

VAISALA

PROGRAMMER'S MANUAL

IRIS

PUBLISHED BY

Vaisala Oyj	Phone (int.):	+358 9 8949 1
P.O. Box 26	Fax:	+358 9 8949 2227
FI-00421 Helsinki		
Finland		

Visit our Internet pages at www.vaisala.com

© Vaisala 2013

No part of this manual may be reproduced in any form or by any means, electronic or mechanical (including photocopying), nor may its contents be communicated to a third party without prior written permission of the copyright holder.

The contents are subject to change without prior notice.

Please observe that this manual does not create any legally binding obligations for Vaisala towards the customer or end user. All legally binding commitments and agreements are included exclusively in the applicable supply contract or Conditions of Sale.

Table of Contents

CHAPTER 1	
GENERAL INFORMATION	9
1.1 About This Manual	9
1.2 Version Information	10
1.3 Related Manuals	10
1.4 Documentation Conventions	10
CHAPTER 2	
INTRODUCTION TO IRIS PROGRAMMING	13
2.1 Directory Organization	13
2.2 Setting up the Development Environment	15
CHAPTER 3	
CUSTOM INPUT AND OUTPUT	19
3.1 Input Pipes	19
3.2 User Product Insert	20
3.3 Building the Example Programs	20
3.4 Running the upi_rcv Program	21
3.5 IRIS Product and Data Types	23
3.6 Customizing the Legend	23
3.7 The NORDRAD Area Definition File	25
CHAPTER 4	
DATA FORMATS	29
4.1 Scalar Definitions	29
4.2 Structure Definitions	30
4.2.1 beam_psi_struct Structure	30
4.2.2 cappi_psi_struct Structure	31
4.2.3 catch_psi_struct Structure	31
4.2.4 catch_results Structure	31
4.2.5 color_scale_def Structure	32
4.2.6 cross_psi_struct Structure	34
4.2.7 dsp_data_mask Structure	34
4.2.8 extended_header_v0 Structure	34
4.2.9 extended_header_v1 Structure	35
4.2.10 extended_header_v2 Structure	35
4.2.11 fcast_psi_struct Structure	36
4.2.12 gage_psi_struct Structure	36
4.2.13 gage_results Structure	36
4.2.14 ingest_configuration Structure	37
4.2.15 ingest_data_header Structure	38

4.2.16 ingest_header Structure	39
4.2.17 max_psi_struct Structure	39
4.2.18 ndop_input Structure	40
4.2.19 ndop_psi_struct Structure	40
4.2.20 ndop_results Structure	40
4.2.21 one_protected_region Structure	41
4.2.22 ppi_psi_struct Structure	41
4.2.23 product_configuration Structure	41
4.2.24 product_end Structure	44
4.2.25 product_hdr Structure	46
4.2.26 product_header_extensions	48
4.2.27 product_specific_info Structure	48
4.2.28 protect_setup Structure	49
4.2.29 rain_psi_struct Structure	49
4.2.30 raw_prod_bhdr Structure	49
4.2.31 raw_psi_struct Structure	50
4.2.32 ray_header Structure	50
4.2.33 rhi_psi_struct Structure	51
4.2.34 rti_psi_struct Structure	51
4.2.35 shear_psi_struct Structure	51
4.2.36 sline_psi_struct Structure	52
4.2.37 sline_results Structure	52
4.2.38 sri_psi_struct Structure	54
4.2.39 status_antenna_info Structure	55
4.2.40 status_device_info Structure	55
4.2.41 status_message_info Structure	55
4.2.42 status_misc_info Structure	56
4.2.43 status_one_device Structure	57
4.2.44 status_one_process Structure	57
4.2.45 status_process_info Structure	57
4.2.46 status_results Structure	58
4.2.47 structure_header Structure	58
4.2.48 tape_header_record Structure	58
4.2.49 task_calib_info Structure	59
4.2.50 task_configuration Structure	60
4.2.51 task_dsp_info Structure	60
4.2.52 task_dsp_mode_batch Structure	61
4.2.53 task_end_info Structure	61
4.2.54 task_file_scan_info Structure	62
4.2.55 task_manual_scan_info Structure	62
4.2.56 task_misc_info Structure	62
4.2.57 task_ppi_scan_info Structure	63
4.2.58 task_range_info Structure	63
4.2.59 task_rhi_scan_info Structure	64
4.2.60 task_scan_info Structure	64
4.2.61 task_sched_info Structure	64
4.2.62 tdwr_psi_struct Structure	65
4.2.63 tdwr_results Structure	65
4.2.64 text_results Structure	66
4.2.65 top_psi_struct Structure	66
4.2.66 track_psi_struct Structure	66
4.2.67 track_results Structure	67
4.2.68 vad_psi_struct Structure	67
4.2.69 vad_results Structure	68
4.2.70 vvp_psi_struct Structure	68

4.2.71 vvp_results Structure	68
4.2.72 warn_psi_struct Structure	69
4.2.73 warning_results Structure	70
4.2.74 wind_psi_struct Structure	70
4.2.75 wind_results Structure	71
4.2.76 ymds_time Structure	71
4.3 Data Types	71
4.3.1 Extended_Header Format (DB_XHDR)	72
4.3.2 2-byte Axis of Dilliation Format (DB_AXDIL2)	72
4.3.3 1-byte Reflectivity Format (DB_DBT& DB_DBZ)	72
4.3.4 2-byte Reflectivity Format (DB_DBT2& DB_DBZ2)	73
4.3.5 2-byte Deformation Format (DB_DEFORM2)	73
4.3.6 2-byte Divergence Format (DB_DIVERGE2)	73
4.3.7 2-byte Floating Liquid Format (DB_FLIQUID2)	74
4.3.8 1-byte HydroClass Format (DB_HCLASS)	74
4.3.9 2-byte HydroClass Format (DB_HCLASS2)	77
4.3.10 1-byte Echo Tops Format (DB_HEIGHT)	77
4.3.11 2-byte Horizontal wind direction Format (DB_HDIR2)	77
4.3.12 1-byte KDP Format (DB_KDP)	78
4.3.13 2-byte KDP Format (DB_KDP2)	79
4.3.14 1-byte LDR Format (DB_LDRH & DB_LDRV)	79
4.3.15 2-byte LDR Format (DB_LDRH2 & DB_LDRV2)	80
4.3.16 1-byte Phi Format (DB_PHIH & DB_PHIV)	80
4.3.17 2-byte Phi Format (DB_PHIH2 & DB_PHIV2)	80
4.3.18 1-byte PhiDP Format (DB_PHIDP)	80
4.3.19 2-byte PhiDP Format (DB_PHIDP2)	80
4.3.20 2-byte Rainfall Rate Format (DB_RAINRATE2)	81
4.3.21 1-byte Rho Format (DB_RHOH & DB_RHOV)	82
4.3.22 2-byte Rho Format (DB_RHOH2 & DB_RHOV2)	82
4.3.23 1-byte RhoHV Format (DB_RHOHV)	82
4.3.24 2-byte RhoHV Format (DB_RHOHV2)	82
4.3.25 1-byte Wind Shear Format (DB_SHEAR)	83
4.3.26 1-byte Signal Quality Index Format (DB_SQI)	83
4.3.27 2-byte Signal Quality Index Format (DB_SQI2)	83
4.3.28 2-byte Time Format (DB_TIME2)	84
4.3.29 1-byte Velocity Format (DB_VEL)	84
4.3.30 2-byte Velocity Format (DB_VEL2)	85
4.3.31 1-byte Unfolded Velocity Format (DB_VELC)	85
4.3.32 2-byte Unfolded Velocity Format (DB_VELC2)	85
4.3.33 2-byte VIL Format (DB_VIL2)	86
4.3.34 2-byte Vertical Velocity Format (DB_VVEL2)	86
4.3.35 1-byte Width Format (DB_WIDTH)	86
4.3.36 2-byte Width Format (DB_WIDTH2)	87
4.3.37 1-byte ZDR Format (DB_ZDR)	87
4.3.38 2-byte ZDR Format (DB_ZDR2)	88
4.4 Ingest Data File Format	88
4.4.1 Ingest File Names	89
4.5 Product File Format	89
4.5.1 Cartesian Product Format	90
4.5.2 FCAST Product Format	90
4.5.3 NDOP Product Format	90
4.5.4 RAW Product Format	90
4.5.4.1 Data Compression Algorithm	92
4.5.4.2 Raw Product Example	93

4.5.5 SLINE Product Format	93
4.5.6 TDWR Product Format	94
4.5.7 TRACK Product Format	94
4.5.8 VAD Product Format	94
4.5.9 VVP Product Format	94
4.5.10 WARN Product Format	95
4.5.11 WIND Product Format	95
4.5.12 Product File Names	95
4.6 Tape Format	96
4.7 TIFF Output Format	96
4.8 Constants	97
CHAPTER 5	
UTILITIES	99
5.1 Productx	99
5.1.1 Invoking Productx	99
5.1.2 Productx Examples	100
5.2 Rays	104
5.2.1 Invoking Rays	104
5.2.2 Rays Examples	104
5.2.2.1 Headers only Example	104
5.2.2.2 Velocity Example	106
5.2.2.3 Extended Header Example.	107
APPENDIX A	
RADAR CONTROL PROTOCOL	109
APPENDIX B	
LINK TRANSMISSION FORMATS	121
B.1 AWS (Austrian Weather Service) Format	121
B.2 HKO (HongKong Observatory) Format	122
APPENDIX C	
UF FORMAT	125
C.1 Introduction	125
C.2 Single UF Ray Structure	125
C.3 uf_mandatory_header2 Structure	126
C.4 uf_optional_header Structure	127
C.5 uf_data_header2 Structure	127
C.6 uf_field_header2 Structure	128
C.7 uf_fsi2 Structure	128
APPENDIX D	
RTD FORMATS	131
D.1 Introduction	131
D.2 Rtd_v1_xmt	132
D.3 Rtd_v2_xmt	133
D.4 Rtd_nids3_xmt	133

List of Figures

Figure 1	IRIS Directory Tree	13
Figure 2	Product File Format	90

List of Tables

Table 1	Manual Revisions	10
Table 2	Contents of Library Subdirectories	14
Table 3	Contents of IRIS Subdirectories	14
Table 4	Contents of Utilities Subdirectories	15
Table 5	Contents of Configuration Subdirectories	15
Table 6	Example of NORDRAD_AREAS.DAT	25
Table 7	IRIS Data Types	29
Table 8	IRIS Timezone Recording	47
Table 9	Ingest Data File Format	88
Table 10	Compression Code Meanings	92
Table 11	Raw Product Example	93
Table 12	TIFF Fields Used by IRIS	96
Table 13	Data Type Constants — /include/sigtypes.h	97
Table 14	Status Packet RCV01 Format (RCP to Host)	111
Table 15	Control Packet XMT01 Format (Host to RCP)	111
Table 16	Status Packet RCV02 / RCV04 Format (RCP to Host)	112
Table 17	Control Packet XMT02 / XMT04 Format (Host to RCP)	113
Table 18	Status Packet RCV03 Format (RCP to Host)	114
Table 19	Status Packet RCV05 Format (RCP to Host)	116
Table 20	Control Packet XMT05 Format (Host to RCP)	116
Table 21	Time Packet (RCP to Host)	117
Table 22	Generic BITE Status Packet (Both ways)	117
Table 23	BITE Command Packet (Both ways)	118
Table 24	Auxiliary Control BITE Packets (Both ways)	118
Table 25	Q-BITE Status Packet (Both ways)	118
Table 26	Simple Q-BITE Example	119
Table 27	Q-BITE Interrogate Packet (Both ways)	119
Table 28	BITE Individual Command Packet (Host to RCP)	119
Table 29	Chat-Mode Packet (Both ways)	120
Table 30	HKO Picture types	123

CHAPTER 1

GENERAL INFORMATION

The IRIS programming interface gives you access to the radar, antenna, and signal processing hardware, as well as the IRIS menus and utility software. This means that you can expand the capabilities of IRIS for any special purpose your site may require. For example:

- Create special-purpose products.
- Write applications that accept data generated by IRIS.
- Write new interfaces to IRIS.

1.1 About This Manual

[Chapter 2, Introduction to IRIS Programming](#) - describes where to find the IRIS source code libraries within the directory structure, and how to compile and link IRIS code.

[Chapter 3, Custom Input and Output](#) - shows how to define new IRIS products types, which can be accessed through the IRIS menus. This chapter contains an example program to help you get started.

[Chapter 4, Data Formats](#) - describes the data formats generated by IRIS products and used by the IRIS menus and utilities. Any application that you write must be able to understand the format of the data generated by IRIS.

[Appendix A, Radar Control Protocol](#) - describe the RCP transmit and receive formats.

[Appendix B, Link Transmission Formats](#) - describes the data formats for link outputs to AWS and HKO.

[Appendix C, UF Format](#) - dDescribes IRIS's implementation of the UF (Universal Format).

1.2 Version Information

Table 1 Manual Revisions

Manual Code	Description
M211318EN-C	This manual. Third version. November 2013
M211318EN-B	Previous manual. Second version. March 2013
M211318EN-A	Previous version.

1.3 Related Manuals

Manual Code	Manual Name
M211315EN	IRIS and RDA Installation Manual
M211316EN	IRIS and RDA Utilities Manual
M211317EN	IRIS Radar User's Manual
M211319EN	IRIS Product and Display Manual
M211320EN	RCP8 User's Manual
M211321EN	RVP8 User's Manual
M211452EN	IRIS and RDA Dual Polarization User's Manual

You can download the latest versions of the manuals from Vaisala product website, <http://www.vaisala.com> They can be read online using by Adobe® Reader®, which is installed with IRIS.

Vaisala encourages you to send your comments and/or corrections to:

Vaisala Inc.
7A Lyberty Way
Westford, MA 01886
email helpdesk@vaisala.com

1.4 Documentation Conventions

Throughout the manual, important safety considerations are highlighted as follows:

WARNING	Warning alerts you to a serious hazard. If you do not read and follow instructions very carefully at this point, there is a risk of injury or even death.
----------------	---

CAUTION

Caution warns you of a potential hazard. If you do not read and follow instructions carefully at this point, the product could be damaged or important data could be lost.

NOTE

Note highlights important information on using the product.

The following conventions are used throughout this manual:

\$	The dollar sign is used to show the operating
#	system prompt, the pound sign indicates the OS prompt for
	the root user.

CHAPTER 2

INTRODUCTION TO IRIS PROGRAMMING

2.1 Directory Organization

IRIS source files, object modules, and library routines are placed in separate branches of the IRIS directory tree, as shown in [Figure 1 on page 13](#).

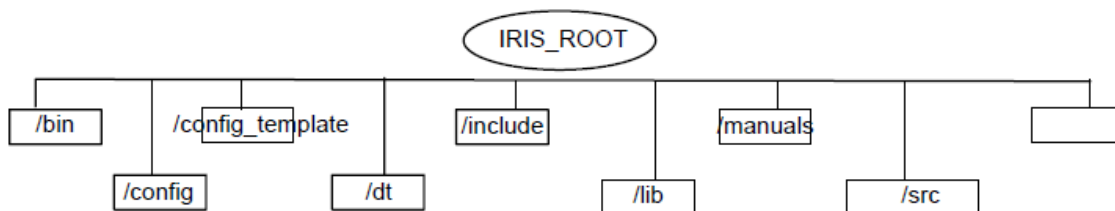


Figure 1 IRIS Directory Tree

IRIS_ROOT refers to the root directory for the IRIS system. It is often `/usr/sigmet`; IRIS_ROOT is defined as an environment variable. When referring to a file within the IRIS directory tree, this manual assumes IRIS_ROOT to be the root directory. All pathnames are relative to IRIS_ROOT, unless otherwise noted. All source code discussed here is placed in the `/src` directory.

[Table 2 on page 14](#) gives a short description of the library routine subdirectories, [Table 3 on page 14](#) lists the IRIS subdirectories, [Table 4 on page 15](#) describes the utility subdirectories, and [Table 5 on page 15](#) lists the Configuration subdirectories. All system wide header files are in the `${IRIS_ROOT}/include` directory, and all library `.a` files are in the

`${IRIS_ROOT}/lib` directory. In the tables below, the public source code is marked with a "*".

Table 2 Contents of Library Subdirectories

Directory	Description
antenna*	Antenna driver library routines that control the radar and antenna.
audio*	Speech output library.
bxutils	GUI utilities library used by BxPro.
config*	Configuration subroutines that save and load setup files.
dpolatten	Dual polarization attenuation library.
dsp*	DSP driver routines that control the actions of the signal processor.
exthdr*	Extended header custom plugin library.
fileformats*	Library to read and write graphical formats like GIF.
hdf5*	Library to read and write HDF5 files (3rd party).
himath*	Math libraries for fft, polynomial, map projections, etc.
lib	Files with the .a extension are stored here.
link*	Link output file.
maps*	Read/write overlays, catchments, terrain maps.
misc*	Miscellaneous libraries.
nordrad2*	Nordrad supplied API.
outlib*	Formats displayed images from IRIS products.
rtq*	Real-time display transmitter library.
share	Source code for shared memory routines that perform various operations, such as handling the ingest file directory, handling errors and other events, etc.
tvsubs*	High level Graphics display library routines.
user*	General-purpose routines.
uxsig	X-Windows support routines.
vtv*	Virtual TV routines that read and write IRIS graphical products to buffers in memory.
xsig*	X functions.

Table 3 Contents of IRIS Subdirectories

Directory	Description
archive	Source code for the archive process.
examiners*	Code for rays and productx .
ingest	Source code for the ingest process.
ingfio	Source code for the ingfio process.
input	The input pipe handler process.
network	Source code for the network process.
nordrad	Nordrad product receiver.
output*	Source code for the output process.
product	Source code for the product process.
reingest	Source code for the reingest process.

Table 3 Contents of IRIS Subdirectories (Continued)

Directory	Description
server	Source code for the IRIS server.
siris	Source code for the <code>siris</code> program.
sserver	Socket server used for IRIS/Web.
watchdog	Source code for the watchdog process.
window*	Source code for the X-window handler.
xuif	IRIS main menu program.

Table 4 Contents of Utilities Subdirectories

Directory	Description
adids*	ADIDS output pipe.
antenna*	AntExport, AntLog, antx, qant utilities.
archive2*	Archive2 input and output pipes.
asterix*	Asterix output pipe.
bufr*	BUFR input and output pipes.
custom*	Misc custom interface programs.
ewis*	EWIS output pipe.
examples*	Simple source code, including UPI routines.
grib1*	GRIB1 output pipe.
hdf5*	HDF5 input and output pipes.
mcidas*	McIDAS pipes.
nexrad*	Nexrad calibration utilities.
nordrad2*	NORDRAD2 relay programs.
pipes_in*	More input pipes.
pipes_out*	More output pipes.
rainbow*	Rainbow input pipe.
tdwr*	TDWR interface programs.
ualf*	UALF Lightning input relay programs.
uf*	UF input and output pipes.

Table 5 Contents of Configuration Subdirectories

Directory	Description
init	Fonts and initializing scripts.
menu	Menu configuration files.
overlay	Overlay definition files.

2.2 Setting up the Development Environment

You may need to compile your own programs using IRIS headers, and which may link to IRIS libraries on occasion, for example:

- If you are writing your own code to read or write IRIS products using the IRIS data structures.
- If you are writing your own input or output pipe program to convert the data to another format.
- If you are writing your own code to interface to the RCP and RVP using our public APIs. This might be an offline diagnostic program, for example.
- When defining new product types using UPI.

First, be sure to install the source files, object modules, libraries, and emacs on your system. This is done by pushing these 4 buttons in the **install** program. See the instructions for your operating system in the *IRIS Installation Manual*.

Setting up a developer source tree:

Before you start writing new code, you need to set up directories to work in. Do this with the following commands in the operator account:

```
$ cd

$ mkdir sigmet_src

$ cd sigmet_src

$ mkdir libs iris utils
```

All the main IRIS source code reside in the release directory tree starting at `${IRIS_ROOT}/src`. It is all in 3 major subdirectories: `libs`, `iris`, and `utils`. When making a change, do not change the source in the release tree. Instead copy the files for the directory you are changing to the same subdirectory of `~/sigmet`. For example, to make a change to the RainbowToIris input pipe, copy the files from `${IRIS_ROOT}/src/utils/rainbow` to `~/sigmet_src/utils/rainbow`. You can then make your changes and compile there. If you are modifying one of IRIS's standard pipes, SIGMET strongly recommends that you rename it to a custom name. That way it will not get replaced by the next upgrade.

To compile and link an IRIS library or application:

Each directory installed on your system contains a makefile to build the application or library contained there. To run the makefile, switch to the directory containing the source code and type "make depend" to build the dependencies, then "make". To install the binaries, run "sudo make install". If you are compiling a whole tree, it is better to type "make -sw".

To compile IRIS/Web code:

You need to install the tomcat5 and Java rpms. Do this by following the instructions in the chapter on *Installing IRIS/Web Server* in the *Software Installation Manual*. Then you need to set the alternatives state of java and javac to use the newly installed rpm. Do this with the "alternatives_java" script file supplied by Vaisala.

To compile IRIS HDF5 code:

You need to install the hdf5-devel rpm. Mount the IRIS/RDA release cdrom and look in the directory RHEL5/extras/RPMS. If you do not have a release cdrom, you can get them from our ftp site: ftp://ftp.sigmet.com/outgoing/os_patches/RHEL5/RPMS. Install with the following command:

```
# rpm -Uhv hdf5-1.8.0-1.el5.rf.i386.rpm
```

To compile IRIS BUFR pipes:

You need to install the OPERA BUFR development rpm. Mount the IRIS/RDA release cdrom and look in the directory RHEL5/extras/RPMS. If you do not have a release cdrom, you can get them from our ftp site: ftp://ftp.sigmet.com/outgoing/os_patches/RHEL5/RPMS. Install with the following commands:

```
# rpm -Uhv bufr-3.0-2.i386.rpm
```

```
# rpm -Uhv bufr-devel-3.0-2.i386.rpm
```


CHAPTER 3

CUSTOM INPUT AND OUTPUT

3.1 Input Pipes

There are two different command line formats used by the IRIS input process to launch an input pipe. In **setup**, these are called "Pathnames" and "Pipe". Sigmet recommends that you code all new pipes using the more flexible Pathnames scheme. In this scheme, the pipe program is invoked with the following command line:

```
$ <pipe-path> -if:<in-file> -ip:<in-path> -op:<out-path> -  
device:<number>
```

In this case, your pipe must explicitly open the input and output files. The purpose of the "-if" argument (which is somewhat redundant with "-ip") is to allow the pipe program to make nice diagnostic printouts, or to parse the filename for information, without having to search the pathname for the final "/".

The only information passed back to the calling program is the 8-bit exit status. In normal cases it should return EXIT_SUCCESS (0). If it returns EXIT_FAILURE (1), then IRIS will signal a message saying "Error running pipe, check log". It is important for the pipe program to write diagnostic information to a log file. Sigmet suggests writing to the `${IRIS_LOG}` directory.

There are 2 other possible exit returns: EXIT_RERUN (2), and EXIT_NOTHING (3). EXIT_RERUN indicates that the pipe has both produced a normal output, and wants to produce a second output file. In this case, the pipe needs to store information about the second output on disk. Sigmet suggests that you write this in the `${IRIS_TEMP}` directory. IRIS will process the first output, then invoke the pipe again with the following command line:

```
$ <pipe-path> -rerun -op:<out-path> -device:<number>
```

Note that the input path is gone, and the option "-rerun" is supplied. When the pipe is thus called, it can retrieve the stored information and write the second output to the specified pathname.

EXIT_NOTHING indicates that the pipe ran without error, but there was no output produced. IRIS will ignore and delete the output file. This is useful for a pipe which needs to read several input files before producing one output file. An example of such a pipe is the Rainbow input pipe RainbowToIris, found in the directory `${IRIS_ROOT}/utils/rainbow`.

3.2 User Product Insert

Customers are generate their own IRIS products, and to process IRIS products within their own programs. The file formats are fully documented in this manual. The difficulty is often in passing the product between the user program and IRIS.

One way to do this is by using IRIS's input and output mechanisms with no handshaking. This scheme uses polling. For example the user places a product file in a know directory, IRIS checks every few seconds. When it finds a file it reads it in. Similarly, IRIS's output can place a file in a know directory, and the user's program can poll for it. When using a polling scheme, it is always desirable to first copy the file there with a "." prefix. Once the file is fully there, rename it to a name without the dot. This prevents the recipient from seeing a partial file.

User Product Insert (UPI) is an alternative mechanism which sends a message to notify the recipient of a product. This removes all time lag and cpu usage due to polling. The user routine communicates with IRIS in exactly the same manner in which different IRIS host systems communicate with each other. The user routine can execute on the same computer as the IRIS host, or it can be on a different CPU because the communication is via network socket routines.

