,
///   field LIDAR information block
///
/// - SITU, struct insitu_descript_t,
///   in-situ description block
///
/// - ISIT, struct insitu_data_t,
///   in-situ parameters block
///
/// - INDF, struct indep_freq_t,
///   independent frequency block
///
/// - MINI, struct minirims_data_t,
///   minirims data block
///
/// - NDDS, struct nav_descript_t,
///   navigation block
///
/// - TIME, struct time_series_t,
///   time series block
///
/// - WAVE, struct waveform_t,
///   Waveform descriptor block
///
/// The most tricky part of a DORADE file is the rotation angle table
/// (RKTB). This table is stored in a block after the NULL block, and
/// is pointed to by the key_table in the SWIB block. The rotation
/// angle table comprises 3 sections:
///
/// -# The rot_angle_table_t structure at the start of the block.
/// -# An array of integers: Radx::si32[ndx_que_size], which is a
///    lookup table for locating the ray for a given angle.
/// -# An array of table entries: rot_table_entry_t[num_rays], which
///    stores the rotation_angle for each ray, as well as the offset
///    and length of the ray data in the file.

//////////////////////////////////////////////////////
// structure definitions

///////////////////////////////////////////////////////
```

```
/// Comment block - COMM

typedef struct comment {

  char id[4];  /**< Comment descriptor identifier: ASCII
                * characters "COMM" stand for Comment
                * Descriptor. */
  si32 nbytes;    /**< Comment descriptor length in bytes. */
  char  comment[500];  /**< comment*/

} comment_t;

////////////////////////////////////
// key tables - in super sweep block
//
// These point to special blocks at the end of the file

const static int MAX_KEYS = 8; /**< dimension for key table in SSIB */

/// type of entry in key table in SWIB

typedef enum {
  KEYED_BY_TIME = 1, /**< time series block */
  KEYED_BY_ROT_ANG = 2, /**< rotation angle table */
  SOLO_EDIT_SUMMARY = 3 /**< block containing ASCII editing details */
} key_table_type_t;

/// indexes for key table in SWIB

typedef enum {
  NDX_ROT_ANG = 0, /**< rotation angle table */
  NDX_SEDS  = 1 /**< editing block */
} key_table_index_t;

/// key table entries in SWIB

typedef struct key_table_info {
  si32 offset; /**< offset from start of file, in bytes */
  si32 size; /**< size of block, in bytes */
  si32 type; /**< see key_table_index_t */
} key_table_info_t;

/////////////////////////////////////
/// super sweep ident block - SSWB

typedef struct super_SWIB {

  char id[4];/**< "SSWB" */
  si32 nbytes; /**< number of bytes in this struct block */

  /**< parameters from the first version */

  si32 last_used; /**< Last time used - Unix time */
  si32 start_time; /**< start time of volume - Unix time */
  si32 stop_time; /**< end time of volume - Unix time */
  si32 sizeof_file; /**< Length of file in bytes */
  si32 compression_flag; /**< See compression_t */
  si32 volume_time_stamp; /**< to reference current volume */
  si32 num_params; /**< number of parameters (fields) */

  /**< end of first version parameters */
```

```
  char radar_name[8]; /**< radar name */

  fl64 d_start_time; /**< Volume start time, high precision */
  fl64 d_stop_time; /**< Volume end time, high precision */

  /**<
   * "last_used" is an age off indicator where > 0 implies Unix time
   * of the last access and
   * 0 implies this sweep should not be aged off
   */

  si32 version_num; /**< version number of this format */
  si32 num_key_tables; /**< number of key tables in this file */
  si32 status; /**< status */
  si32 place_holder[7]; /**< unused */
  key_table_info_t key_table[MAX_KEYS]; /**< key table */

  /**<
   * offset and key info to a table containing key value such as
   * the rot. angle and the offset to the corresponding ray
   * in the disk file
   */

} super_SWIB_t;

/// Some older files have an alternate length for the SWIB block,
/// with only the first part of the struct filled in.

static const int super_SWIB_alt_len = 200;

////////////////////////////////
/// volume description - VOLD

typedef struct volume {

  char id[4];/**< Volume descriptor identifier: ASCII
               * characters "VOLD" stand for Volume Descriptor. */
  si32 nbytes;/**< Volume descriptor length in bytes. */
  si16 format_version;/**< ELDORA/ASTRAEA field file format
                       * revision number. */
  si16 volume_num;/**< Volume Number into current file. */
  si32 maximum_bytes;/**< Maximum number of bytes in any.
                       * physical record in this volume. */
  char proj_name[20];        /**< Project number or name. */
  si16 year;/**< Year data taken in years. */
  si16 month;/**< Month data taken in months. */
  si16 day;/**< Day data taken in days. */
  si16 data_set_hour;/**< hour data taken in hours. */
  si16 data_set_minute;/**< minute data taken in minutes. */
  si16 data_set_second;/**< second data taken in seconds. */
  char flight_num[8];        /**< Flight number. */
  char gen_facility[8];/**< identifier of facility that
                       * generated this recording. */
  si16 gen_year;/**< year this recording was generated
                       * in years. */
  si16 gen_month;/**< month this recording was generated
                       * in months. */
  si16 gen_day;/**< day this recording was generated in days. */
  si16 number_sensor_des; /**< Total Number of sensor descriptors
                         * that follow. */
```

```
} volume_t;

///////////////////////////
/// radar description - RADD

typedef struct radar {

  char id[4];/**< Identifier for a radar descriptor
                         * block (ascii characters "RADD"). */
  si32 nbytes;/**< Length of a radar descriptor block in bytes. */
  char radar_name[8];/**< Eight character radar name. */
  fl32 radar_const;/**< Radar/lidar constant in ?? */
  fl32 peak_power;/**< Typical peak power of the sensor in kw.
                         * Pulse energy is really the
                         * peak_power * pulse_width */
  fl32 noise_power;/**< Typical noise power of the sensor in dBm. */
  fl32 receiver_gain;/**< Gain of the receiver in db. */
  fl32 antenna_gain;/**< Gain of the antenna in db. */
  fl32 system_gain;/**< System Gain in db.
                         * (Ant G - WG loss) */
  fl32 horz_beam_width;/**< Horizontal beam width in degrees.
                         * beam divergence in milliradians
                         * is equivalent to beamwidth */
  fl32 vert_beam_width;/**< Vertical beam width in degrees. */
  si16 radar_type;/**< Radar Type (0)Ground, 1)Airborne
                         * Fore, 2)Airborne Aft, 3)airborne
                         * Tail, 4)Airborne Lower Fuselage,
                         * 5)Shipborne. */
  si16 scan_mode;/**< Scan Mode (0)Calibration, 1)PPI
                         * (constant Elevation) 2)Coplane,
                         * 3)RHI (Constant Azimuth), 4)
                         * Vertical Pointing, 5)Target
                         * (Stationary), 6)Manual, 7)Idle (Out
                         * of Control). */
  fl32 req_rotat_vel;/**< Requested rotational velocity of
                         * the antenna in degrees/sec. */
  fl32 scan_mode_pram0;/**< Scan mode specific parameter #0
                         * (Has different meaning for
                         * different scan modes). */
  fl32 scan_mode_pram1;/**< Scan mode specific parameter #1. */
  si16 num_parameter_des;/**< Total number of parameter
                          * descriptor blocks for this radar. */
  si16 total_num_des;/**< Total number of additional
                         * descriptor block for this radar. */
  si16 data_compress;/**< Data compression. 0 = none, 1 = HRD
                         * scheme. */
  si16 data_reduction;/**< Data Reduction algorithm: 1 = none,
                         * 2 = between 2 angles, 3 = Between
                         * concentric circles, 4 = Above/below
                         * certain altitudes.*/
  fl32 data_red_parm0;/**< 1 = smallest positive angle in
                         * degrees, 2 = inner circle diameter,
                         * km, 4 = minimum altitude, km. */
  fl32 data_red_parm1;/**< 1 = largest positve angle, degress,
                         * 2 = outer cicle diameter, km, 4 =
                         * maximum altitude, km. */
  fl32 radar_longitude;/**< longitude of radar in degrees. */
  fl32 radar_latitude;/**< Latitude of radar in degrees. */
  fl32 radar_altitude;/**<  Altitude of radar above msl in km. */
  fl32 eff_unamb_vel;/**< Effective unambiguous velocity, m/s. */
  fl32 eff_unamb_range;/**< Effective unambiguous range, km. */
```

