M212604EN-D

User Guide

Digital Receiver and Signal Processor

RVP10

VAISALA

PUBLISHED BY

Vaisala Oyj
Vanha Nurmijärventie 21, FI-01670 Vantaa, Finland
P.O. Box 26, FI-00421 Helsinki, Finland
+358 9 8949 1
vaisala.com
docs.vaisala.com

Table of contents

List of figures

List of tables

1. About this document

1.1 Version information

This manual provides technical information for installing, operating, and maintaining RVP10 Digital Receiver and Signal Processor.

Table 1 Document versions

Document code	Date	Description
M212604EN-D	February 2025	Updated for IRIS version 10.2. Updates include: • In Defining IFDR10 input A/D saturation levels (page 71), corrected the signal power limits to: -3 dBm ... +4 dBm. • Changing the **`Current transmit sequence`** in **dspx** now also changes the external transmitter pulse width. • Wide dynamic range functionality added. • Max. number of range bins is now up to 15,000. • Notification conventions updated according to the ANSI Z535.6 standard. See Documentation conventions (page 18).
M212604EN-C	August 2024	Third version. • For RVP10 release 2.0 • With IRIS version 10.1.0
M212604EN-B	May 2023	Second version. Updated information about data transfer method (TCP/IP instead of UDP).
M212604EN-A	April 2023	First version of this document.

1.2 Related documents

Table 2 Related documents (English)

Document code	Name
DOC236879	*IRIS RDA Release Notes*
M212924EN	*IRIS and RDA Software Installation Guide*
M212925EN	*IRIS and RDA Utilities Guide*
M212926EN	*IRIS Radar User Guide*
M212927EN	*IRIS Programming Guide*
M212928EN	*IRIS Product and Display Guide*
M212923EN	*Radar Control Processor RCP8 User Guide*
M212604EN	*RVP10 Digital Receiver and Signal Processor User Guide*

1.3 Documentation conventions

NOTICE! Notice alerts you to a situation that, if not avoided, could result in the product to be damaged or important data to be lost.

Highlights important information on using the product.

Gives tips for using the product more efficiently.

1.4 Trademarks

Vaisala® is a registered trademark and IRIS™ and Sigmet™ are trademarks of Vaisala Oyj.

All other product or company names that may be mentioned in this publication are trade names, trademarks, or registered trademarks of their respective owners.

2. Product overview

2.1 Introduction to RVP10

RVP10 is the next generation of dedicated weather radar signal processors developed by Vaisala and the legacy Sigmet over 40 years of history.

The RVP10 generation brings many needed hardware features to make use of the emergence of the solid-state power amplifier (SSPA) evolution that is about to take over transmitter technology. These features include:

- Dedicated support of multiple transmitters within a single radar system
- Generation and streaming of multiple channels of IQ data over network
- Multiple waveform definitions used in parallel or in sequence transmitted over more than one carrier frequency having pulse compression, phase and polarity modulation
- Dynamic and adaptive receive filter coefficients for the generation of IQ
- Increased stability of time synchronization resulting in more accurate data tagging of antenna location during transmission and improvements in phase coherency
- More linear ZDR measurement over the entire dynamic range when compared to RVP900

Table 3 History of the Vaisala signal processor

Year	Model	Units sold	Major technical milestones
1981	FFT	10	First commercial FFT-based Doppler signal processor for weather radar applications, featured simultaneous Doppler and Intensity processing.
1985	RVP5	161	First single-board low-cost Doppler signal processor. First commercial application of dual-PRF velocity unfolding algorithm.
1986	PP02	12	First high-performance commercial pulse-pair processor with 18.75 m bin spacing and 1024 bins.
1992	RVP6	150	First commercial floating-point DSP-chip based processor. First commercial processor to implement selectable pulse pair, FFT or random phase 2nd trip echo filtering.
1996	RVP7	>200	First commercial processor to implement fully digital IF processing for weather radar. First use of multiple channels for high dynamic range applications.
2003	RVP8	>500	First digital receiver/signal processor implemented using an open hardware/software architecture with public APIs on a standard PC under the Linux operating system. First use of NLFM pulse compression.
2007	RVP900	>1200	Longest deployed weather radar signal processor on the market with zero hardware revisions needed since its first release 15 years ago.

Year	Model	Units sold	Major technical milestones
2022	RVP10		The first public and secured APIs to Transmitter waveform and IQ generator. The first commercial use of multiprocessor system on a chip (MPSoC) running embedded Linux application and FPGA.

2.2 RVP10 product architecture

RVP10 consists of two hardware units:

- IFDR10 Intermediate frequency digital receiver
- RVP10SRV Signal processor computer

These devices interface via 1-Gigabit and 10-Gigabit Ethernet and communicate via gRPC framework over the network layer.

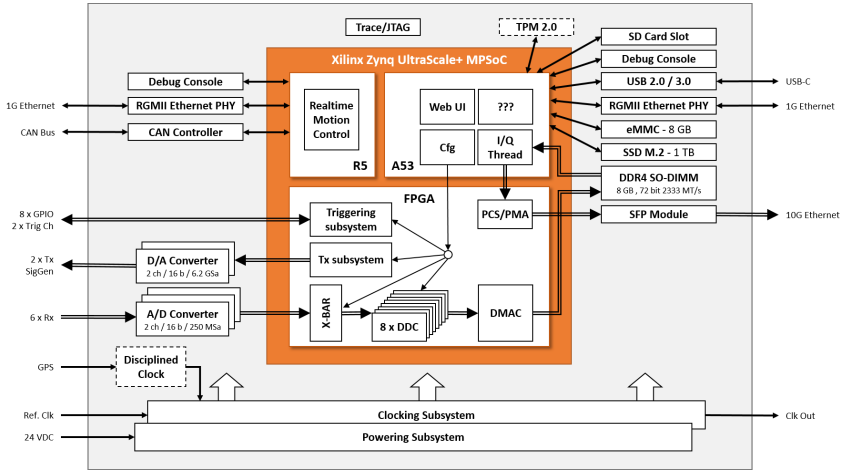


Figure 1 RVP10 architecture

IFDR10 intermediate frequency digital receiver

IFDR10 intermediate frequency digital receiver (IFDR) provides receive, transmit, and IF detector functionality in a single, compact network-attached FGPA-based product able to perform tens of billions of multiply accumulate cycles per second.

The IFDR samples up to 4 receive channels and 2 burst channels of IF inputs and computes `I` and `Q` data from them. The `I` and `Q` data is transmitted over a 10-Gigabit Ethernet connection to the RVP10SRV signal processor for further processing into moments.

RVP10SRV signal processor computer

The RVP10SRV signal processor is a server class computer with a dual-processor multicore Xeon processor running Linux.

RVP10SRV, running RDA software, computes the radar data moments from the **I** and **Q** data provided by IFDR10. The moments include **Z**,**T**,**V**, and **W**. The moments can be distributed internally on RVP10SRV, or externally to other computers running IRIS or third-party software.

2.2.1 Data types

Table 4 Data types

Data type	Description
Ah, Av	Integral attenuation for horizontal (H) and vertical (V) channels
Azdr	Integral attenuation of ZDR (dB) format
CSR	Doppler channel clutter-to-signal (CSR) ratio of dBT to -dBZ
dBZ dBZv dBZe	Clutter-corrected reflectivity Vertical (HV enhanced) reflectivity Total reflectivity from the vertical polarization channel (**ZV**) and combination of the horizontal and vertical channel (**ZE**).Corrected for ground clutter.
dBT dBTv dBTe	Uncorrected reflectivity Total vertical (HV enhanced) reflectivity Total reflectivity from the vertical polarization channel (TV) and combination of the horizontal and vertical channel (TE)
HCLASS	Hydrometeor classification Estimated hydrometeor type in the precipitation area
KDP	Specific differential phase An indicator of the rate of change of the phase difference between horizontally and vertically polarized pulses of the radar. A greater horizontal shift results in a positive KDP value, and a greater vertical shift results in a negative KDP value. Typical cause for a high KDP area is heavy rain.
LDRH, LDRV	Linear depolarization ratio H to V (or V to H) The ratio of cross-polar to co-polar reflectivity measured in dB
LOG	Log receiver signal-to-noise ratio
PHIH, PHIV	Horizontal (V) or vertical (V) differential phase Phase difference for the total round trip between radar and the volume where the signal is reflected. **PHIH** is measured between HH and HV channels. **PHIV** is measured between VV and VH channels.

Data type	Description
PHIDP	Differential phase The phase difference due to propagation between the HH and VV channels of the radar.
PMI	Polarimetric meteo index
R	Rate of accumulation of precipitation in units of mm/hour For snow, this usually refers to the liquid equivalent.
RHOHV, RHOH, RHOV	Correlation coefficient between HH and VV (or HH & HV / VV & VH) channels Higher (>0.95) values indicate uniform precipitation areas, and lower values indicate more mixed hydrometeor types, such as melting snow, wet snowflakes, or airborne debris.
SNR	Signal-to-noise ratio Generic measurement of signal-noise ratio in dB
SQI	Signal quality index A value between 0-1 that measures the signal's Doppler coherency, that is, the correlation between the signal and its Doppler lag: • 0 indicates white noise • 1 is the perfect Doppler point target
TV, TE	Total vertical (HV enhanced) reflectivity Total reflectivity from the vertical polarization channel (TV) and combination of the horizontal and vertical channel (TE)
V	Velocity Average radial velocity (towards or away from the radar) of detected hydrometeor areas
VC	Corrected velocity Same as velocity V, but corrected for effects of range folding and velocity folding
V: SHEAR, Vc: SHEAR	Velocity and corrected velocity of wind shear
W	Spectral width Variability of Doppler velocity values within the measurement area
XCOR	Polar cross-correlation, uncorrected rhohv Because this value is not noise corrected, this is a direct indicator of the PHIDP uncertainty.
ZV, ZE	Vertical (HV enhanced) reflectivity Total reflectivity from the vertical polarization channel (ZV) and combination of the horizontal and vertical channel (ZE). Corrected for ground clutter.

Data type	Description
ZC	Corrected reflectivity Same as reflectivity **Z**, but corrected for attenuation and beam blockage effects
ZDR	Differential reflectivity The ratio of SNR in the horizontal channel to the SNR in the vertical channel. Positive values indicate more prominent horizontal echoes and negative values more prominent vertical echoes. Larger hydrometeor sizes are usually identified by high positive ZDR values.
ZDRC	Corrected differential reflectivity Same as differential reflectivity **ZDR**, but corrected for attenuation and beam blockage effects

2.3 Safety

WARNING! Electricity hazard
Turn off to the server computer and disconnect the mains power before installing or removing PCI boards or otherwise opening the server chassis.

The surface of IFDR10 may be hot.

CAUTION! Follow local regulations in the installation and usage of the device.

NOTICE! Never open the IFDR10 unit. Thermal conductive material does not bind properly when unit is reassembled. **Opening IFDR10 voids all warranties.**

NOTICE! Install IFDR in a suitable enclosure where it is protected from dust and humidity. Ensure the sufficient airflow.

NOTICE! Use ESD protection. See ESD protection (page 24).

NOTICE! Do not modify the unit. Improper modification can damage the product or lead to malfunction.

CLASS 1 LASER PRODUCT
The product includes SFP+ optical transceivers. They are Class 1 Laser Products and comply with US FDA regulations. These products are certified by TÜV and CSA to meet the Class 1 eye safety requirements of EN (IEC) 60825 and the electrical safety requirements of EN (IEC) 60950.
The laser rays are invisible to the human eye.
Even though the optical transceivers are eye-safe, do not look directly into the laser source.

2.3.1 ESD protection

Electrostatic discharge (ESD) can damage electronic circuits. Vaisala products are adequately protected against ESD for their intended use. However, it is possible to damage the product by delivering electrostatic discharges when touching, removing, or inserting any objects in the equipment housing.

To avoid delivering high static voltages to the product:

- Handle ESD-sensitive components on a properly grounded and protected ESD workbench or by grounding yourself to the equipment chassis with a wrist strap and a resistive connection cord.
- If you are unable to take either precaution, touch a conductive part of the equipment chassis with your other hand before touching ESD-sensitive components.
- Hold component boards by the edges and avoid touching component contacts.

2.4 Product and part names

Table 5 RVP10 product and part names

Code	Description
IFDR10	RVP intermediate frequency digital receiver (IFDR)
RVP10SRV	Server computer, RVP10 signal processor
260799WR	Power supply for IFDR10
248382SP	Power supply for RVP10SRV
RVP8-IO	IO card
RCP8-CP	IO panel

Code	Description
223150	Bandpass filter 30 MHz
223151	Bandpass filter 60 MHz
223152	Bandpass filter 57.5 MHz

3. Functional description

3.1 IFDR10 IF digital receiver - IQ generation

IFDR10 is a network-attached FGPA-based product that provides the receive functionality needed in a weather radar signal processor. From the sampled IF input it generates `I` and `Q` data that is sent to the RVP10SRV signal processor computer over standard 10-Gigabit Ethernet. IFDR10 uses TCP/IP for communication and data transfer with RVP10SRV signal processor computer.

3.1.1 Digital receiver function

The signal processor performs the following functions:

- Filtering (smoothing) the IF signal using one of the two types of FIR filters: the legacy bandpass filter where the user sets the filter length and bandwidth, or the matched filter which is a conjugated version of the transmit waveform. These include range gating and option coherent averaging.
- Measuring the transmit burst sample and calculating its frequency, phase, and amplitude
- Correcting the I and Q phase and amplitude based on the transmit burst sample
- Optionally correcting I and Q based on the interference filtering algorithm
- For magnetron systems, calculating the AFC frequency error to apply corrections in the STALO control

The digital approach allows the software algorithms for these functions to be easily changed. Configuration information (for example, processor major mode, PRF, pulse width, range gate spacing, and so on) is supplied from the RVP10SRV host computer.

After IFDR10 has generated I and Q data, these can be corrected in a number of manners. The transmit burst pulse is sampled to measure its frequency, phase, and power. With this information, the following corrections may be performed.

Table 6 I and Q data correction methods

Correction method	Description
Amplitude correction	The RVP10 burst pulse analysis computes a running average of the transmit power. Individual received I and Q samples are corrected for pulse-to-pulse deviations from this average. This can substantially improve the phase stability of a magnetron system for clutter cancellation performance to near klystron levels. This also optionally allows adjustment in the radar constant calibration to account for different power levels pulse-to-pulse.
Phase correction	The phase of the transmitted pulse is measured. The I and Q values are adjusted for this measure phase, so that the phases stored have a common reference point. The coherency achievable is better than 0.1 degree by this technique.

Correction method	Description
Large signal linearization	When an IF signal saturates the ADCs, there is still considerable information in the signal since only the peaks of oscillating wave are clipped. The proprietary large signal linearization algorithm used in RPV10 provides an extra 3 - 4 dB of dynamic range accounting for the effects of saturation.

More information

- IF signal processing (page 155)
- Interference filter (page 161)

3.1.1.1 Digital IF bandpass design tool

The digital matched filter computes `I` and `Q` interactively using a TTY and oscilloscope for graphical display.

You can choose the filter's passband width and impulse response length, and RVP10 constructs the filter coefficients. You can display the frequency response of the filter and compare it to the frequency content of the transmitted pulse.

Microwave energy can come from a variety of transmitters such as ground-based, ship-based, or airborne radars as well as communications links. To minimize the risk of substantial interference to a weather radar system, RVP10 supports interference rejection algorithms. See Interference filter (page 161).

IFDR10 places the WDR `I` and `Q` samples on the Ethernet line, where they are sent to the computer's processor. The `I` and `Q` values are then processed to extract the moment information (`Z`, `V`, `W`, and optional polarization parameters).

Using the digital IF bandpass filter

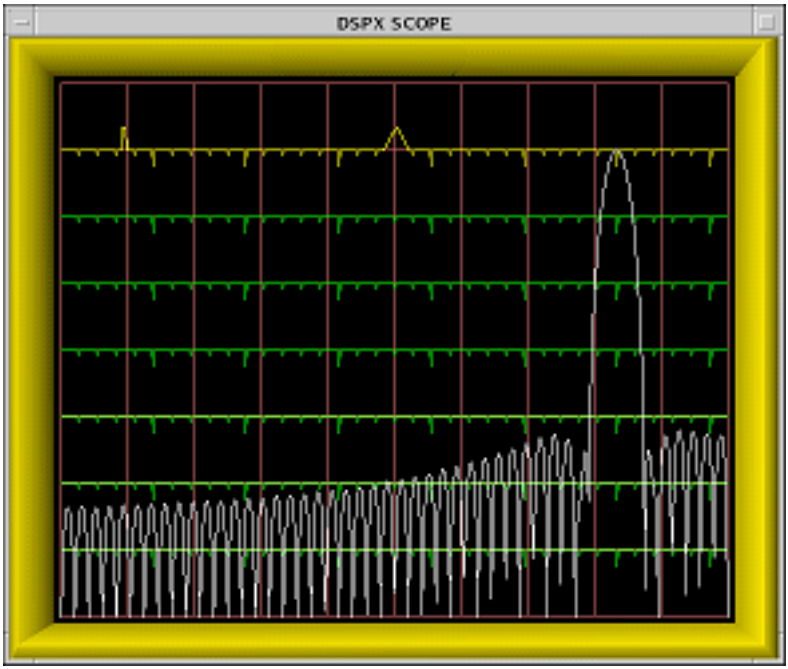


Figure 2 Digital IF bandpass design tool

You can use the filter design tool to design the optimal IF filter to match each pulse width and application.

To show the filter, specify the impulse response and pass band. Use keyboard commands to widen or narrow the filter and to automatically search for an optimal filter.

This display can also show the spectrum of the transmit burst pulse for quality control and comparison with the filter.

For more information on the tool, see chapter *Plot-assisted setups*.

3.1.1.2 Burst pulse alignment

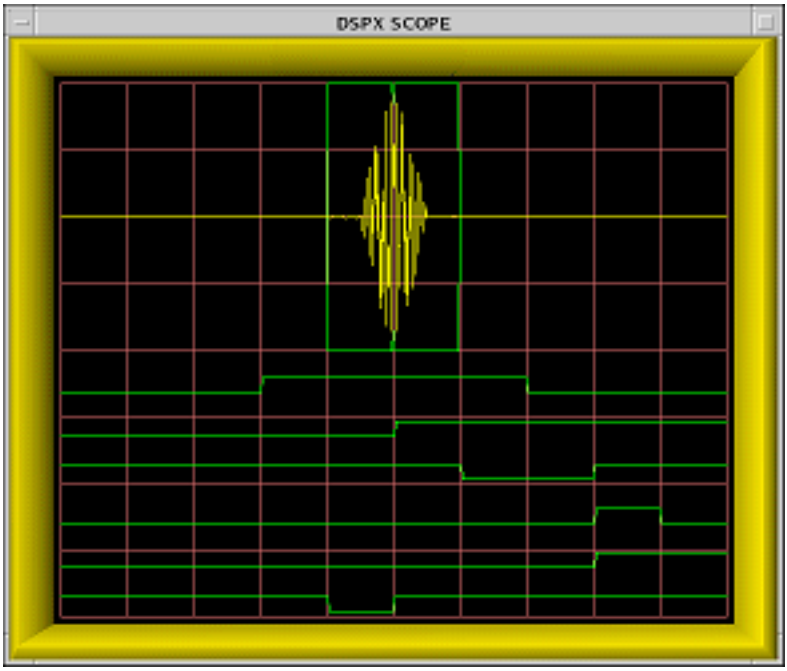


Figure 3 Burst pulse alignment tool

You can check the quality assessment of the transmit burst pulse and its precise alignment at range 0:

- *Manually*, with the burst pulse alignment tool
  This is often done from a central maintenance site.
- *Automatically*, with the burst pulse auto-track feature
  This makes the automatic frequency control (AFC) robust to start-up temperature changes and pulse width changes that can affect the magnetron frequency.

Quality assessments perform a 2D search in both time and frequency space if a valid burst pulse is undetected.

3.1.1.3 Received signal spectrum analysis tool

RVP10 provides plots of the IF signal versus range as well as spectrum analysis of the signal.

You can perform detailed analysis and configuration from a central maintenance facility through the network.

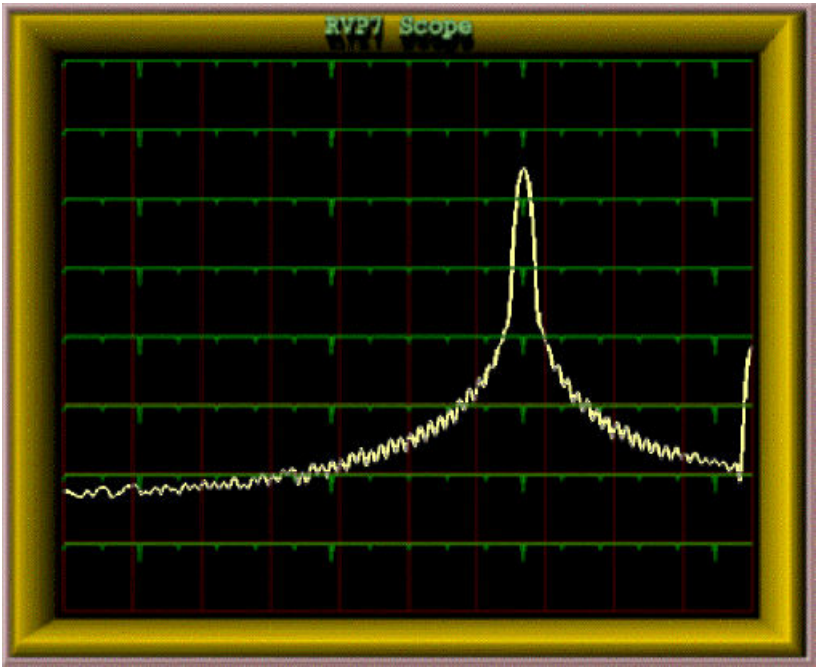


Figure 4 Received signal spectrum analysis tool

3.1.2 Data acquisition and convertors

Six IF channels are present within the IFDR10. These consist of identical receiver paths with 16-bit analog-to-digital convertors (ADC) that sample up to 240 MHz. The actual sampling frequency is user-controllable, 190 ... 240 MHz. These six channels allow an input frequency of 10 ... 120 MHz. The ADCs are transformer coupled with an ideal impedance of 50 ohms. In reality there is slight difference in impedence values depending on the IF center frequency. However, this design introduces virtually no noise, which is advantageous.

Uncertainty in high-speed ADCs may be introduced if the sampling gets confined to a small region, especially with low signal-to-noise ratio. In such a case the integral non-linearity of the ADC may cause a few tenths of a dB difference in power measurement when the ADCs are toggling a small number of bits. This becomes extremely relevant in ZDR, as different offsets will occur across the dynamic range of the receiver. A dithering signal is introduced in front of the ADCs at low frequency to ensure that all bits are being toggled, even when we are only interested in a small region. This avoids the introduction of non-linear response, keeping consistent measured gains between multiple ADCs over the full dynamic range.

IFDR10 uses two RF digital-to-analog convertors (DAC) capable of rates up to 6.2 GSa/s. Each DAC device has two separate data paths, each receiving one complex baseband data stream from the FPGA. Currently, one of these paths is used for producing a low frequency dithering signal described in the previous paragraph. Thus, this allows up to three transmit waveform designs to be used in parallel.

3.1.3 Timing / clocks

IFDR10 clocking and synchronization between the analog-to-digital and digital-to-analog data convertors, FPGA logic devices, and real-time embedded Linux threads are based upon the JESD204B standard. This is the first use of a standardized interface between the data convertors (ADCs and DACs) and logic devices.

JED204B allows IFDR10 to achieve deterministic latency across every serial link, from boot to boot. This enables synchronous sampling, multi-channel phase alignment, and gain control between the six ADCs, 4 TXDACs, and the FPGA within the system. In the first release, IFDR10 supports phase locking to a 10-MHz external reference clock.

3.1.4 IFDR10: IF to I and Q processing

The conversion of IF ADC samples to I and Q within the IFDR10 consists of the *decimation stage*, which combines the moving average filter, and the *down-sampling.* The moving average filter is also known as the convolution completed by a finite impulse response (FIR) filter. Depending on the selected configuration, the FIR filter is either a bandpass filter or a matching filter. In either case, the FIR filter is smoothing the signal, and as a result, most of the samples in the filtered data are redundant and can be dropped. The stage that drops the samples is called down-sampling. The IFDR10 has plenty of FPGA resources to perform these actions in parallel for the 6 receive channels.

The direct implementation of a moving average filter and then down-sampling is extremely inefficient because for each stage of decimation only one filter is used. Therefore, a polyphase filter is implemented, which combines the filtering and down-sampling into one step. The IFDR10 FPGA performs the initial processing of the raw IF digital data generated by the ADCs and output I and Q data values to the RVP10SRV host computer. The frequency, phase, and amplitude of the burst pulse from every transmitter are also measured.

3.1.5 TAG angle samples of azimuth and elevation (RVP10)

Antenna angle encoders typically provide measurement of true antenna pointing at a lower frequency (from some dozens to a few hundred hertz) while the transmitter pulses about ten times faster. There is also latency from the time the antenna position measurement is available to the receiver. To ensure that the I and Q data are tagged with correct pointing information, a model of antenna motion is executed within the FPGA.

This model takes as input the true antenna pointing measurement at a known timestamp. These are assembled with the RVP10SRV host computer. The host computer then provides the IFDR10 the pointing values and timestamp. IFDR10 then upconverts the antenna pointing information into higher frequency data, and extrapolates to account for the latency between gathering the pointing data and transmit pulse generation.

As the IFDR10 knows the exact time of day that a transmit pulse occurs, and the RVP10SRV knows the exact time of day antenna pointing angles are measured, it is essential that the IFDR10 and RVP10SRV are time synchronized to each other. Therefore, the RVP10SRV computer is configured as a time server to the IFDR10. The RPV10 product uses chrony as a versatile implementation of the Network Time Protocol. The IFDR10 has an internal watchdog process to monitor the status of the chrony time synchronization and to take action if it is interrupted.

It is only required that time be synchronized between the RVP10SRV and the IFDR10, but typically the RVP10SRV is also synchronized to an outside world clock reference.

3.2 IFDR10 Waveform design – Tx generation

RVP10 can function as a digital transmitter providing tools to design waveforms which IFDR10 synthesizes to be output at IF. This signal is filtered using analog components, then upconverted to RF, and finally amplified for transmission. The transmitter can be a solid-state or vacuum tube device.

IFDR10 has SMA outputs to distribute the IF Tx waveform labeled **TX 0 OUT** and **TX 1 OUT**. Each of these ports are connected to a 16-bit TxDAC to produce analog versions of the digitally designed IF waveforms. Each TxDAC contains two independent channels allowing quickly swapping channel 1 or 2 to the SMA port. This allows multiple waveforms to be stored within the TxDAC to quickly alternate for output to the SMA. The final analog waveform has 60 dB SNR from 10 to 120 MHz. The maximum power output is +10 dBm for each TX output.

TX 2 OUT is a replica of the **TX 0/1** channels. However, **TX 2 OUT** cannot output a waveform until the **TX 0/1** channels have completed outputting waveforms there. Therefore, **TX 2** is to be used as test source, possibly outputting a signal into a receiver for calibration or test signal monitoring.

The RVP10 digital transmitter complements the digital receiver in the radar system. The receiver samples an IF waveform that has been down-converted from RF. The transmitter synthesizes an IF waveform for up-conversion to RF. This gives RVP10 control over both halves of the radar, making matched Tx/Tx processing algorithms possible.

Table 7 Example Tx/Rx processing algorithms

Algorithm	Description
Phase modulation	Some radar processing algorithms rely on modulating the phase of the transmitter from pulse to pulse. This is traditionally done to separate signals on receive from interfering with each other. The RVP10 supports pseudo-random, SZ8/64, and fixed phase codes. Custom phase codes may also be requested via the Host Computer API when controlling RVP10 directly. However, the Vaisala IRIS Radar software does not support custom phase codes.

Algorithm	Description
Pulse compression	Low peak power transmitters require the use of longer transmit pulse lengths to achieve similar average power output as high power tubes. Normally, a longer pulse width would result in a reduction of range resolution (up to 13km in case of a 90us pulse). However, with the use of pulse compression techniques the range resolution can be recovered. The solution is to increase the Tx bandwidth by modulating the overall pulse envelope to restore a reasonable range resolution. RVP10 supports both Linear Frequency Modulation (LFM) and Non-Linear FM (NLFM). NLFM is typically able to suppress range-time sidelobes more effectively than LFM, and their use is the normal case.
Frequency agility	RVP10 frequency agility is as simple as changing the center frequency of the digitally designed waveform to be synthesized. Many Range/Doppler unfolding algorithms become possible when multiple transmit frequencies can co-exist. Frequency agility can also be combined with pulse compression to remedy the blind spot, at close ranges, while the long pulse is being transmitted. Vaisala calls this use case 'hybrid pulsing'.
Waveform agility	The new capabilities in RVP10 allow for multiples of waveforms to be digitally designed and stored within the TxDAC. Then as the TxDAC is synthesizing the analog output from one of its internal channels, the 2nd channel may be programmed for a new waveform. This allows the chaining of several different pulse designs into a single transmit sequence interval. This capability opens a completely new realm in weather radar signal processor and introduction of new time-space, phase, and polarization modulations within multi-PRI's.

3.3 RVP10 weather signal processing

Radar signal processor (RVP) software triggers radar measurement by producing the trigger signal for the transmitter using the intermediate frequency digital receiver (IFDR) unit in the receiver.

After the IFDR unit has digitized the received echo signal into samples (`I` and `Q` data), the RVP processes the data in the radar server computer using computations such as:

- Converting the received signal amplitude into calibrated radar reflectivity values
- Doppler processing to filter out ground clutter and compute radial velocities
- Polarimetric processing to classify the measured hydrometeors and to apply attenuation correction.

The end product of the RVP process is a radar ray, where selected radar data from a certain short time interval is stored as a function of range.

The following figure shows a block diagram of moment processing.

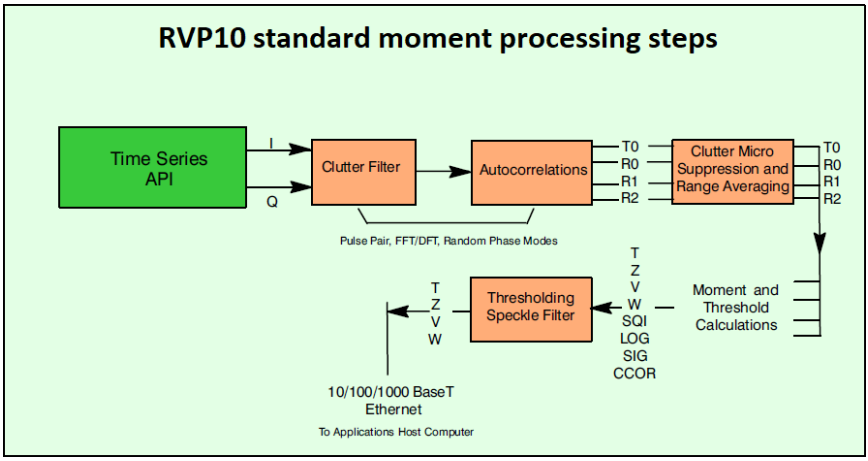


Figure 5 I/Q Processing for weather moment extraction

Using the **R2** lag improves the estimation of signal-to-noise ratio and spectrum width. Processors that do not use **R2** cannot effectively measure the SNR and spectrum width.

Parameters configure signal processing, such as pulse repetition frequency, range resolution, and Doppler filter parameters. You can select these either directly when running RVP as a standalone or through the IRIS software when IRIS controls the RVP process during automatic weather radar measurements.

Applications

The resulting intensity, radial velocity, spectrum width, and polarization measurements are sent to a separate host computer to serve as input for applications such as:

- Quantitative rainfall measurement
- Vertical wind profiling
- Z_{dr} hail detection
- Tornado and microburst detection
- Gust front detection
- Particle identification
- Target detection and tracking
- General weather monitoring

Processing Modes

RVP10 has several major processing modes options for obtaining basic moments:

- Pulse pair mode time domain processing
- DFT/FFT mode frequency domain processing
- Random phase mode for second trip echo filtering
- Polarization mode processing

RVP10 performs discrete Fourier transforms (DFT) and fast Fourier transforms (FFT). FFT is more computationally efficient than DFT, but the sample size is limited to a power of two (16, 32, 64, and so on.) This is too restrictive on the scan strategy for a modern Doppler radar since this means, for example, that a 1° azimuth radial must be constructed from exactly 64 input `I/Q` values. RVP10 has such processing power that when the sample size is not a power of two, a DFT is performed instead of an FFT.

3.3.1 Burst pulse analysis for amplitude, frequency, and phase

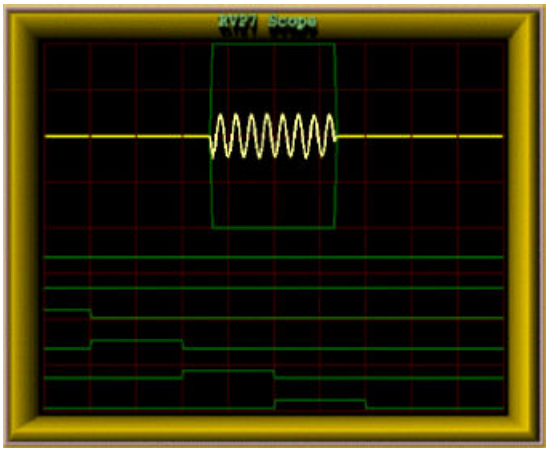


Figure 6 Burst pulse analysis for amplitude/frequency/phase

The burst pulse analysis provides the amplitude, frequency, and phase of the transmitted pulse.

Amplitude measurement can provide enhanced performance by allowing the `I` and `Q` values to be corrected for variations in both the average and the pulse-to-pulse transmitted power. A warning is issued if the burst pulse amplitude falls below a threshold value.

1. The burst pulse data stream is first analyzed by an adaptive algorithm to locate the burst pulse power envelope (for example, 0.8 µsec).
   The algorithm does a coarse search for the burst pulse in the time/frequency domain (by scanning the AFC). It then does a fine search, in both time and frequency, to assure that the burst is centered at range 0 and is at the required IF value.
2. The power-weighted phase of the burst pulse and the total burst pulse power is computed.
3. The power-weighted average phase is used to make the digital phase correction.

Phase jitter for magnetron systems, with good quality modulator and STALO, is better than 0.5° RMS, as measured on nearby clutter targets.

The burst pulse frequency is also analyzed to calculate the frequency error from the nominal IF frequency. For magnetron systems, the error is filtered with a selectable time constant, which is typically set to several minutes to compensate for slow drift of the magnetron. The digital frequency error is sent through the uplink to the IFDR in the receiver cabinet.

3.3.2 Time (azimuth) averaging

Autocorrelations are based on input `I` and `Q` values over a selectable number of pulses between 8 ... 256. Any integer number of pulses in this interval may be used, including `DFT/FFT` and random phase modes.

Selectable angle synchronization using the input `AZ` and `EL` tag lines assures that all possible pulses are used during averaging for each, for example, 1° interval. This minimizes the number of wasted pulses for maximum sensitivity. Azimuth angle synchronization also assures the accurate vertical alignment of radial data from different elevation angles in a volume scan.

3.3.3 Range averaging and clutter microsuppression

Range averaging can improve the accuracy of the reflectivity measurements.

When this is done, autocorrelations from consecutive range bins are averaged, and the result is treated as a single bin. This means that the host computer must process fewer range bins.

Range averaging of the autocorrelations may be performed over 2 ...16 bins. Prior to range averaging, any bins that exceed the selectable clutter-to-signal threshold are discarded. This prevents isolated strong clutter targets from corrupting the range average, which improves the sub-clutter visibility.

3.3.4 Moment extraction

The autocorrelations are the basis for Doppler moment calculations:

- Mean velocity—from `Arg [ R1 ]`
- Spectrum width—from `|R1|` and `|R2|` assuming Gaussian spectrum
- `dBZ`— from `|R0|` with correction for ground clutter, system noise, and gaseous attenuation. Uses calibration information supplied by host computer
- `dBT`—identical to `dBZ`, but without ground clutter

These standard parameters are output to the host application.

3.3.5 Threshold processing

RVP10 calculates several parameters that are used to threshold (discard) bins with weak or corrupted signals. The thresholding parameters are:

- Signal quality index (`SQI=|R1|/R0`)
- `LOG` (or incoherent) signal-to-noise ratio (`LOG`)
- `SIG` (coherent) signal-to-noise ratio
- `CCOR` clutter correction

These parameters are computed for each range bin and can be applied in **AND/OR** logical expressions, independently for **dBZ**, **V**, and **W**.

3.3.6 Speckle filter

A speckle filter is a final pass over each output ray, in which isolated, single bins of velocity, width, or intensity are removed.

There are two speckle removers in RVP10:

- 1D single-ray speckle filter (default)—This is used for any output parameter.
- 2D 3x3 speckle filter—If enabled, this is used for any output parameter.

This eliminates single pixel speckles, which allows the thresholds to be reduced for greater sensitivity with fewer false alarms (speckles).

3.3.7 Velocity unfolding

The RVP10 processor can unfold mean velocity measurements based on a dual -PRF algorithm.

In this technique, 2 radar PRFs are used for alternate **N**-pulse processing intervals. The internal trigger generator automatically produces the correct dual-PRF trigger. An external trigger can also be applied.

For external triggers, the **ENDRAY_** output line indicates when to switch rates. RVP10 measures the PRF to determine which rate (high or low) was present on a given processing interval. It then unfolds based on a 2:3, 3:4, or 4:5 frequency ratio.

Table 8 Example unambiguous velocity intervals

PRF1	PRF2	Unambiguous range (km)	Unambiguous velocity (m/s)			Unfolding
			3 cm wavelength	5 cm wavelength	10 cm wavelength	
500	*	300	3.75	6.25	12.50	None
1000	*	150	7.50	12.50	25.00	
2000	*	75	15.00	25.00	50.00	
500	333	300	7.50	12.50	25.00	Two times unfolding
1000	667	150	15.00	25.00	50.00	
2000	1333	75	30.00	50.00	100.00	
500	375	300	11.25	18.75	37.50	Three times unfolding
1000	750	150	22.50	37.50	75.00	
2000	1500	75	45.00	75.00	150.00	

PRF1	PRF2	Unambiguous range (km)	Unambiguous velocity (m/s)			Unfolding
			3 cm wavelength	5 cm wavelength	10 cm wavelength	
500	400	300	15.00	25.00	50.00	Four times unfolding
1000	800	150	30.00	15.00	100.00	
2000	1600	75	60.00	100.00	200.00	

3.3.8 Autocorrelations

The autocorrelations R0, R1, and R2 are produced by Pulse Pair, Random Phase, and DFT/FFT modes. The way that they are produced is different for each mode, particularly with regard to the filtering.

Table 9 Autocorrelation mode

Mode	Description
Pulse Pair Mode	Filtering for clutter removal may be performed in the time domain or frequency domain. Traditional IIR type clutter filters are available in the time domain. However, the frequency domain filter is more adaptable. Clutter filtering can be optionally performed in the frequency domain, and then inverse FFT or DFT may be performed to return to time domain after clutter removal. Autocorrelations are then computed in the time domain.
DFT/FFT Mode	Filtering for clutter is performed in the frequency domain using both fixed width filters and the Gaussian Model Adaptive Processing (GMAP) technique. Autocorrelations are computed from the inverse transform.
Random Phase	Filtering for clutter and second trip echo is performed in the frequency domain by adaptive algorithms. Autocorrelations are computed from the inverse transform.

3.3.8.1 RVP10 pulse pair time domain processing

Pulse pair processing is done by direct calculation of the autocorrelation.

Prior to pulse pair processing, the input `I` and `Q` values are filtered for clutter using a time domain notch filter, frequency domain fixed, or variable width filters. IIR filters of various selectable widths are available for either 40 dB or 50 dB stop band attenuation. The filtered I/Q values are processed to obtain the autocorrelation lags R0, R1, and R2. The unfiltered power is also calculated (T0). The autocorrelations are sent to the range averaging and moment extraction steps.

3.3.8.2 RVP10 DFT/FFT Processing

The DFT/FFT mode allows clutter cancellation to be performed in the frequency domain. DFT is used in general, with FFTs used if the requested sample size is a power of two.

The following windows provide the best match of window width to the spectrum dynamic range:

- Rectangular
- Hamming
- Blackman
- Exact Blackman
- Von Han

After the FFT step, clutter cancellation is done with the options of using **GMAP**, a selectable fixed width filter that interpolates across the noise or any overlapped weather, or an adaptive filter which automatically determines the optimal width. This technique preserves overlapped weather as compared to time domain notch filters, which always attenuate overlapped weather to some extent, depending on the spectrum width.

After clutter cancellation, `R0`, `R1`, and `R2` are computed by inverse transform and these are used for moment estimation.

3.3.8.3 Random phase processing for second trip echo

Second trip echoes can be a serious problem for applications that operate at a high PRF. Second trip echoes can appear separately, or they can be overlaid on first trip echoes (second trip obscuration).

The random phase technique[1)] separates the first and second trip echoes so that:

- In most cases, the second trip echo can be removed from the first trip, even in the case of overlapped first and second trip echoes. The benefit is a clean first trip display.
- The second trip echoes can be recovered and placed at their proper range at first trip/second trip signal ratios of up to 40 dB difference for overlapped echoes. Because of the wide dynamic range of weather echoes, this power limit is sometimes exceeded.

The technique requires that the phase of each pulse be random. Digital phase correction is applied in the processor for the first and second trips. The adaptive filter removes the echo of the other trip to increase the SNR.

Magnetron radars have a naturally random phase. For Klystron radars, a digitally controlled precision IF phase shifter is required. RVP10 provides an 8-bit RS422 output for the phase shifter.

3.4 Configuration and monitoring

RVP10SRV radar server computer stores the setup configuration on a disk.

1) *Joe, P., and A. Siggia, 1995: Second Trip Unfolding by Phase Diversity Techniques. Preprints of the 27th Conference on Radar Meteorology, American Meteorological Society, 770-772.*

You can access setup information locally or remotely, over the network. For multiple radar networks, you can manage the configuration centrally by copying tested master configuration files to the network radars.

Boot-up process

During the boot process, RVP10SRV is first loaded with a factory configurable diagnostic image.

The diagnostic image initializes devices that need initialization, and leaves the board in a reset condition. It verifies that the RDA software and setup configuration is present and uncorrupted. Once validated, the diagnostic image boots the RDA application.

The application image runs basic memory self-tests before starting.

Monitoring

The platform monitors:

- voltage
- temperature
- locked conditions of the PLLs on the FPGA to verify that the analog and digital clocks are in good health.

3.5 Utilities and applications

RVP10 includes applications and utilities for the calibration, alignment, and configuration. These can be run locally on RVP10 or over the network from a central maintenance facility.

Table 10 RVP10 utilities

Utility	Description
Ascope	A utility for controlling the signal processor manually and to display moments, times series, and Doppler spectra. Includes a signal simulator able to produce first and second trip targets. Can record and playback of time series and moments.
dspx	An ASCII, text-based program for accessing and controlling the signal processor, including accessing local setup menus.
DspExport	Exports RVP10 to another workstation over the network. This allows utilities on a remote network to run locally, instead of exporting the utility display window over the network. Improves the performance of the utilities for network applications by letting them be run on the workstation that is remote from the RVP10. The standard X-Window export is supported, but requires more bandwidth.
Setup	Interactive GUI for creating and editing RVP10 configuration files.
Zauto	Calibration utility for use with a test signal generator.

Table 11 Radar Application Software

Application	Description
IRIS/Radar	The package provides complete local and remote control/monitoring, data processing, and communication for a radar system. • Runs on the same or separate PC. • Interfaces to RVP10 internally or by 100 BaseT Ethernet. • Controls RVP10 and RCP8 radar/antenna control processor.
IRIS/Analysis	Functions as a radar product generator (RPG) to provide outputs such as CAPPI, rain accumulations, echo tops, automatic warning and tracking, and so on. Runs on a separate PC, often at a central site.
IRIS/Focus	Functions as a radar product generator (RPG) to provide outputs such as CAPPI, rain accumulations, echo tops, automatic warning and tracking, and so on. Displays radar products on standard PC (Windows or Linux) running standard browsers. Includes features such as looping, cross-section, track, and alerts.

4. RVP10 hardware

4.1 Hardware system overview

The major modules supplied with RVP10 are:

- IFDR10 Intermediate Frequency Digital Receiver (IFDR), which is typically mounted inside the radar receiver.
- RVP10SRV server main chassis, which is typically mounted in a 19″ EIA rack (radar cabinet).

RVP10 hardware installation includes mechanical installation and siting, electrical specifications of the interface signals, system-level considerations, and the standard connector panel.

Much of the RVP10 I/O is configured through software. Since there is no custom wiring, internal jumpers, or oscillators, it is easy to insert spare modules.

4.2 IFDR10 hardware

Figure 7 IFDR10

The IFDR10 hardware consists of a printed circuit card assembly contained in a mechanical housing. The tight metal housing gives the device the maximum noise immunity. The housing protects the PCB, a heat sink, and an EMI/RFI barrier. The mounting options are designed to be backwards compatible with RVP901 IFDR. IFDR10 may be mounted flat or upright on either of its side.

The dimensions of the module are 246 mm × 136 mm × 52.5 mm (10.6 in × 5.6 in × 2.0 in)

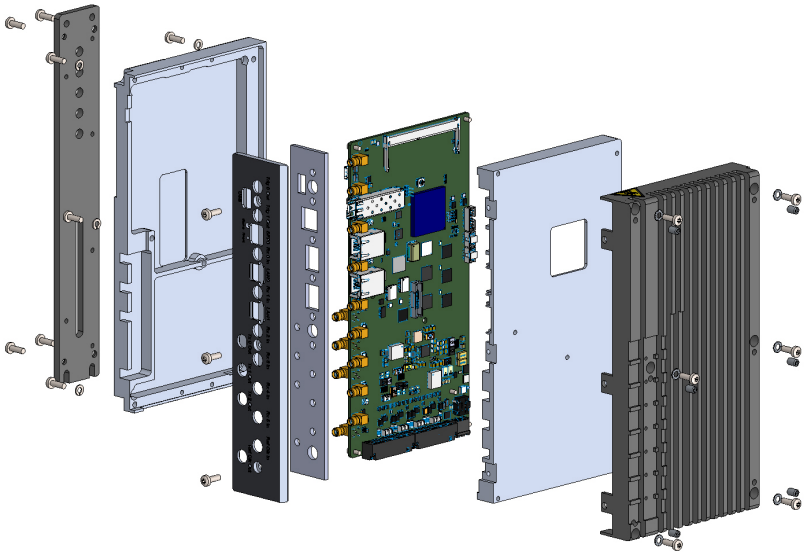


Figure 8 IFDR10 components

More information

- Configuring RVP10 with CLI (page 56)

4.2.1 Hardware security

IFDR10 has been hardened against unprivileged access and execution of malicious software in the device. Firmware integrity is verified in all stages of code execution by checking the firmware signatures before execution. This permits only official signed firmware by the device manufacturer to execute.

Any attempt to use modified firmware in the IFDR10 will lead to a signature verification failure, making the IFDR10 hardware unusable. IFDR10 does not contain any predefined login credentials or passwords, which prevents any unauthorized access into the embedded Linux operating system running the device. The only access into the IFDR10 settings and configuration are via the interfaces Vaisala provides.

More information

- Configuring RVP10 with CLI (page 56)

4.2.2 IFDR10 connectors

Figure 9 IFDR10 front connector panel

Table 12 IFDR10 front connector panel

Label	Type	Description
Trig 0 Out	SMA	General purpose trigger I/O
Trig 1 Out	SMA	General purpose trigger I/O
SFP 0	SFP+	10 Gbps fiber optic network link
Rx 0 In	SMA	Direct IF input functionally assigned with `ifd_settings`
LAN 0	RJ-45	1 Gbps Ethernet
Rx 1 in	SMA	Direct IF input functionally assigned with `ifd_settings`
LAN1	RJ-45	1 Gbps Ethernet
Rx 2 in	SMA	Direct IF input functionally assigned with `ifd_settings`
Rx 3 in	SMA	Direct IF input functionally assigned with `ifd_settings`
Rx 4 in	SMA	Direct IF input functionally assigned with `ifd_settings`
Rx 5 in	SMA	Direct IF input functionally assigned with `ifd_settings`
Ref Clk in	SMA	Reference clock input
USB	USB-C	Used for manufacturing purposes. Currently no field applications.
Tx 0 out	SMA	Direct Transmit IF output
Tx 1 out	SMA	Direct Transmit IF output
Tx 2 out	SMA	Direct Transmit IF output
Ref Clk out	SMA	Reference clock output

Label	Type	Description
Status	LED indicator light	Reserved for future development
Reset	Switch	Tactile, momentary on switch. Returns the IFDR10 IP Address and FPGA image to factory boot defaults.

The GPIO connector panel consist of 8 independent I/O stages, which can be flexibly configured as one differential input or output, or two single-ended input or output. Differential interface is implemented using a standard half-duplex RS485 transceiver. Both differential transmitter and receiver are independently controlled. In the single-ended mode, both differential transmitter and receiver are disabled. The input/output and single-ended/ differential modes are set for pin pairs (GPIO0 and GPIO1, GPIO2 and GPIO3, and so on). Setting pins of the same pin pair to different modes is not possible.

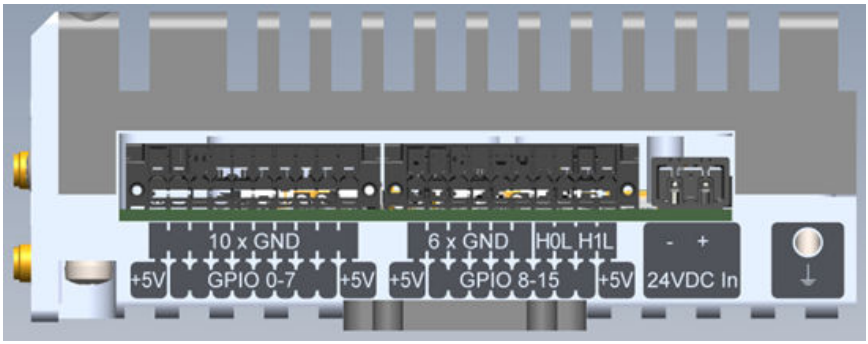


Figure 10 IFDR10 GPIO connector panel

Table 13 Electrical properties of GPIO pins in single-ended mode

Parameter	Value
Minimum high level input voltage	3.5 V
Maximum low level input voltage1	1.5 V
Absolute maximum input voltage	5.0 V
Absolute minimum input voltage	0.0 V
Even pins in input mode	Pulled low to 0.6 V with 17 kΩ
Odd pins in input mode	Pulled high to 4.9 V with 18 kΩ
Input time delay (typ)	137 ns

Parameter		Value
Output voltage high (typ)	Iout = -0.0 mA	5.0 V
	Iout = -10.0 mA	4.3 V
	Iout = -20.0 mA	3.6 V
	Iout = -35.0 mA	2.6 V
Output voltage low (typ)	Iout = 0.0 mA	0.0 V
	Iout = 10.0 mA	0.7 V
	Iout = 20.0 mA	1.4 V
	Iout = 35.0 mA	2.5 V

Table 14 Electrical properties of GPIO pins in differential mode

Parameter		Value
Differential input threshold voltage (typ)1		-30 ... -70 mV
Differential input threshold voltage (max)2		-200 ... + 200 mV
Maximum differential input voltage		±5 V
Absolute maximum input voltage to gnd		5.0 V
Absolute minimum input voltage to gnd		0.0 V
Common mode voltage to gnd if not limited by absolute max/min input voltage		0.0 ... 5.0 V
Differential voltage in input mode (typ) due to pull-up and pull-down resistors		4.3 V
Differential input resistance (typ)		24 kΩ
Even pins in input mode		Pulled low to 0.4 V with 18 kΩ
Odd pins in input mode		Pulled high to 4.8 V with 16 kΩ
Input time delay (typ)		149 ns
Differential output voltage high (typ)	Iout = 0.0 mA	4.1 V
	Iout = 10.0 mA	2.8 V
	Iout = 20.0 mA	2.3 V
	Iout = 35.0 mA	1.6 V

4.2.2.1 IFDR10 inputs

All IFDR10 inputs are on SMA connectors. The IF signal input is made immediately after the STALO mixing/sideband filtering step of the receiver. The required signal level for both the IF signal and burst is +11.2 dBm for the strongest expected input signal.

You can use a fixed attenuator or IF amplifier to adjust the signal level to be in this range. The maximum signal level is 16 dBm.

Table 15 IFDR10 inputs and outputs

Input	Description
IF signals	IFDR10 has 6 Rx input channels for IF signals: **Rx 0** - **Rx 5**. The user can select which channels are used for primary and secondary polarization IF signals and for wide dynamic range.
IF burst pulse sample for magnetron	Received over **ADC–E**.
Trigger input or output	2 trigger outputs on SMA connectors: **Trig 0** and **Trig 1** (50 Ohms / 4 V). 4 trigger inputs in the GPIO panel

Digitizing is performed for both the IF signal and burst channels from a user-selectable sampling frequency range of 190 ... 240 MHz at 16-bit resolution to sub-nanosecond accuracy. This provides 92 ... 107 dB of dynamic range (depending on pulse width) without using complex AGC, dual A/D ranging, or down-mixing to a lower IF frequency. Each A/D converters is time synced within 1 nanosecond to ensure sampling in multiple channels is of the nearly equivalent targets.

RVP10 provides AFC support for tuning the STALO of a magnetron system. Alternatively, the magnetron can be tuned by a motorized tuning circuit controlled by RVP10.

4.2.3 IFDR10 power, size, and mounting considerations

Table 16 IFDR installation considerations

Consideration	Description
Communication	The IFDR communicates with a host computer through 10Gb Ethernet. The Auto-MDIX feature eliminates concerns about cross-over versus straight cables. The platform provides support for TCP and UDP packets. The default IP address, shipped with each system, is 10.0.3.254. The IFDR supports jumbo packets. Vaisala recommends the UDP packet sizes be set to 8192 on the host computer. To comply with industry standards, use a shielded CAT5e cable (certified to 350 MHz), with shielded RJ45 plugs on each end. Ensure the Ethernet connectors on the host computer and IFDR make contact with the metal on the shielded cable, which provides DC return path. This design prevents ground loop currents from flowing between units, even when they are plugged into different AC/Mains.
Cooling	The unit is cooled by direct conduction of heat through the metal chassis. One side of IFDR10 serves as a heat sink. The hotter chips mounted on the printed circuit board are bonded to the heat sink. Position IFDR10 so that air can freely convect around it. A minimum of 0.6 m^3/min of air flow required. The ambient air temperature range is -40 ... +55 °C (-40 ... +131 °F).
Filters	Reserve mounting space for the external, analog, anti-alias filters. The filters can be mounted nearby in the radar cabinet, or they can be attached directly to the IFDR, on the opposite side from the power module.
Mounting	The unit is designed to be mounted on an edge. The IFDR is a compact sealed module with dimensions 246 mm × 136 mm × 51 mm (9.7 in × 5.4 in × 2.0 in). With the mounting bracket, the length of the module is 270 mm (10.6 in) and height 52.5 mm (2.1 in). See IFRD10 technical drawings (page 352).
Power	Nominal operating voltage is 24 VDC. Operating voltage range is 20 ... 30 VDC. Typical power consumption is 36 W.

4.2.4 Installing IFDR10

When installing IFDR10, note the following:

1. Mount the device so that the connectors are easy to access.
2. When connecting the cables, clean the connectors carefully with suitable tools.
3. Tighten the coaxial SMA connectors to 0.57 Nm with a torque wrench.

   It is important that the connectors are in the correct tightness.
4. When using insulated sleeves, use ferrules with 10 mm contact length.

4.3 RVP10SRV signal processing computer

The RVP10SRV signal processing computer hosts the Linux operating system and provides the computing resources for processing the I/Q values that are generated by IFDR10.

4.3.1 LAN connection for data transfer or parallel processing

For external communication, RVP10SRV supports a standard 1 Gbit Ethernet.

For communication between RVP10SRV and IFDR10, there is a 10 Gbit fiber optic cable connection.

For most applications, the IRIS Radar software is installed on the same server with RDA software. Moment results (`Z`, `T`, `V`, and `W`) are transferred internally.

The Ethernet is used to transfer moment results (`Z`, `T`, `V`, and `W`) to third-party application host computers, such as a product generator.

IFDR10 communicates over 100 Base T or 10-Gigabit Ethernet using UDP packets to send the `I` and `Q` values to RVP10SRV, and can broadcast them - allowing for archiving or parallel processing.

The digital `I` and `Q` data produced by IFDR10 is sent to the RVP10SRV server to perform the processing using pulse pairs, Fourier transforms, or random phase techniques.

4.3.2 Open hardware and software design

RVP10SRV is an open-architecture radar signal processor. The RVP10 software runs on AlmaLinux operating system.

Using public APIs, researchers and OEM manufacturers can modify or replace existing algorithms, or write their own software using the RVP10 software as a foundation.

To support software upgrades, RVP10SRV can flash IFDR10 software. The RDA version can be updated in the field with minimal risk. RVP10SRV software provides a configuration interface. For more information, see *IRIS and RDA Software Installation Guide (M212924EN)*.

4.3.3 RVP10 socket interface

RVP10SRV is configured to listen on a network port. It is ready to interface to a host computer through the network using the **DspExport** program. When the IRIS Radar software is installed on the same RVP10SRV computer, it is pre-configured to communicate with RVP10SRV processes through the native interface, bypassing the socket interface.

RVP10SRV includes built-in utilities such as **Setup**, **dspx**, and **Ascope**. See *IRIS and RDA Utilities Guide (M212925EN)*.

Because RVP10SRV can only have one program controlling it at a time, using a local program such as **dspx** blocks network access.

4.3.4 RVP10SRV socket protocol

The socket interface supports the commands in appendix *RVP10 programming interface*.

All messages going both ways consist at the lowest level of an 8-character decimal ASCII number, followed by a block of data. The decimal number indicates how many bytes follow. Generally, all data transfers are initiated by the host computer by sending a block of data, consisting of a command word followed by the | character, followed by optional data.

It responds to all commands with either an `Ack|`, indicating acknowledgment that the command was `OK`, or `Nak|`, indicating that there was an error. For `Nak|`, the reply always includes a string indicating what the error was. For Ack, there is optional data following.

On initial socket connection request **DspExport** provides a response of either `Nak|`, indicating the connection failed and why, or `Ack|`, followed by some connection information. The`Ack|` string is in the form of name/value pairs, for example:

```
Ack|CanCompress=1,Model=RVP10,Version=10.0
```

Your program can evaluate, or ignore, these keywords. `CanCompress=1` indicates that the **DspExport** computer supports compression. The host computer can then choose to use compression. When you first connect, you are in the "info only" mode. This means that the server only responds to **INFO** and **OPEN** commands.

DspExport supports the following commands:

- Read command (**READ**)
- Write command (**WRIT**)
- Read Status command (**STAT**)
- Set Information command (**INFO**)
- Read data available command (**RDAV**)
- Open the connection for I/O (**OPEN**)

Table 17 Socket protocol commands

Command	Example
Read (**READ**)	Example: `READ|100|` means: Read 100 bytes from the RVP10. Since the RVP10 interface is a 16-bit word interface, these read sizes should always be even. This command returns a `Ack|` followed by 100 bytes of binary data, or with a `Nak|`, meaning there can be no partial reads.
Write (**WRIT**)	Example: `WRIT|<data>`, where `<data>` is some binary data. This data is written to RVP10. The data size should be even.
Read Status (**STAT**)	Example: `STAT|` reads the status bits back from the RVP10. This is a 1-bit value, set to **1** if RVP10 has data available in its output buffer. The command returns `Ack|0`, `Ack|1`, or `Nak`. This is equivalent to the `dspr_status()` call in the `dsp` library.
Set Information (**INFO**)	Example: `INFO|ByteOrder=LittleEndian,WillCompress=1,Version=10.0`. This command can be used to inform RVP10SRV `DspExport` about the host computer. Available options are: • `ByteOrder`—Informs `DspExport` of the byte order of the host computer. This is needed because all the data read or written to/from the RVP10 is in 16-bit words. If the host computer has a different byte order from the RVP10, DspExport byte swaps the data. • `WillCompress`—Informs `DspExport` to use compression or not. Compression is only used if both sides agree to use it. The host computer should only set this to 1 if it received a `CanCompress` of 1 on initial connection. The data from normal **READ** commands is compressed. If the data is compressed, it replies with the acknowledge compressed string of `AkC`. The compression program is the `zlib` compress and uncompress. The uncompress function requires that the caller know the expected uncompressed size. This is true for RVP10 reads, because the reader always specifies the read size. • `Version`—Sends the IRIS version.
Read Data Available (**RDAV**)	Example: `RDAV|100|2|` means read up to 100 bytes of data from the RVP10SRV in individual DMA transfers of 2 bytes each. Before each read, the status is checked to see if there is more data available. If not, the read stops, and the number of bytes read is returned. This is merely a performance enhancing command, since the same feature is available by using the **READ**and **STAT** commands.

Command	Example
Open the Connection for I and O (**OPEN**)	Example: **OPEN** means: Switch from "open to info only" mode to open for I/O. If the signal processor is in use by another device, this command returns an error. Multiple clients are allowed to connect for info only, but only one can do I/O. If you run `DspExport` with the `-803` command line option, you get the legacy behavior, which means that every connection automatically sends the **OPEN** command. There is no reverse command to switch back to open for info only. There is also no such library call in the driver.
Read **Zcal** Information (**RCAL**)	Example: `ZCAL` means: Read the `dsp_refl_cal` structure from RVP10 and send it back in an ASCII name=value pair format. This is the structure configured by **Zauto** and **Zcal**. The configuration is served to all clients using RVP10.
Reset Kernel FIFOs (**RKFF**)	Example: `RKFF|2|` means: Reset the kernel FIFOs on the RVP10. The argument specifies which direction FIFOs to reset.
Read **Setup** Information (**SETU**)	Example: `SETU` means: Read the `dsp_manual_setup` structure from RVP10 and send it back in an ASCII `name=value` pair format. This is the structure configured in the RVP section of the **Setup** utility. The configuration is served to all clients using RVP10.
Write **Zcal** Information (**WCAL**)	Example: `WCAL|...` writes the `dsp_refl_cal` structure to RVP10 for saving.

4.3.5 Public API

To support writing your own signal processing algorithms for RVP10, the RVP10 software architecture enables you to statically link plug-in modules to the running code. The following table shows how the API supports adding software extensions to the RVP10 framework to modify some signal processing stages.

Table 18 API support for modifying signal processing

Processing stage	API support
Tx/Rx waveform synthesis and matched filter generation	Define transmit waveforms from pulse-to-pulse, along with the corresponding FIR coefficients that extract (`I`,`Q`) from the Tx waveform. Allows users to experiment with arbitrary waveforms for pulse compression and frequency agility.
Time series and spectra processing from (`I`,`Q`)	Modify default time series and spectra data, for example, to perform averaging or windowing in a different way.

Processing stage	API support
Parameter generation from (`I`,`Q`)	• Redefine how standard parameters (`dBZ`, `Velocity`, `Width`, `PHIDP`, and so on) are computed from the incoming (`I`,`Q`) time series. • Create new parameter types.

The standard scientific algorithms are not public. Many RVP10 library routines are also documented and can be called by user code, but the source to these routines is not generally released. Development tools must be purchased separately.

4.3.6 Installing the RVP10SRV server computer

WARNING! Electricity hazard
The RVP10SRV server computer has two power supplies.
Disconnect power from both power supplies when powering down or servicing the unit.

1. Mount the RVP10SRV server chassis in an equipment rack on rack slides.
2. Use the fiber optic cable to connect IFDR10 to the RVP10SRV server chassis.

   Clean the connectors carefully with appropriate tools. If the connectors are not properly cleaned, the data quality may suffer.
3. Connect the power cables.

   The device includes dual-redundant power supplies, with two line cords.

More information

- RVP10SRV signal processing computer specifications (page 347)

4.3.6.1 Powering up RVP10SRV

1. Power-up and boot IFDR10.

   If IFDR10 is not powered-up and booted, the `rvp10` process cannot start on RVP10SRV.
2. Start or reset RVP10SRV.

   The host Linux PC goes through an automated boot process that ultimately starts the RVP10 application.
3. While RVP10SRV runs extensive internal diagnostics, if no display is connected yet, you can connect a display to view any error messages or monitor the startup log in */var/log/irisrda/rvp10.log*.

4.4 Configuring network interface from RVP10SRV to IFDR10

This chapter describes how to configure the network interface from RVP10SRV server to IFDR10.

Configure the following parameters in the *profile.conf* file, located in */etc/vaisala/irisrda/profile.conf*.

```
IP Adress of networked RVP10/IFD IQ stream
```

The parameter **IP Adress of networked RVP10/IFD IQ stream** defines the Ethernet address of the source IFDR10's IQ data stream. For the best results, attach the IFDR to a dedicated Ethernet port of the host computer with no routers, switches, or hubs in between.

```
Port of networked RVP10/IFD IQ stream
```

The parameter **Port of networked RVP10/IFD IQ stream** sets the port, or unique identification of the RDA application running on the IFDR10 IP address location.

```
IP address of networked RVP10/IFD gRCP
```

The question **IP address of networked RVP10/IFD gRPC** defines the Ethernet address where the RVP10 server finds the IFDR10's gRPC control interface.

```
Port of networked RVP10/IFD gRCP
```

The parameter **Port of networked RVP10/IFD gRPC** provides the RVP10 server the unique port identification of the IFDR10's controlling application.

Example of *profile.conf*:

```
install_usr_root=/usr
install_etc_root=/etc
data_root=/srv/iris_data
log_root=/var/log
network_port="TCP 30725"
operator=operator
operators="radarop operator sipan admin root"
observers="observer"
tsexport_port=30780
tsimport_port=30781
tsexport_ip=192.168.76.36
ifdr10_iq_stream_ip_addr=10.0.3.254
ifdr10_iq_stream_port=1001
ifdr10_grpc_ip_addr=10.0.2.254
ifdr10_grpc_port=5000
```

4.5 STALO control

The earlier generations of Vaisala RVPs used a legacy Sigmet digital automatic frequency control (DAFC) device for STALO control. RVP10 is using a new approach with off-the-shelf industrial automation products for STALO control. For controlling STALOs having RS-422 serial data interfaces, the Moxa NPort IA5150AI-T device is used. It receives control commands over the internal radar network from the RVP10 server, and converts them to RS-422 serial output. Currently, this is the only method provided for controlling STALOs, but other methods can be implemented on request.

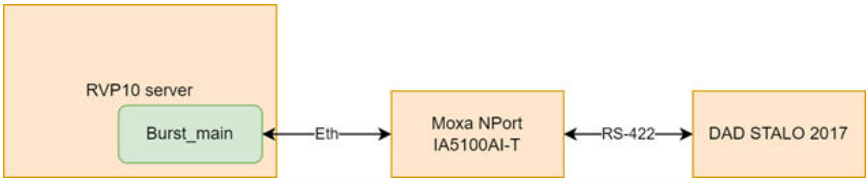


Figure 11 STALO control

5. Using RVP10

5.1 Configuring RVP10 with CLI

One interface for configuring RVP10 is the **dspx** utility, which has been present in Vaisala (Sigmet) RVP products for the past 30 years. **dspx** can be used, for example, for configuring RVP10 signal processing options, trigger timing, sector blanking, and AFC loops. For detailed information, see chapter *TTY non-volatile setups*. In addition, IFDR10 has configurations unique for the hardware, and a new command line interface via gRCP calls. In the future RVP10 releases, more and more configuration settings will migrate from the **dspx** utility to the IFDR10 configuration interface.

The IFDR10 configuration can be reviewed and changed using a series of command options from a terminal window. As the IFDR10 is hardware-secured, this is the only method of access into the embedded Linux operating system to make changes. To access the list of commands:

1. Log in as **radarop**.
2. Type:

   ```
   $ ifdr_
   ```

3. Press the **TAB** key twice.

   This produces a listing of all possible command line entries into the IFDR10.
   Example:

   ```
   ifdr_antenna_config_get
   ifdr_antenna_config_modify
   ifdr_antenna_config_set
   ifdr_get_device_info
   ifdr_settings_get
   ifdr_settings_modify
   ifdr_settings_restore
   ifdr_settings_save
   ifdr_settings_set
   ifdr_status
   ifdr_stop
   ifdr_timesync_config_get
   ifdr_timesync_config_set
   ```

4. To modify the configurations, use a **`get`** version of a command to retrieve the configuration state, and write the output to a file written in ***`json`*** format.

   a. Stop any controlling hosts, such as IRIS Radar, **Ascope**, or **Zauto** utility before making the request to get the configuration state.

      If the hosts are not stopped, IFDR10 will return an error when the request is made.

   b. After writing the configuration to a file, use a text editor (for example, **vi**) to modify and save the setting within the file.

   c. Finally, write the new configurations within the file to the IFDR10 using the **`set`** version of the command.

      For example, the following command writes the IFDR10's stored settings to an editable file named ***`output.file`***:

      ```
      $ ifdr_antenna_config_get > output.file
      ```

      The following command reads the contents of the ***`input.file`***, and if the contents match the expected name value pairs, IFDR10 accepts and employs the new content:

      ```
      $ ifdr_antenna_config_set < input.file
      ```

5. In some cases, a **`modify`** version of the command is available. This is useful for sending a new configuration into IFDR10 by an expression of changes pair as part of the argument.

   ```
   $ ifdr_antenna_modify ifdr_antenna_modify '.udp_client = {enabled: true,
    address: "10.0.2.1", port: 30745}'
   ```

More information

- Hardware security (page 43)
- IFDR10 hardware (page 42)

5.1.1 Device and status information and control

`ifdr_get_device_info`

The **`ifdr_get_device_info`** command shows the hardware, FPGA, and embedded software versions currently installed onto the IFDR10. The security key, mac address, and serial number are also shown.

```
EXAMPLE:
{
product: ifdr10,
version_hw: unknown,
  version_fpga: 0.20.0,
  version_sw_app: v0.19.1+58.2e2c6d5,
  version_fw: 0.0.1,
  build_time: 2022-11-22T18:55:34Z,
  image_name: devel-image,
  security_keyset: rdkeys-test,
  mac_address: unknown,
  serial_number: unknown
}
```

`ifdr_status`

The **`ifdr_status`** command shows if settings have been changed from the last power cycle, and whether IFDR10 is currently processing (creating data) or in an idle state.

```
{
  "settings_changed": true,
  "engine_state": "IDLE"
}
```

`ifdr_stop`

The **`ifdr_stop`** command forces IFDR10 into the **`IDLE`** state.

5.1.2 General configuration

`ifdr_settings_{get, modify, set, save, restore}`

The **`ifdr_settings_save`** command enables users to save the current configurations to a backup memory location on the IFDR10.

The **`ifdr_settings_restore`** command enables the retrieval of the saved settings from the backup location. Thus, if bad commands are sent into IFDR10 using the **`_set`** functions, the complete **`saved`** configuration from a previous point can be restored.

When performing the **`save`** and **`restore`** functions within the **dspx** utility, **dspx** also makes an **`ifdr_settings_save/restore`** call to ensure the IFDR10 configuration state is the same.

An example output from the **`ifdr_settings_get`** call is shown below. Users can directly change configurations for the **`run_triggers_in_idle_state`** and the **`rx_channels: function`** assignments.

Leave all other name value pairs within the **`ifdr_settings`** output unchanged. These are managed in the **dspx** utility.

Table 19 Configurable parameters in **`ifdr_settings`**

Name	Type	Description
`run_triggers_in_idle_state`	Boolean	Tells IFDR10 whether to provide triggering when the hardware is not processing ADC data into IQ. This may be beneficial to magnetron radar systems to ensure the transmitter continues radiating during short idle times between sweeps and scans.

Name	Type	Description
`rx_channels: function`	string	Sets the source of the receiver input to each of the 6 receiver inputs to the ADCs. `Unused` – this channel is not expected to have any input. `CoPol` – the channel is selected as the co-polarization receiver when used in a dual-polarization radar. By convention, the horizontally polarized transmission is the primary polarization. This becomes the H-channel receiver input. When RVP10 is used in single-polarization radars, `CoPol` is selected to designate this channel as the receive channel. `CxPol` – the channel designated to be the orthogonal polarization of the primary polarization. Typically, this is the vertical polarization receive channel. `CoCxPolBurst` – the channel is the burst pulse sample input from the horizontal and vertical transmit channel for a dual-polarization radar. The burst sample from a single transmitter is both the `Co-pol` and `Cross-polarization` sample in a simultaneous transmit and receive (STAR) radar. `CoPolBurst` – this designates a receive channel to be the co-polar burst pulse sample having same polarization as the primary polarization, typically horizontal. This is the use case for a single-polarization radar. In systems with two transmitters, each dedicated to a polarization channel, this is the burst pulse of the primary polarization. `CxPolBurst` – designates a receive channel as the cross-polarization burst pulse sample. Used in systems with two transmitters which trigger simultaneously or alternating. Typically, this is the vertical polarization burst pulse sample.

An example output from the ifdr_settings_get call:


```
{
  "_schema": "RdaSettings",
  "_schemaVersion": 1,
  "_serial": 7,
  "adc_clock_frequency": 235000000,
  "angle_poll_enabled": true,
  "blank_sectors": [
    {
      "annotation": "Set not to radiate above 20 degrees elevation per
city(?)",
      "az_end_deg": 360,
      "az_start_deg": 0,
      "el_high_deg": 95,
      "el_low_deg": 20,
      "enabled": true,
      "pedestal": false
    }
  ],
  "default_tx_sequence_id": "mt_0",
  "delay_reference_frequency": 240000000,
  "ext_clock_in_enabled": true,
  "ext_clock_in_frequency": 10000000,
  "ext_clock_out_enabled": false,
  "ext_clock_out_frequency": 10000000,
  "external_trigger_delay": {
    "tick": 0,
    "usec": 0
  },
  "external_trigger_on_rising": false,
  "gpio_configuration": {
    "gpio_pairs": [
      {
        "output": false,
        "pair_type_differential": false,
        "pin_n": {
          "function": "EMPTY",
          "inverted": false
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false
        }
      },
      {
        "output": false,
        "pair_type_differential": false,
        "pin_n": {
          "function": "EMPTY",
          "inverted": false
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false
        }
      },
      {
        "output": false,
        "pair_type_differential": false,
```

```
        "pin_n": {
          "function": "EMPTY",
          "inverted": false
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false
        }
      },
      {
        "output": false,
        "pair_type_differential": false,
        "pin_n": {
          "function": "EMPTY",
          "inverted": false
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false
        }
      },
      {
        "output": false,
        "pair_type_differential": false,
        "pin_n": {
          "function": "EMPTY",
          "inverted": false
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false
        }
      },
      {
        "output": false,
        "pair_type_differential": false,
        "pin_n": {
          "function": "EMPTY",
          "inverted": false
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false
        }
      },
      {
        "output": false,
        "pair_type_differential": false,
        "pin_n": {
          "function": "EMPTY",
          "inverted": false
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false
        }
      },
      {
        "output": false,
        "pair_type_differential": false,
```

```
        "pin_n": {
          "function": "EMPTY",
          "inverted": false
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false
        }
      }
    ],
    "trigger_io": [
      {
        "output": false,
        "pin": {
          "function": "EMPTY",
          "inverted": false
        }
      },
      {
        "output": false,
        "pin": {
          "function": "EMPTY",
          "inverted": false
        }
      }
    ]
  },
  "net_data_format": "frame",
  "polarization": "DualH",
  "pulses": {
    "mw_0": {
      "filter": {
        "center_freq": 60000000,
        "design_coeffs": [
          2400000
        ],
        "design_type": "sigmet",
        "edge": 0,
        "length_usec": 0.65,
        "window_type": "Rectangular"
      },
      "pulse_indexed": {
        "center_freq": 60000000,
        "index": 0,
        "length_usec": 0.65
      }
    },
    "mw_1": {
      "filter": {
        "center_freq": 60000000,
        "design_coeffs": [
          1350000
        ],
        "design_type": "sigmet",
        "edge": 6.938893903907228e-18,
        "length_usec": 1,
        "window_type": "Rectangular"
      },
      "pulse_indexed": {
        "center_freq": 60000000,
```

```
        "index": 1,
        "length_usec": 1
      }
    },
    "mw_2": {
      "filter": {
        "center_freq": 60000000,
        "design_coeffs": [
          1344000
        ],
        "design_type": "sigmet",
        "edge": 0.1,
        "length_usec": 1.1,
        "window_type": "Hamming"
      },
      "pulse_indexed": {
        "center_freq": 60000000,
        "index": 2,
        "length_usec": 1.1
      }
    },
    "mw_3": {
      "filter": {
        "center_freq": 60000000,
        "design_coeffs": [
          405000
        ],
        "design_type": "sigmet",
        "edge": 0.5,
        "length_usec": 2.4,
        "window_type": "Hamming"
      },
      "pulse_indexed": {
        "center_freq": 60000000,
        "index": 3,
        "length_usec": 2.4
      }
    }
  },
  "rpc_listen": "0.0.0.0:5000",
  "run_triggers_in_idle_state": false,
  "rx_channels": [
    {
      "function": "CoPol",
      "gain": 1,
      "phase_deg": 0
    },
    {
      "function": "CxPol",
      "gain": 1,
      "phase_deg": 0
    },
    {
      "function": "Unused"
    },
    {
      "function": "Unused"
    },
    {
      "function": "CoCxPolBurst",
```

```
      "gain": 1
    },
    {
      "function": "Unused",
      "gain": 1
    }
  ],
  "transmit_mode": "External",
  "transmit_sequences": {
    "mt_0": {
      "bfe_window": {
        "length_frac": 0.6999999999999997,
        "start_frac": 0.15000000000000002
      },
      "default_prf": 590.0000610351562,
      "min_range_bin_m": 50,
      "pulses": [
        "mw_0"
      ],
      "steps": [
        {
          "max_prf": 2200,
          "min_prf": 200,
          "min_prt": 416.6666667,
          "triggers": [
            {
              "length_usec": 1,
              "pull_up": true,
              "start_usec": -1.4999999999999938
            },
            {
              "length_usec": 1,
              "pull_up": true,
              "start_usec": -1.875
            }
          ]
        }
      ]
    },
    "mt_1": {
      "bfe_window": {
        "length_frac": 0.6999999999999997,
        "start_frac": 0.15000000000000002
      },
      "default_prf": 1000,
      "min_range_bin_m": 125,
      "pulses": [
        "mw_1"
      ],
      "steps": [
        {
          "max_prf": 1500,
          "min_prf": 200,
          "min_prt": 666.6666667,
          "triggers": [
            {
              "length_usec": 1,
              "pull_up": true,
              "start_usec": -1.4949991229999997
            },
```

```
            {
              "length_usec": 1,
              "pull_up": true,
              "start_usec": -1.4949991229999997
            }
          ]
        }
      ]
    },
    "mt_2": {
      "bfe_window": {
        "length_frac": 0.7499999999999998,
        "start_frac": 0.15000000000000002
      },
      "default_prf": 300.0000305175781,
      "min_range_bin_m": 125,
      "pulses": [
        "mw_2"
      ],
      "steps": [
        {
          "max_prf": 1200,
          "min_prf": 200,
          "min_prt": 833.3333333,
          "triggers": [
            {
              "length_usec": 1,
              "pull_up": true,
              "start_usec": -1.4999998569999986
            },
            {
              "length_usec": 0.125,
              "pull_up": true,
              "start_usec": -1.4999998569999986
            }
          ]
        }
      ]
    },
    "mt_3": {
      "bfe_window": {
        "length_frac": 0.6999999999999997,
        "start_frac": 0.15000000000000002
      },
      "default_prf": 590.0286254882812,
      "min_range_bin_m": 125,
      "pulses": [
        "mw_3"
      ],
      "steps": [
        {
          "max_prf": 600,
          "min_prf": 200,
          "min_prt": 166.6666667,
          "triggers": [
            {
              "length_usec": 1,
              "pull_up": true,
              "start_usec": -1.4200003149999985
            },
```

```
            {
              "length_usec": 1,
              "pull_up": true,
              "start_usec": -1.4200003149999985
            }
          ]
        }
      ]
    }
  },
  "trigger_count": 1,
  "trigger_is_external": false,
  "tx_channels": [
    {
      "function": "CoCxPol",
      "max_frequency": 64000000,
      "min_frequency": 56000000,
      "output_power_dbm": 1
    },
    {
      "function": "TestA",
      "output_power_dbm": -12,
      "test_delay_usec": 50,
      "test_signal": "CoCxPol"
    },
    {
      "function": "TestB",
      "output_power_dbm": -6,
      "test_signal": "CW",
      "test_signal_cw_freq": 60000000
    }
  ]
}
```


5.1.3 Antenna configuration

IFDR10 uses the multicast network interface to the shared antenna library in order to obtain antenna pointing information. It must be aware of the multicast address, port number, and interface. These must match those set in the RCP8 **antenna** menu of the **antx** utility.