SIGMET supplies a sample routine to get you started. The routine is written in C++, and uses sockets to communicate with a host IRIS system.

3.3 Building the Example Programs

There are three example programs shipped with the iris release media to demonstrate the UPI process. These are all in the `utils/examples` directory, and are called `upi_rcv`, `upi_xmt`, and `change_product`. The `upi_rcv` program simply receives a file from IRIS, prints out the file name, then deletes it. It can easily be customized to your requirements. The

change_product program will modify an existing product file to add in a circular target. The upi_xmt program sends files back to IRIS.

To compile these programs, first configure your IRIS compiling environment as discussed in [Chapter 2, Introduction to IRIS Programming, on page 13](#). Then go to the directory containing the source code: `${IRIS_ROOT}/utils/examples`. Compile it using the resident Makefile, as follows:

```
$ make upi_rcv
```

You may want to rename it, copy it elsewhere, and make a script file to compile it. You can take the compile and link commands from the makefile.

3.4 Running the upi_rcv Program

These instructions for running the sample program assume that you are running upi_rcv on a computer node called "target", and that IRIS is running on a computer called "host". Running on a single computer is somewhat simpler.

First create the directory on target into which IRIS can copy products. A typical example might be `/usr/iris_data/UPI`. The actual directory name is not important.

Copy the upi_rcv to the target computer (if it is different from where you compiled it) so it can run there. You will need to set the mode again after copying.

On the IRIS host system, create a new network output device called "ToUPI" for sending products to the upi_rcv program. This is all done from the **setup** utility - an example is shown below:

Output Device #7

Help

Device type	Network <input type="checkbox"/>
Unit number	<input type="text" value="2"/>
Menu alias	<input type="text" value="ToUPI"/>
File format	IRIS (Def) <input type="checkbox"/>
Filename format	Long <input type="checkbox"/>
Data format	<input type="checkbox"/> Normal
Notification scheme	TCPIP <input type="checkbox"/>
Target directory	<input type="text" value="target:/usr/iris_data/UPI/"/>
Copy scheme	RCP <input type="checkbox"/>
Notification port number	<input type="text" value="i 30726"/>
Recipient node name	<input type="text" value="target"/>

Next, on the target computer, run the upi program:

```
$ upi_rcv -d /usr/iris_data/UPI
```

It should print something like the following:

```
Initializing UPI receiver
```

```
Connection established
```

```
Received file: '/usr/iris_data/UPI/host950710171113.PPI'
```

```
Connection closed
```

Finally, using the Product Output menu, send a horizontal reflectivity product to "ToUPI".

3.5 IRIS Product and Data Types

User generated products must fit into one of the IRIS product and data types. A good way to help understand these is to look at the display of a typical square product (such as a PPI) on a medium sized window. The product type is used by IRIS to decide in general the interpretation of the image, and the data type is used to figure out how to convert between data number and colors.

The data type in a product file controls the color legend (the box containing an array of colored rectangles with text next to them) The data type is converted to text to label the legend, and is used internally to help control how data values in Cartesian products are converted to colors. Some data types do not display a color legend. Examples of these are products like WARN, SLINE, TRACK, and VVP displayed as graphs.

The product type in a product file controls some of the text legend as well as the geometry. Based on the product type, IRIS divides products into several geometrical categories which determine what kind of overlays can be drawn on the image:

- **Horizontal:** These products (such as CAPPI) can have political overlays, product overlays, range rings, and latitude/longitude grids drawn on them. The user cursor works ok. There are minor variations within this category, such as: constant elevation (e.g. PPI), constant height (e.g. CAPPI), and no height implied (e.g. Echo tops).
- **Vertical:** These products (such as Cross Section) can have a range/height overlay grid. The user cursor works ok.
- **None:** These products (such as VVP) have not geometrical interpretation. There are no overlays drawn, and the user cursor does not return information.
- **No Display:** These products (such as RAW) cannot be displayed.

The definitions for the data type parameters are found in the include/dsp_lib.h file. They are also discussed in more detail in [4.3 Data Types on page 71](#).

The definitions for the product types are found in the include/product.h file.

3.6 Customizing the Legend

The legend on user defined products can be customized to some extent. The text displayed in the 5-line legend box on the window legend can be overridden as well as the 6 lines above the time on the old-medium sized

legend, and the 4 lines above the time on the old-small sized legend. This is controlled by the OUTPUT_LEGEND.DAT file. An example of this file is below. As you can see, the format is more complicated. The column headed by "R" stands for resolution, the number of characters on the line. Low resolution windows can fit only 13 characters, while medium and high resolution can fit 17 characters. Therefore, in general you must add two definitions for each string you want to control. However, IRIS defaults to the 11-character format for bigger displays if no 17-character entry is defined. The column headed by "L" stands for line: "1" means line 7 on the text legend, and "2" means line 8 on the text legend. The first entry in the table is the product type, which must be USER, VUSER, or OTHER.

The general pattern of operation is as follows: When IRIS draws the legend and the product type is either PROD_USER, PROD_USERV, or PROD_OTHER, it searches this table. If it finds a match of the product type, product name, resolutions, and line number, it is considered a match. If it matches everything except the product name, then the first such match in the table is taken as a default for all product names, and is considered a match. If it would otherwise match, except that the desired resolution is larger than the table, it is considered a match. If no match is found, it displays blank.

If the flag is set to "SCALE", a 32-bit word is grabbed from the first byte of the product header "product_specific_info" field offset by the offset in the file. This number is converted to floating point, and multiplied by the scale factor in the file. The C library routine `sprintf` is called with the format statement and the resulting number as arguments. The resulting string is displayed on the legend.

If the flag is set to "REF", a pointer to the product header "product_specific_info" field offset by the offset in the file is passed to `sprintf`. This is useful if you want to display a character string from the header. The resulting string is displayed on the legend.

If the flag is neither of the above, a 32-bit word is grabbed from the first byte of the product header "product_specific_info" field offset by the offset in the file without any conversion. The C library routine `sprintf` is called with the format statement and the 32-bit number as arguments. The resulting string is displayed on the legend

```
# File: output_legend.dat

# Format:

# Valid flags are: SCALE, REF, <anything else>
```

R=resolution, L=line, O=offset

#ptype	productname	R	L	O	scale	flag	format
USER	DEFAULT	11	1	4	0.00001	SCALE	"Hght:%3.1f"
USER	Z_010_120	11	1	4	0.00001	SCALE	"Hght:%3.1f"
USER	DEFAULT	11	2	8	1	-	"Flag:%d"
USER	Z_010_120	11	2	8	1	-	" "
VUSER	DEFAULT	11	1	4	1	-	" "
USER	DEFAULT	17	1	4	0.00001	SCALE	"Height:%3.1f"
USER	Z_010_120	17	1	4	0.00001	SCALE	"Height:%3.1f"
USER	DEFAULT	17	2	8	1	-	"Flag:%d"
USER	Z_010_120	17	2	8	1	-	" "
VUSER	DEFAULT	17	1	4	1	-	" "

3.7 The NORDRAD Area Definition File

You want to make a configuration file to control converting between NORDRAD areas and IRIS sites. This file is called NORDRAD_AREAS.DAT, and it is in the IRIS_CONFIG directory. An example of this file is shown in [Table 6 on page 25](#).

Table 6 Example of NORDRAD_AREAS.DAT

```
# Example file for NORDRAD_AREAS.DAT file
# This controls the scale used when converting to the RRSRT_Z product: rainfall_accumulation_span
0.01 100.0
# This controls the height (in km) used when converting a NORDRAD
# CAPPI to IRIS if the Height attribute is missing from the header.
default_cappi_height 1.0
# Network access information used for the NORDRAD API.
# Either 1 or two devices can be defined. The output device
# number is used to select which to use. For input, the first
# device is used.
api_device_number 1 2
api_access_control "operator xxxxxx" "operator xxxxxx"
api_host_name "haze" "haze"
api_receiver_id "iris_trans" "iris_trans"
# Below there are two mapping tables used to convert
# between IRIS sites, and NORDRAD areas. One map for each direction
# Mapping of NORDRAD areas to IRIS sitenames
# areaname sitename wavelength in 1/100 of cm
ntoi_map FIABO1 "MASKU" 500
ntoi_map FIABO2 "MASKU with s" 500
```

Table 6 Example of NORDRAD_AREAS.DAT (Continued)

```

ntoi_map FIABO3 MASKU 500
ntoi_map FIABO4 MASKU 500
ntoi_map FIABO6 MASKU 500
ntoi_map FIHEL Helsinki 500
ntoi_map FIROVMTA COMPOSITE 500
ntoi_map FIROV1 ROVANIEMI 500
ntoi_map FIROV2 ROVANIEMI 500
ntoi_map FIAB01 "UND Huntsville" 500
ntoi_map FIAB03 "SIGMET, haze" 500
ntoi_map FIHOT1 "SIGMET, HOT" 500
ntoi_map FIDRY "SIGMET, dry" 500
ntoi_map FIHOT2 "SIGMET, HOT" 500
ntoi_map FISTO3 STOCKHOLM 500
ntoi_map SVNKP2 NORRK #with diaeresis# PING 500
ntoi_map SVOVIK1 OVIK 500
ntoi_map SVOVIK2 OVIK 500
# Mapping of IRIS sitenames to NORDRAD areas.
# There are two kinds of entries, iton_prod and iton_map
# Map based on sitename, product type, and product name.
# If there is an exact match here, take it. This overrides
# the iton_map lines. The first match will control. Size
# and resolution are not used for matching.
# areaname sitename ptype pname h-size v-size resolution in meters
iton_prod FIABO1 "MASKU" CAPPI DEFAULT 150000 150000 1000
# Map based on sitename and resolution. For multiple areas from
# the same sitename, look for the closest in resolution. If
# there is a resolution tie, then look for the closest in size.
# areaname sitename h-size v-size resolution in meters
iton_map FIABO1 "MASKU" 150000 150000 1000
iton_map FIABO2 "MASKU with s" 300000 300000 1000
iton_map FIROV1 ROVANIEMI 300000 300000 1000
iton_map FIROV2 ROVANIEMI 600000 600000 1000
iton_map FIAB01 "UND Huntsville" 100000 100000 1000
iton_map FIAB03 "SIGMET, haze" 200000 200000 1000
iton_map FIHOT1 "SIGMET, HOT" 200000 200000 1000
iton_map FIDRY "SIGMET, dry" 200000 200000 2000
iton_map FIHOT2 "SIGMET, HOT" 200000 200000 2000
iton_map FIVAN1 VANTA 480000 480000 1000
iton_map FIVAN10 VANTA 480000 480000 1000
iton_map SVKKR1 KARLSKRONA 480000 480000 1000
iton_map SVKKR2 KARLSKRONA 240000 240000 1000
iton_map SVOVIK1 OVIK 480000 480000 1000
iton_map SVOVIK2 OVIK 240000 240000 1000

```

The NORDRAD_AREAS.DAT file consists mainly of a list of all the NORDRAD areas with an associated IRIS site name. The file is used for conversions both ways between IRIS and NORDRAD products. When a NORDRAD product is converted to an IRIS product, the NORDRAD area name is looked up in the ntoI_map list, and the associated IRIS site name is used. There is only one entry for each area in the table. The radar wavelength in the file is put into the IRIS header. The wavelength and PRF are needed to display velocities and widths in IRIS.

When an IRIS product is converted to a NORDRAD product, the iton_map list is searched for site names that match. IRIS looks for the closest match between the product size and the size in the file. IRIS converts the product to the exact size (in meters) of the NORDRAD area by either clipping the edges or padding with blank. If you want the product to be properly handled by NORDRAD, you must make the area size match. It is slightly less important to make the resolution in pixels match. Formally, the NORDRAD area defines only a rectangle on the earth, not the resolution of the data. However to do compositing, the resolutions have to be converted, and it may be easier to start at the expected resolution. If there is an exact match to a iton_prod map, then that overrides the iton_map table.

The file can optionally include a line starting with "rainfall_accumulation_span". This is used to control the scaling of rainfall accumulation data when it is converted to NORDRAD format. The two numbers are the minimum and maximum rainfall in mm. Rainfall is converted to NORDRAD using the formula:

$$\text{FILE} = (\log_{10}(\text{Physical}) - \log \text{MIN}) / ((\log \text{MAX} - \log \text{MIN}) / 254)$$

The minimum must be between 0.001 and 1000.0, the maximum between 0.01 and 10000.0, and the span must be at least a factor of 10. If there is an error, IRIS will signal when it reads the file. If there is no such line, IRIS defaults to a span of 0.1 to 1000.

CHAPTER 4

DATA FORMATS

This chapter describes the archived data formats used by IRIS. These formats include data storage on disk files, and output tape formats. Within this chapter, all sizes and addresses are given in bytes. Also included are tables of antenna, signal processor, and general-purpose constants.

4.1 Scalar Definitions

The file `${IRIS_ROOT}/include/sigtypes.h` defines a few scalar data typedefs that can be used across platforms. These types are listed in [Table 7 on page 29](#). They are used by the IRIS library routines for defining return values and routine arguments.

Table 7 **IRIS Data Types**

Type	Mapped-to	Description
SINT1	int8_t	Signed 8-bit integer
UINT1	uint8_t	Unsigned 8-bit integer
SINT2	int16_t	Signed 16-bit integer
UINT2	uint16_t	Unsigned 16-bit integer
SINT4	int32_t	Signed 32-bit integer
UINT4	uint32_t	Unsigned 32-bit integer
FLT4	float	Floating-point number
FLT8	double	Double-precision floating-point number
BIN2	uint16_t	16-bit binary angle
BIN4	uint32_t	32-bit binary angle
MESSAGE	uint32_t	Encoded error value

A binary angle is an efficient way to store an angle into an integer. In this format the MSB has weight 180 degrees, the next bit means 90 degrees, the

next 45 degrees, etc. So you can convert an N-bit binary angle to degrees with:

$$\text{Degrees} = 360 * (\text{Binary Angle}) / 2^N$$

To see C language examples of how to do these conversions, please look in `src/libs/user/angle.c`.

4.2 Structure Definitions

Structures are described in alphabetical order. Tables give the following information about each structure:

- Source — The name of the include file containing the C structure definition. The source files can be found in the `${IRIS_ROOT}/include` directories.
- Byte — The byte offset of the structure element.
- Size — The data type of the element from the above table. For structures and spare space the size in bytes is displayed here. Arrays are designated by a data type followed by a number in square brackets.
- Contents — A brief description of the structure element. Structure elements which are themselves structures are identified with angle brackets such as "<ymds_time>". Note, however, that "<spare>" indicates reserved spare space, not a special structure.

4.2.1 beam_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	UINT4	Minimum range in cm
4	UINT4	Maximum range in cm
8	BIN4	Left azimuth
12	BIN4	Right azimuth
16	BIN4	Lower elevation
20	BIN4	Upper elevation
24	BIN4	Azimuth smoothing
28	BIN4	Elevation smoothing
32	BIN4	Az of sun at start
36	BIN4	El of sun at start
40	BIN4	Az of sun at end
44	BIN4	El of sun at end

4.2.2 cappi_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	4	Shear flags, same as <code>shear_psi_struct</code>
4	SINT4	Height of CAPPI (cm. above reference)
8	UINT2	Flags: bit0=make pseudo CAPPI bit1=Velocity is horizontal winds
10	BIN2	Azimuth smoothing for shear
12	char[12]	VVP name to use for shear correction
24	UINT4	Max age for vvp shear correction in seconds

4.2.3 catch_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	UINT4	Flags in low 16-bits; From 8.09: RAIN1 flags in upper 16-bits Bit 0: Enable warning if catchments exceed threshold
4	UINT4	Hours of accumulation
8	SINT4	Threshold offset in 1/1000 or mm
12	SINT4	Threshold faction in 1/1000
16	char[12]	Name of RAIN1 product to use
28	char[16]	Name of catchment file to use
44	UINT4	From 8.09: Seconds of accumulation in low 16-bits
48	UINT4	From 8.09: RAIN1 min Z
52	UINT4	From 8.09: RAIN1 span in seconds
56	UINT4	From 8.09: Average Gage correction factor in low 16-bits

4.2.4 catch_results Structure

Source: `product.h`

Byte	Size	Contents
0	char[16]	Name of catchment area, null terminated

16	UINT4	Number of catchment area
20	BIN4	Latitude of label
24	BIN4	Longitude of label
28	SINT4	Area of catchment in 1/100 of square km
32	SINT4	Number of pixels in the catchment area
36	SINT4	Number of pixels scanned in the catchment area
40	UINT4	Flags Bit 0: Warning enabled for this catchment Bit 1: Warning triggered for this catchment
44	UINT2	Rainfall accumulation in catchment, DB_FLIQUID2 format
46	UINT2	Rainfall accumulation warning threshold, DB_FLIQUID2
48	52	<spare>
100	UINT2[96]	Rainfall accumulation for each input, DB_FLIQUID2 format

4.2.5 color_scale_def Structure

NOTE

The changes in color_scale_def are non-critical (unused), for consistency only.

Source: headers.h

Byte	Size	Contents
0	UINT4	iflags: Bit 8=COLOR_SCALE_VARIABLE Bit 10=COLOR_SCALE_TOP_SAT Bit 11=COLOR_SCALE_BOT_SAT Bit 12=ENUMERATED DATA
4	SINT4	istart: Starting level
8	SINT4	istep: Level step
12	SINT2	icolent: Number of colors in scale
14	UINT2	iset_and_scale: Color set number in low byte, color scale number in high byte.
16	UINT2[16]	ilevel_seams: Variable level starting values

This structure appears at the end of the product_configuration structure. It holds information configured into the product about how to quantize the data into color levels for display. We may remove this structure in the near future because IRIS can override all this at display time. This is only used

if the user selects "Use Default Scale" in the Color Scale Tool. The bit 11 is used to identify enumerated data type such as DB_HCLASS (DB_HCLASS2), requiring further information about the discrete data values.

There are three ways of specifying the color scale:

- a fixed scale has uniformly spaced numbers fully specified by a start and step value
- a variable scale is specified by a set of 16 individually controlled data seams
- the data are enumerated i.e. each value has an discrete meaning, defined by the string data in enumeration_def structures, which can represent enumeration data as multiples of 14 classes

The **iflags** bits COLOR_SCALE_VARIABLE, over ruled by ENUMERATED_DATA (relates to products using DB_HCLASS) indicate which mode is in effect.

In the first two cases, the number of colors can be anywhere from 2 to 16, and optionally zoomed or split to effectively give up to 32 colors. True color representation is possible in IRIS, too. The bits 10 and 11 in **iflags** specify what to do with data off the end of the scale. There are two choices, either saturate at the end (used the last color), or threshold (do not display). For example a typical treatment of reflectivity would be to saturate the high end of the scale (so as not to lose the strong features), and threshold at the low end (to inhibit display of very weak speckles).

The number of main colors and labels in the legend is set by **icolent**. To select which subset of colors is used for the scale, fill in the low byte of **iset_and_scale**. The upper byte is not used in the product header.

For fixed spacing, set the **istart** and **istep** to the start and step values desired. These values are integers which are 10 times the physical units as discussed in the **color_setup** utility chapter of the *IRIS Utilities Manual*. The labels of **istart**, **istart+istep**, **istart+2*istep**, etc. will show up on the right side of the IRIS legend. There are two special cases: Width assumes that the starting value is zero, and velocity assumes that the middle of the scale is at zero. If the velocity step is set to zero it means automatically scale to fit the Nyquist velocity.

For variable spacing (bit 8 ON), the numbers in the array **ilevel_seams** control the starting values for each of the colors. The first seam is the bottom of the first color. The top of the 16th color is computed by adding the step between the 15th and 16th seam to the 16th seam. These seams are data values, using the 16-bit versions of the data. For shear and height data, for which there is no 16-bit version, the 8-bit version is used.

Enumerated data (bit 12 ON) can be treated as other data types, that is either with fixed spacing or variable spacing.

4.2.6 cross_psi_struct Structure

Source: headers.h

Byte	Size	Contents
0	BIN2	Azimuth angle of line from left to right
2	10	<spare>
12	SINT4	East coordinate of center (in cm. relative to radar)
16	SINT4	North coordinate of center (in cm. relative to radar)
20	SINT4	[15]User miscellaneous

4.2.7 dsp_data_mask Structure

Source: sigtypes.h

Byte	Size	Contents
0	UINT4	Mask word 0
4	UINT4	Extended header type
8	UINT4	Mask word 1 Contains bits set for all data recorded.
12	UINT4	Mask word 2 See parameter DB_* in Table 13 on page 97 for
16	UINT4	Mask word 3 bit specification.
20	UINT4	Mask word 4

4.2.8 extended_header_v0 Structure

Source: ingest.h

Byte	Size	Contents
0	SINT4	Time in milliseconds from the sweep starting time
4	SINT2	Calibration Signal level
6	14	<spare>

4.2.9 extended_header_v1 Structure

Source: ingest.h

Byte	Size	Contents
0	SINT4	Time in milliseconds from the sweep starting time
4	SINT2	Calibration Signal level
6	BIN2	Azimuth (binary angle)
8	BIN2	Elevation (binary angle)
10	BIN2	Train Order (binary angle)
12	BIN2	Elevation Order (binary angle)
14	BIN2	Pitch (binary angle)
16	BIN2	Roll (binary angle)
18	BIN2	Heading (binary angle)
20	BIN2	Azimuth Rate (binary angle/second)
22	BIN2	Elevation Rate (binary angle/second)
24	BIN2	Pitch Rate (binary angle/second)
26	BIN2	Roll Rate (binary angle/second)
28	BIN4	Latitude (binary angle)
32	BIN4	Longitude (binary angle)
36	BIN2	Heading Rate (binary angle/second)
38	SINT2	Altitude (meters above MSL)
40	SINT2	Velocity East (cm/second)
42	SINT2	Velocity North (cm/second)
44	SINT4	Time since last update (milliseconds)
48	SINT2	Velocity Up (cm/second)
50	UINT2	Navigation System OK flag
52	SINT2	Radial velocity correction (same units as velocities)

4.2.10 extended_header_v2 Structure

Source: none

Byte	Size	Contents
0	SINT4	Time in milliseconds from the sweep starting time
4	SINT2	Calibration Signal level
6	2	<spare>
8	SINT4	Number of bytes in the header
12	?	Remainder customer specified

4.2.11 fcast_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	UINT4	Correlation threshold, 0—100
4	SINT4	Data threshold
8	SINT4	Mean speed (cm/hour), zero if none
12	BIN4	Direction of mean speed
16	UINT4	Maximum time between inputs (seconds)
20	SINT4	Maximum allowable velocity (mm/sec)
24	UINT4	Flags
28	SINT4	Desired output resolution (cm)
32	UINT4	Type of input product
36	char[12]	Name of input product (space padded)

4.2.12 gage_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	UINT4	Time span of the rain gage data in seconds.
4	UINT4	Flag bits: 0 = File has distrometers 1 = File does not have gages

4.2.13 gage_results Structure

Source: `product.h`

Byte	Size	Contents
0	char[16]	Name of raingage, null terminated
16	BIN32_T	Latitude of raingage
20	BIN32_T	Longitude of raingage
24	uint16_t	Average rainrate in DB_RAINRATE2 format for the time span
26	uint16_t	Correction factor last calculated in DB_CDBZ2 format
28	uint8_t	Data Quality, range 0-10

29	uint8_t	Flag bits: 0 = Skip this gage for correction calculation 1 = Gage has Z/R numbers 2 = Gage does not have rainrate 3 = Radar rainrate filled in
30	uint16_t	Z/R constant in 1/10
32	uint16_t	Z/R exponent in 1/1000
34	uint16_t	Average radar rainrate in DB_RAINRATE2 format for the span
36	4	<spare>

4.2.14 ingest_configuration Structure

Source: `ingest.h`

Byte	Size	Contents
0	char[80]	Name of file on disk
80	SINT2	Number of associated data files extant
82	SINT2	Number of sweeps completed so far
84	SINT4	Total size of all files in bytes
88	12	<ymds_time> Time that volume scan was started, TZ spec in bytes 166 & 224
100	112	<spare>
112	SINT2	Number of bytes in the ray headers
114	SINT2	Number of bytes in extended ray headers (includes normal ray header)
116	SINT2	Number of task configuration table
118	SINT2	Playback version number
120	4	<spare>
124	char[8]	IRIS version, null terminated
132	char[16]	Hardware name of site
148	SINT2	Time zone of local standard time, minutes west of GMT
150	char[16]	Name of site, from setup utility
166	SINT2	Time zone of recorded standard time, minutes west of GMT
168	BIN4	Latitude of radar (binary angle: 20000000 hex is 45° North)
172	BIN4	Longitude of radar (binary angle: 20000000 hex is 45° East)
176	SINT2	Height of ground at site (meters above sea level)
178	SINT2	Height of radar above ground (meters)

180	UINT2	Resolution specified in number of rays in a 360° sweep
182	UINT2	Index of first ray from above set of rays
184	UINT2	Number of rays in a sweep
186	SINT2	Number of bytes in each gparam
188	SINT4	Altitude of radar (cm above sea level)
192	SINT4[3]	Velocity of radar platform (cm/sec)(east, north, up)
204	SINT4[3]	Antenna offset from INU (cm) (starboard, bow, up)
216	UINT4	Fault status at the time the task was started, bits: 0:Normal BITE 1:Critical BITE 2:Normal RCP 3:Critical RCP 4:Critical system 5:Product gen. 6:Output 7:Normal system
220	SINT2	Height of melting layer (meters above sea level) MSB is complemented, zero=Unknown
222	2	<spare>
224	char[8]	Local timezone string, null terminated
232	UINT4	Flags, Bit 0=First ray not centered on zero degrees
226	char[16]	Configuration name in the dpolapp.conf file, null terminated
252	228	<spare>

4.2.15 ingest_data_header Structure

Source: ingest.h

Byte	Size	Contents
0	12	<structure_header> The size stored in here is the total size of the file. That is 76 + 4*Irtotl + Iwritn*raysize.
12	12	<ymds_time> (4.2.76 ymds_time Structure on page 71) Date and time that sweep was started TZ specified in ingest_configuration
24	SINT2	Sweep number, origin 1
26	SINT2	Resolution specified in number of rays in a 360° sweep
28	SINT2	Index of first ray from above set of rays
30	SINT2	"Irtotl" Number of rays (and pointers) expected in the file
32	SINT2	"Iwritn" Number of rays actually in the file
34	BIN2	Fixed angle for this sweep (binary angle)

36	SINT2	Number of bits per bin for this data type
38	UINT2	Data type in the file: 0=extended header 1=Total power (dBZ) 2=Reflectivity (dBZ) 3=Velocity 4=Width 5=ZDR
40	36	<spare>

4.2.16 ingest_header Structure

A file containing the ingest_header structure is written for each volume scan. This file is updated each time a sweep has finished. This allows product generators to get data from the ingest files as soon as a sweep is completed. Note on RVP6 and 7 the GPARM is read just after the processor is configured. It will contain configuration fault bits, but not the results of a noise sample taken before that task. On the RVP8 starting in release 8.09 it is read with the first valid ray of the task.