```
  si16 num_freq_trans;/**< Number of frequencies transmitted. */
  si16 num_ipps_trans;/**< Number of different inter-pulse
                         * periods transmitted. */
  fl32 freq1;/**< Frequency 1. */
  fl32 freq2;/**< Frequency 2. */
  fl32 freq3;/**< Frequency 3. */
  fl32 freq4;/**< Frequency 4. */
  fl32 freq5;/**< Frequency 5. */
  fl32 prt1;      /**< Interpulse period 1. */
  fl32 prt2;            /**< Interpulse period 2. */
  fl32 prt3;            /**< Interpulse period 3. */
  fl32 prt4;            /**< Interpulse period 4. */
  fl32 prt5;            /**< Interpulse period 5. */

  /**< 1995 extension #1 */
  si32  extension_num;  /**< not sure */
  char config_name[8];/**< used to identify this set of
                         * unique radar characteristics */
  si32  config_num;/**< facilitates a quick lookup of radar
                         * characteristics for each ray */
  /**
   * extend the radar descriptor to include unique lidar parameters
   */
  fl32 aperture_size;  /**< Diameter of the lidar aperature in cm. */
  fl32 field_of_view;  /**< Field of view of the receiver. mra; */
  fl32 aperture_eff;   /**< Aperature efficiency in %. */
  fl32 aux_freq[11];   /**< make space for a total of 16 freqs */
  fl32 aux_prt[11];    /**< and ipps */

  /**
   * other extensions to the radar descriptor
   */
  fl32 pulse_width;  /**< Typical pulse width in microseconds.
                      * pulse width is inverse of the
                      * band width */
  fl32 primary_cop_baseln; /**< coplane baselines */
  fl32 secondary_cop_baseln; /**< not sure */
  fl32 pc_xmtr_bandwidth;  /**< pulse compression
                            * transmitter bandwidth */
  si32  pc_waveform_type;  /**< pulse compression waveform type */
  char site_name[20]; /**< instrument site name */

} radar_t;

/// some files have an older structure, with only the
/// first part filled in

static const int radar_alt_len = 144;

/////////////////////////////
/// correction factors - CFAC

typedef struct correction {

  char id[4];/**< Correction descriptor identifier: ASCII
                 * characters "CFAC" stand for Volume
                 * Descriptor. */
  si32 nbytes; /**<Correction  descriptor length in bytes. */
  fl32 azimuth_corr;     /**< Correction added to azimuth[deg] */
  fl32 elevation_corr;   /**< Correction added to elevation[deg] */
  fl32 range_delay_corr; /**< Correction used for range delay[m] */
```

```
  fl32 longitude_corr;   /**< Correction added to radar longitude */
  fl32 latitude_corr;    /**< Correction added to radar latitude */
  fl32 pressure_alt_corr;/**< Correction added to pressure altitude
                          * (msl)[km] */
  fl32 radar_alt_corr;   /**< Correction added to radar altitude above
                          * ground level(agl) [km] */
  fl32 ew_gndspd_corr;   /**< Correction added to radar platform
                          * ground speed(E-W)[m/s] */
  fl32 ns_gndspd_corr;   /**< Correction added to radar platform
                          * ground speed(N-S)[m/s] */
  fl32 vert_vel_corr;    /**< Correction added to radar platform
                          * vertical velocity[m/s] */
  fl32 heading_corr;     /**< Correction added to radar platform
                          * heading [deg]) */
  fl32 roll_corr;        /**< Correction added to radar platform
                          * roll[deg] */
  fl32 pitch_corr;       /**< Correction added to radar platform
                          * pitch[deg] */
  fl32 drift_corr;       /**< Correction added to radar platform
                          * drift[deg] */
  fl32 rot_angle_corr;   /**< Corrrection add to radar rotation angle
                          *[deg] */
  fl32 tilt_corr;        /**< Correction added to radar tilt angle */

} correction_t;

/////////////////////////////////////////////
/// parameter (data field) description - PARM

typedef struct parameter {

  char id[4];/**< Parameter Descriptor identifier
               * (ascii characters "PARM"). */
  si32 nbytes;/**< Parameter Descriptor length in bytes.*/
  char parameter_name[8]; /**< Name of parameter being described. */
  char param_description[40]; /**< Detailed description of this parameter.
*/
  char param_units[8];/**< Units parameter is written in. */
  si16 interpulse_time;/**< Inter-pulse periods used. bits 0-1
                         * = frequencies 1-2. */
  si16 xmitted_freq;/**< Frequencies used for this
                         * parameter. */
  fl32 recvr_bandwidth;/**< Effective receiver bandwidth for
                         * this parameter in MHz.*/
  si16 pulse_width;/**< Effective pulse width of parameter
                         * in m. */
  si16 polarization;/**< Polarization of the radar beam for
                         * this parameter (0 Horizontal,1
                         * Vertical,2 Circular,3 Elliptical) in na. */
  si16 num_samples;/**< Number of samples used in estimate
                         * for this parameter. */
  si16 binary_format;/**< Binary format of radar data. */
  char  threshold_field[8];/**< Name of parameter upon which this
                                * parameter is thresholded (ascii
                                * characters NONE if not
                                * thresholded). */
  fl32 threshold_value;/**< Value of threshold in ? */
  fl32 parameter_scale;/**< Scale factor for parameter. */
  fl32 parameter_bias;/**< Bias factor for parameter. */
  si32  bad_data;/**< Bad data flag. */
```

```
  /**< 1995 extension #1 */

  si32 extension_num;   /**< not sure */
  char config_name[8];/**< used to identify this set of
                         * unique radar characteristics */
  si32 config_num;      /**< not sure */
  si32 offset_to_data;/**< bytes added to the data struct pointer
                         * to point to the first datum whether it's
                         * an RDAT or a QDAT
                         */
  fl32 mks_conversion;  /**< not sure */
  si32 num_qnames;/**< not sure */
  char qdata_names[32];/**< each of 4 names occupies 8 characters
                         * of this space
                         * and is blank filled. Each name identifies
                         * some interesting segment of data in a
                         * particular ray for this parameter.
                         */
  si32 num_criteria;    /**< not sure */
  char criteria_names[32];/**< each of 4 names occupies 8 characters
 * and is blank filled. These names identify
 * a single interesting fl32ing point value
 * that is associated with a particular ray
 * for a this parameter. Examples might
 * be a brightness temperature or
 * the percentage of cells above or
 * below a certain value */
  si32 number_cells;    /**< number of gates */
  fl32 meters_to_first_cell; /**< distance to center - meters */
  fl32 meters_between_cells; /**< gate spacing - meters */
  fl32 eff_unamb_vel;/**< Effective unambiguous velocity, m/s. */

} parameter_t;

/// some files have an older structure, with only the
/// first part filled in

static const int parameter_alt_len = 104;

/// dimension of dist_cells in cell_vector_t

static const int MAXCVGATES = 1500;

//////////////////////////////
/// cell (gate) spacing - CELV

typedef struct cell {

  char id[4]; /**< Cell descriptor identifier: ASCII
               * characters "CELV" stand for cell vector. */
  si32 nbytes;     /**< Comment descriptor length in bytes */
  si32 number_cells; /**< Number of sample volumes */
  fl32 dist_cells[MAXCVGATES]; /**< Distance from the radar to cell
                                * n in meters */