Table 20 Configurable parameters for antenna configuration

Name	Type	Description
`address`	string	Multicast IP address. Vaisala recommends using the default address of **`224.0.0.3`**. This is a “link-local” address, which by default is not passed through routers.
`interface`	string	Specify which interface to use for RCP communication, **lan0** or **lan1**.

Name	Type	Description
`port`	string	Suggested default: **`30785`**.
`udp_client`	boolean	Enables a unicast interface. For lab use and testing. Not intended for use in fielded units.

Example:

```
{
  "_schema": "Antenna",
  "_schemaVersion": 1,
  "_serial": 0,
  "multicast": {
    "address": "224.0.0.3",
    "interface": "lan1",
    "port": 30785
  },
  "udp_client": {
    "enabled": false
  }
```

For more detailed information on the multicast interface, see chapter *Network Interface Configuration Options* in *IRIS and RDA Utilities Guide (M212925EN)*.

5.1.4 Chrony TimeSync configuration

ifdr_timesync_{get, set}

Time synchronization between the IFDR10 and RCP8 is essential to modeling the antenna location within the IFDR10. This model accounts for the lower availability rate of the antenna pointing information and time latency to then correctly tag each pulse with the correct location. IFDR10 uses Chrony as the synchronization tool.

The configurable parameters in the *`json`* table are described in the following table.

Table 21 Configurable parameters for time synchronization configuration

Name	Type	Description
`useNtp`	Boolean	Defines whether NTP may be used as a time source.
`ntpservers`	array	List of strings, each of which is an NTP server address. Addresses may be either **DNS** names or **IPV4** addresses

Name	Type	Description
`useGps`	Boolean	Defines whether GPS may be used as a time source. **`useGps`** is not supported in this release.

```
{
  "_schema": "TimeSynchronizationConfiguration",
  "_schemaVersion": 1,
  "_serial": 1,
  "ntpServers": [
    "10.0.2.2",
    “ntp.vaisala.com’
  ],
  "useGps": false,
  "useNtp": true
}
```

5.2 Diagnostics

5.2.1 IFDR10 system logs for diagnostics

To retrieve the contents of a system log from IFDR10, use the **`ifdr_rpc_get_logs`** command. When used without any options, this command fetches all the log records from the default IFDR IP address, and exits. One log record is printed as a single line of text.

The program prints log records to ***`stdout`***.

Syntax of the command:

```
$  ifdr_rpc_get_logs [-h | --help] [-r | --rda-rpc ADDRESS[:PORT]] [-f | --
follow] [-s | --since DATETIME] [-n | --num-last N] [-m | --max-records N]
```

Table 22 Command options for **`ifdr_rpc_get_logs`**

Option	Description
`-h, --help`	Print help message and exit.

Option	Description
-r, --rda-rpc ADDRESS[:PORT]]	Specifies the IFDR address to use for RPC communication. • **ADDRESS** = IFDR numeric IPv4 address in **a.b.c.d** format • **PORT** = gRPC service port number Option **-r** is only needed if the IFDR IPv4 address is changed from the default value. Specifying PORT should not be needed in normal use.
-f, --follow	Follow the log in real time as new records are being added. • If given, the program runs until it is interrupted, for example, by **Ctrl+C**. • If not given, the command exits after printing the log records.
-s, --since DATETIME	Get log records since **DATETIME**. • If neither **-s** nor **--since** is given, the program gets all the available log records. • **DATETIME** uses the 24h ISO format "year-month-day hours:minutes:seconds". For example: "2022-10-30 18:17:16". Do NOT use with **-n**
-n, --num-last N	Get **N** last records. Do NOT use with **-s**
-m, --max-records N	Put at most **N** log records into a single gRPC message (default: 100). Option **-m** should not be needed in normal use.

5.3 Configuring hardware

5.3.1 Defining IFDR10 input A/D saturation levels

Two analog signals must be supplied to the RVP10 IFDR:

- IF receiver signal
- IF Tx sample (burst pulse) for magnetron

Both inputs are on SMA connectors.

IF input and burst input

The A/D input saturation level for the IF input and burst input is +13.3 dBm.

In most installations, an external, anti-alias filter is installed on both of these inputs. These filters (when supplied by Vaisala) are mounted externally on one side of the IFDR, and have an insertion loss of 0.5 dB ... 1.0 dB.

For the burst pulse, it is important not to exceed the A/D saturation level. This reference signal should be strong enough, so that most of the bits in the A/D converter are used effectively, but it should also allow a few decibels below the saturation level for safety. The recommended signal power level is -3 dBm ... +4 dBm. This is important for making a precise phase measurement on each pulse.

See Burst pulse timing plot (Pb) (page 113).

IF receiver input

For the IF receiver input, it is useful to occasionally exceed the A/D input saturation level at the strongest targets. However, inputs must not exceed 20 dBm. RVP10 uses a statistical linearization algorithm to derive correct power levels from targets that are as much as 6 dB above saturation.

Establish the IF signal level by weak-signal and noise considerations, rather than by working backwards from the saturation level.

5.3.2 Configuring the IFDR10 clock subsystem

IFDR10 provides a programmable low jitter clock generator used in sampling the IF inputs and generating the IF outputs.

Table 23 Clock generator

Source	Description
Master clock source	The IFDR10 ADCs and DACs are clocked by internal VCXO (Voltage Controlled Crystal Oscillator), which can be optionally locked to an external reference.

Source	Description
IF sampling frequency	The sampling clock frequency is fully programmable 190 ... 240 MHz with microhertz (µHz) resolution. You can choose the clock frequency independently of the original reference clock frequency that produces it; they do not have to be small integer multiples of each other. See Choosing A/D sample rate or Tx synthesis rate (page 73) The IFDR10 sampling clock is derived from the master clock source. The architecture minimizes jitter, while allowing full flexibility in selecting sampling frequencies 190 ... 240 MHz. The output clock runs at the same frequency as the sampling clock.
Clock jitter	IF clock jitter is sub-picosecond allowing the system to maximize the benefits of the 16-bit A/D converters.

When RVP10 is used in a klystron system, or a synchronous radar, the radar COHO is supplied to the RVP10 IFDR, so that the sampling clock can digitally lock to it. The COHO phase is measured at the beginning of each transmitted pulse, and is used to lock the subsequent (**I**,**Q**) data for that pulse. The COHO phase is measured relative to the RVP10 IFDR internal stable sampling clock, which is user selectable. The internal sampling clock is not affected by the application of the COHO. A/D samples of the COHO are obtained at the fixed sampling rate, and the (**I**,**Q**) data are digitally locked downstream in the RVP10 IF-to-**I/Q** processing chain (see IFDR10: IF to I and Q processing (page 31)). The procedure is identical to the manner in which phase is recovered in a magnetron system, except that the COHO signal is used in place of a sample of the transmit burst.

There are two concerns that may occur when RVP10 is used in the above manner within a synchronous radar system. Both concerns are the result of the RVP10 IFDR sampling clock being asynchronous with the radar system clock.

Table 24 Clock generator concerns in a synchronous radar system

Concern	Description
RVP10 generates the radar trigger	The IFDR10 trigger signals are inherently synchronous with the data sampling clock, This is accomplished by a clock recovery PLL that provides on-board timing, which is identical to the sampling clock in the IFDR10. Since the IFDR10 sampling clock is asynchronous with the radar clocks, RVP10 trigger outputs are similarly asynchronous. The result is that each transmitted pulse envelope is triggered independently of the COHO phase. The transmitted pulse is still synchronous, but the precise alignment of the amplitude modulated envelope varies. In almost all cases, the exact placement of the transmitter's amplitude envelope does not affect the overall system stability, nor the ability of RVP10 to reject ground clutter and to process multi-mode return signals. For this reason, a synchronous radar system, which is triggered using RVP10 triggers, still performs optimally using the standard digital COHO locking techniques. In spite of this, some system designers may still prefer that the amplitude envelope be locked to the COHO.
RVP10 Receives the Existing Radar Trigger	When an external trigger is supplied to RVP10, the processor synchronizes its internal range bin selection circuitry to that external trigger. The placement of the range bins themselves, however, is always synchronous with the IFDR10 selectable sampling clock. The result is that 27.8 nanoseconds of jitter is introduced in the placement of RVP10 range bins relative to the transmitted pulse. The effect of this synchronization jitter is that targets appear fluctuate in range by approximately 4.2 m. Although this is small, relative to the range bin spacing, and does not affect the range accuracy of the data, the effect on overall system stability is more severe. Using both numerical modeling and field measurements, we have found that sub-clutter visibility of a µsec pulse may be limited to approximately 43 dB as a result of this 27.8 nanoseconds range jitter. This falls quite short of the usual expectations of a synchronous radar system in which clutter rejection of 55 ... 60 dB should be attainable.

The solution to these concerns is to provide a way for the IFDR10 internal sampling clock to be phase-locked to the radar system. If RVP10 provides the radar triggers, then those triggers become synchronous with the radar COHO. If RVP10 receives an external trigger, then its range bin clock is synchronous with that external trigger, and there is no synchronization jitter in the range bins.

The IFDR10 can lock its sampling clock to an external system clock reference through an SMA input. This results in an RVP10 that is fully synchronous with the existing radar timing.

5.3.2.1 Choosing A/D sample rate or Tx synthesis rate

The internal system clock, which samples the IF input signals and synthesizes the Tx output waveforms, can be configured to run at any frequency between 190 MHz ... 240 MHz.

The setup questions in the **Mc** menu select the sampling clock frequency and whether the clock is derived from a stable on-board crystal oscillator or from the external SMA reference. See **Mc**— top-level configuration (page 84).

The sample clock frequency affects many components of the radar and signal processing system.

Table 25 Sample clock frequency considerations

Consideration	Description
A/D quantization noise and dynamic range	The inherent SNR of the A/D converter chip is spread over a Nyquist band, whose width is determined by the sampling frequency. As the sampling frequency increases, the A/D quantization noise that is contained within a given Rx bandwidth decreases, which means that RVP10 becomes more quiet. The dynamic range varies linearly with sampling frequency.
Quantization of trigger timing and range bin placement	Triggers generated by RVP10 are specified by their start time in microseconds, width in microseconds, and polarity. However, there are triggers always produced that are deviant from these values by half of the sample clock for Tx timings, and by a sample clock for Rx timings. If you want the triggers to be precisely aligned down to the exact clock edge, the sample clock frequency should be chosen so that trigger edges fall on integer multiples of the clock period. Similarly, the range bin spacing is specified in meters and always within half a clock period of the ideal value. The bins can also be placed precisely in range, by choosing a clock period that is an integer multiple of the desired spacing.
Maximum length of FIR down-conversion filters	The FIR filters that compute (**I**,**Q**) time series from raw IF samples must process those samples at the acquisition clock rate. A filter of a given length in microseconds must contain a greater number of taps (coefficients) as the sample rate increases. For very long filters (more than 40 µsec), it is sometimes necessary to limit the clock rate in order to achieve the desired impulse response length. The **Mt<n>** and **Ps** menus are helpful in determining the maximum length filter that can be achieved for a given RVP10 processing mode (affected by single/dual polarization, range bin spacing, and so on).

Use the RVP10 setup menus to cross-check the above constraints. The system installer must choose a sample clock frequency that achieves the best set of trade-offs at each radar site.

5.3.3 Configuring external pre-trigger input

You can supply RVP10 with your own CMOS-level pre-trigger for installations in which adequate trigger control already exists.

1. In the *softplane.conf* file, configure the trigger input.

   The trigger input is provided on the IFDR10 GPIO.
   The trigger input threshold is 3.5 V and can tolerate a 5 V maximum input when DC coupled and can accept either single-ended TTL or a differential pair as trigger source.
   The rising or falling edge of this external trigger signal is interpreted by RVP10 as the pretrigger point. The pulse width of the signal does not matter.

2. Configure the delay to `range0` in the TTY Setups.

   See **Mt<n>**— Transmit sequence #n (page 95).

3. Synchronize the other trigger outputs to the input trigger.

   The synchronization jitter between the user pretrigger and the other trigger outputs are less than the period of the A/D sampling clock, for example, 5 nanoseconds at a 240 MHz rate.
   In coherent systems, you can improve trigger jitter by phase locking the RVP901 IFDR to the same reference clock used to generate the external triggers (typically the COHO). The improved IF samples may provide as much as 10 dB of additional sub-clutter visibility.

5.3.4 Choosing sampling frequency and intermediate frequency

RVP10 does not assume a particular relationship between the A/D sampling frequency and the receiver's intermediate frequency. You can operate at any IF that is at least 2 MHz away from any multiple of half sampling rate. If the IF bandwidth is higher than ~2 MHz (for example, with hybrid pulsing), then IF needs to be ~4 MHz away.

The RVP10 sampling frequency can be set at 5 MHz steps. There are free web tools available for the frequency planning, such as **Frequency Folding Tool** by www.analog.com. For example, the following sampling frequencies are optimal for 8 MHz IF bandwidth:

IF frequency / MHz	Sampling frequency / MHz
60	210
57.5	205
30	250

5.3.5 IF bandwidth and dynamic range

Components of the receive signal path, particularly the IF filter bandwidth, should be wide enough so that they do not distort the receive signal. Both amplitude and phase response at the signal bandwidth are important.

IFDR10 includes an external analog IF filter at each IF and burst sample input. The IF filters prevent aliasing of noise at undesired frequencies during the A/D conversion into the desired receive signal. Vaisala provides IFDR10 with IF filters with 14 MHz nominal pass band width at IF center frequencies 30, 57.5 and 60 MHz. For example, the 60 MHz filter pass band is 53 MHz ... 67 MHz.

IFDR10 uses high-performance 16-bit A/D converters with a typical wide band SNR of 75.6 dB. The maximum power level at the input of the IFDR is +11.2 dBm.

The noise density at 1 MHz bandwidth using 210 MHz sampling rate can be calculated as follows:

$$+12dbm - 75.6dB - 10\log 10\left(\frac{105MHz}{1MHz}\right) = -84dbm \quad (at\ 1MHz\ BW)$$

RVP10 uses statistical linearization performed on signals that exceed the saturation level. RVP10 can recover the signal power accurately, even when the A/D converter is driven beyond saturation. The velocity data is also valid, but spectral width may be overestimated. This improves the dynamic range by 4 dB.

The weak coherent signal below noise floor at -4 dB SNR can easily be measured when 25 or more pulses are used. This improves the dynamic range by 4 dB compared to the signal level at SNR 0 dB.

As a summary, the total dynamic range of the IFDR10 is:

$$+12dbm + 4dB + 4dB - (-84dbm) = 104dB$$

1 MHz bandwidth corresponds to approximately a 1-µsec transmit pulse. With a 2-µsec transmit pulse, the dynamic range is 3 dB higher. With a 0.5-µsec transmit pulse, the dynamic range is 3 dB lower.

5.3.6 Configuring receiver gain

The configuration of the receiver gain involves a trade-off between receiver sensitivity and dynamic range. As the gain gets higher, the sensitivity gets better, but the dynamic range gets lower. The higher gain improves the noise figure and, thus, the sensitivity of the receiver. However, the A/D converter also begins to saturate at weaker signal levels at the receiver input. This trade-off is illustrated in figure Figure 12 (page 77).

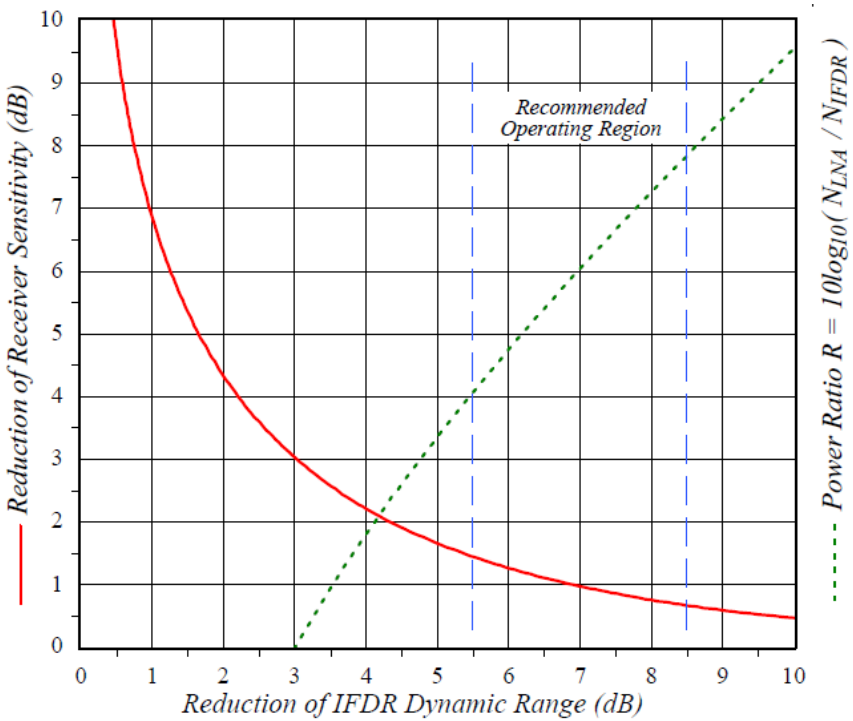


Figure 12 Trade-off between dynamic range and sensitivity

N_{IFDR} represents the stand-alone (terminated input) noise power of the IFDR over some bandwidth.

N_{LNA} represents the Rx path (LNA/Mixer) thermal noise power over that same bandwidth at the input of the IFDR.

Calculation example:

Select $\frac{N_{LNA}}{N_{IFDR}} = 6dB$ (1 dB reduction IN sensitivity, and 7 dB reduction in IFDR dynamic range).

The desired noise level of the Rx parts at the IFDR input is as follows:

$$N_{LNA}(dBm) = N_{IFDR}(dBm) + 6dB = -84dBm/MHz + 6dB = -78dBm/MHz$$

If the noise figure NF(dB) of the Rx parts is 2 dB, then the desired Rx gain is as follows:

$$Rx\ gain = \ -78dBm/MHz - \left(-174dBm/Hz + \ 10 * \log_{10}(1Mhz/Hz) \ + \ (NF(dB)\right) \ = \\ -78 - \left(-114 + \ 60 + 2\right) \ = \ 34dB$$

, where -174 dBm/Hz is the noise level of a matched termination at room temperature, and 1 MHz is the noise bandwidth of the signal.

The dynamic range estimate is $104dB - 7dB \ = \ -97dB(at\ 1MHzBW)$

When designing your RF and IF components, remember that the final amplifier driving the IFDR10 must be capable of driving sometimes up to +17 dBm, for example, so that signals above saturation can be correctly measured.

After assembling all the RF and IF components, check whether you have the correct gain by verifying a 7 dB rise (independent of bandwidth) in RVP10 filtered power, when the IF- Input cable is connected and disconnected. (N_{IFDR} = -84 dBm, N_{LNA} = -78 dBm, (N_{LNA}+N_{IFDR}) = -77 dBm, resulting in the 7 dB difference).

6. TTY non-volatile setups

6.1 Using the TTY setup menu

You can view and modify most RVP10 operating parameters with the TTY setup menu. For example:

- Make TTY connections
- Save and restore configurations
  For example, you can configure custom trigger patterns, pulse width control, matched FIR filter specs, PRF, and so on, in the field.
- Access graphical setup and monitoring procedures that use an ordinary oscilloscope as a synthesized visual display.
  The burst pulse and receiver waveforms are displayed in the time and frequency domain. The digital FIR filter matches the characteristics of the transmitted pulse.

1. To access the TTY menu:

   a. In the serial TTY or host computer interface, type: **dspx**

      The following prompt is shown:

      ```
      $dspx
      Digital Signal Processor 'Chat'
      Checking for code upgrades...Okay
      (Type ^C to exit Chat Mode)
      ```

   b. On the TTY, press **Esc**.

      RVP10 command prompt is shown (example):

      ```
      Vaisala, Oyj
      RVP10 Digital IF Signal Processor V15 IRIS-10.0.0-rc1
      ------------------------------------------------
      Intermediate Frequency Digital Receiver.
      IFDR firmware version 1.0.0
      ---------------------------------------
      RVP10>
      ```

2. To view RVP and IRIS software versions, type **V**
3. For a list of available commands, type **help**

4. To exit the menu and reload RVP10 with the changed set of current values, type: **Q**

   Settings are saved in a configuration file so they take immediate effect on start-up.

   You must exit the menus using the **Q** before resuming normal RVP operation. However when exiting menu with Q, changes that may have been made are not transferred from the RPV10 Server to the IFDR10.
   Portions of the RVP command interpreter continue to run while the menus are active, but the processor does not function until you exit the menus.

5. To return to the Linux command prompt, press **Ctrl+C**.

6.1.1 Factory settings, saved settings, and current settings

RVP settings included:

- Current settings—Values with which RVP10 is currently operating.
- Saved settings—Values stored in a permanent configuration file. The saved settings are restored (made current) each time RVP10 starts.

RVP10 retains the saved settings when new software releases are installed. The new version of the code automatically uses all of the previous saved values. If RVP10 detects a new setup parameter, it is set to a factory default value and a warning is printed if this occurs. See **V**and**Vz**– View system status (page 81).

Table 26 Setting commands

Command	Description
S	Saves the current settings into the rvp10.conf configuration file.
R	Restores those non-volatile values so that they become the current settings.
F	Initializes the current settings with factory default values of RVP10. The factory default values do not correspond to any user installation, and they are not meant to be applied in normal situations.
F S	Saves factory defaults in non-volatile RAM. The site-specific power up settings are overwritten, irreversibly, and RVP10 powers up in its manufacturing mode. **This is not recommended.**
V	Shows the currently running versions of RVP and IRIS software.

You must exit the menus using the **Q** before resuming normal RVP operation. However when exiting menu with Q, changes that may have been made are not transferred from the RPV10 Server to the IFDR10.
Portions of the RVP command interpreter continue to run while the menus are active, but the processor does not function until you exit the menus.

6.1.2 **V** and **Vz** – View system status

The **V** command displays internal diagnostics.

This information is for inspection only, and cannot be changed from the TTY.

If invoked as **Vz**, the counters are cleared, so that subsequent **V** commands show what has accumulated since the last **Vz**.

The view listing displays:

```
Configuration and Internal Status
---------------------------------
---------------------------------
  Vaisala, Oyj
RVP10 Digital IF Signal Processor V15 IRIS-10.1.0
------------------------------------------------------------
  Settings were last saved using V15
  RVP10 started at: 12:07:44 24 MAY 2024
  Current time is: 15:18:20.117 24 MAY 2024
```

This group displays status of the RVP10 server. The line `Settings were last saved using...` displays the version of RVP10 code that was the last to write into the non-volatile RAM. It is printed only if that last version was different from the version that is currently running.

The lines `RVP10 started at...` and `Current time is...` display information about when RVP10 was started, current system time, and implicitly, the uptime.

```
Intermediate Frequency Digital Receiver.
  IFDR firmware version 1.1.0
-----------------------------------------

IPP-Library: ippSP AVX2 (l9 threaded) 8.2.0 (r43166) 8.2.0.43166
Intel CPU capabilities:MMX/SSE/SSE2/SSE3/SSSE3/MOVBE/SSE41/SSE42/AVX/OS-
AVX/AES/SLMUL/
```

This group provides hardware status of the IFDR10 device. The line `IPP-Library` shows information about the processor and the Intel libraries used for RVP10 processing and information of the IFDR10 CPU is displayed.

```
Diagnostics: PASS
```

Line `Diagnostics`: If errors were detected by the startup diagnostics, then an error bitmask is shown. **PASS** indicates that no errors were detected.

```
Processes and Threads:
RVP10Proc-0  -   PID:6917Priority:10 Policy:RealTimeRR
RVP10Proc-1  -   PID:6918  Priority:10 Policy:RealTimeRR
  Chat/Plot   -   PID:6909Priority:10 Policy:RealTimeRR
  Watchdog    -   PID:6909Priority:10 Policy:RealTimeRR
  Burst/AFC   -   PID:6909Priority:10 Policy:RealTimeRR
  HostCmds    -   PID:6909Priority:11 Policy:RealTimeRR
    IQ-Data     -   PID:6909  Priority:13 Policy:RealTimeRR
```

The **Processes and Threads** list displays RVP10 processes and their related priority. All RVP10 processes and threads should run under **RealTimeRR** policy to guarantee adequate attention from the processors.

```
Shared library build dates:
RVP10/Main/Core:MonApr216:01:12EDT2022
RVP10/Main/Open:MonApr216:01:13EDT2022
RVP10/Main/Site:MonApr216:01:12EDT2022
RVPX/Proc/Core:MonApr216:01:14EDT2022
RVPX/Proc/Open:MonApr216:01:16EDT2022
RVPX/Proc/Site:MonApr216:01:14EDT2022
```

The **Shared library build dates** section provides RVP10 developers with information about code resources.

```
GPS:Unused
```

The **GPS** line shows the status of optional GPS time synchronization of the RVP10 triggers and range bins.

```
AFC:0.00% (Disabled), Burst Pwr:-48.6 dBm, Freq:30.000 MHz
```

AFC indicates the level and status of the AFC voltage at the IFDR module. The number is the present output level in D-Units ranging -100 ... +100. The shorter **%** symbol is used since percentage units correspond in a natural way to the D-Units.

Burst Pwr indicates the mean power within the full window of burst samples. DC offsets in the A/D converter do not affect the computation of the power, that is, the value shown truly represents the (signal + noise) energy of the waveform. **Freq** indicates the mean frequency of the burst, derived from a fourth order correlation model.

```
Tx seq. mt_0 , main pulse mw_0 burst pulse levels - PriRx:   0.04 dBm,
SecRx:  -1.96 dBm
Tx seq. mt_1 , main pulse mw_1 burst pulse levels - PriRx:  -0.02 dBm,
SecRx:  -2.03 dBm
Tx seq. mt_2 , main pulse mw_2 burst pulse levels - PriRx:   7.33 dBm,
SecRx:   5.47 dBm
Current second mw_1 burst pulse levels - PriRx:  -0.46 dBm,  SecRx:  -2.47 dBm
Tx seq. mt_3 , main pulse mw_3 burst pulse levels - PriRx:  12.95 dBm,
SecRx:  11.04 dBm
```

The following session is only displayed if the RVP10 is configured to provide transmit waveforms to a coherent transmitter. A summary of the burst pulse powers that are measured for each transmit sequence is provided.

6.1.3 **Vp** – View processing and threshold values

The **Vp** command displays internal parameters that affect the moment processing within RVP10. This information is for inspection only and cannot be changed from the TTY.

The threshold parameters are **LOG** (receive power above noise), **CSR**, weather signal power (**WSP**), **SQI**, and Polarimetric Met Index (**PMI**). The threshold control flags (**TCF**) column shows numeric and symbolic forms.

```
Threshold Settings for All Data Parameters
       LOG       CSR       WSP       SQI       PMI       TCF (Equation)
       ---       ---       ---       ---       ---       --------------

DBZ:    0.75dB    -18.0dB    5.0dB    0.149    0.449    0x8888( LOG& CSR )
DBT:    0.75dB    -18.0dB    5.0dB    0.149    0.449    0xFFFF( AllPass )
VEL:    0.75dB    -18.0dB    5.0dB    0.398    0.449    0xC0C0( CSR& SQI )
WID:    0.75dB    -18.0dB    5.0dB    0.398    0.449    0xA000( LOG& SQI &SIG )
ZDR:    0.75dB    -18.0dB    5.0dB    0.398    0.449    0xAAAA( LOG )
KDP:    0.75dB    -18.0dB    5.0dB    0.398    0.449    0xC0C0( CSR& SQI )
PHIDP:  0.75dB    -18.0dB    5.0dB    0.398    0.449    0xC0C0( CSR& SQI )
RHOHV:  0.75dB    -18.0dB    5.0dB    0.398    0.449    0xC0C0( CSR& SQI )
SQI:    0.75dB    -18.0dB    5.0dB    0.398    0.449    0xFFFF( AllPass )
LDRH:   0.75dB    -18.0dB    5.0dB    0.398    0.449    0xAAAA( LOG)
RHOH:   0.75dB    -18.0dB    5.0dB    0.398    0.449    0xC0C0( CSR& SQI )
PHIH:   0.75dB    -18.0dB    5.0dB    0.398    0.449    0xC0C0( CSR& SQI )
LDRV:   0.75dB    -18.0dB    5.0dB    0.398    0.449    0xAAAA( LOG )
RHOV:   0.75dB    -18.0dB    5.0dB    0.398    0.449    0xC0C0( CSR& SQI )
PHIV:   0.75dB    -18.0dB    5.0dB    0.398    0.449    0xC0C0( CSR& SQI )
HCLASS: 0.75dB    -18.0dB    5.0dB    0.398    0.449    0xC0C0( CSR& SQI )
SNR:    0.75dB    -18.0dB    5.0dB    0.398    0.449    0xFFFF( AllPass )
DBZA:   0.75dB    -18.0dB    5.0dB    0.149    0.449    0x8888( LOG& CSR )
DBTA:   0.75dB    -18.0dB    5.0dB    0.149    0.449    0xFFFF( AllPass )
DBZE:   1.00dB    -50.0dB    0.0dB    0.130    0.449    0xA0A0 ( LOG & SQI )
DBTE:   1.00dB    -50.0dB    0.0dB    0.130    0.449    0xFFFF ( All Pass )
PMI:    1.00dB    -50.0dB    0.0dB    0.359    0.449    0xFFFF ( All Pass )
LOG:    1.00dB    -50.0dB    0.0dB    0.359    0.449    0xFFFF ( All Pass )
CSR:    1.00dB    -50.0dB    0.0dB    0.359    0.449    0xFFFF ( All Pass )
XCOR:   1.00dB    -50.0dB    0.0dB    0.359    0.449    0xFFFF ( All Pass )
AH:     1.00dB    -50.0dB    0.0dB    0.359    0.449    0xFFFF ( All Pass )
AV:     1.00dB    -50.0dB    0.0dB    0.359    0.449    0xFFFF ( All Pass )
AZDR:   1.00dB    -50.0dB    0.0dB    0.359    0.449    0xFFFF ( All Pass )
```

6.1.4 @ - Displaying and changing current major mode

The **@** command provides developers with a simple way of switching modes enabling on–the–fly testing of code.

For information on the top level RVP10 see **SOPRM** command word #9 in Setup operating parameters (**SOPRM**) (page 368). This question allows you to use the mode that has been selected by that command, or to force the use of a particular mode.

6.2 Managing settings with **M** menus

You can use the **M** menus to view and modify current settings.

Type **??** to print the entire set of questions.

The **M** menu works from the current parameter values, not from the saved values stored in the *rvp10.conf* location. If the host computer has modified some current values, you see these changes as you browse the setup list.

Vaisala recommends making incremental changes to saved settings.

1. Type **R** to restore the saved values from *rvp10.conf*.
2. Type **M** to make the changes starting from that point.

   The current value of each parameter is displayed on the screen. The TTY pauses for input at the end of the line:
   - Press **Enter** to move to the next parameter and leave the present parameter unchanged.
   - Type **U** to move back up in the list.
   - Type **Q** to exit the list at any time.
   - Type a number or a **YES/NO** response to change the parameter's value. The system displays the line again with the new value.
     An error message is displayed if entries are invalid.
3. Type **S** to save the new values.

You must exit the menus using the **Q** before resuming normal RVP operation. However when exiting menu with Q, changes that may have been made are not transferred from the RPV10 Server to the IFDR10.
Portions of the RVP command interpreter continue to run while the menus are active, but the processor does not function until you exit the menus.

6.2.1 **Mc** — top-level configuration

You can use this set of commands to configure general properties of IFDR10.

```
This version is compatible with rvp9: YES/NO
```

The question **This version is compatible with rvp9** defines if the parameter set of saved values being stored in the *rvp10.conf* file is backwards compatible with the RVP9 parameter set. Setting this to **YES** also limits the RVP10 to be functionally equivalent to the RVP900.

```
IFD synthesized system clock: 210.0000000 MHz
```

The question **IFD synthesized system clock** defines the frequency of the acquisition sampling clock in the IFDR module. The clock is synthesized on-board using a low-jitter PLL that allows many frequencies to be generated within the span of 181 – 240 MHz. The IFDR10 will select the closest frequency it can reproduce from the value entered.

```
IFD clock is derived from an external input reference (10.000000000 MHz): YES
```

The question **IFD clock is derived from an external reference** defines whether the synthesized system clock is derived from an internal crystal oscillator or from an externally applied reference clock.

```
PWINFO command enabled: NO
```

The question **PWINFO command enabled** enables use of the PWINFO operations via the controlling host interface into the RVP10. This is currently used internally for testing and should be left to NO for operational use.

```
IFDR is using angles from antenna library : YES
```

The question **IFDR is using angles from antenna library** defines the source of antenna angles used by the IFDR10 as the shared RCP8/IRIS/RDA antenna library.

```
Output external reference clock enabled (10.000000000 MHz): YES
```

Limits: 0.1 MHz ... 1000 MHz

To define the external input reference frequency, modify the *ext_clock_out_frequency* field in the *settings.json* file.

The question **Output external reference clock enabled** defines if the IFDR10 is capable of generating a sinusoidal oscillation to be used as a clocking reference. This parameter enables the output of this signal and sets the signal frequency.

```
Polarization:
 0: Circular, left is co-pol
 1: Circular, right is co-pol
 2: Dual, H is co-pol
 3: Dual, V is co-pol
 4: Single H
 5: Single V
Polarization <Dual, H is co-pol> : 2
```

The **polarization** section defines what type of polarization is in use within the radar system. The standard practice in weather radars is using the linear dual-polarization where the horizontal is the primary channel. However, circular polarizations, single-polarization, or declaring the vertical channel as primary are all supported.

```
Enable Wide Dynamic Range: YES
Rx Channels:
 0: CoPol
 1: CxPol
 2: Unused
 3: Unused
 4: CoCxPolBurst
 5: Unused

Co-pol low gain idx. Unused : [2, 3, 5]. Currently selected : 2
Separation : 30.00 dB
Phase : 0.00
Cx-pol low gain idx. Unused : [5]. Currently selected : 3
Separation : 30.00 dB
Phase : 0.00
Rx Channels:
 0: CoPol
 1: CxPol
 2: CoPolLowGain, Separation:  30.00, Phase:   0.00
 3: CoPolLowGain, Separation:  30.00, Phase:   0.00
 4: CoCxPolBurst
 5: Unused
```

When you enable the **Wide Dynamic Range** mode in RVP10, a listing of the current configuration of the Rx Channel assignment appears. This listing is informational only.

After this, the **Co-pol low gain idx.** question requests which channel should be assigned as the low-gain channel for the co-polar polarization. A listing shows which channels are currently unassigned. Select one of these channels for assignment.

Separation: Set the gain separation.

Phase: Set the phase difference between the high-gain and low-gain channels for the co-polar polarization.

Cx-pol low gain idx defines the channel to be assigned as the low-gain channel for the cross-polar polarization, with the same set of questions as above.

When you have completed these questions, the table of channel assignments is then again presented as an information summary.

You must determine the **Channel separation** and **Overlap/Interpolateinterval** in the **Pr** printout. Sweep a signal generator across the shared power region of the two channels to determine a representative channel separation, along with the size of the overlap region at the top of the high-gain channel within which that separation remains steady and constant, that is, unaffected by eventually approaching the noise floor of the low-gain channel. In the Vaisala weather radars, the differential gain and phase between channels is determined during the factory acceptance testing. See Wide dynamic range considerations (page 167) for more information on using wide dynamic range.

6.2.2 **Mb** — Burst pulse and AFC

Type **Mb** to view and manage parameters that influence the phase and frequency analysis of the burst pulse, and the operation of the AFC feedback loop.

When set to **YES**, the functions that implement phase locking and tracking as well as AFC and MFC functionality are applied only in the primary Rx channel.

Frequency and power parameters

The following questions define the centers of the transmit and receive intermediate frequency bands. Although the Tx and Rx intermediate frequencies are usually the same, you can choose them separately so that the RF up-conversion chain for transmission can be different from the down-conversion chain for reception.

Limits: 1 ... 120 MHz

```
Tx Intermediate Frequency: 30.0000 MHz
Rx Intermediate Frequency: 30.0000 MHz
```

The intermediate frequency is derived at the receiver's front-end by a microwave mixer and sideband filter. The filter passes either the lower sideband or the upper sideband, and rejects the other.

```
IF increases for an approaching target: YES
```

Depending on the chosen sideband, an increase in microwave frequency may increase (STALO below transmitter) or decrease (STALO above transmitter) the receiver's intermediate frequency. This parameter influences the sign of the Doppler velocities computed by RVP10.

```
Use the same channel for Rx and Burst Pulse Sampling: YES
```

In the past front-end receiver designs, it was common to separate the Rx channel and burst pulse sampling channel into two distinct lines. In Vaisala WRS300 and WRS400 weather radar systems, which introduce continuous degradation and calibration monitoring of the radar system, the burst pulse sample and Rx channel are combined onto a single input. IFDR10 then has an internal switching within the FPGA to send data from either channel to the RVP10 server. This question lets RVP10 become aware which front-end receiver design is being employed.

The question **`Use the same channel for Rx and Burst Pulse Sampling`** defines the channel used for Rx and burst pulse sampling:

- Type **`NO`** to use the default **`Burst Sample Input`** of **`ADC-E`**.
- Type **`YES`** to sample the burst pulse on the channel configured as the IF signal input. See IFDR10 inputs (page 47).
  In this configuration, the IF signal and the burst pulse signal are received on the same ADC channel.

```
PhaseLock to the burst pulse: YES
```

The **`PhaseLock to the burst pulse`** question controls whether RVP10 locks the phase of its synthesized **`I`** and **`Q`** data to the measured phase of the burst pulse.

- Type **YES** for an operational magnetron system, since the transmitter's random phase must be known to recover Doppler data.
- Type **AUTO** for phase locking only when the burst pulse power exceeds the power defined in the '**Minimum power for valid burst pulse** question.
- Type **NO** for non-phase modulated Klystron systems in which the IFDR sampling clock is locked to the STALO.
  NO is also useful for bench testing. In these cases, the phase of **I** and **Q** is determined relative to the stable internal sampling clock in the IFDR module.

```
Minimum power for valid burst pulse: -15.0
```

Limits: -60 ... +10 dBm

The question **Minimum power for valid burst pulse** is the minimum mean power required in the burst pulse for it to be considered valid and suitable for input into the algorithms for frequency estimation and AFC. For information on the reporting burst pulse power, see Burst pulse timing plot (Pb) (page 113).

The mean power level of the burst is computed within the narrowed set of samples used for AFC frequency estimation. The narrow pane contains only the active portion of the burst, and thus a mean power measurement is meaningful. The full FIR pane includes the leading and trailing pulse edges and would not produce a meaningful average power. Since radar peak power tends to be independent of pulse width, this single threshold value can be applied for all pulse widths.

The value entered here should be approximately 8 dB less. This makes sure that burst pulses are properly detected even if the transmitter power fades slightly.

```
Design/Analysis Window- 0:Rect , 1:Hamming, 2:Blackman : 1
```

With the question **Design/Analysis Window**, you can choose the analysis window used in the design of the FIR matched filter and the presentation of the power spectra for the scope plots.

The accuracy for the question **Design/Analysis Window** cannot be achieved with only one pulse. However, several hundred (unbiased) individual estimates can be averaged to produce an accurate mean. This averaging is done with an exponential filter with the time constant chosen here.

Table 27 Analysis window options

Option	Recommended use
Rectangular	Included as a teaching tool. Do not use for operation.
Hamming	Best overall choice.

Option	Recommended use
Blackman	Useful for seeing plotted spectral components more than 40 dB below the strongest signal present. Useful in the **Pr** plot when a long span of data is available. FIR filters designed with the Blackman window have a greater stopband attenuation than those designed with the Hamming window, but the wider main lobe may be undesirable.

```
Settling time (to 1%) of burst frequency estimator: 5.0 sec
```

Limits: 0.1 ... 120 s

The burst frequency estimator uses a fourth-order correlation model to estimate the center frequency of the transmitted pulses. Each burst pulse typically occupies approximately 1 µsec. The frequency estimate feeding the AFC loop must be accurate to approximately 10 KHz.

AFC and MFC parameters

```
Enable AFC and MFC functions: YES
```

AFC is required in magnetron systems to maintain the fixed intermediate frequency difference between the transmitter and the STALO.

AFC is not required in coherent transmitter systems, such as Klystron systems, because the transmitted pulse is inherently at the correct frequency.

The following AFC questions appear only if you enable the AFC and MFC functions:

```
AFC/MFC control server address : 127.0.0.1
```

```
AFC/MFC control server port : 4001
```

These questions set the IP address and unique application port where the RVP10 server may find the STALO controller.

AFC loop

```
Wait time before applying AFC: 10.0 sec
AFC hysteresis -- Inner: 5.0 KHz, Outer: 15.0 KHz
```

Limits: 0 ... 300 s

After turning on a magnetron transmitter, it can take several seconds or minutes until the output frequency is stable. The AFC loop does not need to run during this time. Use the question **`Wait time before applying AFC`** to set a holdoff delay from the time that valid burst pulses are detected to the time that the AFC loop begins running.

In general, the AFC feedback loop is active only when RVP10 is not processing data rays. This is because the Doppler phase measurement seriously degrades when the AFC control voltage makes a change. To avoid this, the AFC loop can only run between intervals of sustained data processing. This is fine as long as the host computer allows a few seconds of idle time every few minutes. If the RVP10 were constantly busy, the AFC loop would never have a chance to run.

The loop applies active feedback when the outer frequency limit is exceeded, but holds a fixed level once the inner limit has been achieved. The hysteresis zone minimizes the amount of thrashing done by the feedback loop. The AFC control voltage remains constant most of the time; making small and brief adjustments only occasionally as the need arises.

```
AFC outer tolerance during data processing: 50.0 KHz
```

Limits: 15 ... 4000 KHz

The question **`AFC outer tolerance during data processing`** defines the frequency error tolerances for the AFC loop by placing an upper bound on the frequency error that is tolerated during sustained data processing. AFC is applied when this limit is exceeded.

AFC loop feedback computations

```
AFC feedback slope:    0.0100 D-Units/sec / KHz
AFC minimum slew rate: 0.0000 D-Units/sec
AFC maximum slew rate: 0.5000 D-Units/sec
AFC span- [-100%,+100%] maps into [ -32768 , 32767 ]
```

The control applied to the AFC is specified in **`D-Units`**, that is, arbitrary units ranging from –100 ... +100 corresponding to the complete span.

Since the **`D-Units`** correspond to a percentage scale, the shorter **`%`** symbol is sometimes used.

AFC feedback is applied in proportion to the frequency error that the algorithm is attempting to correct. The feedback slope determines the sensitivity and time constant of the loop by establishing the AFC rate of change in (D–Units/sec) per thousand Hertz of frequency error. For example, a slope of 0.01 and a frequency error of 30 KHz results in a control voltage slew of 0.3 D–Units per second. At that rate it would take approximately 67 seconds for the output voltage to slew one tenth of its total span (20 D–Units / (0.3 D–Units / sec) = 67 sec). AFC is intended to track slow drifts in the radar system, so response times of this magnitude are reasonable.

Note that the feedback slew is based on a frequency error that is derived from a time averaging process (see burst frequency estimator **`Settling Time`** above).

The AFC loop becomes unstable if a large feedback slope is used with a long **`Settling Time`** constant, due to the phase lag introduced by the averaging process. Keep the loop stable by choosing a small enough slope that the loop easily comes to a stop within the inner hysteresis zone.

Burst parameters

```
Burst frequency increases with increasing AFC voltage: YES
```

If the frequency of the transmit burst increases when the AFC control voltage increases, type **Yes**. Otherwise type **No**.

When this parameter is set correctly, a numerical increase in the AFC drive (D–Units) results in an increase in the estimated burst frequency. If the AFC loop is completely unstable, try reversing this parameter.

```
Enable Burst Pulse Tracking: YES
```

The question **Enable Burst Pulse Tracking** enables the burst pulse tracking algorithm (see Burst pulse tracking (page 161)). The characteristic settling times for the burst are defined elsewhere in this menu, and the tracking algorithm uses dynamic thresholds to control the feedback.

```
Enable Time/ Freq  hunt for missing burst: Yes
 Number of frequency intervals to search: 5
 Settling time for each frequency hop: 0.25 sec
Automatically hunt immediately after being reset: YES
    Repeat auto-hunt every: 60.00 sec
```

This option is only available when the AFC and MFC function is enabled. These parameters configure the process of hunting for a missing burst pulse.

- The trigger timing interval that is checked during Hunt Mode is always the maximum ±20 µsec. No further setup parameters are needed to define the hunting process in time.
- For the hunt in frequency, the overall frequency range is always the full -100 … +100% AFC span.
  The number of sub-intervals to check must be specified, along with the STALO settling time after making each AFC change.

With the default values shown, AFC levels of -66%, -33%, 0%, +33%, and +66% are tried, with a one–quarter second wait time before checking for a valid burst at each AFC setting.

```
Enable burst power based correction of Z0: NO
```

The corrected calibration reflectivity numbers to do this are stored with the time series. To apply the correction, turn on the **OPTS_ZCAL** flag in the **SOPRM** command. See **XARG 6** in Setup operating parameters (**SOPRM**) (page 368).

```
Time constant of mean burst pulse power estimator: 500 pulses
```

The question **Time constant of mean burst pulse power estimator** defines how many pulses are used when computing the mean burst pulse power. The per pulse power based corrections of Z_0 are the difference between the individual burst pulse power sample and this mean power.

```
Simulate burst pulse samples: NO
```

RVP10 can simulate a 1 µsec envelope of burst samples.

NOTICE! Use the parameter **`Simulate burst pulse samples`** as a testing and teaching aid only. Never use this in an operational system.

A two–tone simulation is produced when RVP10 is in dual-receiver mode. The pulse is the sum of 2 transmit pulses at the primary and secondary intermediate frequencies. To make the simulation more realistic, the signal strengths are unequal - the primary pulse is 3 dB stronger than the secondary pulse.

The simulated burst responds to AFC like a real radar.

```
Frequency span of simulated burst: 27.00 MHz to 32.00 MHz
```

The question **`Frequency span of simulated burst`** sets the bandwidth of the simulated burst pulse.

6.2.3 **`Mt`** — Triggers and timing

To configure the general properties of the RVP10 trigger generator, type **`Mt`** with no additional arguments.

```
Number of transmit sequences : 4
```

RVP10 can currently support up to 16 definitions of a transmit sequence. However, IRIS Radar and the Zauto calibration utility currently support four. This practically limits the number of transmit sequences supported by Vaisala radar systems to four in the IRIS 10.1 release.

```
Current transmit sequence [0-3]: 0
```

Here you can select which transmit sequence is currently in use. Use this setting to define the pulse width used in transmission. Changing the **`Current transmit sequence`** here also changes the external transmitter pulse width.

External pretrigger

```
Use external pretrigger : YES
  PreTrigger active on rising edge: YES
```

When an external pretrigger is applied to the RVP10 **`TRIGIN`** input, either the rising or falling edge of that signal initiates operation.

This decision also affects which signal edge becomes the reference point for the pretrigger delay times given in the **`Mt<n>`** section.

Output triggers

```
Number of user-defined output triggers: 2
```

This question defines the number of user-defined output triggers. The number is limited to 12 (including polarization output controls).

Polarization output controls

```
Number of polarization output controls: 2
```

This question defines the number of polarization output controls.

Blankable triggers

```
Blank output triggers within AZ and EL sectors: YES
Sector#1  InUse:NO AZ:  0.0,0.0  EL:0.0,0.0 Ped:NO
Sector#2  InUse:NO AZ:  0.0,0.0  EL:0.0,0.0 Ped:NO
Sector#3  InUse:NO AZ:  0.0,0.0  EL:0.0,0.0 Ped:NO
Sector#4  InUse:NO AZ:  0.0,0.0  EL:0.0,0.0 Ped:NO
Sector#5  InUse:NO AZ:  0.0,0.0  EL:0.0,0.0 Ped:NO
Sector#6  InUse:NO AZ:  0.0,0.0  EL:0.0,0.0 Ped:NO
Sector#7  InUse:NO AZ:  0.0,0.0  EL:0.0,0.0 Ped:NO
Sector#8  InUse:NO AZ:  0.0,0.0  EL:0.0,0.0 Ped:NO
```

These settings reduce erroneous transmissions into physical obstructions.

RVP10 can inhibit the subset of blankable trigger lines when a noise measurement is taken. This is forced when trigger blanking (based on `TAG0`) is enabled, but it can also be selected in general through this question. Since noise triggers must be blanked when trigger blanking is enabled, this question only appears if trigger blanking is disabled.

These settings permit the state of the triggers during noise measurements to be consistent and known, regardless of whether the antenna is in a blanked sector.

```
Blank output triggers when changing pulsewidth : YES
    Times to wait (milliseconds) before/after PW change: 0 0
```

The question **`Blank output triggers when changing pulsewidth`** defines whether blanked noise triggers are used when changing pulsewidth.

```
Blank output triggers during noise measurement : YES
 Blank triggers : #1:Y #2:Y
```

The question **`Blank output triggers during noise measurement`** defines whether blanked noise triggers are used all the time.

Transmitter-related and receiver-related triggers

You can specify which triggers are transmitter-related and which are receiver-related.

Answer **YES** or **NO** for each trigger line, for the two polarization control lines, and for the timing of the phase control lines.

Answer **NO** for any trigger that is involved with the pre–fire timing of the transmitter.

If you enable the burst pulse tracker, you probably want to assign a **YES** to some of your triggers so that they remain fixed relative to the burst itself. See Burst pulse tracking (page 161).

The trigger outputs are defined by the following:

- Which RVP10 trigger outputs are timed relative to the transmitter pre–fire sequence. You can move these triggers left/right using the **L/R** keys in the **Pb** plot. These triggers are also skewed in response to Burst pulse tracking. See Burst pulse tracking (page 161).
- Which RVP10 trigger outputs are relative to the received target ranges. These triggers remain fixed relative to "receiver range zero", and are not affected by the **L/R** keys or by tracking.

For the trigger start times, move triggers that fire the transmitter, either directly or indirectly as a group when hunting for the burst pulse and moving it to the center of the FIR window. Fix triggers that function as range strobes relative to range zero, that is, the center of that window, and the center of the burst. This distinction is important when the transmitter's pre–fire delay drifts with time and temperature.

```
Rx–Fixed Triggers: #1 :N #2:N #3:N #4:N #5:N #6:N P0:N P1:N Z:N
```

```
2–way (Tx+ Rx ) total waveguide length: 0 meters
```

Use the parameter **2–way (Tx+ Rx) total waveguide length** to compensate for the offset in range that is due to the length of waveguide connecting the transmitter, antenna, and receiver. Specify the total 2–way length of the waveguides. The RVP10 range selection compensates for the additional waveguide length to within plus–or–minus half a bin, and works properly at all range resolutions.

```
Output triggers while idle: 0 meters
```

The RVP10 has option to create triggers only when actively acquiring data during a radar scan, then stop producing triggers between scans or elevation changes. However some transmitter types, such as magnetrons, prefer to be radiating continuously. This question sets if triggers are emitted during 'idle' periods between scans and elevation changes.

```
Default transmit sequence [0 – 3]: 0
```

Sets the transmit sequence that should be active on startup of the RVP10 server

```
Max duty cycle:  10.00%
```

This tells the RVP10 the maximum duty cycle of the transmitter. The RVP10 will not create any trigger sequence that exceeds this duty cycle given current pulse widths and PRFs requested. For each individual transmit sequence defined in the Mt<n> menu's, this value will also constrain the 'Maximum PRF' value that may be requested.

6.2.4 **Mt<n>** — Transmit sequence #n

To configure transmit sequences, type **Mt** with an additional argument for the transmit sequence number. For example, type **Mt1** to configure transmit sequence #1.

You can configure the following parameters for each transmit sequence:

```
Transmit Sequence #0 (ID: "mt_0")
-------------------------------------
Minimum range bin : 50.0 meters
Minimum PRF : 250.00 Hz
Maximum PRF : 2400.00 Hz
Default PRF: 300.0 Hz
Minimum PRT : 416.666667 usec
```

Triggers

Trigger parameters list the starting times (in microseconds relative to range zero), the lengths (in microseconds), and the active sense of each trigger generated by the internal trigger generator.

Set a length to 0 to inhibit the trigger on that line.

The **Start** time can include an additional term consisting of the formula: pulse period x a fractional multiplier between -1.0 ... +1.0. This allows you to produce trigger patterns that would not otherwise be possible, for example, a trigger that occurs half way between every pair of transmitted pulses, and remains correctly positioned regardless of changes in the PRF. If you do not want to use this term, enter this multiplier as **0**, so the term will be omitted from the printout.

In the following example, **Trigger #2** is a 10.0 µsec active–high pulse with a leading edge that occurs precisely halfway between the zero–range of every pair of pulses. Similarly, **Trigger #6** is a 2.0 µsec active–low pulse with a falling edge that is nominally 5.0 µsec prior to range zero, but that is advanced by 1.0 µsec for every millisecond of trigger period. All other triggers behave normally, and have fixed starting times that do not vary with the trigger period.

```
Triggers
--------

Trigger #0
----------
Start: -2.870000 usec
Length : 1.000000 usec
Pull up : YES

Trigger #1
----------
Start : -2.870000 usec
Length : 0.125000 usec
Pull up : YES
```

Note the following about these variable start times:

- The **PRT** multipliers can only be used with the RVP10 internal trigger generator. The **PRT**–relative start times are completely disabled when an external trigger source is chosen from the **Mt** menu.
- When **PRT**–relative triggers are plotted by the **Pb** command, the active portion of the trigger is drawn cross–hatched at a location computed according to the current PRF. The cross–hatching serves as a reminder that the location of that trigger may vary from its presently plotted position.
- The **PRT** multiplier for a given pulse is applied to the interval of time between that pulse and the next one.
  This distinction is important when RVP10 generates multiple–PRT triggers, for example, during DPRT mode or during dual–PRF processing.
  Multipliers from **0.0 ... +1.0** are generally safe to use because they shift the trigger into the same pulse period that originally defined it. For example, a start time of (**0.0 µsec + (0.98 * PRT)**) positions a trigger 98% of the way up to the next range zero. But, if **–0.98** were used, and if the period of the previous pulse was shorter than the current one, then that shorter period would become incorrect (longer) as a result of having to fit in the very early trigger.

In some situations, waveforms that do not fit are zeroed (not output) to preserve the desired period. This means that you can define triggers with large positive start times, and they come into existence only when the PRF is low enough to accommodate them. It applies when:

- The trigger period is internally determined, that is, the external pretrigger input is not being used.
- The overall span of the six trigger definitions combined does not fit into that period.

For example, if **Trigger #2** is defined as a 200.0 µsec pulse starting at +400.0 µsec:

- The trigger is suppressed if the PRF is 2000 Hz.
- The trigger is present at a PRF of 1000 Hz.
- If a trigger does not completely fit within the overall period it is suppressed.

This means that although the +400.0 µsec start time is still valid at 2000 Hz, the entire 200.0 µsec pulse would not fit, and the pulse is eliminated.

Start limits: -5000 ... 5000 µsec

Width limits: 0 ... 5000 µsec

Pulses

```
Pulses
------
Number of Pulses : 2
  Blending area as % of short and long pulse overlap (0:None) 50 %
```

In a magnetron system, the pulse definition is external to the RVP10. Thus, this question tells RVP10 which digital filter design should be used for the particular pulse selected in this transmit sequence.

```
Pulse #0(ID: "mw_0")
--------------------
  Pulse ID: "mw_X", X = 0
 Burst Freq Estimator (fraction of FIR filter) - Start: 0.10 , Length: 0.35
```

This section defines the pulse definition from **Mw<n>** of the first pulse transmitted in the transmit sequence. In the current IRIS RDA version, the **Mw<n>** index must be the same value as the **Mt<x>** index. For example, if this is configuration of **Mt<2>** the first pulse must be defined in **Mw<2>**.

The **Burst Freq Estimator** field shows the location of the aperture window which is the narrowed set of samples used for AFC frequency estimation. This estimator is mostly used with the Pb (plotting commands). See **Pb**subcommands (page 116)

```
Pulse #1(ID: "mw_1")
--------------------
  Pulse ID: "mw_X", X = 1
  Second pulse start delay from end of first pulse : 10.0 nsec
```

When using two pulses in the sequence, the **Pulse#1 ID** defines which pulse defined in the **mw<n>** menus will be the second to be transmitted. It is possible to select any **Mw<n>** index to use for this pulse. It is possible to configure a delay time between the first pulse and second pulse.

```
Hybrid Pulse Info
-----------------------
 First pulse: mw_2
   Start:    -0.21 usec
   Length:   90.00 usec
   End:      89.79 usec
 Second pulse: mw_1
   Delay:    10.00 nsec
   Start:    89.80 usec
   Length:    4.00 usec
```

Hybrid Pulse Info is information only which summarizes the pulses configured in the selected transmit sequence.

Blending

RVP10 enables blending of long and short pulses. When blending is used, the transmitter alternates between short and long pulses, and the data streams are processed simultaneously and then combined. The result is improved data quality at the short range, and greater sensitivity at the long range.

RVP10 collects IQ data from the short pulse up to the range that is equivalent to twice the pulse length of the long pulse. For example, if the long pulse is 80 µs in length (13 km long), the short pulse IQ data is collected up to 26 km in range. This provides an area of overlapping data that is equivalent to the common length of the long pulse, 13 km in this example. The blending methodology is to average IQ together from the short and long pulse where overlapping data exists. As the long pulse is generally used at further ranges, a weighting function is applied, so that the short pulse gets the full weighting near the radar and zero weighting at further range, and vice versa for the long pulse IQ.

The user can adjust the strength of the blending by reducing the overlap area in range. In this example, a 100% value would perform the weighted averaging of IQ data over the full 13 km distance. A 50% value would produce a weighted averaged IQ over a 6.5 km range, starting at the point where IQ data from the long pulse is available. A 0% overlap area value would tell the signal processor to do no averaging of IQ data. In this case, the short pulse data would be shown from 0 to 13 km, and then long pulse data would be shown from 13.1 km to the end range.

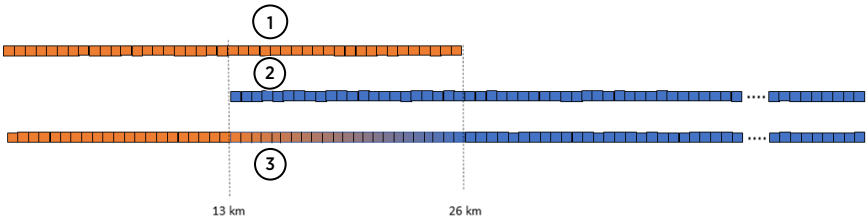


Figure 13 Blending

1 IQ data - short pulse
2 IQ data - long pulse
3 Merged IQ data (distance-weighted averaging withing the overlap region)

To use blending, set the following parameters:

Hybrid pulse chained PW index

- If the value is other than **−1**, the blending is used.
- If the value is **−1**, the second, short pulse, is not transmitted, and there will be no data in the near range.
- When the second pulse index is selected in that setting, one can adjust the amount of overlap.
- If the value is **0**, there is no overlap, and there is a sharp transition between the short pulse and the long pulse. This setting is useful for verifying the match in calibration of two pulses, and for adjusting PhiDP offsets.

Blending area as % of overlap with the chained pulse
This parameter controls the level of the data overlap between two pulses. The recommended value is **50%**, which is also the default value. **50%** means that the overlap is the half length of the long pulse.

Example:

```
Number of Pulses:  2
      Blending area as % of short and long pulse overlap (0:None) 50%
```

6.2.5 **Mw<n>** - Transmit waveforms

Mw<n> menu is used for defining the waveform to be used within a transmit sequence. The FIR downconversion filter and any windowing functions are also defined here. Applying windowing functions to the ADC samples when deriving IQ is a new functionality available in the RVP10.

Entering **Mw** without any index:

```
Number of pulse/filter definitions: 4
```

The maximum number of different transmit waveforms is 16.

Configure settings for each transmit waveform by typing **Mw<n>**, where **n** is the number of the pulse.

Intermediate frequency

```
Pulse/Filter #2 (ID: "mw_2")
----------------------------
Tx Intermediate Frequency  : 62.0000 MHz
Rx Intermediate Frequency : 62.0000 MHz
```

Define the center of the intermediate frequency of the transmit pulse to be produced, if applicable, and the center of the digital receiver. The limits are 1 – 12 MHz.

The radar intermediate frequency can be selected separately for each pulse width, so that different width pulses can occupy different frequency bands if desired. The Tx and Rx intermediate frequencies can also be different from each other, so that the RF up-conversion chain for transmission can be different from the down-conversion chain for reception.

```
Current IFD noise levels – PriRx: -84.60 dBm,  SecRx: -84.80 dBm
Powerup IFD noise levels – PriRx: -84.60 dBm,  SecRx: -84.80 dBm
```

These questions allow you to set the current value and the power-up value of the receiver noise level for either a single or dual receiver system. These questions are intended for applications in which RVP10 must operate with a reasonable default value, up until the time that an **SNOISE** command is received. They may also be used to compare the receiver noise levels during normal operation, which serves as a check that each FIR filter is behaving as expected when presented with thermal noise.

- The noise level(s) are shown in dBm, and you may alter either one from the TTY.
- The power-up levels are assigned by default when the RVP10 first starts, and when the **RESET** opcode is issued with Bit #8 set.
- The current noise level is revised when the **SNOISE** opcode is issued.

Transmit pulse

```
Pulse
-----
Pulselength of transmit pulse: 1.0000 usec
Pulse index (aka 4-bit code): 0x2
```

The value of the question **Pulselength of transmit pulse** represents the entire time duration of the waveform, including whatever amplitude modulations may be included at the tails. This may be derived from the width of the pulse as observed in the **Ps** plot, or using an external oscilloscope. For more information, see Burst pulse timing plot (Pb) (page 113).

Waveforms are synthesized by the RVP using a 16-bit TxDAC followed by an analog bandpass filter centered at the midpoint of the IF interval. The analog filter is necessary for removing out-of-band components from the TxDAC, but has a side effect of introducing a bandwidth limitation within the IF passband. The result is that the shape of very short pulses (on the order of 100 nanoseconds) is dominated by the impulse response of the analog filter rather than by the exact digital synthesis.
In practice, the Tx pulse length should be longer than 300 nanoseconds for the final analog signal to be a faithful reproduction of the intended waveform.

```
Pulse index
```

The question **pulse index** defines the 4-bit code a controlling host may provide to the RVP10, requesting the RVP10 to switch to this Waveform/Filter definition.

If the RVP10 is configured to create transmit waveforms for coherent transmitters, the following questions will appear:

```
Waveform Design Type:
 1: CW
 2: LFM
 3: NLFM
Waveform <NLFM> : 3
```

The RVP10 supports three standard Tx waveforms:

- Conventional fixed– frequency CW pulse
  The CW Pulse can be used as a pulsed Doppler waveform in all the same way that a Klystron or Magnetron system with a traditional pulse forming network would be used.
- Linear and non–linear FM chirp
  The linear and non–linear FM waveforms, however, are compressed pulses that are intended to be transmitted by a wide–bandwidth Klystron/TWT/solid-state amplifier.

```
Bandwidth of transmit pulse: 4.00 MHz
TxWave tuning parameter: 0.0990
```

The next questions select the bandwidth of the Tx waveform. The bandwidth value represents the true spectrum width of the complete waveform, that is, including all the effects of the frequency modulation and amplitude modulation used by the waveform. A spectrum analyzer (or the RVP10 `Ps` plot) shows an overall spectrum width equal to this desired value.

```
TxWave tuning parameter: 0.0990
```

If Non-Linear Frequency Modulation (NLFM) is selected as the waveform type, a tangent function frequency modulation is applied to the pulse. The `TxWave` tuning parameter allows the user to modify the properties of the tangent function, sharpening or shallowing the rate of frequency modulation. Generally, a value of 0.05 – 0.10 works well with Vaisala SSPA weather radars.

```
Waveform Window Type:
 0: Blackman
 1: Blackman Exact
 2: Flat Top
 3: Hamming
 4: Hann
 5: Kaiser
 6: Rectangular
 7: Triangular
 8: Tukey
Window type <Tukey> : 8
Window edge (tapering, 0 to 0.5) : 0.08
```

To apply amplitude tapering to the transmit waveform the RVP10 allows a selection of multiple windowing functions. Generally, we recommend a Tukey window function with 0.05 to 0.1 edge points.

Filter

This section defines the type of digital filter to be used in this pulse definition. If RVP10 is configured for magnetron system the digital filters supported are the legacy bandpass FIR and an idealized matched filter. The legacy bandpass filter is same methodology used in RVP900 and earlier products where user directly sets the bandwidth and time window using the plotting tools within **dpsx**. The new idealized matched filter is computed given the pulse length of transmission and Rx intermediate frequency values. If configured for coherent transmitters, the legacy bandpass filter is not available and the question about the filter type is hidden.

```
Filter
------
Type <Matching> (0 - Bandpass; 1 - Matching) : 1
 Length of matched filter: 1.00 usec

Filter Window Type:
 0: Blackman
 1: Blackman Exact
 2: Flat top
 3: Hamming
 4: Hann
 5: Kaiser
 6: Rectangular
 7: Triangular
 8: Tukey
Window type <Hamming>: 3
Window edge (tapering, 0 to 0.5): 0.50
```

The menu above shows the options for the idealized matched filter. The first question defines the length of the matched filter to be used in microseconds, and the question **`Filter Window Type`** defines the type of windowing applied over the filter length to the raw ADC samples when computing IQ. The windowing functions here are as defined within MATLAB.

The question **`Window edge (tapering, 0 to 0.5)`** defines how much window tapering is applied to the raw ADC samples when computing IQ. 0 means no tapering is present, giving the same output as the 'rectangular' window. A value of 0.1 implies that 10% of the ADC samples at the start and end of the burst pulse time domain are tapered. A value of 0.5 implies that 50 % on each side of the time domain are tapered, or in other words the windowing function would be applied to all ADC samples.

```
Filter
------
Type <Bandpass> (0 - Bandpass; 1 - Matching) : 0
  Bandwidth of bandpass filter: 1.78 MHz
  Length of bandpass filter : 1.25 usec
```

The menu above shows the options for the legacy band-pass filter. The first question defines the bandwidth of the band-pass filter in MHz and the second question defines the length of the band-pass filter in microseconds.

6.2.6 **Mp** — processing options

In the **Mp** menu, you can configure the processing options.

```
Allow continuous sizes for power spectra: YES
```

The power spectra computed within RVP10 are normally not constrained to be powers of two in length. Answer **NO** to the question **`Allow continuous sizes for power spectra`** to reapply the constraint to mimic the behavior of older processors.

Spectrum width

```
R2 Processing Algorithms  – 0:Never , 1:User, 2:Always : 1
```

Th question **R2 Processing Algorithms** controls whether **R0/R1/R2** or **R0/R1** estimates are used to compute the spectrum width (see Spectrum width algorithms (page 190) and Setup operating parameters (**SOPRM**) (page 368)):

- Select **0** to unconditionally disable the **R2** algorithms, regardless of what the host computer requests in the **SOPRM** command.
- Select **2** to unconditionally enable **R2** processing. Bit-7 of **SOPRM** word #2 is the host computer's interface to this function when the **1:User** case is selected.
- IRIS radar application uploads **R0/R1/R2**.

Ascope uploads the **R2 algorithms** button setting in the block *Gen Setup*. When IRIS Radar is the controlling host, the **User** setting is defined in the **IRIS Ingest Setup** menu. For more information, see *IRIS and RDA Utilities Guide (M212925EN)*.

Clutter microsuppression

```
Clutter MicroSuppression – 0:Never , 1:User, 2:Always : 1
```

This question controls whether cluttered bins are rejected before being averaged in range. Bit-8 of **SOPRM** word #2 is the host computer's interface to this function when the **1:User** case is selected (see Setup operating parameters (**SOPRM**) (page 368)). The functionality is dependent on the application settings for CCOR thresholds, in addition (see Range averaging and clutter microsuppression (page 36)).

IRIS requests clutter microsuppression. **Ascope** uploads the **Clutter Microsuppression** button setting in the block *Gen Setup*.

Autocorrelation

```
PPP  autocorels from FFTs – 0:Never , 1:User, 2:Always : 1
```

When autocorrelation terms are computed from power spectra, the results differ from a strict time-domain calculation in that an "end-around" term is introduced as a result of circular rather than linear convolution.

Requesting PPP-style autocorrelations correct this spurious term when the computations are done through FFTs. Bit-11 of **SOPRM** word #10 is the host computer's interface to this function when the **1:User** case is selected (see Setup operating parameters (**SOPRM**) (page 368)). IRIS does not request PPP-style autocorrelations, but **Ascope** requests it.

Velocity and processing algorithms

```
Unfold Velocity ( Vh – Vl ) – 0:Never , 1:User, 2:Always : 0
```

The question **`Unfold Velocity`** allows you to choose whether RVP10 unfolds velocities using a (V_{high}– V_{low}) algorithm, instead of the improved algorithm described in Dual PRF velocity unfolding (page 194). Bit– 11 of **SOPRM** word **`#10`** is the host computer's interface to this function when the "**`1:User`**" case is selected (see Setup operating parameters (**SOPRM**) (page 368)).

```
Process w/ custom trigs – 0:Never , 1:User, 2:Always : 0
```

The **`Process w/ custom trigs`** parameter allows you to choose whether RVP10 attempts to run its standard processing algorithms even when a custom trigger pattern has been selected through the **SETPWF** command. Usually, this is not recommended, so the default setting is **`0:Never`**. Bit–12 of **SOPRM** word #10 is the host computer's interface to this function when the **`0:Never`** option is selected (see Setup operating parameters (**SOPRM**) (page 368)).

Dual-polarization attenuation correction

```
DualPol Atten Correction –  0:Never, 1:User, 2:Always : 1
```

The question **`DualPol Atten Correction`** defines when RVP10 applies attenuation correction based on the integration of differential phase. Answer **`0:Never`** to never apply the correction, **`2:Always`** to always apply the correction, or **`1:User`** for this to be defined by the controlling host. For more information, see Dual-polarimetric attenuation correction (page 231).

Additional SNR

```
Use High-SNR 16-bit packed timeseries format: Yes
```

This question provides an additional 6 dB of SNR. The parameter can be disabled to provide compatibility with legacy systems.

Free-running rays

```
Minimum freerunning ray holdoff : 100% of dwell
```

Limits: 0 ... 100%

This question controls the rate at which the RVP10 processes free-running rays. This prevents rays from being produced at the full CPU limit or I/O limit of the processor (whichever was slower), which could result in highly overlapping data being output at an unusably fast rate. This behavior only occurs when running without angle syncing, such as during IRIS manual and RHI scans.

To make these free-running modes more useful, you can establish a minimum hold-off between successive rays, expressed as a percentage of the number of pulses contributing to each ray:

- **`100%`** (default) produces rays whose input data do not overlap at all, that is, whose rate is exactly the PRF divided by the sample size.

- **0%** gives the unregulated behavior in which no minimum overlap is enforced and rays can be quickly produced.

Saturation algorithm

```
Linearized saturation headroom: 4.0 dB
```

Limits: 0 dB ... 5 dB

RVP10 uses a statistical saturation algorithm that estimates the real signal power correctly even if the IF receiver is over-driven (that is, for input power levels above +4 dBm). The algorithm extends the headroom above the top end of the A/D converter, although the accuracy decreases as the overdrive becomes more severe. This parameter allows you to place an upper bound on the maximum extrapolation that is applied. Choose **0 dB** to disable the algorithm.

This linearized saturation headroom function is not available when using compressed pulses, even if selected.

Amplitude correction

```
Apply amplitude correction based on Burst/COHO: YES
 Time constant of mean amplitude estimator: 70 pulses
```

Limits: 10 pulses ... 500 pulses

RVP10 can perform pulse-to-pulse amplitude correction of the digital (**I**,**Q**) data stream based on the amplitude of the Burst/STALO input. See Amplitude correction for Tx power fluctuations (page 166).

Interface filter and unfolding

RVP10 can apply an interference filter to remove impulsive-type noise from the demodulated (**I**,**Q**) data stream. See Interference filter (page 161).

```
Interference Filter 0:None , Alg.1, Alg.2, Alg.3:
Provide WSR88D legacy BATCH major mode: YES
  Maximum range to unfold: 600.0 km
  Low-PRF bins range averaged on each side: 2
  Low-PRF increased noise threshold margin: 0.00 dB
  Overlay power - Refl :5.0 dB Vel:8.0 dB Width:12.0 db
  LowSamps  = ( 0.00000 x  HiSamps ) + 6.00 :
    LowPRF = ( 0.00000 x HiPRF ) + 250.00 :
```

This is a general implementation of a Lo/Hi Surveillance/Doppler PRF unfolding scheme that provides the legacy features as special cases. The parameters are defined as follows:

- The parameter **maximum range to unfold** (in km) allows you to set an upper bound on how many Doppler trips unfold according to the echoes seen in the surveillance data.

- The surveillance data set uses very few pulses and is somewhat noisy. You can choose the number of bins that are range averaged from both sides of these bins to provide a lower variance power estimate. A value of **0** means **No averaging**, a value of **1** averages 3 points, and so on.
- The unfolding algorithm flags obscured range bins according to three different power thresholds for reflectivity, velocity, and width, and outputs these bits in the **DB_FLAGS** data parameter. Each threshold is specified in decibels.
- The fundamental RVP10 operating parameters (such as PRF and Sample Size) apply to the high PRF portion of the BATCH trigger waveform. The low PRF rate and sample size are derived from these high values using a slope and offset. In the example, the slopes are both 0, so that the surveillance data is fixed at 6-pulses and 250 Hz. Making the slopes non-zero would cause the low-PRF parameters to vary automatically if desired.

These setup parameters are accessible through the DSP driver using the entry points **dspw_batchSetup()** and **dspw_batchSetup()**. These use the custom opcode that is defined separately by each major mode. The **customUserOpcode_batch()** is a useful model building these entry points.

Standard parameters in multi-polarization systems

```
T/Z/V/W computed from: H- Xmt :YES  V- Xmt:NO
```

The question **T/Z/V/W computed from** controls how the standard parameters (Total Reflectivity **T**, Corrected Reflectivity **Z**, Velocity **V**, and Width **W**) are computed in a multiple-polarization system. Applying **YES** with **H-Xmt** or **V-Xmt** means that data from those transmit polarizations is used when there is more than one choice available. These selections only apply to the Alternating and Simultaneous transmit modes.

```
Polarimetric Power Params -  NoiseCorrected:YES
 Polarimetric Correlations -  NoiseCorrected:YES
```

These questions specify whether noise correction is applied to dual-polarization measurements.

You can configure the sign and offset corrections to correct for intervening weather attenuation. It uses the change of angle in DP. The tuning numbers are taken from the *dualpol.conf* file.

```
PhiDP - Negate: NO ,  Offset:90.0 deg
```

The 2-bit PhiDP datatype has a range of 360 degrees. The question **PhiDP - Negate: NO , Offset:90.0 deg** allows setting an offset so that the initial PhiDP values being measured by the radar system starts in the lower end of the data range format. Vaisala recommends setting the offset so that the initial PhiDP values are near 10 degrees of value. The negate option may be flipped if PhiDP values are decreasing in range instead of increasing.

```
KDP computation - 0 :LSQ, 1:Weighted LSQ, 2:Cubic Splines
```

Select the K_{dp} computation:

LSQ
: Least square fit to a linear function.

Weighted LSQ
: Weighted least square fit to a linear function. FIR filter coefficients are used as weights.

For **LSQ** K_{dp} computation, select the length:

```
KDP – LSQ  Weights(FIR) Width: 4.00 km
```

A typical installation uses **H–Xmt:YES, V–Xmt:No**. Including both transmissions decreases the variance of (**T/Z/V/W**), but most researchers prefer excluding **V–Xmt** because that is more standard in the literature.

Melting height

```
Melting height: 3000 meters
```

The melting height is used in **HydroClass** calculations and is recorded with the data.

This is the height above the radar, so must include the altitude of the radar when computing. Normally this is set automatically by the controlling application.

Noise correction

```
Enable noise power based correction of Z0: No
```

If the noise correction is set to **Yes**, RVP10 adjusts the calibration reflectivity value **Z0** when the current noise level changes from the level measured when the calibration was done.

See Noise correction to reflectivity calibration (page 188).

Wide dynamic range

```
Wide dynamic range:
   Maximum allowable amplitude error in range: 0.5 dB
   Maximum allowable phase error in range: 5.0 deg
   Overlap/interpolate interval in range: 40.0 dB
```

RVP10 continually measures and updates the complex channel separation during normal operation. Ratios of echoes that fall within the overlap/interpolate interval are averaged over several minutes, tracking gain and phase variations that occur with temperature changes and component aging. If the channel separation exceeds the specified maximum deviation, the **GI4S_IFDCHANERR** bit (11) is set in **GPARM Immediate Status Word #4**.

6.2.7 `Mf` — Clutter filters

When clutter correction is applied to the reflectivity data, you must increase the **LOG** noise threshold slightly to continue to provide reliable qualification of the corrected values. The reason for this is that the uncertainty in the corrected reflectivity becomes greater after the clutter is subtracted. For example, if we observe 20 dB of total power above receiver noise, and then apply a clutter correction of 19 dB, we are left with an apparent weather signal power of +1 dB above noise. However, the uncertainty of this +1 dB residual signal is much greater than that of a pure weather target at the same +1 dB signal level.

```
Default residual clutter LOG noise margins:
 Baseline :  0.15 dB/dB for Clutter/Noise above 10dB
 HiSignal  : 1.00 dB/dB for Clutter/Noise above 50dB
```

The `Residual Clutter LOG Noise Margin` allows you to increase the **LOG** noise threshold in response to increasing clutter power. In the previous example, and with the default setting of 0.15 dB/dB, the **LOG** threshold increases by `19 x 0.15 = 2.85 dB`. This helps eliminate noisy speckles from the corrected reflectivity data.

When RVP10 computes power spectra, the time series data are multiplied by a (real) window before computing the Fourier Transform (DFT).

```
Default Spectral Window
 0:User , 1:Rect, 2:Hamming
 3:Black, 4:ExBlack, 5:VonHann, 6:Adaptive:0
```

You can select the window through **SOPRM** word #10 `0:User`, or force a particular window:

- In the IRIS application, the setting `0:User` refers to the window defined in the **Setup** utility block RVP signal processing options.
- In the **Ascope** application, the setting `0:User` enables the **Spect Win** button.