Source: ingest.h

```

<structure_header> 12 Bytes
<ingest_configuration> 480 Bytes
<task_configuration> 2612 Bytes
Spare 732 Bytes
GPARM from DSP 128 Bytes
Reserved 920 Bytes

```

4.2.17 max_psi_struct Structure

Source: headers.h

Byte	Size	Contents
0	4	<spare>
4	SINT4	Bottom of interval in cm
8	SINT4	Top of interval in cm
12	SINT4	Number of pixels in side panels
16	SINT2	Horizontal smoother in side panels
18	SINT2	Vertical smoother in side panels

4.2.18 ndop_input Structure

Source: `headers.h`

Byte	Size	Contents
0	char[12]	Task name
12	char[3]	Site code
15	UINT1	Flags

4.2.19 ndop_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	3*16	<ndop_input>
48	SINT4	Time window
52	SINT4	Cappi height (cm above reference)
56	SINT4	Output resolution (cm)
60	BIN4	Minimum permitted crossing angle
64	UINT4	Flags: Bit 0=Make diagnostic products
68	char[4]	Output site code, added in 8.09.6, use first input if zeroed
72	8	<spare>

4.2.20 ndop_results Structure

Used for both the NDOP and FCAST products. The change rate is always zero in NDOP products. The SQI is unused in FCAST products. The SQI is a function of the variance of the calculated velocity normalized such that random vectors at half the Nyquist velocity would produce an SQI of zero.

$$SQI = 1 - \left[\sqrt{\frac{VarianceEast + VarianceNorth}{2}} \right] / Vnorm$$

Source: `product.h`

Byte	Size	Contents
0	UINT2	Velocity East in cm/second (DB_VEL2 format)
2	UINT2	Velocity North in cm/second
4	UINT2	Change rate in db/min (DB_CDBZ2 format)
6	UINT1	Signal Quality Index * 256
7	5	<spare>

4.2.21 one_protected_region Structure

Source: `setup.h`

Byte	Size	Contents
0	SINT4	East center from radar in cm
4	SINT4	North center from radar in cm
8	SINT4	East-West size in cm
12	SINT4	North-South size in cm
16	UINT2	Orientation angle in binary angle
18	2	<spare>
20	char[12]	Name of the region, all spaces means unused

4.2.22 ppi_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	BIN2	Elevation angle

4.2.23 product_configuration Structure

Source: `product.h`

Byte	Size	Contents
0	12	<Structure_Header>
12	UINT2	Product type code: 1:PPI 2:RHI 3:CAPPI 4:CROSS 5:TOPS 6:TRACK 7:RAIN1 8:RAINN 9:VVP 10:VIL 11:SHEAR 12:WARN 13:CATCH 14:RTI 15:RAW 16:MAX 17:USER 18:USERV 19:OTHER 20:STATUS 21:SLINE 22:WIND 23:BEAM 24:TEXT 25:FCAST 26:NDOP 27:IMAGE 28:COMP 29:TDWR 30:GAGE 31:DWELL 32:SRI 33:BASE 34:HMAX
14	UINT2	Scheduling code: 0:hold; 1:next; 2:all
16	SINT4	Number of seconds to skip between runs
20	12	<ynds_time> (4.2.76 ynds_time Structure on page 71) Time product was generated (UTC)

32	12	<ymds_time> (4.2.76 ymds_time Structure on page 71) Time of input ingest sweep (TZ flex)
44	12	<ymds_time> (4.2.76 ymds_time Structure on page 71) Time of input ingest file (TZ flexible)
56	6	<spare>
62	char[12]	Name of the product configuration file
74	char[12]	Name of the task used to generate the data
86	UINT2	Flag word: (Bits 0,2,3,4,8,9,10 used internally) Bit1: TDWR style messages Bit5: Keep this file Bit6: This is a clutter map Bit7: Speak warning messages Bit11: This product has been composited Bit12: This product has been dwelled Bit13: Z/R source0, 0:Type-in; 1:Setup; 2:Disdrometer Bit14: Z/R source1
88	SINT4	X scale in cm/pixel
92	SINT4	Y scale in cm/pixel
96	SINT4	Z scale in cm/pixel
100	SINT4	X direction size of data array
104	SINT4	Y direction size of data array
108	SINT4	Z direction size of data array
112	SINT4	X location of radar in data array (signed 1/1000 of pixels)
116	SINT4	Y location of radar in data array (signed 1/1000 of pixels)
120	SINT4	Z location of radar in data array (signed 1/1000 of pixels)
124	SINT4	Maximum range in cm (used only in version 2.0, raw products)
128	2	<spare>
130	UINT2	Data type generated (See 4.8 Constants on page 97 for values)
132	char[12]	Name of projection used
144	UINT2	Data type used as input (See 4.8 Constants on page 97 for values)
146	UINT1	Projection type: 0=Centered Azimuthal, 1=Mercator
147	1	<spare>
148	SINT2	Radial smoother in 1/100 of km
150	SINT2	Number of times this product configuration has run

152	SINT4	Z/R relationship constant in 1/1000
156	SINT4	Z/R relationship exponent in 1/1000
160	SINT2	X-direction smoother in 1/100 of km
162	SINT2	Y-direction smoother in 1/100 of km
164	80	<product_specific_info>
244	char[16]	List of minor task suffixes, null terminated
260	12	<spare>
272	48	<color_scale_def>(4.2.5 color_scale_def Structure on page 32) Color scale definition May be removed in the future.

4.2.24 product_end Structure

Source: `product.h`

Byte	Size	Contents
0	char[16]	Site name — where product was made (space padded)
16	char[8]	IRIS version where product was made (null terminated)
24	char[8]	IRIS version where ingest data came from
32	12	<ynds_time> Time of oldest input ingest file (only RAIN1 and RAINN, TZ flexible)
44	28	<spare>
72	SINT2	Number of minutes local standard time is west of GMT
74	char[16]	Hardware name where ingest data came from (space padded)
90	char[16]	Site name where ingest data came from (space padded)
106	SINT2	Number of minutes recorded standard time is west of GMT
108	BIN4	Latitude of center (binary angle) ¹
112	BIN4	Longitude of center (binary angle) ²
116	SINT2	Signed ground height in meters relative to sea level
118	SINT2	Height of radar above the ground in meters
120	SINT4	PRF in hertz
124	SINT4	Pulse width in 1/100 of microseconds
128	UINT2	Type of signal processor used
130	UINT2	Trigger rate scheme
132	SINT2	Number of samples used
134	char[12]	Clutter filter file name
146	UINT2	Number of linear based filter for the first bin
148	SINT4	Wavelength in 1/100 of centimeters
152	SINT4	Truncation height (cm above the radar)
156	SINT4	Range of the first bin in cm
160	SINT4	Range of the last bin in cm
164	SINT4	Number of output bins
168	UINT2	Flag word Bit0: Disdrometer failed, we used setup for Z/R source instead

170	SINT2	Number of ingest or product files used to make this product (only on RAIN1 and RAINN)
172	UINT2	Type of polarization used
174	SINT2	I0 cal value, horizontal pol, in 1/100 dBm
176	SINT2	Noise at calibration, horizontal pol, in 1/100 dBm
178	SINT2	Radar constant, horizontal pol, in 1/100 dB
180	UINT2	Receiver bandwidth in kHz
182	SINT2	Current noise level, horizontal pol, in 1/100 dBm
184	SINT2	Current noise level, vertical pol, in 1/100 dBm
186	SINT2	LDR offset, in 1/100 dB
188	SINT2	ZDR offset, in 1/100 dB
190	uint16_t	TCF Cal flags, see struct task_calib_info (added in 8.12.3)
192	uint16_t	TCF Cal flags2, see struct task_calib_info (added in 8.12.3)
194	uint8_t	if enumerated data (DB_HCLASS) the identifier of classifier in the first bit segment else <spare>
195	uint8_t	if enumerated data (DB_HCLASS) the identifier of classifier in the 2nd bit segment else <spare>
196	uint8_t	if enumerated data (DB_HCLASS) the identifier of classifier in the 3rd bit segment else <spare>
197	uint8_t	if enumerated data (DB_HCLASS2) the identifier of the 4th classifier else <spare>
198	uint8_t	if enumerated data (DB_HCLASS2) the identifier of the 5th classifier else <spare>
199	uint8_t	if enumerated data (DB_HCLASS2) the identifier of the 6th classifier else <spare>
200	12	<spare>
212	BIN4	More projection info these 4 words: Standard parallel #1
216	BIN4	Standard parallel #2
220	UINT4	Equatorial radius of the earth, cm (zero = 6371km sphere)
224	UINT4	1/Flattening in 1/1000000 (zero = sphere)
228	UINT4	Fault status of task, see ingest_configuration 4.2.14 ingest_configuration Structure on page 37 for details
232	UINT4	Mask of input sites used in a composite
236	UINT2	Number of log based filter for the first bin
238	UINT2	Nonzero if cluttermap applied to the ingest data
240	BIN4	Latitude of projection reference ³
244	BIN4	Longitude of projection reference ⁴

248	SINT2	Product sequence number
250	32	<spare> (was used before 8.11.4)
282	SINT2	Melting level in meters, msb complemented (0=unknown)
284	SINT2	Height of radar above reference height in meters
286	SINT2	Number of elements in product results array
288	UINT1	Mean wind speed
289	BIN1	Mean wind direction (unknown if speed and direction 0)
290	2	<spare>
292	char[8]	TZ Name of recorded data
300	UINT4	Offset to extended product header from file (added in 8.13.0)
304	4	<spare>

1. Note on Latitude and longitudes: Interpretation varies with product type. They are as documented for CAPPI, FCAST, MAX, NDOP, PPI, RAIN1, RAINN, SHEAR, SLINE, TOPS, TRACK, VIL, USER and WARN. For all other products, the Center location is the radar location, and the reference location is zero.
2. Note on Latitude and longitudes: Interpretation varies with product type. They are as documented for CAPPI, FCAST, MAX, NDOP, PPI, RAIN1, RAINN, SHEAR, SLINE, TOPS, TRACK, VIL, USER and WARN. For all other products, the Center location is the radar location, and the reference location is zero.
3. Note on Latitude and longitudes: Interpretation varies with product type. They are as documented for CAPPI, FCAST, MAX, NDOP, PPI, RAIN1, RAINN, SHEAR, SLINE, TOPS, TRACK, VIL, USER and WARN. For all other products, the Center location is the radar location, and the reference location is zero.
4. Note on Latitude and longitudes: Interpretation varies with product type. They are as documented for CAPPI, FCAST, MAX, NDOP, PPI, RAIN1, RAINN, SHEAR, SLINE, TOPS, TRACK, VIL, USER and WARN. For all other products, the Center location is the radar location, and the reference location is zero.

4.2.25 product_hdr Structure

Source: `product.h`

<pre><structure_header> (4.2.47 structure_header Structure on page 58) 12 Bytes <product_configuration> (4.2.23 product_configuration Structure on page 41) 320 Bytes <product_end> (4.2.24 product_end Structure on page 44) 308 Bytes Data Size Varies</pre>
--

Starting with version 7.31, IRIS allows data to be recorded in UTC on computers with a timezone set to local time. IRIS also records timezone information about the local computer to support optional displaying of different times at output time. [Table 8 on page 47](#) documents the way this information is stored in the product and ingest headers. For backwards compatibility, **iMinutesWest** (byte 106 in product_end) is the difference between the recorded standard time and UTC. All other field are new in version 7.31. **iLocalWest** (byte 72) is the difference between local standard time and UTC. **sTZName** is the text name for the local timezone at the radar. There are also 3 new bits recorded in the upper bits of the milliseconds field of the milliseconds field of all ymds_time structures: **UTC** means that the time is recorded in UTC. **DST** means that the time is in summer time. **LDST** means that the local time on the computer is in summer time. A 60 minute adjustment to the times may be required to correctly deal with summer times. Because the **sLocalTZName** field is new, all old data when displayed using local time will display with a timezone offset, such as "-05" for US EST. Note that ascope recorded data before 7.31 did not include the timezone, so all old data will display with "+00" for a timezone, independent of what was recorded.

Previous to 7.31, IRIS acted as if the new setup question was set to record data using local time. Some systems had their timezone set to UTC to make sure data was recorded in UTC. This can now be changed. It is never recommended to record data using summer time.

Table 8 IRIS Timezone Recording

System	Record UTC in setup	Record Local in setup
Computer timezone UTC	iMinutesWest=0 iLocalWest=0 sLocalTZName="UTC" UTC Flag=1 DST Flag=0 LDST Flag=0 Default Display is "UTC"	iMinutesWest=0 iLocalWest=0 sLocalTZName="UTC" UTC Flag=0 DST Flag=0 LDST Flag=0 Default Display is "UTC"
Computer timezone EST	iMinutesWest=0 iLocalWest=300 sLocalTZName="EST" UTC Flag=1 DST Flag=0 LDST Flag=0 Default Display="UTC"	iMinutesWest=300 iLocalWest=300 sLocalTZName="EST" UTC Flag=0 DST Flag=0 LDST Flag=0 Default Display is "EST"
Computer timezone EDT	iMinutesWest=0 iLocalWest=300 sLocalTZName="EDT" UTC Flag=1 DST Flag=0 LDST Flag=1 Default Display is "UTC"	iMinutesWest=300 DO iLocalWest=300 NOT sLocalTZName="EDT" RECORD UTC Flag=0 SUMMER DST Flag=1 TIME! LDST Flag=1 Default Display is "EDT"

4.2.26 product_header_extensions

If the `hdr.end.iExtendedHeaderOffset` is non-zero, then there is a product header extension, and this number is the byte offset from the beginning of the file to the extension.

This product header extension is in an ASCII format, containing meta-data in name=value pairs. The name value pairs are separated by newline characters. The ASCII meta-data section is terminated by an ASCII zero. This null may be followed by more null, and will finally end with a `0xff`. This allows a parser to find the end by looking for a final `0x00` followed by a `0xff`, and allows the size to be padded to make the next section aligned, if desired. Following this product header extension, there may be a binary data section. This binary data section will be aligned based on the size of the data elements, which is specified in the meta-data. This is a general design intended to handle many needs for extending headers, such as listing sites in a composite, or listing algorithm attributes used to generate a data moment. In some cases, we might have multiple product header extension sections.

For example, in the the echo thickness product, we store two binary data arrays, the first one is the thickness, and the second one is the average reflectivity over that interval. The following name value pairs (the values shown are example, but the names shown will be used: `"data_type=Z"`, `"data_units=dBZ"`, `"x_size=120"`, `"y_size=120"`, `"data_storage=uint8_t"`, `"data_bits=8"`, `"data_source=DB_DBZ"`, `"data_size=14400"`). We then add a binary blob after the extension containing the average reflectivity.

4.2.27 product_specific_info Structure

Source: `headers.h`

Byte	Size	Contents
0	10	<cappi_psi_struct> (if CAPPI)
0	44	<catch_psi_struct> (if CATCH)
0	80	<cross_psi_struct> (if XSECT or USERV)
0	48	<fcast_psi_struct> (if FCAST)
0	20	<maximum_psi_struct> (if MAX)
0	2	<ppi_psi_struct> (if PPI)
0	24	<rain_psi_struct> (if RAIN1 or RAINN)
0	20	<raw_psi_struct> (if RAW)
0	2	<rhi_psi_struct> (if RHI)
0	28	<shear_psi_struct> (if SHEAR)
0	60	<sline_psi_struct> (if SLINE)

0	14	<tdwr_psi_struct> (if TDWR)
0	6	<top_psi_struct> (if TOPS, BASE, HMAX, or THICK)
0	52	<track_psi_struct> (if TRACK)
0	16	<vad_psi_struct> (if VAD)
0	4	<vil_psi_struct> (if VIL)
0	28	<vvp_psi_struct> (if VVP)
0	80	<warn_psi_struct> (if WARN)
0	40	<wind_psi_struct> (if WIND)
0	80	<spare> (if USER, OTHER, TEXT)

4.2.28 protect_setup Structure

Source: `setup.h`

Byte	Size	Contents
0	1024	<one_protected_region>[32] Protected region definitions

4.2.29 rain_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	UINT4	Minimum Z to accumulate (in 2-byte Reflectivity Format)
4	UINT2	Average gage correction factor
6	UINT2	Seconds of accumulation (Starting at 8.09)
8	UINT2	Flag word: Bit0: Apply clutter map Bit1: Apply gage correction Bit2: Clutter map was applied Bit3: Gage correction was applied
10	SINT2	Number of hours to accumulate (RAINN only)
12	char[12]	Name of input product to use (space padded)
24	UINT4	From 8.09: Span in seconds of the input files

4.2.30 raw_prod_bhdr Structure

Source: `product.h`

Byte	Size	Contents
------	------	----------

0	SINT2	Record number within the file (origin 0: record 3 contains a 2)
2	SINT2	Sweep number (1 is first sweep)
4	SINT2	Byte offset of first full ray in this record (-1 if none)
6	SINT2	Ray number within sweep for above pointed to ray
8	UINT2	Flags
10	2	<spare>

4.2.31 raw_psi_struct Structure

Source: headers.h

Byte	Size	Contents
0	UINT4	Data type mask word 0
4	SINT4	Range of last bin in cm
8	UINT4	Format conversion flag: 0=Preserve all ingest data 1=Convert 8-bit data to 16-bit data 2=Convert 16-bit data to 8-bit data
12	UINT4	Flag word: Bit 0=Separate product files by sweep Bit 1=Mask data by supplied mask
16	SINT4	Sweep number if separate files, origin 1
20	4	Xhdr type (unused)
24	4	Data type mask 1
28	4	Data type mask 2
32	4	Data type mask 3
36	4	Data type mask 4
40	4	Playback version (low 16-bits)

4.2.32 ray_header Structure

Source: ingest.h

Byte	Size	Contents
0	BIN2	Azimuth at beginning of ray (binary angle) If dual-PRF: bit0=ray's PRF was high
2	BIN2	Elevation at beginning of ray (binary angle) If trigger blanking on: bit0=Trigger was not blanked

4	BIN2	Azimuth at end of ray (binary angle)
6	BIN2	Elevation at end of ray (binary angle)
8	SINT2	Actual number of bins in the ray
10	UINT2	Time in seconds from start of sweep (unsigned)

4.2.33 rhi_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	BIN2	Azimuth angle

4.2.34 rti_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	BIN4	Nominal sweep angle
4	UINT4	Starting time offset from sweep time, ms
8	UINT4	Ending time offset
12	BIN4	Azimuth of the first ray in the file
16	BIN4	Elevation of the first ray in the file

4.2.35 shear_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	BIN4	Azimuthal smoothing angle
4	BIN2	Elevation angle
6	2	<spare>
8	UINT4	Flag word: Bit0=Do radial shear Bit1=Do azimuthal shear Bit2=Do mean wind correction to azimuthal shear using VVP Bit3=Mean wind correction to azimuthal shear done Bit4=Unfolding done in associated VVP product Bit5=Do elevation shear
12	char[12]	Name of VVP product to use (space padded)
24	UINT4	Maximum age of VVP to use (seconds)

4.2.36 sline_psi_struct Structure

Source: headers.h

Byte	Size	Contents
0	SINT4	Area in square meters
4	SINT4	Shear threshold (cm/sec/km)
8	UINT4	Bit flags to choose protected areas
12	SINT4	Maximum forecast time in seconds
16	UINT4	Maximum age between products for motion calculation
20	SINT4	Maximum velocity allowed (mm/sec)
24	UINT4	Flag word: Bit0=Do radial shear Bit1=Do azimuthal shear (both bits mean do combined) Bit2=Do mean wind correction to azimuth shear using VVP Bit3=Mean wind correction to azimuthal shear done Bit4=Unfolding done in associated VVP product Bit8=Use two elevation angles Bit9=Generate diagnostic output Bit10=Max centroid count exceeded
28	BIN4	Azimuthal smoothing angle (0=none)
32	BIN4	Elevation binary angle
36	BIN4	Elevation binary angle
40	char[12]	Name of VVP task
52	UINT4	Maximum age of VVP in seconds
56	SINT4	Curve fit standard deviation threshold in cm.
60	UINT4	Low byte: Min length of sline (unsigned 1/10 km)

4.2.37 sline_results Structure

Source: product.h

Byte	Size	Contents
0	SINT4	East coordinate of center point, cm
4	SINT4	North coordinate of center point, cm
8	BIN4	Rotation angle to X-Y coordinates of curve fit polynomial Span is -90 to +90 degrees.
12	SINT4	X coordinate of left of curve, cm
16	SINT4	X coordinate of right of curve, cm

20	SINT4[6]	6 polynomial coefficients
44	SINT4	Standard deviation of fit
48	SINT4	Propagation speed (mm/second)
52	BIN4	Propagation direction (binary angle)
56	SINT4	Reference side wind speed (mm/second)
60	BIN4	Reference side wind direction (binary angle)
64	SINT4	Other side wind speed (mm/second)
68	BIN4	Other side wind direction (binary angle)
72	SINT4[32]	ETA in seconds for each protected area (0 if in area, -1 if not expected)
200	UINT4	Flags: Bit0=Propagation speed available Bit1=Wind speeds valid
204	796	<spare>

The polynomial curve fit is calculated as follows: First, threshold a shear product based on the shear threshold specified in the product configuration. Let *East* and *North* refer to the distance of the center of each pixel from the radar position in centimeters. This coordinate system is rotated by an angle θ clockwise about the radar location to produce a new coordinate system with distances also in centimeters. This is the "Rotation angle to X-Y coordinates of curve fit polynomial" above. I use the variables *X* and *Y* in this coordinate system. The equations for the transformation are:

$$X = North \sin \theta + East \cos \theta$$

$$Y = North \cos \theta - East \sin \theta$$

Call this transformation T_{rotate} , and the reverse transformation T_{rotate}^{-1} .

In this coordinate system, I will call the *X*-coordinate of the left most end of the line X_l and similarly the right most end X_r . Next we shift and scale the coordinate system to keep the polynomial coefficients from becoming too large. The equations for the transformation are:

$$X' = \frac{X - X_l}{X_r - X_l}$$

$$Y' = Y$$

Call this transformation T_{scale} , and the reverse transformation T_{scale}^{-1} .