} cell_vector_t;

//////////////////////////////
/// cell spacing table - CSFD

typedef struct cell_spacing_fp {
```

```
  char id[4];   /**< Identifier for a cell spacing descriptor
                 * (ascii characters CSFD). */
  si32 nbytes;  /**< Cell Spacing descriptor length in bytes. */
  si32 num_segments;  /**< Number of segments that contain cells of */
  fl32 dist_to_first; /**< Distance to first gate in meters. */
  fl32 spacing[8];    /**< Width of cells in each segment in m. */
  si16 num_cells[8] ; /**< Number of cells in each segment.
                       * equal widths. */

} cell_spacing_fp_t;

////////////////////////////////////////
/// sweep information description - SWIB

typedef struct sweepinfo {
  char id[4];            /**< Comment descriptor identifier: ASCII
                          * characters "SWIB" stand for sweep info
                          * block Descriptor. */
  si32 nbytes;     /**< Sweep  descriptor length in bytes. */
  char radar_name[8];    /**< comment*/
  si32 sweep_num;  /**< Sweep number from the beginning of the volume*/
  si32 num_rays;   /**<number of rays recorded in this sweep*/
  fl32 start_angle;/**<true start angle [deg]*/
  fl32 stop_angle; /**<true stop angle  [deg]*/
  fl32 fixed_angle; /**< not sure */
  si32 filter_flag; /**< not sure */

} sweepinfo_t;

///////////////////////////////
/// platform description - ASIB

typedef struct platform {

  char id[4];/**< Identifier for the aircraft/ship
                 * parameters block (ascii characters ASIB) */
  si32 nbytes;/**< Length in Bytes of the
               * aircraft/ship arameters block */
  fl32 longitude;/**< Antenna longitude (Eastern
                        * Hemisphere is positive, West
                        * negative) in degrees */
  fl32 latitude;/**< Antenna Latitude (Northern
                        * Hemisphere is positive, South
                        * Negative) in degrees */
  fl32 altitude_msl;/**< Antenna Altitude above mean sea
                        * level (MSL) in km */
  fl32 altitude_agl;/**< Antenna Altitude above ground level
                        * (AGL) in km */
  fl32 ew_velocity;/**< Antenna east-west ground speed
                        * (towards East is positive) in m/sec */
  fl32 ns_velocity;/**< Antenna north-south ground speed
                        * (towards North is positive) in m/sec */
  fl32 vert_velocity;/**< Antenna vertical velocity (Up is
                        * positive) in m/sec */
  fl32 heading;/**< Antenna heading (angle between
                        * rotodome rotational axis and true
                        * North, clockwise (looking down)
                        * positive) in degrees */
  fl32 roll;/**< Roll angle of aircraft tail section
                        * (Horizontal zero, Positive left wing up)
```

```
                         * in degrees */
  fl32 pitch;/**< Pitch angle of rotodome (Horizontal
                         * is zero positive front up) in degrees*/
  fl32 drift_angle;/**< Antenna drift Angle. (angle between
                         * platform true velocity and heading,
                         * positive is drift more clockwise
                         * looking down) in degrees */
  fl32 rotation_angle;/**< Angle of the radar beam with
                         * respect to the airframe (zero is
                         * along vertical stabilizer, positive
                         * is clockwise) in deg */
  fl32 tilt;/**< Angle of radar beam and line normal
                         * to longitudinal axis of aircraft,
                         * positive is towards nose of
                         * aircraft) in degrees */
  fl32 ew_horiz_wind;/**< east - west wind velocity at the
                         * platform (towards East is positive)
                         * in m/sec */
  fl32 ns_horiz_wind;/**< North - South wind velocity at the
                         * platform (towards North is
                         * positive) in m/sec */
  fl32 vert_wind;/**< Vertical wind velocity at the
                         * platform (up is positive) in m/sec */
  fl32 heading_change;/**< Heading change rate in degrees/second. */
  fl32 pitch_change;/**< Pitch change rate in degrees/second. */

} platform_t;

//////////////////////////////////////////////////////////////////////
/// ray information - RYIB

typedef struct ray {

  char id[4];/**< Identifier for a data ray info.
                         * block (ascii characters "RYIB"). */
  si32 nbytes;/**< length of a data ray info block in bytes. */
  si32 sweep_num;/**< sweep number for this radar. */
  si32 julian_day;      /**< Guess. */
  si16 hour;/**< Hour in hours. */
  si16 minute;/**< Minute in minutes. */
  si16 second;/**< Second in seconds. */
  si16 millisecond;/**< Millisecond in milliseconds. */
  fl32 azimuth;/**< Azimuth in degrees. */
  fl32 elevation;/**< Elevation in degrees. */
  fl32 peak_power;/**< Last measured peak transmitted
                         * power in kw. */
  fl32 true_scan_rate;/**< Actual scan rate in degrees/second. */
  si32 ray_status;/**< 0 = normal, 1 = transition, 2 = bad. */

} ray_t;

////////////////////////////////////////
/// field parameter data - RDAT

typedef struct paramdata {

  char id[4];         /**< parameter data descriptor identifier: ASCII
                          * characters "RDAT" stand for parameter data
                          * block Descriptor. */
  si32 nbytes;   /**< parameter data descriptor length in bytes. */
  char pdata_name[8];  /**< name of parameter */
```

```
} paramdata_t;

////////////////////////////////////////////
/// field parameter data - extended - QDAT

typedef struct qparamdata {

  char id[4]; /**< parameter data descriptor identifier: ASCII
               * characters "QDAT" for a block that contains
               * the data plus some supplemental and
               * identifying information */

  si32 nbytes; /**< parameter data descriptor length in bytes.
                * this represents the size of this header
                * information plus the data
                *
                * for this data block the start of the data
                * is determined by using "offset_to_data"
                * in the corresponding parameter descriptor
                * "struct parameter_d"
                * the offset is from the beginning of
                * this descriptor/block
                */

  char pdata_name[8];/**< name of parameter */

  si32 extension_num; /**< not sure */
  si32 config_num; /**< facilitates indexing into an array
                     * of radar descriptors where the radar
                     * characteristics of each ray and each
                     * parameter might be unique such as phased
                     * array antennas */

  si16 first_cell[4]; /**< see num_cells */
  si16 num_cells[4]; /**< first cell and num cells demark
                       * some feature in the data and it's
                       * relation to the cell vector
                       * first_cell[n] = 0 implies the first datum
                       * present corresponds to "dist_cells[0]
                       * in "struct cell_d"
                       * for TRMM data this would be the
                       * nominal sample where the cell vector is
                       * at 125 meter resolution instead of 250 meters
                       * and identified segments might be the
                       * rain echo oversample "RAIN_ECH" and the
                       * surface oversample "SURFACE" */

  fl32 criteria_value[4]; /**< criteria value associated
                            * with a criteria name
                            * in "struct parameter_d" */
} qparamdata_t;


//////////////////////////
/// extra stuff - XSTF

typedef struct extra_stuff {

  char id[4];/**< "XSTF" */
  si32 nbytes; /**< number of bytesin this struct */
```

```
  si32 one;/**< always set to one (endian flag) */
  si32 source_format;/**< as per ../include/dd_defines.h */

  si32 offset_to_first_item; /**< bytes from start of struct */
  si32 transition_flag; /**< beam in transition? */

} extra_stuff_t;

////////////////////////////
/// null block - NULL

typedef struct null_block {

  char id[4];/**< "NULL" */
  si32 nbytes; /**< number of bytesin this struct */

} null_block_t;

/// entry for rotation angle table

typedef struct rot_table_entry {
  fl32 rotation_angle; /**< azimuth or elevation angle, depending
                        * on scan mode */
  si32 offset; /**< offset of ray from start of file, in bytes */
  si32 size; /**< ray data length, in bytes */
} rot_table_entry_t;