At RVP10 startup, `0:User` refers to the settings saved in *`rvp10.conf`* block `opprm.filter` bits 9, 10, 11.

```
Spectral Clutter Filters
------------------------
  Window -1:Default   0:Rectangular  1:Hamming
  Code    2:Blackman  3:ExBlackman   4:VonHann  5:Adaptive
Filter #1  Type:0(Fixed)              Win:1   WidthPts:1 EdgePts:2
Filter #2  Type:0(Fixed)              Win:2   WidthPts:2 EdgePts:2
Filter #3  Type:0(Fixed)              Win:2   WidthPts:3 EdgePts:3
Filter #4  Type:1(Variable)           Win:2   WidthPts:3 EdgePts:2 HuntPts:3
Filter #5  Type:3(Gaussian Adaptive)  Win:-1  Spectrum width: 0.200 m/sec
Filter #6  Type:3(Gaussian Adaptive)  Win:-1  Spectrum width: 0.300 m/sec
Filter #7  Type:3(Gaussian Adaptive)  Win:-1  Spectrum width: 0.500 m/sec
```

The question `Spectral Clutter Filters` defines the clutter filters that operate on power spectra during the DFT-type major modes (**PPP**, **FFT**, and **RPH**).

`Filter #0` is reserved as `all pass`, and cannot be redefined here. For filters `#1` to `#7`, enter a digit to choose the filter type, followed by the parameters that the type requires. See Clutter filtering approaches (page 175).

Fixed width filters (Type 0)

Fixed width filters are defined by the following additional parameters:

Width
Sets the number of spectral points that are removed around the zero velocity term. A **Width** of 1 removes the DC term. A **Width** of 2 removes the DC term plus 1 point on either side. A **Width** of 3 removes DC plus 2 points on either side, and so on. Spectral points are removed by replacing them with a linear interpolating line.

EdgeMinPts
The endpoints of the line are determined by taking the minimum of **EdgeMinPts** past the removed interval on each side.

Variable width, single slope (Type 1)

RVP10 supports variable-width, frequency-domain clutter filters. These filters perform the same spectral interpolation as the fixed–width filters, except that their notch width automatically adapts to the clutter.

The filters are characterized by the **Width** and **EdgeMinPts** parameters in the **Mf** menu, except that the **Width** is now interpreted as a minimum width.

The **Hunt** parameter allows you to choose how far to extend the notch beyond **Width** to capture the clutter power. Setting **Hunt=0** converts a variable–width filter to a fixed–width filter.

The algorithm for extending the notch width is based on the slope of adjacent spectral points.

- The beginning (**Width–1**) points away from 0, the filter is extended in each direction as long as the power continues to decrease in that direction, up to adding a maximum of **Hunt** additional points.
- If you run with a fixed **Width=3** filter, try experimenting with a variable **Width=2** and **Hunt=1** filter.
- If the original fixed width occasionally fails, but you are reluctant to increase it to cover those rare cases, try selecting a variable **Width=2** and **Hunt=2** filter.

In general, make your variable filters "wider" by increasing **Hunt** rather than by increasing **Width**. This preserves more flexibility in how they can adapt to whatever clutter is present.

Gaussian model adaptive processing (GMAP) (Type 3)

This is the most advanced form of clutter filtering and moment estimation. See Clutter filtering approaches (page 175).

For GMAP processing, you must specify the spectrum width of clutter. The algorithm is not very sensitive to the exact value. Configure several widths to cover the antenna rotation rates that are commonly used. It is useful to turn off clutter filtering (select the all pass filter **#0**) and then look at measurements of the clutter width while the antenna is rotating using, for example, the **Ascope** utility or application software such as IRIS.

```
Whitening Parameters for  Tx :Random
```

```
Secondary SQI Threshold Slope:0.50 Offset:-0.05
```

The question **Secondary SQI Threshold Slope** defines the secondary **SQI** threshold that is traditionally used to qualify **LOG** data in the random phase processing mode. The secondary **SQI** threshold is applied uniformly in all processing modes when reflectivity data are specified as being thresholded by **SQI**.

The secondary **SQI** level is computed by multiplying the primary user-supplied **SQI** threshold by the **SLOPE**, and adding the **OFFSET**. See Tuning for optimal performance (page 201).

Limits: **SLOPE**: 0.0 ... 2.0, **OFFSET** -2.0 ... 1.0

```
Whitening Parameters for Tx :SZ (8/64)
-------------------------------------
Max power mismatch across octants: 4.0 dB
 High power rejection threshold: 8.0 dB
Maximum KEY phase error: 12.0 deg
```

The parameters in the question **Whitening Parameters for Tx** tune the phase whitening process for SZ(8/64) transmissions.

6.2.8 **Mz** — Transmissions and modulations

Use these questions to tell RVP10 what type of transmitter is in use and to configure the phase modulation codes that may be used to control the phase of a coherent transmitter.

Transmit modes

```
Transmit Modes:
0: External Transmitter
1: Single Channel Transmitter/Amplifier
2: Dual Channel Transmitter/Amplifier
```

Use the **Transmit Mode** question to select what kind of transmitter is used for generating the transmit pulses. External transmitter (**0**) can be, for example, a Magnetron or Klystron transmitter. Options **1** and **2** mean that transmit pulse is being generated within IFDR10 with either a single transmitter or a system having two transmitters.

```
Phase angle to apply when idle: 0.00 deg (0x0)
Modulation - 0:None, 1:Random, 2:Custom, 3:Sz(8/64)
```

6.2.9 **M+** Debug options

Noise level for simulated data

RVP10 debugging options help you develop and debug application code. You can also use these questions to put RVP10 in unusual operating states. These options are usually disabled during normal radar operation.

```
Noise level for simulated data: -50.0 dB
```

Limits: –100dB ... 0dB

The question **`Noise level for simulated data`** defines the noise level that is assumed when simulated **`I`** and **`Q`** data is injected into RVP10 with the **`LSIMUL`** command. The noise level is measured relative to the power of a full–scale complex (**`I`**,**`Q`**) sinusoid, and matches the levels shown on the slide pots of the **Ascope** digital signal simulator.

Nyquist sign flip of plotted IF samples

```
Nyquist sign flip of plotted IF samples: NO
```

This question defines whether IF samples should be plotted with Nyquist flipping (multiplication by +1,–1,+1,–1...).

If **`YES`**, the **`Pr`** and **`Pb`** plots modify their rendering of IF samples. In those menus, the text **`(NyFlip)`** appears in the plot text heading when sample flipping is being applied. Other plots in these menus, such as **spectra** and **LOG magnitude**, are unaffected by the Nyquist flipping, nor are any live parameters derived from the IF samples. Only the visual appearance of the IF samples is affected.

7. Plot-assisted setups

7.1 Plot-assisted setup overview

RVP10 provides an interactive graphical alignment tool for burst pulse detection, trigger timing,transmit waveform and digital receive filter design in time and frequency domain, Tx/Rx phase locking, and calibration of the AFC feedback loop. The tool includes:

- Viewing samples of the burst pulse and receiver waveform to examine their frequency content, design an appropriate passband FIR filter or have the RVP10 calculate an ideal matched FIR filter, and observe live operation of the AFC.
- Checking the spectral purity of the transmitter on a regular basis to check for unwanted noise or harmonics.
- Design of continuous wave (CW) and linear and non-linear pulse compression (LFM, NLFM) pulses to be used within a transmit sequence.

RVP10 can track and modify the initial settings so that proper operation is maintained even with changes in temperature and aging of the microwave components.

You can use the **dspx** utility to display the plots, and you can perform graphical checkup and alignment procedures remotely through a network. To access the plot-assisted setups, type the **P** commands in the TTY setup interface.

RVP10 supports opcodes that allow the host computer to monitor the data being plotted.

7.2 Plot command conventions

The **Pa**, **Pb**, **Ps**, and **Pr** commands have a similar structure in their TTY user interface.

Each command first prints a list of subcommands that are valid in that context.

The working and measured parameters for each plot command are printed on the TTY as two lines of information following the subcommand list:

- The first line contains settings that change when a subcommand is issued.
  The first line is printed once.
- The second line is live and reflects the current status of the burst input, the IF input, or the AFC output.
  The second line is continually overprinted. This makes it appear as a live status line with up-to-date values.

The content on these lines may at times exceed the shell window width. When using these plots, it is recommended to increase the windows width so that all characters fit onto the two lines described.

The **Pb**, **Ps**, and **Pr** commands report `No Trigger` on the TTY status line when the external trigger is expected but missing.

Table 29 Plot command conventions

Convention	Description
Display commands	Most plot commands support subcommands that alter the appearance of the display, for example, zoom and stretch. These subcommands make no changes to the working RVP10 calibrations. Display settings are stored in nonvolatile RAM, like the other setup parameters. Previous display settings are restored when you restart each plot command. If the initial list of subcommands disappears off the top, type **?** to force a reprint.
Exit	To exit the plot command and return to the TTY main menu, type **Q** or **Esc**. All other keys return the plot command to its normal live updating, but the key is otherwise discarded (for example, subcommand keys are not executed while exiting from single step mode).
Lower case commands Upper case commands	Most commands have a lowercase and an uppercase version. If a lowercase command performs an action, then its uppercase version performs either the same thing on a larger scale, OR the reverse action. For example, if the **w** subcommand widens a view a little bit, the **W** widens it a lot.
Single Step mode	By default, the graphical display and TTY status lines are updated with fresh data several times per second. If you wish to freeze a plot for further analysis or comparison enter **Single Step** mode by typing: "**.**" (period). This causes the display and TTY status lines to freeze in their present state, and prints the message **"Paused..."** To move to the next data update, retype "**.**" (period). The plot and printout remain frozen at the next data update.
Scrolling	The TTY screen scrolls upward each time a new subcommand is executed to show a history of information lines and command activity. Press **Enter** to scroll the display up at any time.
%	Press **%** to toggle between the IF input SMA connectors on the IFDR.

7.3 Burst pulse timing plot (Pb)

For magnetron radars, RVP10 relies on samples of the transmit pulse to lock the phase of its synthesized **I** and **Q** data and to run the AFC feedback loop.

Use the **Burst pulse timing (Pb)** plot to view and adjust the trigger timing and sampling window so that the burst pulse is correctly measured.

This is used, for example, after radar installation, or when troubleshooting the radar system. Configuring the Burst pulse timing needs to be done for each pulse width. Vaisala weather radars have 4 pre-defined pulse widths, and this plot is used for fine-tuning the transmitter.

You can modify the pulse and the triggers as follows with the **Pb** plot:

- Impulse response length: select the pulse, and modify the width of the window
- Aperture and start of the AFC window: select a portion of the Tx signal that will be used for sampling the frequency for AFC (only relevant for magnetron transmitters)
- Trigger timing: shift triggers left or right. Each trigger has its own length and start time with respect to each other (defined in the **Mt** menu). In the Burst pulse timing plot, all the triggers are moved simultaneously as a group. The number of triggers for the pulses is defined in the Mt menu.
- You can select the IF channel. The definition of the channels depends on the systems.

7.3.1 Burst timing plot display

The following figure shows an example of a burst timing plot display.

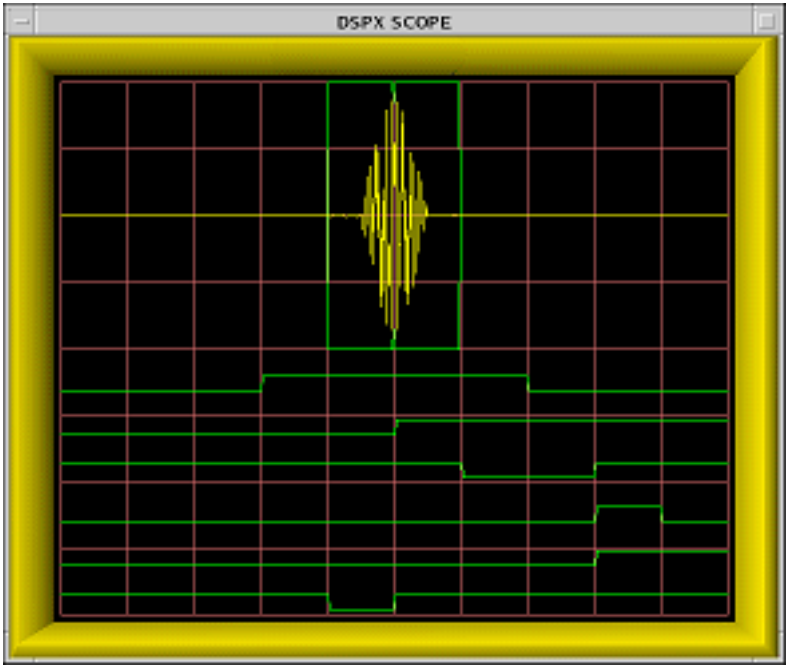


Figure 14 Burst timing plot

X and Y axis: time and amplitude

The vertical axis indicates the frequency amplitude.

The horizontal axis indicates time. The duration can be adjusted using the **T/t** keys to increase or decrease the time span of the plot.

Pulse signal

In the graphical interface, the burst pulse signal is shown as a yellow line. The signal is shown in (nearly) real-time, constantly changing according to the transmitted pulses. If no pulse is shown (just a flat line), it means that either the transmitter is off or the trigger has not been properly configured.

The same FIR coefficients that compute **I** and **Q** are used to compute the reference phase vectors for the burst pulses.

Sampling window

The upper portion of the display shows the sampling window where the burst pulse is measured.

The outer green window in the middle of the screen is the **Impulse response length window (sampling window)**. The pulse should occur at the center of this window. If not, you need to adjust the triggers until the pulse is at the center of the window. The sampling window is always in the middle of the display.

RVP10 computes the power-weighted center-of-mass (COM) of the burst pulse envelope. This allows the processor to determine the location of the "middle" of the transmitted pulse within the burst analysis window.

The display has small tick marks on the top and bottom of the burst sample window to indicate the location of the COM. These markers are only displayed when valid burst power is detected. A square surrounding the tick mark indicates the level of uncertainty of the mark. This error interval is used by the burst pulse tracking algorithm to decide when a timing change can be made with confidence.

AFC window

The inner green window is the **AFC window**. It should be at the middle of the pulse, so that the rising and falling edges of the pulse are excluded. The AFC window is essential for magnetron radar systems.

You can independently choose a sub-interval of burst pulse samples that are used by the AFC frequency estimator.

The AFC feedback loop is not constrained to use the same set of samples that are chosen for the FIR filter window. The FIR window typically is longer than the transmitted pulse, and thus, the samples contributing to the frequency estimate includes the leading and trailing edges of the pulse. These edges tend to have severe chirps and sidebands, compared to the more pure center portion of the pulse.

The AFC frequency estimate (which is power-weighted) may be misled by these edges and may not tune to the optimum center frequency if they were included.

Triggers

The lower portion of the plot shows the triggers created by RVP10. The triggers are shown as green lines. The number of triggers is set in the **Mt** menu (number of user-defined output triggers). `Trigger #1` is at the top. The triggers are drawn in their correct polarity and timing relative to each other, and relative to the burst sample window.

In the graphical interface, you can move all the triggers together. You can also do this on the command line with the **Pb** commands. On the command line, you can also move individual triggers. To do this, go to the individual trigger settings.

Time zero is indicated in the window. RVP10 defines `Range Zero` to occur at the center of the burst sample window. This also defines the zero reference point for the starting times of the six programmable triggers. For example, a trigger whose starting time is zero is plotted with its leading edge in the exact horizontal center of the display.

7.3.2 **Pb** subcommands

When using the passband filter, the available subcommands are:

```
Available Subcommands within 'Pb':
I/i               Impulse response length up/down
A/a & S/s         Aperture & Start of AFC window
L/l & R/r         Shift triggers Left/Right
T/t               Plot Time span up/down
Z/z               Amplitude Zoom
> and <   Start/Stop logging to rvp10-pb.log
%                 Toggle between IF input channels
.                 Single step
```

These subcommands change depending on whether the passband filter or the matched filter is configured. If the matched filter is selected the filter impulse response becomes the burst pulse length and the value may be lengthened or shortened using **P/p** keys.

Table 30 **Pb** subcommands

Command	Description
I/i or **P/p**	Increments or decrements the length of the passband filter's impulse response. Each keystroke raises or lowers the FIR length by one tap. OR Increments or decrements the pulse width definition. Each keystroke raises or lowers the pulse width by 0.1 microseconds for use in the idealized matched filter.

Command	Description
A/a & S/s	Raise/lower the aperture/start of the subwindow of burst pulse samples for AFC. If you do not use these commands, the full FIR window is used. Shortening the AFC interval results in two sample windows being drawn on the plot. Position the smaller AFC window positioned in the center portion of the transmitted pulse, where the burst amplitude and frequency are fairly stable.
L/l & R/r	Shift the group of RVP10 triggers left or right (earlier or later in time). The lowercase commands shift in 0.025 µsec steps, and the uppercase commands shift in 1.000 µsec steps (approximately). The reason for shifting all 6 triggers at once is that the relative timing among the triggers remains unchanged. However, the absolute timing (relative to range zero) varies, and this causes the burst pulse A/D samples to move within the sample window.
T/t	Increments or decrements the overall time span of the plot. The available spans are 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, and 5000 microseconds. The value is reported on the TTY as `PlotSpan`.
Z/z	Zooms the amplitude of the burst pulse samples so that they can be seen more easily. The value is reported on the TTY as `Zoom`.
> and <	The live plotted data can be logged to the `$(IRIS_LOG)/rvp10-pb.log` text file; one line per plot.
%	Toggle between the IFDR IF-Input sources.

7.3.3 TTY information lines in **Pb**

An example of TTY information lines:

```
Zoom :x2, PlotSpan:5 usec, FIR:1.36 usec (259 Taps) IF:Ch5(TXB)
 Freq:59.987  MHz, Pwr :-12.9 dBm, DC:0.14%, Trig#1:-5.00
```

Table 31 **Pb** TTY information lines

Information line	Description
`Zoom`	Indicates the magnification (in amplitude) of the plotted samples. A zoom level of "x1" means that a full scale A/D waveform exactly fills the height of the sample window. Generally, the signal strength of the burst pulse is not quite this high. Use larger zoom levels to see the weaker samples more clearly. You may zoom in powers of two up to x128.

Information line	Description
`PlotSpan`	Indicates the overall time span in microseconds of the complete scope display, from left edge to right edge.
`FIR`	Indicates the length of the impulse response of the passband FIR filter, and hence, the duration of the burst pulse sample window. The length is reported both as a number of taps, and as a time duration in microseconds. If you are using an idealized matched filter, this value must be the burst pulse length.
`IF`	Indicates which IF input source (SMA edge connector) is providing the data being plotted.
`Freq`	Indicates the mean frequency of the burst, derived from a fourth order correlation model. The control parameters for this model are set using the **`Mb`** command. See **`Mb`**— Burst pulse and AFC (page 86).
`Pwr`	Indicates the mean power within the full window of burst samples. DC offsets in the A/D converter do not affect the computation of the power, that is, the value shown truly represents the waveform's (Signal+Noise) energy.
`DC`	Indicates the nominal DC offset of the burst pulse A/D converter. This is of interest only as a check on the integrity of the front end analog components. The value should be in the range ± 2.0%.
`Trig#1`	Indicates the starting time of the first RVP10 trigger outputs. This number varies as the **`L`** and **`R`** subcommands cause the triggers to slew left and right. If the radar transmitter is directly fired by an external pretrigger, the pretrigger delay (in the form `PreDly:6.87`) is printed instead.

7.3.4 Adjusting burst pulse timing

You must calibrate the burst pulse timing separately for each pulse width. Each iteration is independent.

1. Set up the relative timing for all used RVP10 triggers.

   The trigger output lines are interchangeable, and each may be assigned to any function within the radar system.
   For example, `Trigger #1` might be the transmitter pretrigger, `Triggers #3` and `#4` might synchronize external displays, and `Triggers #2`, `#5`, and `#6` might be unused.

2. Choose an initial impulse response length of 1.5 times the transmit pulse width.

   This length is refined later when the passband filter is designed.
   See Burst spectra and AFC plot (Ps) (page 120).
   If using the matched filter, adjust the pulse length so that just the rising and falling edges of the burst are captured.

3. Adjust the plot time span to view a small region around the sample window, and set the initial amplitude zoom to x16.

   This assures that the plotted waveform is still noticeable, even if the burst signal strength is very weak.

4. Verify that the transmitter is radiating, and observe the burst pulse samples on the display.

   Use the **L** and **R** commands to shift the timing of all 6 triggers relative to range zero. This moves the burst sampling window relative to the transmitted pulse.
   Depending on whether the triggers are set properly, you may at first see nothing more than a flat line of misplaced A/D samples. However, after a few moments of hunting, the burst pulse should appear on the display screen.

5. Fine-tune the triggers so that the burst envelope is centered in the window, and adjust the amplitude zoom for a comfortable size display.

6. Isolate the clean center portion of the burst pulse isolated to a narrower sub-window of the overall FIR interval.

   Use the **A** and **S** commands to change the aperture and start of the narrowed region from which the AFC frequency estimators data are derived.

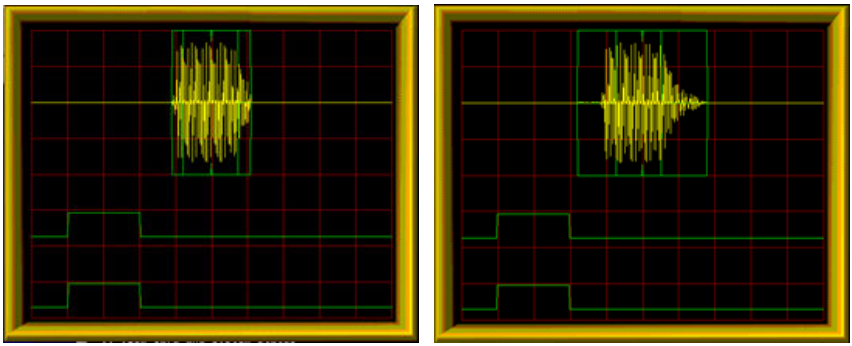


Figure 15 Burst timing plot for a properly captured burst pulse using a matched filter (on the left) and a passband filter (on the right).

7. Check that the burst pulse signal strength is reasonably matched to the input span of the IFDR A/D converter.

   The maximum analog signal level is +12 dBm. Exceeding this level produces distorted samples that would seriously degrade the algorithms for phase locking and AFC. However, if the signal is too weak, then the upper bits of the A/D converter are wasted and noise is unnecessarily introduced.
   Vaisala recommends a peak signal level between -3 dBm and +4 dBm, that is, a signal that might be viewed at x2 or x4 zoom.
   Take note of the burst energy level reported on the TTY. It helps decide the minimum energy threshold for a valid burst pulse.
   See **Mb**— Burst pulse and AFC (page 86).

7.4 Burst spectra and AFC plot (Ps)

Use the **Burst spectra and AFC plot (Ps)** to analyze the frequency content of the burst pulse, and to design an optimized passband filter, to view the matched filter bandwidth after windowing function for the pulse, and to align the AFC.

This process must be done for each new pulse width that is added to the system. Vaisala weather radars have four predefined pulse widths. In their case, the **Ps** plot is used for fine-tuning.

You can view three alternative graphs in the display:
- the actual spectrum of the burst signal
- the digital filter curve computed by the RVP
- a combination of these two

7.4.1 Burst spectra and AFC plot display

Horizontal and vertical axis: frequency and amplitude

The horizontal axis of the spectrum plot represents frequency. The overall span from the left edge, set at DC or 0 MHz, to the right edge is half the acquisition system clock frequency selected in the **Mc** menu. The frequency span is printed on the TTY when the command is first entered.

The vertical axis of the spectrum plot represents amplitude. The scale is logarithmic, and is marked with faint horizontal lines in 10 dB increments. An overall dynamic range of 70 dB can be viewed at once.

Horizontal lines

The horizontal lines contain major and minor tick marks to help calibrate the frequency axis.

Major marks are small downward triangles that represent integer multiples of 5 MHz; minor marks are in between and represent 1 MHz steps.

The power spectrum in the example is from a system with an intermediate frequency of 30 MHz. The left edge of the plot begins at DC, and the graph is centered on the sixth major tick, that is, at 30 MHz.

Horizontal line at the top of the plotting area

The horizontal line at the top of the plotting area is marked with an upward pointing major and minor tick to indicate the present value of the burst pulse frequency estimator.

The major tick is a triangle whose position along the horizontal axis corresponds directly to the estimated frequency. It should always be positioned directly over the main lobe of spectral power.

The minor tick gives finer scale resolution by indicating the fractional part of each 1 MHz multiple.

It is helpful to read the minor tick relative to the 10 horizontal division lines visible on most scopes. Motion of the minor tick is apparent even with very small changes in burst pulse frequency; a change of just 5 KHz can easily be seen. This means that you can observe the frequency drift of the magnetron in great detail, and watch the AFC behavior in real-time.

AFC line

The horizontal line at the top of the display indicates the value of the AFC control span.

The line contains an upward pointing major and minor tick, similar to the ones used to represent the burst frequency estimate on the line below. However, the horizontal axis here represents where in the total control span the AFC is currently set, and the overall span is the complete range of the AFC's digital-to-analog converter. The left edge would represent -100% and right edge +100% of the controllable span.

The major tick moves from the left edge to the right edge as the AFC varies from its minimum to maximum value. The minor tick traverses the screen at 10 times this rate.

7.4.2 Burst spectra and AFC plot example

The following figures show examples of burst spectra plots. The display screen is divided into 2 areas:

- The top portion (yellow single line) shows the present AFC level. Ticks on the yellow line show the observed frequency and the frequency requested by the signal processor.
- The lower portion shows power spectrum plots of the burst pulse and/or the theoretical matched filter or passband filter response.

This figure is an example of a single filter response plot:

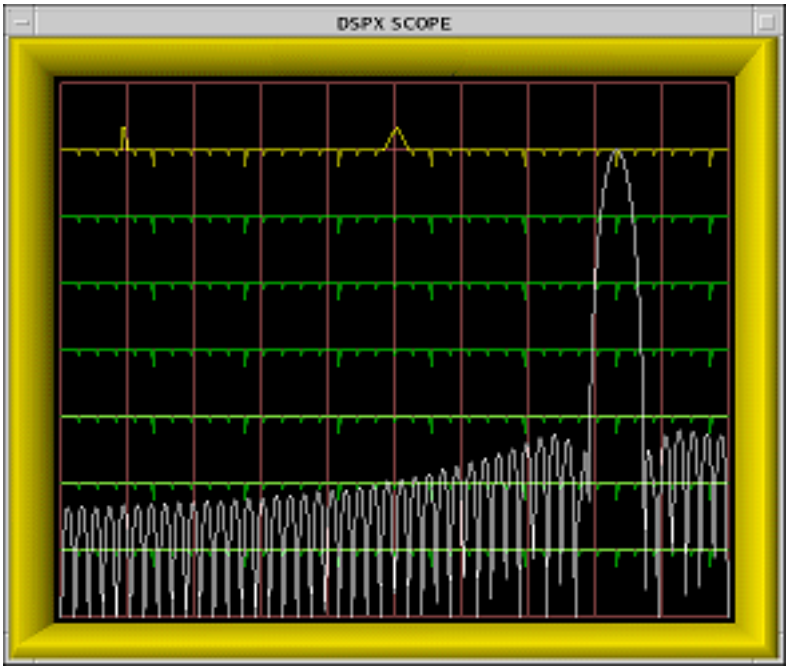


Figure 16 Example of a filter with excellent DC rejection

The following figure shows a combined display of the actual measured spectrum of burst signal and the digital filter curve computed by RVP. The combined display makes it easy to compare the filter being designed with the live waveform that it is intended to selectively pass.

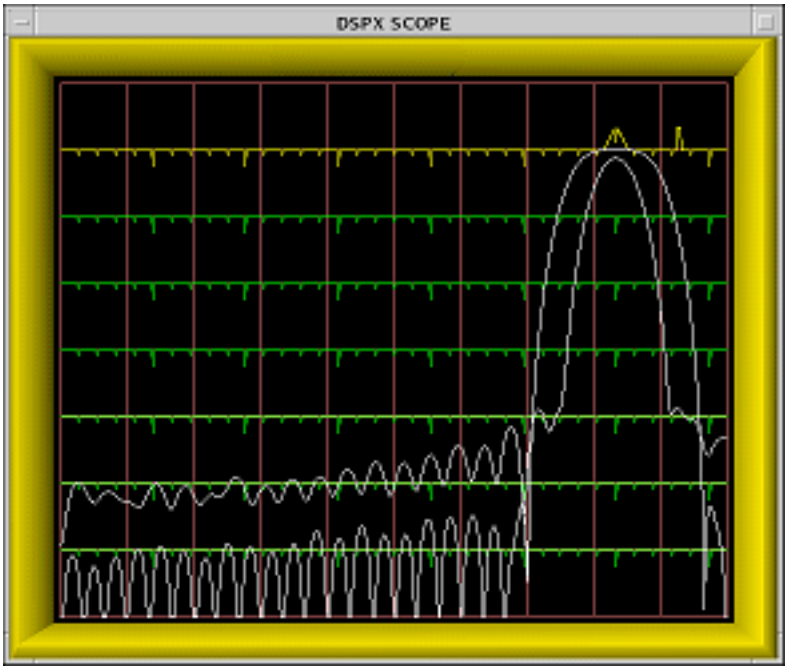


Figure 17 Example of a filter with a poorly designed passband filter

The filter's frequency response is always drawn with its passband peak touching the top of the plot. The vertical height of the burst spectrum varies with signal strength but can be adjusted using the **Z** subcommand.

7.4.3 **Ps** subcommands

Ps subcommand functions:

- change the design of the passband FIR filter, or select which windowing function to use with an idealized matched filter
- help calibrate the AFC loop
- and alter the display format

Available subcommands when using the **self-configured passband filter**:

```
Frequency span of the plot is 0.0 MHz to 120.0 MHz.
Available Subcommands within 'Ps':
  I/i         Impulse response length Up/Down
  N/n & W/w   Filter bandwidth Narrower/Wider
  U/u & D/d   MFC Up/Down (On/Off '=')
  A/a & S/s   Aperture & Start of AFC window
  #/*/&       Print filter coefficients/FreqResp/Dump everything to temp file
  V/v         Number of spectra averaged
  Z/z         Amplitude zoom
  <space>     Alternate Plots
  %           Toggle between IF input channels
  -           Single Step
  X/x         X axis zoom
  L/l & R/r   Shift plot left/right along the x-axis
  b           Toggle labels
```

Available subcommands when using the **matched filter**:

```
Frequency span of the plot is 0.0 MHz to 120.0 MHz.
Available Subcommands within 'Ps':
  P/p         Pulse (burst window) length Up/Down
  F/f         Iterate Up/Down through window types
  E/e         Edge (tapering) from 0 to 0.5 Up/Down
  U/u & D/d   MFC Up/Down (On/Off '=')
  A/a & S/s   Aperture & Start of AFC window
  #/*/&       Print filter coefficients/FreqResp/Dump everything to temp file
  V/v         Number of spectra averaged
  Z/z         Amplitude zoom
  <space> Alternate Plots
  %           Toggle between IF input channels
  -           Single Step
  X/x         X axis zoom
  L/l & R/r   Shift plot left/right along the x-axis
  b           Toggle labels
```

Table 32 **Ps** subcommands

Command	Description
I/i OR **P/p**	Increments or decrements the length of the passband filter's impulse response. Each keystroke raises or lowers the FIR length by one tap. Often the passband filter's characteristics can be improved by changing the FIR length by one or two taps. Experiment with this as you design your filter. OR Increments or decrements the pulse width definition. Each keystroke raises or lowers the pulse width by 0.1 microseconds.

Command	Description
N/n & W/w OR **F/f & E/e**	Changes the bandwidth of the passband filter to make it narrower or wider. The lower case commands make changes in 1 KHz steps. The upper case commands use 100 KHz steps. The value is reported on the TTY as **BW**. Often a small change in passband width shifts the exact locations of the filter's zeros, and may improve the DC rejection. OR When using an idealized matched filter, the bandwidth is calculated for you and can only be adjusted by applying windowing functions which change the NEBW. Each **F/f** key press iterates forwards or backwards through the list of available functions. The number of edge points defines the strength of one side of the windowing function, which is replicated to the opposite side. Thus a value of **50%**, when replicated to the opposite side, means all samples in the time domain are windowed. The setting of **50%** is recommended for all windows.
U/u & D/d	Implements the Manual Frequency Control (MFC) override, and allow the RVP10/IFDs AFC output voltage to be manually set to any fixed level. The lower case commands make changes in 0.05 D-Unit steps, and the upper case commands use 1.0 D-Unit steps. The value is reported on the TTY as **AFC**.
=	Toggles MFC mode or and off. A warning is printed if you exit the **Ps** command while MFC is enabled, and you are given a chance to re-enable AFC.
#	Prints the coefficients of the current FIR filter. The values are scaled by the coefficient with the largest absolute value, so that they all fall in the 1 ... +1 range. When printing the coefficients of the matched filter, the values after application of the windowing function are given. You can use this information to model the filter behavior for point targets that fall between discrete range bins, for example, when performing a radar sphere calibration. See FIR (matched) filter (page 155).
*****	Sends content to a ***.tmp*** file.

Command	Description
&	Saves Tx waveform and Rx filter coefficients as well as parameters defining the *$IRIS_DATA/temp/pwd_<pulse_number>.dat* file, usually */usr/iris_data*. Files are in Octave data format. All data files contain the RVP10 clock frequency (variable **RVP_clock_MHz**) pulse IF frequency (variable **IF_freq_MHz**). CW pulse data contain the **CW_bandwidth_MHz** variable. NLFM pulse data contain the following pulse variables: **NLFM_bandwidth_MHz**, **NLFM_pulsewidth_usec**, **NLFM_param_0**, **NLFM_param_1**, and **NLFM_param_2**. Each data file contains the following waveforms and filters: • **tx_waveform_0** - Zero frequency centered Tx waveform • **tx_waveform_if** - Tx waveform up-converted to IF • **rx_filter_0** - Zero frequency centered Rx filter • **rx_filter_if** - Rx filter up-converted IF If a pulse has an associated chained pulse, there are the following additional waveforms: • **tx_chained_waveform_0** - Zero frequency-centered combined Tx waveform • **tx_chained_waveform_if** - Tx waveform up-converted to the main pulse IF
V/v	Increments or decrements the number of burst pulse spectra that are averaged together to create the plot. The count ranges from one (no averaging) to 25, and is reported on the TTY as **Navg**.
Z/z	Zooms (on a logarithmic scale) in 1.0-dB steps the amplitude of the burst pulse spectra. This is useful when the overall 70 dB plot span is not sufficient to hold the full range. Zoom can also be used to line up the burst spectrum with the filter response so that they can be compared. The zoom level is not printed on the TTY.
X/x	Zooms along the x-axis reducing the frequency span of the plot. This is useful for displaying a plot with a higher resolution of information.
L/l & R/r	Shifts the plot left/right along the x-axis. These commands move the center of the x-axis left (downwards in frequency) or right (upwards in frequency) in small or large steps. This is useful after zooming in order to re-center the spectrum of the burst pulse power within the domain of the frequency being plotted.

Command	Description
b	This add labels to the plot showing the total frequency span and the frequency values of the major tick marks being plotted.
< space >	The space bar alternates between the following options for the type of plotted spectra: • FIR frequency response • Burst pulse spectrum • Both
%	IF input channel selection. In dual-receiver mode, the **%** command toggles between each receiver. The printed status line is prefixed with **Rx:Pri** or **Rx:Sec** according to the selected receiver. Type **%** to toggle the plot of the FIR filters frequency response, and the printout of its **DC-Gain**. The plotted spectrum and printed power levels are based on the sum of all input signals and do not change with **%**.
-	Single step

7.4.4 TTY information lines within **Ps**

An example of the TTY information lines:

```
Navg:3,  FIR:1.10 usec (259 Taps), Edge:0.100, Win: Tukey,
BW:0.889 MHz, NEBW: 0.980 MHz, DC-Gain: Unknown

Freq:59.966 MHz, Pwr:-12.76 dBm, PkBrstPwr(PriRx: -7.12 dBm,
SecRx: -7.10 dBm), RxLoss:1.627 dB, fltMtch:0.215 dB,
brstEg:0.05896 nJ, fltBrstEg:0.04054 nJ, AFC:28.22% [1822] (Locked)
```

Table 33 **Ps** TTY information lines

Information line	Description
Navg	The number of burst sample spectra that are averaged together prior to plotting. Larger amounts of averaging increase the ability to see subtle spectral components, but the display updates more slowly.
FIR	Indicates the length of the impulse response of the passband FIR filter, and hence, the duration of the burst pulse sample window. The length is reported both as a number of taps and as a time duration in microseconds. If using an idealized matched filter, this value should be the burst pulse length. See TTY information lines in**Pb** (page 117).

Information line	Description
`Edge`	Fractional percentage of edge points being used within the windowing function.
`Win`	Name of the windowing function in use.
`BW`	The 3 dB bandwidth of the passband filter. This is the complete width of the passband from the lower frequency edge to upper frequency edge. Note that the filters center frequency is fixed at the radars intermediate frequency, as chosen in the **`Mb`** setup command. Both requested (as would be calculated from a perfectly matched filter to the pulse length) and actual bandwidth of the pulse are shown.
`NEBW`	Noise Equivalent Bandwidth (NEBW) is the bandwidth of a rectangular window that would pass the same amount of power from flat noise as the current filter does.
`DC-Gain`	The filter's response to DC (zero frequency) input. The value is a negative number in decibels, or the word `ZERO` if the filter has a true zero at DC. The filters DC gain should be kept at a minimum so that fixed offsets in the A/D converters do not propagate into the synthesized `I` and `Q` values.
`IF`	Shows the name of the channel being plotted. It is possible to change channels being plotted using the %
`PkBrstPwr`	The maximum power in a burst pulse. If burst pulse is configured to be taken from Ch. 5 of the IFDR, then **`PriRx`** and **`SecRx`** display the same value (**`Ch. 5`**). If burst pulse is configured to be taken separately for H and V polarizations, **`PriRx`** and **`SecRx`** display the burst power values of Ch. 1 and Ch. 2, respectively. Ch. 1 is typically used for H polarization, and Ch. 2 for V polarization.
`rxLoss`	The filter loss is a measure of how much information (or energy loss) is discarded by the filter. See Computing filter loss (page 130).
`fltMtch`	This value is displayed for debugging purposes and does not have any direct physical meaning.
`brstEg`	The energy of the full burst sample window.
`fltBrstEg`	The energy of the filtered burst.

Information line	Description
AFC	The level and status of the AFC voltage at the IFDR module. The number is the present output level in D-Units ranging from 100 to +100. The shorter % symbol is used since percentage units correspond in a natural way to the D- Units. An additional number in square brackets is printed to the right of the AFC level to show the encoded bit pattern which corresponds to that level. This appears when RVP10 detects a special digital format. That is, when the back panel phase-out lines have been configured for AFC, or when any of the following are not true: 1. The low and high numeric AFC span is -32768 ... +32767 2. The uplink is enabled 3. The uplink format is binary 4. The pinmap protocol is OFF Binary format is printed in base-10, BCD format is printed in Hex, and 8B4D format is printed with the low 16-bits (four BCD digits) in Hex and the upper bits in base-10. The AFC modes shown to the right of the numerical value(s), and can take the following states: • `(Disabled)` Indicates that neither AFC nor MFC are enabled. The output voltage remains fixed at 0% (center of its range). • `(Manual)` Manual Frequency Control (MFC) is overriding AFC. The **U** and **D** commands can be used to slew the voltage up and down. If any of the following states appears, it implies that AFC is enabled and that MFC is disabled: • `(NoBurst)` The energy in the burst is below the minimum energy threshold for a valid pulse. The AFC loop remains idle. • `(Wait)` The burst pulse has become valid just recently, but the AFC loop is idle until the transmitter stabilizes. • `(Track)` The burst pulse is valid, and the AFC loop is tracking in order to bring the burst frequency within the inner hysteresis limits. • `(Locked)` The burst pulse is valid and the AFC loop is locked. The burst frequency is now within the outer hysteresis limits and has previously been within the inner limits while tracking. This is the stable operational mode in which data acquisition should take place.

PkBrstPwr, **rxLoss**, **fltMtch**, **brstEg**, and **fltBrstEg** are displayed if the overall burst power exceeds the minimum valid burst threshold set in the **Mb** command. (It would not be possible to compute these values when the burst waveform is missing.) These values are computed from the current channel, indicated in the **IF** field of the information line (for example: **IF: Ch. 1**).

7.4.5 Computing filter loss

RxLoss is a measure of how much information (or energy) is discarded by the filter. This allows you to correct the radar calibration for the difference between what a broad-band power sensor measures as the overall transmit energy, and what the RVP10 narrow-band receiver detects within its passband. The filter loss is a subtle quantity that depends on the combined characteristics of both the transmit waveform and the receiver FIR filter.

RVP10 computes these losses in two different ways. The loss displayed in **Ps** is based on sampling and filtering the actual burst pulse while the loss displayed in **Pa** is computed using the waveform and filter vector data of the signal processor.

Let **b** be the transmitted burst pulse vector. This means the voltage samples of the entire burst pulse at the actual sampling rate of the IFDR's burst channel. Filter loss is computed by square of **b** over square of filtered **b** (convolution of **b** and **f**):

$$Loss = \frac{|b|^2}{|b \otimes f|^2}$$

Because burst pulse has real values, it has two frequency peaks. Only one of them matches the filter frequency. To compensate the double loss, the computed **Loss** above is divided by 2:

$$RxLoss_{dB} = 10\ log_{10}\left(\frac{Loss}{2}\right)$$

7.4.6 Adjusting burst spectra and AFC plot when using passband filter or matched filter

1. Make sure you have successfully captured the burst pulse with the **Pb** command.
2. Type the **Ps** command.
3. Use the space bar to display the burst spectrum plot by itself, and use the **Z** key to shift the entire graph into view.

   The plot shows the frequency content of the transmitted pulse.

   a. Check that the plot shows a clean main power lobe centered at the receivers intermediate frequency.

   b. Check the spectrum for spurious harmonics, excessive width, and other out-of-band noise.

   c. Adjustment the transmitter to give a sharper main lobe or reduce spurious noise.

4. Begin designing the passband FIR filter.

   Use the space bar to display both the filter response and the burst spectrum on the same plot.
   Use the **Z** key to shift the bursts main lobe up to the top horizontal line of the graph. This makes it level with the filters peak lobe, which is always drawn tangent to the same top line.

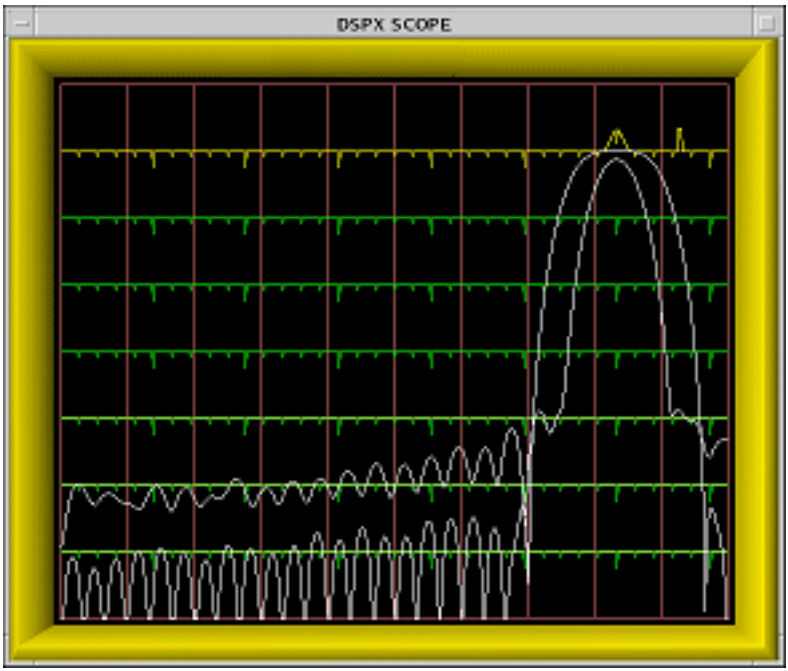


Figure 18 Example of a poorly designed passband filter

   Begin with the FIR length that was chosen previously in the **Pb** command. When using the passband filter, begin with the **N** and **W** keys to set an initial bandwidth equal to the reciprocal of the pulse width.

5. The main lobes of the two plots should more-or-less overlap. Experiment with changing the FIR length and bandwidth to achieve a filter with the following properties.

   - The filter width should be no greater than the burst spectral width. A wider passband reduces the SNR of the received signal because out-of-band noise would be allowed to pass.
   - The DC gain should be as small as possible, preferably less than -65 dB.
   - If there are conspicuous interference spikes at particular frequencies, try to adjust the location of the filter's zeros so that the interference is maximally attenuated.

6. Optimize the performance of the FIR filter.

   When using the matched filter, much of this work is automated for you. Make sure that the pulselength value in the **mw** menu section is set correctly, so that the entirety of the pulse is captured in the **pb** plot. The matching bandwidth given the pulse length will be calculated and applied to the matched filter for you. You may also use the **F** key to iterate through the windowing functions to find the one that provides the best matching shape to your burst pulse.
   The filter should not pass any frequencies that do not contain useful information from the original transmitted pulse. If anything, choose a filter whose width is slightly narrower than the bursts spectral width.
   The previous figure shows an example of a filter that is poorly matched to the pulse. Although the filter has fairly good DC rejection, it passes frequencies that are outside of the transmitter's broadcast range. These frequencies only bring noise to the synthesized `I` and `Q` data stream.
   You can optimize the passband FIR filter manually:
   Defining a nearly optimal filter requires a few minutes of hunting with the I, W, and N keys. Each time you press any of these keys, RVP10 designs a new FIR filter from scratch, and displays the results. Even though you must still control two degrees of freedom (length and bandwidth), the RVP10 design calculations perform several hundred iterative steps each time, which preferentially select for the best filter. Because the FIR coefficients are quantized in the filter chips themselves, the process of finding an optimal filter becomes quite nonlinear.

Example

The offset error of the IFDR A/D converter is at most 10mV, that is, -27 dBm into its 50 Ω input.

To achieve 90 dB of dynamic range below the converters +8 dBm saturation level, we expect usable `I` and `Q` values to be obtainable from a (sub-LSB) input signal at -82 dBm. This is 55 dB below the interference that would result from the worst-case A/D offset.

But a weak input signal at -82 dBm would still be damaged by even an equal level of DC interference. Therefore, adding another 10 dB safety margin, we get -65 dB as the recommended maximum DC gain of the matched filter. This DC gain should be reduced even further if it is known that coherent leakage is present in the receive signal at a level greater than the -27 dBm worse-case A/D offset.

The following figure shows a 60 MHz filter with particularly poor (-42 dB) DC rejection. The frequency range of the plot is 36 MHz to 72 MHz, therefore, DC appears aliased at the right edge and we can see a peak in the filters stopband at DC. Contrast this with the filter shown previously that has a true zero at DC. In general, a poor filter can be significantly improved by making only incremental changes to the impulse response length and/or desired bandwidth.

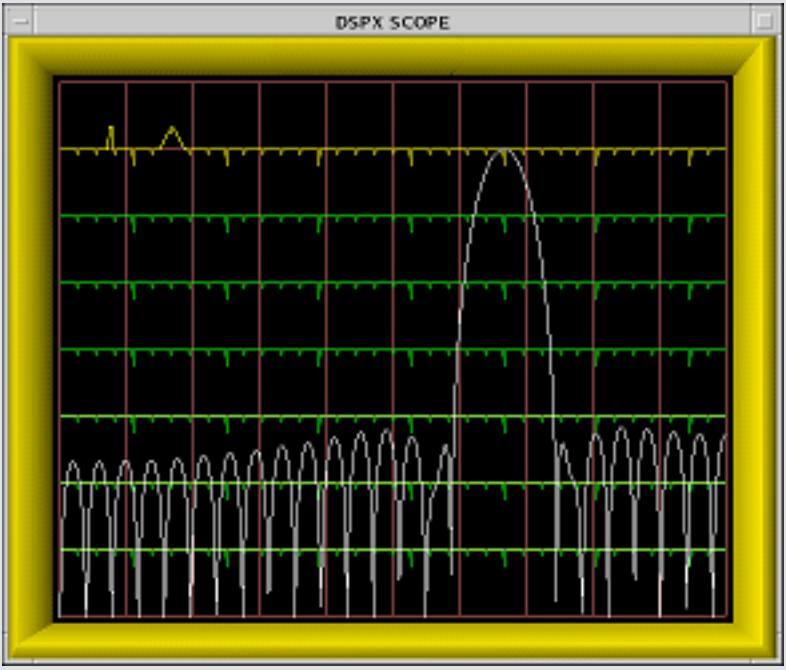


Figure 19 Example of a Filter With Poor DC Rejection

7.5 Transmitter waveform ambiguity plot (Pa)

RVP10 can make radar observations using compressed pulse waveforms. This opens many new opportunities for using low-power solid-state transmitters that employ very long pulse lengths (20 – 200 µsec). However, the signal processing and waveform design required to make good use of these long transmit pulses is also much more complex. To help with this, RVP10 provides the **Pa** (plot ambiguity) command in which compressed transmit waveforms can be designed, studied, and optimized. In the **Pa** plots, you can experiment with different waveform designs, try out bandwidth and pulse width options, and examine and optimize the range/time sidelobes of your waveform.

7.5.1 Ambiguity plot display

The Figure 20 (page 134) shows an example of the **Pa** plot showing the ambiguity function of the digital receiver for an NLFM compressed waveform. In the figure, the pulse width is 90 µsec, bandwidth is 4 MHz, PSL is -52 dB, and ISL is -50.8 dB.

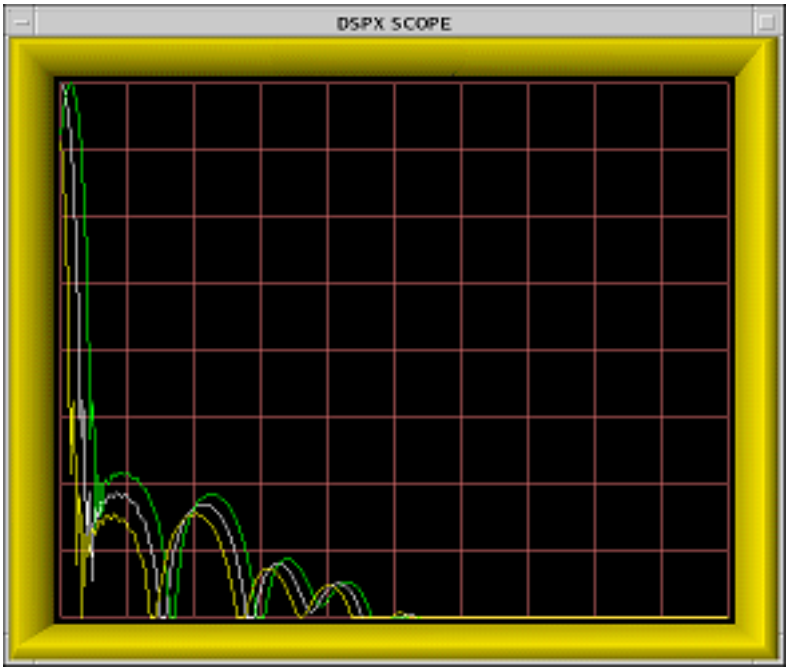


Figure 20 Ambiguity diagram of a compressed Tx pulse

In this example, the **vertical axis** can indicate several things, depending on the form of the plot. In the example above, it indicates the magnitude of the Tx/Rx range sidelobes on a logarithmic scale, where you can see markings in 10 dB steps.

The **horizontal axis** shows the length of the pulse in the time domain. In the example above, the horizontal span of the plot is equal to the length of the pulse, and consequently, only half of the complete ambiguity diagram is shown. The x-axis can also span the entire duration of the pulse, as shown in the next example.

The **white** plot line shows Zero-Doppler response in the digital filter.

The**yellow**and **green** plot lines show the Tx/Rx responses when the overall waveform is modified by a 50 KHz target Doppler shift. Real weather targets never have such a large Doppler component, but the **Pa** menu enables you to study its effect.

Frequency, phase, and amplitude of a compressed Tx pulse

Figure 21 (page 135) shows an alternate form of the **Pa** plot of the same Tx waveform as above.

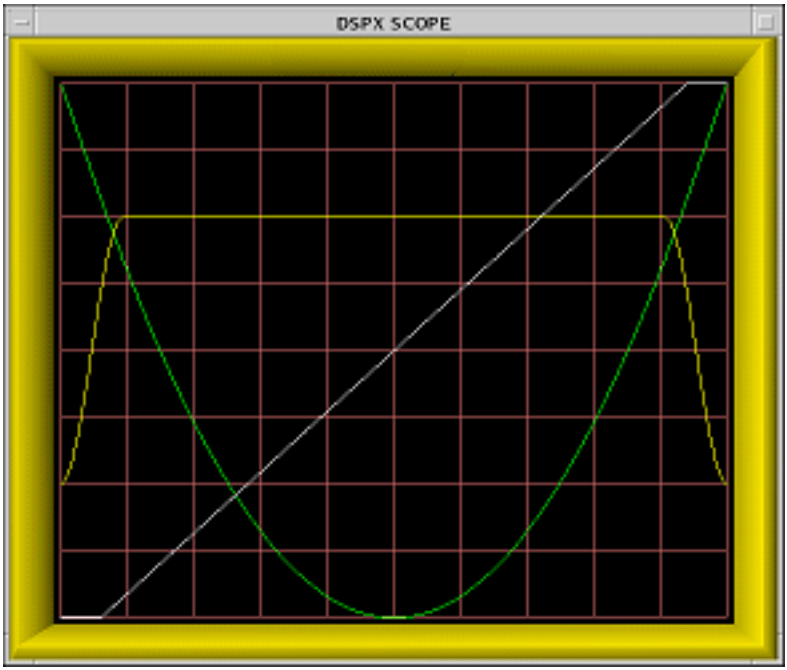


Figure 21 Frequency, phase, and amplitude of a compressed Tx pulse

By pressing **Space**, an alternate version of the ambiguity plot shown in Figure 21 (page 135) is displayed, which is showing properties of the transmit waveform. The horizontal axis again represents time.

Three plots are drawn, and the vertical axis is interpreted differently in each case:

- The instantaneous frequency across the full length of the pulse is shown in white. The vertical scale is normalized to hold the overall frequency span, which is also shown numerically in the **Pa** TTY output.
- The waveform baseband phase is shown in green and is normalized so that the vertical axis holds the full span of values. Note that the phase, which generally spans a few thousand degrees, is "unwound" in this plot so that you can see its behavior.
- The amplitude of the Tx waveform envelope is shown in yellow. It is drawn using a linear vertical scale which occupies only the middle half of the plot. This is to avoid creating too much "plotting clutter" in the corners.

The figure shows that the waveform consists of a non-linear FM chirp with a tangent function. The frequency changes abruptly at the start and end of the pulse with near-linear modulation in the middle.

The central chirp is contained within a larger amplitude modulation envelope that applies full scale power within the middle 82% of the pulse, and also provides band-limited shaping of the leading and trailing edges.

7.5.2 **Pa** subcommands

Pa subcommands change the bandwidth, pulse width, and shaping parameters of the transmit waveform, and alter the format of the display:

```
Available subcommands within 'Pa':
F/f         Iterate Up/Down through window types
E/e         Waveform Edge (aka tapering) from 0 to 0.5 Up/Down
A/a         Window alpha parameter (for Kaiser, Blackman or Chebyshev windows)
R/r & L/l   Shift plot       Right/Left
S/s & B/b   Pulse Length    Smaller/Bigger
N/n & W/w   Bandwidth       Narrower/Wider
D/d & U/u   Non-linearity parameter Down/Up
V/v         Doppler Frequency Shift Up/Down
T/t         Time span zoom
Z/z         Amplitude Zoom
<space>     Alternate Plots
```

Table 34 **Pa** subcommands

Command	Description
F/f	Select the windowing function of the transmitter side. This control the window used to perform Amplitude Tapering of the chirp. The upper case command scrolls the list of the windowing functions forwards. The lower case command scrolls the list of the windowing functions backwards.
E/e	When the window has been selected, use these commands to modify the sharpness of the edge (how much of the frequency domain this window is applied to).

Command	Description
A/a	If the transmit window selected is a Kaiser, Blackman, or Chebyshev window these keys allow adjusting the alpha parameter or shape of the window used for amplitude tapering.
R/r & L/l	The R/r" and L/l commands shift the ambiguity plot right and left, respectively. The lower case commands shift the plot in 0.25 µsec steps. The upper case commands shift the plot in 10 µsec steps.
S/s & B/b	The **S/s** ("smaller") command decreases the time duration (pulse length) of the transmitter pulse. The **B/b** ("bigger") command increases the time duration (pulse length) of the transmitted pulse. The lower case commands use 0.05 µsec steps. The upper case commands use 1 µsec steps.
N/n & W/w	The **N/n** ("narrower") and **W/w** ("wider") commands decrease or increase the bandwidth of the transmitted pulse, respectively. The lower case commands shift in 10 kHz steps. The upper case commands use 200 kHz steps.
D/d & U/u	Decrease or increase the non-linearity parameter of the chirp. The lower case commands use 0.001 (dimensionless) steps. The upper case commands use 0.05 steps. Present value is printed each time a Down/Up key is pressed, for example: `Non-linearity parameters: 0.9000`
V/v	Shows how the overall compressed pulse Tx/Rx system responds to the effects of target Doppler shift. You can introduce frequency shifts as large as 100 kHz in order to see their effect on the Range/Time side lobes. The **Pa** plot shows the effects of +V and V shifts as green and yellow plots in addition to the standard white (zero Doppler) plot.
T/t	Increments or decrements the time duration of the window of the ambiguity plot.
Z/z	The dynamic range of the **Pa** side lobe plot is 80 dB. Usually this gives plenty of room to examine the properties of the waveform. For very wide dynamic range pulses, you can shift the plot up/down in 10 dB steps using this command. By changing the y-axis to have 5 dB or 20 dB resolution, resulting in 40 dB or 160 dB span.

7.5.3 TTY information lines in **Pa**

An example of the TTY information lines in **Pa** plot:

```
BW :3.40MHz PW:90.20 usec PSL:61.2dB ISL:51.3dB RxLoss:5.82
FIR:90.20 usec (18943 Taps), Edge:0.100, Win: Kaiser, Alpha 0.00
Non-linearity parameter: 0.0990
```

Table 35 **Pa** TTY information lines

Information line	Description
`BW`	Bandwidth of the Tx waveform in MHz.
`PW`	Pulse width (pulse length) of the Tx waveform in microseconds. If two pulses are within the transmit sequence currently in use, both pulse lengths will appear. For example, a 90 µsec pulse followed by a 3 µsec pulse will be shown as 90.00/3.00.
`PSL`	Peak sidelobe level of the ambiguity diagram, expressed in dB relative to the main lobe level. This is the peak height of the strongest range/time sidelobe, and it measures the ability of the compressed pulse to distinguish a given target from a small number of individual point targets that also lie within the pulse volume. The ability of the waveform to "see" between clutter targets is largely determined by the PSL level.
`ISL`	Integrated sidelobe level of the ambiguity diagram, expressed in dB relative to the main lobe. This is the total power in all range/time sidelobes divided by the total power in the main lobe. ISL measures the ability of the compressed pulse to distinguish a given target from other distributed targets, such as rain, that also lie within the pulse volume.
`RxLoss`	`RxLoss` is a measure of how much information (or energy) is discarded by the filter. This allows you to correct the radar calibration for the difference between what a broad-band power sensor measures as the overall transmit energy, and what the RVP10 narrow-band receiver detects within its passband. The filter loss is a subtle quantity that depends on the combined characteristics of both the transmit waveform and the receiver matched filter. RVP10 computes these losses in two different ways. The loss displayed in a **Ps** plot is based on sampling and filtering the actual burst pulse, while the loss displayed in a **Pa** plot is computed using the waveform and filter vector data of the signal processor. If **a** is the transmitted waveform vector, and **f** is the filter vector, then filter loss is computed by square of **a** over square of filtered **a** (convolution of **a** and **f**): $RxLoss_{dB} = 10\,log_{10}\left(\frac{|a|^2}{|a \otimes f|^2}\right)$
FIR	This is the length if the matched receive filter in usec and the number of taps within the filter.
Edge	Current value of Waveform Edge (aka tapering) from 0 to 0.5.
Win	Current windowing function being applied to transmit waveform for amplitude tapering.

Information line	Description
Alpha	Current alpha parameter if window being applied is Kaiser, Blackman, or Chebyshev.
Non-linearity parameter	Current value of the amount of non-linearity being applied to the chirp phase.

Given a compressed transmit waveform, RVP10 designs the appropriate mismatch Rx filter automatically, using an optimized Blackman window in all cases. Developers can also access the internal APIs directly to design any desired transmit waveform along with the associated FIR filter to receive it.

7.6 Receiver waveforms plot (Pr)

Use the **Receiver waveforms (Pr)** plot to check the signal received by the radar antenna and IFDR. The goal is to verify that the received signal is clean and appropriately scaled, and that nearby targets can be seen clearly.

In the **Pr** plot screen, you can view four different plots:

- Received raw sample as a function of time
- Received filtered power as a function of time
- Both of the above as a function of time
- Frequency spectrum of the received samples

The parameter shown on the horizontal axis varies according to which of the alternative plots viewed.

7.6.1 Interpreting receiver waveform plots

Display

The vertical axis represents the amplitude of the received signal.

The horizontal axis can represent time (range) or frequency, depending on which of the alternative plots is viewed. In case of time (range), the initial offset and span are editable.

The white and green graph lines show the received signal data.

In the frequency spectrum plot, the major ticks on the horizontal green lines have an interval of 5 MHz, and the minor ticks an interval of 1MHz.

Example 1

The following figure shows an example of a receiver waveform plot. In this figure, the horizontal axis represents time. The data is acquired from a single transmitted pulse, and plotted both as raw IF samples and as the LOG of the detected power using the FIR filter for the current pulse width.

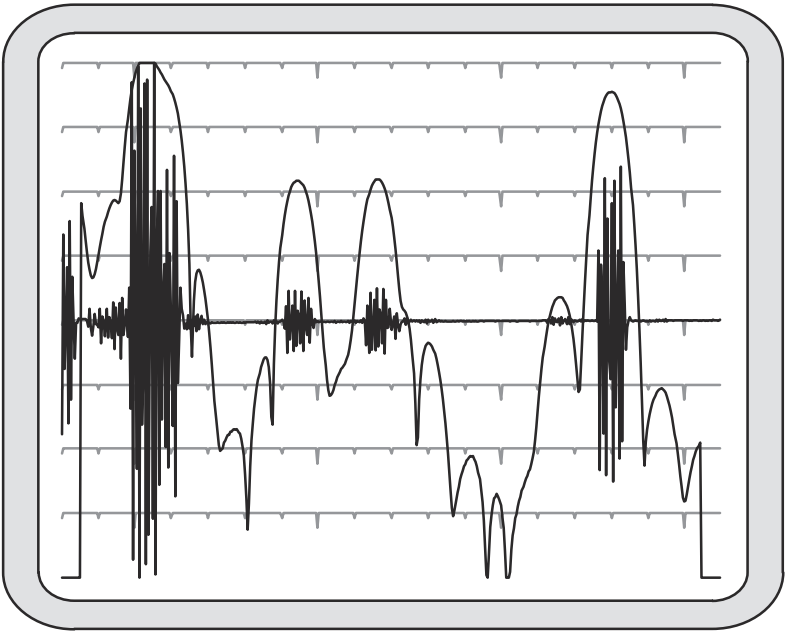


Figure 22 Example of combined IF sample and LOG plot

The IF samples are plotted on a linear scale as signed quantities, with 0 appearing at the center line of the scope. Any DC offset present in the A/D converter is not removed, and is seen as a shift in the baseline at higher zoom levels. For example, the converter's worst-case DC offset of 10 mv would appear as a several-hundred-count offset in the 16-bit A/D range. At the x32 or higher zoom scales, this offset would peg the sample plot off scale. Typically, the DC offset is much less than this worst case value, but RVP preserves the DC term in the **Pr** sample plot so that its presence is not forgotten.

The AC amplitude of the IF samples increases when targets are present. On top of these samples, the detected power is drawn on a logarithmic scale. Each horizontal line represents a 10 dB change in power. The graph is scaled so that the LOG power reaches the top display line when the samples occupy the full amplitude span. Using the previous figure as an example, the two equal-power targets just to the left of center are approximately 18 dB down from the top. The amplitude of the samples is $10^{(-18/20)} = 0.13$, that is, 13% of full scale. This correspondence between the LOG scale and the amplitude scale applies regardless of the plot's zoom level. As the IF samples are zoomed up and down by factors of two, the LOG plot shifts up and down in 6 dB steps.

The LOG plot is obtained by convoluting the FIR filter coefficients with the raw IF data samples, and then plotting log ($I^2 + Q^2$) at each possible offset along the sampling interval. This convolution produces only (`1 + N− I`) output points, where `N` is the number of sample points and `I` is the length of the FIR filter. For this reason the LOG plot begins approximately I/2 samples from left side and ends approximately I/2 samples from the right.

The LOG points are computed at each possible offset within the raw IF samples. The LOG plot gives a very detailed view of received power versus range. Of course, successive LOG points are highly correlated because successive input data intervals differ by only one sample point. This is why the LOG plots appear smooth compared to the instantaneous variation of the raw IF samples.

As the starting offset of the **Pr** plot is decreased to range 0, you begin to see part of the burst pulse (the second half of it) appear at the left edge of the plot. This is because the burst data samples are multiplexed onto the same fiber cable that carries the IF data samples. Zero range is defined to occur at the center of the burst window; the latter half of the burst pulse is visible when the plot begins at range 0.

Example 2

The following figure shows a **Pr** display with a frequency spectrum of the received data samples in a format that is nearly identical to the **Ps** display.

- The horizontal axis represents the same band of frequencies (half the sampling rate).
- The vertical axis represents power in 10 dB steps. The entire vertical axis is used so that an overall span of 80 dB is visible.

In this example, the time span is set to 50 µsec, with a 1 m antenna attached to the IF input so that a broad range of signals (radio stations, electrical noise, and so on) would be detected.

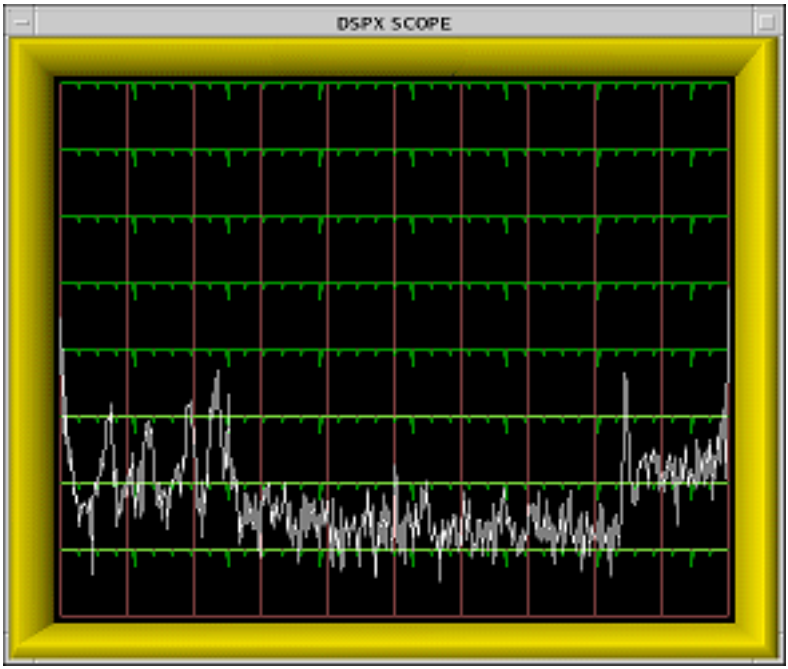


Figure 23 Example of a noisy high resolution Pr spectrum

The purpose of the **Pr** power spectrum is to check for spurious interference in the IF signal from the radar receiver. View the spectrum with the transmitter turned off, and with the starting range moved out so that the burst samples are not mixed with the receiver data.

The power spectrum is computed using the complete interval of raw IF samples which, depending on the chosen time span, may contain many hundreds of points. The frequency resolution of the **Pr** spectrum can be quite fine to make it possible to discern any interfering frequencies with some detail.

The **Pr** spectrum plot shows a 0 Hz peak from any DC offset in the A/D converter, and is consistent with how the DC offset is presented in the **Pr** sample plot.

Both plots preserve the DC component of the IF samples so that it can be monitored as part of the routine maintenance of the receiver system. This is one of the few places in the RVP10 menus and processing algorithms where the DC term deliberately remains intact.

7.6.2 Available subcommands in **Pr**

These subcommands change the start time and span of the IF sampling window, and alter the format of the display.

Available subcommands:

```
Available Subcommands within  Pr :
L/l & R/r       Start range Left/Right
T/t             Plot time span Up/Down
V/v             Number of spectra averaged
Z/z             Amplitude zoom
<space>         Alternate Plots
> and <         Start/Stop logging to rvp10-pr.log
%               Toggle between IF input channels
-               Single Step
```

Table 36 **Pr** subcommands

Command	Description
L/l & R/r	Shift left and right the starting point of the window of IF samples. The lower case commands shift in 0.25 µsec steps, and the upper case commands use 10 µsec steps. The starting point is displayed both in microseconds and kilometers on the TTY, and is not allowed to be set earlier than range zero.
T/t	Increments or decrements the time duration of the window of IF samples. The window is not allowed to become shorter than the impulse response length of the FIR filter, since that would preclude calculating even a single LOG power point. The value is reported in microseconds on the TTY, and the largest permitted span is 50 µsec.
V/v	Increments or decrements the number of power spectra that are averaged together to create the plot. The count ranges from one (no averaging) to 25, and is reported on the TTY as **Navg**.
Z/z	Zooms the amplitude of the IF samples by factors of two from one to 128. The LOG plots are shifted in corresponding 6 dB increments as the amplitude is zoomed up and down. The zoom level is reported on the TTY so that absolute power levels and A/D usage can be assessed.
< space >	The space bar alternates among three choices for the type of data that are plotted: Received Samples, Received Samples and LOG Power, and Received Power Spectrum.
> and **<**	The live plotted data can be logged to the *$(IRIS_LOG)/rvp10-pr.log* text file, one line per plot.
%	Toggle between the IFDR IF-Input sources.

7.6.3 TTY information lines in **Pr**

An example of the TTY information lines:

```
Zoom:x1, Navg:4, Start:0.00 usec (0.00 km), Span:5 usec
Total:-63.3 dBm, Filtered:-77.6 dBm, MidSamp:-77.4 dBm
```

Table 37 **Pr** TTY information lines

Information line	Description
`Zoom`	The magnification (in amplitude) of the plotted samples. A zoom level of "x1" means that a full scale A/D waveform exactly fills the vertical height of the plot. Generally, the IF signal strength is not quite this high. Use larger zoom levels to see the weaker samples more clearly. You may zoom in powers of two up to x128.
`Navg`	The number of spectra and/or LOG powers that are averaged together prior to plotting. Larger amounts of averaging increase the ability to observe subtleties of the signals, but the display updates more slowly.
`Start`	The starting time of the IF sample window relative to range zero. The time is shown both in microseconds and in kilometers.
`Span`	Time span of the IF sample window in microseconds.
`IF`	The IF-Input sources (SMA edge connectors) that provide the data being plotted.
`Total`	The total RMS power that is being detected by the IF-Input A/D converters. This total is computed using all of the raw IF samples in the chosen interval, and is the sum of power at all frequencies other than 0 Hz (and its aliases).
`Filtered`	The RMS power that falls only within the passband of the FIR filter for the current pulse width. This is computed using all of the raw IF samples in the chosen interval.
`MidSamp`	The RMS power within the passband of the FIR filter using only the raw IF samples in the center of the chosen interval.

The computation of **`Total Power`** is performed using the same subset of central IF samples that are used to compute **`Filtered Power`**. This smaller subset of IF samples comes about because filtering the data requires a convolution with the current FIR filter, and this computation does not produce results all the way to the edges of the input data. This is the same reason that the LOG plots do not extend across the full screen.

Because of this definition, it is valid to intercompare the **`Total Power`** and **`Filtered Power`**. The two numbers match exactly as long as all of the incoming power falls within the passband of the FIR filter. The difference between the two powers can be used as a measure of the filter loss for a given pulse shape, that is, the portion of signal that is lost outside of the filter's passband.

The **`Total`**, **`Filtered`**, and **`MidSamp`** values represent true RMS power (that is, variance), and not merely a sum-of-squares. Any DC offset present in the A/D converter does not affect these power levels.

7.7 Plot test pattern (P+)

You can use the **P+** command to generate a simple test pattern and verify that the display software works properly.

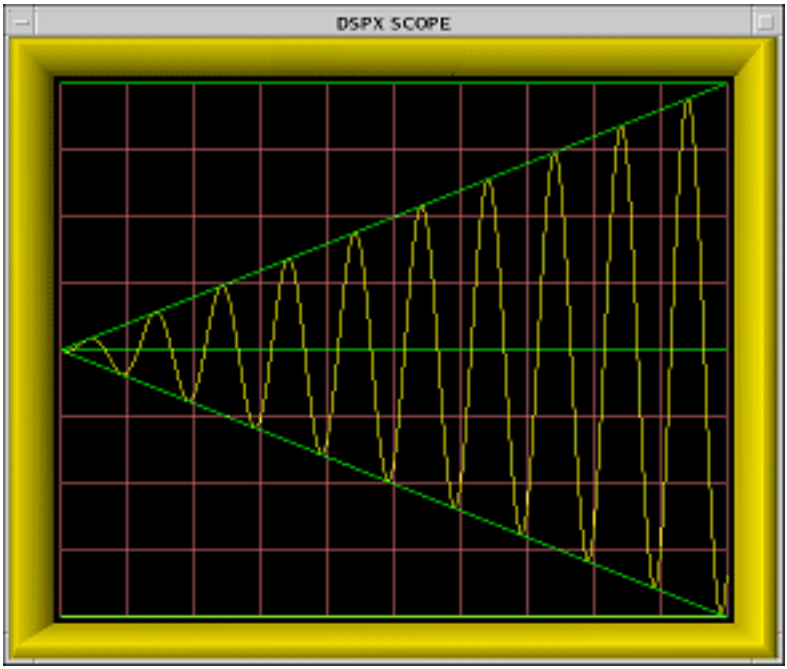


Figure 24 Test pattern display

1. In the TTY monitor, type: **P+**

   This prints the message `Plotting Test Pattern...` on the TTY and then produces the plot shown in the following figure. The display is an overlay of the following strokes:
   - Bottom line
   - Middle line
   - Top line
   - Line sloping up
   - Line sloping down
   - Sine wave pattern that changes phase with each plot so that it appears to radiate from the left side of the display.
2. When you are satisfied that the plot is drawn correctly, type **Q** or press ESC to return to the TTY monitor.

7.8 Configuring RVP10 digital front-end

This section describes configuring RVP10 digital front-end using the **dspx** tool.

Chapter General procedure for configuring the digital front-end (Magnetron) (page 146) introduces a generalized procedure for a magnetron system.

If your radar system has a coherent transmitter, such as SSPA or Klystron transmitter, or you have a more complex use case, you will need to modify the generalized procedure as described in the following chapters:

- Chapter Configuration for coherent systems with short CW (page 147) describes the configuration of simpler, short continuous wave (CW) pulses for coherent radar systems.
- Configuring longer compressed pulses (page 149) describes the configuration for compressed pulses.
- Configuring a sequence of pulses (hybrid pulsing) (page 150) describes the sequencing of multiple pulses (also known as hybrid pulsing).

The first step in any of these processes is to identify the use case of the transmit sequence, such as obtaining high resolution data near the radar, for example, for windshear identification, or creating a design to use in longer range surveillance scans where sensitivity becomes more important than range resolution. Longer pulses will increase the sensitivity but lower the range resolution. Thus, there is a trade-off which needs to be considered in the design targeted to the appropriate use case.

7.8.1 General procedure for configuring the digital front-end (Magnetron)

The following procedure is for first-time setups of a magnetron radar system.

Use the **dspx** tool to configure the RVP10 digital front-end.

You can do this at any time.

Sometimes it is useful to run plot commands with samples from the IF-Input of the IFDR, rather than from the Burst-Input.
Press **%** to toggle between the IF input SMA connectors on the IFDR.

1. Use **Burst pulse and AFC** setting to set the systems intermediate frequency.

   See **Mb**— Burst pulse and AFC (page 86).

2. Use **General trigger setups** to choose the PRF and pulse width. Also, choose the range resolution, as it may constrain the design of the digital filters later.

   See **Mt**— Triggers and timing (page 92).

3. Use **Triggers for Pulsewidth n** to set the relative timing of the RVP10 triggers used by the radar.

   Initially, you do not need to set the exact values of the trigger start times. Set their polarity and width, and their start times relative to each other.
   If you are using the bandpass filter, make an initial estimate of FIR filter length as 1.5 times the pulse width.
   If you are using a matched filter, set the pulse length value to capture the entire burst pulse width, but without extra time spacing. The ideal matched filter bandwidth and resulting filter coefficients will be computed from the length of the burst pulse value.
   See **Mw<n>**- Transmit waveforms (page 99).

4. Use **Plot Burst pulse timing** to slew the start times of all triggers so that the burst pulse is properly sampled.

   Refine the impulse response length if necessary so that all samples easily fit within the display window.
   See Burst pulse timing plot (Pb) (page 113).

5. Use **Plot burst spectra and AFC** to design the passband FIR filter, or, in the case of a matched filter, to select a windowing function.

   See Burst spectra and AFC plot (Ps) (page 120).

   a. Further refine the impulse response length and passband width to achieve a filter that matches the spectral width of the burst, and that has strong attenuation at DC.

      If the FIR length has changed, return to **Burst pulse timing** to verify that the burst is still sampled properly.

6. Continue using **Plot burst spectra and AFC** and **Burst pulse and AFC** to tune the AFC feedback loop.

   The settings that work for one pulse width should also work for the others.

7. Use **Plotting receiver waveforms** to verify that targets are being detected with good sensitivity.

   See Receiver waveforms plot (Pr) (page 139).

8. Repeat the procedure for each pulse width supported by the radar.

7.8.2 Configuration for coherent systems with short CW

If your radar system has a coherent transmitter, such as SSPA or Klystron transmitter, you will need to modify the generalized procedure described in chapter General procedure for configuring the digital front-end (Magnetron) (page 146). This chapter describes the configuration of simpler, short continuous wave (CW) pulses for coherent radar systems. The next chapters expand upon the concept to include compressed pulses and the sequencing of multiple pulses (or hybrid pulsing).

1. Select the *index of the transmit sequence* in which to store the design.

   Currently, there are four indexes for storing the transmit sequences. The transmit sequence indexes are defined by the **Mt<n>** menus. See **Mt<n>**— Transmit sequence #n (page 95).

2. Define the pulse length to the desired value in the **Mw<n>** menu.

   The index of the Mw section must be the same as defined in step 1.
   Select a pulse length that provides the desired range resolution and sensitivity properties. See **Mw<n>**- Transmit waveforms (page 99).
   Furthermore, the pulse length that gives the range resolution and sensitivity that is desired should be based upon the 3 dB pulse length. It is good practice to view the pulse in the Pb plot and adjust the configured pulse length so that the sampled pulse length is of the desired length.

3. Set the amount of amplitude tapering of transmit pulse.

   The style of amplitude tapering is a choice of windowing function and edge extent applied to the pulse. See **Mw<n>**- Transmit waveforms (page 99). Vaisala recommends using the Tukey window function, which is a rectangular window where the first and last portion of the window is replaced by a cosine function. The edge percentage determines the amount of window where the cosine function is defined, or effectively the sharpness of the transition from the cosine function to rectangular. The impact of amplitude tapering is lowering the effective bandwidth of the pulse, helping the transmit bandwidth be equal to the receiver bandwidth.
   Selection of 0.2 percent works well with SSPA transmitters used in Vaisala weather radar systems.