In this new coordinate system, the polynomial is simple to express:

$$Y' = A_0 + A_1X' + A_2X'^2 + A_3X'^3 + A_4X'^4 + A_5X'^5$$

or

$$Y' = P[X']$$

The standard standard deviation is computed as follows: Let $[X'_i, Y'_i]$ represent the i th point in the data set in the rotated and scaled coordinate system, and N the total number of points, then the standard deviation is:

$$standard\ deviation = \sqrt{\frac{1}{N} \sum_{i=1, N} [Y'_i - P[X'_i]]^2}$$

The center point is computed as follows: Let $[East_i, North_i]$ represent the i th point in the data set in the original coordinate system, and N the total number of points, then the center point is:

$$East\ center = \frac{1}{N} \sum_{i=1, N} East_i$$

$$North\ center = \frac{1}{N} \sum_{i=1, N} North_i$$

4.2.38 sri_psi_struct Structure

Source: headers.h

Byte	Size	Contents
0	UINT4	Flags Bit 0=Do profile correction Bit 3& 4: Melt src: 0=Ingest, 1=Setup, 2=TypeIn Bit 5=Check for convection
4	SINT4	Total number of bins inserted
8	SINT4	Number of bins with data
12	SINT4	Number of data bins profile corrected
16	SINT2	Surface height (m above reference)
18	SINT2	Maximum height (m above reference)
20	SINT2	Melting height (m above MSL)
22	SINT2	Melting level thickness (m)
24	SINT2	Gradient above melting (1/100 dB/km)
26	SINT2	Gradient below melting (1/100 dB/km)

28	SINT2	Convective check height (m above melting)
30	SINT2	Convective check level (DB_DBZ2 format)

4.2.39 status_antenna_info Structure

Source: `product.h`

Byte	Size	Contents
0	BIN4	Azimuth position
4	BIN4	Elevation position
8	UINT4	Azimuth velocity
12	UINT4	Elevation velocity
16	UINT4	Command bits
20	UINT4	Command bit availability mask
24	UINT4	Status bits
28	UINT4	Status bit availability mask
32	UINT4	Bite fault flag, 0=OK, 1=Fault, 2=Critical
36	SINT4	Lowest field number generating a fault
40	UINT4	Status bits which can cause critical faults
UINT4[3]	Mask	44 indicating the state of each BITE field
56	UINT4[3]	Mask indicating which BITE fields are faulted
68	32	<spare>

4.2.40 status_device_info Structure

Source: `product.h`

Byte	Size	Contents
0	40	<status_one_device> dsp
40	40	<status_one_device> antenna
80	720	<status_one_device>[18] output devices

4.2.41 status_message_info Structure

Source: `product.h`

Byte	Size	Contents
------	------	----------

0	SINT4	Message count
4	MESSAGE	Actual message number
8	SINT4	Number of times it was repeated
12	char[16]	Process name, null terminated
28	char[80]	Text of message, null terminated
108	char[32]	Name of signal, null terminated
140	20	<serv_ymds_time> Time of message (TZ flexible)
160	UINT4	Message type: 1=info, 2=normal, 3=say
164	36	<spare>

4.2.42 status_misc_info Structure

Source: product.h

Byte	Size	Contents
0	char[16]	Radar status configuration name, null terminated
16	char[16]	Task configuration name, null terminated
32	char[16]	Product scheduler configuration name, null terminated
48	char[16]	Product output configuration name, null terminated
64	char[16]	Active task name, null terminated
80	char[16]	Active product name, null terminated
96	UINT4	Site type (IRIS style)
100	SINT4	Number of incoming network connects
104	SINT4	Number of IRIS clients connected
108	4	<spare>
112	SINT4	Number of output devices
116	UINT4	Flags: bit 0=Automatic mode switching is enabled bit 1=Regular Fault bit 2=Critical Fault bit 3=Regular message fault bit 4=Critical message fault
120	char[4]	Node status fault site
124	12	<ymds_time> Time of active task (TZ flexible)
136	64	<spare>

4.2.43 status_one_device Structure

Source: `product.h`

Byte	Size	Contents
0	UINT4	Device type
4	SINT4	Unit number
8	UINT4	Status
12	4	<spare>
16	UINT4	Mode from process table
20	char[16]	String (null terminated)
36	4	<spare>

4.2.44 status_one_process Structure

Source: `product.h`

Byte	Size	Contents
0	UINT4	Command
4	UINT4	Mode
8	12	<spare>

4.2.45 status_process_info Structure

Source: `product.h`

Byte	Size	Contents
0	20	<status_one_process> Ingest process
20	20	<status_one_process> Ingfin process
40	20	<spare>
60	20	<status_one_process> Output master process
80	20	<status_one_process> Product process
100	20	<status_one_process> Watchdog process
120	20	<status_one_process> Reingest process
140	20	<status_one_process> Network process
160	20	<status_one_process> Nordrad process
180	20	<status_one_process> Server process
200	20	<status_one_process> RibBuild process
220	180	<spare>

4.2.46 status_results Structure

Source: `product.h`

Byte	Size	Contents
0	200	<status_misc_info> Miscellaneous info
200	400	<status_process_info> Status of processes
600	800	<status_device_info> Status of devices
1400	100	<status_antenna_info> Status of antenna
1500	200	<status_message_info> Message information

4.2.47 structure_header Structure

Source: `headers.h`

Byte	Size	Contents
0	SINT2	Structure identifier: Value; Description 22 Task_configuration 23 Ingest_header 24 Ingest_data_header 25 Tape_inventory 26 Product_configuration 27 Product_hdr 28 Tape_header_record
2	SINT2	Format version number (see headers.h)
4	SINT4	Number of bytes in the entire structure
8	SINT2	Reserved
10	SINT2	Flags: bit0=structure complete

4.2.48 tape_header_record Structure

Source: `output.h`

Byte	Size	Contents
0	12	<structure_header>
12	char[16]	Tape identification name
28	char[16]	Name of site that created tape
44	12	<ynds_time> Time that the tape was created (TZ flexible)
56	4	<spare>
60	char[8]	IRIS version when tape was initialized

68 252 <spare>

4.2.49 task_calib_info Structure

Source: iris_task.h

Byte	Size	Contents
0	SINT2	Reflectivity slope (4096*dB/ A/D count)
2	SINT2	Reflectivity noise threshold (1/16 dB above Noise)
4	SINT2	Clutter Correction threshold (1/16 dB)
6	SINT2	SQI threshold (0–1)*256
8	SINT2	Power threshold (1/16 dBZ)
10	8	<spare>
18	SINT2	Calibration Reflectivity (1/16 dBZ at 1 km)
20	UINT2	Threshold flags for uncorrected reflectivity
22	UINT2	Threshold flags for corrected reflectivity
24	UINT2	Threshold flags for velocity
26	UINT2	Threshold flags for width
28	UINT2	Threshold flags for ZDR
30	6	<spare>
36	UINT2	Flags: Bit 0: Speckle remover for log channel Bit 3: Speckle remover for linear channel Bit 4: Flag to indicate data is range normalized Bit 5: Flag to indicate pulse at beginning of ray Bit 6: Flag to indicate pulse at end of ray Bit 7: Vary number of pulses in dual PRF Bit 8: Use 3 lag processing in PP02 Bit 9: Apply velocity correction for ship motion Bit 10: Vc is unfolded Bit 11: Vc has fallspeed correction Bit 12: Zc has beam blockage correction Bit 13: Zc has Z-based attenuation correction Bit 14: Zc has target detection Bit 15: Vc has storm relative velocity correction
38	2	<spare>
40	SINT2	LDR bias in signed 1/100 dB
42	SINT2	ZDR bias in signed 1/16 dB
44	SINT2	NEXRAD point clutter threshold in 1/100 of dB
46	UINT2	NEXRAD point clutter bin skip in low 4 bits
48	SINT2	I0 cal value, horizontal pol, in 1/100 dBm
50	SINT2	I0 cal value, vertical pol, in 1/100 dBm

52	SINT2	Noise at calibration, horizontal pol, in 1/100 dBm
54	SINT2	Noise at calibration, vertical pol, in 1/100 dBm
56	SINT2	Radar constant, horizontal pol, in 1/100 dB
58	SINT2	Radar constant, vertical pol, in 1/100 dB
60	UINT2	Receiver bandwidth in kHz
62	uint16_t	Flags2: Bit 0: Zc and ZDRc has DP attenuation correction Bit 1: Z and ZDR has DP attenuation correction
64	256	<spare>

4.2.50 task_configuration Structure

Source: iris_task.h

```

<structure_header>(4.2.47 structure_header Structure on page 58) 12 Bytes
<task_sched_info>(4.2.61 task_sched_info Structure on page 64) 120 Bytes
<task_dsp_info>(4.2.51 task_dsp_info Structure on page 60) 320 Bytes
<task_calib_info>(4.2.49 task_calib_info Structure on page 59) 320 Bytes
<task_range_info>(4.2.58 task_range_info Structure on page 63) 160 Bytes
<task_scan_info>(4.2.60 task_scan_info Structure on page 64) 320 Bytes
<task_misc_info>(4.2.56 task_misc_info Structure on page 62) 320 Bytes
<task_end_info>(4.2.53 task_end_info Structure on page 61) 320 Bytes
Comments 720 Bytes

```

4.2.51 task_dsp_info Structure

Source: iris_task.h

Byte	Size	Contents
0	UINT2	Major mode
2	UINT2	DSP type
4	24	<dsp_data_mask> (4.2.7 dsp_data_mask Structure on page 34) Current Data type mask
28	24	<dsp_data_mask> (4.2.7 dsp_data_mask Structure on page 34) Original Data type mask
If Batch Major mode:		
52	32	<task_dsp_mode_batch>
Else:		
52	32	<task_dsp_mode_other>
84	52	<spare>
136	SINT4	PRF in Hertz
140	SINT4	Pulse width in 1/100 of microseconds

144	UINT2	Multi PRF mode flag: 0=1:1, 1=2:3, 2=3:4, 3=4:5
146	SINT2	Dual PRF delay
148	UINT2	AGC feedback code
150	SINT2	Sample size
152	UINT2	Gain Control flag (0=fixed, 1=STC, 2=AGC)
154	char[12]	Name of file used for clutter filter
166	UINT1	Linear based filter number for first bin
167	UINT1	Log based filter number for first bin
168	SINT2	Attenuation in 1/10 dB applied in fixed gain mode
170	UINT2	Gas attenuation in 1/100000 dB/km for first 10000, then stepping in 1/10000 dB/km
172	UINT2	Flag nonzero means cluttermap used
174	UINT2	XMT phase sequence: 0:Fixed, 1:Random, 3:SZ8/64
176	UINT4	Mask used for to configure the ray header.
180	UINT2	Time series playback flags, see OPTS_* in dsp.h
182	2	<spare>
184	char[16]	Name of custom ray header
200	120	<spare>

4.2.52 task_dsp_mode_batch Structure

Source: iris_task.h

Byte	Size	Contents
0	UINT2	Low PRF in Hz
2	UINT2	Low PRF fraction part, scaled by $2^{**}-16$
4	SINT2	Low PRF sample size
6	SINT2	Low PRF range averaging in bins
8	SINT2	Theshold for reflectivity unfolding in 1/100 of dB
10	SINT2	Threshold for velocity unfolding in 1/100 of dB
12	SINT2	Threshold for width unfolding in 1/100 of dB
14	18	<spare>

4.2.53 task_end_info Structure

Source: iris_task.h

Byte	Size	Contents
0	SINT2	Task major number
2	SINT2	Task minor number

4	char[12]	Name of task configuration file
16	char[80]	Task description
96	SINT4	Number of tasks in hybrid task
100	UINT2	Task state: 0=no task; 1=task being modified; 2=inactive; 3=scheduled, 4=running.
102	2	<spare>
104	12	<ynds_time> Data time of task (TZ flexible)
116	UINT1[6]	Identifiers of the echo classifiers in the HCLASS bit segments
122	198	<spare>

4.2.54 task_file_scan_info Structure

Source: iris_task.h

Byte	Size	Contents
0	UINT2	First azimuth angle (binary angle)
2	UINT2	First elevation angle (binary angle)
4	char[12]	Filename for antenna control
16	184	<spare>

4.2.55 task_manual_scan_info Structure

Source: iris_task.h

Byte	Size	Contents
0	UINT2	Flags: bit 0=Continuous recording
2	198	<spare>

4.2.56 task_misc_info Structure

Source: iris_task.h

Byte	Size	Contents
0	SINT4	Wavelength in 1/100 of cm
4	char[16]	T/R Serial Number
20	SINT4	Transmit Power in watts
24	UINT2	Flags: Bit 0: Digital signal simulator in use Bit 1: Polarization in use Bit 4: Keep bit

26	UINT2	Type of polarization
28	SINT4	Truncation height (centimeters above the radar)
32	18	<spare for polarization spec>
50	12	<spare>
62	SINT2	Number of bytes of comments entered
64	BIN4	Horizontal beamwidth (binary angle, starting in 7.18)
68	BIN4	Vertical beamwidth (binary angle, starting in 7.18)
72	UINT4[10]	Customer defined storage (starting in 7.27)
112	208	<spare>

4.2.57 task_ppi_scan_info Structure

Source: `iris_task.h`

Byte	Size	Contents
0	BIN2	Left azimuth limit (binary angle, only for sector)
2	BIN2	Right azimuth limit (binary angle, only for sector)
4	UINT2[40]	List of elevations (binary angles) to scan at
84	115	<spare>
199	1	Start of first sector sweep: 0=Nearest, 1=Left, 2=Right Sector sweeps alternate in direction.s

4.2.58 task_range_info Structure

Source: `iris_task.h`

Byte	Size	Contents
0	SINT4	Range of first bin in centimeters
4	SINT4	Range of last bin in centimeters
8	SINT2	Number of input bins
10	SINT2	Number of output range bins
12	SINT4	Step between input bins
16	SINT4	Step between output bins (in centimeters)
20	UINT2	Flag for variable range bin spacing (1=var, 0=fixed)
22	SINT2	Range bin averaging flag
24	136	<spare>

4.2.59 task_rhi_scan_info Structure

Source: `iris_task.h`

Byte	Size	Contents
0	UINT2	Lower elevation limit (binary angle, only for sector)
2	UINT2	Upper elevation limit (binary angle, only for sector)
4	UINT2[40]	List of azimuths (binary angles) to scan at
84	115	<spare>
199	1	Start of first sector sweep: 0=Nearest, 1=Lower, 2=Upper Sector sweeps alternate in direction.

4.2.60 task_scan_info Structure

Source: `iris_task.h`

Byte	Size	Contents
0	UINT2	Antenna scan mode 1:PPI sector, 2:RHI, 3:Manual, 4:PPI cont, 5:file
2	SINT2	Desired angular resolution in 1/1000 of degrees
4	2	<spare>
6	SINT2	Number of sweeps to perform
If RHI scan:		
8	200	<task_rhi_scan_info>
If PPI sector, or PPI continuous scan:		
8	200	<task_ppi_scan_info>
If File scan:		
8	200	<task_file_scan_info>
If PPI Manual or RHI Manual scan:		
8	200	<task_manual_scan_info>
In all cases:		
208	112	<spare>

4.2.61 task_sched_info Structure

Source: `iris_task.h`

Byte	Size	Contents
0	SINT4	Start time (seconds within a day)
4	SINT4	Stop time (seconds within a day)

8	SINT4	Desired skip time (seconds)
12	SINT4	Time last run (seconds within a day)(0 for passive ingest)
16	SINT4	Time used on last run (seconds) (in file time to writeout)
20	SINT4	Relative day of last run (zero for passive ingest)
24	UINT2	Flag: Bit 0 = ASAP Bit 1 = Mandatory Bit 2 = Late skip Bit 3 = Time used has been measured Bit 4 = Stop after running
26	94	<spare>

4.2.62 tdwr_psi_struct Structure

Source: headers.h

Byte	Size	Contents
0	UINT4	Flags, bit0=LLWAS, bit1=WARN, bit2=SLINE
4	UINT4	Maximum range in cm
8	char[4]	Source ID
12	char[3]	Center field wind direction
15	UINT1	<spare>
16	char[2]	Center field wind speed
18	char[2]	Center field gust speed
20	UINT4	Mask of protected areas checked
24	UINT4	Warning count
28	UINT4	SLINE count
32	UINT4	Forecast time

4.2.63 tdwr_results Structure

Source: product.h

Byte	Size	Contents
0	char[16]	Corridor name, null terminated
16	char[8]	Arena Name, null terminated
24	char[3]	Alert Type
27	1	<spare>
28	char[3]	Threshold direction
31	1	<spare>
32	char[2]	Threshold speed

34	2	<spare>
36	SINT4	Alert speed loss/gain in mm/sec, negative=loss
40	UINT4	Mask of protected areas checked for this corridor
44	UINT4	Mask of protected areas hit in this corridor
48	52	<spare>

4.2.64 text_results Structure

Source: `product.h`

Byte	Size	Contents
0	char[512]	null terminated string of arbitrary text to be spoken by IRIS

4.2.65 top_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	uint32_t	Flags: Bit 0: For THICK product only, make pseudo thickness
4	int16_t	Z threshold in 1/16 dBZ

4.2.66 track_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	SINT4	Centroid area threshold in square meters
4	SINT4	Threshold level for centroid
8	UINT4	Protected area mask
12	SINT4	Maximum forecast time in seconds
16	UINT4	Maximum age between products for motion calculation
20	SINT4	Maximum motion allowed in mm/sec
24	UINT4	Flag word: Bit9=Generate diagnostic output
28	SINT4	Maximum span of track points in the file (seconds)
32	UINT4	Input product type
36	char[12]	Input product name

48 SINT4 Point connecting error allowance

4.2.67 track_results Structure

Source: product.h

Byte	Size	Contents
0	BIN4	Latitude (32-bit binary angle)
4	BIN4	Longitude
8	SINT4	Height (cm. above reference)
12	UINT4	Flags: bit0=forecast, bit1=manual, bit2=text, bit3=icon
16	SINT4	Area of centroid (1/100 of square km)
20	SINT4	Major axis of equal area ellipse (cm)
24	SINT4	Minor axis of equal area ellipse (cm)
28	BIN4	Orientation angle of ellipse
32	UINT4	Protected area mask of areas hit
36	SINT4	Maximum value of data within area
40	8	<spare>
48	SINT4	Average value of data within area
52	8	<spare>
60	SINT4	Scale factor of input data
64	SINT4	Track index number
68	char[32]	Text
100	12	<ynds_time> Time (TZ flexible)
112	SINT4[32]	ETA in seconds for each protected area (0 if in area, -1 if not expected)
240	UINT4	Data type of input data
244	8	<spare>
252	SINT4	Propagation speed (mm/second)
256	BIN4	Propagation direction (binary angle)
260	UINT4	Text size
264	UINT4	Color
268	32	<spare>

4.2.68 vad_psi_struct Structure

Source: headers.h

Byte	Size	Contents
0	int32_t	Minimum slant range in cm.

4	int32_t	Maximum slant range in cm.
8	uint32_t	Flags: Bit 0: Unfold based on VVP product
12	uint32_t	Count of the number of elevation angles in the file

4.2.69 vad_results Structure

Source: headers.h

Byte	Size	Contents
0	BIN2	Elevation angle
2	BIN2	Azimuth angle
4	uint16_t	Count of the number of bins averaged
6	uint16_t	Average velocity (DB_VEL2 format)
8	uint16_t	Standard deviation (cm/sec)
10	uint16_t	Elevation index, origin 0
12	8	<spare>

4.2.70 vvp_psi_struct Structure

Source: headers.h

Byte	Size	Contents
0	SINT4	Minimum range to process in cm
4	SINT4	Maximum range to process in cm
8	SINT4	Minimum height to process (cm above reference)
12	SINT4	Maximum height to process (cm above reference)
16	SINT2	Number of intervals to process at
18	uint16_t	Minimum velocity (cm/sec) (version 8.13 and later, previously 0.02 times nyquist)
20	SINT4	Quota number of bins per interval
24	SINT4	Wind parameters mask. See bits defined in vvp_results structure below.

4.2.71 vvp_results Structure

Source: product.h

Byte	Size	Contents
0	SINT4	Number of data points used

4	SINT4	Height of the center of interval (cm above reference)
8	SINT4	Number of reflectivity data points used
12	8	<spare>
Bit Description		
20	SINT2	0 Wind speed in cm/sec
22	SINT2	1 Wind speed standard deviation
24	SINT2	2 Wind direction in 1/10 of degrees
26	SINT2	3 Wind direction standard deviation
28	SINT2	4 Vertical wind speed in cm/sec
30	SINT2	5 Vertical wind speed standard deviation
32	SINT2	6 Horizontal divergence in $10^{*-7}/\text{sec}$
34	SINT2	7 Horizontal divergence standard deviation
36	SINT2	8 Radial velocity standard deviation
38	SINT2	9 Linear averaged reflectivity (DB_CDBZ2 format)
40	SINT2	10 Log averaged reflectivity standard deviation
42	SINT2	11 Deformation in $10^{*-7}/\text{sec}$
44	SINT2	12 Deformation standard deviation
46	SINT2	13 Axis of dilatation in 1/10 of degrees
48	SINT2	14 Axis of dilatation standard deviation
50	SINT2	15 Log averaged reflectivity (DB_CDBZ2 format)
52	SINT2	16 Linear averaged reflectivity standard deviation
54	30	<spare>

4.2.72 warn_psi_struct Structure

Source: headers.h

Byte	Size	Contents
0	SINT4	Centroid area threshold in square meters
4	SINT4[3]	Threshold levels (1/100 of user units)
16	SINT2[3]	Data valid times (seconds)
22	2	<spare>
24	char[12]	Symbol to display
36	char[36]	Names of the product files
72	UINT1[3]	Product types
75	char[1]	<spare>
76	UINT4	Protected area bit flag

4.2.73 warning_results Structure

Source: `product.h`

Byte	Size	Contents
0	BIN4	Latitude (32-bit binary angle)
4	BIN4	Longitude
8	SINT4	Height (cm above reference)
12	UINT4	Flags: Bit 0: Skip data label
16	SINT4	Area of centroid (1/100 of square km)
20	SINT4	Major axis of equal area ellipse (cm)
24	SINT4	Minor axis of equal area ellipse (cm)
28	BIN4	Orientation angle of ellipse
32	UINT4	Protected area mask of areas hit
36	SINT4[3]	Maximum value of data within the area (1/100 of user units)
48	SINT4[3]	Average value of data within the area (1/100 of user units)
60	SINT4	Scale factor of input data
64	4	<spare>
68	char[16]	Text, null-terminated
84	156	<spare>
240	UINT4[3]	Data type of input data
252	SINT4	Propagation speed (mm/second)
256	BIN4	Propagation direction (binary angle)
260	40	<spare>

4.2.74 wind_psi_struct Structure

Source: `headers.h`

Byte	Size	Contents
0	SINT4	Minimum height (cm above reference)
4	SINT4	Maximum height (cm above reference)
8	SINT4	Minimum range in cm
12	SINT4	Maximum range in cm
16	SINT4	Number of point in range
20	SINT4	Number of points in panels
24	SINT4	Sector length in cm
28	BIN4	Sector width in binary angle
32	UINT4	Flag word: bit0=Subtract mean wind
36	UINT4	Wind parameters mask of included VVP

4.2.75 wind_results Structure

Source: `product.h`

Byte	Size	Contents
0	SINT4	Number of possible hits based on geometry
4	SINT4	Number of data points actually used
8	SINT4	Range of center of sector
12	BIN2	Azimuth of center of sector
14	SINT2	Velocity East in cm/second
16	SINT2	Velocity East standard deviation in cm/second
18	SINT2	Velocity North in cm/second
20	SINT2	Velocity North standard deviation in cm/second
22	10	<spare>

4.2.76 ymds_time Structure

Source: `sigtypes.h`

Byte	Size	Contents
0	SINT4	Seconds since midnight
4	UINT2	Milliseconds in lower 10 bits. Bit 10: Time is daylight savings time Bit 11: Time is UTC Bit 12: Local time is daylight savings time
6	SINT2	Year
8	SINT2	Month
10	SINT2	Day

4.3 Data Types

The following sections document the data formats for both ingest data (stored in the raw product) and other products. The data formats for reflectivity, velocity, width, and ZDR differ slightly in the two cases. When these data types are converted to a Cartesian product, the data values of 255 are replaced with 254, and 255 takes on the meaning of area not scanned. Polar data does not have this meaning. The name in parenthesis after each section title is the data type name used by IRIS to identify the data. Parameters for these numbers are defined in the `sigtypes.h` file.

4.3.1 Extended_Header Format (DB_XHDR)

The extended header is optionally recorded, as controlled by a question in Setup. There are two versions, either `extended_header_v0` or `extended_header_v1`. Generally, the v1 version is used for moving-platform systems, such as on ships.