/// rotation angle table - RKTB block

typedef struct rot_ang_table {
  char id[4];/**< "RKTB" */
  si32 nbytes; /**< number of bytesin this struct */
  fl32 angle2ndx; /**< ratio 360.0 / ndx_que_size */
  si32 ndx_que_size; /**< lookup table size */
  si32 first_key_offset; /**< offset of start of lookup table,
                          * from start of file, in bytes */
  si32 angle_table_offset; /**< offset of start of angle table,
                            * from start of file, in bytes */
  si32 num_rays; /**< number of rays in file */
} rot_angle_table_t;

/// radar angles for
/// on-the-fly utility structure, not for file storage

typedef struct radar_angles {
  double azimuth; /**< azimuth in degrees */
  double elevation; /**< azimuth in degrees */
  double x; /**< Cartesian X */
  double y; /**< Cartesian Y */
  double z; /**< Cartesian Z */
  double psi; /**< not sure */
  double rotation_angle; /**< rotation in degrees */
  double tilt; /**< tilt in degrees */
} radar_angles_t;

/////////////////////////////////////////////
/// Radar test pulse and status block - FRAD

typedef struct radar_test_status {
```

```
  char id[4]; /**< Field parameter data identifier
               * (ascii characters FRAD) */
  si32 nbytes; /**< Length of the field parameter
                * data block in bytes */
  si32 data_sys_status;/**< Status word, bits will be assigned
                         *  particular status when needed */
  char radar_name[8];/**< Name of radar from which this data ray
                         * came from */
  fl32 test_pulse_level; /**< Test pulse power level as measured by the
                          *  power meter in dbm */
  fl32 test_pulse_dist; /**< Distance from antenna to middle of
                         * test pulse in km */
  fl32 test_pulse_width; /**< Test pulse width in m  */
  fl32 test_pulse_freq; /**< Test pulse frequency in Ghz */
  si16 test_pulse_atten; /**< Test pulse attenuation in db */
  si16 test_pulse_fnum; /**< Frequency number being calibrated
                         * with the test pulse (what mux on
                         * timing module is set to) */
  fl32 noise_power; /**< Total estimated noise power in dbm */
  si32 ray_count; /**< Data Ray counter For this
                   * particular type of data ray */
  si16 first_rec_gate; /**< First recorded gate number (N) */
  si16 last_rec_gate; /**< Last recorded gate number (M) */

} radar_test_status_t;

////////////////////////////////////////
/// Field radar information block - FRIB

typedef struct field_radar {

  char id[4];/**< Identifier for a field written
                * radar information block
                * (ascii characters FRIB). */
  si32 nbytes;/**< Length of this field written radar
                * information block in bytes. */
  si32 data_sys_id; /**< Data system identification. */
  fl32 loss_out; /**< Waveguide Losses between Transmitter and
                  * antenna in db. */
  fl32 loss_in;/**< Waveguide Losses between antenna and Low
                * noise amplifier in db. */
  fl32 loss_rjoint; /**< Losses in the rotary joint in db. */
  fl32 ant_v_dim; /**< Antenna Vertical Dimension in m. */
  fl32 ant_h_dim; /**< Antenna Horizontal Dimension in m. */
  fl32 ant_noise_temp; /**< Antenna Noise Temperature in degrees K. */
  fl32 r_noise_figure; /**< Receiver noise figure in dB*/
  fl32 xmit_power[5]; /**< Nominal Peak transmitted power in dBm
                       * by channel */
  fl32 x_band_gain; /**< X band gain in dB */
  fl32 receiver_gain[5]; /**< Measured receiver gain in dB (by channel) */
  fl32 if_gain[5]; /**< Measured IF gain in dB (by channel) */
  fl32 conversion_gain; /**< A to D conversion gain in dB */
  fl32 scale_factor[5]; /**< Scale factor to account for differences in
                         * the individual channels, and the inherent
                         * gain due to summing over the dwell time */
  fl32 processor_const; /**< Constant used to scale dBz to
                         * units the display processors understand */
  si32 dly_tube_antenna; /**< Time delay from RF being applied to
                          * tube and energy leaving antenna in ns. */
  si32 dly_rndtrip_chip_atod; /**< Time delay from a chip generated in
                               * the yiming module and the RF pulse
```

```
                                * entering the A to D converters.
                                * Need to take the RF input to the HPA
                                * and inject it into the waveguide back
                                * at the LNA to make this measurement
                                * in ns */
  si32 dly_timmod_testpulse; /**< Time delay from timing Module test
                               * pulse edge and test pulse arriving at
                               * the A/D converter in ns. */
  si32 dly_modulator_on; /**< Modulator rise time (Time between
                           * video on into HPA and modulator full up in
                           * the high power amplifier) in ns. */
  si32 dly_modulator_off; /**< Modulator fall time (Time between
                            * video off into the HPA
                            * and modulator full off) in ns. */
  fl32 peak_power_offset; /**< Added to the power meter reading of the
                            * peak output power this yields actual
                            * peak output power (in dB) */
  fl32 test_pulse_offset; /**< Added to the power meter reading of the
                            * test pulse this yields actual injected
                            * test pulse power (dB) */
  fl32 E_plane_angle; /**< E-plane angle (tilt) this is the angle in
                        * the horizontal plane (when antennas are
                        * vertical) between a line normal to the
                        * aircraft's longitudinal axis and the radar
                        * beam in degrees.  Positive is in direction
                        * of motion (fore) */
  fl32 H_plane_angle; /**< H plane angle in degrees - this follows
                        * the sign convention described in the
                        * DORADE documentation for ROLL angle */
  fl32 encoder_antenna_up; /**< Encoder reading minus IRU roll angle
                             * when antenna is up and horizontal */
  fl32 pitch_antenna_up; /**< Antenna pitch angle (measured with
                           * transit) minus IRU pitch angle when
                           * antenna is pointing up */
  si16 indepf_times_flg; /**< 0 = neither recorded, 1 = independent
                           * frequency data only, 3 = independent
                           * frequency and time series data recorded */
  si16 indep_freq_gate; /**< gate number where the independent frequency
                          * data comes from */
  si16 time_series_gate; /**< gate number where the time series data come
                           * from */
  si16 num_base_params; /**< Number of base parameters. */
  char  file_name[80]; /**< Name of this header file. */

} field_radar_t;

////////////////////////////////////////
/// Lidar description - LIDR

typedef struct lidar {

  char id[4]; /**< Identifier  a lidar descriptor
               * block (four ASCII characters
               * "LIDR"). */
  si32 nbytes; /**< Length of a lidar descriptor block. */
  char lidar_name[8]; /**< Eight character lidar
                        * name. (Characters SABL) */
  fl32 lidar_const; /**< Lidar constant. */
  fl32 pulse_energy; /**< Typical pulse energy of the lidar. */
  fl32 peak_power; /**< Typical peak power of the lidar. */
  fl32 pulse_width; /**< Typical pulse width. */
```