4. Set the amount of digital receiver tapering or windowing function on the receive side.

   The general rule of thumb is to set about double the amount of tapering used on the transmit side for the transmit and receiver bandwidths to match. See **Mw<n>**- Transmit waveforms (page 99). It is also possible to set the receive filter windowing function in the **Ps** plot. Generally, the goal is to select a window function where the main lobe of the transmit pulse matches the filter bandwidth, and that the filter provides sufficient sidelobe suppression.
   For more information, see Burst spectra and AFC plot (Ps) (page 120).

5. Adjust the trigger and pulse timing, and set the burst pulse sampling aperature.

   In the **Mt<n>** menu or in the **Pb** plot, set the trigger timing so that the middle of the pulse aligns with the middle of the plot. This aligns ‘range zero’ for proper range spacing and burst pulse sampling with respect to the trigger time. There may also be ‘clipping’ or shortening of theif the correct trigger time to center the burst pulse at range zero does not match the time needed for the signals to be upconverted and transmitted by the amplifier. Therefore, the **Mt<n>** submenu also includes the ability to apply a time offset per pulse. Adjusting this value allows the pulse to move forward/backward in time without impacting the range zero time. Setting the burst sample aperatures to contain only the full peak power of the transmit pulse ensures accurate measurement of pulse frequency.

6. Adjust amplifier `GATE` signal.

   Some amplifiers may use the trigger length as a gate which indicates the total time the transmitter is on. For transmitters used in Vaisala radar systems, this gate signal should be set approximately 0.5 usec longer than the entire transmit sequence duration.

   The gate length may be limited by the amplifier duty cycle given choice of PRF.

7.8.3 Configuring longer compressed pulses

After configuring a simple 1 usec CW pulse in Configuration for coherent systems with short CW (page 147), we can re-use some of those settings to configure longer compressed pulses. The trigger and start pulse timings should all be equal for each transmit sequence.

1. Copy the trigger timings and pulse start sequence as determined with the simple pulse configuration into all `Mt<n>` sections.

   The master trigger start time and pulse start time should be equal.

2. Define the pulse length and the type of frequency modulation to be used.

   See **`Mw<n>`**- Transmit waveforms (page 99). Generally, when configuring pulse compression, the pulse length should be 20 µsec in length or greater for proper frequency modulation. The RVP10 can support pulse lengths up to 200 µsec. However, this may further be limited by the duty cycle of the transmitter at PRFs commonly used in the weather radar application. For long-frequency modulated pulses, the full pulse length defined in `Mw<n>` should be about 5% longer than the desired 3bB pulse width due to amplitude tapering. For example, if an 80 µsec 3 dB pulse width is desired, the full pulse length of about 84 µsec should be entered into `Mw<n>`.

3. Set the tapering of transmit pulse from 0.05 to 0.10 in **dspx** to get the correct 3 dB pulse length. For all configured pulses, add the pulse width values into the **IRIS Setup Utility**.

   For more information, See chapter *Setup Utility* in *IRIS and RDA Utilities Guide (M212925EN)*.

4. Set the bandwidth of transmit pulse.

   See **`Mw<n>`**- Transmit waveforms (page 99). The choice of bandwidth relates back to the initial selection of range resolution desired for uses of this pulse. With pulse compression, in theory, the range resolution of the compressed pulse is the ratio of the pulse length and the Time-Bandwidth product of the pulse. For example, a 100 µsec pulse length using a 2 MHz bandwidth would have a Time-Bandwidth product equal to 200.The pulse length (100 µsec) divided by the Time-Bandwidth product (200) would give a range resolution equivalent to a 0.5 µsec pulse or 75 meters. However, after application of windowing functions to suppress range-time sidelobes to levels recommended, Vaisala recommends that the bandwidth value should be set approximately 1.8 times bigger than the ideal theory would suggest, or 3.6 MHz in this example.

5. Set the amount of Non-Linear Frequency Modulation to be used within the pulse.

   See **Mw<n>**- Transmit waveforms (page 99). Field experience shows that a value of 0.1 to 0.15 is required for the best suppression of range-time sidelobes.

   Setting this to a value of zero is equivalent to a Linear Frequency Modulation.

6. (*Optional:*) Review pulse in time domain using the **Pb** plot. Further adjustments of trigger or pulse timing could be made so that the pulse is both centered at 'range zero' and not being clipped.

   Setting all **Mt<n>** sections to the values defined in step 1 should avoid this need.

7. You can further optimize the Peak Side Lobe (PSL) and Integrated Side Lobe (ISL) by tuning the **Alpha** parameter of the Kaiser window function that is used with pulse compression. Tune this parameter up/down to minimize the PSL/ISL values which are computed for you in the Pa plot.

7.8.4 Configuring a sequence of pulses (hybrid pulsing)

This chapter describes combining individual pulses configured in Mw<n> into a sequence of pulses transmitted within a transmit sequence. This methodology allows the creation of time-frequency modulated transmit sequence, sometimes referred to as *Hybrid pulsing*. To get started, at least two pulses must be configured in the Mw<n> sections. It is common to use a short CW pulse and a longer NLFM pulse, but it is also possible to use a short NLFM pulse followed by a longer NLFM pulse.

1. Copy the trigger timings and pulse start sequence as determined with the simple pulse configuration in General procedure for configuring the digital front-end (Magnetron) (page 146) into all **Mt<n>** sections. The master trigger start time and pulse start time of the first in the sequence should be equal to values determined in previous processes.
2. Decide which **Mw** pulse (or index) should be the first pulse transmitted in the sequence. The equivalent **Mt** index value should be used to define the transmit sequence. For example, if **Mw<3>** is to be the first pulse then **Mt<3>** should be the index of the transmit sequence.
3. Add the next pulse into the sequence creating the 'hybrid' or time-frequency modulated series of pulses.

   See **Mt<n>**— Transmit sequence #n (page 95).

4. If the transmit trigger also acts as a gate, ensure the trigger length is longer than the combination of the two pulse lengths being combined. However, the trigger length should not be so long as to limit the choice of PRFs prior bumping into the transmitter duty cycle. Thus, the trigger length should be made to be just 1 or 2 usec longer than the total time span of the 'hybrid' pulse.

8. Processing algorithms

8.1 RVP10 algorithm overview

It is often convenient to combine two simultaneous samples of `I` and `Q` into a single complex number (called a phaser) of the form:

```
s = I + jQ
```

where `j` is the square root of -1.

Most of the algorithms presented here are defined in terms of the operations performed on the `s`'s, rather than the `I` and `Q`'s. The use of the complex terms leads to a more concise mathematical expression of the signal processing techniques being used.

During operation, the complex arithmetic is broken down into its real-valued component parts in order to be computed by RVP10. For example, the complex product:

```
s = W × Y
```

is computed as:

```
Real{s} = Real{W} Real{Y} – Imag{W} Imag{Y}
Imag{s} = Real{W} Imag{Y} + Imag{W}Real{Y}
```

where `Real{}` and `Imag{}` represent the real and imaginary parts of their complex-valued argument. Note that all of the expanded computations are themselves real-valued.

In addition to the usual operations of addition, subtraction, division, and multiplication of complex numbers, we use three additional operators known as **modulus**, **argument**, and **complex conjugate** (`||`, `Arg` and `*`). They are defined as follows.

Given a number `s` in the complex plane, the magnitude (or modulus) of `s` is equal to the length of the vector joining the origin with `s`, that is by Pythagoras:

$$|s| = \sqrt{\mathrm{Re}al\{s\}^2 + Imag\{s\}^2}$$

The signed (CCW positive) angle made between the positive real axis and the above vector is:

$$\measuredangle = Arg\{s\} = \arctan\left[\frac{Imag\{s\}}{\mathrm{Re}\,al\{s\}}\right]$$

where this angle lies between - π and + π and the signs of `Real{}` and `Imag{}` determine the proper quadrant. Note that this angle is real, and is uniquely defined as long as `|s|` is non-zero. When `|s| = 0`, `Arg{s}` is undefined.

Finally, the complex conjugate of `s` is the value obtained by negating the imaginary part of the number, that is:

$$s^* = \mathrm{Re}\,al\{s\} - jImage\{s\}$$

`Arg{s*} = –Arg{s}`.

8.1.1 Measured quantities

The following table summarizes the quantities that are measured and computed by RVP10.

Subscripts are sometimes used to denote successive samples in time from a given range bin. For example, s_n denotes the `I` and `Q` time series from the n^{th} pulse from a given range bin. In cases where it is obvious, the subscripts denoting the pulse (time) are dropped. The descriptions of all the data processing algorithms are phrased in terms of the operations performed on data from a single range bin, identical processing then being applied to all of the selected ranges. There is no need to include a range subscript in this data notation.

Table 38 Algebraic quantities within the RVP10 processor

Quantity	Description	Type
`p`	Analog-to-digital convertor receiver data sample	`Real`
`b`	Analog-to-digital convertor burst-pulse data sample	`Real`
`I, Q`	Instantaneous quadrature receiver components	`Real`
`s`	Instantaneous time series phaser value	`Complex`
`s'`	Time series after clutter filter	`Complex`
T_0	Zeroth lag autocorrelation of `s`values	`Real`
R_0	Zeroth lag autocorrelation of `s'` values	`Real`
R_1	First lag autocorrelation of `s'` values	`Complex`
R_2	Second lag autocorrelation of `s'` values	`Complex`
`SQI`	Signal Quality Index	`Real`
`V`	Mean velocity	`Real`
`W`	Spectrum width	`Real`
`CCOR`	Clutter correction	`Real`

Quantity	Description	Type
LOG	(Signal+Noise)/Noise ratio for thresholding	Real
SIG	Signal power of weather	Real
C	Clutter power	Real
N	Noise power	Real
Z	Corrected reflectivity factor	Real
T	Uncorrected reflectivity factor	Real
PMI	Polarimetric Meteorological Index	Real

8.1.2 IF signal conversion process

The following figure shows the process RVP10 follows to convert the IF signal into corrected reflectivity, velocity, and width, including:

- IF signal processing
- I/Q processing and clutter filtering
- Range averaging and clutter microsuppression
- Moment calculations (reflectivity, velocity, spectrum width)
- Thresholding for data quality and speckle filtering
- Reflectivity calibration
- Algorithms for ambiguity resolution (dual PRF, dual PRT, random phase)
- Calibration and testing

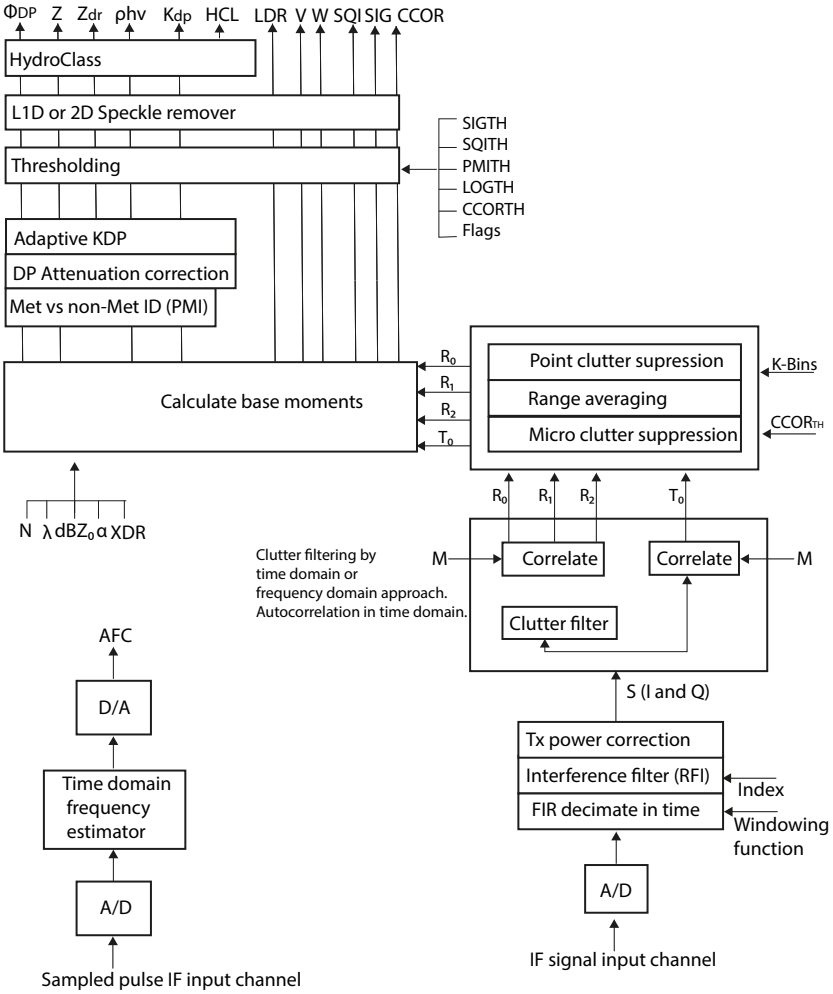


Figure 25 RVP10 processing flow diagram

8.2 IF signal processing

The starting point for RVP10 computations are the instantaneous ADC IF-receiver samples $\mathbf{p_n}$ and the instantaneous ADC burst-pulse or **COHO** reference samples $\mathbf{b_n}$.

This data is available at a very high sampling rate (up to 240 MHz).

8.2.1 FIR (matched) filter

RVP10 includes two methodologies to design a digitally matched filter: the continuous wave (CW) bandpass filter, and the idealized matched filter.

The continuous wave (CW) bandpass filter is the legacy methodology used within RVP900 and older RVP versions. In this method, user defines the filter length (number of taps), center frequency, and bandwidth.

The idealized matched filter is a new method. It is calculated by taking the conjugated transmit waveform and applying a windowing to the $\mathbf{p_n}$ samples.

For information on designing the digital matched filter using the graphical user interface, see Burst spectra and AFC plot (Ps) (page 120).

The legacy bandpass design procedure computes two sets of filter coefficients, f_n^i and f_n^q, such that the instantaneous quadrature samples at a given bin are:

$$I = \sum_{n=0}^{N-1} f_n^i \times p_n, \; Q = \sum_{n=0}^{N-1} f_n^q \times p_n$$

where **N** is the length of the filter. The input samples $\mathbf{p_n}$ are centered on the range bin to which the (**I**, **Q**) pair is assigned. Note that some $\mathbf{p_n}$ may overlap among adjacent bins. That is, the filter length may be greater than the bin spacing. The overlap introduces a slight correlation between successive bins, but the longer length allows a better filter to be designed. In the situation where the filter length design is smaller than the bin spacing, remaining samples $\mathbf{p_n}$ are skipped until the start of the next range bin. In the legacy bandpass filter, Vaisala generally recommends the filter length be about 1.5 x the pulse length.

The sums above for **I** and **Q** are computed on RVP10 using a flexible FGPA that can perform billions of sums of products per second.

Reference phase

The reference phase for each transmitted pulse is computed using the same two FIR sums, except that $\mathbf{b_n}$ is substituted for $\mathbf{p_n}$ and choices of window function are either Rectangular, Hamming, or Blackman.

For magnetron systems, the **N** $\mathbf{b_n}$ samples are centered on the transmitted burst.

For Klystron systems, the **N** $\mathbf{b_n}$ samples may be obtained from the burst pulse (recommended) or from the CW STALO. If the Klystron is phase modulated by an external phase shifter (instead of the IFDR digital transmitter board), the samples must be from the modulated STALO.

Coefficients

The f_n^i coefficients are computed as:

$$f_n^i = l_n \times \sin\left[\frac{\pi}{4} + 2\pi\frac{f_{IF}}{f_{SAMP}}\left(n - \frac{N-1}{2}\right)\right],\ n = 0 \ldots N-1$$

where f_{IF} is the radar intermediate frequency, f_{SAMP} is the IFDR sampling frequency, and l_n represents the coefficients of an N-point symmetric low-pass FIR filter that is matched to the bandwidth of the transmitted pulse. The multiplication of the l_n terms by the **sin()** terms effectively converts the low-pass filter to a band-pass filter centered at the radar IF.

The formula for the f_n^q coefficients is identical except that **sin()** is replaced with **cos()**.

The phase of the sinusoid terms, and the symmetry of the $\mathbf{l_n}$ terms, has been chosen to have a valuable overall symmetry property when **n** is replaced with **(N-1)-n**, that is, the sequence is reversed:

$$f^i_{(N-1)-n} = l_{(N-1)-n} \times \sin\left[\frac{\pi}{4} + 2\pi\frac{f_{IF}}{f_{SAMP}}\left(((N-1)-n) - \frac{N-1}{2}\right)\right]$$

$$f^i_{(N-1)-n} = l_{(N-1)-n} \times \cos\left[\frac{\pi}{4} + 2\pi\frac{f_{IF}}{f_{SAMP}}\left(n - \frac{N-1}{2}\right)\right]$$

$$f^i_{(N-1)-n} = f_n^q$$

The coefficients needed to compute **I** are the reverse of the **i** coefficients needed to compute **Q**. That is, if you know $f^i{}_n$, then you also know $f^q{}_n$.

When selecting to use the idealized matching filter, the f^icoefficients are computed similarly, except that the symmetric low-pass FIR filter $\mathbf{l_n}$ is replaced by a scaling function so the gain/loss at the center frequency is 0 dB, and *W[n]* and windowing function.

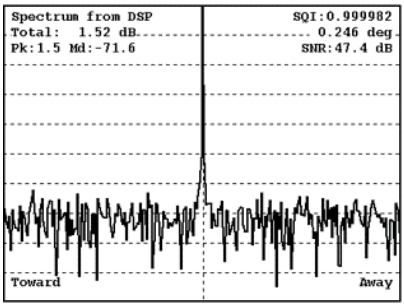
$$f_n^i = scale \times W\left[n\right] \times \sin\left[\frac{\pi}{4} + 2\pi\frac{f_{IF}}{f_{SAMP}}\left(n - \frac{N-1}{2}\right)\right],\ n = 0 \ldots N-1$$

Windowing functions **W[n]** are a signal shaping technique by increasingly reducing the amplitudes of samples from the center to the edge of the time domain. They may be used to increase signal coherency, which improves ground clutter suppression, or to increase the differentiation of 1st trip from multi-trip echo or from other types of interference. However, applying a windowing function will also generally decrease sensitivity and increase the variability seen in the final data types from the signal processor. For more information on windowing functions, see Frequency Domain Processing- Doppler power spectrum (page 171).

The types of windowing functions supported within the matched filter are Rectangular (no window), Triangular, Hann, Hamming, Blackman, Blackman Exact, Tukey, Kaiser, and Flat top. For any given noise bandwidth, the Rectangular window function provides the lowest root mean square error of the data estimates, and the Triangular window provides the best signal coherency.

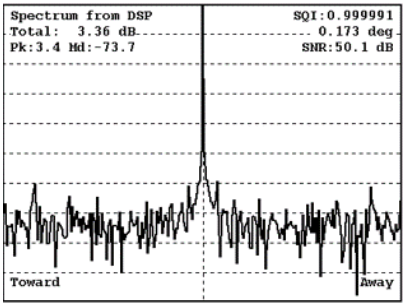
The following chart shows windowing function properties matched to a 2.0 µs burst pulse. The Doppler spectrum is that of a nearby point clutter target after the application of each window.

Rectangular



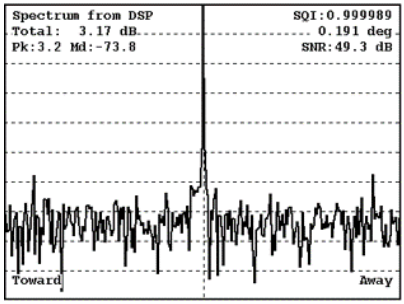
Filter loss	1.08 dB
BW	0.368 MHz
NEBW	0.416 MHz
Noise sample	-84.54 (H) -83.49 (V)
MDS (Noise - Filter loss)	-83.46
Clutter rejection	49 dB
brstEg	0.145 nJ
fltBrstEg	0.114 nJ
fltMtch	0.13 dB

Triangular



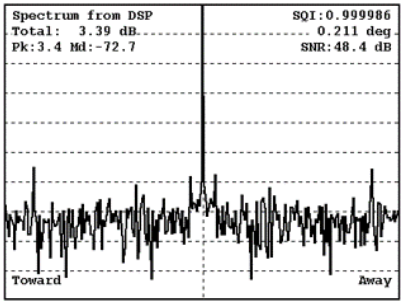
Filter loss	0.83 dB
BW	0.530 MHz
NEBW	0.555 MHz
Noise sample	-83.68 (H) -83.29 (V)
MDS (Noise - Filter loss)	-82.85
Clutter rejection	51 dB
brstEg	0.145 nJ
fltBrstEg	0.118 nJ
fltMtch	0.11 dB

Hann



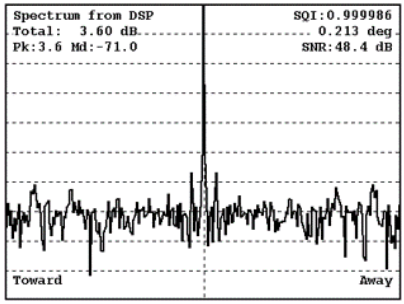
Filter loss	0.76 dB
BW	0.598 MHz
NEBW	0.624 MHz
Noise sample	-83.18 (H) -82.80 (V)
MDS (Noise - Filter loss)	-82.42
Clutter rejection	50 dB
brstEg	0.145 nJ
fltBrstEg	0.122 nJ
fltMtch	0.08 dB

Hamming



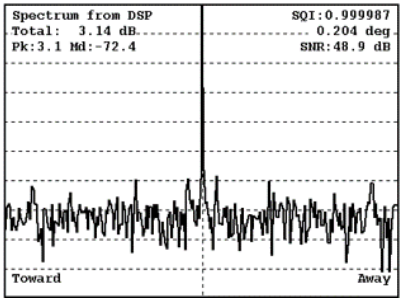
Filter loss	0.79 dB
BW	0.541 MHz
NEBW	0.567 MHz
Noise sample	-83.55 (H) -83.16 (V)
MDS (Noise - Filter loss)	-82.76
Clutter rejection	50.5 dB
brstEg	0.145 nJ
fltBrstEg	0.121 nJ
fltMtch	0.08 dB

Blackman



Filter loss	0.67 dB
BW	0.603 MHz

Blackman Exact

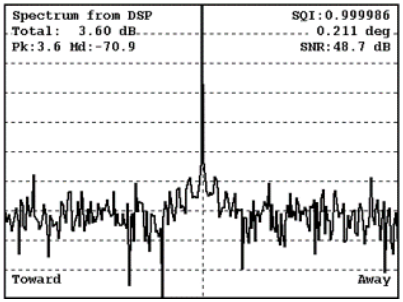


Filter loss	-0.85 dB
BW	-0.535 MHz

Blackman	
NEBW	0.718 MHz
Noise sample	-82.62 (H) -82.13 (V)
MDS (Noise - Filter loss)	-81.95
Clutter rejection	49 dB
brstEg	0.145 nJ
fltBrstEg	0.125 nJ
fltMtch	0.05 dB

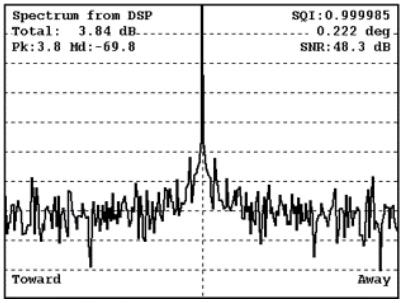
Blackman Exact	
NEBW	-0.577 MHz
Noise sample	-83.69 (H) -83.25 (V)
MDS (Noise - Filter loss)	-82.84
Clutter rejection	48 dB
brstEg	0.145 nJ
fltBrstEg	0.120 nJ
fltMtch	0.07 dB

Tukey



Filter loss	0.73 dB
BW	0.598 MHz
NEBW	0.624 MHz
Noise sample	-83.18 (H) -82.82 (V)
MDS (Noise - Filter loss)	-82.45
Clutter rejection	48.5 dB
brstEg	0.145 nJ
fltBrstEg	0.122 nJ
fltMtch	0.07 dB

Kaiser



Filter loss	0.59 dB
BW	0.770 MHz
NEBW	0.813 MHz
Noise sample	-82.10 (H) -81.68 (V)
MDS (Noise - Filter loss)	-81.58
Clutter rejection	49 dB
brstEg	0.145 nJ
fltBrstEg	0.125 nJ
fltMtch	0.07 dB

8.2.2 RVP10 receiver modes

Use the **Mc** menu to select the receiver configuration that defines IF processing for your site.

```
# Rx Mode Description
- ----------------------
0 Standard single channel
1 --Reserved--
2 --Reserved--
3 Standard dual channel
```

Table 39 Receiver mode configuration options

Mode	Description
Mode-0: Standard single channel	This is the most common mode used by single-polarization CW-pulsed radars whose front-end LNA has a dynamic range less than ≤95 dB. The (I,Q) data are computed from IF samples.
Mode-1: Reserved	Reserved for future development.
Mode-2: Reserved	Reserved for future development.
Mode-3: Dual-Rx from two IF inputs	Standard dual polarization mode. The H and V channels are fed to 2 separate IF inputs using the same intermediate frequency.

8.2.3 Automatic frequency control (AFC)

AFC is used for tuning the STALO to compensate for magnetron frequency drift.

RVP10 analyzes the burst pulse samples, and produces a running estimate of the power-weighted center frequency of the transmitted waveform. This frequency estimate is the basis of the RVP10 AFC feedback loop, whose purpose is to maintain a fixed intermediate frequency from the radar receiver.

The STALO is typically tuned 30 or 60 MHz away from the magnetron frequency. The maximum tuning range of the AFC feedback is approximately 5 MHz on each side of the center frequency. This is limited by the analog filters that are installed just before the signal and burst IF inputs on the IFDR.

The instantaneous frequency estimate is computed using four autocorrelation lags from each set of N b_n samples. This estimate is valid over the entire Nyquist interval (for example, 18 ... 36 MHz), but becomes noisy within 10% of each end. Since the span of the burst pulse samples is only approximately one microsecond, several hundred estimates must be averaged together to get an estimate that is accurate to several kHz. The AFC feedback loop typically has a time constant of several seconds or more.

Most burst pulse analysis routines, including the AFC feedback loop, are inhibited from running immediately after making a pulsewidth change. The center-of-mass calculations are held off according to the value of Settling time (to 1%) of burst frequency estimator, and the AFC loop is held off by the wait time before applying AFC. This prevents the introduction of transients into the burst analysis algorithms each time the pulse width changes.

8.2.4 Burst pulse tracking

RVP10 can track the power-weighted center-of-mass of the burst pulse and automatically shift the trigger timing so that the pulse remains in the center of the burst analysis window of the **Pb** plot.

This means that external sources of drift in the timing of the transmitted pulse (temperature, aging, and so on) are tracked and nulled out during normal operation so that fixed targets remain fixed in range and clean Tx phase measurements are always available on every pulse.

The burst pulse tracker feedback loop changes the trigger timing in response to the measured position of the burst. Timing changes are generally made only when RVP10 is not actively acquiring data, in the same way that AFC feedback is held off for similar quiet times.

However, if the center-of-mass has drifted more than 1/3 the width of the burst analysis window, then the timing adjustment is done immediately. Also, there is an approximately 5 ms interruption in the normal trigger sequence when any timing changes are made.

The burst pulse tracker and AFC feedback loop are fine-tuning servos that keep the burst pulse centered in time and frequency. These servos include a combined hunt mode that tracks down a missing burst pulse when we are uncertain of both its time and frequency. This coarse-tuning mode is valuable for initializing the 2 fine-tuning servos in radar systems that drift significantly with time and temperature.

When the radar transmitter is on but the burst pulse is missing, check the following possible reasons:

- Burst pulse is misplaced in time.
  The Tx pulse is outside of the window displayed in the **Pb** plotting command.
  The trigger timing must be changed to bring the center of the pulse back to the center of the window.
- Burst pulse is mistuned in frequency.
  The AFC feedback is incorrect and has caused the burst frequency to fall outside of the passband of the RVP10 anti-alias filters.
  The AFC (or DAFC) must be changed to restore the correct tuning.

The hunt mode performs a 2D search in time and frequency to locate the burst by searching across a ±20 µsec time window, and across the entire AFC span. If a valid Tx pulse (that is, meeting the minimum power requirement) is found anywhere in those intervals, then the Burst Pulse Tracker and AFC loops are initialized with the time and frequency values that were discovered. The fine servos then start running with a good burst signal starting from those initial points.

Depending on how the hunting process has been configured in the **Mb** menu, the procedure may take several seconds to complete. The RVP10SRV host computer interface remains functional during this time, but any acquired data is questionable. **GPARM** status bits in `word #55` indicate when the hunt procedure is running, and whether it has completed successfully.

8.2.5 Interference filter

The interference filter is an optional processing step that can be applied to the raw (`I`,`Q`) samples from the FIR filter chips.

The filter can remove strong but sporadic interfering signals that are occasionally received from nearby man-made sources. The technique relies on the statistics of such interference being noticeably different from that of weather.

For each range bin at which (**I**,**Q**) data are available, the interference filter algorithm uses the received power (in dB) from the 3 most recent pulses. This 3-pulse algorithm is only intended to remove interference that arrives on isolated pulses, and for which there are at least 2 clear pulses in between. Interference that tends to arrive in bursts are not rejected.

$$P_n - 2,\ P_n - 1,\ and\ P_n$$

where:

$$P_n = 10\log 10\left(I_n^2 + Q_n^2\right)$$

If the 3 pulse powers have the property that:

$$\left|P_n - 1 - P_n - 2\right| < C_1 \quad and \quad \left|P_n - 1 - P_n - 2\right| > C_2$$

(Alg. 1)[1)]

then ($\mathbf{I_n}$, $\mathbf{Q_n}$) is replaced by ($\mathbf{I_{n-1}}$, $\mathbf{Q_{n-1}}$). Here $\mathbf{C_1}$ and $\mathbf{C_2}$ are constants that can be tuned by the user to match the type of interference that is anticipated, and the error rates that can be tolerated. For some environments, it is possible that good results can be obtained with $\mathbf{C_1 = C_2}$; but RVP10 does not force that restriction.

Two variations on the fundamental algorithm are also defined. The **CFGINTF** command allows you to choose algorithms to use, and to tune the 2 threshold constants. You may also do this from the **Mp** setup menu. See Configure interference filter (**CFGINTF**) (page 435) and **Mp**—processing options (page 102).

$$\left|P_n - 1 - P_n - 2\right| < C_1 \quad and \quad \left|P_n - P_n - 1\right| > C_2$$

(Alg. 2)

$$\left|P_n - 1 - P_n - 2\right| < C_1 \quad and \quad \left|P_n - LinAvg\left(P_n - 1,\ P_n - 2\right)\right| > C_2$$

(Alg. 3)

Where **LinAvg()** denotes the decibel value of the linear average of the two decibel powers. The **Alg.2** and **Alg.3** algorithms also include the receiver noise level(s) as part of their decision criteria. When power levels are compared in the algorithms, any power that is less than the noise level is first set equal to that noise level. This makes the filters more robust and properly tunable, so that interference is more successfully rejected on top of blank receiver noise.

1) *The JMA internal specification for Interference Filter algorithm for use on Chitose airport Doppler weather radar is the basis for* **Alg.1**

Optimum values for C_1 and C_2 vary from site to site, but some guidance can be obtained using numerical simulations. The results shown below were obtained when the algorithms were applied to realistic weather time series having a spectrum width = 0.1 (Nyquist), SNR = +10 dB, and an intermittent additive interference signal that was 16 dB stronger than the weather. The interference arrived in isolated single pulses with a probability of 2%.

The algorithm performance is summarized in the first 3 columns of the following table, for which C_1 and C_2 have the common value shown. The fourth column also uses Algorithm #3, but with the value of C_1 raised by 2 dB. The **Missed** rate is defined as the percentage of interference points that manage to get through the filtering process without being removed. The **False** (false alarm) rate is the percentage of non- interference points that are incorrectly modified when they should have been left alone.

Table 40 Algorithm results for +16 dB interference

C1,C2	Alg.1 Missed/ False		Alg.2 Missed/ False		Alg.3 Missed/ False		Alg.3, C1+=2 dB Missed/False	
6.0dB	17.8%	10.91%	17.8%	4.06%	17.8%	3.48%	10.3%	4.15%
8.0dB	10.5%	6.57%	10.5%	2.42%	10.4%	1.71%	6.1%	1.92%
9.0dB	8.5%	5.09%	8.5%	1.81%	8.3%	1.16%	5.4%	1.28%
10.0dB	7.3%	4.01%	7.3%	1.42%	7.5%	0.79%	5.4%	0.85%
11.0dB	8.9%	3.14%	8.9%	1.06%	8.3%	0.51%	6.5%	0.54%
12.0dB	11.6%	2.53%	11.6%	0.85%	11.3%	0.33%	9.9%	0.35%
13.0dB	17.0%	2.07%	17.0%	0.67%	16.3%	0.22%	15.3%	0.23%
14.0dB	23.5%	1.70%	23.5%	0.54%	22.4%	0.14%	21.6%	0.15%
16.0dB	39.2%	1.21%	39.2%	0.35%	39.6%	0.06%	38.9%	0.06%
20.0dB	67.3%	0.65%	67.3%	0.14%	72.5%	0.01%	72.4%	0.01%

A false alarm in actual precipitation echo affects the values of moments calculated at the gate of the false alarm. For example, the measured power and estimates of reflectivity, subsequently, become slightly lower. The following table maps the rates of false alarms to mean relative changes of reflectivity (in dB).

Table 41 Impact of false alarms on reflectivity estimates

Alg.1 False Bias (dB)		Alg.2 False Bias (dB)		Alg.3 False Bias (dB)		Alg.3, +2 dB False Bias (dB)	
10.91%	..	4.06%	..	3.48%	..	4.15%	..
6.57%	..	2.42%	..	1.71%		1.92%	..
5.09%	..	1.81%	..	1.16%	..	1.28%	..
4.01%	..	1.42%	..	0.79%	..	0.85%	..

Alg.1 False Bias (dB)		Alg.2 False Bias (dB)		Alg.3 False Bias (dB)		Alg.3, +2 dB False Bias (dB)	
3.14%	..	1.06%	..	0.51%	..	0.54%	..
2.53%	..	0.85%	..	0.33%	..	0.35%	..
2.07%	..	0.67%	..	0.22%	..	0.23%	..
1.70%	..	0.54%	..	0.14%	..	0.15%	..
1.21%	..	0.35%	..	0.06%	..	0.06%	..
0.65%	..	0.14%	..	0.01%	..	0.01%	..

It is important to minimize both types of errors. If too much interference is missed, then the filter does not do an adequate job of cleaning up the received signal. If the false alarm rate is too high, then background damage is done at all times and the overall signal quality (especially sub-clutter visibility) may be compromised. Try to keep the false alarm rate fairly low, perhaps below 1%; and let the missed percentage fall where it may.

The following table shows the results obtained if the interference dominates by 26 dB. To summarize the numerical results in the table:

- The **Missed** rates of **Alg.1** and **Alg.2** are identical, but the **False** rate of **Alg.1** is much higher. **Alg.1** does not perform as well for additive interference, but it is included in the suite for historical reasons.
- The **Missed** error rate for **Alg.3** is nearly identical to that of **Alg.2**, but **Alg.3** has a significantly lower false alarm rate. This is because of the somewhat improved statistics that result when the linear mean of P_{n-2} and P_{n-1} is used in the second comparison, rather than just P_{n-1} alone. We recommend that **Alg.3** generally be chosen in preference to the other two.
- **Alg.3** can be further tuned by allowing the two constants to differ. For example, by raising C_1 slightly above C_2 (fourth column), we can trade off a decrease in the **Missed** rate for an increase in the **False** rate. Lowering C_1 would have the opposite effect.

Optimum tuning depends on the type of interference you are trying to remove. In the previous example, where the interfering signal is only 16 dB stronger than the weather, there was a close trade-off between the **Missed** and **False** error rates.

Table 42 Algorithm results for +26 dB interference

C1,C2	Alg.1 Missed/ False		Alg.2 Missed/ False		Alg.3 Missed/ False		Alg.3, C2+=5 dB Missed/False	
6.0dB	17.8%	10.75%	17.8%	3.95%	17.8%	3.44%	17.8%	0.34%
8.0dB	9.9%	6.48%	9.9%	2.31%	9.9%	1.68%	9.9%	0.15%
9.0dB	7.4%	4.99%	7.4%	1.75%	7.4%	1.14%	7.4%	0.10%
10.0dB	5.9%	3.91%	5.9%	1.36%	5.9%	0.76%	5.9%	0.06%
11.0dB	4.8%	3.06%	4.8%	1.06%	4.8%	0.50%	4.8%	0.04%

C1,C2	Alg.1 Missed/ False		Alg.2 Missed/ False		Alg.3 Missed/ False		Alg.3, C2+=5 dB Missed/False	
12.0dB	3.2%	2.37%	3.2%	0.83%	3.2%	0.33%	3.2%	0.03%
13.0dB	2.6%	1.83%	2.6%	0.62%	2.6%	0.20%	2.8%	0.01%
14.0dB	1.9%	1.45%	1.9%	0.50%	1.9%	0.12%	2.6%	0.01%
16.0dB	1.3%	0.90%	1.3%	0.30%	1.3%	0.05%	5.8%	0.00%
20.0dB	3.1%	0.39%	3.1%	0.12%	2.0%	0.01%	31.5%	0.00%

Note that we can re-tune the constants and operate with C_1 = 13dB and C_2= 18dB (fourth column); which yields a low 2.8% **Missed** rate, and an extremely low 0.01% false alarm rate. Since the false alarm rate is (approximately) independent of the interference power, these filter settings would leave "clean" weather virtually untouched. That is, we would have a safe filter that only removes fairly strong interference. You could leave such a filter running at all times without too much worry about side effects.

8.2.6 Large-signal linearization

When an IF signal saturates, there is still considerable information in the signal since only the peaks are clipped.

The proprietary large signal linearization algorithm used in RVP10 provides an extra 3 ... 4 dB of dynamic range by accounting for the effects of saturation. This is possible because an overdriven IF waveform spends some of its time in the valid range of the converter, and it is possible to deduce information about the signal.

The following figure shows signal generator test measurements with normal A/D saturation (lower line), and with the extrapolation algorithms turned on (upper line). The high-end linear range begins to roll off at approximately +10 dBm, instead of +5 dBm, and has been extended by 5 dB.

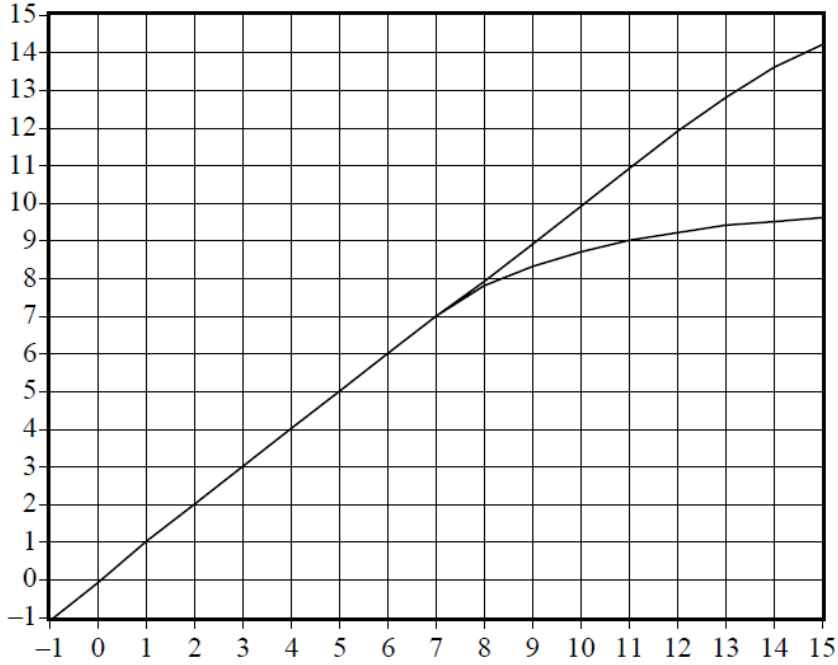


Figure 26 Linearization of Saturated Signals Above +8 dBm

8.2.7 Amplitude correction for Tx power fluctuations

RVP10 can perform pulse-to-pulse amplitude correction of the digital (**I**,**Q**) data stream based on the amplitude of the Burst/STALO input. The technique computes a (real valued) correction factor at each pulse by dividing the mean amplitude of the burst by the instantaneous amplitude of the burst. The (**I**,**Q**) data for that pulse are then multiplied by this scale factor to obtain corrected time series. The amplitude correction is applied after the Linearized Saturation Headroom correction.

Instantaneous amplitude correction is a unique feature of the RVP10 digital receiver. Bench tests with a signal generator reveal that an amplitude modulated waveform having 2.0 dB of pulse-to-pulse variation is reduced to less than 0.02 dB RMS of (**I**,**Q**) variation after applying the amplitude correction.

The mean burst amplitude is computed by an exponential average whose (1/e) time constant is selected as a number of pulses. See **Mp**— processing options (page 102).

A short time constant settles faster, but is not as thorough in removing amplitude variations (since the mean itself varies). Longer time constants are better in this sense, but require a few seconds before valid data is available when the transmitter is first turned on. The default value of 70 gives excellent results in almost all cases.

When RVP10 enters a new internal processing mode (time series, FFT, PPP, and so on), the burst power estimator is reinitialized from the level of the first pulse encountered, and an additional pipeline delay is introduced to allow the estimator to completely settle. Valid corrected data is produced even when RVP10 alternates rapidly between different data acquisition tasks, for example, in a multi-function **Ascope** display. The additional pipeline delay does not affect the high-speed performance when RVP10 runs continuously in a single mode.

For amplitude correction to be applied, the instantaneous Burst/STALO signal level must exceed the minimum valid burst power specified in the **Mb** setup section. If that level is not met (for example, if the transmitter is turned off), then no correction is performed. The amplitude correction feature is not in operation when receiver-only tests are performed.

The maximum applied correction is ±5 dB. If the burst power in a given pulse is more than 5 dB above the mean, or less than 5 dB below it, then the correction is clamped at those limits. The power variation of a typical transmitter is easily contained within this interval (it is typically less than 0.3 dB).

8.2.8 Wide dynamic range considerations

When a two-channel IFD is used as an extended dynamic range receiver, there are some important issues to consider with respect to setting up the RF/IF levels that drive the IFD.

Signal level separation

One important issue is the amount of signal level separation between the high gain and the low gain IFD inputs. There is an absolute minimum and absolute maximum channel separation that still allows the IFD to capture the full dynamic range of the receiver. If a signal level separation is made that is outside of these absolute limits, valuable receiver dynamic range will be lost.

- **The absolute minimum separation** of the channels is equal to the total dynamic range of the receiver minus the dynamic range of a single channel of the IFD. Generally, the total dynamic range of the receiver is set by the LNA.
  For example, if we are considering a 1 µs pulse (1MHz bandwidth), the dynamic range of the LNA may be about 105 dB, and the dynamic range of a single channel of the IFD is about 84 dB (-78 dBm ... +6 dBm). In this case, the minimum separation would be 21 dB. At minimum separation, the overlap of the low gain channel and the high gain channel are maximized, and that overlap is equal to the dynamic range of a signal channel of the IFD minus the separation. In this case, the overlap is (84 dB - 21 dB) = 63 dB.
- **The absolute maximum separation** of the channels is simply the dynamic range of a single channel of the IFD. In the above example, this would be 84 dB. At the maximum separation, the overlap of the low gain channel and the high gain channel is zero: the system starts using one as soon as the other has begun to saturate.

There can be a great difference between the absolute minimum and maximum signal level separations. Thus, additional criteria must be considered to choose the optimum value that is between these limits.

Choosing a proper separation value is a tradeoff of several factors. If the separation value is too low, the IFDs may end up operating very close to their noise floors. And if the separation is too high, then the overlap between the two channels is reduced, which makes it difficult for the IFD to make a smooth transition as it combines the data from both channels.

Too high a separation may also result in receiver components that are not practical to build.

Receiver requirements

Once a separation value has been chosen, we need to consider how to build the receiver to achieve this. The basic receiver will take the form of an LNA and a mixer followed by a splitter resulting in a low gain channel and a high gain channel. We know the gain difference in the two channels (the separation value), but we must find the actual gain to use in each channel.

If we consider the total system dynamic range as generally set by the LNA (105 dB in the above example), we can estimate the minimum detectable signal input to the LNA as well as the maximum usable linear level at the IFD. If the LNA has a noise figure of 1 dB and we are using a 1 µs pulse, the minimum detectable signal at the LNA input is -113 dBm, and thus the maximum signal is 105 dB above this, that is, -8 dBm. If we add to these number the gain of the LNA and the conversion loss of the mixer (and any other losses experienced through the power splitter for the low gain and high gain channels), we can use this information to determine the signal values of the components in these two channels.

For example, if the LNA has a gain of 17 dB, the mixer has a conversion loss of 7 dB, there is 1 dB miscellaneous losses and 3 dB loss in the power splitter, then the signal level at the output of the power splitter is (-113 + 18 - 7 - 1 - 3) = -106 dBm for the minimum signal, and -1 dBm for the maximum signal. In the low gain channel, we need to bring the -1 dBm up to the maximum input value of the IFD (+6 dBm). To do this, we need about 8 dB of amplification (7 dB plus one more dB to account for the anti–alias filter loss of the IFD). If we assume 25 dB of channel separation, on the high gain channel we require about +33 dB of amplification. Finally, this tells us that on the low gain channel, the minimum and maximum signals presented to the IFD are (-106 + 8) = -98 dBm and (-1 + 8) = 7 dBm. For the high gain channel, the signal levels are (-106 + 33) = - 73 dBm and (-1 + 33) = +32 dBm. Note that as +32 dBm is above the maximum input level tolerated by the IFD, the amplifier on the high gain channel must limit its output to less than +16 dBm. Thus, an amplifier with an output saturation value of between +10 dBm and +15 dBm should be used.

8.3 Time series signal processing

Radar time series data (also called linear "video" or `I` and `Q`) processing is done to obtain the meteorologically significant moment parameters: reflectivity, total power, velocity, width, signal quality index, clutter power correction, and optional polarization variables.

The time series are the starting point for all calculations performed in RVP10.

The time series synthesized by the FIR filter consist of an array of complex numbers:

$$s_m = [I_m + jQ_m] \; for \quad m = 1, 2, 3, \ldots, M$$

where `j` is - $1^{1/2}$.

Time series signal processing categories

There are two broad categories of time series signal processing:

- Time Domain Processing using the `I` and `Q` samples directly to calculate "autocorrelations" and then using the autocorrelations to compute the moments. This is used by many systems since the algorithms are very efficient requiring minimal storage and computational power. However, time domain algorithms are generally not adaptive or very flexible.
- Frequency Domain Processing using the `I` and `Q` samples to calculate a Doppler power spectrum and then applying algorithms, such as clutter filtering or second trip echo filtering/extraction, in the frequency domain. The Doppler spectrum is then inverted to obtain the autocorrelation functions and these are used to calculate the moments. The frequency domain is well suited to more complex adaptive algorithms, that is, where the processing algorithm is optimized for the data.

Supported time series processing modes

RVP10 supports the following major modes, that is, processing modes to process the time series:

Table 43 Supported time series processing modes

Mode	Description
DFT/FFT Mode	A frequency domain approach that is used for most operational processing applications. There are a variety of clutter filtering options, including the Gaussian Model Adaptive Processing (GMAP) algorithms.
Random Phase Mode (RPHASE)	A frequency domain approach similar to the DFT/FFT, except that filtering and extraction of both the first and second trip echoes is supported.
Pulse Pair Processing (PPP) mode	Within PPP, it is possible to first enter the frequency domain to perform spectrum based clutter filtering. The fixed and adaptive width clutter filters including the Gaussian Model Adaptive Processing (GMAP) are available with dual-polarization processing. Both of these clutter filters perform interpolation across the spectrum after ground clutter spectrum points are removed. Time domain 5-pole IIR filtering with 40 and 50 dB rejection are also available in PPP mode, but Vaisala recommends frequency-based filtering.
Batch Mode	A small batch of low PRF pulses is transmitted (for example, for 0.1° of scanning) followed by a large batch of higher PRF pulses (for example, for 0.9° of scanning) to determine which ranges are likely contaminated by second trip echo. This was developed to support a US WSR88D legacy requirement.

8.3.1 Time series and Doppler power spectrum example

The following figure shows the components of the Doppler spectrum, that is, white noise, weather signal, and ground clutter. Other target types such as sea clutter, birds, insects, aircraft, surface traffic, second trip echo, and so on may also be present.

- The top part shows the `I` and `Q` values for a simulated time series using the **Ascope** utility.

- The bottom part shows an example of a Doppler power spectrum for the time series shown in the upper part of the figure.

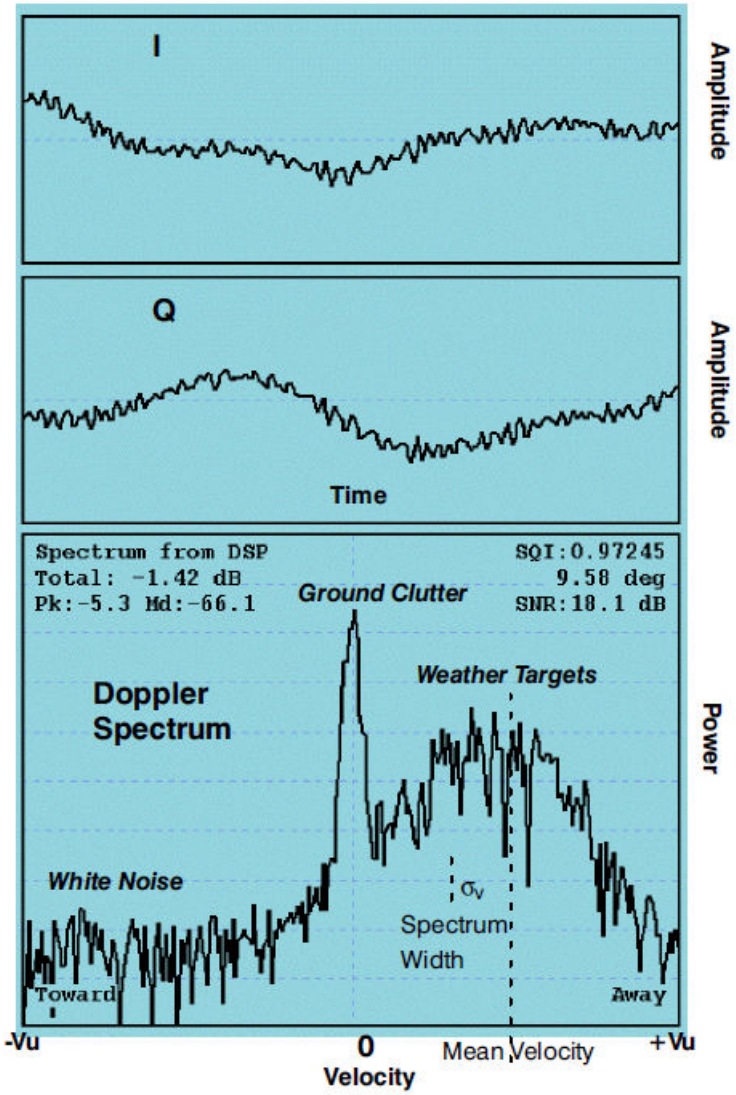


Figure 27 Time series and Doppler power spectrum example

The image shows a time series of I and Q and the corresponding Doppler power spectrum obtained from the **Ascope** utility using the built-in simulator. The Doppler spectrum displays the radial velocity on the X-axis over the unambiguous range (Nyquist interval) and the power (in dB) relative to the saturation of the Y-axis. For the sake of illustration, this example is based on 256 time series points (one point per pulse) which yields 256 spectrum components. This is more than is usually processed in actual operation.

The spectrum shows the three major components of the Doppler spectrum:

- white noise
- ground clutter at zero radial velocity
- a spectrum of the weather targets having a Gaussian shape characterized by the weather power, mean velocity, and width (standard deviation), that is, the spectrum moments.

8.3.2 Frequency Domain Processing- Doppler power spectrum

The Doppler power spectrum (also known as Doppler spectrum) is the easiest way to visualize the meteorological information content of the time series.

The Doppler power spectrum is obtained by taking the magnitude squared of the input time series, that is, for a continuous time series,

$$S(\omega) = |f\{s(t)\}|^2$$

Here `S` denotes the power spectrum as a function of frequency ω, and `f` denotes the Fourier transform of the continuous complex time series `s(t)`. The Doppler power spectrum is real-valued since it is the magnitude squared of the complex Fourier transform of `s(t)`.

In practice, a pulsed radar operates with discrete rather than continuous time series. That is, there is an `I` and `Q` value for each range bin for each pulse. In this case we use the discrete Fourier transform or DFT to calculate the discrete power spectrum.

When we have `2n` input time series samples (for example, 16, 32, 64, 128, ...), we use the fast Fourier transform algorithm (FFT), which is significantly faster than the full DFT.

The DFT has the form:

$$S_k = |DFT_k\{w_m s_m\}|^2 = \left| \sum_{m=0}^{M} w_m s_m m e^{-j\left(2\left(\frac{\pi}{M}\right)\right)mk} \right|^2$$

Typically a weighting function or "window" wm is applied to the input time series s_m to mitigate the effect of the DFT assumption of periodic time series. RVP10 supports different windows such as the Hamming, Blackman, Von Han, exact Blackman, and the rectangular window for which all spectral components are weighted equally.

The following figure shows the typical form of a spectrum window to illustrate how the edge points of the time series are de-emphasized and the center points are over emphasized. The dashed line corresponds to the rectangular window. The gain of the window is set to preserve the total power.

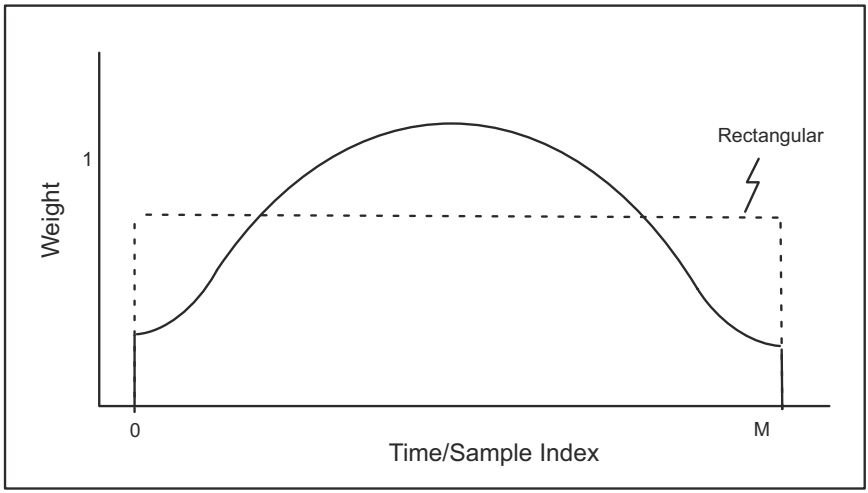


Figure 28 Typical form of time series window

Although the window gain can be adjusted to conserve the total power, there is an effective reduction in the number of samples which increases the variance (or uncertainty) of the moment estimates. For example, the variance of the total power is greater when computed from a spectrum with Blackman weighting compared to using a rectangular window. This is because there are effectively fewer samples due to the de-emphasis of the end points. This is a negative side to using a window.

The DFT of the window itself is known as its impulse response which shows all of the frequencies that are generated by the window itself. A generic example is shown in the following figure which illustrates that these side lobe frequencies can have substantial power. This is not a problem for weather signals alone, but if there is strong clutter mixed in, then the side lobe power from the clutter can obscure the weaker weather signals. The rectangular window has the worst sidelobes, but the narrowest window width. However, the rectangular window provides the lowest variance estimates of the moment parameters (in the absence of clutter).

More aggressive windows have lower side lobe power at the expense of a broader impulse response and an increased variance of the moment estimates.

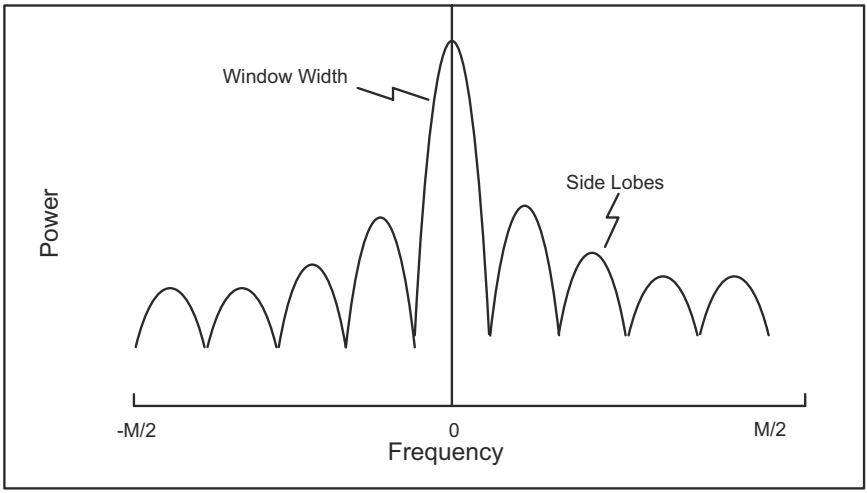


Figure 29 Impulse response of typical window

In summary of the DFT approach and spectrum windows:

- When the clutter is strong, an aggressive spectrum window is required to contain the clutter power so that the side lobes of the window do not mask the weather targets. The side lobe levels of some common windows are:
  - Rectangular 12 dB
  - Hamming 40 dB
  - Blackman 55 dB
- More aggressive windows typically have a wider impulse response. This effectively increases the spectrum width. Rectangular is narrow, Hamming intermediate, and Blackman the widest.
- Windows effectively reduce the number of samples resulting in higher variance moment estimates. Rectangular is the best case, Hamming is intermediate, and Blackman provides the highest variance moment estimates.

The best approach is to use the least aggressive window possible in order to contain the clutter power that is present. That is, an adaptive approach is the best.

8.3.3 Autocorrelation for moment estimations

The final spectrum moment calculation (for total power or SNR, mean velocity, and spectrum width) in all processing modes is based on autocorrelation moment estimation techniques.

Typically, the first three lags are calculated, denoted as R_0, R_1 and R_2. There are two ways to calculate these, that is, time domain or frequency domain calculation.

- In the PPP mode for dual-polarization, the autocorrelations are computed directly in the time domain.

- In DFT mode, they are computed by taking the inverse DFT of the Doppler power spectrum in the frequency domain.
  In the DFT case, only the first 3 terms must be calculated.

The time domain and frequency domain techniques are nearly identical, except for that the method of taking the inverse DFT of the power spectrum relies on the assumption that the time series is periodic. Another difference is that for time domain calculation, only a rectangular weighting is used.

In the following table, `M` is the number of pulses in the time average. Here, `s'` denotes the clutter-filtered time series, `s` denotes the original unfiltered time series and the * denotes a complex conjugate. `gr` and `gt` represent the transmitter and receiver gains, that is, their product represents the total system gain.

Table 44 Time domain calculation of autocorrelations and corresponding physical models

Parameter and definition	Physical model
$T_0 = \frac{1}{M}\sum_{n=1}^{M} s_n{}^* s_n$	$g^r g^t (S + C) + N$
$R_0 = \frac{1}{M}\sum_{n=1}^{M} s'_n{}^* s'_n$	$g^r g^t S + N$
$R_1 = \frac{1}{M-1}\sum_{n=1}^{M-1} s'_n{}^* s'_{n+1}$	$g^r g^t S + e^{j\pi V' - \frac{\pi^2 W^2}{2}}$
$R_2 = \frac{1}{M-2}\sum_{n=1}^{M-2} s'_n{}^* s'_{n+2}$	$g^r g^t S + e^{j\pi V' - 2\pi^2 W^2}$

Since the RVP10 is a linear receiver, there is a single gain number that relates the measured autocorrelation magnitude to the absolute received power. However, since many of the algorithms do not require absolute calibration of the power, the gain terms are ignored in the discussion of these. T_o for the unfiltered time series is proportional to the sum of the meteorological signal `S`, the clutter power `C` and the noise power N. R_0 is equal to the sum of the meteorological signal `S` and noise power `N` which is measured directly on the RVP10 by periodic noise sampling. T_o and R_0 are used for calculating the dBZ values - the equivalent radar reflectivity factor which is a calibrated measurement. The physical models for R_0, R_1 and R_2 correspond to a Gaussian weather signal and white noise. `W` is the spectrum width and `V'` the mean velocity, both for the normalized Nyquist interval on `[-1 to 1]`.

The autocorrelation lags above and the corresponding physical models have five unknowns: `N`, `S`, `C`, `V'`, `W`. Because the R_1 and R_2 lags are complex, this yields, effectively, 5 equations in 5 unknowns using the constraint provided by the argument of R_1. This closed system of equations can be solved for the unknowns which is the basis for calculating the moments from the autocorrelations.

8.3.4 Ray synchronization on angle boundaries

The exact number of samples (**M**) that is used for each time average is usually set when setting up the operating parameters of RVP10.

The exact value of **M** that is used for each time average is generally the sample size that is selected by the **SOPRM** command.

However, when the RVP10 is aligning its processed rays to AZ/EL angle boundaries, the actual number of pulses used may be limited by the number that fit within each ray's angular limits at the current antenna scan rate. So the antenna scan rate may limit the number of samples averaged. The number of samples will never be greater than the set sample size, even if the antenna is scanning at low angular speed that would allow more samples.

Example: RVP10 is operating at 1 KHz PRF, 20-deg/sec scan rate, 1° ray synchronization, and a sample size of 80. Then, if the **LSYNC** **Dyn** bit is set, rays consist of a full 80-pulses ending at each angle and extending back 30-pulses into the previous angle sector. However, when **Dyn** is clear, there are 50 pulses used for each ray (not 80) and those pulses exactly fill the scanning angle sector.

8.3.5 Clutter filtering approaches

The following table shows how each major mode implements clutter filtering.

Table 45 Clutter filtering in major modes

Mode	Clutter filtering
FFT Major Mode	Uses frequency domain clutter filters, including GMAP. The power spectra are restored by interpolating across the gap of removed spectral points.
PPP Mode	In this mode, RVP10 performs a DFT/FFT into the frequency domain just for clutter filtering, such as GMAP. In dual-polarization, the same spectral points that were removed and interpolated in the co-polar realm are treated the same in the cross-polar realm. After removal of ground clutter, the RVP10 inverts back to the time domain for moment computations.
Random Phase Mode	Uses frequency domain clutter filters, including GMAP. The power spectra are restored by interpolating across the gap of removed spectral points.
Batch Mode	Uses a simple DC removal for the small batch clutter filter. The high PRF large batch is then processed using frequency domain clutter filters, including GMAP. The power spectra are restored by interpolating across the gap of removed spectral points.

RVP10 enables time-domain IIR clutter filtering techniques. However, Vaisala does not recommend using these techniques, as the IIR filter has several drawbacks:

Instead of IIR clutter filtering Vaisala recommends using other filtering techniques. The frequency domain filters available in RVP10 are configured using the **mf** setup command (See **Mf**— Clutter filters (page 108)):

- Type 0: Fixed width filters with interpolation
- Type 1: Variable width single slope adaptive processing
- Type 2: Reserved for future development
- Type 3: GMAP

8.3.5.1 Fixed width clutter filters

This filter removes a specified number of spectrum components (5 in the following example) and then interpolates across the gap using the minimum of a specified number of "edge points" (2 in the example) to anchor the interpolation at each end of the gap.

This is a fairly simple legacy approach that uses interpolation to repair the damage caused by the removal of components.

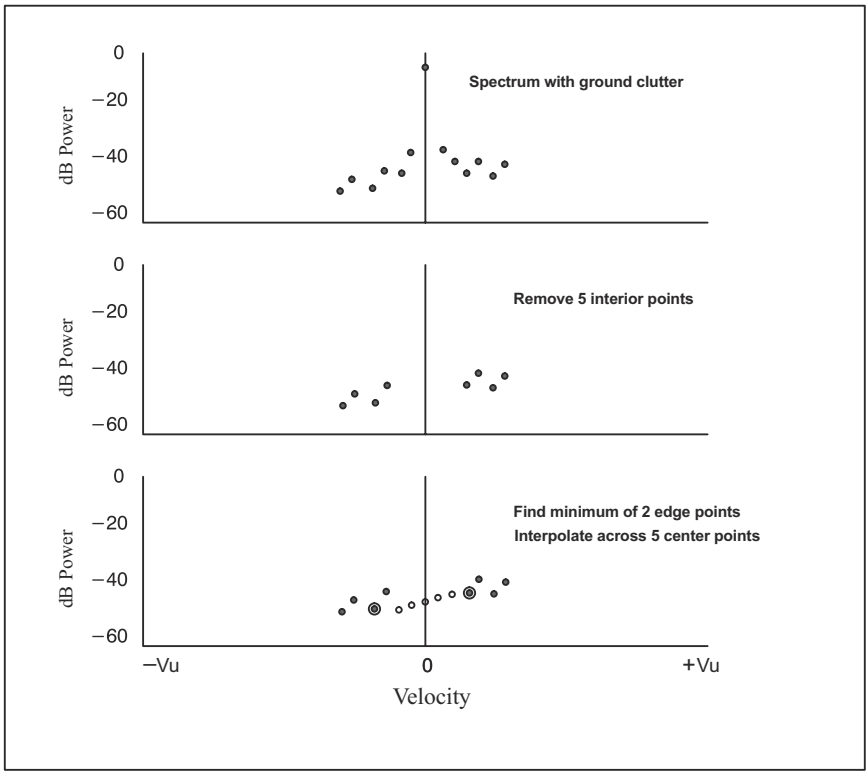


Figure 30 Fixed Width Clutter Filter Examples

This procedure attempts to preserve the noise level and/or overlapped weather targets. The result is that more accurate estimates of dBZ are obtained. In extreme cases when the weather spectrum is very narrow, there can still be some attenuation of weather if a broad filter is selected.

8.3.5.2 Variable Width Clutter Filter

This is similar to the fixed width filter, except that the algorithm attempts to extend the boundary of the clutter by determining which is the first component outside the clutter region to increase in power.

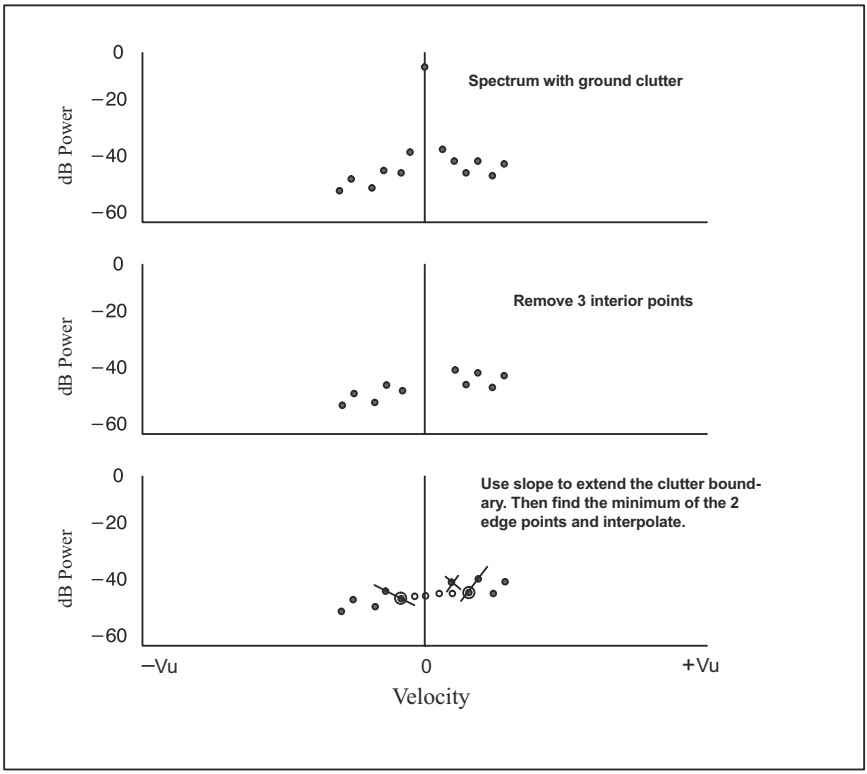


Figure 31 Variable Width Clutter Filter

In the figure, the minimum number of points to reject is set to 3. The filter starts at zero velocity and checks the slope to determine the point at which the power starts to increase. In the example, this results in the filter being extended by one point on the right. Note that there is a selectable maximum number of points that the filter "hunts". The use of the edge points for interpolation is identical to the fixed width case.

This filter allows users to specify a narrower nominal filter than the fixed width case and then when the clutter is strong, this width is extended by the algorithm (the "hunt"). The interpolation attempts to preserve any overlapped clutter and weather.

8.3.5.3 Gaussian model adaptive processing (GMAP) filtering

GMAP processing has the following advantages compared to fixed width frequency domain filters or time domain filtering such as the IIR approach:

- The width adapts in the frequency domain to adjust for the effects of PRF, number of samples and the absolute amplitude of the clutter power. This means that minimal operator intervention is required to set the filter.
- If there is no clutter present, GMAP does little or no filtering.
- GMAP repairs the damage to overlapped (near zero velocity) weather targets.
- The DFT window is determined automatically to be the least aggressive possible to remove the clutter. This reduces the variance of the moment estimates.

GMAP model assumptions

- The spectrum width of the weather signal is greater than that of the clutter. This is a fundamental assumption required for all Doppler clutter filters.
- The Doppler spectrum consists of ground clutter, a single weather target, and noise. Bi-modal weather targets, aircraft or birds mixed with weather would violate this assumption.
- The width of the clutter is approximately known. This is determined primarily by the scan speed and to a lesser extent by the climatology of the local clutter targets. The assumed width is used to determine how many interior clutter points are removed.
- The shape of the clutter is approximately Gaussian. This shape is used to calculate how many interior clutter points are removed.
- The shape of the weather is approximately Gaussian. This shape is used to reconstruct filtered points in overlapped weather.

GMAP algorithm steps

The following figure shows the steps used to implement the GMAP approach.

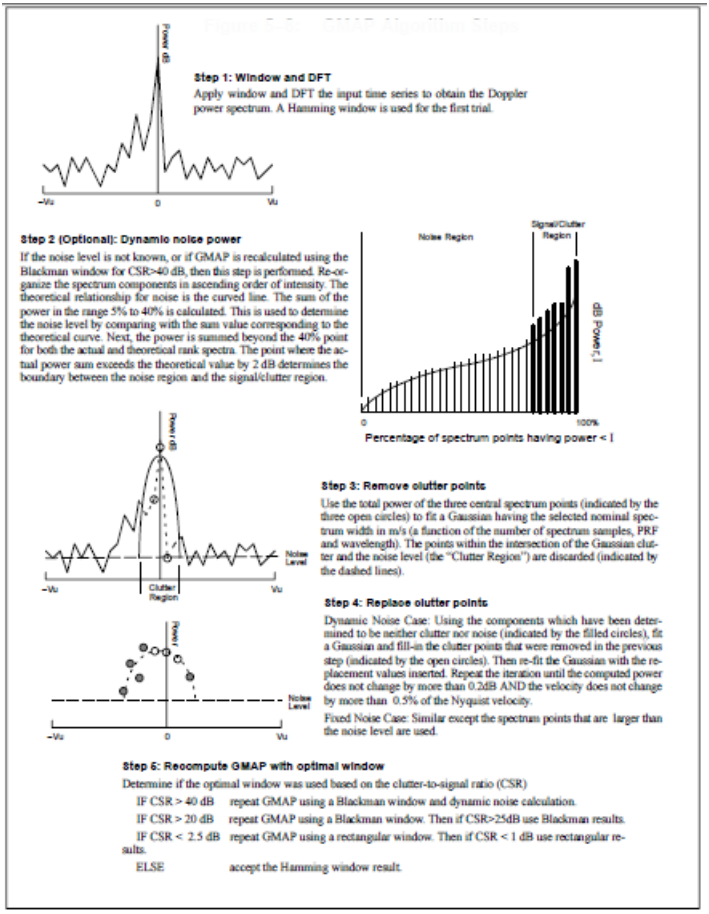


Figure 32 GMAP algorithm steps

Table 46 GMAP algorithm steps

GMAP Step	Description
Step 1: Window and DFT	First, a Hamming window weighting function is applied to the **IQ** values. Then, a DFT is performed. This provides a better spectrum resolution than an FFT, which requires that the number of **IQ** values be a power of 2. Note that if the requested number of samples is exactly a power of 2, then an FFT is used. As mentioned in Frequency Domain Processing- Doppler power spectrum (page 171), when there is very little or no clutter, use of a rectangular weighting function leads to the lowest-variance estimates of intensity, mean velocity and spectrum width. When there is a very large amount of clutter, then the aggressive Blackman window is required to reduce the "spill-over" of power from the clutter target into the sidelobes of the impulse response function. The Hamming window is used as the first guess. After the first pass GMAP analysis is complete, a decision is made to either accept the Hamming results, or recalculate for either rectangular or Blackman depending on the clutter-to-signal ratio (CSR) computed from the Hamming analysis. The recalculated results are then checked to determine whether to use these or the original Hamming result.

GMAP Step	Description
Step 2: Determine the noise power	In general, the spectrum noise power is known from periodic noise power measurements. Since the receiver is linear and requires no STC or AGC, the noise power is well-behaved at all ranges. The only time that the spectrum noise power differs from the measured noise power is for very strong clutter targets. In this case, the clutter contributes power to all frequencies, essentially increasing the spectrum noise level. This occurs for two reasons: in the presence of very strong clutter, even a small amount of phase noise causes the spectrum noise level to increase, and there is significant power that occurs in the window side-lobes. For a Hamming window, the window side lobes are down by 40 dB from the peak at zero velocity. 50 dB clutter targets have spectrum noise that is dominated by the window sidelobes in the Hamming case. The more aggressive Blackman window has approximately 55 dB window sidelobes at the expense of having a wider impulse response and larger negative effect on the variance of the estimates. When the noise power is not known, it is optionally computed using a dynamic approach similar to that of Hildebrand and Sekhon (1974). The Doppler spectrum components are first sorted in order of their power. As shown in the figure, the sorting places the weakest component on the left and the strongest component on the right. The vertical axis is the power of the component. The horizontal axis is the percentage of components that have power less than the y-axis power value. Plotted on a dB scale, Poisson distributed noise has a distinct shape, as shown by the curved line in the figure. This shape shows a strong singularity at the left associated with taking the log of numbers near zero, and a strong maximum at the right where there is always a finite probability that a few components have extremely large values. There are generally two regions: a noise region on the left (weaker power) and a signal/clutter region on the right (stronger power). The noise level and the transition between these two regions is determined by first summing the power in the range 5% ... 40%. This sum is used to determine the noise level by comparing with the sum value corresponding to the theoretical curve. Next, the power is summed beyond the 40% point for both the actual and theoretical rank spectra. The point where the actual power sum exceeds the theoretical value by 2 dB determines the boundary between the noise region and the signal/clutter region. Finally there are two outputs from this step: a spectrum noise level and a list of components that are either signal or clutter.

GMAP Step	Description
Step 3: Remove the clutter points	The inputs for this step are the Doppler power spectrum, the assumed clutter width in m/s and the noise level, either known from noise measurement or optionally calculated from the previous step. First the power in the three central spectrum components is summed (DC ±1 component) and compared to the power that would be in the three central components of a normalized Gaussian spectrum having the specified clutter width and discretized in the identical manner. This serves as a basis for normalizing the power in the Gaussian to the observed power. The Gaussian is extended down to the noise level and all spectral components that fall within the Gaussian curve are removed. The power in the components that are removed is the "clutter power". A subtle point is the use of the three central points to do the power normalization of the actual vs the idealized spectrum of clutter. This is more robust than using a single point since for some realizations of clutter targets viewed with a scanning antenna, the DC component is not necessarily the maximum. Averaging over the three central components is a more robust way to characterize the clutter power. The very substantial algorithmic work that has been done so far is to eliminate the proper number of central points. The operator only has to specify a nominal clutter width in m/s. This means that the operator does not need to consider the PRF, wavelength, or the number of spectrum points. GMAP accounts for these automatically. A key point is that in the event that the sum of the three central components is less than the corresponding noise power, then it is assumed that there is no clutter and all of the moments are then calculated using a rectangular window. If the power in the three central components is only slightly larger than the noise level, then the computed width for clutter removal is so narrow that only the central (DC) point shall be removed. This is very important since, if there is no clutter then we want to do nothing or at worst only remove the central component. Because of this behavior, there is no need to do a clutter bypass map, that is, turn off the clutter filter at specific ranges, azimuths and elevations for which the map declares that there is no clutter. Because of the day-to-day variations in the clutter and the presence of AP, the clutter map is often incorrect. Since GMAP determines the no-filter case automatically and then processes accordingly, a clutter map is not required.

GMAP Step	Description
Step 4: Replace clutter points	The assumption of a Gaussian weather spectrum now comes into play to replace the points that have been removed by the clutter filter. There are two cases depending on how the noise level is determined under Step 2, that is, the dynamic noise case and the fixed noise level case. **Dynamic noise level case:** From Step 2, we know which spectrum components are noise. From Step 3, we know which spectrum components are clutter. Presumably, everything that is left is weather signal. An inverse DFT using only these components is performed to obtain the autocorrelation at lags 0, 1. This is very computationally efficient since there are typically few remaining points and only the first two lags need be calculated. The pulse pair mean velocity and spectrum width are calculated using the Gaussian model. Note that since the noise has already been removed, there is no need to do a noise correction. The Gaussian model is then applied using the calculated moments to determine a substitution value for each of the spectrum components that were removed in Step 3. In the case of overlapped weather as shown in the GMAP example, the replacement power is typically too small. For this reason, the algorithm recomputes `R0` and `R1` using both the observed and the replacement points and computes new replacement points. This procedure is done iteratively until the power difference between two successive iterations is less than 0.2 dB and the velocity difference is less than 0.5% of the Nyquist interval. In summary of this step, the Gaussian weather model is used to repair the filter bias, that is, the damage that is caused by removing the clutter points. An IIR filtering approach makes no attempt to repair filter bias, rather the filter "digs a hole" into overlapped weather.
Step 5: Check for appropriate window and recalculate the moments, if necessary	The clutter power is known from the spectrum components that were removed in Step 3. Since the weather spectrum moments and the noise are also known from Step 4, the CSR can be calculated. The value of the CSR, is used to decide whether the Hamming window is the most appropriate. The end result is that very weak clutter is processed using a rectangular window, moderate clutter a Hamming window, while severe clutter requires a Blackman window. Note that if no clutter were removed in Step 3, then the spectrum is processed with a rectangular window. The benefit of adaptive windowing is that the least aggressive window is used for the calculation of the spectrum moments, resulting in the minimum variance of the moment estimates.

GMAP configuration

The **mf** command in the **dspx** TTY setups is used to configure GMAP filters. In the section for the spectrum filters select filter "Type 2" and specify the width of the ground clutter in m/s. This width is determined largely by your antenna rotation rate, so you should configure several widths to deal with the different rotation rates in your operational scenario. An example might be filters indexed 1-5 corresponding to widths from 0.1 ... 0.5.

A good practice is to make a scan on a clear day while using **Ascope** or other utility and observe the width of the clutter for your scan rates. You must turn off the clutter filtering to do this (select `filter 0` for the all pass filter).

Implementation example

GMAP has been extensively evaluated for use in the US WSR88D ORDA network up-grade (Ice et al, 2004). They conclude that GMAP meets the ORDA requirements. Their study was based on a built-in simulator that is provided as part of the RVP10 and the **Ascope** utility. The simulator allows users to construct Doppler spectra, process them and evaluate the results (Sirmans and Bumgarner, 1975). This is an essential tool for evaluating the system performance.