4.3.2 2-byte Axis of Dilliation Format (DB_AXDIL2)

The angle in degrees is stored as a signed number in tenths of degrees.

$$angle = \frac{N}{10}$$

-1800	-180.0 degrees
0	0.0 degrees
10	1.0 degree

4.3.3 1-byte Reflectivity Format (DB_DBT& DB_DBZ)

For reflectivity data, the number in decibels is computed from the unsigned output with the formula:

$$dBZ = \frac{N - 64}{2}$$

The overall range is from -31.5 dBZ to +95.5 dBZ in half dB steps as follows. In data product files, the value of 255 indicates areas not scanned.

	Ingest Data	Products
0	No data available	No data available
1	-31.5 dBZ	-31.5 dBZ
64	0.0 dBZ	0.0 dBZ
128	32.0 dBZ	32.0 dBZ
129	32.5 dBZ	32.5 dBZ
254	95.0 dBZ	95.0 dBZ or above
255	95.5 dBZ or above	Area not scanned

4.3.4 2-byte Reflectivity Format (DB_DBT2& DB_DBZ2)

For reflectivity data, the number in decibels is computed from the unsigned output with:

$$dBZ = \frac{N - 32768}{100}$$

The overall range is from -327.67 to +327.66 in 1/100 of a dB steps as follows.

0	No data available
1	-327.67 dBZ
32768	0.00 dBZ
32769	0.01 dBZ
65534	327.66 dBZ
65535	Reserved for area not scanned in product files

4.3.5 2-byte Deformation Format (DB_DEFORM2)

Deformation is stored in a signed 16-bits number scaled to 10**-7. Only positive numbers are possible. These number are normally displayed in units of 10**-4.

0	0 deformation
1	0.001 10**-4 deformation
32766	32.766 10**-4 deformation
32767	Area not scanned

4.3.6 2-byte Divergence Format (DB_DIVERGE2)

Divergence is stored in a signed 16-bits number scaled to 10**-7. Positive number indicate divergence while negative are convergence. These number are normally displayed in units of 10**-4.

-32768	32.768 10**-4 convergence
0	0 divergence
1	0.001 10**-4 divergence
32766	32.766 10**-4 divergence

32767

Area not scanned

4.3.7 2-byte Floating Liquid Format (DB_FLIQUID2)

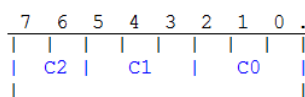
Rainfall accumulations are stored in 16-bit floating-point representation of a 27-bit integer in units of 0.001 mm. This product does not have a code for thresholded data, instead that area is assumed to have zero rain. The floating-point number consists of a 4-bit exponent in the high four bits, followed by a 12-bit mantissa. It uses implied digits and soft underflow. If the exponent is 0, the mantissa consists of the value. If the exponent is 1 through 15, the 12-bit mantissa has a 1 prefixed to it and is shifted up by one less than the exponent. The table below gives some examples:

Float		Fixed	Meaning
0	—>	0	0.000 mm
1	—>	1	0.001 mm
255	—>	255	0.255 mm
1000	—>	1000	1.000 mm
9096	—>	10000	10.000 mm
22634	—>	100000	100.000 mm
34922	—>	800000	800.000 mm
50000	—>	10125312	10125.312 mm
65534	—>	134184960	134184.960 mm or above
65535	—>		Area not scanned

4.3.8 1-byte HydroClass Format (DB_HCLASS)

HydroClass consists of enumerated data to communicate echo identification results. The 1-byte data type can accommodate results from upto three independent identification methods or classification hypotheses, as described next. The decimal values of "0" (0x00) and "255" (0x255) are reserved for special uses of "No data" and "Area not scanned", respectively.

1-byte of binary DB_HCLASS data is split into three bit segments as follows:



Each segment can be associated with an identification method (echo classifier). The available classification methods are listed as enumerated logical names in the public IRIS/RDA header *sig_data_types.h*.

The bytes "Enums" of the `task_dsp_info` structure (see [4.2.51 task_dsp_info Structure on page 60](#)) contain the list of classification methods (if any) that have been allocated to each bit segment of the HClass data. The information are associated to each IRIS RAW data file header, and can be inspected with the **productx** utility.

The bits in each segment are interpreted as an unsigned integer value belonging to an enumerated class type (the class base), specific to each classifier. The enumerated class types are defined in *sig_data_types.h*, too, by using unambiguous logical names that are short but meteorologically explicit.

For DB_HCLASS data generated with an IRIS/RDA version of 8.12.5 or older, all the values of DB_HCLASS (as well as DB_HCLASS2) in the range (1..7) are interpreted as classification results of the HydroClass (METEOCLASSIFIER). These results are stored in the first method space (C0), with the following logical names:

DV_HCLASS_THRESHOLD	=0	No data available
DV_HCLASS_NON_MET	=1	Non-meteorological target
DV_HCLASS_RAIN	=2	Rain
DV_HCLASS_WET_SNOW	=3	Wet Snow
DV_HCLASS_SNOW	=4	Snow
DV_HCLASS_GRAUPEL	=5	Graupel
DV_HCLASS_HAIL	=6	Hail
	=7	Unused

In the IRIS/RDA version 8.12.6 and later, the default configuration of IRIS/RDA delivers the following echo classification results (the configuration can be modified by settings in *dpolapp_*.conf*).

Method 1 - METEOCLASSIFIER (HydroClass):

MET_CLASS_THRESHOLD	=0	No data available
MET_CLASS_NON_MET	=1	Non-meteorological target
MET_CLASS_RAIN	=2	Rain

MET_CLASS_WET_SNOW	=3 Wet Snow
MET_CLASS_SNOW	=4 Snow
MET_CLASS_GRAUPEL	=5 Graupel
MET_CLASS_HAIL	=6 Hail
	=7 Unused

Method 2 - PRECIPCLASSIFIER:

PRE_CLASS_THRESHOLD	=0 No data available
PRE_CLASS_GC_AP	=1 Ground clutter / anomalous propagation
PRE_CLASS_BIO	=2 Bio scatter
PRE_CLASS_PRECIP	=3 Precipitation
PRE_CLASS_LARGE_DROPS	=4 Large drops
PRE_CLASS_LIGHT_PRECIP	=5 Light precipitation
PRE_CLASS_MODERATE_PRECIP	=6 Moderate precipitation
PRE_CLASS_HEAVY_PRECIP	=7 Unused

For PRECIPCLASSIFIER, the interpretations into classes from "Large Drops" to "Heavy precipitation" are optional, available in case HClass PrecipClassifier is configured ON. Without PrecipClassifier, all precipitation are classified as "Precipitation".

Method 3 - CELLCLASSIFIER:

CELL_CLASS_STRATIFORM	=0 Stratiform
CELL_CLASS_CONVECTION	=1 Convection
	=2 Unused
	=3 Forbidden

The value 255 (0xFF) of DB_HCLASS is reserved for "area not scanned", and the use of value 3 (0x03) is forbidden in Method 3.

The methods METEO- and PRECIPCLASSIFIER are available both in RDA real-time ray level processing, as well as in IRIS reprocessing (reingest). CELLCLASSIFIER is available in IRIS post-processing, only, and its outcomes can be interpreted in the context of ray level classifications.

Further classification information may become available, which can be appended to HydroClass data of the 2-byte mode, see [4.3.9 2-byte HydroClass Format \(DB_HCLASS2\) on page 77](#).

4.3.9 2-byte HydroClass Format (DB_HCLASS2)

The interpretation of the eight least significant bits is the same as the 1-byte format, except that 65535 (0xFFFF) is used for "area not scanned".

Currently available identification methods (as of IRIS/RDA 8.12.6) can be accommodated in the 8 least significant bits of DB_HCLASS2, see [4.3.8 1-byte HydroClass Format \(DB_HCLASS\) on page 74](#).

As default, the remaining 8 most significant bits of DB_HCLASS2 replicate the same formatting as the lower 1-byte bits, for support of three further identification methods. They are reserved for future uses, such as re-analysis of precipitation type at surface. It is also possible to merge the bit segments by combining the segments within the 8 most significant bits, multiplicatively. In that scenario, DB_HCLASS2 data type is capable of supporting a user specified identification method in large bases up to $7 \times 7 \times 3 = 147$ classes.

4.3.10 1-byte Echo Tops Format (DB_HEIGHT)

Echo tops are stored to the nearest 100 meters above the ground. Because 0 is the code for no data, values stored in the bytes can be converted to kilometers by subtracting 1 and dividing by 10. The value 254 indicates that an echo is known to exist from the data, but it could not be measured.

0	No echo tops data available
1	0.0 km
128	12.7 km
129	12.8 km
253	25.2 km or above
254	Top exists above the maximum tilt
255	Area not scanned

4.3.11 2-byte Horizontal wind direction Format (DB_HDIR2)

The angle in degrees is stored as a signed number in tenths of degrees.

$$angle = \frac{N}{10}$$

-1800	-180.0 degrees
0	0.0 degrees
10	1.0 degree

4.3.12 1-byte KDP Format (DB_KDP)

Specific differential phase (KDP) is stored in a signed 8-bit number using a log scale. The KDP angles are multiplied by the wavelength in cm then scaled using a log scale separately for both signs. The minimum value is 0.25, and the maximum value is 150.0 deg*cm/km. KDP values are defined as the one-way differential effect of the intervening weather. The table below gives some examples:

$$KDP \times \lambda = minimum \times \left[\frac{minimum}{maximum} \right]^{\left[\frac{N-129}{126} \right]}$$

Here is the conversion equation for positive values (stored value above 128):

$$KDP \times \lambda = 0.25 \times 600 \left[\frac{N-129}{126} \right]$$

Here is the conversion equation for negative values (stored value below 128):

$$KDP \times \lambda = -0.25 \times 600 \left[\frac{127-N}{126} \right]$$

Value	Meaning KDP*L	KDP for 10 cm	KDP for 5 cm
0	No data available		
1	-150.00 deg*cm/km	-15.00 deg/km	-30.00 deg/km
2	-142.58	-14.26	-28.51
127	-0.250	-0.025	-0.050
128	0.000	0.000	0.000
129	0.250	0.025	0.050
130	0.263	0.026	0.053
254	142.58	14.58	28.51
255	Area not scanned		

4.3.13 2-byte KDP Format (DB_KDP2)

Specific differential phase (KDP) in degrees per kilometer is computed from the unsigned output with:

$$KDP = \frac{N - 32768}{100}$$

The overall range is from -327.67 to +327.66 in 1/100 of a degree/kilometer steps as follows.

0	No data available
1	-327.67 deg/km
32768	0.00 deg/km
32769	0.01 deg/km
65534	327.66 deg/km
65535	Reserved for area not scanned in product files

4.3.14 1-byte LDR Format (DB_LDRH & DB_LDRV)

LDR is short for "Linear Depolarization Ratio". This is the ratio of the power received in the depolarized channel to the power received in the main channel. There are 2 flavors: If you are transmitting horizontal, then the ratio of the vertical power to the horizontal power is "LDRH". Similarly if transmitting vertical then the ratio of horizontal power to vertical power is "LDRV". Note that in simultaneous transmission LDR cannot be computed because the signal in the depolarized channel is dominated by co-polarized returns.

$$LDR(db) = \frac{N-1}{5} - 45.0$$

Like most power ratios this is expressed in dB. The overall range is from -45 to +5.6 in steps of 0.2 dB as follows.

0	No data available
1	-45.0 dB
2	-44.8 dB
226	0.0 dB
254	+5.6 dB
255	Reserved for area not scanned in product files

4.3.15 2-byte LDR Format (DB_LDRH2 & DB_LDRV2)

Same as DB_DBZ2 format, see [4.3.4 2-byte Reflectivity Format \(DB_DBT2& DB_DBZ2\)](#) on page 73.

4.3.16 1-byte Phi Format (DB_PHIH & DB_PHIV)

The cross channel differential phase. Same format as DB_PHIDP, see [4.3.18 1-byte PhiDP Format \(DB_PHIDP\)](#) on page 80.

4.3.17 2-byte Phi Format (DB_PHIH2 & DB_PHIV2)

The cross channel differential phase. Same format as DB_PHIDP2, see [4.3.19 2-byte PhiDP Format \(DB_PHIDP2\)](#) on page 80.

4.3.18 1-byte PhiDP Format (DB_PHIDP)

PhiDP is defined as the two-way measured phase effect after the signal has travelled through the intervening weather twice. Differential phase (PhiDP) in degrees is computed from the unsigned output with:

$$\Phi_{DP} \bmod 180 = 180 * \frac{N-1}{254}$$

The overall range is from 0 to 180 degrees in steps of 0.71 as follows.

0	No data available
1	0.00 deg
2	0.71 deg
101	70.87 deg
254	179.29 deg
255	Reserved for area not scanned in product files

4.3.19 2-byte PhiDP Format (DB_PHIDP2)

PhiDP is defined as the two-way measured phase effect after the signal has travelled through the intervening weather twice. Differential phase (PhiDP) in degrees is computed from the unsigned output with:

$$\Phi_{DP} \bmod 360 = 360 * \frac{N - 1}{65534}$$

The overall range is from 0 to 360 degrees in steps of 0.0055 as follows. In cases where the transmitter was alternating polarization, or the data was converted from 1-byte format the values will only cover 0–180 degrees.

0	No data available
1	0.0000 deg
2	0.0055 deg
65534	359.9945 deg
65535	Reserved for area not scanned in product files

4.3.20 2-byte Rainfall Rate Format (DB_RAINRATE2)

Rainfall rates are stored in 16-bit floating-point representation of a 27-bit integer in units of 0.0001 mm/hr. The floating-point number consists of a 4-bit exponent in the high four bits, followed by a 12-bit mantissa. It uses implied digits and soft underflow. If the exponent is 0, the mantissa consists of the value. If the exponent is 1 through 15, the 12-bit mantissa has a 1 prefixed to it, making it a 13-bit mantissa. This 13-bit mantissa is shifted up by one less than the exponent. The table below gives some examples:

Float		Fixed	Meaning
0	—>	0	No data available
1	—>	1	0.0000 mm/hr
2	—>	2	0.0001 mm/hr
225	—>	225	0.0254 mm/hr
1000	—>	1000	0.0999 mm/hr
9096	—>	10000	0.9999 mm/hr
22634	—>	100000	9.9999 mm/hr
34922	—>	800000	79.9999 mm/hr
50000	—>	10125312	1012.5311 mm/hr
65534	—>	134184960	13418.4959 mm/hr or above
65535	—>		Area not scanned

4.3.21 1-byte Rho Format (DB_RHOH & DB_RHOV)

The cross channel correlation coefficient. Same format as DB_RHOHV, see [4.3.23 1-byte RhoHV Format \(DB_RHOHV\)](#) on page 82.

4.3.22 2-byte Rho Format (DB_RHOH2 & DB_RHOV2)

The cross channel correlation coefficient. Same format as DB_SQI2, see [4.3.27 2-byte Signal Quality Index Format \(DB_SQI2\)](#) on page 83.

4.3.23 1-byte RhoHV Format (DB_RHOHV)

RhoHV a measure of how correlated the power fluctuations in reflectivity in the horizontal receiver is to the vertical receiver. It is a dimensionless number on a scale for which 0 means no correlation, and 1 means complete correlation. Since numbers are more likely near 1, the data is scaled with a square root.

$$RhoHV = \sqrt{\frac{N-1}{253}}$$

0	Data not available at this range
1	0.0000
2	0.0629
128	0.7085
253	0.9980
254	1.0000
255	Area not scanned

4.3.24 2-byte RhoHV Format (DB_RHOHV2)

Same format as DB_SQI2, see [4.3.27 2-byte Signal Quality Index Format \(DB_SQI2\)](#) on page 83.

4.3.25 1-byte Wind Shear Format (DB_SHEAR)

Wind shear is stored to the nearest 0.2 meters per second per kilometer. This is a signed number, with positive indicating wind is increasing in velocity away from the radar with increasing range. To convert, subtract 128 and multiply by 0.2.

0	No wind shear data available
1	-25.4 m/s/km or above
128	0.0 m/s/km
129	+0.2 m/s/km
254	+25.2 m/s/km or above
255	Area not scanned

4.3.26 1-byte Signal Quality Index Format (DB_SQI)

The Signal Quality Index (SQI) is the ratio of the magnitude of R1 to the magnitude of R0. It is a dimensionless number on a scale for which 0 means pure noise, and 1 means a pure sine wave.

$$SQI = \sqrt{\frac{N-1}{253}}$$

0	SQI data not available at this range
1	0.0000
2	0.0629
128	0.7085
253	0.9980
254	1.0000
255	Area not scanned

4.3.27 2-byte Signal Quality Index Format (DB_SQI2)

Stored linearly using 16-bits as follows:

$$SQI = \frac{(N-1)}{65533}$$

0	SQI data not available at this range
1	0.00000
2	0.00002
128	0.00194
65533	0.99998
65534	1.00000
65535	Area not scanned

4.3.28 2-byte Time Format (DB_TIME2)

Stored as time in seconds, so in minutes the conversion is as follows:

$$Time = \frac{N - 32768}{60}$$

0	Time data not available at this range
1	- 9:06:07:
32768	00:00:00
32828	00:01:00
65535	Area not scanned

4.3.29 1-byte Velocity Format (DB_VEL)

Mean velocity is with respect to the Nyquist velocity, and is expressed as follows. In data product files, the value 255 is used to indicate area not scanned. The Nyquist velocity is wavelength times PRF divided by 4. For 2:3 dual PRF mode, it is doubled; for 4:3 PRF mode, it is tripled, and for 4:5 PRF mode it is quadrupled over that of the higher PRF. For alternating polarization it is halved.

$$Velocity = \frac{N - 128}{127} \times Nyquist$$

0	Velocity data not available at this range
1	Unambiguous velocity towards the radar
128	Zero velocity
255	Unambiguous velocity away from the radar

4.3.30 2-byte Velocity Format (DB_VEL2)

Mean velocity in meters per second is computed from the unsigned output with

$$velocity = \frac{N - 32768}{100}$$

The overall range is from -327.67 to +327.66 in 1/100 of a meter per second steps as follows.

0	No data available
1	-327.67 m/s (towards the radar)
32768	0.00 m/s
32769	0.01 m/s
65534	327.66 m/s (away from the radar)
65535	Reserved for area not scanned in product files

4.3.31 1-byte Unfolded Velocity Format (DB_VELC)

This is mean radial velocity corrected for both Nyquist folding and fall speed. It is scaled to cover a fixed span of +/- 75 meters/second. In data product files, the value 255 is used to indicate area not scanned.

0	Velocity data not available at this range
1	-75.0 m/s (towards the radar)
2	-74.4 m/s
128	Zero velocity
129	+0.6 m/s
254	+75.0 m/s (away from the radar)
255	Area not scanned

4.3.32 2-byte Unfolded Velocity Format (DB_VELC2)

Mean velocity in meters per second is computed from the unsigned output with

$$velocity = \frac{N - 32768}{100}$$

The overall range is from -327.67 to +327.66 in 1/100 of a meter per second steps as follows.

0	No data available
1	-327.67 m/s (towards the radar)
32768	0.00 m/s
32769	0.01 m/s
65534	327.66 m/s (away from the radar)
65535	Reserved for area not scanned in product files

4.3.33 2-byte VIL Format (DB_VIL2)

Vertically integrated liquid is stored in 16-bits to the nearest 0.001 mm. Because 0 is the code for no data, values stored in the bytes can be converted to millimeters by subtracting 1 and dividing by 1000.

0	No VIL data available
1	0.000 mm
128	0.127 mm
129	0.128 mm
255	0.254 mm
65534	65.533 mm or above
65535	Area not scanned

4.3.34 2-byte Vertical Velocity Format (DB_VVEL2)

Vertical velocity is stored in a signed 16-bits number scaled to 0.01 meters/second. Positive number indicate upward motion, negative downward.

-32768	327.68 m/s upward motion
0	0 motion
1	0.01 m/s fall speed
32766	327.66 m/s fall speed
32767	Area not scanned

4.3.35 1-byte Width Format (DB_WIDTH)

Spectrum width is computed from the unsigned output as:

$$W = \frac{n}{256}$$

The overall range is therefore a fraction between 1/256 and 255/256. The code of 0 indicates that width data is not available at this range. To convert the width to meters per second, multiply by the unambiguous velocity. Thus the width has twice the resolution of the velocity. This unambiguous velocity is not enlarged by the dual PRF scheme, but is halved by alternating polarization. In data products, the value 255 indicates area not scanned. Note that width unambiguous velocities are not changed for dual PRF unfolding.

4.3.36 2-byte Width Format (DB_WIDTH2)

Spectral width in meters per second is computed from the unsigned output with

$$width = \frac{N}{100}$$

The overall range is from 0.01 to 655.34 in 1/100 of a meter per second steps as follows.

0	No data available
1	0.01 m/s
32768	327.68 m/s
32769	327.69 m/s
65534	655.34 m/s
65535	Reserved for area not scanned in product files

4.3.37 1-byte ZDR Format (DB_ZDR)

For differential reflectivity data, the number in decibels is computed from the unsigned output with the formula:

$$dB(ZDR) = \frac{N - 128}{16}$$

The overall range is from -7.94 dBZ to +7.94 dBZ in sixteenth of a dB steps as shown below. Positive ZDR means that the horizontal return is stronger than the vertical return. In data products, the value 255 indicates area not scanned.

0	No ZDR data available
---	-----------------------

1	-7.94 dB
128	0.00 dB
129	+0.06 dB
255	+7.94 dB

4.3.38 2-byte ZDR Format (DB_ZDR2)

For differential reflectivity data, the number in decibels is computed from the unsigned output with

$$dB(ZDR) = \frac{N - 32768}{100}$$

The overall range is from -327.67 to +327.66 in 1/100 of a dB steps as follows.

0	No data available
1	-327.67 dB
32768	0.00 dB
32769	0.01 dB
65534	327.66 dB
65535	Reserved for area not scanned in product files

4.4 Ingest Data File Format

Each ingest data file contains one data type from one sweep of a volume. If extended headers are recorded, they are treated as another data type and are placed in a separate file. Each of these files consists of a fixed length ingest_data_header structure followed by a variable-length ray pointer table, followed by a variable-length segment of data, as shown in [Table 9 on page 88](#). The ray pointer table contains a 32-bit pointer for each ray in the file. These are byte pointers (origin one) referenced to the beginning of the data area. Thus, a pointer value of 1 refers to the first byte of the data area. Ray data are forced onto 16-bit word boundaries, so all the pointers are odd. A pointer value of 0 indicates that no data are available for that slot.

Table 9 Ingest Data File Format

<ingest_data_header>
76 Bytes
Pointer Table
4 Bytes per expected ray
Data Area
Undetermined Size

Ray data are partially compressed with a truncation scheme. If a ray does not contain data all the way out to the last range bin, those trailing bins are removed from the archive. This means that the storage space required for each ray varies, which requires the pointer table for quick random access. The pointer points to a ray_header structure, which is followed by an array of range bins. The count of the number of range bins could be zero, in which case the ray contains only the 12-byte ray header.

4.4.1 Ingest File Names

The IRIS ingest data for a volume scan is stored on disk in multiple files. The ingest summary file name is just the prefix with a trailing ".", but without the suffix. Each sweep and data type is stored in a separate file. These files must be named correctly, **siris** will delete any with an incorrect name. If the two-digit year is less than 50, it means add 2000, otherwise add 1900.

General Format	Example
SSSYMMDDHHMMSS.##DD	SIG010620141921.01dBZ
Where:	
SSS—Three letter site code from Setup	SIG—Abbreviation for SIGMET
YYMMDDHHMMSS—Data time	940620141921—14:19:21 20 June 2001
##—Two-digit sweep number	01—Sweep #1
DD—data type (1–5 chars)	dBZ—Reflectivity

4.5 Product File Format

Each product is stored in a separate file in the directory specified by the `IRIS_PRODUCT` environment variable. Raw products are stored in a separate directory specified by the `IRIS_PRODUCT_RAW` environment variable. Product file names consist of the 15-character site and time to the left of the dot, and a 7-character product type code and machine generated string to the right.

The file consists of the product_hdr structure followed by the data. The product configuration structure is exactly what is in one of the product configuration files. It specifies how a product should be generated, and so includes some information not strictly required to appear in the final disk file.

<product_hdr> (3.2.28) 640 Bytes
Optional <protect_setup> (3.2.31) 1024 Bytes
Binary Data Area Variable Size
Optional Extended Product Header Variable Size
Optional Additional Binary Data Area Variable Size

Figure 2 Product File Format

4.5.1 Cartesian Product Format

The data portion of Cartesian product files is not compressed and consists of an array. The first byte contains the lower left corner of the image, the second contains the pixels to the right of that, and so on from the bottom of the window to the top. For 3-D products, the lowest 2-D image comes first.

In some cases we will add a product header extension to the Cartesian products, and may store multiple data types. See section [4.2.26 product_header_extensions on page 48](#) for details.