```
fl32 aperture_size; /**< Diameter of the lidar aperture. */
fl32 field_of_view; /**< Field of view of the receiver. mra; */
fl32 aperture_eff; /**< Aperture efficiency. */
fl32 beam_divergence; /**< Beam divergence. */
si16 lidar_type; /**< Lidar type: 0) Ground,  1) Airborne
                  * fore,  2) Airborne aft,  3)
                  * Airborne tail,  4) Airborne lower
                  * fuselage,  5) Shipborne. 6)
                  * Airborne Fixed */
si16 scan_mode; /**< Scan mode:  0) Calibration,  1) PPI
                 * (constant elevation),  2) Co-plane,
                 * 3) RHI (Constant azimuth),  4)
                 * Vertical pointing up,  5) Target
                 * (stationary),  6) Manual,  7) Idle
                 * (out of control), 8) Surveillance,
                 * 9) Vertical sweep, 10) Vertical
                 * scan. 11) Vertical pointing down,
                 * 12 Horizontal pointing right, 13)
                 * Horizontal pointing left */
fl32 req_rotat_vel; /**< Requested rotational velocity of
                     * the scan mirror. */
fl32 scan_mode_pram0; /**< Scan mode specific parameter #0
                       * (Has different meanings for
                       * different scan modes) (Start angle
                       * for vertical scanning). */
fl32 scan_mode_pram1; /**< Scan mode specific parameter #1
                       * (Has different meaning for
                       * different scan modes) (Stop angle
                       * for vertical scanning). */
si16 num_parameter_des; /**< Total number of parameter
                         * descriptor blocks for this lidar. */
si16 total_num_des; /**< Total number of all descriptor
                     * blocks for this lidar. */
si16 data_compress; /**< Data compression scheme in use:  0)
                     * no data compression, 1) using HRD
                     * compression scheme. */
si16 data_reduction; /**< Data reduction algorithm in use:
                      * 0) None, 1) Between two angles, 2)
                      * Between concentric circles. 3)
                      * Above and below certain altitudes. */
fl32 data_red_parm0; /**< Data reduction algorithm specific/
                      * parameter  #0:  0) Unused, 1)
                      * Smallest positive angle in degrees,
                      * 2) Inner circle diameter in km,  3)
                      * Minimum altitude in km. */
fl32 data_red_parm1; /**< Data reduction algorithm specific
                      * parameter  #1 0) unused, 1) Largest
                      * positive angle in degrees, 2) Outer
                      * circle diameter in km,  3) Maximum
                      * altitude in km. */
fl32 lidar_longitude; /**< Longitude of airport from which
                       * aircraft took off  northern
                       * hemisphere is positive, southern
                       * negative. */
fl32 lidar_latitude; /**< Latitude of airport from which
                      * aircraft took off eastern
                      * hemisphere is positive, western
                      * negative. */
fl32 lidar_altitude; /**< Altitude of airport from which
                      * aircraft took off up is positive,
                      * above mean sea level. */
```

```
  fl32 eff_unamb_vel; /**< Effective unambiguous velocity. */
  fl32 eff_unamb_range; /**< Effective unambiguous range. */
  si32 num_wvlen_trans; /**< Number of different wave lengths
                         * transmitted. */
  fl32 prf; /**< Pulse repetition frequency. */
  fl32 wavelength[10]; /**< Wavelengths of all the different
                        * transmitted light. */
} lidar_t;

////////////////////////////////////////
/// Field lidar information block - FLIB

typedef struct field_lidar {

  char id[4];/**< Identifier for a field written
                 * lidar information block
                 * (ascii characters FLIB). */
  si32 nbytes;/**< Length of this field written lidar
                 * information block in bytes. */
  si32 data_sys_id; /**< Data system identification number. */
  fl32 transmit_beam_div[10]; /**< Transmitter beam divergence. Entry
                               * [0] is for wavelength #1 etc. */
  fl32 xmit_power[10]; /**< Nominal peak transmitted power (by
                        * channel). Entry [0] is for
                        * wavelength #1 etc. */
  fl32 receiver_fov[10]; /**< Receiver field of view. */
  si32 receiver_type[10]; /**< 0=direct detection,no
                           * polarization,1=direct detection
                           * polarized parallel to transmitted
                           * beam,2 = direct detection,
                           * polarized perpendicular to
                           * transmitted beam,3= photon counting
                           * no polarization, 4= photon counting
                           * polarized parallel to transmitted
                           * beam,5 = photon counting, polarized
                           * perpendicular to transmitted beam. */
  fl32 r_noise_floor[10];    /**< Receiver noise floor. */
  fl32 receiver_spec_bw[10]; /**< Receiver spectral bandwidth */
  fl32 receiver_elec_bw[10]; /**< Receiver electronic bandwidth */
  fl32 calibration[10];      /**< 0 = linear receiver,  non zero log
                              * reciever */
  si32 range_delay; /**< Delay between indication of
                     * transmitted pulse in the data
                     * system and the pulse actually
                     * leaving the telescope (can be
                     * negative). */
  fl32 peak_power_multi[10]; /**< When the measured peak transmit
                              * power is multiplied by this number
                              * it yields the actual peak transmit
                              * power. */
  fl32 encoder_mirror_up; /**< Encoder reading minus IRU roll
                           * angle when scan mirror is pointing
                           * directly vertically up in the roll
                           * axes. */
  fl32 pitch_mirror_up; /**< Scan mirror pointing angle in pitch
                         * axes, minus IRU pitch angle, when
                         * mirror is pointing directly
                         * vertically up in the roll axes. */
  si32 max_digitizer_count; /**< Maximum value (count) out of the
                             * digitizer  */
  fl32 max_digitizer_volt; /**< Voltage that causes the maximum
```

```
                         * count out of the digitizer. */
  fl32 digitizer_rate; /**< Sample rate of the digitizer. */
  si32 total_num_samples; /**< Total number of A/D samples to
                         * take. */
  si32 samples_per_cell; /**< Number of samples average in range
                          per data cell. */
  si32 cells_per_ray; /**< Number of data cells averaged
                       per data ray. */
  fl32 pmt_temp; /**< PMT temperature */
  fl32 pmt_gain; /**< D/A setting for PMT power supply */
  fl32 apd_temp; /**< APD temperature */
  fl32 apd_gain; /**< D/A setting for APD power supply */
  si32 transect; /**< transect number */
  char derived_names[10][12]; /**< Derived parameter names */
  char derived_units[10][8]; /**< Derived parameter units */
  char temp_names[10][12]; /**< Names of the logged temperatures */

} field_lidar_t;

///////////////////////////////////////////////
/// entry for params list in insitu_descript_t

typedef struct insitu_parameter {
  char name[8]; /**< parameter name */
  char units[8]; /**< parameter units */
} insitu_parameter_t;

//////////////////////////////
/// in-situ parameters - SITU

typedef struct insitu_descript {
  char id[4]; /**< Identifier = SITU. */
  si32 nbytes; /**< Block size in bytes. */
  si32 number_params;/**< Number of paramters. */
  insitu_parameter_t params[256]; /**< Is this enough? */
} insitu_descript_t;

//////////////////////////////
/// in-situ parameters - ISIT

typedef struct insitu_data {
  char id[4];/**< Identifier = ISIT. */
  si32 nbytes;/**< Block size in bytes. */
  si16 julian_day; /**< day in year */
  si16 hours; /**< time - hours */
  si16 minutes; /**< time - minutes */
  si16 seconds; /**< time - seconds */
} insitu_data_t;

/////////////////////////////////
/// independent frequency - INDF

typedef struct indep_freq {
  char id[4];/**< Identifier = INDF. */
  si32 nbytes;/**< Block size in bytes. */
} indep_freq_t;

/////////////////////////////////
/// MiniRims data - MINI

typedef struct minirims_data {
```

```
  char id[4]; /**< Identifier = MINI. */
  si32 nbytes; /**< Block size in bytes. */
  si16 command; /**< Current command latch setting. */
  si16 status; /**< Current status. */
  fl32 temperature; /**< Degrees C. */
  fl32 x_axis_gyro[128]; /**< Roll axis gyro position. */
  fl32 y_axis_gyro[128]; /**< Pitch axis gyro position. */
  fl32 z_axis_gyro[128]; /**< Yaw axis gyro position. */
  fl32 xr_axis_gyro[128]; /**< Roll axis redundate gyro position. */
  fl32 x_axis_vel[128]; /**< Longitudinal axis velocity. */
  fl32 y_axis_vel[128]; /**< Lateral axis velocity. */
  fl32 z_axis_vel[128]; /**< Vertical axis velocity. */
  fl32 x_axis_pos[128]; /**< Roll axis gimbal. */
} minirims_data_t;