The following figure shows an example of the simulations for the very difficult case when the weather has zero velocity, that is, it is perfectly overlapped with clutter. The upper left graph shows the weather signal with -40 dB power without any clutter or GMAP filtering. The graph on the upper right shows the same spectrum with 0 dB of clutter power added for a clutter width of 0.012 (0.3 m/s at S band, 1000 Hz PRF). This is a CSR of 40 dB. The panel at the lower left shows the weather signal after GMAP filtering.

In each of the moment plots, there are several values that are displayed. The left-most number shows the value at the range cursor which is positioned as indicated by the vertical line. To the right, the `m` value is the mean and the `s` value the standard deviation as averaged over all range bins (1000 in this example). For velocity these are in normalized units expressed as a fraction of the Nyquist interval. For reflectivity the values are in dB.

Some key points are:

- The mean velocity is correctly recovered as expected (the `m` value in the plot), but the standard deviation is higher (0.06 vs 0.04 in normalized units).
- The `Cor dBZ` shows 40.2 dB of `C.Rej`. This is the difference between the `Tot dBZ` and the `Cor dBZ` values. The expected value is 40 dB in this case. This indicates that GMAP has recovered the weather signal in spite of the aggressive clutter filtering that is required.
- The standard deviation of the `Tot dBZ` is greater in the weather plus clutter (4.35 normalized units) as compared to the weather-only case. This is caused by the fluctuations in the clutter power in the Gaussian clutter model.
- The standard deviation of the `Cor dBZ` after GMAP filtering, while not as low as for the weather-only case, is lower than the weather plus clutter case. In other words, the GMAP processing removes some of the high variance in the dBZ estimates that is caused by clutter, but a better result is achieved when leaving it out.

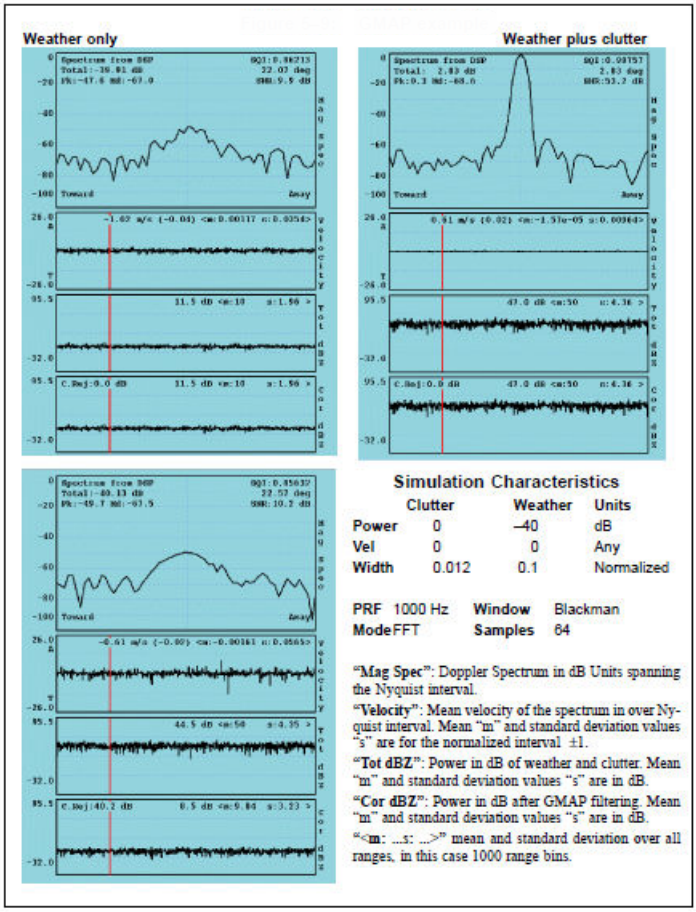


Figure 33 GMAP example

Clutter filtering considerations with dual-polarization

In any ground clutter filtering application, there is a need to retain coherency within the cross-polarization receiver channel. This is achieved by allowing any of the above-mentioned frequency domain ground clutter cancellation techniques to run on the co-polar data. We then know which spectrum points were removed due to ground clutter and remove the same points in the cross-polar channel. The same interpolation technique, and in the case of GMAP, the same windowing function, is then applied to the remaining spectrum points in the cross-polar data.

8.4 Autocorrelation R(n) Processing

8.4.1 Point Clutter Remover

Point Clutter is a target that has strong total power in one or two successive range bins but is bordered on either side in by bins of significantly lower power. These are usually non-meteorological targets such as airplanes, ships, or other moving objects.

The first step in autocorrelation processing is the optional removal of point clutter.

There are 2 adjustable parameters for the point clutter algorithm:

- TCM Point clutter threshold factor
- Offset Point clutter range offset in bins

The first pass called *Clutter detection* is to screen all range bins to see if their Doppler filtered power exceeds the normal level before and after in range. In other words, flag all bins if:

$$[(R_0(r) > TCM * R_0(r - Offset)) \quad and \quad (R_0(r) > TCM * R_0(r + Offset))]$$

These flag bits are output in the optional flag word. The second pass called *Clutter sensoring* involves linearly interpolating all the autocorrelation values in range over the interval of clutter bins including the offsets at either sides if the offset is larger than 1:

$$R_0(r) = \left(\frac{r_{end} - r}{r_{end} - r_{start}}\right) * R_0(r_{start}) + \left(\frac{r - r_{start}}{r_{end} - r_{start}}\right) * R_0(r_{end}), \quad where \quad r = r_{start}, r_{end}$$

Note that the same thing is done for all the filtered autocorrelations and crosscorrelations (lagged moments $\mathbf{R_1}$, $\mathbf{R_2}$, and dual-polarization crosscorrelations) as soon as any of the filtered autocorrelation $\mathbf{R_{0H}}$, $\mathbf{R_{0V}}$, or $\mathbf{R_{0HV}}$ exceeds the threshold. The unfiltered autocorrelations (total power) $\mathbf{T_{0H}}$, $\mathbf{T_{0V}}$, and $\mathbf{T_{0HV}}$ are conserved.

8.4.2 Range averaging and clutter microsuppression

Range averaging can be performed over 2, 3, ..., 16 bins. This reduces the number of bins in the final output to save processing both in the RVP10 and in the host computer.

This is accomplished by averaging the $\mathbf{T_0}$, $\mathbf{R_0}$, $\mathbf{R_1}$ and $\mathbf{R_2}$ values.

At the user's option, the range averaged data can be restricted to include only those bins which have an estimated clutter-to-signal ratio that falls within the CCOR threshold interval.

By excluding isolated point clutter targets from the range average the sub-clutter visibility of the averaged data is increased. Specifically, the Doppler test that is applied to each bin in order that it contribute to the overall sum is:

$$10\log R_0 - 10\log T_0 > CCOR_{thresh}$$

The total power $\mathbf{T_0}$ is conserved after the exclusion of the isolated point clutter targets.

8.4.3 Reflectivity

The corrected reflectivity **Z** is output using a log scale based on the following equation:

$$dBZ = 10\log\left[\frac{T_0 - N}{N}\right] + dBZ_0 + 20\log r + ar + CCOR$$

This equation is a dB version of the familiar radar equation for distributed targets. The relationship between the measured autocorrelation function, the received signal and the noise can be expressed as:

$$T_0 = g^t g^r S + N$$

where $\mathbf{g^t}$ and $\mathbf{g^r}$ represent the transmitter and receiver gains, S is the average back scattered power from the targets and N is the measured average noise power. Neglecting attenuation and the contribution of ground clutter (for the moment), the radar equation can be written as.

$$Z = CSr^2 = \left[\frac{Cr_0^2 N}{g^r g^t}\right]\left[\frac{r^2}{r_0^2}\right]\left[\frac{T_0 - N}{N}\right]$$

where **C** is the radar constant and **ro** is a reference range which we later set to 1 km. This is identical to the first three terms of the dB version of the equation with the definition that:

$$Z_0 = \frac{Cr_0^2 N}{g^r g^t} = Cr_0^2 I_0 \quad where\ I_0 = \frac{N}{g^r g^t}$$

where:

Z_0
: Calibration reflectivity factor. It is the equivalent radar reflectivity factor at the reference range when the return signal power is equal to the noise power (SNR=0 dB). It is sometimes called the minimum detectable dBZ at 1 km, though it is more correct to call it the 0dB SNR detection level.

I_o
: Measured noise power at IF with appropriate calibration for the system gain.

The measurement of $\mathbf{I_o}$ is based on the measurement of the system noise at the time of calibration. However, if the receiver gain changes after calibration, the use of periodic noise sampling properly corrects for this. For example, if the receiver gain were to change by a factor k, then we would measure a noise value of kN and an autocorrelation value of $\mathbf{kT_o}$, that is:

$$Z = CSr^2 = \left[\frac{Cr_0^2 N}{g^r g^t}\right]\left[\frac{r^2}{r_0^2}\right]\left[\frac{kT_0 - kN}{kN}\right]$$

Thus the **k**'s cancel to give us the same result for **Z**. This makes the approach robust to system gain fluctuations. As long as the system sensitivity (noise figure) does not change, then the system does not require re-calibration.

Calibrating RVP10 involves defining the radar constant **C** and measuring the value of $\mathbf{I_o}$. See Reflectivity Calibration (page 303).

The following table shows the individual terms in the dB form of the equation.

Table 47 Terms in the dB equation format

Term	Term name	Description
1st Term	$10\log\left[\frac{T_0 - N}{N}\right]$ *Signal to Noise Ratio*	The effect of this term is to subtract the measured noise and then divide by that noise. The result is a Signal-to-Noise ratio.
2nd Term	$\mathbf{dBZ_o}$: Calibration Reflectivity (see discussion above)	$\mathbf{dBZ_o}$ is the minimum detectable dBZ at a reference range $\mathbf{r_o}$ **=1 km**
3rd Term	**20 log r** : Range Normalization	This term is the range normalization expressed in dB form. $\left[\frac{r}{r_0}\right]^2$
4th Term	**ar**: Gaseous Attenuation Correction	This term accounts for gaseous attenuation. The constant **a** is set in the RVP10 EEPROM since it is a function of wavelength. For a C-band system the default value is 0.016 dB per km (for two-way path attenuation).
5th Term	**CCOR**: Clutter Correction	This term corrects for the measured ground clutter. See Clutter correction (CCOR threshold) (page 192).

8.4.3.1 Noise correction to reflectivity calibration

The **dBZ0** number in the above equation is the number which sets the sensitivity of the radar. Lower numbers mean greater sensitivity. In $Z = CSr^2 = \left[\frac{Cr_0^2 N}{g^r g^t}\right]\left[\frac{r^2}{r_0^2}\right]\left[\frac{T_0 - N}{N}\right]$ it was assumed that the noise level at calibration time is the same as the noise level at run time. And that any changes in measured noise level were due to changes in receiver gain not sensitivity.

Modern digital receivers and low-noise amplifiers are very gain stable, and this is generally not true. One example of a relatively large noise level variation is the thermal noise from the relatively warm earth and atmosphere. So, the bottom degree or so of elevation has a different noise level, and thus a different sensitivity, and thus a different **dBZ0**. Normally we calibrate while aiming the antenna up in the air in a direction away from the surface, or the sun. For this discussion, let us define two new noise values:

N sub c
: Noise level at calibration time.

N sub r
: Noise level at ray processing time.

If we answer yes to "**Enable noise power based correction of Z0**" in the **Mp** non-volatile setup section, then the new radar equation becomes:

$$Z = CSr^2 = \left[\frac{Cr_0^2 Nc}{g^r g^t}\right]\left[\frac{Nr}{Nc}\right]\left[\frac{r^2}{R_0^2}\right]\left[\frac{T_0 - Nr}{Nr}\right]$$

In this equation, the **dBZ0** is the term $\left[\frac{Cr_0^2 Nc}{g^r g^t}\right]\left[\frac{Nr}{Nc}\right]$

The dBZ0 fed into RVP10 is the basic $\left[\frac{Cr_0^2 Nc}{g^r g^t}\right]$, while the processor modifies the value by the ratio **Nr/Nc**. Values read out by the **GPARM** command, and so on are the modified value.

This is the only place where the calibration-time noise level **Nc** is used in the processing.
It is possible for this value to be unknown, in which case it is set to **nan** internally. In this case, if you request the corrected values, the correction is not applied, and you get the message **"Cannot enable Z0 noise correction because calibration is missing"** in the *rvp* log file.

8.4.4 Velocity

For a Doppler power spectrum that is symmetric about its mean velocity, the velocity is obtained directly from the argument of the autocorrelation at the first lag, that is,

$$V = \frac{\lambda}{4\pi\tau_s}\theta_1 \quad where \quad \theta_1 = \arg\{R_1\}$$

λ is the radar wavelength, τ_s is the sampling time (1/PRF). θ_1 is constrained to be on the interval [-π, π]. When $\theta_1 = \pm\pi$, then $V = \pm V_u$ where the unambiguous velocity is ,

$$V_u = \frac{\lambda}{4\tau_s}$$

If the absolute value of the true velocity of the scatterers is greater than V_u, then the velocity calculated by RVP10 is folded into the interval $[-V_u, V_u]$, which is called the Nyquist interval. Folding is usually easily recognized on a color display by a discontinuous jump in velocities. For example, if the true velocity is:

$V_u + \Delta V$

then the velocity calculated by the RVP10 is:

$-V_u + \Delta V$

which is $2V_u$ away from the true mean velocity.

For 8-bit outputs, rather than calculating the absolute velocity in scientific units, RVP10 calculates the mean velocity for the normalized Nyquist interval [-1,1], that is, the output values are,

$$V' = \frac{\theta_1}{\pi}$$

For example, an output value of -0.5 corresponds to a mean velocity of $-V_u/2$. The normalized velocity **V'** is more efficient use of the limited number of bits.

8.4.5 Spectrum width algorithms

The spectrum width is a measure of the combined effects of shear and turbulence.

To a lesser extent, the antenna rotation rate can also effect the spectrum width. At high elevation angles, the fall speed dispersion of the scatterers also effects spectrum width.

There are two choices for the spectrum width algorithm used in the RVP10, depending on the speed and accuracy that are required for the application:

- R_0, R_1 "fast" algorithm valid when SNR >> 10 dB
- R_0, R_1 , R_2 "accurate" algorithm for SNR >> 0 ... 5 dB

Use the **SOPRM** command to select the approach.

8.4.5.1 R0, R1 width algorithm

Given samples of the Doppler autocorrelation function, numerous estimates of spectral variance can be computed (Passarelli & Siggia, 1983). The particular estimator used by the RVP10 employs the magnitudes of R_0 and R_1 and assumes that the Doppler spectrum is Gaussian and that the signal-to-noise ratio is large.

Specifically we have (similar to Srivastava, et al 1979):

$$Variance = 2\ln\left[\frac{R_0}{|R_1|}\right] = -2\ln[SQI]$$

where **ln** represents the natural logarithm. This can be compared to the expression in the preceding section for SQI to illustrate that this expression for the variance is only valid when:

$$\frac{SNR}{SNR+1} \approx 1$$

which occurs when the SNR is large.

This variance estimator is normalized to the Nyquist interval in units of [-π, π]. For example, a variance of $\Pi^2/25$ would be obtained from a Gaussian spectrum having a standard deviation equal to one fifth of the total width of the plotted spectral distribution. For scientific purposes, the spectrum width (standard deviation) is more physically meaningful than the variance, since it scales linearly with the severity of wind shear and turbulence. For these reasons, the width **W** is output by RVP10:

$$W = \frac{\sqrt{Variance}}{\pi}$$

For efficient packing in 8-bits, width is normalized to the Nyquist interval [-1, 1]. For the example given above, the output width **W** would be (1/5). To obtain the width in m/s, multiply the output width by $\mathbf{V_u}$.

8.4.5.2 R0, R1, R2 width algorithm

The width algorithm in this case is similar except that the addition of $\mathbf{R_2}$ extends the validity of the width estimates to weak signals. In this case the variance is:

$$Variance = \frac{2}{3}\ln\left[\frac{|R_1|}{|R_2|}\right]$$

The output width **W** is defined as in R0, R1 width algorithm (page 190).

8.4.6 Signal Quality Index (SQI threshold)

Using the signal quality index (SQI), RVP10 can eliminate signals which are either too weak to be useful, or which have widths too large to justify further analysis.

SQI is defined as:

$$SQI = \frac{|R_1|}{R_0}$$

The SQI is the normalized magnitude of the autocorrelation at lag 1 and varies between 0 for an uncorrelated signal (white noise) and 1 for a noise-free zero-width signal (pure tone).

Mean velocity estimates are degraded when the spectrum, width is large or when the signal-to-noise ratio is weak.

The SQI is a good measure of the uncertainty in the velocity estimates and is a convenient screening parameter to compute.

For very large signal-to-noise ratios (SNR) the SQI is a function of the spectrum width only. For a zero-width pure tone (**W=0**), the SQI is a function of the SNR only (for example, for **W=0**, an SNR of 1 corresponds to SQI=0.5). The SQI threshold is typically set to a value of 0.4 ... 0.5.

In terms of the Gaussian model, the SQI is :

$$SQI = \frac{SNR}{SNR+1} e^{\frac{-\pi^2 W^2}{2}}$$

where the SNR is the signal-to-noise ratio.

8.4.7 Clutter correction (CCOR threshold)

The following content is based on legacy implementation in which R_1 , and R_2 are defined in terms of PPP (pulse-pair-processing).
Current major modes use FFT (DFT) processing instead.

In addition to calculating the R_0, R_1 and optional R_2 autocorrelation terms, which are based on filtered time series data, RVP10 computes T_0 , which is the total unfiltered power.

By comparing the total filtered and unfiltered powers at each range bin, a clutter power, and hence a clutter correction, for that bin can be derived. The clutter correction is defined as,

$$CCOR = 10\log\frac{S}{C+S} = 10\log\frac{1}{CSR+1}$$

where **S** is the weather signal power, **C** is the clutter power and **CSR** is the clutter-to-signal ratio.

The algorithm for calculating CCOR depends on whether the optional R_2 autocorrelation lag is computed.

8.4.7.1 R0, R1 clutter correction

In this case, CCOR is estimated from the following formula:

$$CCOR_{est} = 10\log\left[\frac{R_0}{T_0}\right] = 10\log\left[\frac{S+N}{C+S+N}\right] = 10\log\left[\frac{1+\frac{1}{SNR}}{CSR+1+\frac{1}{SNR}}\right]$$

Here, the expression is strictly valid only when the signal-to-noise ratio (**SNR=S/N**) is large.

When the 2-lag approach is used, the clutter corrections are not as accurate for weak weather signals. However, the error is typically less than 3 dB.

8.4.7.2 R0, R1 clutter correction using clutter signal and noise power calculations

In this case, there is enough information to compute the clutter signal and noise power independently. The algorithm for CCOR is:

$$CCOR_{est} = 10\log\left[\frac{S}{C+S}\right] = 10\log\left[\frac{1}{CSR+1}\right]$$

The clutter power is computed from:

$$C = T_0 - R_0 = [C+S+N] - [S+N]$$

The signal power **S** is then computed from:

$$S = \left|R_1\right| \exp\frac{\pi^2 W^2}{2}$$

W is the width that has been previously calculated. This approach yields more accurate results for the clutter correction in the case of a low SNR.

8.4.8 Weather Signal Power (SIG threshold)

The **SIG** parameter provides an estimate of the weather signal-to-noise ratio in dB for thresholding.

The **SIG** calculation is different depending on whether the optional R_2 autocorrelation is computed.

Table 48 SIG calculation

Autocorrelation	Equation	Description
R_0, R_1	$SIG = 10\log\left[\frac{T_0 - N}{N}\right] + CCOR$	Represents the SNR after the removal of clutter. The **CCOR** value is that described for R_0, R_1 in Clutter correction (CCOR threshold) (page 192).
R_0, R_1 ,R_2	$SIG = 10\log\left[\frac{2\pi S}{R_0 - 2\pi S}\right]$ $+ CCOR$	The SIG is computed based on the SNR. The signal power **S** is determined as described in Clutter correction (CCOR threshold) (page 192).

8.4.9 (Signal+Noise) to Noise ratio (LOG threshold)

The **LOG** threshold parameter is calculated to provide a signal strength estimate that is useful for qualifying reflectivity.

For historical reasons, the **LOG** threshold is not the true SNR (whose dB representation can be both positive and negative) but rather, the ratio of (Signal + Noise) to Noise, which always has a positive representation in dB. Specifically:

$$LOG = 10\log\left[\frac{T_0}{N}\right]$$ *(when applied to the dBT parameter)*

$$LOG = 10\log\left[\frac{R_0}{N}\right]$$ *(when applied to the other parameter)*

8.5 Dual PRF velocity unfolding

For a radar of wavelength λ operating at a fixed sampling period T_s = **1/PRF**, the unambiguous velocity and range intervals are given by the following equation:

$$V_u = \frac{\lambda}{4\tau_s} \; and \;\; R_u = c\frac{\tau_s}{2}$$

where **c** is the speed of light. Often these intervals do not fully cover the span of velocity and range that one would like to measure. The problem is generally worse for short wavelength radars, since that unambiguous velocity span is directly proportional to λ for a given T_s. If the unambiguous range interval is made sufficiently large by increasing T_s, then the resulting velocity span may be unacceptably small.

RVP10 provides a built-in mechanism for extending the unambiguous velocity span by a factor of two, three, or four beyond that given above. The technique, called Dual PRF velocity unfolding, uses two pulse periods rather than one, and relies on the extra information thus obtained to correct (that is, unfold) the mean velocity measurement from each individual period. The Dual PRF trigger pattern consists of alternating (**N+k**)-pulse intervals where the period in each interval is either T_l (for the low-PRF) or T_h (for the high-PRF). Here **N** is the sample size, and **k** represents a delay that permits the clutter filter to equilibrate to the new PRF after each change. The clutter filter impulse response lengths vary according to which filter is selected.

The two trigger periods T_l and T_h must be chosen in either a 3:2, 4:3, or 5:4 ratio. These ratios give factors of two, three, and four times velocity expansion over the T_h period alone. The unfolding algorithm makes use of the following results. Suppose that the radar observes a target with mean velocity V at each of the two trigger periods. The measured phase angles for the R_1 autocorrelations at the two PRFs are:

$$\theta_l = \frac{4\pi V\tau_l}{\lambda} \; and \;\; \theta_h = \frac{4\pi V\tau_h}{\lambda}$$

where angles outside the basic [- π, π] interval are returned to that interval by appropriate additions of ± 2π . These angles correspond to the ordinary single-PRF Doppler velocity measurements, and the ± 2π uncertainties reflects the fact that each measurement is folded into its own unambiguous interval:

$$V_{ul} = \frac{\lambda}{4\tau_l} \quad and \quad V_{uh} = \frac{\lambda}{4\tau_h}$$

If we define ϕ to be the difference between the two measured phases then:

$$\phi = \theta_l - \theta_h = \frac{4\pi}{\lambda}[\tau_l - \tau_h]V$$

which can be interpreted as a phase angle within the unfolded interval:

$$V_u unfold = \frac{\lambda}{4(\tau_l - \tau_h)}$$

Now if τ_l and τ_h are in a 3:2 ratio, then:

$$\tau_l - \tau_h = \frac{\tau_l}{3} = \frac{\tau_h}{2}$$

and thus:

$$V_u unfold = 3V_{ul} = 2V_{uh}$$

The angle Ø represents a velocity phase angle in [- π, π] , but with respect to an enlarged unambiguous interval. By differencing the folded angles from the high and low PRFs, we obtain an angle that is unfolded to a larger velocity span. Similar reasoning shows that the 4:3 ratio gives a factor of three improvement over V_{uh} , and 5:4 gives a factor of four.

Velocity estimator

In practice, the unfolded angle Ø is not in itself a suitable velocity estimator. This is because the variance of Ø is equal to the sum of the variances of each of its components, that is, twice that of the individual measurements alone. If the target is at all noisy, then this increase in variance can be severe. Rather than use Ø directly, RVP10 uses it only as a rough estimate in determining how to unfold the individual velocity measured from each PRF.

This technique is illustrated in Figure 34 (page 196). The figure shows how the low-PRF and high-PRF angles are unfolded based on the difference angle. The diagrams show phase planes representing the large unfolded velocity interval, and the locations of vectors on those planes. In the diagram on the right, the difference angle is plotted, and the plane is divided into two equal size regions, one of which is centered on the difference vector. The high-PRF angle is then divided by two and plotted. The resultant unfolded velocity angle must either be this vector, or this vector plus. Since adding places the vector into acceptance **`Region 1`**, where it is nearest the difference angle, it can be concluded that this is the correct unfolding. Likewise, on the diagram on the left, we unfold the low-PRF angle by dividing the plane into thirds centered on the difference angle. The result angle is one of the following equations, depending on which one falls into the acceptance **`Region 1`**. The resultant angle is the same in each case.

$$\frac{\theta_l}{3}, \frac{\theta_l}{3} + \frac{2\pi}{3} \; or \; \frac{\theta_l}{3} + \frac{4\pi}{3}$$

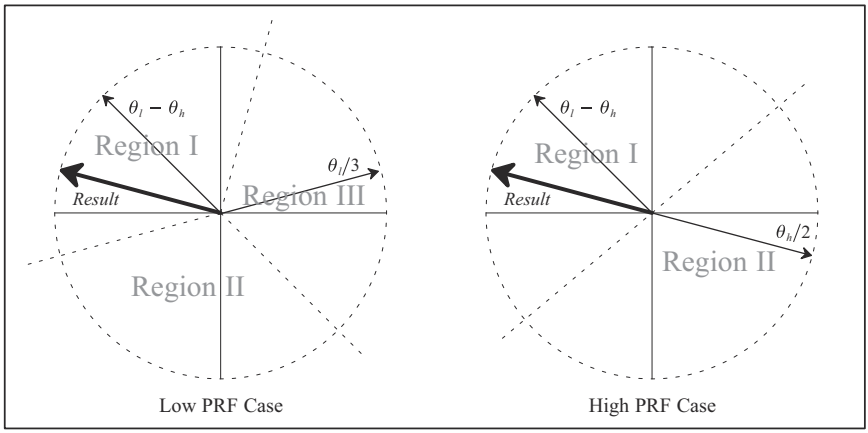


Figure 34 Dual PRF concepts

RVP10 makes efficient use of the incoming data by unfolding velocities from both the low and the high-PRF data, making use each time of information in the previous ray. When low-PRF data is taken the derived velocities are unfolded by combining information from the previous high-PRF interval. Likewise, when high-PRF data are acquired the velocities are unfolded based on the previous low-PRF interval. Thus, when operating in the Dual PRF mode, RVP10 outputs one data ray for each `(N+k)`-pulse interval. However, the velocity data in the Dual PRF rays are unfolded, so that the [-1,+1] interval now represents either two or three times the prior velocity range. Put another way, the data are still interpreted as described in the section on mean velocity estimation, except that V_u is now larger.

Width data

The width data is modified during Dual PRF unfolding.

Although valid widths are obtained independently on all rays, those measured at low-PRF are larger than those at high-PRF. This is because the dimensionless width units are with respect to a larger velocity interval in the latter case. To compensate for this, low-PRF widths are multiplied by either 2/3 or 3/4 before being output. This puts them in the same scale as the high-PRF values, and thus, the widths do not vary on alternate pulses. A useful consequence of this is that width data can be sent directly to a color display generator without having to plot every other ray in a different scale.

Limitations of Dual PRF processing

The unfolding algorithms make the assumption that targets are more or less continuous from ray to ray. Otherwise, it would not make sense to use data from a previous ray to unfold velocities in the current ray. You must ensure that their antenna scan rate and beamwidth are such that each target is illuminated, at least partially, over each full **2(N+k)**-pulse interval. In practice, a certain amount of decorrelation from ray to ray is acceptable, since the previous rays are used only to decide into which unfolded interval the current ray should be placed. Small errors in the previous ray data, therefore, cause no error in the output. However, large previous-ray errors would lead to incorrect unfolding.

A more subtle side effect of Dual PRF processing arises from clutter filtering because clutter notches now appear at several locations in the unfolded velocity span, rather than just at zero velocity. These additional rejection points come about because the original velocity intervals are mapped some integer number of times to create the unfolded interval.

Since each original interval has a clutter notch at DC, it follows that the final expanded velocity interval has several such notches. For example, in the 3:2 case, in addition to removing DC, the clutter filter removes velocities at - $2V_u/3$, + $2V_u/3$, and V_u .

These clutter filter "images" are a consequence of the Dual PRF processing technique and are not easily removed. They can cause trouble for the velocity unfolding itself, and they may cause the computed clutter corrections to be wrong at the image points.
To minimize their impact, turn the clutter filter off at far ranges where little clutter is expected, and use a narrow clutter filter.

The 4:3 and 5:4 PRF unfolding ratios are more susceptible to unfolding errors in cases where the spectrum width is large and/or the SNR is low. You must experiment with these ratios to determine which provides the best results for their particular application. Although the RVP10 trigger generator can produce any trigger frequency, only the 3:2, 4:3, and 5:4 ratios can be used with the built-in unfolding algorithms. The RVP10 still permits other PRT ratios to be explored, but the unfolding technique must then be manually programmed on your host computer.

Example

The following example shows seven possible oscilloscope traces (and their associated probabilities) for the RVP10 trigger during Dual PRF operation.

The PRF ratio is 4:3, and the sample size is 50 pulses at the high PRF, and 37 pulses at the low PRF. The signal labeled **SCOPE** is the composite of these traces, and is what is shown on an oscilloscope.

Note that there are a number of low probability pulses. The exact details of the sample sizes and the trigger hold-off time can make the low probability pulses appear to come and go randomly. This is normal and no cause for alarm.

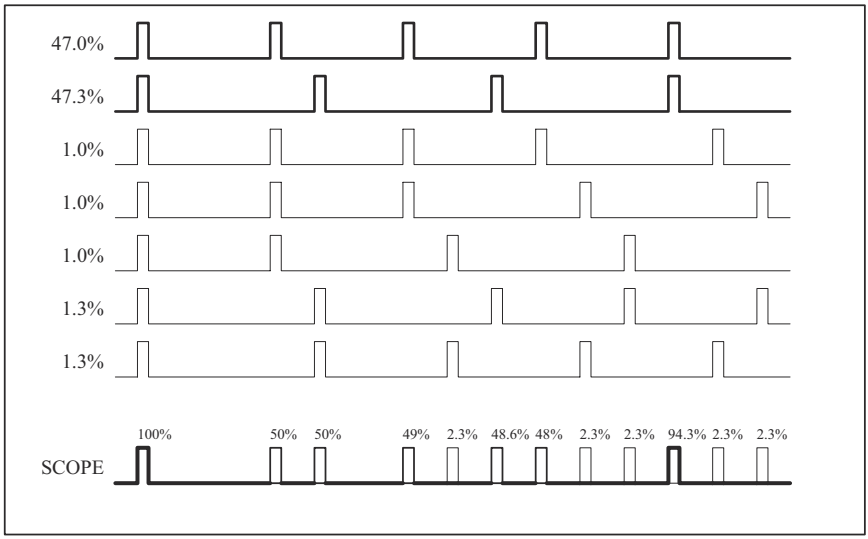


Figure 35 Example of Dual PRF trigger waveforms

8.6 Random phase second trip processing

Second trip echoes can be a serious problem for applications when the radar is operated at high PRF (for example, >500 Hz). Second trip echoes are caused by the range aliasing of targets. They appear as false echoes on the display, usually elongated in the radial direction. On Klystron systems they have valid Doppler velocities. On magnetron systems, the Doppler velocities are not valid, but the noise from the 2nd trip echoes can obscure valid first trip velocity information.

RVP10 has optional random phase processing for the filtering and recovery of second trip echoes. While details of the technique are proprietary to Vaisala, we describe the general principle and the configuration options to optimize the algorithm performance.

The information that is used to separate the first and second trip echoes is the phase.

Magnetron radars

For a magnetron radar, the phase of each pulse is different. When 1st and 2nd trip echoes are received simultaneously, the phase of the first trip return is different from the phase of the second trip return. RVP10 measures the phase of the transmitted pulse and the phase locking is done digitally.

Klystron radars

For a Klystron radar, the phase is controlled by RVP10 through a digital phase shifter that is precisely calibrated. Typically, the Klystron Stalo is phase shifted so that each transmit pulse has a different phase.

The sequencing is controlled by RVP10.

8.6.1 Random phase second trip processing algorithm

The following figure shows a schematic of the data processing for random phase. The figure shows the Doppler spectra for the first and second trip in the processing stages. The vertical scale is in dB and the horizontal scale is velocity. In this example, the second trip echo is shown as being stronger than the first trip echo (usually the reverse is true).

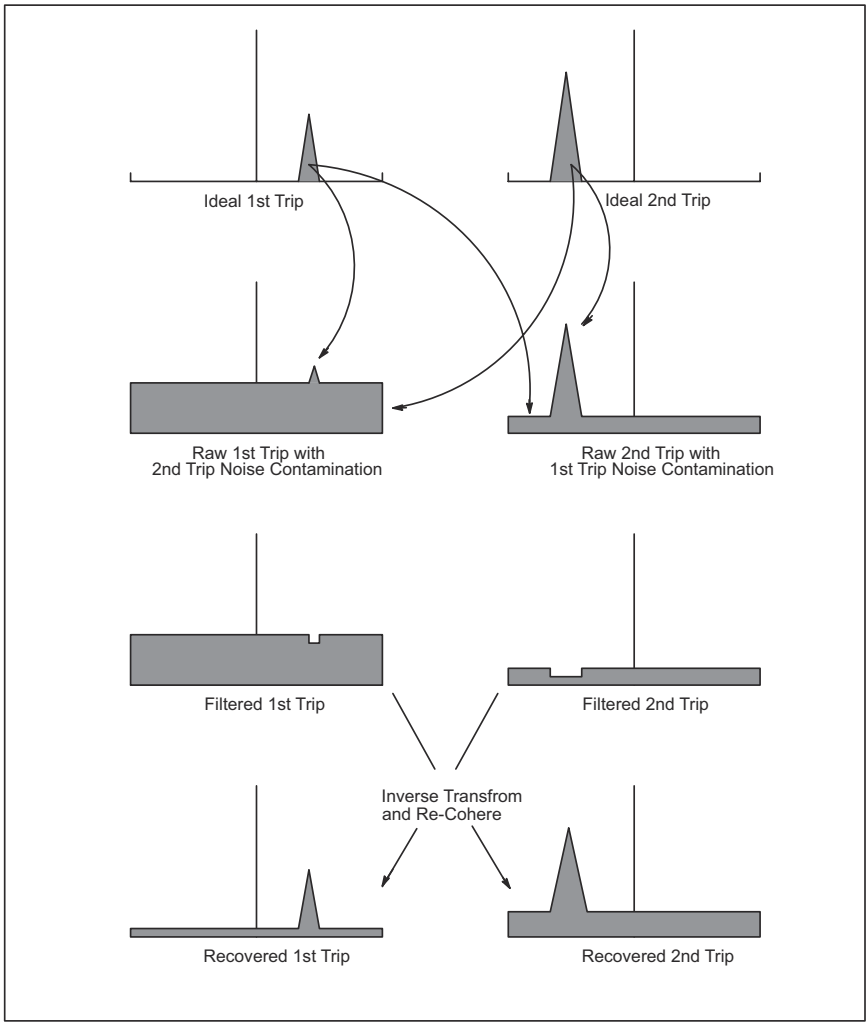


Figure 36 Random phase processing algorithm

Ideal 1st and 2nd trip echoes

The ideal 1st and 2nd trip echoes represent the echoes as they would appear individually. The ideal 1st trip echo is the echo that would be measured if there was no second trip echo interference. The ideal second trip echo represents what would be measured if there was no 1st trip echo interference. If there is no interference from the other trip, a standard Klystron system can measure the ideal spectra, but there is no way to know whether the echoes are in the first or second trip.

Raw 1st and 2nd trip echoes

The figure Figure 36 (page 200) above shows how the echoes from the first trip and second trip interfere with each other. For the case of a standard magnetron system, the first trip echo is coherent, while the second trip echo is incoherent (white noise) since the phase of the second trip echo is random. This is because the receiver is phase locked only to the first trip.

Another way to implement a magnetron system is to let the Stalo free-run (rather than phase locking to the transmit pulse), measure the phase of each transmit pulse and digitally correcting for the transmit phase. Using this digital phase locking technique, the RVP10 can phase lock or "cohere" to either the first or the second trip.

Using this technique alone, it is possible to distinguish between 1st and 2nd trip echoes for the case when the echoes are not overlapped. The echoes appear as the idealized first and second trip echoes. This range de-aliasing effectively doubles the range of the radar. The problem is that when echoes are overlapped, the noise contamination from the stronger echo make it impossible to measure the weaker echo. This is illustrated in the figure. Thus if the first trip echo has a good signal-to-noise ratio of 10 dB, then the 2nd trip echo has a signal-to noise-ratio no better than -10 dB. This is the fundamental problem with using phase alone to separate the 1st and 2nd trip echoes.

Filtered 1st and 2nd trip echoes

Since the strong echo generates noise that obscures the weaker echo, the approach used in RVP10 is to filter the echo from the other trip — the whitening filter. This is shown in the figure. The adaptive whitening filter removes both the clutter and the weather. All of the phase information for the other trip is then contained in the white noise portion of the spectrum. The phase information under the coherent echo that is removed is dominated by the coherent echo, that is, the other trip phase information is contaminated. For this reason, the filtering should affect as small a region of the spectrum as possible.

8.6.2 Tuning for optimal performance

The random phase algorithms are controlled by the same collection of setup and operational parameters that apply to all of the other processing modes, for example, choice of sample size, clutter filter, angle sync, calibration, and so on.

However, the following parameters are special to random phase mode.

Secondary SQI threshold

In standard Doppler processing, an SQI threshold is normally not applied to reflectivity data, because it would cause that data to be rejected in regions of high spectral width. However, in the random phase mode, if SQI is applied to reflectivity or dual-polarization data, we need to do it, especially in random phase processing, because reflected power can only be assigned to a particular trip when it is coherent within that trip. Incoherent echoes, regardless of their strength, cannot be placed in either trip.

An SQI threshold is required to qualify reflectivity and dual-polarization data in all the processing modes. RVP10 defines a secondary SQI threshold $\mathtt{SQI}_2$ that is computed from the standard threshold value as:

$$SQI_2 = Offset + (Slope \times SQI)$$

where **Slope** and **Offset** are the secondary SQI threshold parameters defined in the **Mf** setup section.

The factory default values are (**Slope** = 0.50) and (**Offset** = 0.05), that is, the secondary threshold is a little less than half of the standard value.

The algorithms check whether the SQI of each recovered trip is less than the secondary SQI threshold, and if so, the **LOG** portion of the data are rejected. This SQI test is necessary for a clean **LOG** picture, but we need to use a more permissive (lower) threshold value than would usually be applied to the reflectivity and dual pol data alone.

The **Slope** and **Offset** values should be adjusted so that the density of speckles in LOG data is approximately the same as the density of speckles in FFT velocity data for a given primary SQI value. You may then adjust the primary SQI threshold to achieve the appropriate trade-off of speckles versus sensitivity for your system in all modes of operation. Even with proper adjustment, it is normal for dual-polarization, dBZ and dBT data to show gaps in regions of weather that have high turbulence or shear when SQI threshold is applied to that data. These dropouts usually match similar gaps in the velocity and width data, both of which are traditionally thresholded by SQI.

Maximum power ratio between trips

The adaptive filtering that is performed on the data for each trip greatly extends the visibility of a weak echo that is overlapped with a much stronger one. In practice, the filtering process is often able to remove 25 ... 35 dB of dominant power in order to reveal a much weaker echo in the other trip. The performance depends on many factors, primarily the spectral width of the dominant echo, and the overall stability of the radar system.

The difficulties of removing a dominant "other trip" echo from a weather signal are analogous to the challenge of removing a dominant clutter target from that same signal. In both cases we are trying to extract a weak weather signature using a filtering procedure that relies on the spectral confinement of the stronger signal.

The Clutter-to-Signal Ratio (CSR) parameter can be adjusted to control sub-clutter visibility. Just as the CSR applies to the clutter filters, it can also be used to place similar limits on the depth of visibility of the adaptive filters.

Example: RVP10 is operating in random phase mode at a PRF of 1500Hz, and is observing widespread weather having uniform intensity in both the first 100-km trip and the second 100-km trip. If the CSR was set too conservatively at only 15 dB, then the algorithm would generally be blind to the second-trip weather in the range interval 100 km ... 100 km.

The explanation for this can be found in the $1/r^2$ geometric correction for weather echo intensity. At ranges less than 17.8 km, the first trip weather would generally dominate the second trip weather by more than 15 dB. Thus, the initial 17.8 km ring of second trip data would be rejected by the CSR criteria. However, if the CSR were increased to 30 dB, then the size of this missing ring would be reduced to only 3.2 km.

If the CSR is set too low, there is an abrupt ring of missing data in the beginning of the second trip. If set too high, there are speckles and other spurious effects within this same interval. The optimum setting should strike a balance between these two effects.

R1 vs. R2 algorithms

The random phase algorithms for adaptive filtering and separation of trips relies on having the best possible information about the weather's SNR and spectral width. Thus, the **R2** Doppler algorithms are always used, regardless of the setting of the **R1/R2** flag in the operational parameters.

Random phase and Dual PRF

The random phase processing works seamlessly with the dual PRF processing to provide advanced range and velocity ambiguity resolution. Both the first and second trip echoes can be recovered and displayed to a maximum range of 2X the unambiguous range corresponding to the high PRF.

For optimum performance, the 2D 3x3 speckle filter should be used to smooth the second trip seams that occur for each ray. In fact, this smoothing of the second trip seam makes the dual PRF random phase mode work even better than the single PRF random phase.

For more information, see Random phase second trip processing algorithm (page 199).

8.7 Signal generator algorithm testing

The IF signal generator tests that can be used to verify the RVP10 processing algorithms.

Perform these tests after adding new algorithms or major modes to the processor.

You can use the test descriptions to debug your system, or better understand how they work.

8.7.1 Linear Ramp of Velocity with Range

Suppose that a continuous-wave IF waveform has an instantaneous frequency f (t)in Hertz (cycles/sec). Consider a range bin located at time

τ_{bin} within a set of pulses that are separated by τ_s = 1/PRF. The phase measured at that bin on the n^{th} pulse is the integral of the frequency within that pulse starting from range zero (since RVP10 is phase-locked to range 0):

$$\Phi_n = \int_{n\tau_s}^{n\tau_s + \tau_{bin}} f(t)\,dt$$

If we assume that the input frequency is a linear Frequency Modulation (FM) at the rate of M cycles/sec/sec on top of a base frequency T_o, then:

$$\Phi_{n+1} - \Phi_n = \int_{(n+1)\tau_s}^{(n+1)\tau_s + \tau_{bin}} (T_0 + Mt)\,dt - \int_{n\tau_s}^{n\tau_s + \tau_{bin}} (T_0 + Mt)\,dt = (M\tau_s)\tau_{bin}$$

which is independent of both T_o and **n**. Thus, a linear FM input signal produces a fixed (**I**,**Q**) phase difference from pulse-to-pulse at any given range. The magnitude of the phase difference is proportional to the range, and the slope is(**M** τs) cycles for each second of delay in range. For example, if the test signal generator is sweeping 100 KHz every two seconds, then the velocity observed at a range of 300 km at 250 Hz PRF is:

$$\Phi_{n+1} - \Phi_n = \left(\frac{100KHz}{2\text{sec}}\right) \times \left(\frac{1}{250}\text{sec}\right) \times (300km) \times \left(\frac{6.6\mu\text{sec}}{1km}\right) = 0.40\ cycles$$

We would thus observe a velocity of (0.8 × **Vu**) at 300 km, where **Vu** is the unambiguous Doppler velocity in meters/sec. Note that these phase difference calculations have made no assumptions about the RVP10 processing mode, and thus are valid in all major modes (PPP, FFT, DPRT, RPH), as well as in all Dual-PRF unfolding modes.

This simple FM signal generator also produces valid second trip velocities that can be seen during Random Phase processing. This follows from the above analysis because we've never assumed that τbin was smaller than τs, that is, it is fine for the range bin to be located in any higher-order trip.

8.7.2 Verifying PHIDP and KDP

The PHIDP and KDP processing algorithms can be tested using CW signal sources at IF.

In the alternating-transmitter single-receiver case, a single FM signal generator is modulated with an RVP10 polarization select line so that slightly different frequencies are generated for the H and V pulses. A maximum FM depth of several kilohertz is all that is required.

In the dual-receiver case, two (unmodulated) signal generators are used for each of the H and V intermediate frequencies, and one or the other is detuned slightly from its correct center frequency.

In either case the frequency difference that produces a KDP value of 1.0 degree/km is:

$$\left(1.0\frac{\text{deg}\,ree}{km}\right) \times \left(\frac{1}{360}\frac{cycles}{\text{deg}\,ree}\right) \times \left(299792\frac{km}{\text{sec}\,ond}\right) = 833\frac{cycles}{\text{sec}\,ond}$$

8.7.3 Verifying RHOH, RHOV, and RHOHV

These terms measure the normalized cross-channel covariance in a polarization radar. They all are computed having the form:

$$RHOAB = \frac{< S_A^n \; S_B^{n*} >}{\sqrt{< S_A^2 \; S_B^2 >}}$$

Where the S_A^n and S_B^n are complex (**I**,**Q**) vectors from two receiver channels A and B, and "<>" denotes expected value. This suggests that some form of amplitude modulation (AM) of the input signal might be helpful.

Suppose that the S_A^n and S_B^n samples are coming from two signal generators installed on a dual-receiver system, and that only the B-Channel is AM modulated so that:

$$\left|S_A^n\right| = \{S_A, S_A, S_A, S_A, S_a \ldots\}, \;\; \left|S_B^n\right| = \left\{S_B, 0, S_b, 0, S_B \ldots\right\}$$

Then the above estimator reduces to:

$$RHOAB = \frac{\left(\frac{1}{2}\right) S_A S_B}{\sqrt{S_A^2 \times \left(\frac{1}{2}\right) S_B^2}} = 07.07$$

A simple way to create these data is to set the A-Channel siggen for 95% AM depth, and use a sinusoidal modulation source of, perhaps, 400 Hz. We do not choose 100% depth because we would lose the burst phase reference when the amplitude became smallest. The 26 dB reduction in $\mathbf{S_B}$ is a close enough approximation to zero in the above formula.

If we now observe the two receive channels with the RVP10 at a PRF of 800Hz, we see the RHOAB terms varying with range; reaching a high value of 1.00, and a low value of 0.707. The plots are nearly stationary on the **Ascope** screen because the PRF is almost precisely twice the modulation rate (though they are free-running relative to each other).

Adjusting the amplitude of either signal generator is not affect the p terms, but it does have an interesting effect on SQI. If (T,Z,V,W) are computed from both channels combined, then the SQI is:

$$SQI = \frac{S_A^2}{S_A^2 + \left(\frac{1}{2}\right) S_B^2}$$

If we solve this equation for **SQI=0.5** we find that the individual $\mathbf{S_A}$ terms must have twice the power of the individual $\mathbf{S_B}$ terms. This can be checked by adjusting either signal generator until the minimum plotted SQI is 0.5, and then verifying that the average H and V powers are identical; or, equivalently, that ZDR, LDRH and LDRV are 0.

The linear FM ramp (see Linear Ramp of Velocity with Range (page 203)) can also be used as a test of RHOAB in a dual-receiver system. With one siggen modulated and the other fixed, one receive channel appears to rotate relative to the other. If the FM modulation is such that **1/N** of a full revolution occurs per pulse at a given range, then if the sample size is **N** pulses we observe **RHOAB = 0** at that range. The plot of RHOAB shows a characteristic **sin (x)/x** behavior as a function of range.

9. Dual-polarization algorithms

9.1 Introduction to dual-polarization

9.1.1 Dual-polarization overview

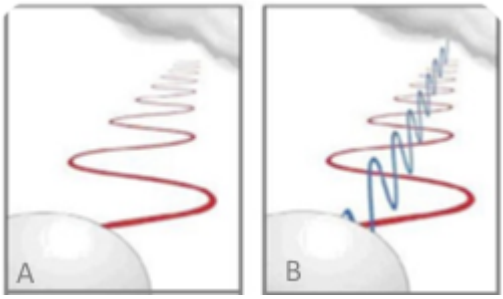


Figure 37 Single-polarization and dual-polarization radars

A Single-polarization weather radar
B Dual-polarization weather radar

Single-polarization weather radars transmit and receive only a single polarization (usually horizontal). Dual-polarization radars transmit and receive both horizontal and vertical polarization.

Vaisala Hydrometeor Classification (HydroClass) software makes optimal use of dual-channel measurements to detect the types of scatterers present in the atmosphere, such as rain, hail, snow, graupel, and even non-meteorological targets such as insects, chaff, and sea clutter.

9.1.2 Radar system considerations

A polarization radar is characterized by how it transmits and receives. For simplicity, we assume that the radar uses horizontal and/or vertical polarization. However, other polarization pairs could be used (for example, right and left circular polarization).

9.1.2.1 Transmit Modes

Fixed (horizontal or vertical)
Controlled by a switch, or the radar can be fixed to transmit in single-polarization. If a switch is used, it can be a simple, slow waveguide switch rather than a fast switch (pulse-to-pulse).

Alternating (horizontal and vertical)
Radar alternates pulse-to-pulse between horizontal and vertical. A high-power fast switch is used to switch the polarization between the 2 channels.

Simultaneous (horizontal and vertical)
Horizontal and vertical are transmitted simultaneously.

9.1.2.2 Receive modes

Single-channel receiver
Used only for alternating transmission. The receiver typically receives the co-polarized radiation (transmit H and receive H, then transmit V and receive V).

Dual-channel receiver
Receives in 2 channels (H and V) simultaneously.

The following table summarizes the transmit and receive cases, and the polarization variables that are available for each.

The fixed, dual-channel cases allow the optional dual-polarization cross-polarization LDR and the co-pol/cross-pol correlation amplitude and phase to be measured (for example, RhoH and PhiH).

Standard parameters are available for all cases (dBT, dBZ, V, and W). RVP10 supports all these cases.

Table 49 Transmitter types

Receiver type	Transmitter type			
	Fixed H	**Fixed V**	**Alternating H and V**	**Simultaneous H+V**
Single-channel	Single-polarization radar	Single-polarization radar	Z_V, Z_{HV}, Z_{dr}, RhoHV PhiDP, K_{dp}	N/A
Dual-channel	LDRH, RhoH, PhiH	LDRV, RhoV, PhiV	The alternating transmission, dual-channel receiver allows both the co-pol and the cross-pol measurements to be made. It is the most complete. Z_V, Z_{HV}, LDRH	The simultaneous H+V transmission and dual-channel reception (also called the STAR mode for simultaneous transmit and receive). This allows the co-pol measurements to be made (Z_{dr}, RhoHV, PhiDP, K_{dp}). Z_V, Z_{HV}, Z_{dr}, RhoHV, PhiDP, K_{dp} (STAR mode)
			LDRV, RhoH, RhoV, PhiH, PhiV, Z_{dr} RhoHV, PhiDP, K_{dp}	

9.1.2.3 Summary of radar system characteristics

RVP10 supports all these transmit and receive modes.

The measurement of cross-pol parameters, such as LDR (fixed or alternating transmission and dual-channel reception), requires a radar system that has been optimized for cross-pol isolation, for example, an offset feed antenna and no radome. By removing the feed, support struts, and radome from the path of the radiation, the cross-pol isolation can be improved.

The single-channel, alternating method has been used in several polarization radars for Z_{dr} measurement. The advantage of this approach is that it is relatively easy to modify a single-polarization radar by adding a dual-port feed and a high-power, fast switch above the antenna rotary joints. The disadvantage is that the switch is costly and requires maintenance.

The STAR (simultaneous transmit and receive) mode does not require a switch and the components are fairly reliable. The disadvantage of the approach (as it is usually implemented) is that a dual-rotary joint and dual waveguides are required to duct both the H and the V through the antenna pedestal up to the antenna feed. Despite this, the STAR mode offers the best approach for upgrading an existing radar.

9.1.3 Horizontal, vertical, and enhanced reflectivities (Z_H, Z_V, Z_{HV})

In the case of amplitude (power) measurements, the power returned to the radar can be related either to the particles diameter in the horizontal plane or vertical plane. When expressing this power through the radar equation, these become horizontal reflectivity (Z_H), and vertical reflectivity (Z_V).

Z_H and Z_V are typically calculated using an autocorrelation power estimator, or by performing a cross-correlation of the signal with itself.

The autocorrelation technique finds periodic signals buried within noise. A special case is to perform the cross-correlation of the horizontal power to the vertical power. This becomes enhanced reflectivity (Z_{HV}) and can be shown to have greater ability to weak signals in the presence of noise, compared to Z_H or Z_V alone.

All power estimators are sensitive to intervening attenuation from precipitation, requiring a data correction for quantitative use with C- or X-band radars.

9.1.3.1 Differential reflectivity (Zdr)

In the case of amplitude (power) measurements, the larger horizontal axis of drops causes the power measured at horizontal polarization (of the electric field) to be larger than the power measured at vertical polarization. The ratio of the reflectivity factors Z_H/Z_V, expressed in dB, is called differential reflectivity (Z_{dr}).

- Zdr is generally positive in rain (that is, >1) and is usually less than 5 dB. When the rainfall rate is large, there are typically more large drops, so Z_{dr} is larger.
- Low Z_{dr} and high dBZ indicates the presence of hail, which may be tumbling with no preferred orientation.

Z_{dr}, because it is a ratio of powers, is not sensitive to the radar calibration, as long as the overall gain of the H and V channels is the same (or calibrated).

9.1.3.2 Differential Phase (PhiDP) and Specific Differential Phase (Kdp)

In the case of phase measurement, the speed of propagation is also affected by the asymmetry of the larger drops. Because of the longer dimension of the horizontal axis of drops, the medium is more dense for horizontal than for vertical polarization. This causes the horizontal wavelength to propagate slightly slower (more phase cycles per unit distance) in comparison with the vertical wavelength, which leads to a phase difference between horizontal and vertical.

This difference is called the differential phase shift, PhiDP. Phase difference accumulates (PhiDP increases) with range as a function of the relative difference in density that the horizontal and vertical microwaves must propagate through. An important concept to understand about PhiDP is that the measurement value is not indicative of what is present at a particular range, but what the microwaves have travelled through to get to that point (and back).

The range derivative of PhiDP, that is, the change of phase per unit distance, is called specific differential phase (K_{dp}). K_{dp} is almost directly proportional to the rainfall rate, so it has the potential to improve precipitation rate measurements compared to traditional Z-R relationship measurements, which can be highly inaccurate.

9.1.3.3 Linear Depolarization Ratio (LDR)

Alternating polarization radars can transmit a single polarization and receive simultaneously in two channels, usually the co-polarized and cross-polarized components. For example, when transmitting horizontal, both horizontal (co-polarized) and vertical (cross-polarized) polarized energy are received by 2 separate channels.

In the case of vertical or horizontal, the ratio of the power Zcross or Zco is called the linear depolarization ratio (LDR). The amount of incident radiation that is depolarized by a particle, depends on the particle shape and orientation (for example, canting angle with respect to horizontal). Perfectly spherical particles do not depolarize either horizontal or vertical polarization, so the LDR is 0. Particles that are wet, tumbling and irregularly shaped give larger LDR values. Therefore, LDR values in rain tend to be small, for example, less than -25 dB. Larger values of LDR can occur in the bright band or in the presence of hail.

A radar and antenna system must be optimized to measure LDR by assuring that the antenna, feed and supporting struts, and radome are not themselves depolarizing the transmitted and received radiation. This is called "cross-pol isolation". The integrated cross-pol isolation of the antenna pattern must be more than 35 dB for LDR measurement. Stratiform rain typically causes the least amount of depolarization which had been thought to be around a value of -35 dB in LDR. However, as radar designs have improved, lower levels of depolarization in rainfall have been measured.

If the radar cannot achieve this level of cross-pol isolation it is limited to how accurate the rainfall estimates can be from the dual-polarization data.

9.1.3.4 [RhoHV, PhiDP] [RhoH, PhiH] [RhoV, PhiV]: Correlation Variables

Depending on the capabilities of the radar, several correlation functions can be calculated. These are generally complex, having both an amplitude and phase. These are normalized so that a perfect correlation magnitude is 1 and perfectly decorrelated is 0. In locations with low signal-to-noise ratio, there is typically a noise correction contained within the calculation. Otherwise, the non-correlated noise would lower the measure correlation of the targets.

RhoH and PhiDP are the magnitude and phase of the correlation between the horizontal and vertical co-polarized channels. These are available on H/V switching systems or on systems that transmit simultaneous H and V.

PhiDP can be used to infer precipitation rate. RhoHV, in rain, is typically very close to 1 (0.98). RhoHV values can be reduced in the case of irregularly shaped, randomly oriented, wet tumbling particles; thus, RhoHV provides information on the particle type.

RhoH and PhiH are the magnitude and phase of the correlation between the co-polarized and cross-polar channels for H transmission, and simultaneous H and V reception. RhoV and PhiV denote the cross-channel correlation magnitude and phase for vertical transmission. These are available on a dual-channel receiver with transmit, either fixed or alternating. The information content of the cross-pol correlations is the topic of current research.

9.1.4 Processing algorithms

RVP10 supports four polarization modes summarized in the table below. For each case, the standard moments (T, Z, V, and W) are also calculated.

The notation for the outputs used here is similar to that in standard use (for example, Doviak and Zrnic). However, for LDR we use the notation LDRH to indicate that this is the LDR for horizontal transmission. The notation RhoH and PhiH indicates the magnitude and phase of the covariance between the co- and cross-polarized channels for H transmit.

Table 50 Supported polarization modes and outputs

Case	Transmit	Receive	Processing mode	Polarization outputs
1	Fixed horizontal or fixed vertical	Dual-channel	PPP only	LDRH, RhoH, PhiH, or LDRV, RhoV, PhiV
2	Simultaneous H +V (STAR mode)	Dual-channel	PPP or Z_{dr} for FFT, Random Phase, and DPRT1&2	Z_V, Z_{HV}, Z_{dr}, PhiDP, K_{dp}, RhoHV
3	Alternating H/V	Single-channel	PPP only	Z_V, Z_{HV}, Z_{dr}, PhiDP, K_{dp}, RhoHV
4	Alternating H/V	Dual-channel	PPP only	Z_V, Z_{HV}, Z_{dr}, LDRH
				RhoH, PhiH, LDRV, RhoV, PhiV, PhiDP, K_{dp}, RhoHV

9.1.4.1 Input receiver sample notation

For the discussion of polarization, we use the notation used by Doviak and Zrnic. The received signal for pulse **n** from a single range bin is denoted as:

Table 51 Input receiver algorithm notation

Notation	Description	
s^n_{hh}	Receive **h**: Transmit **h**	Horizontal co-polar signal
s^n_{vh}	Receive **v**: Transmit **h**	Horizontal cross-polar signal
s^n_{vv}	Receive **v**: Transmit **v**	Vertical co-polar signal
s^n_{hv}	Receive: **h** Transmit **v**	Vertical cross-polar signal

The pulse index is indicated by the superscript instead of the subscript. The first subscript indicates the received polarization and the second subscript indicates the transmit polarization.

If the transmit is the same as the received polarization, this is the co-polarized signal. If the transmit and receive are different, this is the cross-polarized signal.

These variables are complex and are the same as the s_n notation. For example, we can write:

$$S^n_{hh} = I^n_{hh} + jQ^n_{hh}$$

to show the relationship to the received `I` and `Q` values. Either filtered and unfiltered versions of the samples can be selected for processing. We drop the `s'` notation for filtered samples.

9.1.4.2 Notation and model for correlations

The pulse pair processing mode is used for all polarization calculations, except that Z_{dr}-only processing for the STAR case can be done in either FFT or random phase as well as pulse pair.

As with the standard moments, autocorrelations form the basis for the processing of the polarization variables.

An autocorrelation is the cross-correlation of a signal with itself. Informally, it is the similarity between observations as a function of the time separation between them. It is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal, which has been buried under noise. However, it is also possible to perform a correlation of the S_{hh} and S_{vv} signal, which is used to create Z_{HV}.

The autocorrelations are computed similarly to the standard moments. For example, in pulse pair mode the autocorrelations for the horizontal transmit co-polar channel are:

$$T_{0hh} = \frac{1}{M} \sum_{n=1}^{M} S^n_{hh} \times S^n_{hh}$$

$$R_{0hh} = \frac{1}{M}\sum_{n=1}^{M} s'^{n}_{hh} \times s'^{n}_{hh}$$

$$R_{1hh} = \frac{1}{M-1}\sum_{n=1}^{M-1} s'^{n}_{hh} \times s'^{n+1}_{hh}$$

$$R_{2hh} = \frac{1}{M-2}\sum_{n=1}^{M-2} s'^{n}_{hh} \times s'^{n+2}_{hh}$$

For dual polarization systems, these correlations can be applied up to 4 different ways (R_{hh}, R_{vv}, R_{hv}, and R_{vh}), where R_{hv} and R_{vh} are equivalent. The physical model for the channel powers is identical to the model used for the standard moment cases:

Co-channel power

$$R_0^{hh} = \frac{1}{M}\sum_{n=1}^{M-1} s'^{n}_{hh} \times s'^{n}_{hh} = g_h^r g_h^t S'_{hh} + N_h$$

$$R_0^{vv} = \frac{1}{M}\sum_{n=1}^{M-1} s'^{n}_{vv} \times s'^{n}_{vv} = g_v^r g_v^t S'_{vv} + N_v$$

Diagonal channel power

$$R_0^{hv} = \frac{1}{M}\sum_{n=1}^{M-1} s'^{n}_{hh} \times s'^{n}_{vv} = \frac{g_h^r g_h^t S'_{hh} g_v^r g_v^t S'_{w}}{2}$$

Here **S** denotes the actual backscatter average power to the radar. When multiplied by the appropriate transmitter and receiver gains, **S** yields the actual measured power. Sometimes in comparing powers in 2 channels (for example, Z_{dr} and LDR), we need to know the relative gains of the 2 channels. However, in many calculations, the relative gains cancel out, and in these cases the algorithms are implemented assuming all the gains are equal to 1.

In the $\mathbf{R_{hv}}$ term, the noise variable is not present because the noise between the horizontal receiver and vertical receiver is random, having a normalized coherency of 0 with an infinite number of samples. A finite number of samples needs to be used, typically between 30 ... 60, in weather radar signal processing. However, due to the noise coherency going to 0, the noise variance also becomes smaller, allowing us to lower the detection thresholds, while having same false alarm rates as the traditional $\mathbf{R_{hh}}$ term. Lowering the detection threshold increases the apparent sensitivity when inserting the $\mathbf{R_{hv}}$ term in the radar equation for reflectivity.

In the following algorithm descriptions, we use the notation common in the literature, for example:

$$R_0^{hh} = \frac{1}{M}\sum_{n=1}^{M-1} s'^{n}_{hh} \times s'^{n}_{hh} = \left(\left|s'_{hh}\right|^2\right) = S_{hh}$$

$$R_0^{vv} = \frac{1}{M}\sum_{n=1}^{M-1} s'^{n}_{vv} \times s'^{n}_{vv} = \left(\left|s'_{vv}\right|^2\right) = S_{vv}$$

$$R_0^{hv} = \frac{1}{M}\sum_{n=1}^{M-1} s'^{n}_{hh} \times s'^{n}_{vv} = \left(\left|s'_{hv}\right|^2\right) = S_{hv}$$

9.1.4.2.1 Noise bias in channel powers, correlations, and optional correction

The average noise powers N_v and N_h are assumed to be receiver noise only.

These bias the autocorrelations at lag 0, that is, the channel power measurements, and must be subtracted. Autocorrelations at lags 1 and 2 are not biased by noise, while cross-channel correlations are biased by finite noise. Cross-channel phases are smeared by finite noise while unbiased, assuming that the noises in the 2 channels are independent (a good assumption).

The channel noise values are measured directly by RVP10 during noise sampling. Using these measurements for correcting the effects of finite noise is configured in the TTY setups. The choices are made with the **mp** non-volatile setup settings:

- **Polarimetric Power Params > NoiseCorrected**:**YES**/**NO**
  If **YES**, the Z_{dr} and LDR are computed from channel powers, subtracted for noise levels. The sensitivity is enhanced to observe weather effects in weak echoes (SNR << 10 dB). At the SNR low limit, Z_{dr} approaches 0 dB as soon as noise levels are properly set.
  If **NO**, Z_{dr} values in weak signal regions are biased, and approach the ratio of channel noise levels.
  A finite noise level may set a limit for LDR observations.
- **Polarimetric Correlations > NoiseCorrected**:**YES**/**NO**
  If **NO**, RhoHV, RhoH, and RhoV approach 0 dB in weak echoes
  If **YES**, they approach unity.
  Phase measurements: PhiDP, K_{dp}, PhiH, and PhiV are not affected by these settings.

9.1.4.2.2 Clutter Filtering

The polarization variables are computed from data filtered for clutter. The settings are the same, and are user configurable for standard moments.

The filter is applied identically for each polarimetric channel.

9.1.5 Case 1: Fixed Transmit: Dual-Channel Receiver

9.1.5.1 Case 1: Input Receiver Samples

In fixed mode, the radar is configured (either permanently or by means of a switch) to transmit either vertical or horizontal polarization with dualchannel reception of both the co- and cross-channel polarizations. For example, the radar can transmit horizontal and receive both horizontal (co) and vertical (cross) polarizations.

The received samples in the 2 transmit cases are as follows:

Transmit Horizontal

$$\left[S_{hh}^{1}:S_{vh}^{1}\right]\left[S_{hh}^{2}:S_{vh}^{2}\right]\left[S_{hh}^{3}:S_{vh}^{3}\right]\cdots\left[S_{hh}^{M}:S_{vh}^{M}\right]$$

Transmit Vertical

$$\left[S_{vv}^{1}:S_{hv}^{1}\right]\left[S_{vv}^{2}:S_{hv}^{2}\right]\left[S_{vv}^{3}:S_{hv}^{3}\right]\cdots\left[S_{vv}^{M}:S_{hv}^{M}\right]$$

9.1.5.2 Case 1: Calculation of Polarization Measurands

The processing in this mode is done by pulse pair algorithm. You may select a clutter filter.

Table 52 Polarization Measurands for the Horizontal and Vertical Transmit Cases

Transmit Horizontal	or	Transmit Vertical
$LDRH = 10LOG\left[\frac{S_{vh}}{S_{hh}}\right]$	or	$LDRV = 10LOG\left[\frac{S_{hv}}{S_{vv}}\right]$
$= 10LOG\left[\frac{<\lvert S_{vh}\rvert^2> - N_v}{<\lvert S_{hh}\rvert^2> - N_h}\right] - XDR$	or	$= 10LOG\left[\frac{<\lvert S_{hv}\rvert^2> - N_h}{<\lvert S_{vv}\rvert^2> - N_v}\right] + XDR$
$RHOH = \lvert\rho_h\rvert$	or	$RHOV = \lvert\rho_v\rvert$
$PHIH = \arg\lvert\rho_h\rvert$	or	$PHIV = \arg\lvert\rho_v\rvert$

The H and V average channel powers are computed with optional noise correction as follows.

Table 53 H and V Average Channel Powers

	Transmit Horizontal	or	Transmit Vertical
Co-	$g_h^r g_h^t S_{hh} = <\lvert S_{hh}\rvert^2> - N_h$	or	$g_v^r g_v^t S_{vv} = <\lvert S_{vv}\rvert^2> - N_v$
Cross-	$g_v^r g_h^t S_{vh} = <\lvert S_{vh}\rvert^2> - N_h$	or	$g_h^r g_v^t S_{vh} = <\lvert S_{hv}\rvert^2> - N_h$

The complex covariance ρ (used above) is as follows:

Table 54 Complex Covariance ρ

Transmit Horizontal	or	Transmit Vertical
$\rho_h = \frac{< S_{vh} S_{hh}^* >}{\sqrt{S_{vh} S_{hh}}}$	or	$\rho_h = \frac{< S_{vh} S_{hh}^* >}{\sqrt{S_{vh} S_{hh}}}$

Fortunately, the algorithms do not require us to know each gain term. They cancel in the calculation of `r`, so they are taken as `=1` in the implementation. However, the differential receiver gain LDR Offset must be known from the calibration to calculate LDR:

dB value is: $XDR = 10LOGxdr$ where the linear value is $xdr \frac{g_v^r}{g_h^r}$.

9.1.6 Case 2: Simultaneous Dual Transmit and Receive (STAR)

9.1.6.1 Case 2: Input Receiver Samples

In this mode, there is simultaneous transmit and receive of both vertical and horizontal polarization. For each pulse, there is a measurement of the amplitude in each channel, for example:

$$\left[S_{hh}^1 : S_{vv}^1\right]\left[S_{hh}^2 : S_{vv}^2\right]\left[S_{hh}^3 : S_{vv}^3\right]\cdots\left[S_{hh}^M : S_{vv}^M\right]$$

We assume that the `M` samples are collected for processing.

Even though there is cross-polarized radiation received in each channel, this cross-polar contribution can be neglected, since the co-polarized received signal is much stronger.

9.1.6.2 Case 2 Calculation of polarization measurands

The processing is done by pulse pair mode. However, both FFT and random phase processing can be performed only if Z_{dr} and standard moments are requested for output. In any mode, you may select a clutter filter.

RVP10 calculates the polarization parameters as follows:

$$ZDR = 10LOG\left[\frac{S_{hh}}{S_{vv}}\right]$$

$$ZDR = 10LOG\left[\frac{< |S_{hh}|^2 > - N_h}{< |S_{vv}|^2 > - N_v}\right] + GDR$$

$RHOHV = |\rho_{hv}(0)|$

$PHIDP = \arg[\rho_{hv}(0)]$

Where the following definitions are used:

Transmit horizontal	or	Transmit vertical
$g_h^r g_h^t S_{hh} = <|S_{hh}|^2> - N_h$	or	$g_v^r g_v^t S_{vv} = <|S_{vv}|^2> - N_v$

The noise powers within the 2 channels are denoted as N_h and N_v. The noise corrections to S_{hh} and S_{vv} are optionally configured in the TTY setups. The Z_{dr} Offset is the total (transmit and receive) differential channel gain. It must be calibrated for the system as follows:

dB value is: $ZDR\ Offset = 10LOGoffset$ where the linear value is $offset = \frac{g_v^r g_v^t}{g_h^r g_h^t}$.

The correlation function is computed as follows:

$$\rho_{hv}(0) = \frac{< S_{vh} S_{hh}^* >}{\sqrt{S_{hh} S_{vv}}}$$

The gain terms cancel in the calculation of ρ, so in the implementation they are assumed to be =1.

9.1.7 Case 3: Alternating H/V Transmit: Single Receiver

9.1.7.1 Case 3: Input Receiver Samples

This is the traditional Z_{dr} radar with a high-power, fast switch that alternates between horizontal and vertical on each pulse. The switch is made just prior to the transmit pulse, so that the transmitter radiates, and then receives at a single polarization for each pulse.

The samples are as follows:

$$\left[S_{hh}^1\right]\left[S_{vv}^2\right]\left[S_{hh}^3\right]\cdots\left[S_{vv}^{M+1}\right]$$

For this discussion, we assume that there are `M+1` total samples with `M/2` horizontal pulses indexed by (`1`, `2`, `3` ... `M−1`) and `M/2+1` vertical pulses indexed at (`2`, `4`, `6` ... `M`).

The processor does not assume that the first pulse in a sequence is horizontal.

9.1.7.2 Case 3: Calculation of polarization measurands

The processing is done in pulse pairs with an optional clutter filter.

RVP10 calculates the following:

$$ZDR = 10LOG\left[\frac{S_{hh}}{S_{vv}}\right]$$

$$ZDR = 10LOG\left[\frac{< |S_{hh}|^2 > - N_h}{< |S_{vv}|^2 > - N_v}\right] + GDR$$

$$PHIDP = \frac{1}{2}\arg\left[R_a R_b^*\right]$$

$$RHOHV = \frac{|\rho_{hv}(T_s)|}{[\rho_{hv} 2T_s]^{0.25}}$$

Using the following definitions:

$$\left|\rho_{hv}(T_s)\right| = \frac{|R_a| + |R_b|}{2\sqrt{S_{hh} S_{vv}}}$$

$$\rho(2T_s) = \frac{\Sigma_{n=1}^{\frac{M}{2}-1}\left(S_{hh}^*[2n-1]S_{hh}[2n+1] + S_{vv}^*[2n]S_{vv}[2n+2]\right)}{\left(\frac{M}{2}-1\right)(S_{hh} + S_{vv})}$$

$$R_a = \frac{M}{2}\sum_{n=1}^{\frac{M}{2}} S_{hh}^{2n-1*} S_{vv}^{2n}$$

$$R_b = \frac{1}{\frac{M}{2}}\sum_{n=1}^{\frac{M}{2}} S_{vv}^{2n*} S_{hh}^{2n+1}$$

The calculation of the channel powers ($< |S_{hh}|^2 >$ and $< |S_{vv}|^2 >$) is done using alternating pulses.

In the calculation of R_b, RVP10 uses the extra `M+1` sample. The gain terms cancel in the calculation of r, so in the implementation they are assumed to be 1.

9.1.8 Case 4: Alternating H/V Transmit: Dual Receiver

9.1.8.1 Case 4: Input Receiver Samples

This is the most comprehensive case of polarization operation, since it permits calculation of all of the polarization measurands. In this case, the transmitter alternates pulse-to-pulse between horizontal and vertical polarization, and the dual-channel receiver provides measurement of both the co- and the cross-polarized return as follows:

$$\left[S_{hh}^{1}:S_{vh}^{1}\right]\left[S_{vv}^{2}:S_{hv}^{2}\right]\left[S_{hh}^{3}:S_{hv}^{3}\right]\cdots\left[S_{vv}^{M+1}:S_{hv}^{M+1}\right]$$

We assume that **M+1** samples are collected for processing (an extra sample is required for the calculation **Rb** as in Case 3: Alternating H/V Transmit: Single Receiver (page 217)).

9.1.8.2 Case 4: Calculation of polarization measurands

RVP10 calculates the following:

- **Co-polar channel measurements**
  Z_{dr}, PhiDP, RhoHV
  Identical to alternating case, see Case 3: Alternating H/V Transmit: Single Receiver (page 217).
- **Cross-polar channel measurements**
  LDRH, RhoH, PhiH
  LDRV, RhoV, PhiV
  Identical to fixed case, see Case 1: Fixed Transmit: Dual-Channel Receiver (page 214).

The co-polar channel measurements are the same as for the alternating single-receiver case. The cross-polar measurements are calculated using fixed case algorithms, except they are calculated for both H and V polarizations.

9.1.9 Standard moment calculations (T, Z, V, W)

Standard moments are available for each polarization case. Since there can be up to 4 channels of time series input, there are several choices for computing the standard moments. For example, in the STAR mode (see Case 2: Simultaneous Dual Transmit and Receive (STAR) (page 216)), the standard moments are computed from:

- S_{hh} samples
- S_{vv} samples
- Average of the results from the S_{hh} and S_{vv} samples

The reflectivity moment can be computed from the S_{hh}, S_{vv}, and S_{hv} samples simultaneously in all 3 choices.

In Case 3: Alternating H/V Transmit: Single Receiver (page 217), the case is handled by averaging the individual channel correlations, and then using the average correlations in the standard moment processing. The averaging must take into account the differential gain of the channels.

The method to use is selected in the setup. There are 4 questions in the **mp** section:

```
T/Z/V/W computed from: H-Xmt:YES V-Xmt:YES

T/Z/V/W computed from: Co-Rcv:YES Cx-Rcv:NO
```

Given a choice between vertical and horizontal transmit, the first 2 questions specify which transmit polarization should be used.

When choosing:

- **`H-Xmt:YES V-Xmt:NO`** logic, V and W are computed from the H-channel co-receiver
- **`H-Xmt:NO V-Xmt:YES`** logic, V and W are computed from the V-channel co-receiver
- **`H-Xmt:YES V-Xmt:YES`** logic, V and W are computed from the averaged correlations from both receivers

RVP10 can produce ZH, ZV, and ZHV with any logical selection.

For the fixed H or V case, where there is only one transmit polarization, this question does not apply. The processor uses samples for the polarization that is transmitted.

Given a choice between using the co- or cross-polar receivers, the second 2 questions are define which one should be used. This question applies only to systems that can measure LDR, (fixed or alternating transmit, dual-channel receiver systems).

The tables in Reflectivity calibration parameters (page 221) summarize the standard moment calculations for each mode and how to configure the four TTY setup responses.

These are the only supported modes. Some combinations of responses are unsupported. For example, it is not supported to answer both **`Co-Rcv: NO`** and **`Cx-Rcv: NO`**.

Each table identifies the transmitter/receiver case and what samples are available. The notation **HH** signifies that the s_{hh} samples are available. The tables use "—" to indicate that either a **YES** or **NO** response causes the same result, RVP10 does not care what response is made. In cases where averaging is performed, the type of weighting used is indicated.

9.1.9.1 Model for standard moment autocorrelations

The model for the moment autocorrelation calculations is as follows (using $\mathbf{R_0}$ as an example):

$$R_0^{hh} = g_h^r g_h^t S'_{hh} + N_h$$

$R_0^{vv} = g_t^r g_v^t S'_{vv} + N_v$

R_0^{hh}, R_0^{vh}, R_0^{vv}, R_0^{hv}
: The autocorrelations of the samples at lag 0.

S_{hh}, S_{vh}, S_{vv}, S_{hv}
: The average power returned from the scatterers.

g^r_h, g^r_v
: Receiver gains for horizontal and vertical receive.

g^t_h, g^t_v
: Transmitter gains for horizontal and vertical transmit.

N_h, N_v
: Measured noise power of the samples.

The power that is measured in a channel has comprises:

- Backscattered power from the targets that is effected by the transmitter and receiver channel gains
- Receiver noise, which is measured by RVP10 during noise sampling

In **R1** and **R2** autocorrelations, the model is similar, except there is no noise bias.

9.1.9.2 Reflectivity calibration parameters

For dBZ calculations, the dBZ_o value is required as a calibration constant.

Depending on the polarization case and the technique selected for standard moment calculation, it may also be required to have Z_{dr} Offset and LDR Offset:

- Z_{dr} Offset
  The ratio of the total gains (transmit/receive) of the 2 co-receive channels.
- LDR Offset
  The ratio of the receiver gains in a dual receiver system.
  This is not required for Case 2: Simultaneous Dual Transmit and Receive (STAR) (page 216) or Case 3: Alternating H/V Transmit: Single Receiver (page 217).

RVP10 supports a single calibration reflectivity **dBZ_0**. In most cases, it is assumed that the **dBZ_0** is for the horizontal co-receive (**HH**) channel. The exception is for fixed vertical polarization, in which the algorithm assumes that the calibration is for the vertical co-receive (**VV**) channel. LDR Offset and Z_{dr} Offset are also downloaded and used to adjust the **dBZ_0**, as required, depending on the user's selection for the standard moments. For example, in STAR mode, if the user selects dBZ to be computed from the VV channel, the **dBZ_0** for the HH and a Z_{dr} Offset adjustment are used to calculate the dBZ in the VV channel.

Table 55 Case 1H: Fixed Horizontal Transmit, Dual Channel Receive- (HH, VH)

dBZ_0 from HH Channel	**TTY Setup Question Responses**			
Calculate T, Z, V, W from:	**HXmt**	**VXmt**	**CoRcv**	**CxRcv**
HH (co) (Recommended)	—	—	YES	NO
VH (LDR Offset^{-1} weighting)	—	—	NO	YES
HH+VH (xdr^{-1} weighting)	—	—	YES	YES

HH Channel (Co-Pol)

HH channel (co-pol) is the recommended channel for linear polarization because, for linear polarization, the co-polar channel has the strongest signal. Processing is identical to a conventional radar.