4.5.2 FCAST Product Format

The FCAST product format is the same as other Cartesian products except that the data elements consist of the `ndop_results` structure, rather than a 1- or 2-byte number.

4.5.3 NDOP Product Format

The NDOP product format is the same as other Cartesian products except that the data elements consist of the `ndop_results` structure, rather than a 1- or 2-byte number. 3-D data is supported.

4.5.4 RAW Product Format

The RAW product is a collection of all raw ingest data acquired during a run of a single task (volume scan). Whereas the ingest data files are stored

on disk separately by sweep and by data moment, the raw product is a single file into which many ingest files have been incorporated. For hybrid tasks, individual raw Products are made from each of the individual tasks that make up the overall scan. Optionally the RAW product can be made in separate files on a sweep-by-sweep basis.

Raw product files are blocked into 6144-byte records, which match the record length used if and when the files are eventually written to tape. Mimicking the tape structure on disk permits error recovery to be built directly into the raw data format. For all other types of products, if a tape I/O error occurs within the product's records on tape, then the entire product is lost. For raw archive this would be too great a penalty, since an entire volume scan is at stake. The blocking scheme permits partial error recovery while still maintaining a one-to-one mapping between disk and tape formats.

NOTE

Note: The records are for IRIS interpretation only and do not refer to any operating system file records.

The first 6144 byte record of a RAW product holds the `product_hdr` ([4.2.25 product_hdr Structure on page 46](#)) structure. Thus, the RAW product begins with a `product_hdr` like all the other types of products. The only difference is the zero padding out to 6144 bytes. The next record holds the `ingest_header` ([4.2.16 ingest_header Structure on page 39](#)) structure for the volume scan that supplied the data. Again, this structure is zero-padded to fill the entire second record. All subsequent records hold the actual data, and each record begins with the `raw_prod_bhdr` structure described in [4.3.29 1-byte Velocity Format \(DB_VEL\) on page 84](#).

Records hold data from one and only one sweep, and the sweep number is found in each record's header. The data for the sweep is the concatenation of all the data from as many records as pertain to that sweep. When data for a sweep ends short of the 6144 byte record size, the remainder of that record is padded with zeros. Each of these sweep data sets begins with the `ingest_data_header` ([4.2.15 ingest_data_header Structure on page 38](#)) structures for each data type that was recorded. The same number of headers are found in the beginning of a sweep's data as there are data types acquired during the sweep. The list of data types recorded is specified by the data collection mask in the `task_dsp_info` within the `task_configuration` within the `ingest_header`. The actual rays of data are found immediately after the headers.

Rays are ordered within a sweep by the same ordering sequence used for the ingest data file pointer table. If a ray is missing from the ingest file, a zero-length ray is inserted into the product file as a placeholder. Thus, the number of compressed rays in the file is equal to the product of the number

of data types recorded and the number of angles sampled. This is true even if data were not actually acquired at some of those angles. Within a ray, the recorded data are ordered by increasing data type number. See the `task_dsp_info` structure for a definition of the data type numbers. All data rays are compressed using the algorithm described in [4.5.4.1 Data Compression Algorithm on page 92](#). Note that the raw product headers and the ingest data headers are not compressed. The overall organization of the file is shown below. Raw product files are blocked into 6144-byte records, and all but the first two records begin with the `raw_prod_bhdr` 12-byte structure.

```
Record #1    { <product_hdr> 0,0,0... }
Record #2    { <ingest_header> 0,0,0... }
Record #3    { <raw_prod_bhdr> <ingest_data_header(s)> Data... }
Record #4    { <raw_prod_bhdr> Data... }
.            . .
.            . .
Record #N    { <raw_prod_bhdr> Data 0... }
Record #N+1  { <raw_prod_bhdr> <ingest_data_header(s)> Data... }
Record #N+2  { <raw_prod_bhdr> Data... }
.            . .
.            . .
Record #M    { <raw_prod_bhdr> Data 0... }
```

4.5.4.1 Data Compression Algorithm

To make the best use of storage, all radar rays are compressed before being inserted into the file. The compression algorithm is 16-bit word based, and simply removes runs of zeros. This complements the signal processor, which zeros data that does not meet the threshold requirements in effect. Runs of one or two zeros are not removed because there is no benefit. The data field starts with a compression code value. The code either indicates the number of zeros that were skipped, or the number of data words that follow. In the case of a zero skipped code, it is immediately followed with another code value. In the case of a data code, the next code follows the data.

Table 10 **Compression Code Meanings**

MSB	Low-bits	Meaning
0	0	<unused>
0	1	End of ray
0	2	<unused>

Table 10 Compression Code Meanings (Continued)

MSB	Low-bits	Meaning
0	3 – 32767	3 to 32767 zeros skipped
1	0	<unused>
1	1 – 32767	1 to 32767 data words follow

4.5.4.2 Raw Product Example

Here is an example of what the third record of a simple raw product would look like. This product is for data with only velocity recorded. Shown is the first ray of a PPI at azimuth 0 to 1 degree, elevation 0.5 degrees. It has 200 range bins, with no data for the first 100 bins, then one bin of zero velocity, then 99 bins with no data.

Table 11 Raw Product Example

Byte	Size	Contents	
0	12	<raw_prod_bhdr>	
12	76	<ingest_data_header>	
88	SINT2	-32762 (code for six data words to follow)	
90	BIN2	0 (starting azimuth)	Six
92	BIN2	91 (starting elevation)	words
94	BIN2	182 (ending azimuth)	referred
96	BIN2	91 (ending elevation)	to
98	SINT2	200 (number of range bins)	above
100	UINT2	3 (time)	
102	SINT2	50 (code for fifty zeros skipped)	
104	SINT2	-32767 (code for one data word follows)	
106	SINT2	128 (zero velocity value)	One data word
108	SINT2	49 (code for forty nine zeros skipped)	
110	SINT2	1 (code for end of ray)	
112			

(Continues on to next ray)

4.5.5 SLINE Product Format

The first 1024 bytes of the product hold a copy of the `protect_setup` structure described in [4.2.28 protect_setup Structure on page 49](#). This is followed by an array of `sline_results` structures one for each shearline found. The "Number of elements in product results array" element of the `product_end` structure in the `product_hdr` indicates the number of points in the array. Generally there is at most 1 shearline found.

4.5.6 TDWR Product Format

The first 1024 bytes of the product hold a copy of the `protect_setup` structure described in [4.2.28 protect_setup Structure on page 49](#). This structure is copied from the setup files on the integrating computer. This is followed by an array of `tdwr_results` structures, one for each corridor. The "Number of elements in product results array" element of the `product_end` structure in the `product_hdr` indicates the number of elements in the array. Only corridors covered by one of the input products to the integrator will be included, and corridors which are unused at generation time will normally be removed. This is followed by an array of `warning_results` structures copied from the WARN input product, if any. This is followed by an array of `sline_results` structures copied from the SLINE input product, if any. The sizes of these arrays are in the `tdwr_psi_struct` portion of the product header.

4.5.7 TRACK Product Format

The first 1024 bytes of the product hold a copy of the `protect_setup` structure described in [4.2.28 protect_setup Structure on page 49](#). This is followed by an array of `track_results` structures, one for each track point. The "Number of elements in product results array" element of the `product_end` structure in the `product_hdr` indicates the number of points in the array. Points must be in time order, with the oldest first. Within points of the same time, they are sorted by index number. Only one data point of each index value at each time is allowed, except that multiple text points are allowed (which have index set to zero).

4.5.8 VAD Product Format

The velocities produced by the VAD product are stored in an array of `vad_results` structures. This is essentially a 2D array of results. First included are all the results from the first elevation sweep, followed by the second, etc. You can use the structure `vad_product` defined in `product.h` to reference this data. Note that only data bins with valid velocities are included in the average. It is possible that the number of azimuths would be different for different elevations if this is a hybrid input.

4.5.9 VVP Product Format

The winds produced by the VVP product are stored in an array of `vvp_results` structures for each height. Note that if less than 30 range bins are found in the height interval, then the whole structure is zeroed. Therefore any analysis program should check the "Number of data points

used" field. Also if the calculation cannot be performed for some other reason, then the standard deviation will be set to 32767. Analysis programs should also check the standard deviations. Data fields which are turned off in the product configuration will be set to zero. Therefore the "Wind parameters mask" in the vvp_psi_struct in the product header needs to be checked also. Note that only data bins with valid velocities not near zero are included in the calculation. To compute the average reflectivity, bins must also have a valid reflectivity. Because of this, it can produce a lower number of valid bins, so the number of valid reflectivity bins is also recorded. Same for RhoHV.

4.5.10 WARN Product Format

The first 1024 bytes of the product hold a copy of the protect_setup structure described in [4.2.28 protect_setup Structure on page 49](#). This is followed by an array of warning_results structures. The number of warning_results structures in the array is given in the product_end structure, part of the product_hdr.

4.5.11 WIND Product Format

The first 84 bytes of the product hold a vvp_results structure for the whole volume. This is followed by an array of wind_results structures. The number of wind_results structures in the array is determined by multiplying the number of points in range by the number of points in azimuth stored in the product_specific_info structure in the header.

4.5.12 Product File Names

Because IRIS product files are computer accessed, actual file names are unimportant. IRIS can handle a product with any file name up to 23 characters. When IRIS creates a product file in its own product directory, it uses the following file name syntax. The two digit year used is simply the year modulo 100. Since the product file names are never parsed to generate a full date, there is no need to ever reconstruct the correct century.

General Format	Example
SSSYMMDDHHMMSS.PPPXXXX	SIG940620141921.TRAE090
Where:	
SSS—Three letter site code from Setup	SIG—Abbreviation for SIGMET
YYMMDDHHMMSS—Data time	940620141921—14:19:21 20 June 1994
PPP—Three-letter product type	TRA—Track product
XXXX—Characters for uniqueness	E090

When IRIS copies files to other directories using the network product output, it must generate a file name that is unique in the target directory. To see the choices available for this, see the SETUP/OUTPUT section of the *IRIS Utilities Manual*.

4.6 Tape Format

All tapes made by IRIS hold exact images of corresponding disk-based product files. The tapes are always written using fixed-length 6144-byte records, where the last tape record is padded with zeros, if necessary, to the full 6144-byte length. Products are separated on tape by end-of-file (EOF) marks. Prior to Version 5.00, tapes ended with a double EOF. Because all disk product files begin with a `product_hdr` structure, this is also the structure initially encountered in the first record of each product on tape. By examining the headers, you can determine what kinds of product files have been stored on the tape.

There is a special short record at the beginning of the tape that serves to identify how and when the tape was initially created by the `init_iris_tape` utility. This record contains the `tape_header_record` structure, and is followed by an EOF and the product files, if any. There are no special directory or inventory records on the tape. After a tape has been started, the only additional writing that can be done is to append more product file images to the end.

4.7 TIFF Output Format

IRIS can output images over the network in TIFF format. These files conform to the TIFF revision 6.0 standard. This standard supports many different types of images. Only a small subset of this is required. IRIS uses only baseline TIFF, and none of the TIFF extensions. Only one image is included in each file. It is a Palette Color image. The table below lists all the fields set by IRIS. The Image Description contains the 5 character product type, followed by the 12 character product configuration name. Compression is controlled by a setup question for the output device.

Table 12 **TIFF Fields Used by IRIS**

Software	305	"IRIS 5.56" for example
ImageDescription	270	"IRIS PPPPP NNNNNNNNNNNN"
DateTime	306	Time of ingest data
ImageWidth	256	Width of image in pixels
ImageLength	247	Height of image in pixels
Compression	259	Either none or PackBits
PlanarConfiguration	284	1 (Chunky)
SamplesPerPixel	277	1

Table 12 TIFF Fields Used by IRIS (Continued)

Orientation	274	1 (Top Left)
RowsPerStrip	278	Height of image in pixels

4.8 Constants

Table 13 Data Type Constants — /include/sigtypes.h

Extended Header is included here though it is not generated by the DSP. In general, types 0–31 could be produced by the DSP.		
DB_XHDR	(0)	Extended Headers
DB_DBT	(1)	Total power (1 byte)
DB_DBZ	(2)	Reflectivity (1 byte)
DB_VEL	(3)	Velocity (1 byte)
DB_WIDTH	(4)	Width (1 byte)
DB_ZDR	(5)	Differential reflectivity (1 byte)
DB_DBZC	(7)	Corrected reflectivity (1 byte)
DB_DBT2	(8)	Total power (2 byte)
DB_DBZ2	(9)	Reflectivity (2 byte)
DB_VEL2	(10)	Velocity (2 byte)
DB_WIDTH2	(11)	Width (2 byte)
DB_ZDR2	(12)	Differential reflectivity (2 byte)
DB_RAINRATE2	(13)	Rainfall rate (2 byte)
DB_KDP	(14)	KDP (Differential phase) (1 byte)
DB_KDP2	(15)	KDP (Differential phase) (2 byte)
DB_PHIDP	(16)	PhiDP (Differential phase) (1 byte)
DB_VELC	(17)	Corrected velocity (1 byte)
DB_SQI	(18)	SQI (1 byte)
DB_RHOHV	(19)	RhoHV (1 byte)
DB_RHOHV2	(20)	RhoHV (2 byte)
DB_DBZC2	(21)	Corrected Reflectivity (2 byte)
DB_VELC2	(22)	Corrected velocity (2 byte)
DB_SQI2	(23)	SQI (2 byte)
DB_PHIDP2	(24)	PhiDP (Differential phase) (2 byte)
DB_LDRH	(25)	LDR xmt H, rcv V (1 byte)
DB_LDRH2	(26)	LDR xmt H, rcv V (2 byte)
DB_LDRV	(27)	LDR xmt V, rcv H (1 byte)
DB_LDRV2	(28)	LDR xmt V, rcv H (2 byte)
...		
DB_HEIGHT	(32)	Height (1/10 km) (1 byte)
DB_VIL2	(33)	Linear liquid (.001mm) (2 byte)
DB_RAW	(34)	Raw Data
DB_SHEAR	(35)	Wind Shear (1 byte)
DB_DIVERGE2	(36)	Divergence (2 byte)
DB_FLIQUID2	(37)	Floated liquid (2 byte)
DB_USER	(38)	User type, unspecified data (1 byte)

Table 13 Data Type Constants — /include/sigtypes.h (Continued)

Extended Header is included here though it is not generated by the DSP. In general, types 0–31 could be produced by the DSP.		
DB_OTHER	(39)	Unspecified data, no color legend (1 byte)
DB_DEFORM2	(40)	Deformation (2 byte)
DB_VVEL2	(41)	Vertical velocity (2 byte)
DB_HVEL2	(42)	Horizontal velocity (2 byte)
DB_HDIR2	(43)	Horizontal wind direction (2 byte)
DB_AXDIL2	(44)	Axis of dilatation (2 byte)
DB_TIME2	(45)	Time in seconds (2 byte)
DB_RHOH	(46)	Rho, xmt H, rcv V (1 byte)
DB_RHOH2	(47)	Rho, xmt H, rcv V (2 byte)
DB_RHOV	(48)	Rho, xmt V, rcv H (1 byte)
DB_RHOV2	(49)	Rho, xmt V, rcv H (2 byte)
DB_PHIH	(50)	Phi, xmt H, rcv V (1 byte)
DB_PHIH2	(51)	Phi, xmt H, rcv V (2 byte)
DB_PHIV	(52)	Phi, xmt V, rcv H (1 byte)
DB_PHIV2	(53)	Phi, xmt V, rcv H (2 byte)
DB_USER2	(54)	User type, unspecified data (2 byte)
DB_HCLASS	(55)	Hydrometeor class (1 byte)
DB_HCLASS2	(56)	Hydrometeor class (2 byte)
DB_ZDRC	(57)	Corrected differential reflectivity (1 byte)
DB_ZDRC2	(58)	Corrected differential reflectivity (2 byte)

CHAPTER 5

UTILITIES

5.1 Productx

The **productx** utility, or "product examiner," displays the information contained in a specified product file.

For all product types, **productx** displays the product header, including information such as the site where the ingest data came from, the date and time when the ingest data was gathered, and its size. This is followed by product-specific meta-data from the header for many product types. Finally it displays the data values from the file. Note that for Cartesian data files, this could be a large number of pixels. In this case it skips data to present a summary display which fits on the terminal. If you want no skipping, then specify a very large terminal width, say 10000.

5.1.1 Invoking Productx

Command

```
productx [options] filename
```

filename is the name of a product file stored in `/usr/iris_data/product`.

Raw products are stored in a separate directory — `/usr/iris_data/product_raw`.

Options

<code>-help</code>	Prints the list of options.
<code><filename></code>	Specify which ingest header file to read.
<code>-sweep:#</code>	Specify sweep number, origin 1. Z-axis for 3D data.

-version Print out just the version number.
-width=# Specify maximum line with for data display.

Each product file has a unique name based on the site ID, date, and a randomize algorithm. The first three letters of the file extension state the kind of product stored in the file:

Abbreviation	Product Type	Abbreviation	Product Type
BEA	BEAM product	SLI	SLINE product
CAP	CAPPI product	STA	STAT product
FCA	FCAST product	TDW	TDWR product
IMG	IMAGE product	TOP	TOPS product
MAX	MAX product	TRA	TRACK product
OTH	OTHER product	TXT	TEXT product
PPI	PPI product	USE	USER product
RAW	RAW product	VUS	VUSER product
RN1	RAIN1 product	VIL	VIL product
RNN	RAINN product	VVP	VVP product
RHI	RHI product	WND	WIND product
RTI	RTI product	WRN	WARN product
SHE	Shear product	XSE	XSECT product

5.1.2 Productx Examples

A Status product shows the status of the IRIS processes at a particular site:

```
$ productx HOT000602152122.STAZ0GN
----- Product Summary for HOT000602152122.STAZ0GN -----
Ingest site name : 'SIGMET, HOT', Version: 7.17
Ingest hardware name : 'SIGMET, HOT'
Product site name : 'SIGMET, HOT', Version: 7.17
File size: 2340 bytes (Disk space: 2340 bytes)
Product type is: Status
PCO name: SIGMET, HOT, TCO name: FAULT
PRF: 500Hz, Wavelength: 5.00cm, Nyquist: 6.25m/s(V), 6.25m/s(W)
Polarization: Horizontal, wind:???
Heights: Radar: 600m, Ground: 100m, Melting: ???m MSL
Size is: 0x0x0 pixels
Center Location: 42_33.0'N, 71_25.8'W, ref: 600 meters
Projection type is: Azimuthal Equidistant
```

Projection Reference Point: 42_33.0'N, 71_25.8'W
Radar position is: 0.0, 0.0 pixels
Scale is: 0.000 x 0.000 x 0.000 km/pixel
Product data type is Xhdr (0)
Maximum range: 0.0 km
Ingest time: 15:21:22 2 JUN 2000 UTC (0 minutes west) DST:0/1
Volume scan time: 15:21:22 2 JUN 2000 (LT: EDT 300 minutes)
Oldest Ing time: 15:21:22 2 JUN 2000
Product Gen time: 15:21:22 2 JUN 2000
Input count: 1
Product is not composited
Site style is: RADAR
Overall status is: FAULT
Status of IRIS_INGEST ON/Idle
Status of IRIS_INGFIO ON/NA
Status of IRIS_OUTPUT ON/NA
Status of IRIS_PRODUCT ON/Idle
Status of IRIS_WATCHDOG ON/Running
Status of IRIS_REINGEST ON/Idle
Status of IRIS_NETWORK ON/Idle
Status of IRIS_NORDRAD OFF/Stopped
Status of IRIS_SERVER ON/Idle
Status of IRIS_RIBBON OFF/Stopped
Status of IRIS_INPUT ON/Idle
Status of DSP N/A Stopped
Status of RCP Critical N/A DEAD
Status of WINDOW1 OK Idle test
Status of NETWORK1 OK Idle ToArchive2
Status of NETWORK2 OK Idle netcdf_out
Status of NETWORK3 OK Idle ToBufr
Status of NETWORK4 OK Idle ToHDF5
Status of WINDOW2 OK Idle Joe
Status of ARCHIVE1 OK Idle archive
RST mode: 'DEFAULT'

```
TSC mode: 'DEFAULT'
PSC mode: 'DEFAULT'
POM mode: 'DEFAULT'
Active task: ''
Active product: ''
Antenna Position, azimuth: 20.00, elevation: -0.99
Bite fault summary shows 2
Low Airflow: OK
Interlock: OK
Waveguide: OK
Top message #9, Repeats: 1
Problem starting scan at EL=6 (AZ velocity out of range)
Process: IRIS_INGEST, Name: F:202 M:3
Time: 16:34:36 30 MAY 2000
Message list contains 0 messages:
```

A PPI product shows useful header information:

```
$ productx TMS090520180345.PPIFD6W -width:80
----- Product Summary for TMS090520180345.PPIFD6W -----
Ingest site name : 'tms_rad1', Version: 8.11
Ingest hardware name : 'tms_rad1'
Product site name : 'SIGMET, dry2', Version: 8.12
File size: 519040 bytes (Disk space: 519040 bytes)
Product type is: PPI
PCO name: DEF_DBZ, TCO name: PPIVOL_B
PRF: 840/560Hz, Wavelength: 10.63cm, Nyquist: 44.65m/s(V), 22.32m/s(W)
XMT Polarization: Horizontal, Wind:???
Constant: 72.06 dB, IO: -104.82 dBm, Cal Noise: -70.66 dBm, Bandwidth: 0 kHz.
ZFlags: SP_T, block_zc, attn_zc, target_zc, dpatten_zc, dpatten_z
VFlags: SP_V, 3lag_w, ship_v, unfold_vc, FALL_VC, storm_vc
Heights: Radar: 970m, Ground: 948m, Melting: 4900m MSL
Size is: 720x720x1 pixels
Scale is: 500.00 x 500.00 x 0.00 m/pixel
Center Location: 22 24.7'N, 114 7.4'E, ref: 0 meters
Projection type is: Eqdist Cylinder
```



```

Projection Reference Point: 22 24.7'N, 114 7.4'E
Equitorial Radius: 0.00000 km, Flattening: 1/0.00000
Radar position is: 360.0, 360.0 pixels
Product data type is dBZ (2)
Color count:16, Color set: 1, variable
Seams: 16452.00 16702.00 16952.00 17252.00 17402.00 17502.00 17752.00
18052.00
18202.00 18302.00 18552.00 18852.00 19002.00 19102.00 19352.00 19652.00
Maximum range: 180.0 km

PPI elevation angle: 6.60 degrees

Ingest time: 18:03:45 20 MAY 2009 HKT (-480 minutes west) DST:0/0
Volume scan time: 18:03:45 20 MAY 2009 HKT (LT: HKT -480 minutes)
Oldest Ing time: 18:03:45 20 MAY 2009 LT
Product Gen time: 19:23:31 17 JUN 2009 UTC

Input count: 1
Product is not composited.
Displaying cartesian data with skip factor 40

```

```

719:255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
679:255 255 255 255 255 255 0 0 0 0 0 0 0 255 255 255 255 255
639:255 255 255 255 0 0 0 0 0 0 0 0 0 0 0 255 255 255
599:255 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255
559:255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255
519:255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255
479:255 0 0 0 0 0 0 0 105 117 104 66 0 0 0 0 0 0
439:255 0 0 0 0 0 82 85 121 57 0 101 81 0 0 0 0 0
399:255 0 0 0 0 0 83 111 136 78 0 0 0 0 0 0 0 0
359:255 0 0 0 0 0 83 169 71 0 0 0 0 0 0 0 0 0
319:255 0 0 0 0 0 133 87 0 0 0 0 0 0 0 0 0 0
279:255 0 0 0 0 0 97 93 0 0 0 0 0 0 0 0 0 0
239:255 0 0 0 0 0 78 77 64 0 0 0 0 0 0 0 0 0
199:255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255
159:255 255 0 0 0 0 0 0 0 0 0 255 0 0 0 0 0 255
119:255 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255
79: 255 255 255 255 0 0 0 0 0 0 0 255 0 0 255 255 255
39: 255 255 255 255 255 255 0 0 0 0 0 0 255 255 255 255 255

```

5.2 Rays

The **rays** utility gives information about ingest files. You can choose to display information about various parameters of the ingest data.

5.2.1 Invoking Rays

Command

```
rays [options] filename
```

`filename` is the name of an ingest header file stored in the directory `/usr/iris_data/ingest`. Ingest files are named for the date and time when the data were gathered. For example, ingest data gathered at 10:17:30 on December 2, 1994 would be stored in a file named `941202101730`.

Options

<code>-help</code>	Prints the list of options.
<code>-data:dtype</code>	Specify which data type to display.
<code>[-if:]filename</code>	Specify which ingest header file to read.
<code>-inter</code>	Run in interactive mode. This is the old style, rays will then prompt for some of the options.
<code>-perf</code>	Performance test, display ray headers only.
<code>-range:#</code>	Specify starting range in km.
<code>-sweep:#</code>	Specify sweep number, origin 1.
<code>-terse</code>	Skip showing ingest header info.
<code>-width:#</code>	Specify maximum line width for data display.