/////////////////////////////////
/// Nav description - NDDS

typedef struct nav_descript {
  char id[4]; /**< Identifier = NDDS. */
  si32 nbytes; /**< Block size in bytes. */
  si16 ins_flag; /**< 0 = no INS data, 1 = data recorded. */
  si16 gps_flag; /**< 0 = no GPS data, 1 = data recorded. */
  si16 minirims_flag; /**< 0 = no MiniRIMS data, 1 = data recorded. */
  si16 kalman_flag; /**< 0 = no kalman data, 1 = data recorded. */
} nav_descript_t;

/////////////////////////////////
/// Time series data header - TIME

typedef struct time_series {
  char id[4];/**< Identifier = TIME. */
  si32 nbytes; /**< Block size in bytes. */
} time_series_t;

/////////////////////////////////
/// Waveform descriptor - WAVE

typedef struct waveform {

  char id[4];/**< Identifier for the waveform
               * descriptor (ascii characters "WAVE"). */
  si32 nbytes; /**< Length of the waveform descriptor
              * in bytes. */
  char ps_file_name[16]; /**< Pulsing scheme file name.*/
  si16 num_chips[6]; /**< Number of chips in a repeat.
                    * sequence for each frequency. */
  char blank_chip[256]; /**< Blanking RAM sequence. */
  fl32 repeat_seq; /**< Number of milliseconds in a repeat
                  * sequence in ms. */
  si16 repeat_seq_dwel;/**< Number of repeat sequences in a
                       * dwell time. */
  si16 total_pcp; /**< Total Number of PCP in a repeat sequence. */
  si16 chip_offset[6]; /**< Number of 60 Mhz clock cycles to
                        * wait before starting a particular
                        * chip in 60 MHz counts. */
  si16 chip_width[6]; /**< Number of 60 Mhz clock cycles in
                       * each chip in 60 MHz counts. */
  fl32 ur_pcp; /**< Number of PCP that set the
              * unambiguous range, after real time
              * unfolding. */
```

```
  fl32 uv_pcp; /**< Number of PCP that set the
                * unambiguous velocity, after real
                * time unfolding. */
  si16 num_gates[6]; /**< Total number of gates sampled. */
  si16 gate_dist1[2]; /**< Distance from radar to data cell #1
                       * in 60 MHz counts in 0, subsequent
                       * spacing in 1 for freq 1. */
  si16 gate_dist2[2]; /**< Ditto for freq 2. */
  si16 gate_dist3[2]; /**< Ditto for freq 3. */
  si16 gate_dist4[2]; /**< Ditto for freq 4. */
  si16 gate_dist5[2]; /**< Ditto for freq 5. */

} waveform_t;
```

10 Dorade format printout from RadxPrint

TheRadxPrint utility, when used with the –dorade_format command line argument, prints out the dorade format sizes and offsets, using the C structures in the previous section.

The following is the resulting text.

```
================= DORADE FORMAT ==================
--------------------------------------------------
  struct: 'comment_t'
  size: 508
  id: COMM

     type                     name    size  offset
     ----                     ----    ----  ------
     char                    id[4]       4       0
     si32                   nbytes       4       4
     char             comment[500]     500       8
--------------------------------------------------
--------------------------------------------------
  struct: 'super_SWIB_t'
  size: 196
  id: SSWB

     type                     name    size  offset
     ----                     ----    ----  ------
     char                    id[4]       4       0
     si32                   nbytes       4       4
     si32                last_used       4       8
     si32               start_time       4      12
     si32                stop_time       4      16
     si32              sizeof_file       4      20
     si32         compression_flag       4      24
     si32        volume_time_stamp       4      28
     si32               num_params       4      32
     char            radar_name[8]       8      36
     fl64             d_start_time       8      44
     fl64              d_stop_time       8      52
     si32              version_num       4      60
     si32           num_key_tables       4      64
     si32                   status       4      68
     si32           place_holder[7]     28      72
  key_table:
     si32      key_table[0].offset       4     100
     si32      key_table[0].size         4     104
     si32      key_table[0].type         4     108
     si32      key_table[1].offset       4     112
     si32      key_table[1].size         4     116
     si32      key_table[1].type         4     120
     ....
     ....
     ....
     ....
     si32      key_table[6].offset       4     172
     si32      key_table[6].size         4     176
     si32      key_table[6].type         4     180
```

```
    si32      key_table[7].offset       4     184
    si32      key_table[7].size         4     188
    si32      key_table[7].type         4     192
--------------------------------------------------
--------------------------------------------------
  struct: 'volume_t'
  size: 72
  id: VOLD

    type                      name    size  offset
    ----                      ----    ----  ------
    char                     id[4]       4       0
    si32                    nbytes       4       4
    si16            format_version       2       8
    si16                volume_num       2      10
    si32             maximum_bytes       4      12
    char             proj_name[20]      20      16
    si16                      year       2      36
    si16                     month       2      38
    si16                       day       2      40
    si16             data_set_hour       2      42
    si16           data_set_minute       2      44
    si16           data_set_second       2      46
    char             flight_num[8]       8      48
    char           gen_facility[8]       8      56
    si16                  gen_year       2      64
    si16                 gen_month       2      66
    si16                   gen_day       2      68
    si16         number_sensor_des       2      70
--------------------------------------------------
--------------------------------------------------
  struct: 'radar_t'
  size: 300
  id: RADD

    type                      name    size  offset
    ----                      ----    ----  ------
    char                     id[4]       4       0
    si32                    nbytes       4       4
    char             radar_name[8]       8       8
    fl32               radar_const       4      16
    fl32                peak_power       4      20
    fl32               noise_power       4      24
    fl32             receiver_gain       4      28
    fl32              antenna_gain       4      32
    fl32               system_gain       4      36
    fl32           horz_beam_width       4      40
    fl32           vert_beam_width       4      44
    si16                radar_type       2      48
    si16                 scan_mode       2      50
    fl32             req_rotat_vel       4      52
    fl32           scan_mode_pram0       4      56
    fl32           scan_mode_pram1       4      60
    si16         num_parameter_des       2      64
    si16             total_num_des       2      66
    si16             data_compress       2      68
    si16            data_reduction       2      70
    fl32            data_red_parm0       4      72
    fl32            data_red_parm1       4      76
    fl32           radar_longitude       4      80
    fl32            radar_latitude       4      84
```

```
    fl32         radar_altitude       4      88
    fl32          eff_unamb_vel       4      92
    fl32        eff_unamb_range       4      96
    si16         num_freq_trans       2     100
    si16         num_ipps_trans       2     102
    fl32                  freq1       4     104
    fl32                  freq2       4     108
    fl32                  freq3       4     112
    fl32                  freq4       4     116
    fl32                  freq5       4     120
    fl32                   prt1       4     124
    fl32                   prt2       4     128
    fl32                   prt3       4     132
    fl32                   prt4       4     136
    fl32                   prt5       4     140
    si32          extension_num       4     144
    char         config_name[8]       8     148
    si32             config_num       4     156
    fl32          aperture_size       4     160
    fl32          field_of_view       4     164
    fl32           aperture_eff       4     168
    fl32           aux_freq[11]      44     172
    fl32            aux_prt[11]      44     216
    fl32            pulse_width       4     260
    fl32     primary_cop_baseln       4     264
    fl32   secondary_cop_baseln       4     268
    fl32       pc_xmtr_bandwidth      4     272
    fl32       pc_waveform_type       4     276
    char          site_name[20]      20     280
--------------------------------------------------
--------------------------------------------------
  struct: 'correction_t'
  size: 72
  id: CFAC

    type                   name    size  offset
    ----                   ----    ----  ------
    char                  id[4]       4       0
    si32                 nbytes       4       4
    fl32           azimuth_corr       4       8
    fl32         elevation_corr       4      12
    fl32       range_delay_corr       4      16
    fl32         longitude_corr       4      20
    fl32          latitude_corr       4      24
    fl32      pressure_alt_corr       4      28
    fl32         radar_alt_corr       4      32
    fl32         ew_gndspd_corr       4      36
    fl32         ns_gndspd_corr       4      40
    fl32          vert_vel_corr       4      44
    fl32           heading_corr       4      48
    fl32              roll_corr       4      52
    fl32             pitch_corr       4      56
    fl32             drift_corr       4      60
    fl32         rot_angle_corr       4      64
    fl32              tilt_corr       4      68
--------------------------------------------------
--------------------------------------------------
  struct: 'parameter_t'
  size: 216
  id: PARM
```