VH Channel (Cross-Pol)

VV channel (cross-pol) is used for circular or elliptic transmit polarization. Since the algorithm assumes that dBZ_0 is from the co-polar channel, `xdr` is used to adjust the autocorrelations as follows:

$$T_0 = xdr^{-1} T_0^{vh}$$

$$R_{(0)} = xdr^{-1} R_0^{vh}$$

$$R_1 = xdr^{-1} R_1^{vh}$$

$$R_2 = xdr^{-1} R_2^{vh}$$

$$N = xdr^{-1} N_v$$

These adjusted autocorrelations are used as input to the standard moment processing for a conventional radar. For example, in reflectivity processing, the radar equation can be written as follows:

$$Z^{vh} = CS_{vh} r^2 = \left[\frac{Cr^2 N_v}{g_v^r g_h^t}\right]\left[\frac{r^2}{r_0^2}\right]\left[\frac{T_0^{vh} - N_v}{N_v}\right]$$

where $T_0^{vh} = g_v^r g_h^t S_{vh} - N_v$

$$= \left[\frac{Cr_0^2 N_h}{g_h^r g_h^t}\right]\left[\frac{r^2}{r_0^2}\right]\left[\frac{T_0^{vh} - N_v}{N_h}\right]$$

The third term is `1/XDR` and is written as follows:

$$Z^{vh} = \left[\frac{Cr_0^2 N_h}{g_h^r g_h^t}\right]\left[\frac{r^2}{r_0^2}\right]\left[\frac{xdr^{-1}T_0^{vh} - xdr^{-1}N_v}{N_h}\right]$$

The first term is the $\mathbf{dBZ_0}$ for the HH channel. We can use the $\mathbf{dBZ_0}$ for the HH channel to calibrate the cross-channel, if we first adjust the cross-channel noise and power by `1/xdr`, and then normalize by $\mathbf{N_h}$. The reflectivity calculation assumes that the calibrated `xdr` value compensates for any differences in the radar constant between the 2 channels. We do not need separate radar constants for the 2 channels.

HH+VH Channels

HH+VH channels are used for elliptic transmit polarizations that give comparable return signal in both the co- and cross-channels. The approach is to obtain average autocorrelation functions as follows:

$$T_0 = \frac{T_0^{hh} + xdr^{-1}T_0^{vh}}{2}$$

$$R_0 = \frac{R_0^{hh} + xdr^{-1}R_0^{vh}}{2}$$

$$R_1 = \frac{R_1^{hh} + xdr^{-1}R_1^{vh}}{2}$$

$$R_2 = \frac{R_2^{hh} + xdr^{-1}R_2^{vh}}{2}$$

$$N = \frac{N_h + xdr^{-1}N_v}{2}$$

These adjusted autocorrelations are used as input to the standard moment processing for calibration with respect to the HH channel.

Table 56 Case 1V: Fixed Vertical Transmit and Dual Channel Receive- (VV, HV)

dBZo from VV Channel	TTY Setup Question Responses			
Calculate T, Z, V, W from:	`HXmt`	`VXmt`	`CoRcv`	`CxRcv`
VV (co)	—	—	`YES`	`NO`
HV (xdr weighting)	—	—	`NO`	`YES`
VV+HV (xdr weighting)	—	—	`YES`	`YES`

This is the only case for which the calibration constant $\mathbf{dBZ_0}$ for the VV channel should be downloaded to the signal processor.

VV Channel (Co-Pol)

VV channel (co-pol) is the recommended channel for the case of linear polarization. The reason is that for linear polarization, the co-polar channel has the strongest signal. Processing is identical to a conventional radar.

HV Channel (Cross-Pol)

HV channel (cross-pol) is used for circular or elliptic transmit polarization when most of the return is in the cross-pol channel. Since the algorithm assumes that $\mathbf{dBZ_0}$ is from the co-polar channel, xdr is used to adjust the autocorrelations as follows:

$$T_0 = xdrT_0^{vh}$$

$$R_0 = xdrR_0^{hv}$$

$$R_1 = xdrR_1^{hv}$$

$$R_2 = xdrR_2^{hv}$$

$$N = xdrN_h$$

These adjusted autocorrelations are used as input to the standard moment processing with $\mathbf{dBZ_0}$ calibrated with respect to the VV channel.

VV+HV Channels

VV+HV channels are used for elliptic transmit polarizations that give comparable return signal in both the co- and cross-channels. The approach is to obtain average autocorrelation functions as follows:

$$T_0 = \frac{T_0^{vv} + xdrT_0^{hv}}{2}$$

$$R_0 = \frac{R_0^{vv} + xdrR_0^{hv}}{2}$$

$$R_1 = \frac{R_1^{vv} + xdrR_1^{hv}}{2}$$

$$R_2 = \frac{R_2^{vv} + xdrR_2^{hv}}{2}$$

$$N = \frac{N_v + xdrN_h}{2}$$

These adjusted autocorrelations are used as input to the standard moment processing algorithms with $\mathbf{dBZ_o}$ calibrated with respect to the VV channel.

Table 57 Case 2: Simultaneous Transmit and Receive- STAR (HH, VV) & Case 3: Alternating Transmit Single-Channel Receive (HH, VV)

dBZo from HH Channel	TTY Setup Question Responses			
Calculate T, Z, V, W from:	H-Xmt	V-Xmt	Co-Rcv	Cx-Rcv
HH	YES	NO	—	—
VV (Z_{dr} Offset^{-1} weighting)	NO	YES	—	—
HH+VV (gdr^{-1} weighting)	YES	YES	—	—

A fundamental difference between these 2 cases is that for all standard moment processing choices, the STAR case has double the number of samples compared to the single-channel alternating case. The processing is otherwise identical.

HH Channel

Since the HH channel is directly calibrated, this is the recommended choice. Processing is identical to a conventional radar.

VV Channel

In VV channel, GDR is used to adjust the autocorrelations as follows:

$$T_0 = gdr^{-1} T_0^{vv}$$

$$R_0 = gdr^{-1} R_0^{vv}$$

$$R_1 = gdr^{-1} R_1^{vv}$$

$$R_2 = gdr^{-1} R_2^{vv}$$

$$N = gdr^{-1} N_v$$

These adjusted autocorrelations are used as input to the standard moment processing algorithms with $\mathbf{dBZ_0}$ calibrated with respect to the HH channel.

HH+VV Channels

HH+VV channels approach gives the benefit of doubling the number of samples used for the reflectivity calculation as follow:

$$T_0 = \frac{T_0^{hh} + gdr^{-1} T_0^{vv}}{2}$$

$$R_0 = \frac{R_0^{hh} + gdr^{-1} R_0^{vv}}{2}$$

$$R_1 = \frac{R_1^{hh} + gdr^{-1} R_1^{vv}}{2}$$

$$R_2 = \frac{R_2^{hh} + gdr^{-1} R_2^{vv}}{2}$$

$$N = \frac{N_h + gdr^{-1} N_v}{2}$$

These adjusted autocorrelations are used as input to the standard moment processing algorithms with $\mathbf{dBZ_0}$ calibrated with respect to the HH channel.

Table 58 Case 4: Alternating Dual-Channel (HH, VH, VV, HV)

dBZo from HH Channel	TTY Setup Question Responses			
Calculate T, Z, V, W from:	HXmt	VXmt	CoRcv	CxRcv
HH	YES	NO	YES	NO
VH (xdr[1] weighting)	YES	NO	NO	YES
VV (gdr[-1] weighting)	NO	YES	YES	NO
HV (xdr/gdr weighting)	NO	YES	NO	YES
HH+VV (gdr[-1] weighting)	YES	YES	YES	NO
HV+VH (xdr & gdr weighting)	YES	YES	NO	YES

HH Channel

Since the HH channel is directly calibrated, this is the recommended choice. Processing is identical to a conventional radar.

VH Channel

Processing is identical to Case 1: Fixed Transmit: Dual-Channel Receiver (page 214).

VV Channel

Processing is identical to Case 2: Simultaneous Dual Transmit and Receive (STAR) (page 216)) and Case 3: Alternating H/V Transmit: Single Receiver (page 217).

HV Channel

The weighting in HV channel uses both `xdr` and `gdr` as follows:

$$T_0 = \frac{xdr}{gdr} T_0^{hv}$$

$$R_0 = \frac{xdr}{gdr} R_0^{hv}$$

$$R_1 = \frac{xdr}{gdr} R_1^{hv}$$

$$R_2 = \frac{xdr}{gdr} R_2^{hv}$$

$$N = \frac{xdr}{gdr} N_h$$

These adjusted autocorrelations are used as input to the standard moment processing algorithms with $\mathbf{dBZ_0}$ calibrated with respect to the HH channel.

HH+VV Channels

Processing is identical to Case 2: Simultaneous Dual Transmit and Receive (STAR) (page 216)) and Case 3: Alternating H/V Transmit: Single Receiver (page 217).

HV+VH Channels

The weighting in HV+VH processing must correct for both transmitter and receiver effects in order to use the HH channel $\mathbf{dBZ_0}$ as follows:

$$T_0 = \frac{\frac{xdr}{gdr} T_0^{hv} + xdr^{-1} T_0^{vh}}{2}$$

$$R_0 = \frac{\frac{xdr}{gdr} R_0^{hv} + xdr^{-1} R_0^{vh}}{2}$$

$$R_1 = \frac{\frac{xdr}{gdr} R_1^{hv} + xdr^{-1} R_1^{vh}}{2}$$

$$R_2 = \frac{\frac{xdr}{gdr} R_2^{hv} + xdr^{-1} R_2^{vh}}{2}$$

$$N = \frac{\frac{xdr}{gdr} N_h + xdr^{-1} N_v}{2}$$

These adjusted autocorrelations are used as input to the standard moment processing algorithms with $\mathbf{dBZ_0}$ calibrated with respect to the HH channel.

Suppose that we want to compute the average of the reflectivities for the VH and HV channels. An example this weighted averaging is as follows:

$$Z^{hv+vh} = Cr^2 \frac{S_{hv} + S_{vh}}{2}$$

$$= Cr^2 \frac{\frac{T_0^{hv} - N_h}{g_h^r + g_h^t} + \frac{T_0^{vh} - N_v}{g_v^r + g_h^t}}{2} = \frac{Cr^2}{g_h^r g_h^t} \frac{\left(T_0^{hv} - N_h\right)\frac{g_h^t}{g_v^t} + \left(T_0^{vh} - N_v\right)\frac{g_h^r}{g_v^r}}{2}$$

but since $xdr = \frac{g_h^r}{g_v^r}$ and $gdr = \frac{g_v^r g_v^t}{g_h^r g_h^t}$

$$Z^{vh+hv} = \frac{Cr^2}{g_h^r g_h^t}\left[\frac{\frac{xdr}{gdr}T_0^{hv} + xdr^{-1}T_0^{vh}}{2} - \frac{\frac{xdr}{gdr}N_h + xdr^{-1}N_v}{2}\right]$$

$$Z^{vh+hv} = \frac{Cr^2}{g_h^r g_h^t}[T_0 - N] = \left[\frac{Cr^2 N_h}{g_h^r g_h^t}\right]\left[\frac{r^2}{r_0^2}\right]\left[\frac{T_0 - N}{N_h}\right]$$

The first term in brackets is precisely $\mathbf{dBZ_0}$ for the HH channel. If we average the correlations using the appropriate **gdr** and **xdr** weighting, the average reflectivity is obtained by using conventional processing with the HH channel $\mathbf{dBZ_0}$.

9.2 Attenuation correction of Z

9.2.1 Attenuation correction of Z overview

In dual-polarization radars, the measurements consist of measuring the amplitude and phase of the received waveforms in the two separate channels. PhiDP is the phase difference between the two polarization channels at each range bin.

The PhiDP quantity primarily comes from the propagational effects as the RF energy passes through some medium to the measurement point and back again to the transmitter/receiver. The PhiDP measurement tends to be a noisy, unstable quantity, due to the slightly different backscattering phase shifts from the scatterers at the range point and statistical measurement errors.

The unambiguous range of PhiDP is 360° in the simultaneous operation mode. However, phase changes of more than 360° are common, especially with smaller wavelength radars, and cause PhiDP to be phase wrapped.

PhiDP is directly related to the amount of liquid water content the transmitted energy passes through to get to the scatterers at a specific range bin. Therefore, PhiDP has advantages when used to estimate attenuation caused by liquid water. In dual polarization radars, the range derivative, or specific differential phase (K_{dp}) is also a key data type to indicate rainfall rates.

Before to using PhiDP for any of these applications, it must be conditioned and unfolded. Here, conditioning means to smooth out the inherent noisy properties of the data, yet be able to capture and maintain the range and amplitude resolution common within intense rain cells. See PHiDP Unfolding and Conditioning (page 229).

The conditioned PhiDP is fed to:

- A dual-polarization-based attenuation correction algorithm. See Dual-polarimetric attenuation correction (page 231).
- An adaptive K_{dp} algorithm. See Adaptive K_{dp}Moment Estimation (page 235).

After these are applied, the IRIS and RDA software makes optimal use of the dual-channel measurements, creating data types for the following applications:

- Hydrometeor particle identification
- Lightning hazard potential forecasting
- Detection of convection and stratiform rain
- Highway snow removal
- Airport terminal operation
- Rain/snow line demarcation
- Melting height detection
- Weather modification for hail mitigation
- Insurance industry claims verification
- Military detection of chaff
- Data quality improvement by elimination of non-meteorological targets
- Improved precipitation forecasting
- Hydrological modeling

Dual-polarimetric attenuation correction is a collection of algorithms, which may be performed in real-time, in the RVP10 signal processors, or in the IRIS application software. See Dual-polarimetric attenuation correction (page 231).

9.2.2 PHiDP Unfolding and Conditioning

When computed for each range bin, PhiDP is inherently noisy, especially in volume samples not having meteorological scatterers present.

PhiDP can typically have values greater than 360° of phase difference, causing folding in the valid data ranges used in IRIS/RDA. Therefore, the first processing steps are to smooth and unfold the PhiDP data.

The following methodologies are available for unfolding and conditioning PhiDP data:

- Least squares fit (LSQ). [1)]
- Cubic spline fit. [2)]

The cubic spline fit is the recommended process. The methodology to be executed is a user-selectable option in IRIS and RDA processing.

---

1) *Panov,S., R. Keranen, and V. Chandrasekar, 2008: Assessment of the Polarimetric Attenuation Correction Implementation in the RVP8 Signal Processor. 5th European Conference on Radar in Meteorology and Hydrology, Helsinki, Finland.*

2) *Chandrasekar,V., and Y. Wang, 2009: Adaptive Specific Differential Phase at Dual-Polarization Radar International Application Published under the Patent Cooperation Treaty. WO 2009/114868 A1.*

9.2.2.1 Least Squares Fit

In the Least Squares Fit (LSQ) model, the PhiDP unfolding procedure is combined with the conditioning/filtering of PhiDP.

The measured PhiDP are stored as 32-bit unsigned integers. The PhiDP ambiguity interval is mapped linearly into the full range of unsigned integers.

At each bin, an LSQ, weighted with FIR filter coefficients, is performed to compute "conditioned" PhiDP. This LSQ is performed 3 times at each bin, each having a unique phase shift being tracked. One phase shift, ϕ, is constantly adjusted to keep the current PhiDP in the middle of the ambiguity interval. For the other 2 LSQs, the phases are shifted by the ambiguity interval, ϕ ± ⅔. The LSQ with the lowest signal-to-noise ratio is assumed to be the unfolded quantity.

The following figure shows the input data and outcomes of this LSQ technique.

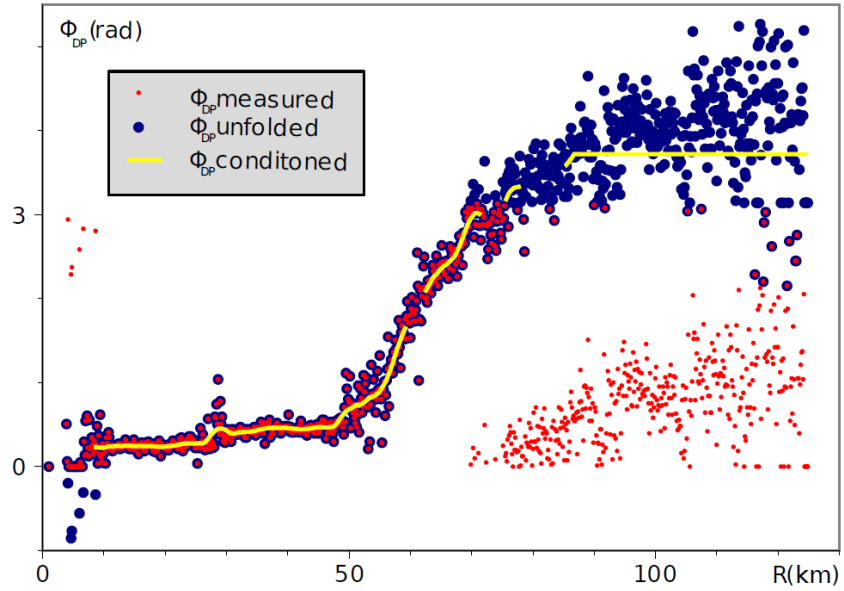


Figure 38 Unfolding and Conditioning of Bin to Bin PhiDP

9.2.2.2 Cubic spline fit

LSQ is a practical phase unfolding approach. However, it creates a test condition to check for phase unfolding when at times the test may fail.

The cubic spline fit approach transforms the PhiDP between the 2 receive channels to the complex domain, creating an Angular PhiDP vector. This is a similar concept used when creating `I`,`Q` for signal processing. The advantage in using Angular PhiDP is that the data remains continuous in the complex domain, where as the traditional PhiDP moments in the real domain phase wraps with abrupt jumps. To compute the Angular PhiDP moment:

$\vartheta(r) = e^{j\psi dp^{(r)}}$ for STAR mode dual-polarization

$\vartheta(r) = e^{j2\psi dp^{(r)}}$ for alternating mode of dual-polarization

Where $\vartheta(r)$ is the Angular PhiDP at some range and ψ_{dp} is the total differential phase.

For information on conditioning and smoothing of Angular PhiDP, see Adaptive K_{dp}Moment Estimation (page 235).

9.2.3 Dual-polarimetric attenuation correction

Dual-polarimetric attenuation correction is a collection of algorithms, using the unfolded and conditioned PhiDP.

These algorithms may be performed in real-time within the RVP10 signal processors, or within the IRIS application software.

Table 59 Performing Attenuation Correction with RVP or IRIS

RVP or IRIS	Description	Advantages and Disadvantages
RVP10 DP Attenuation Correction Processing	Ray data are processed in the RDA, and the bin level moments are output in real-time for each range bin.	The RDA software can apply immediate corrections with the signal processor being available on real-time data output. This becomes important in mission critical now-casting within the 0 ... 1 hr time frame, where the latest observed information is needed for precise decision making. With RDA, all moments are available in every bin. When performing the function in IRIS, certain moments within the bins are thresholded. This approach is well suited for applications where IRIS software is not active, since the particle type can be displayed directly by the customer's display software.

RVP or IRIS	Description	Advantages and Disadvantages
IRIS DP Attenuation Correction Processing	Dual polarization data comes from RVP10, or from a third party processor, and are passed to an IRIS/Radar or an IRIS/ Analysis system that is enabled with the dual polarization feature license. The bin level algorithms produce ingest data, which are formatted as RAW products for archival, or for rerouting in the network. The correction algorithm can also be applied to historical data.	IRIS processing has an advantage since the original Z and Z_{dr} moments are uncorrected for attenuation; IRIS makes new data types. IRIS processing allows for creating the corrected moments with archived data. This allows you to compare the results of the correction. Disadvantages to using IRIS are the time delay to have the corrected moments and that data is likely already been thresholded causing some range bins to have missing data.

Signal attenuation in precipitation is a well known problem for weather radars using smaller wavelengths, like 3 ... 5 cm (1.18 ... 1.97 in) in the X- and C-band spectrums. The attenuation causes a decrease in the Z and Z_{dr} measurements, making their use in quantitative analysis limited. The attenuation effects also impact the outcomes of the 'fuzzy logic' set of algorithms like HydroClass. It is important to estimate the amount of attenuation, and make corrections to the Z and Z_{dr} data types prior to processing algorithms. These corrections ultimately improve the ability to accurately estimate rainfall, differentiate hydrometeors, and assess data quality metrics.

In case of the dual polarization radar, the differential propagation phase moment PhiDP, provides an excellent method to estimate attenuation of Z and Z_{dr}, as the change in PhiDP is directly related to the liquid water content along the propagation path. PhiDP is not affected by attenuation or calibration errors, but allows large, accurate corrections.

The traditional single polarization attenuation correction is based on the following relationship:

Equation 1

$$A(r) = a[Z(r)]^b$$

where $A(r)$ is the specific attenuation and $Z(r)$ is the intrinsic reflectivity at any particular range gate. This particular equation is implicit, where an unknown value $A(r)$, is expressed through another unknown value $Z(r)$. The expression to solve for total two-way attenuation is:

Equation 2

$$dBZc = dBZ + 2C\Delta r\sum Z^E$$

where the $2C\Delta r\sum Z^E$ term estimates the total two-way attenuation to the range bin, based on the accumulated reflectivity measurement. C and E are constants in this estimation. This equation becomes unstable very quickly and must be highly constrained in order that the corrected reflectivity does not end up having less value than the original.

With a dual-polarization radar, an independent estimate of the specific attenuation is possible:

Equation 3

$$A(r) = \alpha K_{dp}^{(r)}$$

In practice, the PhiDP(r) measurements to product K_{dp} have much lower accuracy than Z(r), and deviate significantly from the pure "propagation" component. However, the estimate of total cumulative attenuation along the radial is quite accurate and can be used as a constraint for the range specific attenuation corrections.

The IRIS/RDA implementation of dual-polarimetric attenuation correction is illustrated in the following figure.

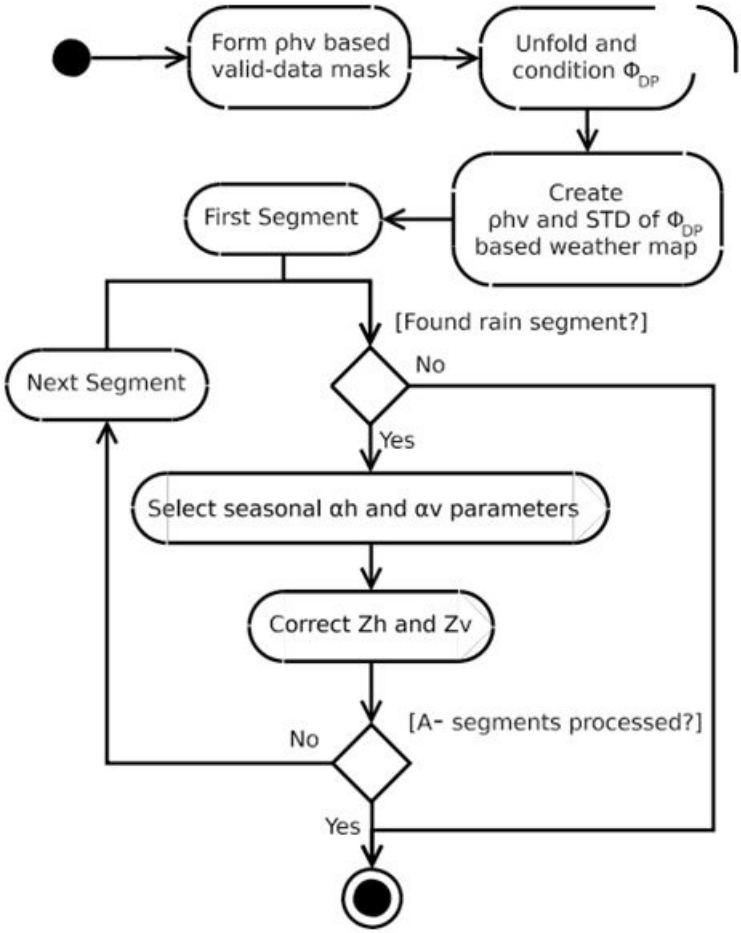


Figure 39 Dual-polarimetric attenuation correction processing stages

9.2.3.1 Weather classification algorithm

The first processing step is to identify segments along the radial where attenuation occurs. The radial is broken into one or more segments where each segment is either flagged as "liquid rain" or "no weather". The liquid rain portions of the segment are used to calculate the constrained PhiDP. This step is a hydrometeor classifier.

The simple weather classification ensures that changes in PhiDP from non-meteorological targets like ground clutter, point targets, refractivity gradients, and insects do not introduce errors in the attenuation estimate. Reflectivity and differential reflectivity values outside of the segments, identified as rain, are not corrected. Also, small ice particles add only a minute amount to the overall one-way path attenuation. Therefore, once the beam propagation along the radial is higher than the freezing level, additional attenuation is no longer calculated.

The weather classification algorithm relies on three principles about hydrometeor particles:

- Hydrometeors particles tend to have a high value of the co-polar correlation coefficient, RhoHV. A mean RhoHV is computed over a defined length. If the mean value becomes lower than the threshold, range bins are flagged to not use the data. The default value used within this classifier is 0.85.
- Quantitative measurements of PhiDP tend to be highly correlated, from range bin to range bin, when hydrometeor targets are present. This gives a "smooth" looking data field versus more "noisy" fields that are likely to have non-meteorological particles. Therefore, the texture over some area is used as a discriminator. The standard deviation of PhiDP is calculated over the same range as the mean RhoHV. This is used to represent texture along the radial.
- Convective and stratiform meteorological events have some size to them; typically more than a few kilometers. This rule is used to decide when to enter a rain segment and how long this segment must be before exiting to a no weather segment.

Each range bin is uniquely evaluated as being rain or no weather using **Equation 1** and **Equation 2** in Dual-polarimetric attenuation correction (page 231).

9.2.3.2 Iterative PhiDP Correction

For segments identified as rain, a first guess of intrinsic reflectivity is created. The approach is to use the same expression seen in **Equation 2** in Dual-polarimetric attenuation correction (page 231), but substituting the change of PhiDP from the beginning to the end of the segment for Z.

This appears as:

Equation 3

$$dBZc = dBZc + 2C\Delta r \sum (\theta_{r\ min} - \theta_{r\ max})^{E}$$

For the first rain segment nearest to the radar, the system PhiDP values are initially set by a configuration setting in the RVP10 **dspx** menus. However, this system PhiDP value is constantly updated with the average PhiDP values seen in the first bins of the rain segments.

The initial guess of the intrinsic reflectivity is adjusted using an iterative process, which is constrained by the total attenuation along the entire radial. The total attenuation is now determined by:

Equation 4

$A = ½\ \alpha_h\ \Delta\ \Phi_{DP}$

where the α coefficient is a constant, slightly dependent on temperature. The adjustment is performed by using the ratio of the total attenuation and a summation of measured reflectivity values to the particular range bin to scale the correction.

Equation 5

$$dBZc = dBZ(r) + \frac{A}{\sum_{r=\min}^{r=\max} 10^{0.1b(dBZ(r-1))}} \sum_{r=\min}^{r} 10^{0.1(dBZb(r-1))}$$

In this manner, the sum of the corrections applied to each range bin are equivalent to the total attenuation, as determined by $\Delta\Phi_{DP}$.

This first guess of intrinsic reflectivity, sometimes called the DP method, may be output from IRIS or the RVP10 software for research purposes. For operational radars, the full IDPC methodology should be used and is the default selection.

9.2.4 Adaptive K_{dp} Moment Estimation

The specific differential phase, K_{dp}, is the range derivative of the Angular PhiDP, described in Cubic spline fit (page 230). This is written as:

$$K_{dp}(r) = -j\left\{\frac{\vartheta'(r)}{\vartheta(r)}\right\} + e\left(r\right)$$

Angular PhiDP has no folding, but is still noisy due to backscattering effects and measurement errors. When performing differentiation of noisy data in the more traditional LSQ method of calculating K_{dp}, the errors are magnified in the K_{dp} output.

With the continuous complex form of Angular PhiDP data, a cubic spline methodology is chosen, because it has the greatest smoothness over all functions evaluating derivatives. Any smoothing function can be overly aggressive causing loss in data fidelity; the over smoothed situation. Therefore, an adaptive process is created to adjust the smoothing function.

The weather classification algorithm described in Weather classification algorithm (page 234) is re-used to calculate K_{dp}. The weather classification algorithms defines segments having liquid rain versus other segments. In areas where there is no liquid rain, K_{dp} should be 0. This assumption provides a convenient end condition to determine all the coefficients needed in the cubic spline equation.

The cubic spline processing [3)] of the Angular PhiDP is performed in 2 steps. The first pass uses a standard fixed smoothing function, which evaluates the mean and dispersion of Angular PhiDP. The dispersion results from the first pass is then used as a weighting function to adapt or scale the smoothing function to match the properties of the Angular PhiDP data. This adaptive function ensures that the trade-off between smoothness and data fidelity is optimized.

9.2.5 Setting up attenuation correction of Z

You need a valid dual-polarization license code to use the dual-polarization attenuation correction function. The code is activated and encrypted in the setup utility GUI field **License**. Two use cases are possible:

- Real-time correction in RVP10
- IRIS dual-polarization attenuation correction during the ingest/re-ingest process

The configuration in both cases is specified within the file *etc/vaisala/irisrda/dualpol.conf*. See Dual-polarization configuration file for attenuation correction of Z (page 237).

9.2.5.1 Setting up real-time correction in RVP10

This corrects the original Z and ZDR moments. To activate the feature, make sure that:

1. The signal processor is licensed for dual-polarization.
2. The signal processor fundamental operating mode in **dspx** settings is configured for dual-polarization.

   Table 60 Fundamental RVP10 operating mode: 3

#	BW	DynR	Filt	Pol	IFD	Description
0	Full	Norm	Norm	1	1	Standard single channel
2	Full	Norm	Norm	2	2	Dual Pol on separate IFDs
3	Half	Norm	Norm	2	1	Dual Pol on single IFD
4	Half	Wide	Norm	1	1	Extra wide dynamic range
5	Half	Norm	Long	1	1	Extra long/fast FIR filters

3. The dual polarization attenuation filter is enabled using the selection in the **mp** TTY setups:

```
Polarimetric Attenuation Correction:"USER" or "ALWAYS"
```

3) *Wang, Y., and V. Chandrasekar, 2009: Algorithm for Estimation of the Specific Differential Phase. J. of Atmospheric and Oceanic Technology, 26, 2565-2578.*

9.2.5.2 Setting-up IRIS Dual polarization Attenuation Correction

You activate the attenuation correction in the IRIS **Setup** utility.

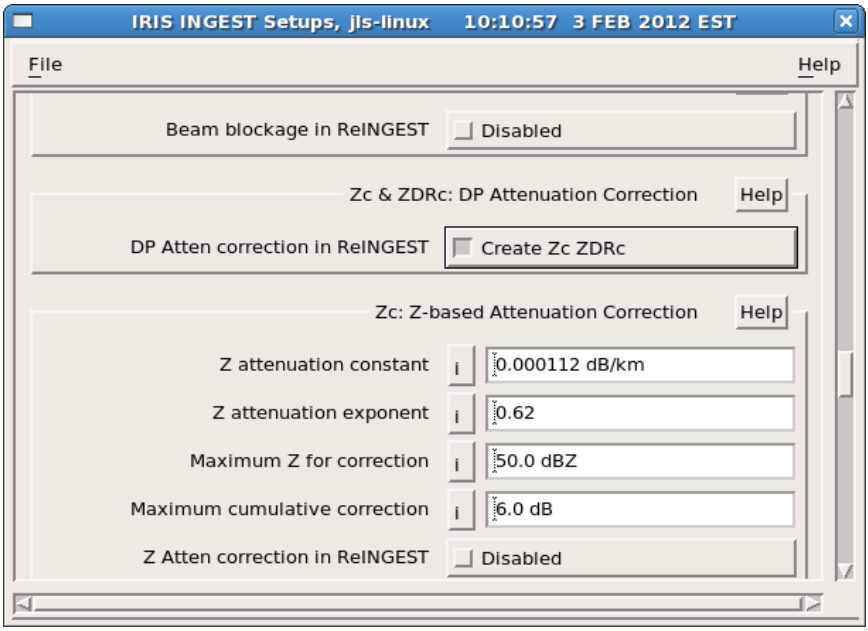


Figure 40 IRIS/Setup Utility - Ingest Tab

1. In IRIS task configuration menu in the processor configuration, enable **DP Attn Cor Z ZDR** and remember to set in the **dspx** utility:

   Polarimetric Attenuation Correction:"USER"

2. In IRIS **Setup** utility ingest tab, enable the **DP Atten correction in ReINGEST**.
3. Restart the IRIS system

The installed and activated **Dual Polarimetric Attenuation Correction** runs promptly on appropriate input data and computes results using the current configuration.

9.2.6 Dual-polarization configuration file for attenuation correction of Z

The distinct parameter lines are listed and described below, with their default settings for C-band radars.

The first section of the configuration file sets the parameters used to calculate the actual attenuation at each range bin.

```
#================================================#
#Attenuation correction
# Algorithm selection. Available options:
# DP - propagational depolarization based attenuation
#      (dBZ_real(r) = dBZ_measured(r) + alpha PhiDP(r)).
# IDPC - iterative DP constrained attenuation correction.
#        Single iteration with DP corrected dBZ(r) as
#        an initial dBZ(r).
algorithm = IDPC
```

This setting selects of the dual-polarimetric attenuation corrections, based on the iterative constrained process or only the first guess field. The recommended setting for all operational systems is **IDPC**.

```
# DP attenuation constant for horizontal polarization alpha_h = 0.105
```

This setting sets the value of the constant used within **Equation 4** (see Iterative PhiDP Correction (page 234)) when computing the total horizontal reflectivity attenuation along the radial. The 0.08 value is the recommended global setting.

This value is slightly variable with temperature. The following table shows the estimated **alpha_h** correction coefficients for several wavelengths and mean temperatures provided by Jameson 1991.

Table 61 Estimated **alpha_h** Correction Coefficients for Different wavelengths and Mean Temperatures

Temperature (°C)	S-band (2.80 GHz)	C-band (5.48 GHz)	X-band (9.34 GHz)
0	0.026	0.097	0.248
10	0.019	0.081	0.241
20	0.015	0.066	0.230
30	0.012	0.053	0.213
40	0.009	0.043	0.195

```
# DP attenuation constant for vertical polarization
alpha_v = 0.065
```

This setting sets the value of the constant used in **Equation 4** when computing the total vertical reflectivity attenuation along the radial. The 0.105 value is the recommended setting. The value is slightly variable with temperature.

```
# The melting layer (aka bright band) above sea level.
# No attenuation is added in the part of the ray above ML
melting_layer = 1300.0
```

This setting sets the height of the 0° C level above the surface reference height in IRIS. This sets a limit in the correction algorithm to not add accumulate additional attenuation when the beam is above this height. This value is only used within the IRIS/Reingest process. RVP10 uses the melting level values specified in setup memory location.

```
# The above value for melting layer is used only if melting
# layer altitude is not set for processed data set or if
# the following flag is set to 'yes'
force_conf_melting_layer = no
```

When re-ingesting data into IRIS and performing dual-polarization attenuation correction, setting this question to **yes** overwrites the melting level height originally stored with the data. This allows users to correct wrong heights or add a height value to the archived data, which may not have had the variable stored.

```
# The b parameter from the reflectivity based attenuation
# equation: A(r) = a Z(r)^b
b_exp = 0.78
```

This sets the value of the exponent within the summation terms seen in equation 5. The recommended value is 0.78 for C-band radars, and 0.46 for X-band radars.[4)]

The PhiDP section of the configuration file sets the parameters used to condition and unfold PhiDP for input to the attenuation correction, either the LSQ or Cubic Spline[5)] techniques.

```
# PHIDP processing parameters

# PHIDP processing algorithm type (in IRIS reingest, use
# dspx for RVP):
# LSQ - Weighted least square fit to a linear function
#       FIR filter coefficients are used as weights. Use
#       fir_width to control width of the filter.
# SPLINES - Smoothing and numerical derivatives calculation using cubic
splines algorithm
PhiDP_alg = SPLINES
```

This parameter allows you to choose SPLINES or LSQ method during IRIS/Reingest processing. In RVP10 processing, a similar question is in the **dspx mp** menu.

```
# Smoothing factor for the first, non-adaptive, stage of
# cubic splines processing (in IRIS reingest, use
# dspx for RVP)
standard_smoothing_factor = 0.1
```

4) *Gorgucci, E., and V. Chandrasekar, 2005: Evaluation of Attenuation Correction Methodology for Dual-Polarization Radars: Application to X-Band Systems. J. of Atmospheric and Oceanic Technology, 22, 1195-1206.*

5) *Chandrasekar,V., and Y. Wang, 2009: Adaptive Specific Differential Phase at Dual-Polarization Radar International Application Published under the Patent Cooperation Treaty. WO 2009/114868 A1.*

For IRIS/Reingest, this parameter sets the smoothing factor during the first pass through the Angular PhiDP data when performing cubic spline processing. In RVP10 processing, a similar question is in the **dspx mp** menu.

```
# Smoothing factor for the second, adaptive, part of
# cubic splines processing (in IRIS reingest, use
# dspx for RVP)
adaptive_smoothing_factor = 1.1
```

For IRIS/Reingest, this parameter sets the adaptive smoothing factor during the second pass through the Angular PhiDP data when performing cubic spline processing. This factor will be adjusted with a weighting function. In RVP10 processing, a similar question is in the **dspx mp** menu.

```
# The width, in km, of FIR filter used as weights in
# LSQ processing. (in IRIS reingest, use
# dspx for RVP)
fir_width = 3.0
```

For IRIS/Reingest attenuation correction, the impulse response of the FIR filter is configured in kilometer widths. In RVP10 processing, these parameters are set in the **dspx mp** menu. This variable also sets the range window for calculating K_{dp}.

The data thresholding parameters are used to reduce the amount of data being considered during the attenuation correction process. This is performed to reduce the CPU load on the computer.

```
# ========================================================
# Data thresholding parameters (for processing mask)

# The shortest distance, in km, at which polarimetric
# measurements, such as PHIDP, RHOHV, ZDR, etc, are
meaningful.
recovery_range = 1.0
```

Close to the radar site the dual polarimetric data values may not be representative of the intrinsic values due to near-field antenna issues. This setting specifies an initial range limit to ignore during the attenuation correction processing.

```
# Lowest allowed/meaningful SNR
snr_lim = 3.0
```

With very weak signals the PhiDP measurements become very noisy. A signal to noise ratio threshold may be set. This flags range bins, having SNR below this value, to not be included into the calculations.

```
# Lowest allowed/meaningful RHOHV
rhohv_lim = 0.8
```

Hydrometeor particles typically have RhoHV values greater than 0.85. The setting creates a threshold to not allow data from range bins having RhoHV values below the criteria to enter the calculations.

The weather classification algorithm parameters are used in creating the rain versus no weather classifications. See Cubic spline fit (page 230).

```
# ========================================================
# The weather classification algorithm parameters. (for the
# build in weather mask)

# Number of sequentially "good" bins needed to transition
# from "no weather" to "rain".
# Note: It is also the width of the gate used to compute
# STD of PHIDP, and mean value of RHOHV. [bins]
long_gate = 15
```

This question defines the width of a sliding window to calculate the average RhoHV and standard deviation of PhiDP at specific range bins. The window width is defined in range bin units.

```
# Number of sequentially "bad" bins needed to transition
# to "no weather". [bins]
short_gate = 10
```

When the range bin is currently in a rain segment, but the average RhoHV and PhiDP standard deviation are below the thresholds, this setting states how many consecutive range bins must be below the criteria to exit the rain segment.

```
# Lowest allowed mean RHOHV at the transition to "rain"
rhohv_enter_r = 0.85
# Highest standard deviation of PhiDP at the transition
# to "rain" [deg]
PhiDP_std_enter_r = 7
```

These 2 criteria are the definitions when the range bin is considered to be the start or a rain segment. Both criteria must be met to define the start of a rain segment.

```
# Low mean RHOHV that triggers exit from "rain"
rhohv_exit_r = 0.70

# High standard deviation of PhiDP that triggers exit
# from "rain" [deg]
PhiDP_std_exit_r = 15.0
```

This sets the thresholds when the range bin is thought to be bad and should be in the No Weather category. When either of the actual values are below the average RhoHV or above the standard deviation of PhiDP, the bin is flagged as bad. When a consecutive number of [short gate] bins are flagged as bad the rain segment transitions to No Weather.

```
# Lowest allowed SNR in "hail" region
# Note: Should be higher then snr_lim
# snr_exit_h = 8.0
# Lowest allowed RHOHV in "hail" region.
# Note: Should be lower then rhohv_lim_r.
# rhohv_exit_h = 0.6

# Standard deviation of PhiDP that triggers exit from "hail"
# Note: Should be higher then PhiDP_std_exit_r
# PhiDP_std_exit_h = 20.0
```

As hail is uniquely different than the presumed rain and No Weather classification, there is a different amount of attenuation than with liquid hydrometeors. In the future, the attenuation correction algorithm will include a different computation to calculate the amount of attenuation within the hail segment. For now, the parameters to distinguish a hail segment are place holders for future development.

This section allows you to specify parameter quantities per site. This is especially useful in the case of IRIS/Reingest receiving data files from multiple radar sites. To use this parameter, state `[SITENAME]` as a header line and repeat any of the name-value pairs from within this configuration file that must be unique to that site.

```
# ========================================================
# per-site sections can be used to add per-site variation
# of the parameters set in the general section.
[wes-grad]
alpha_h = 0.1
```

9.3 HydroClass

9.3.1 HydroClass Overview

Vaisala Hydrometeor Classification (HydroClass) software makes optimal use of dual-channel measurements to detect the types of scatterers present in the atmosphere, such as rain, hail, snow, graupel, and even non-meteorological targets such as insects, chaff, and sea clutter.

In addition to the improvements in precipitation estimation that are achieved with dual-polarization radar, HydroClass provides the ability to deduce and map the types of scatterers enhances the power of dual-polarization radar for applications such as:

- Hail detection
- Lightning hazard potential forecasting
- Detection of convection and stratiform rain
- Highway snow removal
- Airport terminal operation
- Rain/snow line demarcation
- Melting height detection
- Weather modification for hail mitigation
- Insurance industry claims verification
- Military detection of chaff

- Data quality improvement by elimination of non-meteorological targets
- Improved precipitation forecasting
- Hydrological modeling

9.3.2 HydroClass algorithms

HydroClass is a collection of echo identification algorithms in the IRIS and the digital signal processor (RVP10). The HydroClass algorithms output an **HCLASS** data type of echo identification. It is analogous to other data types, such as DBZ, V, Z_{dr}, and so on. For example, the **HCLASS** data type may be used as input to IRIS products such as PPI, CAPPI, XSECT, and WARN.

The HydroClass algorithms can be activated in many steps of radar data processing, depending on the customer use case:

- RVP10 HydroClass processing
  Ray data is processed in RVP10, and the bin level class assignments are output in real-time for each range bin. This approach is well-suited to applications where IRIS software is not active since the particle type can be displayed directly by your display software.
- RVP10 and IRIS/Radar processing
  The identification of convection and stratiform rain is based on vertical radar echo structures, which show up in volumes of sweep data. These data are available in the processes of IRIS/Radar. The HydroClass classification of convective and stratiform rain is implemented as part of the IRIS/Radar real-time process.
- IRIS HydroClass processing
  Dual-polarization data comes from RVP10 or from a third party processor, and is passed to an IRIS/Radar or an IRIS/Analysis system that is enabled with the HydroClass feature. The bin level algorithms produce **HCLASS** ingest data, which can be formatted as RAW products for archival or for rerouting in the network. **HCLASS** products are color-coded maps of precipitation classification categories, which can be output to and displayed on other IRIS workstations and IRIS/Web clients. Output of the GIF and other standard image formats is also supported. Archived RAW data can also be reprocessed into classes.

9.3.2.1 Gate Contents and Precipitation Patterns

Regarding consideration of data radar echo, HydroClass echo identification and classification methods belong to the following categories.

Local (Bin-to-Bin) Classification

These algorithms analyze the echo features in each volume element of observation (gate bin) to estimate the bulk contents of the volume element.

Typically, each gate bin is declared to consist of scatterers of a given type (a class of hydrometeors or of other scatterers). The algorithm may use features of data in the next surroundings of the gate. Even broader information can be used as a specific constraint. In the first order, each bin assignment is deduced from the data features of that particular bin.

Most HydroClass classification methods belong to the local bin-to-bin category.

Spatially Distributed Objects (Weather Pattern)

The type precipitation is sometimes best described as a weather event (pattern), characterized as an extended object in the atmosphere.

A weather event derives its features from large scale spatial (and temporal) distributions. Bin-to-bin consideration ignore these aspects of data. Precipitation patterns are evident to a trained radar meteorologist, and deterministic algorithms exist to identify specified event types.

A classic example is the division into stratiform and convective precipitation. HydroClass supports this type of methodology, too.

9.3.2.2 HydroClass Fuzzy Logic

HydroClass methods use the fuzzy logic approach to polarimetric radar echo identification. Fuzzy logic allows algorithms to cope with varied levels of consistency, instead of exclusive statements of "yes" and "no". The fuzzy logic outputs can be thought of as degrees of consistencies. In most published literature and in the Vaisala documentation, these output values are known as the rule strengths (RS).

The rule strengths should not be confused with probabilities because the fuzzy memberships and rules typically represent empirically defined sets, rather than statistical likelihoods of events. The latter relates to the frequentist approach to probability. Fuzzy logic is well suited for identifying radar echoes and hydrometeor types because the radar moment signatures of different radar targets and hydrometeors are not mutually exclusive or unique.

The input variables to a fuzzy logic algorithm are first passed into membership functions (MBF). Membership functions quantify the consistency of the particular input variable within a specified output set. A value of 0 means the input variable is not included for a given set and a value of 1 describes a fully included variable. Values between 0 ... 1 characterize partial or fuzzy membership. Membership functions can be one-dimensional or multi-dimensional. For example, consider the use of reflectivity (Z_H) for characterizing hail. It is known empirically, that for lower reflectivity values below ~45 dBZ, the presence of significant hail is unlikely. Using this knowledge, a one-dimensional membership function can be designed for this simple case, shown in the following figure.

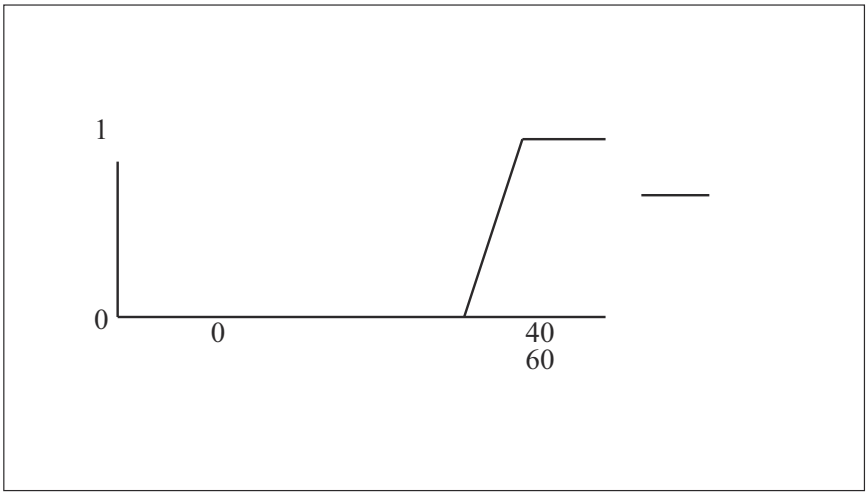


Figure 41 Membership Function in a Generic Fuzzy Logic Algorithm Schematic View

Using this membership function alone, a reflectivity at range bins of lower values implies a rule strength of 0 for hail, that is, the observation is inconsistent with hail. With multiple radar moments available as input, the rule strength for hail is an appropriate functional combination of all the membership functions using the inputs. Multiple outputs can be evaluated by constructing rule strengths to each outcome hypothesis. A deterministic outcome is obtained by comparing the rule strengths, at each range bin.

9.3.2.3 HydroClass: A Synthesis of Public Methods

The HydroClass library is a collection of public echo identification methods resulting from major developments in the weather radar community. The main sources of origin are:

- Meteo
  HydroClass results of fuzzy classification for the S- and C-band polarimetric weather radar echoes, evolved and field tested by the Colorado State University (CSU), Fort Collins, USA, and by their research partners. [6) 7) 8)]

6) *Bringi, V. N., and V. Chandrasekar, 2001: Polarimetric Doppler Weather Radar: Principles and Applications. Cambridge UP, 636.*
7) *Lim, S., V. Chandrasekar, and V.N. Bringi, 2005: Hydrometeor Classification System Using Dual Polarization Radar Measurements: Model Improvements and In Situ Verification. IEEE Transactions on Geoscience and Remote Sensing, 43.4, 792-801.*
8) *Liu, H., and V. Chandrasekar, 2000: Classification of Hydrometeors Based on Polarimetric Radar Measurements: Development of Fuzzy Logic and Neuro-Fuzzy Systems, and In Situ Verification. J. of Atmospheric and Oceanic Technology, 17.2, 140-164.*

- Precip
  HydroClass results of echo and hydrometeor classifiers for the polarimetric version of the S-band WSR-88D, developed by the National Severe Storms Laboratory (NSSL), and evaluated for operational use in the Joint Polarimetric Experiment (JPOLE) and subsequent transition of the NEXRAD radar network into polarimetry. [9] [10] [11]
- Cell
  HydroClass results of analysis of the vertical structure precipitation echo, historical developments for single polarization radars. [12]; originally the method intended for detecting hail events, here applied to detecting general cases of convective precipitation.

The implementation HydroClass is a synthesis that respects the integrity of each algorithm. Their best aspects are materialized through their sequential arrangement with:

- `Preclassifier` (adapted from JPOLE)
  An echo classifier algorithm serves as a prior quality control to:
  - `MeteoClassifiers` (of CSU origin) and `PrecipClassifier` (of NSSL origin)
    Alternative methods of gate-to-gate hydrometeor classification
  - `CellClassifier`
    A weather pattern classifier of convection and stratiform rain used as a further attribute to the previous classifications

HydroClass software supports full tunability of the algorithms through a comprehensive set of parameters. The default parameter settings reflect the original algorithms, with variant settings for polarimetric radars operating at C-band and S-band. Users may customize the settings based on local knowledge, experience, and application needs.

For more information, see HydroClass Classifiers (page 247).

9.3.2.4 HydroClass Classification Data

HydroClass echo identification results are reported as the **HCLASS** data type, which is managed similarly to other IRIS/RDA data types (for example, binary formatted RDA outputs and the IRIS archive data RAW).

The **HCLASS** data type is special in regards to the property of enumeration. The data values do not represent a continuum of measurements. For example, their arithmetic operations have no meaning, which calls for special approaches for the operations of interpolation and averaging which are often needed when the gate data are projected into IRIS products.

---

9) *Park, H., A.V. Ryzhkov, D.S. Zrnic, and K.-E. Kim, 2008: The Hydrometeor Classification Algorithm for the Polarimetric WSR- 88D: Description and Application to an MCS. Weather and Forecasting, 24, 730-748.*

10) *Schuur,T. et al., 2003: Observations and Classifiction of Echoes with Polarimetric WSR-88D Radar. Rep. NOAA, Norman, OK, University of Oklahoma. Web: http://www.cimms.ou.edu/~schuur/jpole/ JPOLE_HCA_report_pdf.pdf.*

11) *Straka, J.M., D.S. Zrnić, and A.V. Ryzhkov, 2000: Bulk Hydrometeor Classification and Quantification Using Polarimetric Radar Data: Synthesis of Relations. J. of Applied Meteorology, 39.8, 1341-1372.*

12) *Waldvogel, A., B. Federer, and P. Grimm, 1979: Criteria for the Detectionof Hail Cells. J.of Applied Meteorology, 18.12, 1521-1525. See also Foote, G.B. at al ibid.*

The **HCLASS** data consist of several bit segments of enumerated data, filled by the multiple HydroClass classification methods. A set of algorithms can be configured to run simultaneously, and provide complementary echo identification information for each bin. Digesting and presenting such rich information is a challenge to selection, but does provide opportunities for logically merging the classification results for best overall interpretation of data. IRIS product software includes extensions for computing IRIS **HCLASS** products as conveniently as other data types.

9.3.2.4.1 Nearest Neighbor in Interpolations

Interpolation of data is needed in IRIS products that include a cross-section (**XSECT**). The **HCLASS** data are interpolated with a rule of the nearest neighbor.

In its simplest form, the **HCLASS** value in any point **X** along a straight line between the points **A** and **B**, in which the **HCLASS** data values are **HCLASS(A)** and **HCLASS(B)**, is obtained in the parametrization:

$\overrightarrow{X}(t) = \overrightarrow{Ax}\left(t\right) + \overrightarrow{B}x(1-t)$, when $t \in [0,1]$

$HCLASS\left(\overrightarrow{X}\right) = HCLASS\left(\overrightarrow{A}\right)$, when $t \leq 0.5$

$HCLASS\left(\overrightarrow{X}\right) = HCLASS\left(\overrightarrow{B}\right)$, when $t \geq 0.5$

This approach generalizes to interpolating in the **N**-dimensional data grid:

$HCLASS\left(\overrightarrow{X}\right) = HCLASS\left(\overrightarrow{A_i}\right)$

where $\overrightarrow{A_i}$ has the shortest distance to the data point $\overrightarrow{X}$

$r_i = \left|\overrightarrow{X} - \overrightarrow{A}_i\right| = \min\left\{r_j = \left|\overrightarrow{X} - \overrightarrow{A}_j\right|, \; when \;\; A_j \in proximatedata\right\}$

9.3.2.4.2 Majority Rule in Averaging

Averaging of data is needed when projecting high resolution data into IRIS products.

The **HCLASS** data are averaged with the rule of majority consensus, which selects the **HCLASS** value that appears most frequently in the set of data being averaged. A value if selected randomly among equally frequent values.

9.3.3 HydroClass Classifiers

9.3.3.1 Preclassifier

The HydroClass **Preclassifier** is an implementation of the JPOLE algorithm, following closely the reference. [13)]

The JPOLE algorithm uses the fuzzy logic approach and is the first processing step for HydroClass. It uses polarimetric variables to classify the data in the range bins as:

- GC/AP (Ground Clutter/Anomalous Propagation)
- BIO or biological targets
- METEO or hydrometeor scatterers

Five radar variables are used as input to the JPOLE algorithm. These variables are:

- Horizontal reflectivity Z_H
- Differential reflectivity Z_{dr}
- Cross-correlation coefficient RhoHV
- Texture parameter of the Z_H field $\texttt{TX}(Z_H)$
- Texture parameter of differential phase TX(PhiDP)

To obtain $\texttt{TX}(Z_H)$, the Z_H data is averaged over a running average window and then the smoothed estimates of Z_H are subtracted from the original values, in other words the standard difference, effectively. A similar procedure is used for computing $\texttt{TX(PhiDP)}$. The length of the running window is configurable, the default being 5 bins for $\texttt{TX}(Z_H)$ and 10 bins for $\texttt{TX(PhiDP)}$.

The following figure shows how trapezoidal membership functions are used for the JPOLE algorithm.

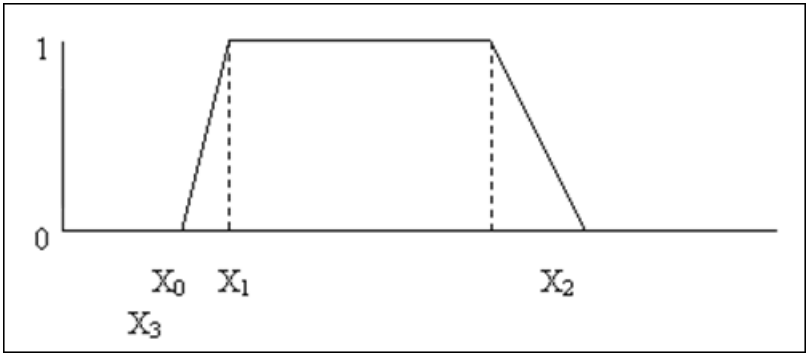


Figure 42 Example of Trapetzoid Function used as a Parametrization of the Membership Function in the JPOLE Algorithm

The following figures show the displays of the default JPOLE membership functions. The computation of additive rule strengths and class identification with maximum aggregation are in the literature reference.

13) *Schuur,T. et al., 2003: Observations and Classifiction of Echoes with Polarimetric WSR-88D Radar. Rep. NOAA, Norman, OK, University of Oklahoma. Web: http://www.cimms.ou.edu/~schuur/jpole/ JPOLE_HCA_report_pdf.pdf.*

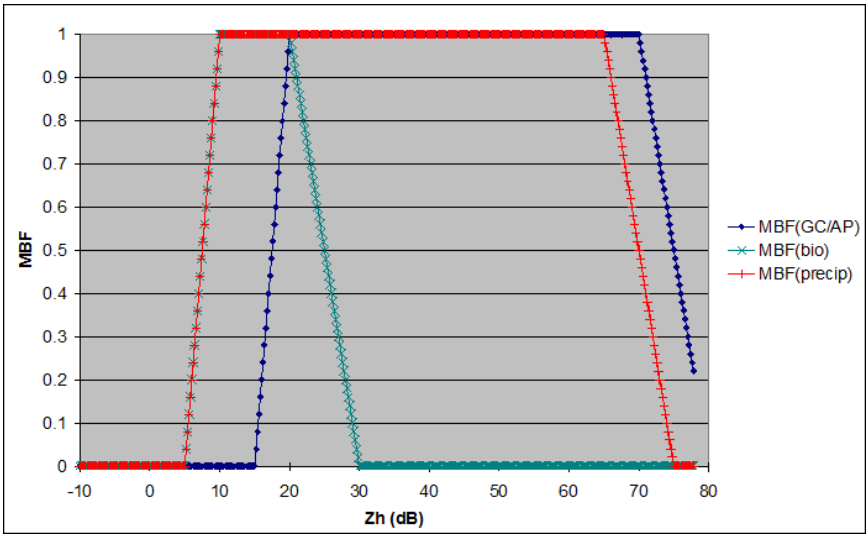


Figure 43 Default Settings of the JPOLE Membership Functions for Z_H

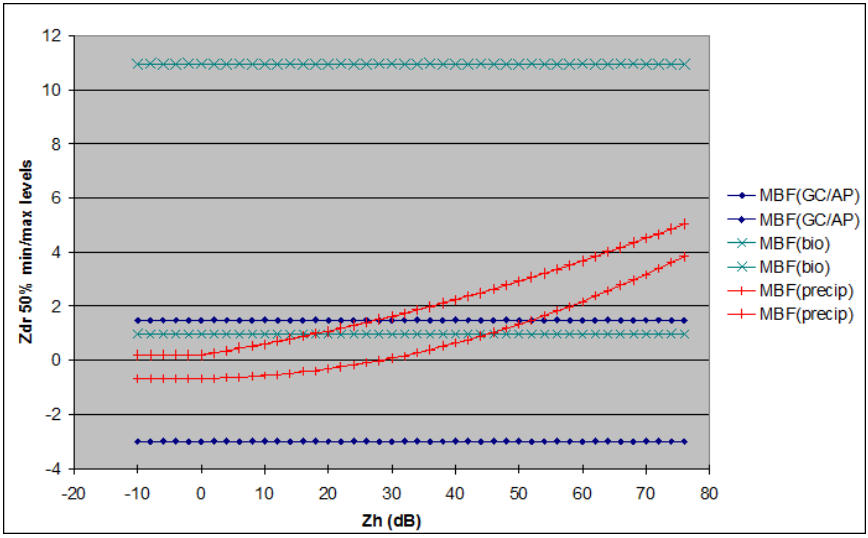


Figure 44 Default Settings of the JPOLE 2D Membership Functions for Z_{dr}

The two-dimensional (2D) membership functions are expressed as 0.5 contour levels as function of reflectivity. Class compatibilities are greater than 0.5 when Z_{dr} values fall between the regions depicted by the contours.

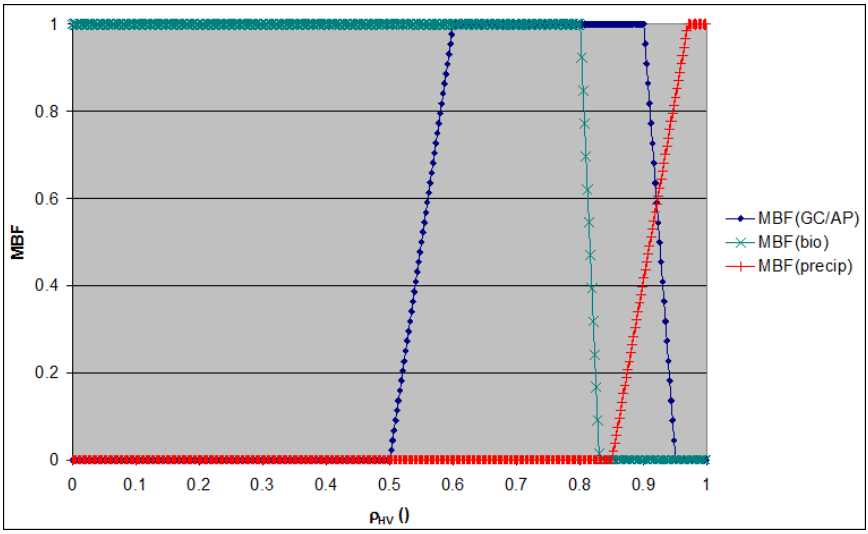


Figure 45 Default Settings of the JPOLE Membership Functions for RhoHV

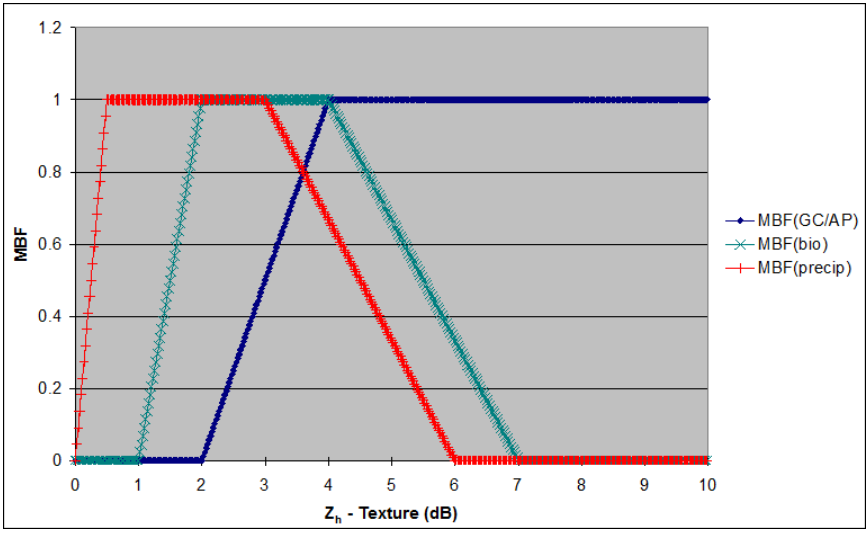


Figure 46 Default Settings of the JPOLE Membership Functions for TX(Z_H)

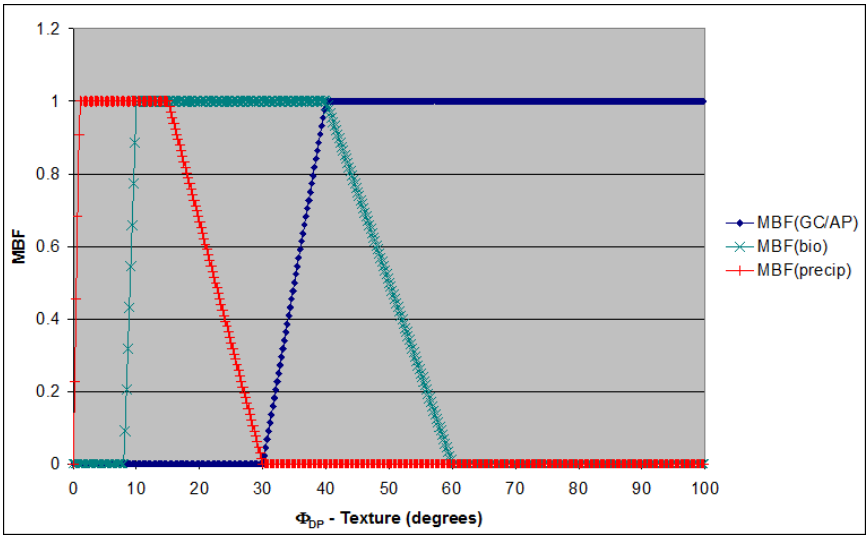


Figure 47 Default Settings of the JPOLE Membership Functions for TX(PhiDP)

9.3.3.2 MeteoClassifiers

The default classification system of warm and cold season hydrometeors is based on the fuzzy method approach. [14)]

The implementation follows closely the updated version [15)], which is the recommended general reference. The main aspects are briefly explained and are followed by the features specific to the present implementation.

The method described in the main reference [15)] was developed using radar measurements at the CSU-CHILL Facility (for a technical description, see http://lab.chill.colostate.edu/chill-technical.html).

The CHU-CHILL facility is an S-band, Doppler radar with full polarization agility and diversity at Colorado State University, U.S. The referenced classification system represents state-of-the-art knowledge and it has been verified by comparing the CSU-CHILL measurements with the in-situ airborne observations made with instruments such as 2D cloud particle measurement probe, high volume particle sampler (HVPS) and hail spectrometer. [15)]

---

14) *Liu, H., and V. Chandrasekar, 2000: Classification of Hydrometeors Based on Polarimetric Radar Measurements: Development of Fuzzy Logic and Neuro-Fuzzy Systems, and In Situ Verification. J. of Atmospheric and Oceanic Technology, 17.2, 140-164.*

15) *Lim, S., V. Chandrasekar, and V.N. Bringi, 2005: Hydrometeor Classification System Using Dual Polarization Radar Measurements: Model Improvements and In Situ Verification. IEEE Transactions on Geoscience and Remote Sensing, 43.4, 792-801.*

In the fuzzy method approach, the signatures of specified hydrometeor classes are quantified as a set of membership functions (MBF), which take the measured dual-polarization parameters obtained at each bin as input. The strength of each hydrometeor class is then expressed as the outcome (rule strength) of an inference function which takes the MBF values as input. The membership functions and the inference rule strength function formalize the meteorological interpretation encoded in the classification method.

The membership functions (MBF) of the observable x are parametrized in the CSU method as beta functions:

$$MBF(x; m, a, b) = \frac{1}{1 + \left(\frac{x - m}{a}\right)^{2b}}$$

in which **x** stands for reflectivity (Z_H), differential reflectivity (Z_{dr}), specific differential phase (K_{dp}), cross-correlation coefficient (**HV**), or observation altitude (**h**). The expression specifies the one dimensional case (1D), while evolutions of **MBF(**Z_{dr}**)** and **MBF(**K_{dp}**)** at varied reflectivities are detailed further by parametrizing $y = y(Z_H)$ where $y = m, a\ ,\ or\ b$. Such parameterization allow extending the 1D MBFs to 2D.

The membership function of the radar observation volume altitude has dependence on the melting layer height (ML), which is thus also used as input. The membership function of altitude can also be formulated as a 2D MBF, in which parameters m and a are dependent on ML.

In IRIS/RDA, the melting layer height is defined to coincide with the 0 °C isotherm, and related expressions in literature algorithms are converted to match the IRIS/RDA definition. ML can be fed in from an external source, or it is acquired from the archived RAW file headers, or it can specified explicitly in the HydroClass configuration.

The classification results are presented by labeling each bin with the hydrometeor class that is most compatible with the observations, that is by choosing the class of highest rule strength. The rule strengths are computed using the equations of the literature reference using the quoted four radar variables and the altitude as input. Threshold parameters associated to each of the rule strengths are used to specify bins for which the class is ambiguous, for example, non-meteorological targets.

9.3.3.2.1 IRIS/RDA MeteoClassifiers Features

9.3.3.2.1.1 Hydrometeor Classes

The reference method characterizes membership functions for 11 classes, while the results were interpreted in the scheme of 9 final output classes.

Simplicity was considered advantageous for typical applications, and 5 final classes have been defined as the basis for classification in the current implementation:

- Rain
- Wet Snow
- Dry Snow
- Graupel
- Hail

In order to characterize the precipitation phenomena in realistic fashion, a commonly emerging mixture 'rain and hail' has been modelled with specific MBFs, in addition. Outcomes in this intermediate class are merged into the final output class 'hail'.

Membership functions (MBFs) and rule strengths (RS) have been re-optimized using the new class basis in a dedicated effort, see Customization to Simultaneous Transmission, Simultaneous Receive Mode (page 253).

9.3.3.2.1.2 Prior Quality Considerations

HydroClass uses standard IRIS/RDA measurands as input, and their quality thresholds propagate into HydroClass quality thresholds.

In addition, the `Preclassifier` of non-meteorological versus precipitation echoes (JPOLE algorithm) is used as a prior quality factor to the hydrometeor classifier (CSU algorithm) by passing in the bins classified as meteorological echoes by JPOLE, only.

9.3.3.2.1.3 Variants for Warm and Cold Seasons

The melting layer height (ML) is one of the inputs for the set of MBFs and RSs. The main mode of the classification system is dedicated to the summer like situation with a positive value of ML. The organization of MBF and RS parameters follow this case of the main reference. See HydroClass: A Synthesis of Public Methods (page 245).

In cold season, any large scale melting layer is absent. The problem of classifying precipitation is better to be started from a different a priori expectation. The task is still nontrivial, as liquid forms of precipitation are not excluded (warm fronts, freezing rain). Significant convection can occur in cold season, too, producing heavy solid hydrometeors.

In order to describe these circumstances, a variant of the method has been constructed for the case ML less than 0, signalling precipitation in cold season. In such climatology, modified MBFs are computed for altitude, and the ML does not enter in MBF calculation in this case. Explicitly, liquid form of precipitation (rain) is allowed up to a fixed altitude of 5 km (3.1 mi) (MSL), wet snow is allowed to a characteristic altitude of winter storms, assumed constant of 2.5 km (1.6 mi), at present. Other types of precipitation are indifferent with respect to altitude.

Other MBFs are unchanged from their warm season settings. Subsequently, modified weights in rule strengths calculation are used.

9.3.3.2.1.4 Customization to Simultaneous Transmission, Simultaneous Receive Mode

The main literature reference method parameters have been tuned, using data obtained in the alternating horizontal/vertical polarization mode, in which both the co-polar terms are observables, that is, H received from H transmitted (HH) and V received from V transmitted (VV). The cross-polar terms, that is, V received from H transmitted (VH) and the (HV) signals are observables as well. As a consequence, the linear depolarization ratio (LDR) can be promptly used as an input observable, paying special attention to cases of LDR data at low signal to noise ratio.

Operation in the mode of simultaneous transmission and simultaneous receive, known as the 'hybrid' mode, excludes availability of LDR at pulse-to-pulse basis. Adding to this, characteristics of other polarimetric observables depend on the processing mode, in presence of significant propagation effects in particular.

These 2 general reasons suggest for method customizations specific to polarimetric processing modes, and has in part motivated the procedure.

9.3.3.2.1.5 Fuzzy Parameters Optimized for C-Band

The new features imply reprocessing of the algorithm parameters in the new class basis, in a variety of climates, and in the hybrid polarimetric processing mode. Adding these customizations, the understanding of data and phenomena at S-band was extended to C-band.

The existing CSU parameter optimization procedure was applied to multi- season data samples obtained at the Vaisala polarimetric weather radar prototype at University of Helsinki.[16)]

The data samples represented a variety of weather cases (see the following 3 figures) as follows:

- Summer convection
- Winter storm
- Stratiform frontal precipitation

As result, new membership functions and rule strength coefficients were obtained that are used as default settings for C-band weather radars operating in the 'hybrid' mode. The distributions of the membership functions are visualized in the following figures.

For numerical values and more details of the membership function parameters, see Parametrized Fuzzy Parameters (page 287).

16) *Puhakka, T. et al., 2006: University of Helsinki Research Radar Setup. 4th European Conference on Radar Meteorology and Hydrology, Barcelona, Spain.*

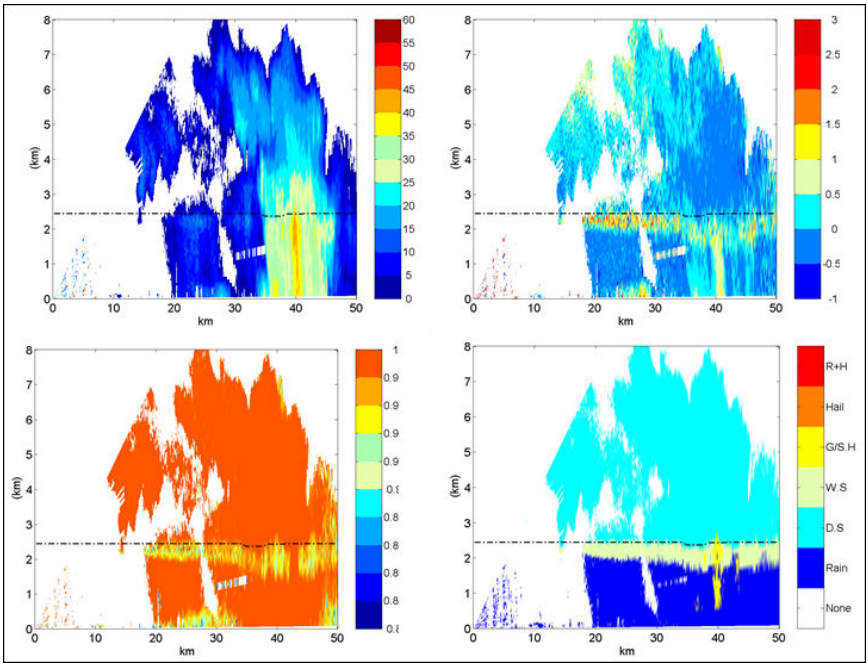


Figure 48 Selected Observables in the RHI Data Sample of a Summer Convection used for Optimizing the Fuzzy Parameters

1 Reflectivity Z (dB)
2 Differential reflectivity Z_{dr} (dB)
3 Cross-correlation coefficient RhoHV(0)
4 Classification result ('D.S': dry snow, 'W.S': wet snow, 'G/S.H' graupel, small hail)

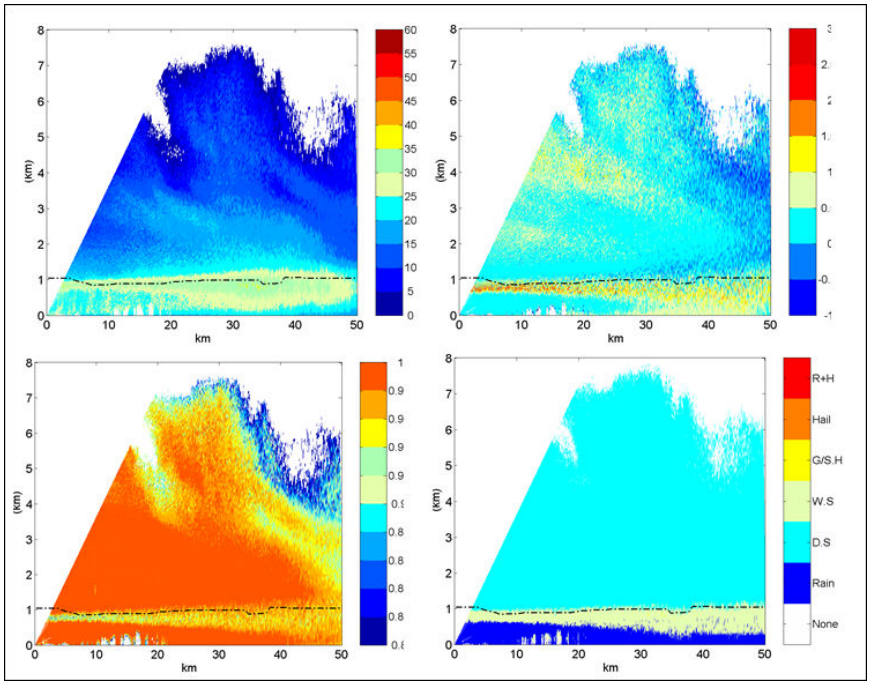


Figure 49 Selected Observables in the RHI Data Sample of a Stratiform Precipitation (Warm Front) used for Optimizing the Fuzzy Parameters

1 Reflectivity Z (dB)
2 Differential reflectivity Z_{dr} (dB)
3 Cross-correlation coefficient RhoHV(0)
4 Classification result ('D.S': dry snow, 'W.S': wet snow)

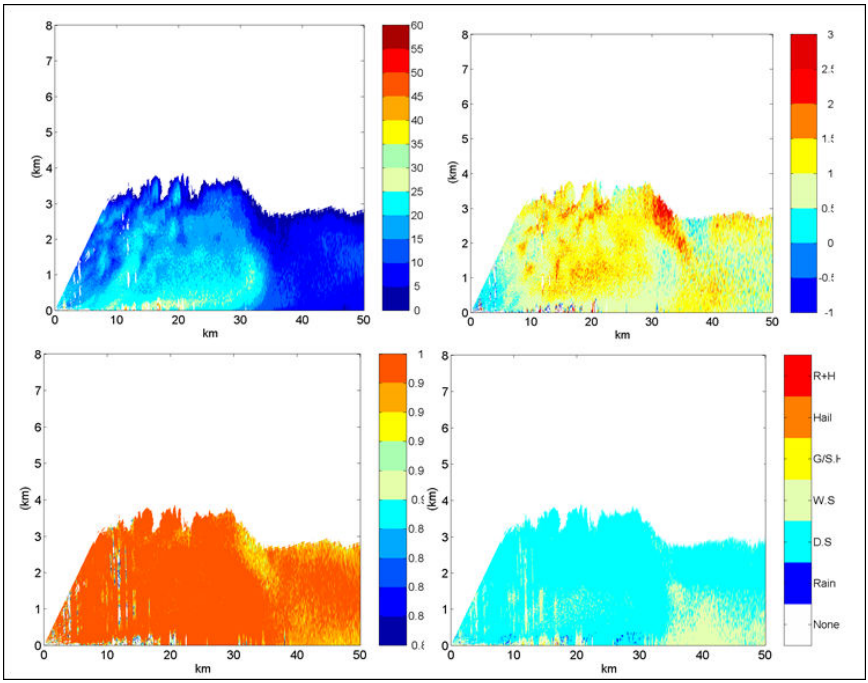


Figure 50 Selected Observables in the RHI Data Sample of a Winter Precipitation Event used for Optimizing the Fuzzy Parameters

1 Reflectivity Z_H (dB)
2 Differential reflectivity Z_{dr} (dB)
3 Cross-correlation coefficient RhoHV(0)
4 Classification result ('D.S': dry snow, 'W.S': wet snow)

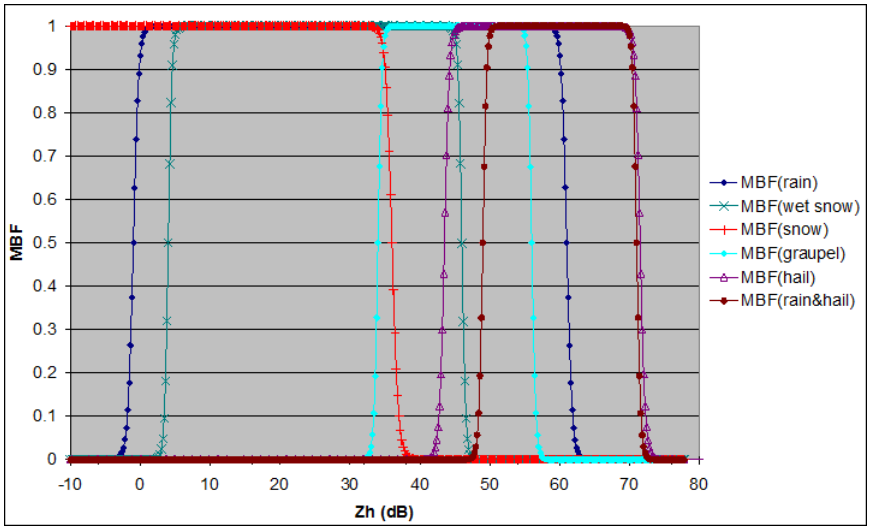


Figure 51 C-band Default Settings of the Membership Functions for Reflectivity (Z_H)

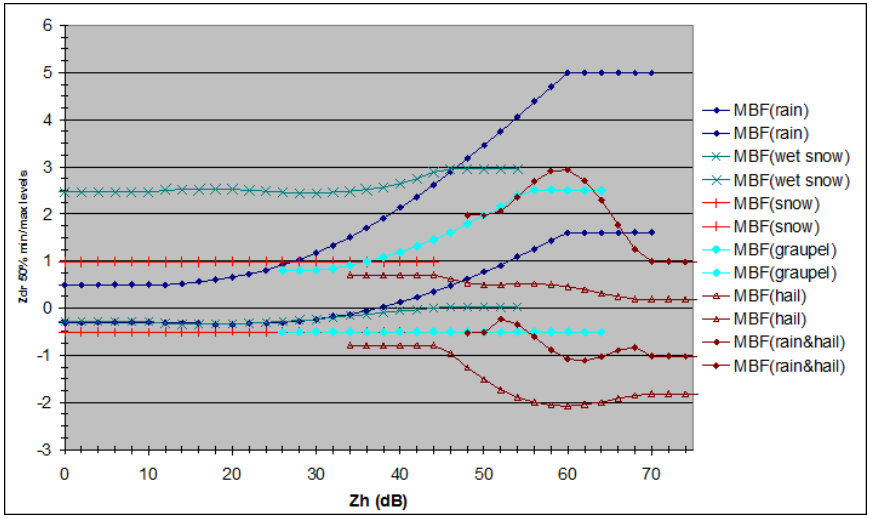


Figure 52 2D Membership Functions for Differential Reflectivity (Z_{dr}) (Expressed as 0.5 Compatibility Contours as Function of Reflectivity)

Class compatibilities are greater than 0.5 when Z_{dr} values fall between the regions depicted by the contours.