To make it easier to enter the names of the ingest files, change your default directory to the ingest directory and get a listing of all the header files (with `"ls *."`). Then select the name of a file with the mouse and paste it into the command line.

5.2.2 Rays Examples

5.2.2.1 Headers only Example

In the following example, **rays** uses the **-perf** option to display the ray header information only. This format lets you evaluate the speed of IRIS or the signal processor and look for missing rays. Note that there are two

azimuth and elevation angles recorded with each ray. These are angles at the beginning and ending of each ray.

```
$ rays -perf -terse DRY020418160516. | more
```

```
Reading file: /usr/iris_data/ingest/DRY020418160516.01dBT
```

```
# 0 Az:      359.80,0.99   El:    0.48,  0.48   Size: 967   16:05:33
# 1 Missing
# 2 Az:      0.99,  2.20   El:    0.48,  0.48   Size: 967   16:05:33
# 3 Az:      2.20,  3.38   El:    0.48,  0.48   Size: 967   16:05:33
# 4 Az:      3.38,  4.59   El:    0.48,  0.48   Size: 967   16:05:33
# 5 Az:      4.59,  5.78   El:    0.48,  0.48   Size: 967   16:05:34
# 6 Az:      5.78,  6.99   El:    0.48,  0.48   Size: 967   16:05:34
# 7 Missing
# 8 Az:      6.99,  8.17   El:    0.48,  0.48   Size: 967   16:05:34
# 9 Az:      9.38,  9.38   El:    0.48,  0.48   Size: 967   16:05:34
# 10 Az:     9.38, 10.57   El:    0.48,  0.48   Size: 967   16:05:34
# 11 Az:     10.57, 11.78  El:    0.48,  0.48   Size: 967   16:05:34
# 12 Az:     11.78, 12.96  El:    0.48,  0.48   Size: 967   16:05:34
# 13 Missing
# 14 Az:     14.17, 14.17  El:    0.48,  0.48   Size: 967   16:05:34
# 15 Az:     14.17, 15.36  El:    0.48,  0.48   Size: 967   16:05:34
# 16 Az:     15.36, 16.57  El:    0.48,  0.48   Size: 967   16:05:34
# 17 Az:     16.57, 17.75  El:    0.48,  0.48   Size: 967   16:05:34
# 18 Az:     17.75, 18.96  El:    0.48,  0.48   Size: 967   16:05:34
# 19 Az:     18.96, 18.96  El:    0.48,  0.48   Size: 967   16:05:34
# 20 Az:     18.96, 20.15  El:    0.48,  0.48   Size: 967   16:05:34
# 21 Az:     20.15, 21.36  El:    0.48,  0.48   Size: 967   16:05:34
# 22 Az:     21.36, 22.54  El:    0.48,  0.48   Size: 967   16:05:34
# 23 Az:     22.54, 23.75  El:    0.48,  0.48   Size: 967   16:05:34
# 24 Az:     23.75, 24.94  El:    0.48,  0.48   Size: 967   16:05:34
```

5.2.2.2 Velocity Example

You can choose any of the data parameters — V in this example — and rays then displays the header, followed by range bins starting from the specified bin.

```
$ rays XXX041029121855. -data:v2 | more
[joe@localhost ingest]$ rays COX071115233217. -data:v | more
Task Summary for: COX071115233217.
Site name: 'cox-radar', Task name: 'PPI_C'
Scan: PPI, Speed: 9.00 deg/sec, Resolution:1.00 deg
Description: 'Doppler Velocity Volume Scan'
Location: 21 26.0'N 91 58.6'E, Altitude: 64 meters, Melting height:Unknown
Dpolapp config:
Volume Time: 23:32:17.442 15 NOV 2007 UTC (0 min. west) (LT: BDT -360 min.)
ZFlags: SP_T, block_zc, attn_zc, target_zc, dpatten_zc, dpatten_z
VFlags: SP_V, 3lag_w, ship_v, unfold_vc, fall_vc, storm_vc
PRF: 720/576Hz, PulseWidth: 1.00 usec (1)
BeamWidth: 1.46/1.50 deg.
Radar constant: 0.00/0.00 dB, Receiver bandwidth 0 kHz.
Calibration I0: -111.95/-113.03 dBm, with noise -79.70/-76.95 dBm.
LOG-Noise: 0.1776, Lin-Noise: 0.1776, I-Off: 0.0000, Q-Off: 0.0000
SOPRM Flags: 0x04af, LOG Slope: 0.480, Z-Cal: -36.81dBZ, H/V: 0.00 dB
Filters: Dop:6, Log:0; PntClt: 3, Thresh: 1.0 dB; Samples: 80
Processing Mode: FFT, Xmt Phase: Fixed
Zdr Threshold: LOG GDR = 0.00 dB, XDR = 0.00 dB
T Threshold: LOG LOG = 2.4 dB
Z Threshold: LOG & CSR SIG = 5.0 dB
V Threshold: SQI & CSR CSR = 15.0 dB
W Threshold: SIG & SQI & LOG SQI = 0.42
Available moments are: dBZ V
Original moments were: dBT dBZ V W
Starting range 0.125 km, range bin spacing 625 meters
There are 8 sweeps, each having 360 rays and 320 bins
Angle list: 0.0 1.5 3.0 4.5 6.0 9.0 12.0 15.0
Reading file: /usr/iris_data/ingest/COX071115233217.01V
Sweep Time: 23:32:17.442 15 NOV 2007
Starting at range 0.12 km (bin 1), bin step: 0.62 km
```

```
# 0 Az: 359.52, 0.47 El: 0.02, 0.02 Size: 320 23:32:32
--.- --.- --.- --.- --.- --.- --.- 17.3 17.3 17.3 17.3 --.- --.- --.-
# 1 Az: 0.47, 1.51 El: 0.02, 0.02 Size: 320 23:32:32
--.- --.- --.- -61.5 -61.5 -62.1 -62.1 -62.7 59.7 59.1 --.- --.- --.- --.-
# 2 Az: 1.53, 2.52 El: 0.02, 0.02 Size: 320 23:32:32
--.- --.- --.- --.- --.- --.- --.- --.- --.- --.- --.- --.- --.-
# 3 Az: 2.54, 3.50 El: 0.02, 0.02 Size: 320 23:32:32
--.- --.- --.- --.- --.- --.- --.- --.- --.- 34.6 -38.8 -1.8 --.- --.-
# 4 Az: 3.52, 4.53 El: 0.02, 0.02 Size: 320 23:32:33
--.- --.- -31.0 -31.0 -31.0 -31.0 -31.0 -30.4 -30.4 -30.4 56.1 -1.8 -1.8 -
1.8
--More--
```

Where no data is available for a ray, it displays "Missing." When no data is available for a range bin within the ray, it displays a series of dashes.

5.2.2.3 Extended Header Example

On shipboard systems, additional housekeeping information can be stored for each ray in what is called an "extended header". If your ingest file has this, you can display it with the option "-data:xhdr". Notice that the extended header includes time recorded to the nearest millisecond.

```
$ rays -data:xhdr -terse -if:DRY020418160516. | more
Reading file: /usr/iris_data/ingest/DRY020418041055.01Xhdr
Starting at range 0.00 km (bin 1), bin step: 0.30 km
# 0 Az: 359.80, 0.55 El: 0.42, 0.42 Size: 1 4:11:08.055
Az: 350.95 El: 0.40 Pitch: -0.13 Roll: -1.10 Head: 346.79
Vel: 18.72 deg/s 0.04 0.70 359.82 359.82

Tr: 2.29 El_or: 0.31 Lat: 1#section#45.5'N Long:138#section# 2.8'E Alt: 14
Cor: 3.59 Age: 291 Vel: 3.69 m/s 0.00 -0.17
```

Here are details, see also the `extended_header_v1` structure in the *IRIS Programmer's Manual*.

```
Az : 350.95 El: 0.40 Pitch: -0.13 Roll: -1.10 Head: 346.79

Vel : 18.72 deg/s 0.04 0.70 359.82 359.82
```

This shows the azimuth and elevation of the antenna, as well as the pitch, roll, and heading of the platform. Also shown are the derivatives of those five numbers in degrees/second. The extended header information is recorded from a serial data stream transmitted from the RCP. Typically this is configured to transmit updates at the maximum possible speed of about

20 times per second. Because rays can be recorded at up to 40 rays per second, and because of pipeline delays in the serial data, it is possible for the extended header azimuth to lag the actual azimuth by up to several degrees. Normally the platform motion period is far slower.

```
Tr: 2.29 El_or: 0.31 Lat: 1#section#45.5'N Long:138#section#  
2.8'E Alt: 14
```

```
Cor: 3.59 Age: 291 Vel: 3.69 m/s 0.00 -0.17
```

"Tr" is the training angle, which is the pedestal relative azimuth of the pedestal. Similarly "El_or" is the pedestal relative elevation angle. "Cor" is the velocity correction in meters/second applied to the velocity data to correct for platform motion. "Age" is the time in milliseconds since this update arrived from the RCP. Finally the position and motion of the platform is recorded. Altitude is in meters, and motions are in meters/second.

APPENDIX A

RADAR CONTROL PROTOCOL

The interface to the RCP is typically via network UDP multicast packets. Alternatively we can use a serial line, or a UNIX FIFO. When using the network, all transmissions both directions are sent to the same address and port number. This allows the seamless merging of data from multiple clients. It also means that there is no longer a separate wire for transmit and receive as with a serial cable. The data format consists of exactly the serial format with a 16-byte prefix. The prefix consists of the packet size in ASCII (8-bytes), followed by a ASCII code for the type of packet. The first letter of the type is either "T" or "R" for transmit or receive. This is relative to the main controlling host. If coding this up, you will need to filter on the direction.

For coding examples, please look in the `utils/antx.C` program, as well as in the `libs/antenna/ant_netrcv.c` and `libs/user/UdpSupport.c` files. The only coding differences for multicast packets over broadcast packets are that you should specify the interface to use, and issue the `IP_ADD_MEMBERSHIP` call. Also, you should read a standard network textbook on multicast addressing. There are certain reserved addresses. We recommend using *site-local* addresses.

When using a two-way asynchronous RS-232 data line, the baud rate is typically 9600 (19200+ for shipboard systems). Through this link, IRIS controls the servo and antenna and receives feedback status. Information is transferred in packets consisting of two or more bytes. Each packet begins with a SYNC byte and ends with an END byte of FF(Hex). All SYNC bytes have the MSB set, and the particular value indicates the type of packet to follow. Types currently available are 80(hex) for antenna, C0(hex) for BITE, AF(hex) for Q-BITE, and B0(hex) for time. The packet layouts are shown below. Each type of packet has a specific direction of travel (to or from IRIS), but packets can arrive in any order within the serial stream.

Several types of antenna communication formats are supported. Older systems use the RCV01 and XMT01 formats but the newer systems can use the RCV02 and the XMT02 formats. The RCV03 format is intended for systems on moving platforms, such as ships or airplanes. One of the challenges of these systems is to correct the radar's measured radial velocity for the motion of the platform. To make this correction, the three-dimensional velocity and orientation of the platform must be recorded. Typically, the information comes from an inertial navigation system. For shipboard system, an update rate of approximately 20 reports per second can satisfy the velocity correction requirements at 19200 baud.

The following angles, with the exception of the latitude and the longitude, are transmitted as 14-bit binary angles. The latitude and longitude are both 21-bit binary angles.

- azimuth and elevation
- train order
- pitch, roll, and heading

In the XMT01 format, the angular speed is a signed number in units of 0.55°/sec. In all other formats, the angular rates are in signed 14-bit binary angles per second. Therefore, the largest possible value is 180°/sec (30 rpm) and the step is 0.022°/sec. All velocities are in signed cm/sec with the altitude in signed meters. If some of the information is not available at the full resolution of the data format, the low bits are filled with zeros.

The azimuth and the elevation angles are corrected angles relative to the north and are the angles that the antenna is pointed relative to the deck of the platform. These calculations are derivable from the other angles but are also reported to assist in the data analysis, especially if one of the sensors or the stabilization fails.

The pitch is the angle between the fore-and-aft axis of the platform and the horizontal is measured in the vertical plane. The pitch is positive when the bow is down and the roll is the rotation angle about the fore-and-aft axis in its pitched position. The pitch is measured in the plane perpendicular to the fore-and-aft axis, which is generally not the vertical plane, and the roll is positive when the deck is down on the port side.

NOTE

Note: The pitch can be directly measured by a level on the fore-and-aft axis but the roll cannot be directly measured by a one-axis tilt meter.

The heading is referred to as the direction the platform is pointed but is not the same as direction of motion. The platform could be pointed one way and drifting backwards.

The time stamp is a 14-bit counter incremented by the RCP once per millisecond. The RCP should latch all the data for a packet at the same time. This counter allows the host computer to accurately judge the time between samples without the serial line latencies and fluctuations due to the time sharing operating system.

The position of the platform is reported by the latitude, the longitude, and the altitude. Since the altitude may not be implemented for systems on ships, the setting will be zero.

Table 14 Status Packet RCV01 Format (RCP to Host)

Char	Function
1	SYNC Byte (80 Hex)
2	Azimuth Low 7 bits
3	Azimuth High 7 bits
4	Elevation Low 7 bits
5	Elevation High 7 bits
6	Status #1 D6 = Low air flow D5 = Low Waveguide pressure D4 = Servo power D3 = Antenna Local mode D2 = Interlock D1 = Standby D0 = Radiate On
7	Status #2 D6 = RCP02 is shutdown D5 = LSB pulse width D4 = T/R power On D3 = T/R Local mode D2 = Encoders calibrated D1 = MSB pulse width D0 = Magnetron current normal
8	End Of Message (FF Hex)

Table 15 Control Packet XMT01 Format (Host to RCP)

Char	Function
1	SYNC Byte (80 Hex)
2	Azimuth Low 7 bits
3	Azimuth High 7 bits
4	Elevation Low 7 bits
5	Elevation High 7 bits

Table 15 Control Packet XMT01 Format (Host to RCP)

Char	Function
6	Control Word #1 D6 = MSB of Pulse Width D5 = Leave Pulse width unchanged D4 = Spare D3 = Signal Generator On D2 = Signal Generator CW D1 = EL (1 = Scan, 0 = Position) D0 = AZ (1 = Scan, 0 = Position)
7	Control Word #2 D6 = Reset RCP02 on edge D5 = Noise Source On D4 = LSB of Pulse width D3 = Radiate On complemented D2 = Radiate On D1 = Servo Power On D0 = T/R Power On
8	Control Word #3 (all spare)
9	Signal generator level (unsigned 0–127dB attenuation)
10	AZ/EL Antenna speed (signed 7 bit, 0.55 degree resolution)
11	END OF MESSAGE (FF Hex)

Table 16 Status Packet RCV02 / RCV04 Format (RCP to Host)

Char	Function
1	SYNC Byte (80 Hex)
2	Azimuth Low 7 bits
3	Azimuth High 7 bits
4	Elevation Low 7 bits
5	Elevation High 7 bits
6	Azimuth Rate Low 7 bits
7	Azimuth Rate High 7 bits
8	Elevation Rate Low 7 bits
9	Elevation Rate High 7 bits
10	Status #1 D6 = Low air flow D5 = Low Waveguide pressure D4 = Servo Power D3 = Antenna Local mode D2 = Interlock Open D1 = Standby D0 = Radiate On

Table 16 Status Packet RCV02 / RCV04 Format (RCP to Host)

Char	Function
11	Status #2 D6 = RCP02 is shutdown D5 = LSB pulse width D4 = T/R Power On D3 = T/R Local mode D2 = Azimuth encoder calibrated D1 = MSB pulse width D0 = Mag. current normal
12	Status #3 D6 = IRIS Mode 2 D5 = IRIS Mode 1 D4 = IRIS Mode 0 D3 = Elevation encoder calibrated D2 = Signal Generator fault D1 = Signal Generator On D0 = Signal Generator CW
13	Signal generator level (0=max power)
14	Time Stamp Low 7 bits
15	Time Stamp High 7 bits
16	END OF MESSAGE (FF Hex)

Table 17 Control Packet XMT02 / XMT04 Format (Host to RCP)

Char	Function
1	SYNC Byte (80 Hex)
2	Azimuth Low 7 bits
3	Azimuth High 7 bits
4	Elevation Low 7 bits
5	Elevation High 7 bits
6	Control Word #1 D6 = MSB of Pulse Width D5 = Leave Pulse Width unchanged D4 = Spare D3 = Signal Generator On D2 = Signal Generator CW D1 = EL (1 = Scan 0 = Position) D0 = AZ (1 = Scan 0 = Position) Note: If EL in position mode, maximum positive velocity is sent in the velocity field, if EL in scan mode, maximum or minimum elevation position is sent in the position field. If AZ in position mode, maximum positive velocity is sent in the velocity field, if AZ in scan mode, 0 is sent in the position field.

Table 17 Control Packet XMT02 / XMT04 Format (Host to RCP) (Continued)

Char	Function
7	Control Word #2 D6 = Reset RCP02 on rising edge D5 = Noise Source On D4 = LSB of Pulse width D3 = Radiate On complemented D2 = Radiate On D1 = Servo Power On D0 = T/R Power On
8	Control Word #3 D6 = IRIS Mode 2 D5 = IRIS Mode 1 D4 = IRIS Mode 0 D3 = Radar Workstation A okay D2 = Radar Workstation B okay D1 = Data Processor A okay D0 = Data Processor B okay
9	Signal Generator level (0–127 dB attenuation)
10	AZ Antenna Speed Low 7 bits
11	AZ Antenna Speed High 7 bits
12	EL Antenna Speed Low 7 bits
13	EL Antenna Speed High 7 bits
14	END OF MESSAGE (FF Hex)

Table 18 Status Packet RCV03 Format (RCP to Host)

Char	Function
1	SYNC Byte (80 Hex)
2	Identification byte
3	Azimuth Low 7 bits (Earth relative)
4	Azimuth High 7 bits
5	Elevation Low 7 bits (Earth relative)
6	Elevation High 7 bits
7	Train Order Low 7 bits (azimuth of pedestal relative to the ship)
8	Train Order High 7 bits
9	Elevation Order Low 7 bits (elevation of pedestal relative to the ship)
10	Elevation Order High 7 bits
11	Pitch Low 7 bits
12	Pitch High 7 bits
13	Roll Low 7 bits
14	Roll High 7 bits
15	Heading Low 7 bits
16	Heading High 7 bits
17	Azimuth Rate Low 7 bits
18	Azimuth Rate High 7 bits

Table 18 Status Packet RCV03 Format (RCP to Host)

Char	Function
19	Elevation Rate Low 7 bits
20	Elevation Rate High 7 bits
21	Pitch Rate Low 7 bits (LSB = Zero)
22	Pitch Rate High 7 bits
23	Roll Rate Low 7 bits (LSB = Invalid Roll)
24	Roll Rate High 7 bits
25	Heading Rate Low 7 bits (LSB = Invalid Heading)
26	Heading Rate High 7 bits
27	Status #1 D6 = Low air flow D5 = Low Waveguide pressure D4 = Servo power D3 = Antenna Local mode D2 = Interlock open D1 = Standby D0 = Radiate ON
28	Status #2 D6 = RCP02 is shutdown D5 = LSB pulse width D4 = T/R Power on D3 = T/R Local mode D2 = Azimuth encoder calibrated D1 = MSB pulse width D0 = Mag. current normal
29	Status #3 D6 = Reserved D5 = Reserved D4 = Reserved D3 = Elevation encoder calibrated D2 = Signal Generator fault D1 = Signal Generator On D0 = Signal Generator CW
30	Signal generator value (0=full signal)
31	Time Stamp Low 7 bits
32	Time Stamp High 7 bits
33	Latitude Low 7 bits
34	Latitude Middle 7 bits
35	Latitude High 7 bits
36	Longitude Low 7 bits
37	Longitude Middle 7 bits
38	Longitude High 7 bits
39	Altitude Low 7 bits
40	Altitude High 7 bits
41	Velocity East Low 7 bits (LSB = Invalid Lat/Lon)
42	Velocity East High 7 bits
43	Velocity North Low 7 bits (LSB = Zero)
44	Velocity North High 7 bits

Table 18 Status Packet RCV03 Format (RCP to Host)

Char	Function
45	Velocity Up Low 7 bits (LSB = Invalid Altitude)
46	Velocity Up High 7 bits
47	END OF MESSAGE (FF Hex)

Table 19 Status Packet RCV05 Format (RCP to Host)

Char	Function
1–15	These bytes exactly match the RCV02 / RCV04 format
16	Dual-System Status D6 = RCP02 is configured as a Dual-System D5 = Dual-System Mode MSB D4 = Dual-System Mode LSB D3 = This packet was sent from Unit "A" D2 = Information is known about the "Other" unit D1 = Unit "A" is the preferred system D0 = Unit "B" is disabled Note: The 2-bit Dual-System Mode codes are: 00 : Unknown mode 01 : System "A" 10: System "B" 11 : Auto Switch
17	Dual-System Status D6 = Unit "B" is okay D5 = Unit "B" Activity Code MSB D4 = Unit "B" Activity Code LSB D3 = Unit "A" is disabled D2 = Unit "A" is okay D1 = Unit "A" Activity Code MSB D0 = Unit "A" Activity Code LSB Note: The 2-bit Dual-System Activity codes are: 00 : Inactive 01 : Warmup 10: Active Now 11 : Reserved
18	Dual-System Status D6 = RCP02 is configured for voluntary flipping D5 = Unit "B" is offering to give up control D4 = Unit "A" is offering to give up control D3 = Unit "B" would be used if it were available D2 = Unit "A" would be used if it were available
19	D0:2 = Current Polarization XMT control 0=Horizontal; 1=Vertical; 2=Alternating; 3=Simultaneous D3 = Polarization switch is OK to run
20	Spare
21	Spare
22	Spare
23	Spare
24	END OF MESSAGE (FF Hex)

Table 20 Control Packet XMT05 Format (Host to RCP)

Char	Function
1–13	These bytes exactly match the XMT02 / XMT04 format

Table 20 Control Packet XMT05 Format (Host to RCP)

Char	Function
14	Control Word #4 D6 = Dual-System: Mode MSB D5 = Dual-System: Mode LSB D4 = Dual-System: Offer to relinquish control D3 = Dual-System: Unit would be used if available D2 = Spare D1 = Spare D0 = Spare Note: The 2-bit Dual-System Mode codes are: 00 : No mode 01 : System "A" 10: System "B" 11 : Auto Switch
15	D0:2 = Requested Polarization XMT control 0=Horizontal; 1=Vertical; 2=Alternating; 3=Simultaneous; 7=Unchanged
16	Spare
17	Spare
18	END OF MESSAGE (FF Hex)

Table 21 Time Packet (RCP to Host)

Char	Function
1	SYNC Byte (B0 Hex)
2	Year Low 7 bits
3	Year High 7 bits
4	Month
5	Day
6	Hour
7	Minute
8	Second
9	1/100 of Second
10	Status
11	END OF MESSAGE (FF Hex)

Table 22 Generic BITE Status Packet (Both ways)

Char	Function
1	SYNC Byte (C0 Hex)
2	BITE Unit ID byte (selectable in the range 00–7F Hex)
3	Status byte #1
4	Status byte #2
.	
.	
.	
N–1	Status byte #N–3
N	END OF MESSAGE (FF Hex)

The BITE status packet consists of from 3 to 20 bytes. The first two and the last bytes are used for identification purposes. The middle bytes must have the MSB set to zero and can contain arbitrary status in the lower 7 bits. Typically this is used to report back individual bits containing the results of tests such as cabinet interlocks, airflow sensors, and power supply checks. This report should be sent by the BITE every time the status changes. It should also send a report in response to the interrogate command. IRIS sends the interrogate command every 60 seconds.

Table 23 BITE Command Packet (Both ways)

Char	Function
1	SYNC Byte (C0 Hex)
2	Command (0x4D = Interrogate, 0x44=Sample Data, 0x43=Reset)
3	End Of Message (FF Hex)

Table 24 Auxiliary Control BITE Packets (Both ways)

Char	Function							
1	SYNC Byte (C0 Hex)							
2	BITE Unit ID byte (User Choice)							
3	Control/Status Bits	6	5	4	3	2	1	0
4	Control/Status Bits	13	12	11	10	9	8	7
5	Control/Status Bits	20	19	18	17	16	15	14
6	Control/Status Bits	27	26	25	24	23	22	21
7	Control/Status Bits	34	33	32	31	30	29	28
8	Control/Status Bits	41	40	39	38	37	36	35
9	Control/Status Bits	48	47	46	45	44	43	42
10	Control/Status Bits	55	54	53	52	51	50	49
11	Control/Status Bits	62	61	60	59	58	57	56
12	Control/Status Bit							63
13	END OF MESSAGE (FF Hex)							

This example BITE packet is supported by the SIGMET RCP02 and RCP8 Radar Control Processors, and is an example of a mapping of arbitrary control and status bits into a BITE packet.