```
    type                     name    size  offset
    ----                     ----    ----  ------
    char                    id[4]       4       0
    si32                   nbytes       4       4
    char        parameter_name[8]       8       8
    char    param_description[40]      40      16
    char           param_units[8]       8      56
    si16          interpulse_time       2      64
    si16             xmitted_freq       2      66
    fl32          recvr_bandwidth       4      68
    si16              pulse_width       2      72
    si16             polarization       2      74
    si16              num_samples       2      76
    si16            binary_format       2      78
    char       threshold_field[8]       8      80
    fl32          threshold_value       4      88
    fl32          parameter_scale       4      92
    fl32           parameter_bias       4      96
    si32                 bad_data       4     100
    si32            extension_num       4     104
    char           config_name[8]       8     108
    si32               config_num       4     116
    si32           offset_to_data       4     120
    fl32           mks_conversion       4     124
    si32               num_qnames       4     128
    char          qdata_names[32]      32     132
    si32             num_criteria       4     164
    char       criteria_names[32]      32     168
    si32             number_cells       4     200
    fl32     meters_to_first_cell       4     204
    fl32     meters_between_cells       4     208
    fl32             eff_unamb_vel       4     212
--------------------------------------------------
--------------------------------------------------
  struct: 'cell_vector_t'
  size: 6012
  id: CELV

    type                     name    size  offset
    ----                     ----    ----  ------
    char                    id[4]       4       0
    si32                   nbytes       4       4
    si32             number_cells       4       8
    fl32        dist_cells[1500]     6000      12
--------------------------------------------------
--------------------------------------------------
  struct: 'cell_spacing_fp_t'
  size: 64
  id: CSFD

    type                     name    size  offset
    ----                     ----    ----  ------
    char                    id[4]       4       0
    si32                   nbytes       4       4
    si32             num_segments       4       8
    fl32            dist_to_first       4      12
    fl32               spacing[8]      32      16
    si16             num_cells[8]      16      48
--------------------------------------------------
--------------------------------------------------
  struct: 'sweepinfo_t'
```

```
  size: 40
  id: SWIB

     type                    name    size  offset
     ----                    ----    ----  ------
     char                   id[4]       4       0
     si32                  nbytes       4       4
     char           radar_name[8]       8       8
     si32               sweep_num       4      16
     si32                num_rays       4      20
     fl32             start_angle       4      24
     fl32              stop_angle       4      28
     fl32             fixed_angle       4      32
     si32             filter_flag       4      36
--------------------------------------------------
--------------------------------------------------
  struct: 'platform_t'
  size: 80
  id: ASIB

     type                    name    size  offset
     ----                    ----    ----  ------
     char                   id[4]       4       0
     si32                  nbytes       4       4
     fl32               longitude       4       8
     fl32                latitude       4      12
     fl32            altitude_msl       4      16
     fl32            altitude_agl       4      20
     fl32             ew_velocity       4      24
     fl32             ns_velocity       4      28
     fl32           vert_velocity       4      32
     fl32                 heading       4      36
     fl32                    roll       4      40
     fl32                   pitch       4      44
     fl32             drift_angle       4      48
     fl32          rotation_angle       4      52
     fl32                    tilt       4      56
     fl32           ew_horiz_wind       4      60
     fl32           ns_horiz_wind       4      64
     fl32               vert_wind       4      68
     fl32          heading_change       4      72
     fl32            pitch_change       4      76
--------------------------------------------------
--------------------------------------------------
  struct: 'ray_t'
  size: 44
  id: RYIB

     type                    name    size  offset
     ----                    ----    ----  ------
     char                   id[4]       4       0
     si32                  nbytes       4       4
     si32               sweep_num       4       8
     si32              julian_day       4      12
     si16                    hour       2      16
     si16                  minute       2      18
     si16                  second       2      20
     si16             millisecond       2      22
     fl32                 azimuth       4      24
     fl32               elevation       4      28
     fl32              peak_power       4      32
```

```
     fl32           true_scan_rate       4      36
     si32               ray_status       4      40
--------------------------------------------------
--------------------------------------------------
  struct: 'paramdata_t'
  size: 16
  id: RDAT

     type                     name    size  offset
     ----                     ----    ----  ------
     char                    id[4]       4       0
     si32                   nbytes       4       4
     char            pdata_name[8]       8       8
--------------------------------------------------
--------------------------------------------------
  struct: 'qparamdata_t'
  size: 56
  id: QDAT

     type                     name    size  offset
     ----                     ----    ----  ------
     char                    id[4]       4       0
     si32                   nbytes       4       4
     char            pdata_name[8]       8       8
     si32            extension_num       4      16
     si32               config_num       4      20
     si16            first_cell[4]       8      24
     si16             num_cells[4]       8      32
     fl32        criteria_value[4]      16      40
--------------------------------------------------
--------------------------------------------------
  struct: 'extra_stuff_t'
  size: 24
  id: XTSF

     type                     name    size  offset
     ----                     ----    ----  ------
     char                    id[4]       4       0
     si32                   nbytes       4       4
     si32                      one       4       8
     si32            source_format       4      12
     si32     offset_to_first_item       4      16
     si32          transition_flag       4      20
--------------------------------------------------
--------------------------------------------------
  struct: 'null_block_t'
  size: 8
  id: NULL

     type                     name    size  offset
     ----                     ----    ----  ------
     char                    id[4]       4       0
     si32                   nbytes       4       4
--------------------------------------------------
--------------------------------------------------
  struct: 'rot_angle_table_t'
  size: 28
  id: RKTB

     type                     name    size  offset
     ----                     ----    ----  ------
```

```
    char                  id[4]       4       0
    si32                 nbytes       4       4
    fl32              angle2ndx       4       8
    si32           ndx_que_size       4      12
    si32       first_key_offset       4      16
    si32     angle_table_offset       4      20
    si32               num_rays       4      24
--------------------------------------------------
--------------------------------------------------
  struct: 'rot_table_entry_t'
  size: 12

    type                   name    size  offset
    ----                   ----    ----  ------
    fl32         rotation_angle       4       0
    si32                 offset       4       4
    si32                   size       4       8
--------------------------------------------------
--------------------------------------------------
  struct: 'radar_test_status_t'
  size: 52
  id: FRAD

    type                   name    size  offset
    ----                   ----    ----  ------
    char                  id[4]       4       0
    si32                 nbytes       4       4
    si32        data_sys_status       4       8
    char          radar_name[8]       8      12
    fl32       test_pulse_level       4      20
    fl32        test_pulse_dist       4      24
    fl32       test_pulse_width       4      28
    fl32        test_pulse_freq       4      32
    si16       test_pulse_atten       2      36
    si16        test_pulse_fnum       2      38
    fl32            noise_power       4      40
    si32              ray_count       4      44
    si16         first_rec_gate       2      48
    si16          last_rec_gate       2      50
--------------------------------------------------
--------------------------------------------------
  struct: 'field_radar_t'
  size: 264
  id: FRIB

    type                   name    size  offset
    ----                   ----    ----  ------
    char                  id[4]       4       0
    si32                 nbytes       4       4
    si32            data_sys_id       4       8
    fl32               loss_out       4      12
    fl32                loss_in       4      16
    fl32            loss_rjoint       4      20
    fl32              ant_v_dim       4      24
    fl32              ant_h_dim       4      28
    fl32         ant_noise_temp       4      32
    fl32         r_noise_figure       4      36
    fl32          xmit_power[5]      20      40
    fl32            x_band_gain       4      60
    fl32       receiver_gain[5]      20      64
    fl32             if_gain[5]      20      84
```