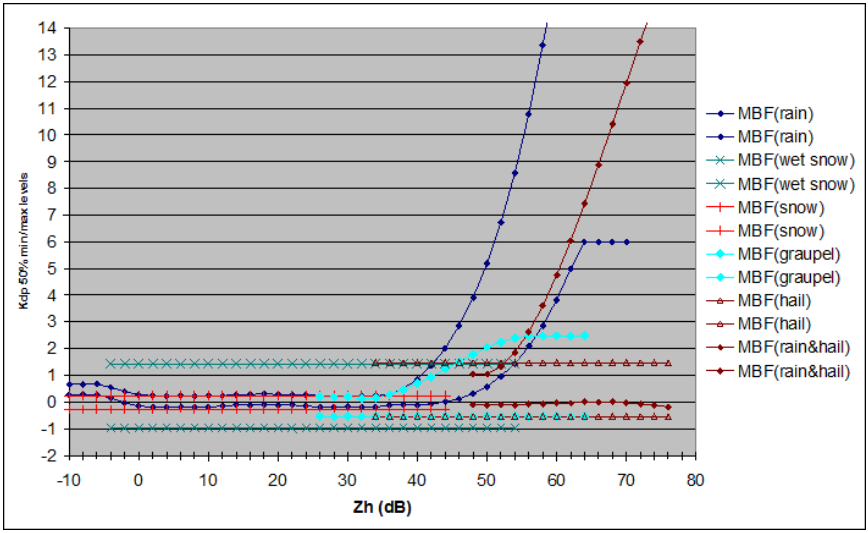


Figure 53 2D Membership Functions for Specific Differential Phase (K_{dp}) (Expressed as 0.5 Compatibility Contours as Function of Reflectivity)

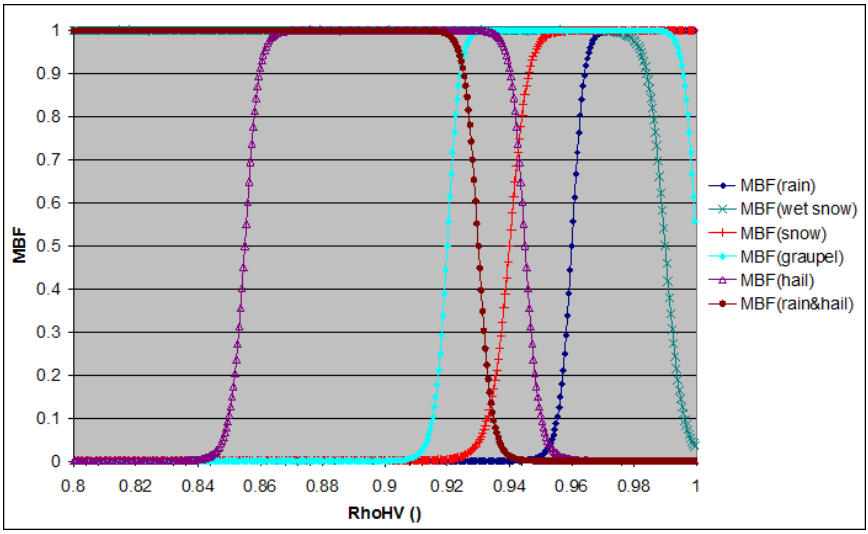


Figure 54 Membership Functions of the Cross-Correlation Coefficient (RhoHV)

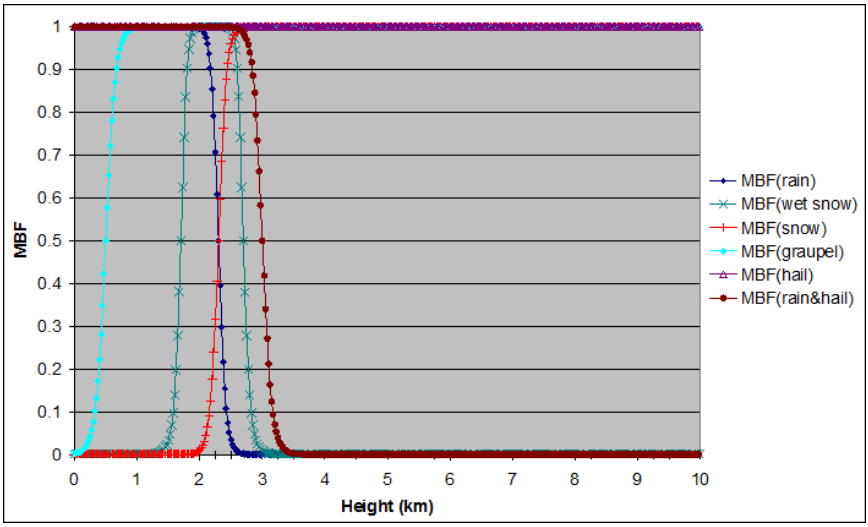


Figure 55 Membership Functions for Altitude (h) (Depicted for Melting Layer Height at 2.5 km)

The contours shift linearly as function of the melting layer height. All heights are expressed with respect to mean sea level.

The optimized expressions for rule strengths (RS) are analogous to the literature reference with the following new default coefficients:

$$RS_i(ML > 0) = MBF_i(Z_H) \times MBF_i(h) \times MBF_i(Z_{dr}) + 0.5 \times MBF_i\left(K_{dp}\right) + 0.5 \times MBF_i(RhoHV)2$$

$$RS_i(ML < 0) = MBF_i(Z_H) \times 0.7 \times MBF_i(h) \times MBF_i(Z_{dr}) + 0.3 \times MBF_i\left(K_{dp}\right) + 0.5 \times MBF_i(RhoHV)3$$

In these expressions $\mathtt{RS_i(ML>0)}$ and $\mathtt{RS_i(ML<0)}$ refer to rule strengths of a class i for warm and cold season cases, respectively. $\mathtt{MBF_i(x)}$ refers to membership functions of variable `x` for the class `i`.

9.3.3.2.1.6 Unimplemented Features

The melting layer height is one of the variables used in the hydrometeor classification fuzzy method. The main literature reference introduces a radar based estimate for a variable melting layer heights in convective storms. In essence, the estimate is based on vertical gradient profiles of differential reflectivity. Other approaches have been proposed for stratiform cases and they also use vertical profiles and/or gradients in reflectivity, differential reflectivity, and/or polarization cross-correlation coefficient, as prime signatures of melting layer.

Observing vertical profiles and/or gradients, unambiguously, requires dedicated scan strategies such as RHI scans, or alternatively several horizontal sweeps at varied elevations in a well designed volume scan. Such requirements contradict a HydroClass design goal of independence on scan strategy. Subsequently, the melting layer height is left as an open method in HydroClass, and the information is fed in as an external input variable.

Methods are available for feeding in the values at momentary basis, see Running HydroClass (page 262).

9.3.3.3 PrecipClassifier

The `PrecipClassifier` implementation closely follows the fuzzy method of Version 2: Warm season[17] , which includes the default fuzzy parameter values. For the classes of rain, the default values of the membership functions for textures of reflectivity and differential phase take the common values of precipitation of the preclassifier.

This classifier has the same default settings for S-band and C-band radars. The optimal parameters of membership functions for differential reflectivity are somewhat different for C-band radars.

17) *Schuur,T. et al., 2003: Observations and Classifiction of Echoes with Polarimetric WSR-88D Radar. Rep. NOAA, Norman, OK, University of Oklahoma. Web: http://www.cimms.ou.edu/~schuur/jpole/ JPOLE_HCA_report_pdf.pdf.*

9.3.3.4 CellClassifier

The `CellClassifier` implementation is based on the historical signature of hail and convection[18] : high level of reflectivity at a specified height above the melting layer can be used as a signature of hail (and convection in general).

In this implementation, the general signature of convection is fuzzified as a rule strength, which is a product of 2 membership functions (trapezoids):

- Minimal reflectivity required for convection
- Minimum altitude, with respect to the 0 °C isotherm

For simplicity of configuration, the shapes of the trapezoids are fixed. The full rule strength is reached when the reflectivity is 5 dB more than the minimal threshold, and when the height of those reflectivity levels reaches 1 km (0.6 mi) above the minimum altitude.

9.3.4 Setting up HydroClass

HydroClass is part of IRIS and RDA releases, and no specific actions are necessary in standard IRIS and RDA installations.

You must have a valid HydroClass license code to use HydroClass. The code is activated and encrypted in the setup utility graphical user interface (GUI) field **License**. Two license codes are provided:

- HydroClass in Reingest
  Allows activating the classifier methods of `Preclassifier`, `MeteoClassifiers`, and `PrecipClassifier` in IRIS/Analysis, that is in post processing. This license code also enables processing real-time data with the `CellClassifier` in IRIS/Radar.
- RVP HydroClass Generation
  Allows activating the classification methods of `Preclassifier`, `MeteoClassifiers`, and `PrecipClassifier` in RDA real-time ray processing.

1. Enable HydroClass in IRIS/reingest using the menu **HClass: Hydrometeor Classification in ReINGEST** in the setup block **Ingest**.

   The installed and activated HydroClass runs promptly on appropriate input data and computes results using the current configuration, when appropriate input data arrive or a task with `HCLASS` data type is running.

2. Enable HydroClass in RDA using the menu **Optional Data Parameters** in the setup block **RVP Radar Video Processor**.

   HydroClass algorithms process data if the `HCLASS` data type is configured to be reported.

9.3.5 Running HydroClass

HydroClass is implemented in the major mode of PPP in RVP10.

18) *Waldvogel, A., B. Federer, and P. Grimm, 1979: Criteria for the Detectionof Hail Cells. J.of Applied Meteorology, 18.12, 1521-1525. See also Foote, G.B. at al ibid.*

Alternatively, HydroClass can process any appropriate polarimetric radar data converted into IRIS RAW format. By construction, HydroClass sets minimal requirements on task parameters, while driving certain task settings to extremes may compromise meteorological performance.

Minimally, HydroClass requirements for input data (effectively, the RVP10 run-time parameters, IRIS task parameters, or input RAW characteristics) are:

- Polarization mode: H+V
- Major mode PPP
- Active data types: DBZ, Z_{dr}, K_{dp}, PhiDP and RhoHV
- If HydroClass is configured to use the external melting layer height (ML), a valid ML value in RVP10, or in the RAW header structure

For uniform meteorological performance, it is recommended to:

- Constrain the observation distances at the scale of 100 km (62.1 mi); far beyond 100 km (62.1 mi), the radar observation volume is expected to contain a lot of structures, in vertical direction in particular, and unambiguous classification of hydrometeors is not well justified
- Prefer reasonable samplings (32 or more) and modest antenna speeds
- Pay balanced attention to all aspects of radar data quality (pedestal alignment, calibration of reflectivity, H- to V-channel calibration balance, noise samplings, radome quality)

9.3.5.1 Operating RDA/HydroClass with IRIS/Radar

IRIS/Radar is a typical method of operating a radar equipped with RVP10 signal processing.

The most convenient mode of using HydroClass at these radars is to activate it at RVP10 and run it within standard IRIS tasks configured to **H+V** polarization mode and to the RVP10 major mode of pulse-pair-processing (PPP).

In a given IRIS task, you can switch on HydroClass in the data pop-up in the **Processor Configuration** block by enabling:

- **HClass** data type
- Required data types

Given a valid HydroClass configuration file, and valid melting layer height input (if specified that way in HydroClass configuration) no other action is necessary. HydroClass generates the **HClass** data type in the ingest data stream, analogously to other data types.

9.3.5.2 Operating HydroClass in RVP10

HydroClass is built in as an output data type in RVP10 processing, and it is available to the use case in which RVP10 is operated with the radar software.

In this case, the feature is available by activating the polarimetric **HClass** data type, assuming a standard HydroClass configuration.

9.3.5.3 HydroClass in IRIS

If HydroClass is enabled in IRIS/reingest, it considers each RAW file routed to reingest and appends the **HClass** data type in the ingest collection when the RAW data input is suitable to HydroClass.

The input radar moments must be recorded in 16-bit data format. The algorithm does not create output from 8-bit RAW data, directly. However, it is relatively straightforward to convert such data sets into 16-bit presentation in IRIS (intermediate RAW products) and use them for obtaining the **HClass** data type.

9.3.6 Configuring HydroClass

HydroClass application includes a configuration interface that is common to HydroClass in RVP10 and in IRIS. The configuration data is in the configuration template files for C-band radars and for S-band radars, respectively:

```
/etc/vaisala/irisrda/templates/hydroclass-C-band.conf
/etc/vaisala/irisrda/templates/hydroclass-S-band.conf
```

Copy one of these into the following file:

```
/etc/vaisala/irisrda/hydroclass.conf
```

If you have an X-band radar, use the *hydroclass-C-band.conf* file as a starting point, and edit it for X-band.

The files are equivalent in structure, but with default parameter sets that match with these widely used radar types.

Note that the HydroClass functionality is ready to use after the software install, radar calibration and start of radar operations. The installed configuration meets the first needs of evaluation and general use. HydroClass automatically selects the correct configuration file to match with the radar frequency.

Site-specific customizations are usually only needed in the advanced use cases, that is for optimal performance. The optimization is best made with feedback and needs of the end-user applications. Substantial amount of observational data and their careful consideration are needed for consistent customization of HydroClass.

The configurable features are:

- Choices of activation, and relations of uses of the distinct classification algorithms **PreClassifier**, **MeteoClassifiers**, **PrecipClassifier**, and **CellClassifier**
- Complete sets of fuzzy parameters of these classifier
- Miscellaneous settings, such as origin of melting layer height, data corrections, and quality considerations of the results

HydroClass polls for possible updates of the configuration file and reloads upon file change. The update takes place for each RAW file in IRIS/reingest. Restart of IRIS is not required. Consecutive RAW files are allowed to originate from radars that operate at different frequencies.

HydroClass configuration is checked for each period of PPP major mode in a RVP10 run. Note that for RVP10 radars operated by IRIS, a new task does not trigger a HydroClass configuration update, but a switch to another major mode, for example, FFT, and back to PPP is required. The updates initially trigger an IRIS/RDA message, which are later muted off, in order not to fill log files.

9.3.6.1 HydroClass Configuration File

A parameter line of the HydroClass configuration file has one of the following format:

```
dpolapp[.SubSpecifier].[ParameterGroup].ParameterItem = 1234
```

`SubSpecifier`
: Parts are optional, typically referring to distinct classifier methods `Preclassifier`, `MeteoClassifiers`, `PrecipClassifier`, and `CellClassifier`.

`ParameterGroup`
: Refers to (a vector of) parameter group, such as membership functions `MBFinputs`.

`ParameterItem`
: Refers to (a vector or a matrix of) parameters.

Each `ParameterGroup` is typically preceded by comment lines, starting with a "#". The number of subsequent parameter lines and the dimensions of the parameter vectors and matricer reflect structure of the algorithm. The structure of the algorithm is briefly explained in the paragraph of comments at start of each classifier method, indicated by a comment line:

```
#==================================================
```

9.3.6.1.1 HydroClass Configuration File Structure

The `HydroClass` configuration files consist of the main blocks listed below.

In typical first usages, you must consider the parameter settings in block (2), with a few additional key in the group (4).

Modifying parameters in groups (5) to (8) is seen as advanced use.

1. Title block and configuration nick name

   ```
   # === RDA/IRIS HydroClass configuration file (C-band) ==
   ```

2. Main algorithm selection menu selectors for activating distinct algorithms (blocks)

   ```
   # ================================================
   ```

3. RDA/IRIS major version, current IRIS/RDA version, and HydroClass revision

   ```
   # ================================================
   ```

4. Detailed data quality configurations: miscellaneous quality settings in the block `Quality`

   ```
   # ================================================
   ```

   As soon as a consistent source of melting layer height is determined at the radar, it is recommended to change the setting:

   ```
   dpolapp.Quality.Aux[0] = 1
   ```

   To

   ```
   dpolapp.Quality.Aux[0] = 0
   ```

5. Detailed configuration of the Preclassifier output settings and fuzzy parameters of the Preclassifier

   ```
   # ================================================
   ```

6. Detailed configuration of the MeteoClassifiers (warm and cold season) output settings and fuzzy parameters of the MeteoClassifiers

   ```
   # ================================================
   ```

7. Detailed configuration of the `PrecipClassifier` output settings and fuzzy parameters of the `PrecipClassifier`

   ```
   # ================================================
   ```

8. Detailed configuration of the `CellClassifier`, fuzzy parameters of the `CellClassifier`

   ```
   # ================================================
   ```

9.3.6.1.2 HydroClass configuration file contents

This section lists the distinct parameter lines with default settings for C-band radars.

Title and configuration nickname

The title comment line is machine written:

```
dpolapp.sConfigurationName ="dpolapp_C-band"
```

A descriptive name for the current configuration settings.

This parameter is the only one communicated as meta data in the header structures of IRIS/RDA data. The nickname provides a way of indicating possible evolutions and modifications made to the HydroClass configuration.

The default string **"dpolapp_C-band"** suggests the default settings. It is recommended that you change to a descriptive name when you modify algorithm parameters.

Main algorithm selection menu

```
dpolapp.bRunQuality = 1
```

Activates the quality considerations in the main block **Quality**.

```
dpolapp.usClassificationMethods = 222
```

Activates a combination of the distinct echo classifier methods. Possible settings are:

- "1": Preclassifier, only (the JPOLE algorithm for meteo versus bio versus GC/AP)
- "2": Meteoclassifiers, with the Preclassifier as a quality mask
- "21": PrecipClassifier: the JPOLE warm algorithm, with the Preclassifier mask
- "22": PrecipClassifier and Meteoclassifiers, with the Preclassifier mask
- "201": CellClassifier for stratiform and convective rain, with the Preclassifier mask
- "202": Meteoclassifiers and CellClassifier, with the Preclassifier mask
- "221": CellClassifier and PrecipClassifier, with the Preclassifier mask
- "222:" CellClassifier, PrecipClassifier, Meteoclassifiers and CellClassifier, with the Preclassifier mask

Software version information

```
dpolapp.sVersion = "8.13"
```

Machine written major IRIS/RDA version. No need to modify this.

```
dpolapp.sRevision = "8.13"
```

Machine written minor IRIS/RDA version. No need to modify this.

```
dpolapp.sDpolappRevision = "15"
```

Machine written HydroClass running version. No need to modify this.

Miscellaneous quality settings

```
dpolapp.Quality.Aux[0] = 1
```

Source of the melting layer height input:

- "1": HydroClass uses the melting layer height value suggested in the configuration file.
- "0": instructs HydroClass in IRIS to use the ML value in the input RAW file, and HydroClass in RDA to use the RVP10 process value of ML. The RVP10 value can be updated during operation (dynamically).
- In case **dpolapp.Quality.Aux[0] =0** but the requested values (in RAW or RVP10) appear undefined, HydroClass reports about the missing input. HydroClass in IRIS moves over to using the value specified in the configuration file, while HydroClass in RDA switches off until a well defined value is provided.

```
dpolapp.Quality.Aux[1] =2300
```

An estimate of the current ML in meters above mean sea level (MSL). This estimate is used when dpolapp.Quality.Aux[0] =1, see previous field. It is recommended to move to using IRIS/RDA current values in continued use.

```
dpolapp.Quality.OtherInputs[0].dParams[0] =0.0
```

A posteriori HydroClass specific off-set to H-/V-channel balance (Z_{dr}). The off-set is intrinsic to HydroClass and does not affect the Z_{dr} output data type.

Use of non-zero value is discouraged in real-time applications (HydroClass in RVP10). The recommended IRIS/RDA approach is to use the tools described in chapter Calibration considerations for dual-polarization (page 310).

General settings of PreClassifier

```
dpolapp.Preclassifier.iClassifierID =-1
```

Are the PreClassifier results formatted as in the field of **PRECIP_CLASSIFIER** in the bits 3,4,5 from **LSB=0** of the **DB_HCLASS** data type?

- "−1": **NO**, the legacy HydroClass format
- "1": **YES**, **JPOLE** classes of **PRECIP_CLASSIFIER** used

```
dpolapp.Preclassifier.uiMaxNonMeteo = 1
```

This parameter is irrelevant in all cases, except:

```
dpolapp.usClassificationMethods =1
```

This parameter defines what to do with the bins that fail acceptance to any class of **PreClassifier**. It allows simplifying the class settings further through regrouping. You can re-group **PreClassifier** (un)identified results:

- "1": unidentified are **'uiNonMeteoID**
- "2": in addition, '**GC/AP**' are '**uiNonMeteoID**'
- "3": in addition, '**BIO**' are '**uiNonMeteoID**'

```
dpolapp.Preclassifier.uiNonMeteoID =1s
```

What to do with the regrouped (un)identified results:

- "0": set as thresholded in IRIS/RDA
- "1": report as GC/AP
- "2": report as BIO
- "3": report as PRECIP

```
dpolapp.Preclassifier.dMinRS[0] =0
```

Minimum rule strength for accepting GC/AP.

```
dpolapp.Preclassifier.dMinRS[1] =0
```

Minimum rule strength for accepting BIO.

```
dpolapp.Preclassifier.dMinRS[2] =0
```

Minimum rule strength for accepting PRECIP.

```
dpolapp.Preclassifier.iAux[3] =2
```

The preferred class of the Polarimetric Meteo Index (PMI):

- "2": PMI gets high values when the PRECIP class is preferred
- "1": PMI gets high values when the BIO class is preferred
- "0": PMI gets high values when the GC/AP class is preferred

Settings for Polarimetric Meteo Index

```
dpolapp.Preclassifier.iAux[4] =1
```

PMI mathematical formula.

- "1": the 'score'

$$PMI = 1 - \frac{1}{1 + \left(\frac{RULESTRENGTH_{PREFERRED}}{\max(RULESTRENGTH)_{NONPREFERRED}} \right)}$$

- "0": the likelihood ratio

$$PMI = 1 - \frac{RULESTRENGTH_{PREFERRED}}{\Sigma RULESTRENGTH}$$

Detailed settings of PreClassifier

```
dpolapp.Preclassifier.OtherInputs[0].iParams[0] = 5
```

The number of consecutive bins used to compute texture of reflectivity.

```
dpolapp.Preclassifier.OtherInputs[1].iParams[0] = 10
```

The number of consecutive bins used to compute texture of differential phase.

```
# MBFinputs and their use:
# Reflectivity in preclassifier:
# Input data: Zh.
# MBF(Zh) is computed as JPOLE trapetzoid; an additive factor in rule strength
# MBF: default initialization with parameter settings of Table 1 Ref.1
# Zh MBF trapetzoid for GC/AP.
dpolapp.Preclassifier.MBFinputs[0].dMBF[0][0] = 15
dpolapp.Preclassifier.MBFinputs[0].dMBF[1][0] = 20
dpolapp.Preclassifier.MBFinputs[0].dMBF[2][0] = 75
dpolapp.Preclassifier.MBFinputs[0].dMBF[3][0] = 85
# Zh MBF trapetzoid for bio scatter.
dpolapp.Preclassifier.MBFinputs[0].dMBF[0][1] = 5
dpolapp.Preclassifier.MBFinputs[0].dMBF[1][1] = 10
dpolapp.Preclassifier.MBFinputs[0].dMBF[2][1] = 20
dpolapp.Preclassifier.MBFinputs[0].dMBF[3][1] = 30
# Zh MBF trapetzoid for precipitation.
dpolapp.Preclassifier.MBFinputs[0].dMBF[0][2] = 5
dpolapp.Preclassifier.MBFinputs[0].dMBF[1][2] = 10
dpolapp.Preclassifier.MBFinputs[0].dMBF[2][2] = 75
dpolapp.Preclassifier.MBFinputs[0].dMBF[3][2] = 80
# Differential reflectivity in preclassifier:
# Input data: Zdr (In reingest: adjusted with Quality offset, internally), and
Zh.
# MBF(Zdr;Zh) is computed as JPOLE 2D-trapetzoid; additive in rule strength
# Zdr MBF trapetzoid for GC/AP.
dpolapp.Preclassifier.MBFinputs[1].dMBF[0][0] = -4
dpolapp.Preclassifier.MBFinputs[1].dMBF[1][0] = -2
dpolapp.Preclassifier.MBFinputs[1].dMBF[2][0] = 1
dpolapp.Preclassifier.MBFinputs[1].dMBF[3][0] = 2
# Zdr MBF trapetzoid for bio scatter.
dpolapp.Preclassifier.MBFinputs[1].dMBF[0][1] = 0
dpolapp.Preclassifier.MBFinputs[1].dMBF[1][1] = 2
dpolapp.Preclassifier.MBFinputs[1].dMBF[2][1] = 10
dpolapp.Preclassifier.MBFinputs[1].dMBF[3][1] = 12
# Zdr MBF trapetzoid for precipitation.
dpolapp.Preclassifier.MBFinputs[1].dMBF[0][2] = -0.3
dpolapp.Preclassifier.MBFinputs[1].dMBF[1][2] = 0
dpolapp.Preclassifier.MBFinputs[1].dMBF[2][2] = 0
dpolapp.Preclassifier.MBFinputs[1].dMBF[3][2] = 0.3
# A polynomial dependence of the Zdr precipitation trapezoid on Zh
# The Zh range in which the polynomial is applied:
dpolapp.Preclassifier.MBFinputs[1].dMBF[4][2] = 0
dpolapp.Preclassifier.MBFinputs[1].dMBF[5][2] = 80
# Dependency on Zh: left tail constant term P0, linear P1, 2nd order P2, and
3rd order P3.
# The left shoulder is the same.
dpolapp.Preclassifier.MBFinputs[1].dP[2][0][0] = -0.5
dpolapp.Preclassifier.MBFinputs[1].dP[2][0][1] = 0.0025
dpolapp.Preclassifier.MBFinputs[1].dP[2][0][2] = 0.00075
dpolapp.Preclassifier.MBFinputs[1].dP[2][0][3] = 0
# The right tail constant term P0, linear P1, 2nd order P2 and 3rd order P3
# The right shoulder is the same.
dpolapp.Preclassifier.MBFinputs[1].dP[2][2][0] = 0.08
dpolapp.Preclassifier.MBFinputs[1].dP[2][2][1] = 0.0364
dpolapp.Preclassifier.MBFinputs[1].dP[2][2][2] = 0.000357
dpolapp.Preclassifier.MBFinputs[1].dP[2][2][3] = 0
```

```
# Cross correlation coefficient in preclassifier:
# Data type used as input: RhoHV.
# MBF(RhoHV) is computed as JPOLE trapetzoid; an additive factor in rule
strength
# MBF: default initialization with parameter settings of Table 1 Ref.1
# RHOhv MBF trapetzoid for GC/AP.
dpolapp.Preclassifier.MBFinputs[2].dMBF[0][0]  =  0.5
dpolapp.Preclassifier.MBFinputs[2].dMBF[1][0]  =  0.6
dpolapp.Preclassifier.MBFinputs[2].dMBF[2][0]  =  0.9
dpolapp.Preclassifier.MBFinputs[2].dMBF[3][0]  =  0.95
# RHOhv MBF trapetzoid for bio scatter.
dpolapp.Preclassifier.MBFinputs[2].dMBF[0][1]  =  0
dpolapp.Preclassifier.MBFinputs[2].dMBF[1][1]  =  0
dpolapp.Preclassifier.MBFinputs[2].dMBF[2][1]  =  0.8
dpolapp.Preclassifier.MBFinputs[2].dMBF[3][1]  =  0.83
# RHOhv MBF trapetzoid for precipitation.
dpolapp.Preclassifier.MBFinputs[2].dMBF[0][2]  =  0.85
dpolapp.Preclassifier.MBFinputs[2].dMBF[1][2]  =  0.97
dpolapp.Preclassifier.MBFinputs[2].dMBF[2][2] = 1
dpolapp.Preclassifier.MBFinputs[2].dMBF[3][2] = 1.01
# Differential phase texture in preclassifier:
# The input data type is computed internally.
#--- MBF(Texture-1) is computed as JPOLE trapetzoid; an additive factor in
rule strength
# MBF: default initialization with parameter settings of Table 1 Ref.1
# PHIDP texture MBF trapetzoid for GC/AP.
dpolapp.Preclassifier.MBFinputs[3].dMBF[0][0] = 30
dpolapp.Preclassifier.MBFinputs[3].dMBF[1][0] = 40
dpolapp.Preclassifier.MBFinputs[3].dMBF[2][0] = 10800
dpolapp.Preclassifier.MBFinputs[3].dMBF[3][0] = 10800
# PHIDP texture MBF trapetzoid for bio scatter.
dpolapp.Preclassifier.MBFinputs[3].dMBF[0][1] = 8
dpolapp.Preclassifier.MBFinputs[3].dMBF[1][1] = 10
dpolapp.Preclassifier.MBFinputs[3].dMBF[2][1] = 40
dpolapp.Preclassifier.MBFinputs[3].dMBF[3][1] = 60
# PHIDP texture MBF trapetzoid for precipitation.
dpolapp.Preclassifier.MBFinputs[3].dMBF[0][2] = 0
dpolapp.Preclassifier.MBFinputs[3].dMBF[1][2] = 1
dpolapp.Preclassifier.MBFinputs[3].dMBF[2][2] = 15
dpolapp.Preclassifier.MBFinputs[3].dMBF[3][2] = 30
# Reflectivity texture in preclassifier:
# Data type is computed internally.
# MBF(Texture-2) is computed as JPOLE trapetzoid; an additive factor in the
rule strength
# MBF: default initialization with parameter settings of Table 1 Ref.1
# Zh texture MBF trapetzoid for GC/AP.
```

```
dpolapp.Preclassifier.MBFinputs[4].dMBF[0][0] = 2
dpolapp.Preclassifier.MBFinputs[4].dMBF[1][0] = 4
dpolapp.Preclassifier.MBFinputs[4].dMBF[2][0] = 10000
dpolapp.Preclassifier.MBFinputs[4].dMBF[3][0] = 10000
# Zh texture MBF trapetzoid for bio scatter.
dpolapp.Preclassifier.MBFinputs[4].dMBF[0][1] = 1
dpolapp.Preclassifier.MBFinputs[4].dMBF[1][1] = 2
dpolapp.Preclassifier.MBFinputs[4].dMBF[2][1] = 4
dpolapp.Preclassifier.MBFinputs[4].dMBF[3][1] = 7
# Zh texture MBF trapetzoid for precipitation.
dpolapp.Preclassifier.MBFinputs[4].dMBF[0][2] = 0
dpolapp.Preclassifier.MBFinputs[4].dMBF[1][2] = 0.5
dpolapp.Preclassifier.MBFinputs[4].dMBF[2][2] = 3
dpolapp.Preclassifier.MBFinputs[4].dMBF[3][2] = 6
# Other flags: none.
```

General settings of MeteoClassifiers

```
dpolapp.MeteoClassifiers.uiNonMeteoID = 1
```

This parameter defines what to do with the bins that fail the preceding quality consideration (`PreClassifier`), or fail the acceptance to any class of `MeteoClassifiers`.

- "`0`": `MeteoClassifiers` redirects the unaccepted and unclassified bins as IRIS/RDA '`CLASS_THRESHOLD`'
- "`1`": `MeteoClassifiers` redirects the unaccepted and unclassified bins as IRIS/RDA "`CLASS_NON_MET`"

```
dpolapp.MeteoClassifiers.dMinRS[0] = 0
```

Minimum rule strength for accepting rain

```
dpolapp.MeteoClassifiers.dMinRS[1] = 0
```

Minimum rule strength for accepting wet snow

```
dpolapp.MeteoClassifiers.dMinRS[2] = 0
```

Minimum rule strength for accepting snow

```
dpolapp.MeteoClassifiers.dMinRS[3] = 0
```

Minimum rule strength for accepting graupel

```
dpolapp.MeteoClassifiers.dMinRS[4] = 0
```

Minimum rule strength for accepting hail

```
dpolapp.MeteoClassifiers.dMinRS[5] = 0
```

Minimum rule strength for accepting rain/hail mixture (reported as hail)

Detailed settings of MeteoClassifiers

```
# MBFinputs and their use:
# Reflectivity in MeteoClassifiers:
# Data type used as input: Zh.
# MBF(dBZ) is computed as the CSU beta function.
# MBF: C-band, default settings:
# Zh beta MBF for rain: central value, width, slope
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[0][0] = 30
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[1][0] = 31
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[2][0] = 40
# Zh beta MBF for wet snow.
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[0][1] = 25
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[1][1] = 21
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[2][1] = 40
# Zh beta MBF for snow.
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[0][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[1][2] = 36
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[2][2] = 40
# Zh beta MBF for graupel/small hail.
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[0][3] = 45
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[1][3] = 11
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[2][3] = 20
# Zh beta MBF for large hail.
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[0][4] = 57.5
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[1][4] = 14
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[2][4] = 20
# Zh beta MBF for rain+hail mixture.
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[0][5] = 60
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[1][5] = 11
dpolapp.MeteoClassifiers.MBFinputs[0].dMBF[2][5] = 20
# Altitude in MeteoClassifiers:
# The IRIS/RDA estimate of the melting level height will be used.
# MBF(altitude;melting level) is computed as the CSU beta(ML) function.
# MBF: C-band, default settings:
# Altitude beta MBF for rain: central value, width, slope
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[0][0] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[1][0] = -0.2
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[2][0] = 20
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[3][0] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[4][0] = 0.5
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[5][0] = 5
# Altitude beta MBF for wet snow.
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[0][1] = -0.3
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[1][1] = 0.5
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[2][1] = 5
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[3][1] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[4][1] = 1
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[5][1] = 5
# Altitude beta MBF for snow.
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[0][2] = 10
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[1][2] = 10.2
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[2][2] = 60
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[3][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[4][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[5][2] = 0
# Altitude beta MBF for graupel/small hail.
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[0][3] = 10
```

```
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[0][3] = 10
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[1][3] = 12
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[2][3] = 60
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[3][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[4][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[5][3] = 0
# Altitude beta MBF for hail.
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[0][4] = 10
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[1][4] = 15
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[2][4] = 20
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[3][4] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[4][4] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[5][4] = 0
# Altitude beta MBF for rain+hail mixture.
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[0][5] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[1][5] = 0.5
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[2][5] = 20
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[3][5] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[4][5] = 0
dpolapp.MeteoClassifiers.MBFinputs[1].dMBF[5][5] = 0
# Differential reflectivity in MeteoClassifiers:
# Input data: Zdr (adjusted with Quality offset, internally in reingest), and
Zh
# MBF(Zdr) is computed as the CSU 2D-beta function of Zh.
# MBF: C-band, default settings:
# Zdr 2D beta MBF for rain at Zh=<Zh> dB (central value, width, slope):
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[0][0] = 0.48155
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[1][0] = 0.70578
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[2][0] = 10.958
# Polynomial dependence of rain MBF(Zdr) on Zh-<Zh>
# The center <Zh> and the Zh range in which the polynomial is applied:
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[3][0] = 30
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[4][0] = 30
# Polynomial coefficients; the linear, the 2nd, the 3rd, and the 4th order;
min/max constraints:
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][0][0] = 0.047498
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][0][1] = 0.0017624
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][0][2] = 6.993E- 06
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][0][3] = -4.4975E-07
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][0][4] = 0.1
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][0][5] = 5.5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][1][0] = 0.025917
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][1][1] = 0.00043621
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][1][2] = -4.8951E-06
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][1][3] = -5.6218E-08
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][1][4] = 0.4
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][1][5] = 8
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][2][0] = 0.22873
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][2][1] = 0.0054945
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][2][2] = -5.8275E-05
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][2][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][2][4] = 8
dpolapp.MeteoClassifiers.MBFinputs[2].dP[0][2][5] = 25
# Zdr 2D beta MBF for wet snow at Zh=<Zh> (central value, width, slope):
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[0][1] = 1.1031
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[1][1] = 1.3594
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[2][1] = 15
# Polynomial dependence of wet snow MBF(Zdr) on Zh-<Zh>
# The center <Zh> and the Zh range in which the polynomial is applied:
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[3][1] = 27.5
```

```
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[4][1] = 17.5
# Polynomial coefficients; the linear, the 2nd, the 3rd, and the 4th order;
min/max constraint:
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][0][0] = 0.004623
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][0][1] = 0.00064286
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][0][2] = 2.2222E-05
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][0][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][0][4] = 1.1
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][0][5] = 2.5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][1][0] = -0.0091144
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][1][1] = 0.00021429
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][1][2] = 3.8384E-05
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][1][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][1][4] = 1
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][1][5] = 2.5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][2][0] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][2][1] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][2][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][2][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][2][4] = 5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[1][2][5] = 20
# Zdr 2D beta MBF for snow at Zh=<Zh> (central value, width, slope):
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[0][2] = 0.25
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[1][2] = 0.75
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[2][2] = 8
# Zdr 2D beta MBF for graupel at Zh=<Zh> (central value, width, slope):
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[0][3] = 0.42969
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[1][3] = 0.92969
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[2][3] = 12.5
# Polynomial dependence of graupel MBF(Zdr) on Zh-<Zh>
# The center <Zh> and the Zh range in which the polynomial is applied:
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[3][3] = 42.5
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[4][3] = 12.5
# Polynomial coefficients; the linear, the 2nd, the 3rd, and the 4th orde;min/
max constraints:
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][0][0] = 0.033714
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][0][1] = 0.00096429
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][0][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][0][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][0][4] = 0.155
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][0][5] = 1.5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][1][0] = 0.033714
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][1][1] = 0.00096429
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][1][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][1][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][1][4] = 0.5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][1][5] = 2
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][2][0] = 0.25714
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][2][1] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][2][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][2][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][2][4] = 9
dpolapp.MeteoClassifiers.MBFinputs[2].dP[3][2][5] = 17
# Zdr weight and 2D beta MBF for hail at Zh=<Zh> (central value, width, slope):
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[0][4] = -0.75928
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[1][4] = 1.2778
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[2][4] = 15.859
# Polynomial dependence of hail MBF(Zdr) on Zh-<Zh>
# The center <Zh> and the Zh range in which the polynomial is applied:
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[3][4] = 57.5
```

```
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[4][4] = 12.5
# Polynomial coefficients; the linear, the 2nd, the 3rd, and the 4th order;min/
max constraint:
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][0][0] = -0.017336
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][0][1] = 0.0015104
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][0][2] = -8.3333E-05
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][0][3] = 4.1667E-06
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][0][4] = -2
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][0][5] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][1][0] = 0.0057788
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][1][1] = -0.0045313
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][1][2] = 2.7778E-05
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][1][3] = 1.25E- 05
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][1][4] = 0.1
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][1][5] = 1.3
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][2][0] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][2][1] = -0.14167
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][2][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][2][3] = 0.00066667
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][2][4] = 7
dpolapp.MeteoClassifiers.MBFinputs[2].dP[4][2][5] = 17
# Zdr weight and 2D beta MBF for rain+hail mixture at Zh=<Zh> (central value,
width, slope):
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[0][5] = 0.93571
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[1][5] = 2
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[2][5] = 15
# Polynomial dependence of rain+hail mixture MBF(Zdr) on Zh- <Zh>
# The center <Zh> and the Zh range in which the polynomial is applied:
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[3][5] = 60
dpolapp.MeteoClassifiers.MBFinputs[2].dMBF[4][5] = 10
# Polynomial coefficients; the linear, the 2nd, the 3rd, and the 4th order;
min/max constraint:
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][0][0] = -0.054167
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][0][1] = -0.0057143
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][0][2] = 0.00016667
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][0][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][0][4] = -0.5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][0][5] = 1.5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][1][0] = 0.0041667
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][1][1] = -0.02375
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][1][2] = -0.00016667
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][1][3] = 0.00015
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][1][4] = 0.5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][1][5] = 3
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][2][0] = -5.9212E-17
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][2][1] = 0.016667
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][2][2] = 5.9212E-19
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][2][3] = -0.00066667
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][2][4] = 5
dpolapp.MeteoClassifiers.MBFinputs[2].dP[5][2][5] = 20
# Specific differential phase in MeteoClassifiers:
# Input data: Kdp, and Zh.
# MBF(Kdp) is computed as the CSU 2D-beta function of Zh.
# MBF: C-band, default settings:
# Kdp 2D beta MBF for rain at Zh=<Zh> dB (central value, width, slope):
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[0][0] = 0.0079391
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[1][0] = -0.032435
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[2][0] = 11.154
# Polynomial dependence of rain MBF(Kdp) on Zh-<Zh>
# The center <Zh> and the Zh range in which the polynomial is applied:
```

```
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[3][0] = 30
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[4][0] = 35
# Polynomial coefficients; the linear, the 2nd, the 3rd, and the 4th order;
min/max constraint:
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][0][0] = -0.0049584
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][0][1] = 0.0020558
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][0][2] = 0.00019114
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][0][3] = 3.9888E-06
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][0][4] = 0.05
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][0][5] = 14
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][1][0] = 0.011085
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][1][1] = 0.0029526
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][1][2] = 0.00010385
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][1][3] = 6.1086E-07
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][1][4] = 0.2
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][1][5] = 8
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][2][0] = 0.22873
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][2][1] = 0.0054945
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][2][2] = -5.8275E-05
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][2][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][2][4] = 9
dpolapp.MeteoClassifiers.MBFinputs[3].dP[0][2][5] = 16
# Kdp 2D beta MBF for wet snow at Zh=25 dB: central value, width, slope.
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[0][1] = 0.25
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[1][1] = 1.2
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[2][1] = 10
# Kdp 2D beta MBF for snow at Zh=17.5 dB (central value, width, slope):
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[0][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[1][2] = 0.25
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[2][2] = 10
# Kdp 2D beta MBF for graupel at Zh=42.5 dB (central value, width, slope):
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[0][3] = 0.26875
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[1][3] = 0.76875
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[2][3] = 12.5
# Polynomial dependence of graupel MBF(Kdp) on Zh-<Zh>
# The center <Zh> and the Zh range in which the polynomial is applied:
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[3][3] = 42.5
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[4][3] = 12.5
# Polynomial coefficients; the linear, the 2nd, the 3rd, and the 4th order;
min/max constraint:
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][0][0] = 0.068704
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][0][1] = 0.001
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][0][2] = -0.00014815
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][0][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][0][4] = -0.25
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][0][5] = 1.5
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][1][0] = 0.068704
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][1][1] = 0.001
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][1][2] = -0.00014815
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][1][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][1][4] = 0.3
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][1][5] = 2
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][2][0] = 0.25714
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][2][1] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][2][2] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][2][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][2][4] = 9
dpolapp.MeteoClassifiers.MBFinputs[3].dP[3][2][5] = 17
# Kdp weight and 2D beta MBF for hail at Zh=57.5 dB (central value, width,
slope):
```

```
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[0][4] = 0.5
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[1][4] = 1
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[2][4] = 10
# Kdp weight and 2D beta MBF for rain+hail mixture at Zh=60 dB (central value,
width, slope):
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[0][5] = 2.3714
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[1][5] = 2.3971
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[2][5] = 15
# Polynomial dependence of rain+hail mixture MBF(Kdp) on Zh- <Zh>
# The center <Zh> and the Zh range in which the polynomial is applied:
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[3][5] = 60
dpolapp.MeteoClassifiers.MBFinputs[3].dMBF[4][5] = 20
# Polynomial coefficients; the linear, the 2nd, the 3rd, and the 4th order;
min/max constraint:
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][0][0] = 0.30833
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][0][1] = 0.0085714
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][0][2] = -0.00033333
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][0][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][0][4] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][0][5] = 10
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][1][0] = 0.29667
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][1][1] = 0.0088571
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][1][2] = -0.00026667
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][1][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][1][4] = -1
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][1][5] = 10
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][2][0] = -0.16667
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][2][1] = 2.4371E-17
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][2][2] = 0.0066667
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][2][3] = 0
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][2][4] = 8
dpolapp.MeteoClassifiers.MBFinputs[3].dP[5][2][5] = 25
# Cross correlation coefficient in MeteoClassifiers:
#--- Data type used as input: RhoHV.
#--- MBF(Rhohv) is computed as the CSU beta function.
# # MBF: C-band, default settings:
# RHOHV beta MBF for rain: central value, width, slope
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[0][0] = 1
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[1][0] = 0.04
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[2][0] = 10
# RHOHV beta MBF for wet snow.
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[0][1] = 0.88
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[1][1] = 0.11
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[2][1] = 20
# RHOHV beta MBF for snow.
```

```
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[0][2] = 1
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[1][2] = 0.06
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[2][2] = 10
# RHOHV beta MBF for graupel.
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[0][3] = 0.96
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[1][3] = 0.04
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[2][3] = 10
# RHOHV beta MBF for hail.
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[0][4] = 0.9
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[1][4] = 0.045
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[2][4] = 10
# RHOHV beta MBF for rain+hail mixture.
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[0][5] = 0.8
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[1][5] = 0.13
dpolapp.MeteoClassifiers.MBFinputs[4].dMBF[2][5] = 30
```

General settings of PrecipClassifier

```
dpolapp.PrecipClassifier.uiNonMeteoID = 1
```

This parameter defines what to do with the bins that fail the preceding quality consideration (`PreClassifier`), or fail the acceptance to any class of `PrecipClassifier`.

- "`0`": `PrecipClassifier` redirects the unaccepted and unclassified bins as IRIS/RDA '`CLASS_THRESHOLD`'
- "`1`": `PrecipClassifier` reports the unaccepted `PreClassifier` classifications unchanged, and reports the unclassified bins as IRIS/ RDA "`CLASS_NON_MET`"

```
dpolapp.PrecipClassifier.dMinRS[0] = 0
```

Minimum rule strength for accepting light rain.

```
dpolapp.PrecipClassifier.dMinRS[1] = 0
```

Minimum rule strength for accepting moderate rain.

```
dpolapp.PrecipClassifier.dMinRS[2] = 0
```

Minimum rule strength for accepting heavy rain.

```
dpolapp.PrecipClassifier.dMinRS[3] = 0
```

Minimum rule strength for accepting large drops.

Detailed settings of PrecipClassifier

```
# Precip classifier uses the textures of reflectivity and differential phase.
# MBF inputs and their use:
# Reflectivity in rain classifier:
# Input data: Zh.
# MBF(Zh) is computed as JPOLE trapetzoid; an additive factor in rule strength
# MBF: default initialization with parameter settings of Table 2 Ref.2
# Zh MBF trapetzoid for light precipitation.
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[0][0] = 5
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[1][0] = 10
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[2][0] = 35
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[3][0] = 40
# Zh MBF trapetzoid for moderate precipitation.
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[0][1] = 30
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[1][1] = 35
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[2][1] = 45
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[3][1] = 50
# Zh MBF trapetzoid for heavy precipitation.
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[0][2] = 40
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[1][2] = 45
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[2][2] = 75
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[3][2] = 80
# Zh MBF trapetzoid for large drops.
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[0][3] = 15
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[1][3] = 20
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[2][3] = 45
dpolapp.PrecipClassifier.MBFinputs[0].dMBF[3][3] = 50
# Differential reflectivity in rain classifier:
# Input data: Zdr (In reingest: adjusted with Quality offset, internally), and
Zh.
# MBF(Zdr;Zh) is computed as JPOLE 2D-trapetzoid; additive in rule strength
# Zdr MBF trapetzoid for light precipitation.
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[0][0] = -0.3
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[1][0] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[2][0] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[3][0] = 0.3
# The polynomial dependence of the Zdr light precipitation trapezoid on Zh
# The Zh range in which the polynomial is applied:
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[4][0] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[5][0] = 80
# Dependency on Zh: left tail constant term P0, linear P1, 2nd order P2, and
3rd order P3.
# The left shoulder is the same.
dpolapp.PrecipClassifier.MBFinputs[1].dP[0][0][0] = -0.5
dpolapp.PrecipClassifier.MBFinputs[1].dP[0][0][1] = 0.0025
dpolapp.PrecipClassifier.MBFinputs[1].dP[0][0][2] = 0.00075
dpolapp.PrecipClassifier.MBFinputs[1].dP[0][0][3] = 0
# The right tail constant term P0, linear P1, 2nd order P2 and 3rd order P3
# The right shoulder is the same.
dpolapp.PrecipClassifier.MBFinputs[1].dP[0][2][0] = 0.08
dpolapp.PrecipClassifier.MBFinputs[1].dP[0][2][1] = 0.0364
dpolapp.PrecipClassifier.MBFinputs[1].dP[0][2][2] = 0.000357
dpolapp.PrecipClassifier.MBFinputs[1].dP[0][2][3] = 0
# Zdr MBF trapetzoid for moderate precipitation.
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[0][1] = -0.3
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[1][1] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[2][1] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[3][1] = 0.3
```

```
# The polynomial dependence of the Zdr moderate precipitation trapezoid on Zh
# The Zh range in which the polynomial is applied:
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[4][1] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[5][1] = 80
# Dependency on Zh: left tail constant term P0, linear P1, 2nd order P2, and
3rd order P3.
# The left shoulder is the same.
dpolapp.PrecipClassifier.MBFinputs[1].dP[1][0][0] = -0.5
dpolapp.PrecipClassifier.MBFinputs[1].dP[1][0][1] = 0.0025
dpolapp.PrecipClassifier.MBFinputs[1].dP[1][0][2] = 0.00075
dpolapp.PrecipClassifier.MBFinputs[1].dP[1][0][3] = 0
# The right tail constant term P0, linear P1, 2nd order P2 and 3rd order P3
# The right shoulder is the same.
dpolapp.PrecipClassifier.MBFinputs[1].dP[1][2][0] = 0.08
dpolapp.PrecipClassifier.MBFinputs[1].dP[1][2][1] = 0.0364
dpolapp.PrecipClassifier.MBFinputs[1].dP[1][2][2] = 0.000357
dpolapp.PrecipClassifier.MBFinputs[1].dP[1][2][3] = 0
# Zdr MBF trapetzoid for heavy precipitation.
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[0][2] = -0.3
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[1][2] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[2][2] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[3][2] = 0.3
# The polynomial dependence of the Zdr heavy precipitation trapezoid on Zh
# The Zh range in which the polynomial is applied:
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[4][2] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[5][2] = 80
# Dependency on Zh: left tail constant term P0, linear P1, 2nd order P2, and
3rd order P3.
# The left shoulder is the same.
dpolapp.PrecipClassifier.MBFinputs[1].dP[2][0][0] = -0.5
dpolapp.PrecipClassifier.MBFinputs[1].dP[2][0][1] = 0.0025
dpolapp.PrecipClassifier.MBFinputs[1].dP[2][0][2] = 0.00075
dpolapp.PrecipClassifier.MBFinputs[1].dP[2][0][3] = 0
# The right tail constant term P0, linear P1, 2nd order P2 and 3rd order P3
# The right shoulder is the same.
dpolapp.PrecipClassifier.MBFinputs[1].dP[2][2][0] = 0.08
dpolapp.PrecipClassifier.MBFinputs[1].dP[2][2][1] = 0.0364
dpolapp.PrecipClassifier.MBFinputs[1].dP[2][2][2] = 0.000357
dpolapp.PrecipClassifier.MBFinputs[1].dP[2][2][3] = 0
# Zdr MBF trapetzoid for large drops.
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[0][3] = -0.3
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[1][3] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[2][3] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[3][3] = 0.3
# The polynomial dependence of the Zdr large drops trapezoid on Zh
# The Zh range in which the polynomial is applied:
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[4][3] = 0
dpolapp.PrecipClassifier.MBFinputs[1].dMBF[5][3] = 80
# Dependency on Zh: left tail constant term P0, linear P1, 2nd order P2, and
3rd order P3.
# The left shoulder is the same.
dpolapp.PrecipClassifier.MBFinputs[1].dP[3][0][0] = -0.5
dpolapp.PrecipClassifier.MBFinputs[1].dP[3][0][1] = 0.0025
dpolapp.PrecipClassifier.MBFinputs[1].dP[3][0][2] = 0.00075
dpolapp.PrecipClassifier.MBFinputs[1].dP[3][0][3] = 0
# The right tail constant term P0, linear P1, 2nd order P2 and 3rd order P3
# The right shoulder is the same.
dpolapp.PrecipClassifier.MBFinputs[1].dP[3][2][0] = 0.08
dpolapp.PrecipClassifier.MBFinputs[1].dP[3][2][1] = 0.0364
dpolapp.PrecipClassifier.MBFinputs[1].dP[3][2][2] = 0.000357
```

```
dpolapp.PrecipClassifier.MBFinputs[1].dP[3][2][3] = 0
# Cross correlation coefficient in rain classifier:
# Data type used as input: RhoHV.
# MBF(RhoHV) is computed as JPOLE trapetzoid; an additive factor in rule
strength
# MBF: default initialization with parameter settings of Table 2 Ref.1
# RHOhv MBF trapetzoid for light precipitation.
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[0][0] = 0.85
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[1][0] = 0.97
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[2][0] = 1
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[3][0] = 1.01
# RHOhv MBF trapetzoid for moderate precipitation.
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[0][1] = 0.85
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[1][1] = 0.97
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[2][1] = 1
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[3][1] = 1.01
# RHOhv MBF trapetzoid for heavy precipitation.
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[0][2] = 0.85
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[1][2] = 0.97
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[2][2] = 1
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[3][2] = 1.01
# RHOhv MBF trapetzoid for large drops.
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[1][3] = 0.97
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[0][3] = 0.85
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[2][3] = 1
dpolapp.PrecipClassifier.MBFinputs[2].dMBF[3][3] = 1.01
# Differential phase texture in rain classifier:
# The input data type is computed internally.
#--- MBF(Texture-1) is computed as JPOLE trapetzoid; an additive factor in
rule strength
# MBF: default initialization with parameter settings of Table 2 Ref.1
# PHIDP texture MBF trapetzoid for light precipitation.
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[0][0] = 0
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[1][0] = 1
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[2][0] = 15
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[3][0] = 30
# PHIDP texture MBF trapetzoid for moderate precipitation.
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[0][1] = 0
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[1][1] = 1
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[2][1] = 15
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[3][1] = 30
# PHIDP texture MBF trapetzoid for heavy precipitation.
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[0][2] = 0
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[1][2] = 1
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[2][2] = 15
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[3][2] = 30
# PHIDP texture MBF trapetzoid for large drops.
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[0][3] = 0
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[1][3] = 1
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[2][3] = 15
dpolapp.PrecipClassifier.MBFinputs[3].dMBF[3][3] = 30
# Reflectivity texture in rain classifier:
# Data type is computed internally.
# MBF(Texture-2) is computed as JPOLE trapetzoid; an additive factor in the
rule strength
# MBF: default initialization with parameter settings of Table 2 Ref.1
# Zh texture MBF trapetzoid for light precipitation.
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[0][0] = 0
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[1][0] = 0.5
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[2][0] = 3
```

```
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[3][0] = 6
# Zh texture MBF trapetzoid for moderate precipitation.
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[0][1] = 0
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[1][1] = 0.5
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[2][1] = 3
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[3][1] = 6
# Zh texture MBF trapetzoid for heavy precipitation.
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[0][2] = 0
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[1][2] = 0.5
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[2][2] = 3
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[3][2] = 6
# Zh texture MBF trapetzoid for large drops.
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[0][3] = 0
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[1][3] = 0.5
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[2][3] = 3
dpolapp.PrecipClassifier.MBFinputs[4].dMBF[3][3] = 6
```

Settings of CellClassifier

```
dpolapp.CellClassifier.dMinRS[0] = 0.5
```

Minimum rule strength for accepting convective rain.

```
# MBFinputs and their use:
# Reflectivity&Height in cell classifier:
# Input data: Zh and height difference w.r.t. the current 0oC isotherm.
# MBF(Zh,height) is computed as 2D product of trapetzoids; summed up to rule
strength.
# Min Zh required for convection (full fuzzy strength will be +5 dBZ from min).
dpolapp.CellClassifier.MBFinputs[0].dMBF[0][0] = 25
# Min height w.r.t. 0oC isotherm (full fuzzy strength will be +1. km higher ).
dpolapp.CellClassifier.MBFinputs[0].dMBF[4][0] = 0
```

9.3.7 HydroClass FAQ

The following list is a compilation of issues encountered when using HydroClass in the field.

Q: How does HydroClass treat typical failure cases related to *dpolapp.conf*, such as:

- Configuration file missing?
  A: switches off until restart of IRIS/RDA.
- Syntax errors in ***dpolapp.conf*** parameter lines?
  A: switches off until file is fixed.

Q: How to feed in the melting level (ML) value as an external input to HydroClass?

A: When operating:

- RVP10 with IRIS/Radar **`setup_change -load`** tool, see SRI Product documentation
- RVP10 without IRIS, contact Vaisala
- IRIS/reingest, the ML stored in RAW is used

To use these momentary values of ML, HydroClass configuration *dpolapp.conf* must be set up with:

```
dpolapp.Quality.iAux[0] = 0
```

Alternatively, you can edit and update (either manually or with customized script tools)

```
dpolapp.Quality.iAux[0] = 1
```

and

```
dpolapp.Quality.iAux[1] = 3500
```

(where 3500 represents an example ML value, meters above mean sea level)

Q: How to monitor the ML value used/in use?

A: When operating in IRIS/Radar, use the utility command:

`setup_change – list | grep ifalls`.

When operating plain RVP10, use **`dspx > Mp`**

Q: When running HydroClass in distributed PC architectures such as RVPO900/AMR:

- Where is the HydroClass configuration in use?
  A: At PC running IRIS/Radar
- Where to update the melting level with `setup_change`?
  A: At IRIS/Radar

Q: How do HClass data look when HydroClass is tested using noise source input?

A: In commissioning phase and anytime, a viable method of testing functionality of HydroClass is to couple a coherent noise source to RVP10 IF inputs (for example, AMR built-in noise).

HydroClass can process such a data stream, by configuring RVP10 appropriately.

Observing **`HClass`** data stream with ascope, for example, see the data in the following figure.

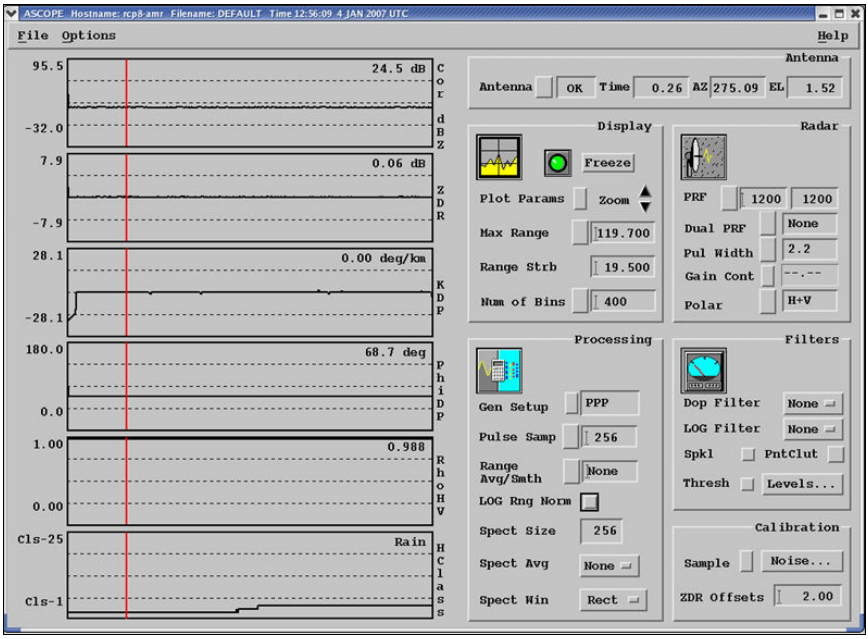


Figure 56 Polarimetric measurements with HClass in an Ascope view when RVP10 is processing artificial noise inputs (correlated IF Signal in both receiver ports H and V)

HydroClass is in default configuration and uses the melting layer height value given in the configuration file (2300 m).

9.3.8 Additional Notes about HydroClass

- There are limitations in ML height settings when running ascope; it is possible to use the default value of ML in RVP10, or a specified value in ***dpolapp.conf*** (dynamic update does not work).
- There are limitations in ML settings when running time series data in play-back. RVP10 uses the ML value stored on TS data.
- The HydroClass ML is defined as the 0 °C isotherm. See CellClassifier (page 262) . The melting layer (or the radar bright band) is typically located at altitudes below ML.

9.3.9 Parametrized Fuzzy Parameters

The hydrometeor classification membership functions are parametrized either as 1D or 2D beta functions.

The evolution of the 2D membership beta function parameters y=y(Z_H) where y=m, a, or b were optimized using the CSU approach.

The parameter dependencies are expressed as fifth order polynomials, with coefficients listed in Table 63 (page 290) to Table 65 (page 291).

Parametrizing 2D membership function with polynomials simplifies the implemented software, as the approach is common to CSU and the JPOLE algorithms.

The following table shows the beta function parameters of all the 1D membership functions. It also gives insight into the 2D membership functions, expressing their projections at the reference levels of the auxiliary input variables. For example, the parameter values of 2D membership functions of MBF(Z_{dr}, Z_H) visualize the shape of the MBF at Z_H = 0 dBZ.

Table 62 Parameters for 1D and 2D Membership Functions at Specified Reference Planes of the Second Function Variable

MBF	Hydrometeor Class ID	m	a	b
MBF(Z_H [dB]) 1D				
-	1) rain	30.0	31.0	40.0
-	2) wet snow	25.0	21.0	40.0
-	3) snow	0.0	36.0	40.0
-	4) graupel/small hail	45.0	11.0	20.0
-	5) hail	57.5	14.0	20.0
-	6) hail, rain mixture	60.0	11.0	20.0
MBF(height [km]) 2D @ melting level height (MSL) = 2.5 km (warm season)				
-	1) rain	0.0	2.3	5.0
-	2) wet snow	2.2	0.5	5.0
-	3) snow	10.0	7.7	60.0
-	4) graupel/small hail	10.0	9.5	60.0
-	5) hail	0.0	15.0	20.0
-	6) hail, rain mixture	0.0	3.0	20.0
MBF(height [km]) 1D @ melting level height < 0 km (cold season, storm height =5 km)				
-	1) rain	0.0	0.5	5.0
-	2) wet snow	0.0	2.5	5.0
-	3) snow	10.0	15.0	20.0
-	4) graupel/small hail	10.0	15.0	20.0
-	5) hail	10.0	15.0	20.0

MBF	Hydrometeor Class ID	m	a	b
-	6) hail, rain mixture	10.0	15.0	20.0
MBF(Z_{dr} [dB]) 2D				
@ Z_H = 30 dBZ	1) rain	0.5	0.7	10.0
@ Z_H = 27.5 dBZ	2) wet snow	1.1	1.4	15.0
@ Z_H = 0 dBZ	3) snow	0.5	0.75	8.0
@ Z_H = 42.5 dBZ	4) graupel/small hail	0.43	0.9.	12.5
@ Z_H = 7.5 dBZ	5) hail	-0.76	1.28	15.9
@ Z_H = 60.0 dBZ	6) hail, rain mixture	0.94	2.0	15.0
MBF(K_{dp} [dgr/km]) 2D				
@ Z_H = 30 dBZ	1) rain	0.05	0.2	11.1
@ Z_H = 25 dBZ	2) wet snow	0.25	1.2	10.0
@ Z_H = 7.5 dBZ	3) snow	0.0	0.25	10.0
@ Z_H = 42.5 dBZ	4) graupel/small hail	0.27	0.77	12.5
@ Z_H = 57.5 dBZ	5) hail	0.5	1.0	10.0
@ Z_H = 60.0 dBZ	6) hail, rain mixture	2.37	2.40	15.0
MBF (Rho_{HV} []) 1D				
	1) rain	1.00	0.04	10.0
-	2) wet snow	0.88	0.11	20.0
-	3) snow	1.0	0.06	10.0
-	4) graupel/small hail	0.96	0.04	10.0
-	5) hail	0.90	0.045	10.0
-	6) hail, rain mixture	0.80	0.13	30.0

In the following tables, parametrization is expressed with 5th order polynomials of the difference with respect to the reference planeo of the quoted auxiliary observable. The polynomial coefficients `P0` ... `P4` are in increasing order, `P0` values shown in Table 62 (page 288). The **Max/Min** column expresses the boundaries that the parametrized polynomial values must fall within.

Table 63 Parametrized Evolution of the Parameter m of the 2D Membership Functions

m	ID	P1	P2	P3	P4	Max/Min
MBF(height [km]) (warm season)						
-	1)	0.0	0	0	0	-
-	2)	0.0	0	0	0	-
ML-2.5	3)	1.0	0	0	0	-
-	4)	0.0	0	0	0	-
-	5)	0.0	0	0	0	-
-	6)	0.0	0	0	0	-
MBF(Z_{dr} [dB])						
(Z_H-30) ±30 dBZ	1)	4.75E-02	1.76E-03	6.99E-06-	4.50E-07	5.5/0.1
(Z_H-27.5) ±17.5 dBZ	2)	4.62E-03	6.42E-04	2.22E-05	0.0000E+0	02.5/1.1
(Z_H-0) ±327 dBZ	3)	0	0	0	0	0.5/-0.5
(Z_H-42.5) ±12.5 dBZ	4)	3.37E-02	9.64E-04	0	0	1.5/0.155
(Z_H-57.5) ±12.5 dBZ	5)	-1.73E-02	1.51E-03	-8.33E-05	4.17E-06	5.5/0.1
(Z_H- 60) ±10 dBZ	6)	-5.41E-02	-5.71E-03	1.67E-04	0	5.5/0.1
MBF(K_{dp} [dgr/km])						
(Z_H-30) ±35 dBZ	1)	-4.96E-03	2.06E-03	1.91E-04	3.99E-06	14/0.05
(Z_H-27.5) ±17.5 dBZ	2)	0	0	0	0	0.25/0.25
(Z_H-0) ±327 dBZ	3)	0	0	0	0	0/0
(Z_H-42.5) ±12.5 dBZ	4)	6.87E-02	1.00E-03	-1.48E-040	0	2.0/0.3
(Z_H-57.5) ±12.5 dBZ	5)	0	0	0	0	0.5/0.5
(Z_H- 60) ±10 dBZ	6)	3.08E-01	8.57E-03	3.33E-04	0	10./0.0

Table 64 Parametrized Evolution of the Parameter a of the 2D Membership Functions

a	ID	P1	P2	P3	P4	Max/Min
MBF(height [km]) (warm season)						
-	1)	1.0	0	0	0	-
-	2)	-1.0	0	0	0	-
ML-2.5	3)	0.0	0	0	0	-
-	4)	-1.0	0	0	0	-

a	ID	P1	P2	P3	P4	Max/Min
-	5)	0.0	0	0	0	-
-	6)	1.0	0	0	0	-
MBF(Z_{dr} [dB])						
(Z_H-30) ±30 dBZ	1)	2.59E-02	4.36E-04	-4.89E-06	-5.62E-08	8/0.4
(Z_H-27.5) ±17.5 dBZ	2)	-9.11E-03	2.1429E-04	3.84E-05	0.0000E +00	2.5/1.1
(Z_H-0) ±327 dBZ	3)	0	0	0	0	0.5/-0.5
(Z_H-42.5) ±12.5 dBZ	4)	3.37E-02	9.64E-04	0	0	2.0/0.5
(Z_H-57.5) ±12.5 dBZ	5)	5.78E-03	4.53E-03	2.78E-05	1.25E-05	0/-2
(Z_H- 60) ±10 dBZ	6)	4.17E-03	-2.38E-02	-1.67E-04	1.50E-04	3.0/0.5
a	ID	P1	P2	P3	P4	Max/Min
MBF(K_{dp} [dgr/km])						
(Z_H-30) ±35 dBZ	1)	1.11E-02	2.95E-03	1.04E-04	6.11E-07	8/0.2
(Z_H-27.5) ±17.5 dBZ	2)	0	0	0	0	0.4/0.4
(Z_H-0) ±327 dBZ	3)	0	0	0	0	0.25/0.25
(Z_H-42.5) ±12.5 dBZ	4)	6.87E-02	1.00E-03	-1.48E-04	0	2.0/0.3
(Z_H-57.5) ±12.5 dBZ	5)	0	0	0	0	1/1
(Z_H- 60) ±10 dBZ	6)	2.97E-01	8.86E-03	2.67E-04	0	10./0.1

Table 65 Parametrized Evolution of the Parameter b of the 2D Membership Functions

b	ID	P1	P2	P3	P4	Max/Min
MBF(height [km]) (warm season)						
-	1)	0	0	0	0	-
-	2)	0	0	0	0	-
ML-2.5	3)	0	0	0	0	-
-	4)	0	0	0	0	-
-	5)	0	0	0	0	-
-	6)	0	0	0	0	-
MBF(Z_{dr} [dB])						
(Z_H-30) ±30 dBZ	1)	2.29E-01	5.49E-03	-5.83E-05	0	25/8

b	ID	P1	P2	P3	P4	Max/Min
(Z_H-27.5) ±17.5 dBZ	2)	0	0	0	0	20/5
(Z_H-0) ±327 dBZ	3)	0	0	0	0	8/8
(Z_H-42.5) ±12.5 dBZ	4)	0	0	0	0	12.5/12.5
(Z_H-57.5) ±12.5 dBZ	5)	2.103E-18-	1.42E-01	2.47E-19	6.67E-04	17/7
(Z_H- 60) ±10 dBZ	6)	-5.92E-17	1.67E-02	5.92E-19	-6.67E-04	20/5
MBF(K_{dp} [dgr/km])						
(Z_H-30) ±35 dBZ	1)	2.29E-01	5.49E-03	-5.83E-05	0	16./9.
(Z_H-27.5) ±17.5 dBZ	2)	0	0	0	0	10/10
(Z_H-0) ±327 dBZ	3)	0	0	0	0	10/10
(Z_H-42.5) ±12.5 dBZ	4)	2.57E-01	0	0	0	17.0/9
(Z_H-57.5) ±12.5 dBZ	5)	0	0	0	0	10/10
(Z_H- 60) ±10 dBZ	6)	-1.67E-01	2.44E-17	6.67E-03	0	25/8.0

9.4 Melting level height detection

9.4.1 **MLHGT**: Melting level height

The melting level bright band typically represents the location of frozen hydrometeors falling into air warmer than the freezing level, where the particles start to melt.

The height of the melting level is an important aspect to understanding what is happening micro-physically within precipitation systems. Changes of melting level height can assist in determining the likelihood of hail and lightning, and is related to estimating strength of downdrafts due to phase change from ice to liquid. During winter conditions determining the location of liquid versus ice precipitation at the surface is improved with knowledge of the melting level, especially in mountainous terrain.

The presence of the bright band associated with the melting level must be accounted for with quantitative interpretations of weather observations.

In IRIS and RDA software, the **SRI** product uses the melting level height to perform Vertical Reflectivity Profile corrections, to apply Fall Speed corrections to radial velocity, to distinguish between convective and non-convective precipitation, to account for the different attenuation between liquid and ice, and as a membership function in the Hydrometeor Classification function. Thus an automated determination of the melting level height and how it varies spatially and temporally adds value to the overall weather radar system software.

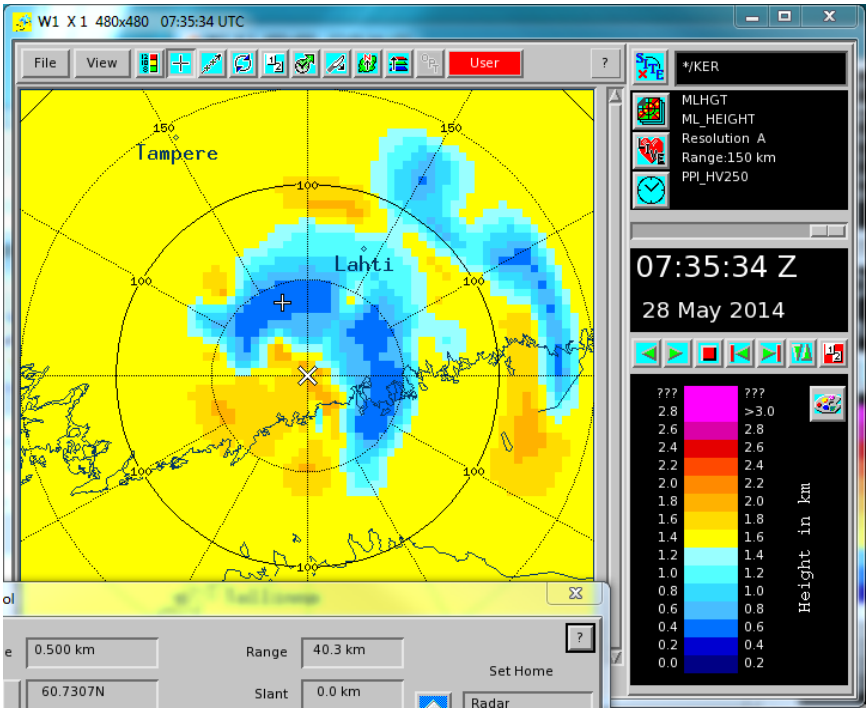


Figure 57 Melting level height detection

The **MLHGT** product produced by IRIS is a depiction of the melting level heights on a Cartesian map. The melting level heights are determined on a user defined grid providing an illustration how the melting level heights changes geographically.

Dual Polarization radar data is used as input along with a priori information. As with other IRIS products, the **MLHGT** product may be animated in time and overlayed with other data to increase situational awareness of the precipitation structure in the atmosphere.

9.4.2 **MLHGT** algorithm

IRIS dual polarization raw volume data is used as input for the **MLHGT** algorithm. This data may come from either single **PPI**, volume, **RHI**, or sector scans. The algorithm also functions independently from scan geometry matching data from the scan polar co-ordinate system to an Earth relative co-ordinate system. However as with any radar observation, a higher number of elevations in a volume scan provides the best results.

The following figure shows the **MLHGT** algorithm flow diagram. The top level illustrates the radar observation inputs, followed by prior information. The blocks represent consecutive functional steps. A key qualifier for each step is expressed in parenthesis.

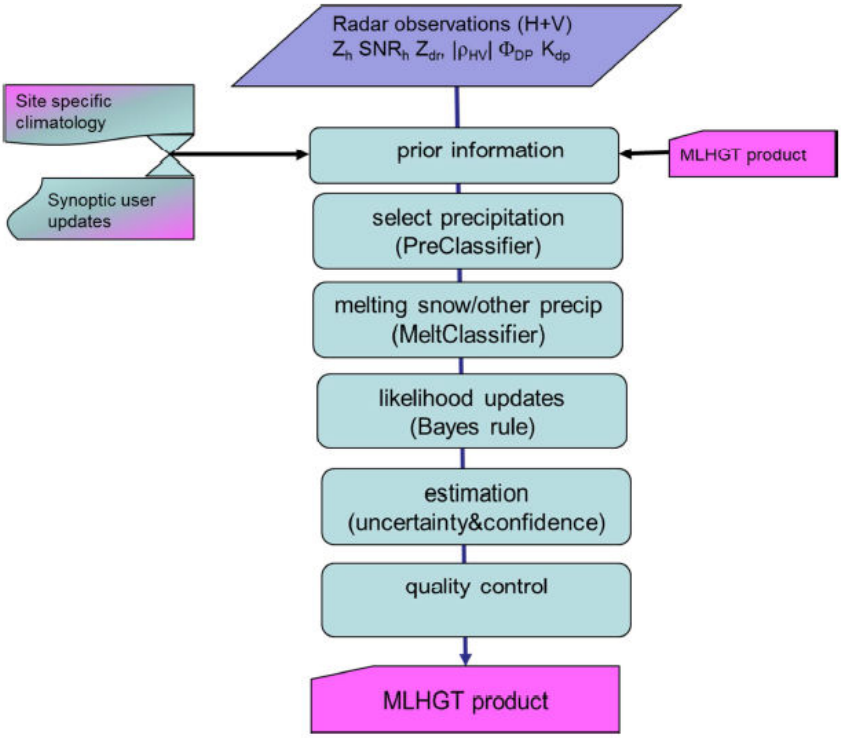


Figure 58 MLHGT algorithm flow diagram

The first step in the processing is to ensure only precipitation particles are being considered for the melting level height estimates. The **`PreClassifier`** tags each range bin as being from precipitation or non-precipitation targets. The non-precipitation flagged bins are removed from the considered data.

The precipitating bins are then classified into 'melting snow' or 'other precipitation' by a dedicated fuzzy algorithm known as **`MeltClassifier`**. **`MeltClassifier`** is similar to the **`MeteoClassifier`** used in **`HydroClass`**. However, the **`MeltClassifier`** substitutes a Signal to Noise Ratio (SNR) Membership function (MBF) in place of the melting level height MBF used by the **`MeteoClassifier`**. The SNR MBF is configured for low SNR to be part of the 'melting snow' and high SNR in 'other precipitation'. Other particle types such as rain, dry snow, hail, and graupel which are normally characterized or also grouped into the 'other precipitation' class.

The statistical method known as Bayesian Rule is used to infer the likelihood of a region being either 'melting snow' or 'other precipitation', using a priori information and posterior conditioning. The a priori information is the first estimate of the 0 °C heights from an outside source. This could be automatically inserted from radiosonde or NWP data, or could come from the climatological values found in IRIS/Setup.

The likelihood of the **MLHGT** can be updated by each subsequent radar observation over time building up a confidence in the estimate. If there are a high number of 'melting snow' or 'other precipitation' bins in a region, and the ratio of one condition versus the other is high, confidence increases quickly. Likewise, if there is a low number of classifications in the region or if the ratio of 'melting snow' to 'other precipitation' is almost equal, the confidence remains low. Once the minimum confidence is reached the melting level height form MLHGT is assigned to that region, else the height remains to be the climatological value. The Bayesian inference can be conceptually understood as constructing a display of vertical cross-sections in which the most likely position of the ML can be visually recognized by inspection of the `MeltClassifier` decisions.