These RCPs contains 64 auxiliary status and control variables, labeled S[0:63] and C[0:63]. These bits may be sent to and from the host computer in the form of 13-byte BITE packets holding the full set of 64 bits. The format of these packets is the same in both directions, and the identification byte is selectable so that conflicts with other BITE packets can be avoided.

Table 25 Q-BITE Status Packet (Both ways)

Char	Function
1	SYNC Byte (AF Hex)
2	BITE Unit ID byte (selectable in the range 00-7F Hex)

Table 25 Q-BITE Status Packet (Both ways) (Continued)

Char	Function
3	Status byte #1
4	Status byte #2
.	
.	
.	
N-1	Status byte #N-3
N	END OF MESSAGE (FF Hex)

The Q-BITE (Quantitative byte) status packets consists of from 3 to 128 bytes. The first two and last bytes are used for identification purposes. The middle bytes must have the MSB set to zero and can contain an arbitrary value in the lower 7 bits. Typically this is used to report back voltage/ power levels. This report should not be sent by the BITE every time the status changes. This report is sent in response to the Q-bite interrogate command. IRIS sends the interrogate command every 60 seconds.

The Q-BITE data stream consists of a series of integer values. Each value is packed into a series of 7-bit characters, using between 1 and 5 depending on the desired resolution. The low bits come first, and IRIS supports up to 32 bits per value. IRIS can easily be configured to display any such values with appropriate units and scaling.

Table 26 Simple Q-BITE Example

Char	Function
1	SYNC Byte (AF Hex)
2	BITE Unit ID byte (Selectable in range 00-7F)
3	V1 Trigger frequency (Hz) (low bits)
4	V1 Trigger frequency (Hz) (High bits)
5	V2 Thyatron voltage (0.1 Volts) (low bits)
6	V2 Thyatron voltage (0.1 Volts) (high bits)
7	END OF MESSAGE (FF Hex)

Table 27 Q-BITE Interrogate Packet (Both ways)

Char	Function
1	SYNC Byte (90 Hex)
2	Command (0x01 or 0x4D=Interrogate, 0x44=Sample Data, 0x43=Reset)
3	End Of Message (FF Hex)

Table 28 BITE Individual Command Packet (Host to RCP)

Char	Function
1	SYNC Byte (C1 Hex)

Table 28 BITE Individual Command Packet (Host to RCP)

Char	Function
2	ID of the BITE unit for which the command will be applied
3	Command: 0x4D=Interrogate, 0x44=Sample Data, 0x43=Reset
4	END OF MESSAGE (FF Hex)

The BITE individual command packet is used to request information about a single BITE unit, separate from all the others.

The RCP should respond to an **Interrogate** packet by sending the current version of the specified BITE status packet. The RCP should respond to a **Sample Data** packet by sending requests out the the remote device to get information, then responding to the host computer with the new BITE status packet when the information arrives. The RCP should responds to the **Reset** packet by sending a reset command to the remote device.

Table 29 Chat-Mode Packet (Both ways)

Char	Function
1	SYNC Byte (F1 Hex)
2...7	7-Bit ASCII characters (possibly NULL terminated)
8	END OF MESSAGE (FF Hex)

These packets are sent in both directions to convey serial TTY communication. Up to six 7-bit characters can be sent in each packet with two characters of overhead for SYNC and END. This allows up to 75% of the available serial bandwidth to be used for chatting. If a "chat" mode packet contains fewer than six characters, then a NULL (zero byte) is inserted after the last one.

APPENDIX B

LINK TRANSMISSION FORMATS

B.1 AWS (Austrian Weather Service) Format

This format uses an 8-bit data stream and can transmit an image of and rectangular product up to 256 pixels high with compression for horizontal runs of the same data. For flexibility, IRIS supports a mapping table to convert the IRIS product color levels to the transmitted levels. Note the expanded color scales with more than 16 colors are not available for link outputs. Throughout this discussion, hexadecimal notation is often used, and those numbers are denoted with a suffix of "h." For example, 12h means 12 base 16, or 18 base 10. Also for this discussion, all line and pixel numbers are origin 1. The upper left pixel is line 1, pixel 1.

All picture transmissions begin with a station ID code, followed by a time code, followed by the data section, followed by the end-of-image code. The data section consists of a series of line number codes, followed by compressed data. Data for a line ends when the next line number code is encountered. All unfilled pixels are assumed to be of the value 0. If a line is repeated in the data stream, any new data overwrites old data. If a line number code is never transmitted, the line is assumed to be filled with zeros.

The meaning of a byte is generally found by looking at the upper 4-bits. If those bits are a value between 0h and Eh, insert the data value in the low 4-bits between 1 and 15 times. If those bits are an Fh, it can be interpreted as one of the following special commands.

FFh — Line number follows in the next byte

The following byte contains one less than the line number for data that follows. For example, FFh 0Ch means line 13 follows. This command also implies that the previous line is zeroed from the current position to the end.

FEh — Extended data repeat command, information in next two bytes

This command is the same as the normal data repeat command, except that the repeat count is 12 bits. The high 8 bits of the repeat count are in the following byte, and the low 4 bits are in the high 4 bits of the third byte. The data value is in the low 4 bits of the third byte. For example, FEh 12h 34h means insert the data value 4 292 times.

FBh — Station identifier follows in next byte

The station identifier is configured from the **setup** utility. If multiple FBh commands are received for a given image, the last one takes effect. There must be at least one FBh command in each image. This command allows data from multiple sites to be merged on one transmission line. IRIS transmits only host site data.

FAh — Date and time follows in next 18 bytes

This command specifies the time of the data in the current image. If multiple FAh commands are received for a given image, the last one takes effect. There must be at least one FAh command in each image. The 18-byte string contains a text version of the time, such as "10-MAY-91 15:45:00." There is one space between the year and hours, and all numeric fields are two digits, with leading zeros, if required. The month contains the first three letters of the English month name.

F8h — End-of-Image

This indicates that an image has been completely sent. Any future transmissions relate to another image. After an F8h command and before an FFh command, only FBh, FAh commands are processed. All other commands are ignored.

B.2 HKO (HongKong Observatory) Format

This format uses an 8-bit data stream and can transmit an image of and rectangular product up to 255 pixels high with compression for horizontal runs of the same data. For flexibility, IRIS supports a mapping table to convert the IRIS product color levels to the transmitted levels. Note the expanded color scales with more than 16 colors are not available for link outputs. Throughout this discussion, hexadecimal notation is often used, and those numbers are denoted with a suffix of "h." For example, 12h means 12 base 16, or 18 base 10. Also for this discussion, all line and pixel numbers are origin 1. The upper left pixel is line 1, pixel 1.

All picture transmissions begin with a picture type ID code, followed by a header code, followed by the data section, followed by the end-of-image code. The data section consists of a series of line number codes, each followed by compressed data. Data for a line ends when the next line

number code is encountered. All unfilled pixels are assumed to be of the value 1. If a line is repeated in the data stream, any new data overwrites old data. If a line number code is never transmitted, the line is assumed to be filled with ones.

The only use of a zero byte is to immediately proceed on the the commands below. This make it easy to sync up to the data stream.

00h 01h — Picture type follows in the next byte

Table 30 HKO Picture types

1	All PPIs			
2	CAPPI	>2.5 km height	<96 km range	data not velocity
3	CAPPI	all heights	<96 km range	data is velocity
4	CAPPI	<2.5 km height	<96 km range	data not velocity
5	CAPPI	>2.5 km height	96<range<192	data not velocity
6	CAPPI	all heights	96<range<192	data is velocity
7	CAPPI	<2.5 km height	96<range<192	data not velocity
8	CAPPI	all heights	>192 km range	data not velocity
9	CAPPI	all heights	>192 km range	data is velocity
10	All other products			

00h 02h — Header follows in the next 40 bytes

The format of the header is "HHPPPPPPPPPPPhh:mm DD-MM-YYYY". Where "HH" is the CAPPI height in km, "PPPPPPPPPPPP" is the product name. All numeric fields are two digits, with leading zeros, if required.

00h 03h — Line number follows in the next byte

The following byte contains the origin 1 line number in the range 1–255. This is followed by a variable length string of bytes containing a data number followed by a repeat count. Data numbers are in the range 1 through 16.

00h 04h — End-of-Image

This indicates that an image has been completely sent. Any future transmissions relate to another image.

APPENDIX C

UF FORMAT

C.1 Introduction

UF (short for "Universal Format") is a radar data format originally proposed and documented in the "Report on a Meeting to Establish a Common Doppler Radar Data Exchange Format", page 1401 of the November 1980 *Bulletin of the American Meteorological Society*.

C.2 Single UF Ray Structure

The data consists of a single file for a complete volume scan. Within this file are a series of stand-alone rays (data acquired for a given pointing direction). All header information is duplicated for each ray. Within a ray, the data is basically all organized as 16-bit words, byte swapped in the big endian convention. The exception to this is that each ray starts and ends with a 32-bit record size indicating the number of bytes in the ray. All ASCII text is supposed to be left justified, space padded but some converter programs produce null terminated text, so the reader must be tollerant. The data format does support breaking a large ray into several records. In this case, the multiple records within the ray will have identical formats, they will have different field headers and data fields. The IRIS convertor programs do not support this feature, and will always place a ray in a single record. The optional header is only placed in the ray for the first ray of a file.

Please look in our uf.h header file for additional documentation. These structure names generally end with a "2" to indicate that they are 2-byte aligned. The default is that all IRIS structures are 4-byte alligned unless they end in a 2 or 8.

ray size 4 Bytes <uf_mandatory_header2> 45 Words

```

<uf_optional_header> 0 or 12 Words
<uf_data_header2> 3 + 2N Words
<uf_field_header2> #1 19 or 21 or 25 Words
data from field #1 M Words
<uf_field_header2>#2 19 or 21 or 25 Words
data from field #2 M Words
...(repeats for each data type)...
ray size 4 Bytes

```

C.3 uf_mandatory_header2 Structure

Source: uf.h

Byte	Size	Contents
0	char[2]	Text "UF"
2	int16_t	Record Size in 16-bit words
4	int16_t	Offset to start of optional header, origin 1
6	int16_t	Local-Use Header Position (origin 1)
8	int16_t	Data Header Position (origin 1)
10	int16_t	Record Number (origin 1)
12	int16_t	Volume number on tape, n/a for disk
14	int16_t	Ray number within the volume scan
16	int16_t	Record number within ray (origin 1)
18	int16_t	Sweep number within the volume scan
20	char[8]	Radar name
28	char[8]	site name
36	int16_t	Latitude degrees (North positive, South negative)
38	int16_t	Latitude minutes
40	int16_t	Latitude seconds*64
42	int16_t	Longitude degrees (East positive, West negative)
44	int16_t	Longitude Minutes
46	int16_t	Longitude Seconds
48	int16_t	Height of antenna above sea level in meters
50	int16_t	Year (time of data acquisition)
52	int16_t	Month
54	int16_t	Day
56	int16_t	Hour
58	int16_t	Minute
60	int16_t	Second
62	char[2]	Time zone, "UT" for universal
64	int16_t	Azimuth (degrees*64) of midpoint of sample
66	int16_t	Elevation (degrees*64)

68	int16_t	Sweep mode: 0:Cal 1:PPI 2:Coplane 3:RHI 4:Vertical 5:Target 6:Manual 7:Idle
70	int16_t	Fixed angle (degrees*64)
72	int16_t	Sweep rate ((degrees/second)*64)
74	int16_t	Year (generation data of UF format)
76	int16_t	Month
78	int16_t	Day
80	char[8]	Name of UF generator program
88	int16_t	Value stored for deleted or missing data (0x8000)

C.4 uf_optional_header Structure

Source: uf.h

Byte	Size	Contents
0	char[8]	sProjectName[8]
8	int16_t	iBaselineAzimuth
10	int16_t	iBaselineelevation
12	int16_t	iVolumeScanHour /* Time of start of current volume scan */
14	int16_t	iVolumeScanMinute
18	char[8]	sFieldTapeName[8]
24	int16_t	iFlag

C.5 uf_data_header2 Structure

Source: uf.h

Byte	Size	Contents
0	int16_t	Number of fields in this ray
2	int16_t	Number of records in this ray
4	int16_t	Number of fields in this record
6	int16_t	Data type of field #1 (SIGMET standard): VR:velocity SW:spectrum width DR:ZDR CZ:Corrected dBZ DZ:Total dBZ RH:RhoHV PH:PhiDP KD:KDP LH:LdrH LV:LdrV
8	int16_t	Field #1 field header position

10	int16_t	Data type of field #2
12	int16_t	Field #2 field header position
...		

C.6 uf_field_header2 Structure

Source: uf.h

Byte	Size	Contents
0	int16_t	Data offset from start of record, origin 1
2	int16_t	Scale factor, met units = file value/scale
4	int16_t	Start range km
6	int16_t	Start range meters
8	int16_t	Bin spacing in meters
10	int16_t	Bin count
12	int16_t	Pulse width in meters
14	int16_t	Horizontal beam width in degrees*64
16	int16_t	Vertical beam width in degrees*64
18	int16_t	Receiver bandwidth in Mhz*64 ?
20	int16_t	Polarization: 1:horz 2:vert 3:circular 4:ellip.
22	int16_t	Wave length in cm*64
24	int16_t	Sample size
26	char[2]	Type of data used to threshold
28	int16_t	Threshold value
30	int16_t	Scale
32	char[2]	EditCode
34	int16_t	PRT in microseconds
36	int16_t	Bits per bin, must be 16
38	12	<uf_fsi2>

C.7 uf_fsi2 Structure

Source: uf.h

Byte	Size	Contents
If velocity data:		
0	int16_t	Nyquist velocity
2	int16_t	<spare>
If DM data:		
0	int16_t	Radar Constant

2	int16_t	Noise Power
4	int16_t	Receiver Gain
6	int16_t	Peak Power
8	int16_t	Antenna Gain
10	int16_t	Pulse Duration (microseconds*64)

If Other data:

nothing

APPENDIX D

RTD FORMATS

D.1 Introduction

IRIS can output a real-time display data stream using UDP data packets. There are several formats provided, and the customer can also write their own. The source code for these transmitter programs is in `{IRIS_ROOT}/libs/rtq`. The formats are documented in detail in the `sig_rtdisp.h` file.

Any computer running IRIS on that network can receive and process (display) the real time display data. One nice factor about the real time display is that it is completely scalable. If the IRIS Radar server is sending real time display data, the load induced on the network, and the servers CPU is constant no matter if none, one or a hundred machines are on that network receiving and processing the real time display data. This is made possible by the fact that the real time display is done in broadcast mode.

It should also be noted that UDP data does not get retransmitted upon packet loss, and at the application level in IRIS, there is no attempt done at detected packet losses nor asking for retransmissions. If such a loss occurs at the transmitter, on the network, or at one or more receiving hosts, then a gap would occur in the picture being drawn on the destination real time displays. This of course differs from IRIS product transmission which are done point-to-point via TCP and are reliable due to error checking and retransmissions.

The load induced on the network by a real time display server broadcasting is a function of the following: Number of moments being broadcast (up to 3 – Z, V and W), the number of bins per radial and the scan rate of the antenna. The total number of bins (combined for all moments) is approximately 1400 per radial. An overhead of about 100 bytes per ray exists. Each bin requires one byte of data. Thus this a scan rate of 5 RPM (30 radials per second), requires about $1500 * 30 = 45000$ bytes per second. One must remember that the real time display bandwidth must be added to

any other IRIS and other bandwidth being occupied on your network. Therefore, running the real time display on a remote radar connected via a 56K baud link would not be practical.

The real time display data stream can be configured to transmit using up to 6 different formats, IP addresses, and ports. See the **setup** utility documentation in the *SIGMET Utilities Manual*. Normally gateways, such as multi-homed hosts, bridges and routers will NOT pass IRIS real time display data. This is because such devices are built to filter broadcast data. This is the mechanism that keeps the real time display data on a single network. However, it is usually possible to configure such devices to pass broadcast data that is occurring on a specific port (like the real time display port). This may be convenient in some installations.

To receive real time display data, the receive process needs to issue a socket and a bind system call. Following this, the receiver can read messages from the socket as they are transmitted.

D.2 Rtd_v1_xmt

There are three different messages transmitted as part of the real time display data stream. The messages are distinguished by the the first two bytes (SHORT) in the message summarized in the following table (see definitions in sig_rtdisp.h):

RTRAY_TYPE_HEADER

RTRAY_TYPE_RAY

RTRAY_TYPE_RAY

The message RTRAY_TYPE_HEADER consists of the two byte ID followed by a struct rtd_v1_vol_header. This message is sent either at the beginning of a new elevation sweep, or periodically if a new sweep has not started recently.

The message RTRAY_TYPE_RAY begins with the two byte ID. What this is followed by depends on the type of data being transmitted by the real time display sender. This is chosen in the senders **setup** utility and can consists of 1, 2 or 3 data types being Z, V or W. The volume header indicates which data types are currently being transmitted and in which order the data types are being presented. For example, if only Z and V are being transmitted, the RTRAY_TYPE_RAY message will consist of its two byte message ID, followed by a struct rtd_v1_ray_header, followed by the data elements for Z, followed by the V elements. The Z and V elements are presented as one byte per range bin. The scaling of the elements is defined in [4.3 Data Types on page 71](#). For example, if the volume header

indicates that 200 bins of each type are being transmitted, then the Z elements (and the V elements) would each be 200 bytes long each, with each byte representing one range bin. The spacing between the range bins is also defined by the volume header.

D.3 Rtd_v2_xmt

This format is designed to support the full number of range bins, almost the full number of data types, and both 8-bit and 16-bit data formats. As a result each radial must be split into potentially many individual 1500 byte UDP packets. Similar to rtd_v1_xmt, the volume header rtd_v2_vol_header is sent periodically. Every 25 rays, or when there is a change. Otherwise the rtd_v2_ray_header is sent followed by data.

D.4 Rtd_nids3_xmt

This format is designed to support a legacy data format used by Radtec. All packets are the same format, with minimal header information. Data can be either 8-bit or 4-bit format. There is a config file rtd_nids3_xmt.conf which controls some details.

INDEX

A

Antenna library, subdirectory	14
AWS product transmission format	121
Axis of Dillitation, 2-byte data format	72

B

beam_psi_struct structure	30
BITE command packet	118
BITE individual command	119
BITE packet	117

C

cappi_psi_struct structure	31
Cartesian product, file format	90
Catch_psi_struct structure	31
Catch_results structure	31
Chat-mode packet	120
color_scale_def structure	32
Config library, subdirectory	14
cross_psi_struct structure	34

D

data compression algorithm	92
data format	29
structure	29
tape	96
data type, constants	97
data types	71
DB_AXDIL2 constant	98
DB_CDBZ constant	97
DB_CDBZ2 constant	97
DB_DEFORM2 constant	98
DB_DIVERGE2 constant	97
DB_FLIQUID2 constant	97
DB_HDIR2 constant	98
DB_HEIGHT constant	97
DB_HVEL2 constant	98
DB_OTHER constant	98
DB_RAW constant	97
DB_SHEAR constant	97
DB_UDBZ constant	97
DB_UDBZ2 constant	97

DB_USER constant	97
DB_VEL constant	97
DB_VEL2 constant	97
DB_VIL2 constant	97
DB_VVEL2 constant	98
DB_WIDTH constant	97
DB_WIDTH2 constant	97
DB_XHDR constant	97
DB_ZDR constant	97
DB_ZDR2 constant	97
Deformation, data format,	73
Divergence, data format	73
DSP library, subdirectory	14
dsp_data_mask structure	34

E

Echo Tops, data format	77
extended_header format, ingest file	72
extended_header_v0 structure	34, 35
in ingest file data format	72
extended_header_v1 structure	35
in ingest file data format	72

F

FCAST product, file format	90
fcast_psi_struct structure	36
floating liquid format, ingest file, 2-byte	74

G

Gage_psi_struct structure	36
Gage_results structure	36

H

HKO picture types	123
HKO product transmission format	122
horizontal product type	23
Horizontal wind direction, 2-byte data format	77
HydroClass, data format	74

I

Ingest file	
-------------	--

data format	89	product_end structure	44
naming convention	89	product_hdr structure	46
ingest_configuration structure	37	product_specific_info structure	48
ingest_data_header structure	38	Productx utility, source code	14
ingest_header structure	39	protect_setup structure	49
		in SLINE product file	93
		in TRACK product file	93
		in WARN product file	94
K		Q	
KDP format	78	Q-BITE interrogate packet	119
KDP, 2-byte data format	79	Q-BITE packet	118
L		R	
LDR		radar control processor, protocol	109
1-byte data format	79	rain_psi_struct structure	49
2-byte data format	79	rainfall rate format	81
legend, for user-generated products	23	RAW product, file format	89
llwas_psi_struct structure	65	example	93
M		raw_prod_bhdr structure	49
Makefile	16	raw_psi_struct structure	50
max_psi_struct structure	39	ray_header structure	50
N		Rays utility, source code	14
NDOP product, file format	90	RCV01 communication format	110
ndop_input structure	40	RCV01 serial format	111
ndop_psi_struct structure	40	RCV02 communication format	110
Ndop_results structure	40	RCV02 serial format	112
NORDRAD_AREAS.DAT	25	RCV03 communication format	110
NORDRAD, area definition file	25	RCV03 serial format	114
O		reflectivity	
one_protected_region structure	41	1-byte data format	71
P		2-byte data format	71
Phi		rhi_psi_struct structure	51
1-byte data format	80	Rho	
2-byte data format	80	1-byte data format	82
PHIdp		2-byte data format	82
1-byte data format	80	RhoHV	
2-byte data format	80	1-byte data format	82
Pipes	19	2-byte data format	82
ppi_psi_struct structure	41	RTD formats	131
Product		rti_psi_struct structure	51
examiner	99	S	
file abbreviation	100	Serial control formats	
Product file		RCV01 format	111
data format	89	RCV02 format	112
naming convention	95	RCV03 format	114
product types	23	XMT01 format	111
product_configuration structure	41	XMT02 format	113
		shear_psi_struct structure	51
		SLINE product, file format	93
		sline_psi_struct structure	52

sline_results structure	52	uf_data_header structure	127
in SLINE product file	93	uf_field_header structure	128
SQI		uf_field_specific_info structure	128
1-byte data format	82	uf_mandatory_header structure	126
2-byte data format	82	uf_optional_header structure	127
sri_psi_struct structure	54	User library, subdirectory	14
status_antenna_info structure	55	User Product Insert (UPI)	20
status_device_info structure	55		
status_message_info structure	55	V	
status_misc_info structure	56	VAD product, file format	94
status_one_device structure	57	vad_psi_struct structure	67
status_one_process structure	57	vad_results structure	68
status_process_info structure	57	Velocity	
status_results structure	58	1-byte data format	84
structure definitions	30	1-byte unfolded format	84
structure_header structure	58	2-byte data format	84
		2-byte unfolded format	84
T		vertical product type	23
tape_header_record structure	58	VIL, ingest file format, 2-byte	86
in tape format	96	Virtual TV library, subdirectory	14
tape, file format	96	VVP product, file format	94
task_calib_info structure	59	vvp_psi_struct structure	68
task_configuration structure	60	vvp_results structure	68
task_dsp_info structure	60		
task_dsp_mode_batch structure	61	W	
task_end_info structure	61	WARN product, file format	95
task_file_scan_info structure	62	warn_psi_struct structure	69
task_manual_scan_info structure	62	warning_results structure	70, 95
task_misc_info structure	62	Width	
task_ppi_scan_info structure	63	1-byte data format	86
task_range_info structure	63	2-byte data format	86
task_rhi_scan_info structure	64	WIND product, file format	95
task_scan_info structure	64	Wind shear, data format	83
task_sched_info structure	64	wind_psi_struct structure	70
TDWR product, file format	94	wind_results structure	71, 95
tdwr_results structure	65		
in TDWR file format	94	X	
text_results structure	66	XMT01 communication format	110
TIFF		XMT01 serial format	111
fields used by IRIS	96	XMT02 communication format	110
file format	96	XMT02 serial format	113
time packet	117		
Time, 2-byte data format	84	Y	
top_psi_struct structure	66	ymds_time structure	71
TRACK product, file format	94		
track_psi_struct structure	66	Z	
track_results structure	67	ZDR	
in track file format	94	2-byte data format	87
TV subroutine library, subdirectory	14	ingest file format, 1-byte	87
U			
UF format	125		



www.vaisala.com