```
     fl32           conversion_gain        4      104
     fl32           scale_factor[5]       20      108
     fl32           processor_const        4      128
     si32          dly_tube_antenna        4      132
     si32     dly_rndtrip_chip_atod        4      136
     si32      dly_timmod_testpulse        4      140
     si32          dly_modulator_on        4      144
     si32         dly_modulator_off        4      148
     fl32         peak_power_offset        4      152
     fl32         test_pulse_offset        4      156
     fl32             E_plane_angle        4      160
     fl32             H_plane_angle        4      164
     fl32        encoder_antenna_up        4      168
     fl32          pitch_antenna_up        4      172
     si16          indepf_times_flg        2      176
     si16           indep_freq_gate        2      178
     si16          time_series_gate        2      180
     si16           num_base_params        2      182
     char             file_name[80]       80      184
--------------------------------------------------
--------------------------------------------------
  struct: 'lidar_t'
  size: 148
  id: LIDR

     type                      name     size  offset
     ----                      ----     ----  ------
     char                     id[4]        4       0
     si32                    nbytes        4       4
     char            lidar_name[8]         8       8
     fl32               lidar_const        4      16
     fl32              pulse_energy        4      20
     fl32                peak_power        4      24
     fl32               pulse_width        4      28
     fl32             aperture_size        4      32
     fl32             field_of_view        4      36
     fl32              aperture_eff        4      40
     fl32           beam_divergence        4      44
     si16                lidar_type        2      48
     si16                 scan_mode        2      50
     fl32             req_rotat_vel        4      52
     fl32           scan_mode_pram0        4      56
     fl32           scan_mode_pram1        4      60
     si16         num_parameter_des        2      64
     si16             total_num_des        2      66
     si16             data_compress        2      68
     si16            data_reduction        2      70
     fl32            data_red_parm0        4      72
     fl32            data_red_parm1        4      76
     fl32           lidar_longitude        4      80
     fl32            lidar_latitude        4      84
     fl32            lidar_altitude        4      88
     fl32              eff_unamb_vel       4      92
     fl32            eff_unamb_range       4      96
     si32           num_wvlen_trans        4     100
     fl32                       prf        4     104
     fl32           wavelength[10]        40     108
--------------------------------------------------
--------------------------------------------------
  struct: 'field_lidar_t'
  size: 748
```

```
  id: FLIB

     type                     name    size  offset
     ----                     ----    ----  ------
     char                    id[4]       4       0
     si32                   nbytes       4       4
     si32              data_sys_id       4       8
     fl32    transmit_beam_div[10]      40      12
     fl32           xmit_power[10]      40      52
     fl32         receiver_fov[10]      40      92
     si32        receiver_type[10]      40     132
     fl32        r_noise_floor[10]      40     172
     fl32     receiver_spec_bw[10]      40     212
     fl32     receiver_elec_bw[10]      40     252
     fl32          calibration[10]      40     292
     si32              range_delay       4     332
     fl32     peak_power_multi[10]      40     336
     fl32        encoder_mirror_up       4     376
     fl32          pitch_mirror_up       4     380
     si32      max_digitizer_count       4     384
     fl32       max_digitizer_volt       4     388
     fl32           digitizer_rate       4     392
     si32        total_num_samples       4     396
     si32         samples_per_cell       4     400
     si32            cells_per_ray       4     404
     fl32                 pmt_temp       4     408
     fl32                 pmt_gain       4     412
     fl32                 apd_temp       4     416
     fl32                 apd_gain       4     420
     si32                 transect       4     424
     char    derived_names[10][12]     120     428
     char     derived_units[10][8]      80     548
     char       temp_names[10][12]     120     628
--------------------------------------------------
--------------------------------------------------
  struct: 'insitu_descript_t'
  size: 4108
  id: SITU

     type                     name    size  offset
     ----                     ----    ----  ------
     char                    id[4]       4       0
     si32                   nbytes       4       4
     si32            number_params       4       8
  params:
     char       params[0].name[8]        8      12
     char       params[0].units[8]       8      20
     char       params[1].name[8]        8      28
     char       params[1].units[8]       8      36
     char       params[2].name[8]        8      44
     char       params[2].units[8]       8      52
     ....
     ....
     ....
     char     params[253].name[8]        8    4060
     char     params[253].units[8]       8    4068
     char     params[254].name[8]        8    4076
     char     params[254].units[8]       8    4084
     char     params[255].name[8]        8    4092
     char     params[255].units[8]       8    4100
--------------------------------------------------
```

```
---------------------------------------------------
  struct: 'insitu_data_t'
  size: 16
  id: ISIT

     type                      name    size  offset
     ----                      ----    ----  ------
     char                     id[4]       4       0
     si32                    nbytes       4       4
     si16                julian_day       2       8
     si16                     hours       2      10
     si16                   minutes       2      12
     si16                   seconds       2      14
---------------------------------------------------
---------------------------------------------------
  struct: 'indep_freq_t'
  size: 8
  id: INDF

     type                      name    size  offset
     ----                      ----    ----  ------
     char                     id[4]       4       0
     si32                    nbytes       4       4
---------------------------------------------------
---------------------------------------------------
  struct: 'minirims_data_t'
  size: 4112
  id: MINI

     type                      name    size  offset
     ----                      ----    ----  ------
     char                     id[4]       4       0
     si32                    nbytes       4       4
     si16                   command       2       8
     si16                    status       2      10
     fl32               temperature       4      12
     fl32          x_axis_gyro[128]     512      16
     fl32          y_axis_gyro[128]     512     528
     fl32          z_axis_gyro[128]     512    1040
     fl32         xr_axis_gyro[128]     512    1552
     fl32           x_axis_vel[128]     512    2064
     fl32           y_axis_vel[128]     512    2576
     fl32           z_axis_vel[128]     512    3088
     fl32           x_axis_pos[128]     512    3600
---------------------------------------------------
---------------------------------------------------
  struct: 'nav_descript_t'
  size: 16
  id: NDDS

     type                      name    size  offset
     ----                      ----    ----  ------
     char                     id[4]       4       0
     si32                    nbytes       4       4
     si16                  ins_flag       2       8
     si16                  gps_flag       2      10
     si16             minirims_flag       2      12
     si16               kalman_flag       2      14
---------------------------------------------------
---------------------------------------------------
  struct: 'time_series_t'
```

```
  size: 8
  id: TIME

     type                       name    size  offset
     ----                       ----    ----  ------
     char                      id[4]       4       0
     si32                     nbytes       4       4
--------------------------------------------------
--------------------------------------------------
  struct: 'waveform_t'
  size: 364
  id: WAVE

     type                       name    size  offset
     ----                       ----    ----  ------
     char                      id[4]       4       0
     si32                     nbytes       4       4
     char          ps_file_name[16]       16       8
     si16              num_chips[6]       12      24
     char           blank_chip[256]      256      36
     fl32                 repeat_seq       4     292
     si16            repeat_seq_dwel       2     296
     si16                  total_pcp       2     298
     si16             chip_offset[6]      12     300
     si16              chip_width[6]      12     312
     fl32                     ur_pcp       4     324
     fl32                     uv_pcp       4     328
     si16              num_gates[6]       12     332
     si16             gate_dist1[2]        4     344
     si16             gate_dist2[2]        4     348
     si16             gate_dist3[2]        4     352
     si16             gate_dist4[2]        4     356
     si16             gate_dist5[2]        4     360
--------------------------------------------------
==================================================
```