The regions are presented as columns of data in an Earth co-ordinate system. The user is allowed to configure the vertical resolution and azimuthal size of each region. The columns size in range domain are scaled to match their size in azimuth. The columns range extent grows proportionally as the size in azimuth as distance from the radar increases. Thus the columns are effectively the same shape throughout the domain. This is advantageous because the statistics is improved at farther distances, due to an increasing number of data points, which compensates for the loss in resolution by beam broadening.

10. Calibration and data quality

10.1 Thresholding

RVP10 can accept or reject incoming real time data to remove range bins. Rejected data may include that with:

- Weak signal power
- Unreliable estimates of Doppler parameters
- Polarimetric parameters that suggest echo is of undesired origin, for example not precipitation

This process is called thresholding. IRIS provides flexibility for thresholding to provide clean displays, promote efficient execution and transmission of the products, and reduce the amount of disk space required to hold compressed data and product archives.

10.1.1 Threshold qualifiers

For data quality control, each RVP10 output parameter can be qualified, that is, accepted or rejected for output, based on threshold criteria:

Table 66 Threshold quality criteria

ID	Criterion name	Pass criterion
`LOG`	(Signal+Noise)-to-Noise Ratio	`LOG` > threshold
`SQI`	Signal Quality Index	`SQI` > threshold
`CCOR`	Clutter Correction	`CCOR` > threshold
`SIG`	Weather Signal Power	`SIG` > threshold
`PMI`	Polarimetric Meteo Index	`PMI` > threshold

Each qualification criterion can be switched on and off independently, and the threshold levels (for example, $\texttt{SQI}_{\texttt{thresh}}$) can each be set independently. Also, each qualifier test can be OR'd with any other. This allows very complex threshold criteria to be constructed as required. The following table shows the threshold qualifiers.

Table 67 Threshold qualifiers

Threshold qualifier	Description	Default value
`LOG`	A measure of signal strength that is usually used for the thresholding of reflectivity data.	0.75 dB

Threshold qualifier	Description	Default value
SQI	Typically used for velocity and width thresholding since it is a measure of the coherency. It is a number between 0 ... 1 (dimensionless), where 0 is perfect white noise and 1 is a pure tone (perfect Doppler signal).). It may also be used to identify and remove 2nd trip echo and RF interference.	0.4
CCOR	The clutter correction threshold is typically used to reject measurements when the clutter in a range bin is very strong compared to the non-clutter signal (that is, when the calculated CCOR is a large negative number in dB). The appropriate value depends on the coherency of the radar system. Threshold values lower than the default (more negative) reject fewer clutter bins. Threshold values closer to 0 reject more clutter bins.	-18 dB.
SIG	Typically used only for thresholding the spectrum width to assure that the signal power is strong enough for an accurate width measurement. If R_2 processing is used, this can usually be reduced to 5 dB for width thresholding.	5 dB
PMI	Typically used to reject echoes that are identified as inconsistent with the preferred hypothesis of precipitation. Identification is based on combined interpretation of polarimetric measurements of reflectivity, differential reflectivity, differential phase and co-polar correlation coefficient by the HydroClass pre-classifier. PMI is typically applied to reflectivity data.	0.45

The following table shows the default threshold combinations for each of the parameters that can be selected for output from RVP10:

Table 68 Default threshold combinations

Parameter	Description	Threshold
dBZ	Reflectivity with clutter correction	LOG , CCOR, SQI,PMI
dBT	Reflectivity without clutter correction	LOG
V	Mean velocity	SQI , CCOR
W	Spectrum width	SQI , CCOR, SIG
Dual Pol	Differential reflectivity	LOG

10.1.2 Adjusting threshold qualifiers

When optimizing thresholds for your application, it is recommended that you change only one parameter (level or criterion) at a time so that you can verify the effect.

The following table shows some tips for optimizing the levels for the default criteria.

Table 69 Adjusting threshold qualifiers

Qualifier	Adjustment
`LOG`	To optimize the LOG level, display dBT or dBZ and select the lowest value of the threshold that eliminates the display noise. If the LOG level is set too high you lose sensitivity. Note that if you average more pulses or ranges, then the threshold level can usually be reduced. The appropriate value is generally unique to each pulse width selection.
`SQI`	To optimize the SQI level, display velocity and select the lowest value of the threshold that eliminates the display noise. If the SQI level is set too high you lose sensitivity. In general, you should see a greater area covered by velocity than reflectivity since the velocity is more sensitive. If you do not, you should reduce your SQI threshold. Note that if you average more pulses or ranges, then the threshold level can usually be reduced. Some users prefer to use just LOG threshold for velocity so that the area covered matches reflectivity.
`CCOR`	This is used to eliminate clutter targets that are very strong. It should not be set to eliminate all clutter targets on a clear day since this means that you are losing sensitivity. To optimize the CCOR threshold it is best to know your system coherency in terms of dB of clutter cancelation. Start at a value of 10 dB greater (closer to 0) than this. Now display a PPI of dBZ at an antenna elevation of approximately1 degree. The display should be relatively clean of any clutter targets since most are rejected. Now reduce the CCOR (more negative) to increase the number of clutter targets on the display until the number of clutter targets does not increase. The optimum value of the CCOR is approximately 5 dB more (closer to zero) than this point. For example, if the number of clutter targets is a maximum at -35 dB, then set the CCOR to approximately - 30 dB. Note that your clutter filter selection effects the result.
`SIG`	This should be done last. To optimize the SIG level, display the width W and select the lowest value of the threshold that eliminates the display noise. If the SIG level is set too high you lose sensitivity. If you average more pulses or ranges, then the threshold level can usually be reduced.

Qualifier	Adjustment
PMI	To optimize the PMI level, consider data acquired in the mode of (H+V) in PPP processing. Having first optimized the previous four Doppler qualifiers, inspect echo classification data of DB_HCLASS and seek for gates declared as “NoMet”, which are unlikely of meteorological origin. These bins may appear as “NoMet” data in your display, or DB_HCLASS data might be readily thresholded, depending on your color scales and the HydroClass configuration. In order to get the other data types thresholded in the same fashion, activate PMI as a thresholding mechanism in task configuration. The PMI threshold value to 0.45 implies the same strength of suppression to other selected data types as seen in DB_HCLASS. It is possible to recover more precipitation data, typically at edges of precipitation and the most far echoes (virga) by reducing the PMI threshold, as appropriate. In these customizations, the behavior of DB_HCLASS remains unchanged.

Secondary SQI threshold

When thresholding dual pol, dBZ, and dBT reflectivity data with **SQI** , the comparison value for accepting those data is the secondary **SQI** threshold that is defined in a slope and offset from the primary user value. See **Mf**— Clutter filters (page 108).

The secondary threshold is more permissive (lower valued), and is traditionally used to qualify LOG data only in the Random Phase processing mode.

The secondary **SQI** threshold is applied uniformly in all processing modes when dual pol or reflectivity data are specified as being thresholded by **SQI**.

This gives you more freedom in applying an **SQI** threshold to your **LOG** data, because the cutoff value for dual pol and reflectivity can be chosen independently from the cutoff value for the other Doppler parameters. The full **SQI** test would not normally be applied to LOG data, because of the so-called "black hole" problem, which is the loss of **LOG** data within regions of high shear, even though, for instance, the reflectivity itself was strong. You can experiment with applying a secondary **SQI** threshold to help clean up the **LOG** data, without introducing any significant black holes.

10.1.3 Speckle filters applied to the thresholded data

A speckle filter is a final pass over each output ray, in which isolated, single bins of velocity, width, or intensity are removed.

There are two speckle removers in RVP10:

- 1D single-ray speckle filter (default)—This is used for any output parameter.
- 2D 3x3 speckle filter—If enabled, this is used for any output parameter.

This eliminates single pixel speckles, which allows the thresholds to be reduced for greater sensitivity with fewer false alarms (speckles).

Both speckle filters remove isolated data points that are likely to be noise, interference, aircraft, birds, or other point targets. Meteorological targets typically occupy multiple range bins, so they are not affected by the speckle filters. The benefits of using a speckle filter are:

- Displays look "cleaner" to observers

- Thresholds can be set slightly more sensitive without increasing the number of noise pixels

The 1D and 2D speckle filters are enabled using the **soprm** command, `Input 2` (see Setup operating parameters (**SOPRM**) (page 368)). If both are turned on, the 1D filter is applied first.

10.1.3.1 1D speckle filter

A ray is the basic azimuth unit of RVP10 (for example, 1°) over which the samples are averaged to obtain the output base data (`T`, `Z`, `V`, `W`).

For this filter, a speckle is defined as any single, valid bin (not thresholded), having thresholded bins on either side of it in range. Any such isolated bin in a ray is set to "threshold".

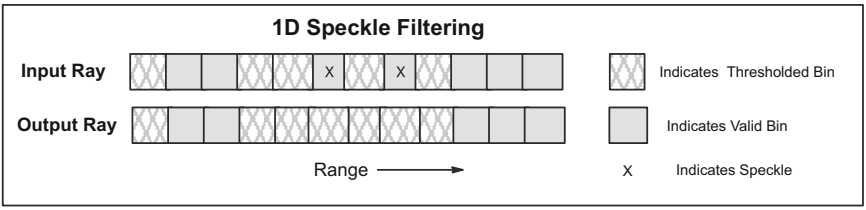


Figure 59 1D Speckle Filtering Algorithm

Table 70 1D Speckle Filters

Purpose	Relevant data types
Reflectivity data	`dBT`, `dBZ`, `ZDR` `LDR` when using dual-pol
Doppler data	`V`, `W`, `PhiDP`, `RhoHV` `KDP` when using dual-pol

When either speckle filter is switched on, it is applied in `HydroClass`.

Each filter must be switched on or off, depending on the nature of the targets being observed. For example, when making a clutter map of the area, it is recommended that you switch both speckle filters off.

10.1.3.2 2D 3x3 Speckle Filter

This is a 3x3 azimuth/range filter to remove single point targets and interpolate single point gaps surrounded by valid data.

The 2D 3x3 speckle filter performs data filling of "missing speckles" as well as eliminating isolated speckle bins.

The filter examines 3 adjacent range bins from 3 successive rays to assign a value to the center point. For each output point, its 8 neighboring bins in range and time are available to the filter. Only the dBZ, dBT, Velocity, and Width data are candidates for this filtering step. All other parameters are processed using the default 1D speckle filter.

Table 71 2D 3x3 Speckle Filter Rules

	Center Point Action	
	Assign Threshold	**Else**
Valid Center Point	If there are none or only one other valid point in the 3x3.	Do Nothing. Pass the center point value as-is.
Thresholded Center Point	If there are five or fewer valid neighbors in the 3x3.	If there are six or more valid neighbors in the 3x3, average to fill the center point.

The 2D 3x3 filter fills by interpolation and thresholdw isolated noise bins. The following figure shows some examples.

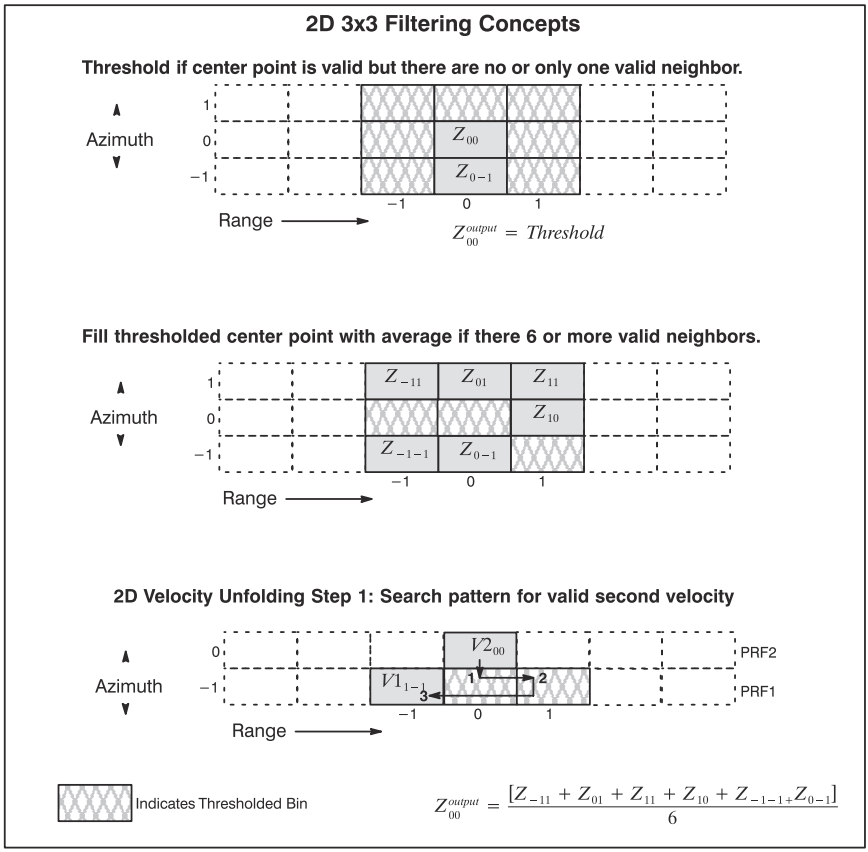


Figure 60 2D 3x3 Filtering Concept Examples

For all the parameters except velocity, the interpolated value for filling is computed as the arithmetic mean of all available neighbors. The procedure for velocity is similar, except that the 8-bit angles are first converted to Cartesian vectors, then averaged and converted back to polar.

The 2D 3x3 speckle filter can be used with or without the 1D speckle filtering. However, the data is cleaner if both filters are applied.

The 2D 3x3 speckle filter has some interesting properties when combined with other algorithms.

10.1.3.2.1 Dual-PRF unfolding

Dual-PRF velocity unfolding is computed within the 3x3 filter when both are enabled. There are two steps to the process:

1. The most recent and the previous ray are used. For every valid point in the most recent ray, the algorithm performs a search among the three nearest neighbors in the previous ray to find a valid velocity. The search pattern is shown at the bottom of Figure 60 (page 302). This larger selection of alternate-PRF bins makes it more likely that the algorithm finds the pairs of Low/High PRF data that are required for unfolding.
2. The unfolded velocities are subjected to standard 3x3 filtering.

When performing Dual-PRF scans, it is possible to collect data to the maximum unambiguous range of the low-PRF radials. Normally, this would result in the high-PRF radials showing no data as their maximum unambiguous range are shorter. However, the 3x3 filter is used to interpolate across this gap and fill in the missing data of the higher-PRF radials. The process is the same as described in Table 71 (page 301), but because there are two input bins less for the evaluation, the number of valid bins required to interpolate is also lowered by two in this case.

10.1.3.2.2 Dual PRF, Random Phase Processing

In random phase processing, the "seam" at the start of the second trip is always problematic, since the transmitter main bang and nearby clutter virtually always wipe out the first few second trip range bins.

At a constant PRF, the second trip seam is always at the same range, but in dual PRF random phase mode, the seam is different each ray.

Thresholded bins at the seam of the high PRF can be surrounded on either side by valid bins taken at the low PRF.

The 3x3 filter has the effect of interpolating the reflectivity and width data over the bins at the 2nd trip seam. Velocity data is also be filled-in using the nearest neighbor. The 2D filter mitigates much of the damage that is caused at the 2nd trip seam to make a nearly seamless display.

10.2 Reflectivity Calibration

The calculation of reflectivity described in Reflectivity (page 187). Here we look at its derivation.

You can use the **Zauto** utility to perform the calibration. See *IRIS and RDA Utilities Guide (M212925EN)*.

10.2.1 Plot method for calibration of Io

This approach generates the curve (red) that determines the value of I_o.

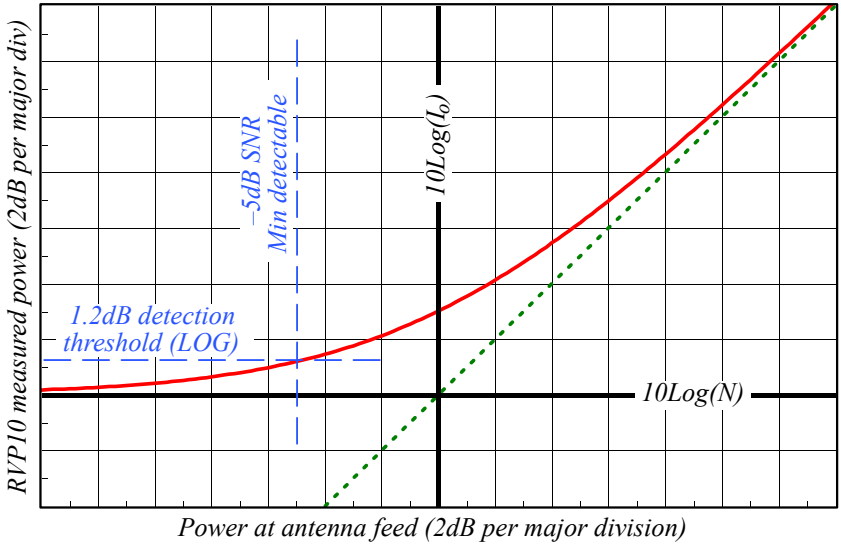


Figure 61 Model intensity curve - power at ntenna feed (2dB per major division)

The procedure is to connect a calibrated signal generator to the radar receiver and inject known power levels to generate a calibration plot of measured power versus the inserted power at the antenna feed, similar to that in shown in the figure.

The calibration reflectivity $\mathbf{dBZ_o}$ is computed from the radar constant and the value of $\mathbf{I_o}$, which is the intercept of the straight line fit (green) with the noise level.

To understand why this geometric construction yields the value of $\mathbf{I_o}$, let $\mathbf{G_{dB}}$ represent the overall gain of the RF and IF components leading up to RVP10. The green line can be interpreted as the response of an ideal noise-free amplifier having gain GdB, while the red curve is the response of the real-world amplifier(s) whose equivalent front-end noise is $\mathbf{I_o}$:

$$\text{(Red)} \quad 10\log_{10}(P_{OUT}) = \text{GdB} + 10\log_{10}(P_{IN} + I_0)$$
$$\text{(Green)} \quad 10\log_{10}(P_{OUT}) = \text{GdB} + 10\log_{10}(P_{IN})$$

The measured receiver noise is the horizontal asymptote of the red curve, that is, the value of the red curve when the input power $\mathbf{P_{IN}}$ is 0:

$$10\log_{10}(N) = \text{GdB} + 10\log_{10}(I_0)$$

Intersecting this measured noise level with the green straight line gives:

$$GdB + 10\log_{10}(I_0) = GdB + 10\log_{10}(P_{IN})$$

From which we see that the input power at the point of intersection is, indeed, I_o.

I_o is the received signal level that produces 0 dB SNR, that is, signal power equal to noise power. Do not confuse this with the minimum detectable power P_{MDS} which typically is several dB lower, depending on processor settings. In the above example, a 1.2 dB **LOG** detection threshold is shown (horizontal blue line) for the received signal. If RVP10 applies sufficient range and time averaging so that thermal noise alone produces very few false alarms above 1.2 dB, then P_{MDS} are a 5dB lower than I_o. We would expect a detection rate of roughly 50% for echoes arriving at this "minimum detectable" level.

Typically a CW test signal is used to generate the test curve shown in the figure. Follow the instructions provided by the radar manufacturer for injecting a test signal. During calibration, the radar should be fully operational, so that all sources of noise are present. Ideally the transmitter should be turned on during calibration.

NOTICE! Verify with the radar manufacturer that no damage can occur to the signal generator if the transmitter is running during the calibration.

1. Raise the antenna up a few degrees to avoid ground thermal noise.
2. Insert signals at steps of 5 or 10 dB over the entire range of the system.
3. Draw the plot shown in the previous figure.

   You can use fine resolution steps at the ends of the scale to observer the details of the roll off.

   If you are using the IRIS software, you can do this in the **Zauto** utility.

4. Tune the frequency of the signal generator using the setup command **pr**, and displaying the received signal spectrum.

   Check the tuning at the end of the calibration to make sure the signal generator and IFDR have not drifted apart.
   Each time that a new signal level is injected, the measured power values are obtained by first invoking the **SNOISE** command and then reading- back the results using the **GPARM** command. Use the `Log of Measured Noise Level (Word 6)` from **GPARM**.
   This procedure averages many samples together.
5. Turn it all the way down and make one more sample to measure the noise level **N**.

   You can obtain I_o from the intercept of the horizontal line at **N** and the straight line fit to the linear portion of the curve.

6. Correct the value for losses.

   See Treatment of Losses in Calibration (page 307).

10.2.2 Single-Point Direct Method for Calibration of I_o

- Signal generator output calibrated in absolute dBm
- Power meter for checking the signal generator calibration

This calibration method uses the TTY setup commands.

1. Use the power meter to check the calibration of the signal generator.
2. Turn off the radiation and connect the signal generator to the test signal injection point.
3. Raise the antenna to at least 20°, and set the azimuth to point away from any known RF sources including the sun.
4. Select the pulse width using the **Mt** command.

   See **Mt**— Triggers and timing (page 92).
5. Select the **pr** command and use the commands to set the following:

   ```
   Plotting Received Power Spectrum...
   Rx:Pri, Zoom:x1-x8, Navg:25, Start:100.01 usec (14.99 km), Span:50 usec
   ```

6. Set the signal generator to the approximate radar RF frequency with a power level corresponding to a strong signal (30 dB above the noise), and use a CW signal (not a pulse).

   This signal should be visible as a peak in the spectrum display.
   Adjust the signal generator's RF signal frequency so that it produces the precise IF frequency (for example, IF frequency of 30 MHz).
7. Turn the signal generator off and record the *Filtered* power level.

   Because of large averaging, it takes several seconds for the average to stabilize.
8. Turn the signal generator on, verify that the peak is still at the IF frequency and adjust the power level to obtain precisely 3 dB more *Filtered* power than was observed with the noise only.

   Allow several seconds for the averaging to stabilize after you make each amplitude adjustment.
   This is the value of I_o, that is, the test signal power equals the noise power.
9. Correct the value for losses.

   See Treatment of Losses in Calibration (page 307).

10.2.3 Treatment of Losses in Calibration

When calibrating the dBm level of the test signal, you must account for any losses that may occur between the antenna feed and the injection point, and in the cable and coupler that connect the signal generator to the injection point.

The following figure shows the nomenclature of the losses that are involved in the calibration.

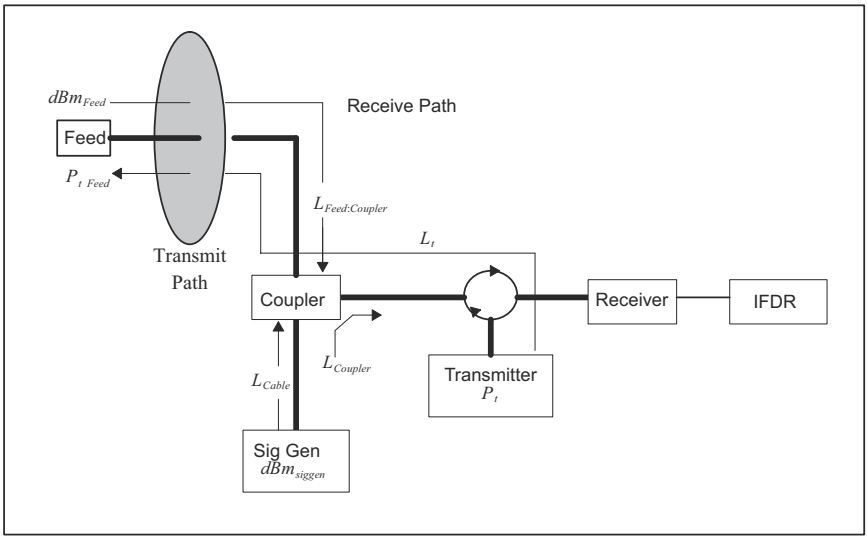


Figure 62 Overview of Losses that Affect LOG Calibration

The relationship between the injected test signal and the value of the received power relative to the feed is:

$$dBm_{Feed} = dBm_{Injected} + dBL_{Feed:Coupler}$$

$$dBm_{Feed} = dBm_{Siggen} - dBL_{Coupler} - dBL_{Cable} + dBL_{Feed:Coupler}$$

For example, assume the following:

Loss	Relationship	Value
Loss between the feed and the coupler	$dBL_{Feed:Coupler}$	3 dB
Loss caused by the coupler	$dBL_{Coupler}$	30 dB
Loss in the cable from siggen to coupler	dBL_{Cable}	2 dB

If the test signal generator output is -50 dBm, the injected power is

$dBm_{Injected} = -50 - [30 + 2] = -82\ dBm$

The equivalent power at the feed is then 3 dB more than this:

$dBm_{Feed} = -82 + 3 = -79\ dBm$

During the calibration, there are several ways to handle the losses using these equations. For example:

- Correct each signal generator value for losses so that the calibration plot shows IFDR measured power against the received power at the feed.
  This is recommended for manual calibration.
- Plot the signal generator values directly and correct the intercept power $\mathtt{I_0}$ for losses so that it is properly referenced to power at the feed.
  This is the approach used by the IRIS **Zauto** utility. See *IRIS and RDA Utilities Guide (M212925EN)*.

10.2.4 Determining dBZo

The calibration reflectivity is determined from the radar equation as follows:

$$dbZ_0 = 10\log\left[Cr_0^2 I_0\right]$$

where $\mathtt{I_o}$ is in **mW** (corrected for receive losses), the reference range $\mathtt{r_o}$ is 1 km, and the radar constant **C** is:

$$C = \frac{2.69 \times 10^{16} \lambda^2}{P_t \tau \theta \phi G^2} L_t$$

where:

λ
: Radar wavelength in cm.

P_t
: Transmitted peak power in kW.

L_t
: Transmit loss (for example, 3 dB corresponds to $L_t = 2$)

τ
: Pulse width in microseconds.

θ
: Horizontal half-power full beamwidth.

φ
: Vertical half-power full beamwidth.

G
: Antenna gain (dimensionless) on beam axis.

The radar constant is determined from the characteristics of your radar (check with the manufacturer if you are unsure of the values). Note that transmit losses are accounted for in the radar constant, while receiver loss is usually included in the calculation of $\mathbf{I_0}$.

If the value of $\mathbf{I_0}$ calculated above was not based on loss-corrected dBm values, correct $\mathbf{I_0}$ as follows:

$$dBI_{0\ corrected} = dBI_0 - dBL_{Coupler} - dBL_{Cable} + dBL_{Feed:Coupler}$$

Example Calculation of dBZo

Use this sample calculation to check your arithmetic. The radar parameters:

Table 72 Radar Parameters

Parameter	Description	Value
λ	Radar wavelength in cm.	5 cm
P_t	Transmitted power in kW.	500 kW
L_t	Transmit loss.	2 (3 dB)
τ	Pulse width in microseconds.	1 microsecond
θ	Horizontal half-power beamwidth in degrees.	1°
φ	Vertical half-power beamwidth in degrees.	1°
G	Antenna gain (dimensionless) on beam axis.	19,953 (43.0 dB)

The radar constant for this example is,

$$C = \frac{2.69 \times 10^{16}\lambda^2}{P_t\tau\theta\phi G^2}L_t = \frac{\left(2.69 \times 10^{16}\right)(5)^2}{\left(500\right)\left(1\right)\left(1\right)(19,953)^2}(2.0) = 6.76 \times 10^6\left[mm^6m^{-3}km^{-2}mW^{-1}\right]$$

Assume that $\mathbf{I_0}$ with loss correction is calculated to be -105 dBm (3.16×10^{-11} mW), then dBZ_o is:

$$dBZ_0 = 10\log\left[Cr_0^2I_0\right] = 10\log\left[\left(6.76 \times 10^6\right)(1)^2\left(3.16 \times 10^{11}\right)\right] = -36.7dB\left(mm^6m^{-3}\right)$$

You can download this value to the signal processor using the **SOPRM** command.

10.3 Calibration considerations for dual-polarization

Dual-polarization systems require additional calibration compared to single-polarization systems. There are three aspeccts to the calibration:

- dBZ_0 measurement in both channels for dBZ and dBT calibration
- GDR measurement for Z_{dr} calibration
- XDR measurement for LDR calibration

10.3.1 dBZ_0 calibration for dBZ

RVP10 supports separate calibration of both polarization channels. Measurement of dBZ_0 for each channel of a dual-polarization system is identical to the conventional radar case.

For a single-channel switching system, the only difference between the horizontal and vertical signal paths occurs after the high power switch, that is, differential insertion loss of the switch itself and any differential insertion loss of the waveguides and feed after the switch. This means that for single-channel switching systems it may be sufficient to calibrate at one polarization and then adjust the calibration of the other channel by the differential gain GDR. See GDR calibration for Z_{dr} (page 310).

10.3.2 GDR calibration for Z_{dr}

The Z_{dr} offset is the dB value of the relative gain between the co-polarized channels including both transmitter and receiver gain, that is:

$$ZDROffset = 10LOG\frac{g_v^r g_v^t}{g_h^r g_h^t} \text{ and } gdr = \frac{g_v^r g_v^t}{g_h^r g_h^t}$$

GDR is input into the processor as a dB value. However, for these analyses, the linear **gdr** value is sometimes more convenient.

In principle, if dBZ_0 could be calibrated perfectly in both channels, measurement of GDR would not be required. In practice, this is not possible because dBZ_0 cannot be calibrated to an absolute accuracy sufficient for Z_{dr}, that is, to 1/16th of a dB. Therefore, RVP10 uses the GDR approach.

Since GDR includes both transmitter and receiver differential gains, accurate calibration requires that an actual target be observed.

The following steps show one way to do this is.

1. Set the GDR to be **0** dB using your application software (for example, for Vaisala IRIS systems in the **Setup** utility RVP section).
2. Disable clutter filtering for Z_{dr} in either your application software (by selecting filter 0) or explicitly in the RVP10 TTY setups **mp** section.

3. Place the antenna at 90° elevation (vertical incidence) during moderate to heavy rain.

   The melting layer should be at a height that is well above the recovery zone of the T/R and in the antenna "far zone". A melting layer higher than 2 km (1.2 mi) is suggested, but you must consider the specific characteristics of the radar.

4. Collect Z_{dr} data at vertical incidence while the antenna is rotating in azimuth.
5. Use a separate application program to average the Z_{dr} values around a full 360° at each range bin (height).

   Generate a plot of 360-average Z_{dr} against height.

6. Check that the average Z_{dr} values in regions of strong signal (>20 dB SNR) below the bright band are approximately constant with height.

   Use this value in your application software for GDR.

7. Enter the value and repeat the calibration to verify that the average Z_{dr} is now 0 dB.

The rationale for this approach is that, when viewed at vertical incidence, rain should have a Z_{dr} of 0 dB since the drops all appear circular. The reason for averaging over 360° is to cancel out effects from sidelobe contamination from nearby ground targets and other artifacts of the antenna/feed/radome system. For example, the radome may have an obstruction light on the top. Some of these artifacts can be minimized by ensuring the weather targets are strong, that is, heavy rain is preferred for this calibration.

10.3.3 Offset calibration for LDR

XDR is the dB value of the relative gain between the co- and cross-receiver channels for LDR measurements. Analogous to GDR, it is defined as the dB value of the ratio of the vertical to horizontal receiver gains, for example:

$$XDR = 10LOG\frac{g_v^r}{g_h^r} \text{ and } xdr = \frac{g_v^r}{g_h^r}$$

The following techniques are available for calibrating XDR:

- Using the Sun to Measure LDR (page 311)
- Using a Signal Generator to Measure LDR (page 312)
- Using a linear feed horn to measure LDR (page 312)

For all these calibration methods, Vaisala recommends that you

- Turn the transmitter off.
- Set XDR to 0 dB in the application user software
- Configure the RVP10 TTY setups as follows:
  - Noise correction enabled for LDR and noise sample taken prior to the measurements (with care not to sample with a test signal turned-on or while looking at the sun).
  - Clutter correction disabled for LDR.

10.3.3.1 Using the Sun to Measure LDR

Use the sun to measure LDR. The measured value of LDR is the XDR offset.

1. Measure the LDR in fixed mode for both LDRH and LDRV.
2. Check that the values are reciprocal (for example, +1 dB and -1 dB).

   If the values are not precisely reciprocal, use the average of the absolute value (for example, for +1.4 and -1.2, use 1.3).
3. Enter the XDR value.
4. Retest to verify that the sun has been properly corrected to have zero LDR.

10.3.3.2 Using a Signal Generator to Measure LDR

You can measure LDR using a signal generator with a waveguide connection.

1. Connect a signal generator with a splitter to both channels and measure XDR directly.

   This does not account for any effects that are before the coupler (for example, waveguide, feed, radome, antenna gain).

10.3.3.3 Using a linear feed horn to measure LDR

This approach requires a calibrated linear feed horn with an RF source located several hundred meters from the radar.

Signal multi-path effects may bias the results from this technique.

1. Maximize the H channel return and measure the response using the RVP10 **pr** command `Filtered` power in the `Primary Channel`.
2. Rotate the feed horn to vertical and maximize the power in the `Secondary Channel`.

   The difference in dB is XDR.

10.4 Thresholding polarization parameters

The thresholding of polarization parameters by the processor eliminates bins with weak or uncertain signals. Note that the thresholding can be disabled if you want to see all of the data regardless of the data quality.

All the polarization parameters are based on power ratios. RVP10 requires that each power term in a ratio pass a signal-to-noise test similar to the log power test. For example, there are up to four different powers that can be calculated (alternating dual-channel case) so the tests for each of these are:

$$\frac{< |S_{hh}|^2 >}{N_h} > N_{thresh}$$

$$\frac{< |S_{hv}{}^2| >}{N_h} > N_{thresh}$$

$$\frac{< |S_{vv}|^2 >}{N_h} > N_{thresh}$$

$$\frac{< |S_{vh}|^2 >}{N_h} > N_{thresh}$$

where the linearized threshold that is input as the dB LOG threshold, that is:

$$N_{thresh} = 10^{\frac{LOGthresh}{10}}$$

For example, a valid LDRH requires both a valid S_{hh} and a valid S_{vh}. The parameters RhoH and PhiH have the same requirement since they are the magnitude and phase of the cross-correlation function which is based on S_{hh} and S_{vh}.

There are 2 exceptions:

Z_{dr}
: Z_{dr} requires that S_{hh} and S_{vv} pass the signal-to-noise tests noted above. However, Z_{dr} can be additionally thresholded by any of the other threshold parameters (`LOG`, `SIG`, `SQI`, `CSR`) similar to a standard moment.

PhiDP for single channel alternating case
: PhiDP requires that S_{hh} and S_{vv} pass the signal-to-noise tests noted above. In the single channel alternating case, PhiDP must also satisfy the additional test that the Doppler velocity at the range bin must be valid, that is, not thresholded by its own criteria. This is because the algorithm for PhiDP in this case subtracts the phase change due to the Doppler velocity. If the Doppler velocity is uncertain, the algorithm cannot produce reliable results.

11. Time series recording

11.1 Time series overview

Time series (TS) recording enables recording and playing back **I**/**Q** data.

Operators can record time series and then use their own off-line programs to process the data.

The time series can be recorded directly on an RVP or on a separate archive host through a gigabit network.

Table 73 Time series software components

Software component	Description
RVP10 TS API	Where time series reside in memory in RVP10.
`Tsexport`	Grabs time series from the TS API and sends them over the network through a UDP broadcast. Typically a gigabit network is used for this.
`Tsimport`	Receives UDP TS packets and recreates the TS API on a local machine.
`Tsarchive`	Records time series to a local disk. This can be on RVP or a separate networked archive host. Supports both archive and playback. (licensed separately)
Ascope utility	Can be used in playback mode to view either raw time series or processed results from RVP10 processing algorithms.
`Tsview` utility	Used to view diagnostic printouts of stored time series files. The complete source code is provided to assist users with writing their own applications.

Only the `Tsarchive` is licensed separately. You can write your own TS record/playback software and still take advantage of the network features such as `Tsimport` and `Tsexport`.

11.2 TS record and playback software architecture

The TS record and playback features on a local RVP10 and a remote archive host share a common software architecture. The following figure shows the most general case.

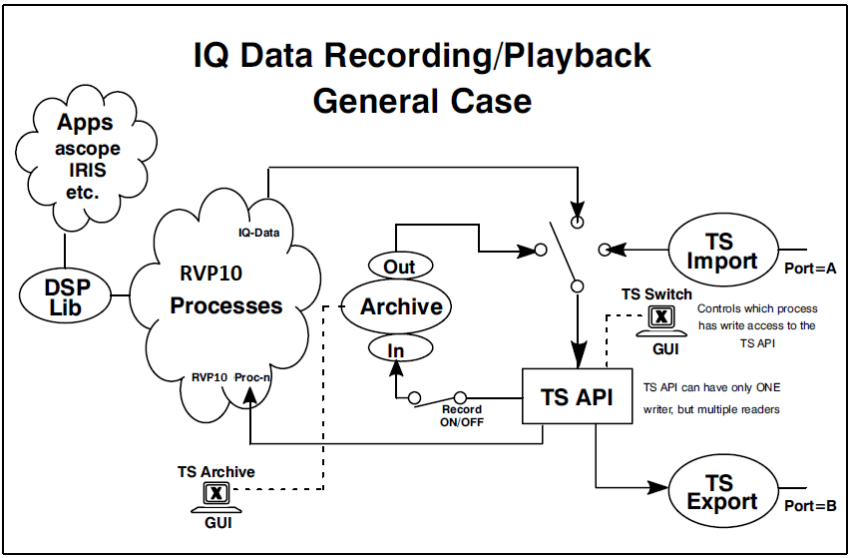


Figure 63 IQ data recording/playback general case

The structure shown runs on a single machine. Replicas of this structure, with pieces included or excluded, can be run simultaneously on several machines to handle different scenarios, such as a separate archive host. This modular design does not make a strong distinction between RVP10 and a separate archive host operating in record or playback mode.

Table 74 Description of `IQ` data recording/playback general case

Process	Description
`TS API`	Where the time series data reside in memory. The time series can be written to the TS API from one of the following sources: • From a local RVP10 • From a local disk archive • From a remote network host (with `tsimport`)
`tsswitch`	GUI that allows the user to select the sole TS writer from among these three possible sources. See TS Switch Utility (page 320).

Process	Description
`tsarchive`	Handles both the recording and playback of data. It has its own GUI to select record/playback mode and which directory contains the local disk archive. `Tsarchive` also shows an inventory of the TS files and has a filter/search for locating specific groups of files (for example, by date and time). See TS Archive Utility (page 322).
`tsimport` `tsexport`	Provide the ability to receive/send time series over a network. Either a 1000 BaseT (gigabit) or 100 BaseT Ethernet can be used depending on the typical mode of operation and the competing network traffic. For example: `1000 bins * 2 parameters/bin/pulse * 16 bits/parameter * 1000 pulses/sec = 32 Mbit/sec` In the example, `2 parameters/bin/pulse` are the `I` and `Q` values, which are represented by 16 bits each (floating point). For dual polarization systems with two receiver channels, the data rate would be doubled. The 32 Mbit/sec basic data rate here would fit comfortably on a dedicated 100 BaseT network, with little or no competing traffic. The output is through UDP broadcast. This is a very efficient way to transfer data since there is virtually no overhead as compared to standard TCPIP. The socket ports are configured so that the **tsexport** on one system connects to the `tsimport` on another system.
RVP10 Processes	Collection of processes that are only present on an RVP10 machine. The important functions are: • `IQ-Data`: writes real time TS to the TS API. These are collected from an IFDR. • `RVP10Proc-n:` extracts TS from the `TS API` and processes the data to obtain the moments. To view these processes, use the **v** command in **dspx**. A remote archive host is most likely not from RVP10. In this case, these processes do not exist.

11.3 Using RVP TimeSeries API

The `TimeSeries` API is the interface through which (`I`,`Q`) data are made available to the application code that requires them.

This API is central to the design and operation of RVP and is used by the parallel computer processes to access incoming time series data.

The `TimeSeries` API is provided within a larger collection of RDA support services in *`rda/rdasubs`*. For API definitions, see the header file documentation.

11.3.1 Reader and Writer Clients

The `TimeSeries` API is stateless and passive from a reader client's point of view. It allows any number of callers to eavesdrop on the (`I`,`Q`) data as they arrive, but there are no control actions passed back in the other direction.

The reason that an event driven model is not provided is that the API is fundamentally a single-writer/multiple-reader interface. There is no private state maintained for each reader client that hooks up to it. As such, the notion of 'notify me when new data are available' would not be well-defined, because 'new' would have to be a per-client notion, that is, new since the last data that each particular client checked.

The API is most valuable in providing random access to the recent buffered (`I`,`Q`) data and can buffer a couple of seconds of data within the `TimeSeries` API. This means that programs using the API only need to check the data every 50 to 100 msec without the risk of losing any data. This has only a minimal load on the CPU.

11.3.2 Attach and Detach Details

Use `rvptsAttach()` to attach and `rvptsRelease()` to release the connection.

Always attach to unit `RVPTS_UNIT_MAIN`. The others are for internal RVP use. You must also specify your client type. Readers are generic, but for writers, there are several choices because we need to switch sources, and we need to know who the sources are.

For commands, see *`program /rda/tsapi_lib/`*. The same directory includes the *`rvpts_example.c`* example.

Other programs using the `TimeSeries` API are:

- `ts/export/tsexport.C`
- `ts/export/tsimport.C`
- `ts/switch/tsswitchMainWin.C`
- `ts/exec/TsArchExec.C`
- `ts/exec/tsclientshell.C`
- `ts/archive/tsarchAS.C`

11.3.3 Extracting Pulses with Sequence Numbers

Because there is a ring buffer filled with time series, a reader should first get the most recent pulse, and start reading after that. We do that with the function `rvptsCurrentSeqNum()`.

Check the source code to `tsexport.C` to see how this is done. To get data starting at a specific sequence number, use `rvptsGetPulses()`. If there are none available, then we should sleep to wait for some more.

If RVP is reconfigured, this causes a change in the acquisition mode. An example might be that the PRF, pulse width, or transmit polarization has changed. All data read from a single call to `rvptsGetPulses()` is for the same acquisition mode.

11.3.4 Using Memory Bandwidth Effectively

Reading each pulse of data individually is inefficient. Instead, a smart reader sleeps until, for example, 10 pulses have arrived before reading them.

To support this, function calls find out how many pulses are available after a give sequence number, and how old a given pulse is.

For more information, see `tsexport.C`.

11.4 Installing and configuring TS recording

You can use TS recording in two configurations. The simplest (but least flexible) configuration is where RVP10 also serves as the archive host.

When using RVP10 as the archive host, create a separate partition for the purpose to prevent system crashes.

The ideal configuration is with a separate archive host with an increased amount of disk space and peripherals (that is, tape drives). The two system configuration should be used in a high bandwidth environment. If this is not possible, RVP10 can be equipped with a separate disk for archive operations.

In two system configurations, the `tsimport` and `tsexport` modules must be turned on at boot time on both systems. The UDP ports for the sending and receiving system must also be configured.

11.4.1 Required software for TS recording

The TS archive software is part of the Vaisala RDA software, which is installed by default on all RVP10s and RCP8s, but it is not installed on IRIS systems.

Table 75 Required software for TS recording

System Setup	Required Software
RVP10/RCP8	No additional software required
Archive system (IRIS)	Install RDA You must select the **Keep old files** option during installation.
Archive system (without IRIS)	Install RDA

11.4.2 Configuring UDP ports for TS recording

You can configure the UDP ports by editing the **tsimport** and **tsexport** scripts in */etc/vaisala/irisrda/templates/rc.d/*.

Copy the files to */etc/rc.d/init.d/*.

If you plan to use **tsarchive** with a separate archive host, you must edit the **tsimport** and **tsexport** files. The export port on the sending system must match the import port on the receiving system so that a connection can be made between the two ports.

Table 76 Recommended UDP ports for TS Recording

RVP10	Archive Host
TSEXPORT: 30780	TSEXPORT: 30781
TSIMPORT: 30781	TSIMPORT: 30780

Each script contains extensive comments. Edit the scripts to suit your configuration.

The scripts are shipped configured for the RVP10 end, so you must edit the archive host's files.

You must also edit all **tsexport** files to explicitly set the target IP address to which it broadcasts the time series. Vaisala recommends that you use a single target host. A broadcast address can be used, but make sure that all recipients can handle the traffic and that there are no **10baseT** network sections.

11.4.3 Configuring automatic start-up of tsimport and tsexport

1. Login as **admin**.
2. Use the **chkconfig** command to add the processes:

```
sudo systemctl enable tsimport.service
sudo systemctl enable tsexport.service
```

3. You can now start and stop these programs with the commands:

```
sudo systemctl start tsimport
sudo systemctl stop tsimport
```

11.4.4 Configuring Network Buffering for tsimport

For **tsimport** to successfully read the pulses of data from the network, you must enlarge the network read buffers.

1. Add the following commands to the end of the */etc/sysctl.conf* file:

```
net.core.rmem_default = 1000000
net.core.rmem_max = 4000000
```

2. Reboot the system.

11.4.5 Running tsimport and tsexport from the Command Line

You can run **tsimport** and **tsexport** from the command line using the following command line options:

```
$ tsimport  -help
tsimport command line options:
  - daemon  - Run as daemon
  - debug  - Print diagnostics
  - help  - Print this list
  - port: <port> - Specify the port number to use
  All other options ignored
```

```
$ tsexport  -help
tsexport command line options:
  - broadcast: <broadcast-address>
  - daemon  - Run as daemon
  - debug  - Print diagnostics
  - port: <port> - Specify the port number to use
  All other options ignored
```

11.5 TS Switch Utility

Only one process may write to the **TS API**. Use the **TS Switch** utility to select the source.

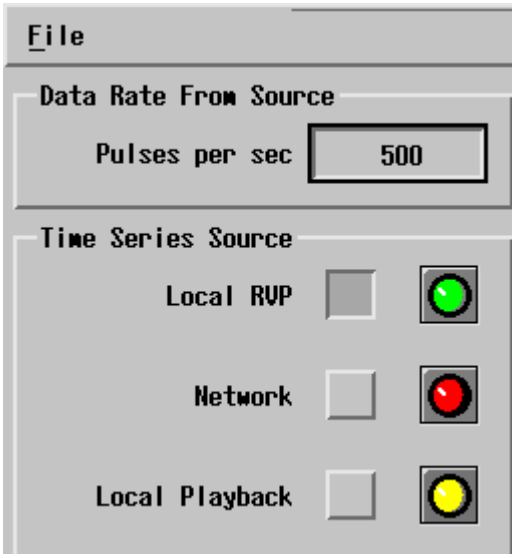


Figure 64 TS Switch Utility

Local RVP
Real time `IQ` from the IFDR. This setting is available only on an RVP.
Used for normal data processing, to archive local RVP10 real time `IQ`, or to export real time `IQ` over the network.

Network
Used to collect time series from a networked RVP10 or from an archive host using the TS import process.
Can be used by an RVP10 for playback or by an archive host for recording.

Local Playback
Used to extract time series from the local disk archive.
On an RVP, the time series can be processed. On a separate archive host, the time series can be sent to an RVP or a custom user application for processing.

Table 77 TS Switch status indicators

Color	Description
Green	Source successfully selected.
Yellow	Source is available, but not currently selected.
Red	Source is unavailable. If you select the source, the status indicator remains red until it is enabled. In the case of a separate archive host, the **Local RVP** choice always shows red, because there is not a local RVP.

1. As an operator, start the **tsswitch** utility by typing:

   $tsswitch

2. Select the process.

11.6 TS Archive Utility

Use the **TS Archive** utility to record and playback archived data.

The utility provides access to the **TS Switch** utility, playback and record modes, and a complete archive file inventory for managing disk space.

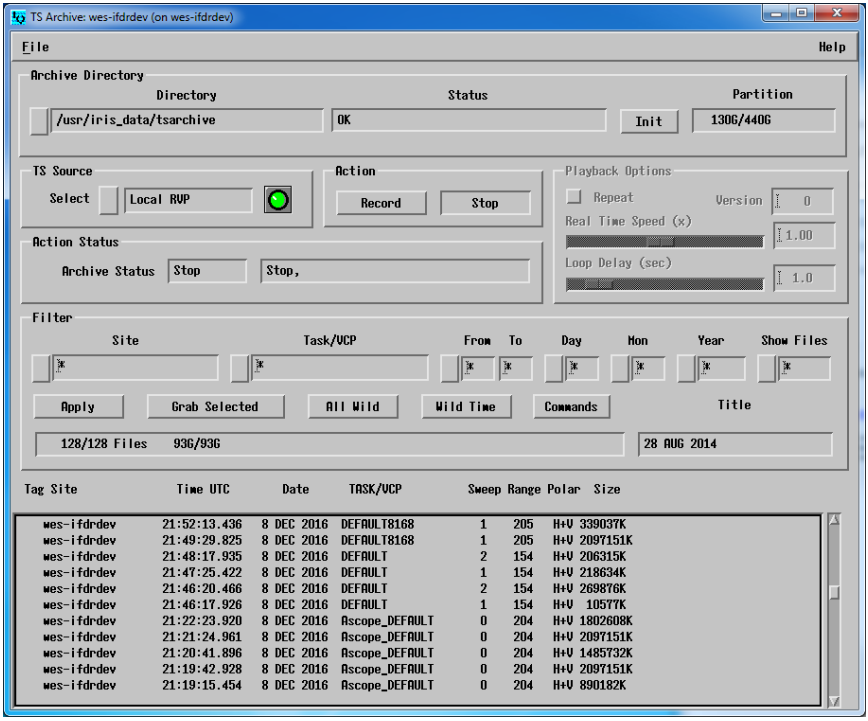


Figure 65 TS Archive Utility

1. Log in as operator.
2. Start the **TS Archive** utility by typing: **$tsarchive**

3. Under **Archive Directory**:
   - Use the **Directory** section to select which archive directory to use, or to add another directory.

     Vaisala recommends you first create the directory from the command prompt.

     ```
     $mkdir /bigdisk/tsarchive
     ```

   - Check **Status** to see information about the chosen archive directory.
   - Select **Init** to initialize the directory. You are prompted to type a title for the directory and to verify the contents of the directory.
     The directory must be empty or contain ONLY `tsarchive` files.
     If a directory contains `tsarchive` files, you are prompted for permission to delete them.

     For system security reasons, **Init** can only delete TS data files.
     If you must delete other, non-`tsarchive` files in the directory, you must use the UNIX **`rm`** command to manually delete them.

   - Check **Partition** to view the ratio of used disk space over total disk space for the selected archive directory partition.
     For example, if the selected archive directory is */bigdisk/tsarchive*, the **Partition** field shows the used/total disk space for */bigdisk/*.
     The values displayed include all file types in the partition.
4. Select **TS Source** to launch the **TS Switch** utility and chose the source used to record data.

   See TS Switch Utility (page 320).
   You can now record or play back data.

   a. Select **Record** to start data recording. The inventory at the bottom of the screen updates with TS file information.

   b. If the **Local Archive** source is selected, you can select **Playback** to play previously recorded data.

      - Select **Repeat Time Series** to repeat the selected files.
      - Select **Real Time Speed** to control the speed at which the files are played back.
      - Select **Loop Delay** to control the length of the pause in between each repetition.

   c. Select **Action Status** to display the current mode of the utility and provide information on what data are being recorded or played back.

   d. Select **Stop** to terminate recording or playback.

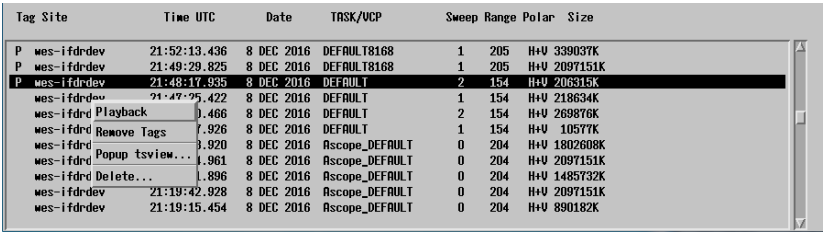
5. Use **Filter** area to manage files that are stored on a disk, including displaying data for a certain site, task, and time.

Option	Description
Site	Enter a site ID to select data from only one site, or enter the wildcard character to select data from all sites.
Task	A **Task** refers to either an IRIS task name (also called a volume control procedure (VCP)) or an **Ascope** saved configuration file name. Enter the name of a task in this field to narrow the list down to only those products generated by a specific task. You can include wildcard characters in the name. A question mark (?) matches any single character, an asterisk (*) matches any sequence of zero or more characters. For example, enter a task name of **`*PPI*_?`** to select all hybrid tasks with **`PPI`** somewhere in their names.
From, To	Enter or select a range of hours in these two fields to narrow the list of products by time of day.
Day, Month, Year	Enter or select a day, month. or year to narrow the list of products by date.
Show Files	Control how many files to include in the list.
Apply	Update the file list based on your selection criteria.
Grab	Select a product and then select **Grab** to insert a selected file information in the filter fields.
All Wild	Return all the fields to the wildcard character.
Wild Time	Change the hours, month, day, and year fields to the wildcard character.
Commands	Shows the following operations that you can perform on the **`tsarchive`** files selected from **Filter**: • **Playback**: Tags files for playback. A **P** appears in the left column. • **Delete**: Deletes files. A **D** appears in the left column. • **Remove Tags**: Untags any **Delete** or **Playback** tags. **NOTICE!** The commands apply to all of the **`tsarchive`** files that match the filter. If you put in wildcards everywhere and give the **Delete** command, every TS archive file could be deleted.

6. View TS Archive logs in the **Log** area.

Archive Log Column	Description
Site	TS site used to create the data.
Time and **Date**	UTC time and date when the data were acquired.
Task/VCP	Name of the associated tasks or **Ascope** file used for controlling RVP.
Sweep	Number of sweeps (for example, 360°) in the Task/VCP.
Range and **Polar**	Range and polarization of the TS data.
Size	Size of the file.

7. Right-click a file or group of files in the **Log** area to show the following commands:



- **Playback**: Marks the files with a **P** in the **Tag** column. When selected, only the tagged files are played.
- **Delete**: Marks the files with a **D** in the **Tag** column, and deletes the files. This is helpful when you select a large list of files for deletion.
- **Remove Tags**: Clears the **D** or **P** tags.
- **Popup tsview**: Provides file information in another window.

11.7 Software application examples

RVP supports four uses of the time series application:

- TS recording on a local RVP10, see TS recording on a local RVP (page 328).
- TS recording on a separate archive host, see TS recording on separate archive host (page 326).
- TS playback on a local RVP10, see TS playback on a local RVP10 (page 330).
- TS playback from a separate archive host to an RVP10, see TS playback from a separate archive host to an RVP10 (page 329).

In the example descriptions, unused processes are omitted for clarity.

11.7.1 RVP10 in normal real-time operation

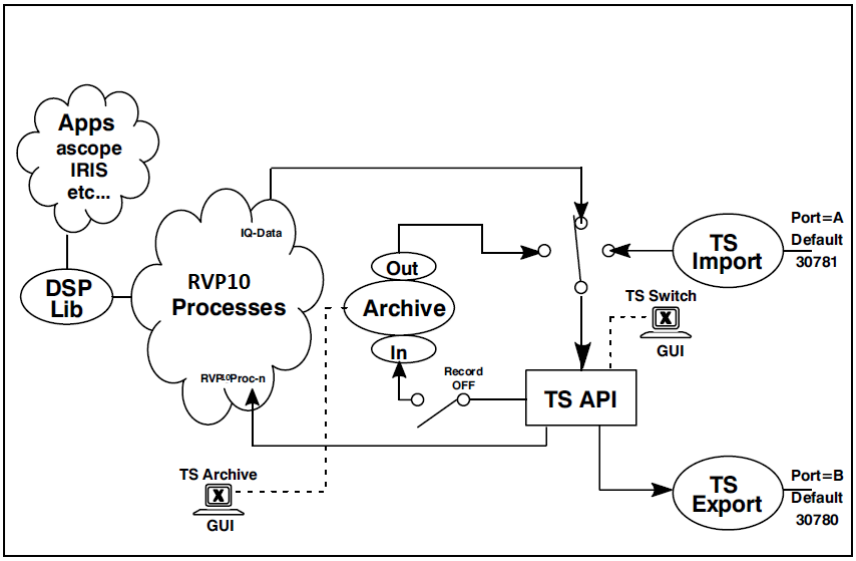


Figure 66 RVP10 in normal real-time operation

In this case, the **TS Switch** is set to write real-time data from the IQ-Data process to the `TS API`.

`RVP10Proc-n` extracts time series data and processes it. The configuration information is obtained from and data are passed to user applications through the `DSP Lib` functions. The `tsexport` process, if started, can extract data simultaneously from the `TS API`.

Utility settings

- **TS Switch**: **Local RVP10**
- **TS Archive**: N/A

11.7.2 TS recording on separate archive host

This is the recommended recording configuration for TS recording.

The advantage of having a separate archive host is that it is easy to install a large disk that is dedicated to time series recording without having record/playback/backup operations interfere with the normal operation of an RVP10. There can be multiple archive hosts on the network.

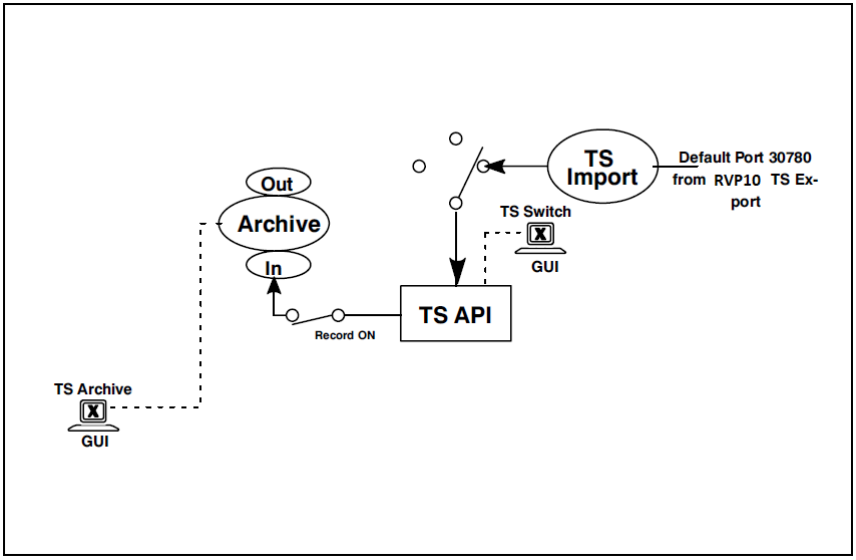


Figure 67 TS recording on separate archive host

In this case, the **TS Switch** is set to write `IQ` data from the `tsimport` network source. Typically this is a 100 or 1000 Base T LAN connection. The time series are placed on the network through UDP broadcast by the `tsexport` process on a networked RVP10.

Utility settings

RVP10:

- **TS Switch**: **Local RVP10**
- **TS Archive**: N/A

Archive host:

- **TS Switch**: **Network**
- **TS Archive**: **Record**

11.7.3 TS recording on a local RVP

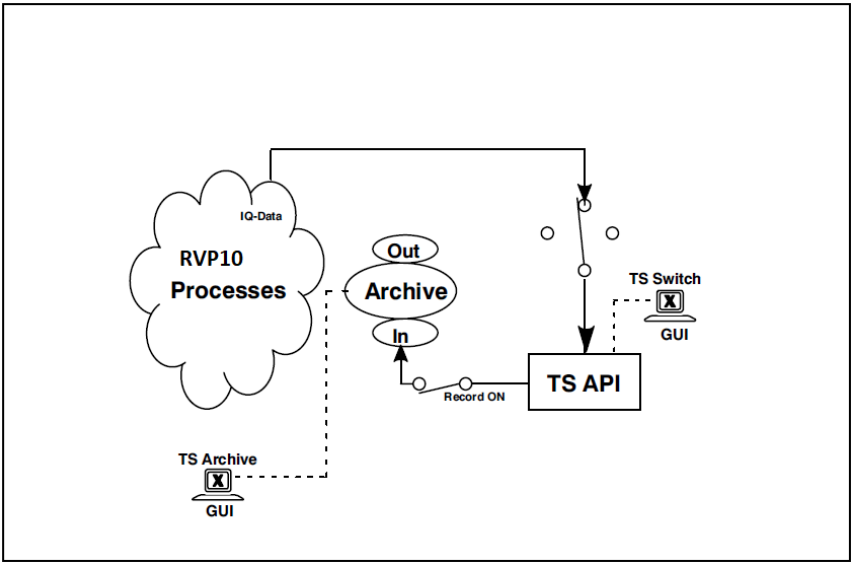


Figure 68 TS record on local RVP10

In this example, the **TS Switch** is set to place the real-time `IQ` values from the IQ-Data Process into the `TS API`. These are extracted and recorded to local disk by the `tsarchive` process.

While TS data are being recorded, RVP10 may still do its normal data processing tasks, as shown in RVP10 in normal real-time operation (page 326) .

Utility settings

- **TS Switch**: **Local RVP10**
- **TS Archive**: **Record**

This configuration records to a local disk on RVP10. Record to a dedicated data partition. DO NOT record to the "/" partition. This is because if the "/" partition fills up, the system crashes.

If the separate data partition fills up, then the system stops recording, but otherwise functions normally.

11.7.4 TS playback from a separate archive host to an RVP10

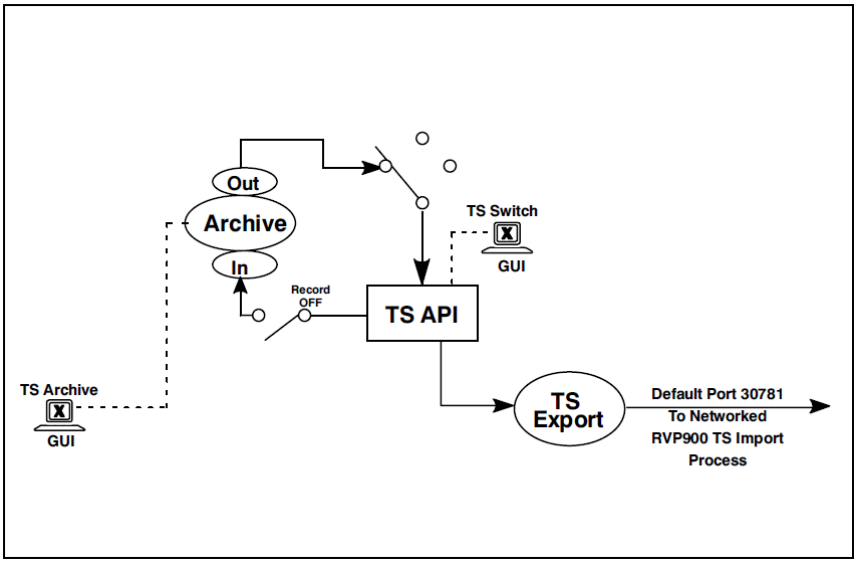


Figure 69 TS playback from a separate archive host

This is the recommended mode of operation.

In this example, the **TS Switch** is set to write data from the `tsarchive` to the `TS API`. `tsexport` sends these through a UDP broadcast over the network to an RVP10 for processing.

The diagram for the corresponding RVP10 would be identical to the one shown in this example, except the **TS Switch** would be set to write data from the `tsimport` process.

Utility settings

RVP10:

- **TS Switch**: **Network**
- **TS Archive**: N/A

Archive host:

- **TS Switch**: **Local Archive**
- **TS Archive**: **Play**

11.7.5 TS playback on a local RVP10

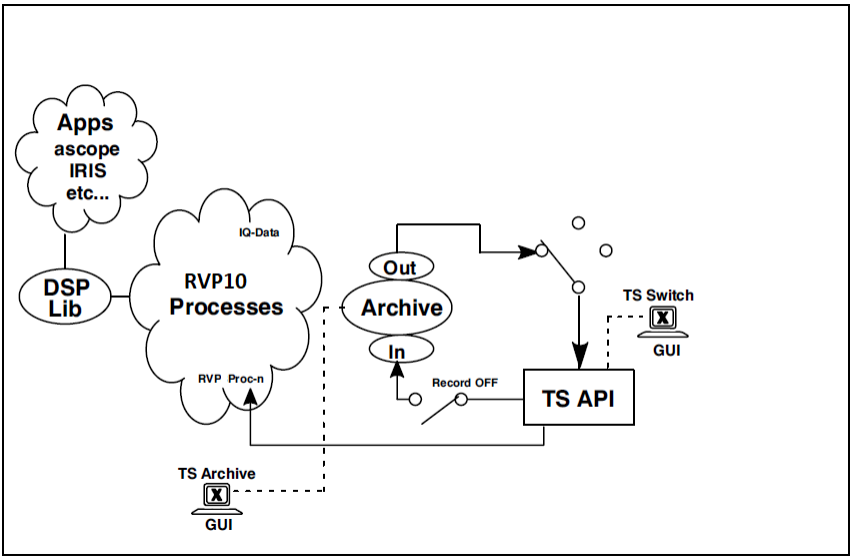


Figure 70 TS playback on local RVP10

In this example, the **TS Switch** is set to write data from the `tsarchive` to the `TS API`. `RVP10Proc-n` processes then reads the time series data from the API.

This is the same as the real-time normal operation case shown in RVP10 in normal real-time operation (page 326), except that the archive is the source rather than the real-time operation.

The difference is that applications that use the time series must know that the data are playback rather than real time data. This information as well as all of the required housekeeping data are part of the time series data format.

Utility settings

RVP10:

- **TS Switch**: **Local Archive**
- **TS Archive**: **Play**

11.7.6 TS Archive Recording Quick Guide

1. Select a directory for the TS data to be written to.

   Make sure the desired directory already exists in the file system.

2. Initialize the directory and give it a title. If the directory has already been used for archiving, then there is no reason to re-initialize it unless you want the data to be erased.
3. Select the data source and make sure the light is green.
4. Select **Record** and watch for data to arrive in the log section.

11.7.7 TS Archive Playback Quick Guide

1. Select the directory that contains the TS data to be played.
2. Under **TS Source**, select **Local Archive**.
3. Make sure the light is green.
4. Select the files that you wish to playback (right-click and select **Playback**) and mark them for playback.
5. Select any playback option.
6. Select **Playback**.

11.8 Ascope playback features

The **Ascope** utility is a stand-alone signal processor configuration and plotting utility. When RVP10 is in playback mode, you can use **Ascope** to configure the processing of the playback data and display the results. See *IRIS and RDA Utilities Guide (M212925EN)*.

The following figure shows the differences in the meanings of menu fields when RVP10 is in playback mode.

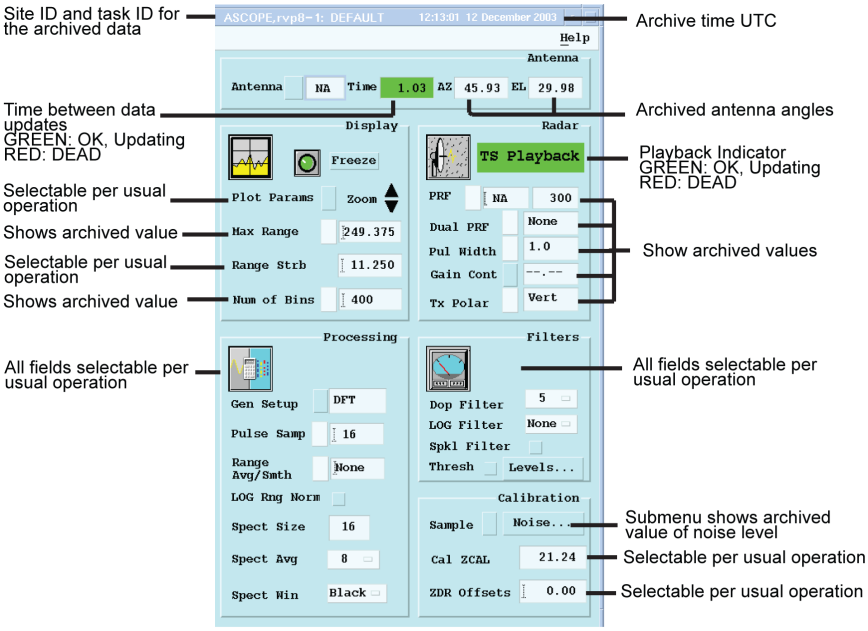


Figure 71 Ascope differences during RVP10 TS Playback

When RVP10 is in playback mode, instead of processing real-time data, the major difference is that some parameters were fixed at the time, when the data were recorded. Some examples of fixed parameters are:

- Maximum range
- PRF
- Transmit polarization
- Number of range bins
- Phase coding
- Value of noise level

These parameters relate to the transmit characteristics of the radar and for these, the archived value is displayed. You cannot alter the fields.

There are other parameter fields that can be changed, just like when you use **Ascope** with real-time data, such as the parameters to configure processing and plotting. For example, you can change processing major mode and clutter filtering parameters while RVP10 is in playback mode.

When using **Ascope** during RVP10 playback, the archive can be on the local RVP10 or on a separate archive host.

It does not matter where **Ascope** runs. It can be on the local RVP10 or on a networked host computer through `DspExport`.

11.8.1 Archive on separate archive host

The utility settings for archiving on a separate archive host are:

RVP10:

- **TS Switch**: **Network**
- **TS Archive**: N/A

Archive host:

- **TS Switch**: **Local Archive**
- **TS Archive**: **Play**

11.8.2 Archive on local RVP10

Utility settings

The support menus settings for archiving on a local RVP are:

- **TS Switch**: **Local Archive**
- **TS Archive**: **Play**

Network note

In most cases, you are not sitting up at the radar, so you need to export the displays for these utilities over the network by doing one of the following:

- Easy way: Use **`sigterm <hostname>`** to open a terminal window.
- Manually: **`rlogin`** and export the display with **`DISPLAY=<hostname>:0.0`**. You may also need to type **`xhost +`** on your local workstation.

It is often convenient to get the playback going in **`tsarchive`** with **Repeat** set, then start **Ascope**.

Ascope starts in the playback mode and updates the display appropriately. You are free to select the data processing and display parameters "on-the-fly" as playback is continuing and repeating.

When **Ascope** is set the way that you want it, you can use the **TS Archive** menu to select files or restart the playback.

11.9 TS playback using IRIS

You must configure some settings in the IRIS user interface.

Table 78 Configuration requirements for IRIS playback

IRIS location	Setting
Ingest > Signal Processing and Data Storage	**Source of recorded angles** = **RVP Tags** **Source of recorded time** = **RVP Tags**

IRIS location	Setting
General > Modes and Protocols	**Timezone for data recording** = **UTC**
General > Scanning Options	**Task Scheduling Control** = **Active/Passive** or **Passive** **Passive type** = **TS–Playback**

When the configuration is complete, you are ready to use IRIS to play back data.

1. Launch `tsarchive` on your RVP10 machine.
2. Select **TS Source** to launch the `tsswitch`.
3. In `tsswitch`, select **Local Playback**.
4. In the `tsarchive` **Playback Options** section, clear the **Repeat** button.
5. Set the **Version** to a unique non-zero value.
6. Select from the inventory the file or files you wish to playback, and set the **P** bit for those.

   If you are playing back a multiple sweep volume scan, there is one file per sweep.
7. Use the **IRIS** menu bar to bring up the **Task Scheduler** menu.

   Make sure it says **Passive** on the top menu bar.
8. Select the task that matches the data.
9. Select **Go/Schedule** for that task.

   It toggles to **running**.
   Wait for data to arrive.
10. On the `tsarchive`, select **Playback**.
11. On the **Task Scheduler** menu, set the task to **Stop (when done)**, so that it stops after one time.

> Because IRIS sorts stored data by time and playback version number, it is important to only playback the data once.

The **Ingest Summary** menu displays a 2-digit number containing the playback version after the site code.

Original data with a playback version of 0, does not include the number.

The ingest filenames have a **V** appended followed by the 2-digit playback number.

11.10 TS View Utility

Use the `tsview` utility to specify a TS file and obtain printout of header information and time series data on a terminal screen.

This is useful for checking how the data were collected, or to verify that **IQ** data were recorded.

Developers can use **tsview** software as a model when building their own off-line applications for reading and processing time series data.

If you have a licence for the **tsarchive**, right click a file TS inventory area to access the **tsview** utility.
See TS Archive Utility (page 322).

11.10.1 Starting tsview

You must have radarop privileges to run **tsview**. The **TS View** utility must be installed on the same node where the **IQ** data are archived.

1. In the archive directory, use the **ls** command to view the files.
2. Run: **tsview <pathname> <-options>**

   For example, **$ tsview - help**.
   To see the options, see Tsview command line options (page 336).

11.10.2 Starting tsview sample session

In this session, you view the pulse header information for the 101st pulse in a TS file. The archive directory is */bigdisk/tsarchive*:

```
//Change directory to location of TS files
$ cd /bigdisk/tsarchive

//List all files from 8 Dec 03 19:25:19
$ ls  *20031208.192519*

//One file found
RVP10.20031208.192519.074.Ascope_DEFAULT.0.H.249

//run tsview
$ tsview  RVP10.20031208.192519.074* -count:1 -skip:100 -v

//tsview output
Pulse Info size :1148
==============TS Pulse Info==============
Site :RVP10 Version:0
Major Mode :1  (FFT) Polarization:1 (Vertical)
Phase mod sequence :0  (Fixed)
ask  Name:Ascope_DEFAULT Sweep Number:0
. . .
```

For information on the header format and information content, see TS record data format (page 338).

For the file naming convention, see Tsview command line options (page 336).

11.10.3 Tsview command line options

-help

Gives a list of available options:

```
-count :N (only print N pulses)
-data (print data values)
-help (print this list and exit)
-length :N (max line length to use)
-skip :N (skip the first N pulses)
-verbose (print full header info)
-noExit (do not exit when done) pathname
(all other arguments ignored)
```

pathname

This is the directory and name of the TS file.

Before starting **tsview**, use the **cd** command to go to the directory where the archive files are located. This saves you from having to specify the full path and to allow the X-window center button and copy/paste technique to be used to avoid typing file names.

In the archive directory, use the **ls** command to view the files that you want by date and time using standard UNIX options for **ls**. The asterisk * is a wild card.

The file names are very long, so you do not want to type them. Instead, highlight the file name and click the middle button to copy the text to the terminal cursor location.

The file name format is designed to make it easy to identify TS files. In this example, we used a file with the name:

```
RVP10.20031208.192519.074.Ascope_DEFAULT.0.H.249
```

The file name format is:

```
site.YYMMDD.HHMMSS.SSS.taskname.sweep.polarization.maxrange
```

The fields are as follows:

site
: Site name typed in the setup program on RVP10 that generated the data. **site** fields are preprocessed to remove characters that would mess up the filename, such as unprintable characters, space and /.

`taskname`
: Identification of the application configuration that was operating when the TS data were collected. In the case of IRIS, it is the IRIS TASK name. In the case of the **Ascope** utility, it is set to `ascope_<filename>` where filename is the name of the saved ascope configuration that was used to collect the data (in the example, `Ascope_DEFAULT`). In the case of a custom user application, the user can specify an appropriate ID for the configuration so that it is archived appropriately. `taskname` fields are preprocessed to remove characters that would mess up the filename, such as unprintable characters, space and `/`.

`sweep`
: A full 360° (or partial sector in sector mode) of data, typically collected at a fixed elevation angle. Most data acquisition software packages, such as IRIS, collect volume scan data this way. The sweeps are indexed 1, 2, 3, ... In the case where **ascope** is used for RVP10 operation, there is no concept of a sweep and the sweep number is set to 0. For RHI scanning, the concept of a sweep is the same, except that it is an elevation sweep rather than azimuth sweep.

`polarization`
: This is the transmit polarization. There are four choices: `H`, `V`, `H+V` (simultaneous H and V transmit) and `ALT` (alternating H and V transmit).

`-count:N`

Each file consists of the `IQ` data for all range bins for each pulse in the sweep.

Use `-count:N` to define how many pulses to display.

In the example in the **`-data`** section, the count is set to 1, that is, only the information for one pulse is displayed.

`-data`

Use the **`-data`** option to see the `IQ` values for each range bin.

This example has the time series values for 400 bins for the 101st pulse:

```
$ tsview RVP10.20031208.192519.074* -count:1 -skip:100 -data Site:RVP8
Pulse #101 at :19:25:19.406 8 DEC 2003 UTC, Az: 9.99, El:29.98
0: (-91.2 ,244 ) (-91.2, 0) (-80.4,209) (-87.4,149) (-82.2,150) (-79.9, 95)
6: (-84.2 ,157 ) (-81.4,182) (-81.6,100) (-83.8,276) (-79.2,331) (-83.9,220)
12: (-83.2 ,202 ) (-84.9, 53) (-78.7, 52) (-82.3,184) (-82.7,121) (-78.7,286)
. . .
390: (-90.2 ,244 ) (-83.5 ,228) (-83.7,264) (-86.4,181) (-85.4,182)  (-
84.8,206)
396: (-84.5, 19) (-118,233) (-81.0 ,149) (-96.5,256) (-90.9,249)
Bin 396 Bin 397 Bin 398 Bin 399 Bin 400
```

The label on the left is the index number of the first range bin on the line. The numbers displayed are the power of the pulse in dBm and the angle in degrees of the `IQ` vector.

-length:N

The **-length:N** option allows you to specify the maximum number of characters to display under the **-data** option. In this example, the length is set to 43 characters.

```
$  tsview RVP8.20031208.192519.074* -count:1 -skip:1 00 -data
-length :43  Site:RVP10
Pulse #101 at :19:25:19.406 8 DEC 2003 UTC, Az: 9.99, El:29.98
0: (-91.2 ,244 ) (-91.2, 0) (-80.4,209)
3: (-87.4 ,149 ) (-82.2,150) (-79.9, 95)
```

-skip:N

Use **-skip:N** to specify how many pulses to skip before starting the terminal listing. In this example, we wanted the 101st pulse, so we specified **-skip:100**.

-verbose

Use **-verbose** to see the detailed information contained in the headers. This was enabled in the example session. See Starting tsview sample session (page 335).

11.11 TS record data format

Each TS file recorded to disk contains a run of 1 or more pulses, which are from the same basic RVP10 configuration. In RVP10 nomenclature, this is called the **Acquisition Mode** (stored in the **rvptsPulseInfo** structure). Each time something changes, such as the PRF, the acquisition mode changes, and a new file is created. If there are no changes, the files are arbitrarily written every 200.000 pulses.

TS files consist of ASCII headers and binary data, shown in the following table. They start with the **rvptsPulseInfo** structure. This is followed by possibly many pulses. Each pulse has a **rvptsPulseHdr** structure followed by an array of 16-bit binary data.

Table 79 TS file format

File component	Description
<rvptsPulseInfo>	Variable size, even
<rvptsPulseHdr #1>	Variable size, even
Pulse Data #1	16-bit words, count from header
<rvptsPulseHdr #2>	Variable size, even
Pulse Data #2	16-bit words, count from header
...	

Each time series sample consists of 2 floating point numbers representing the **I** and **Q** voltages. The values are full magnitude with a value of 1. This represents +8 dBm on the IFDR, but may change in future revisions.

Floating point numbers are packed into 16-bit words using **High SNR** packed floating format. See Initiate processing (**PROC**) (page 383).

The 16-bit words are stored in the little-endian byte order that is native to the Intel processor chips common on PCs, which is the reverse of "Network order" used on sockets. The **tsview** displays the (**I**,**Q**) samples in power and angle format:

```
Power = 6dBm + 10 x log10[I^2 + Q^2]
Angle – atan2(Q, I)
```

The first time series sample number is from the burst pulse. This is followed by a sample from each range bin with data. The **iNumVecs** field in the **pulse hdr** indicates the total number of samples. If it is a dual polarization receiver system, this is duplicated for the second receiver (the **iVIQPerBin** field in the pulse **hdr**). The total number of bytes of data is:

```
Bytes = 2 x 2 x iNumVecs x iVIQPerBin
```

The number of samples can be different in each pulse in the same file. This is because the sampling stops when the next trigger arrives. If triggers are from an external source, the PRT may fluctuate.

To explain the **rvptsPulseInfo** structure, see the following example (for more information, see the **rvpts.h** header file):

```
rvptsPulseInfo start    The structure is bracketed by start and end
iVersion=0              Structure version number
iMajorMode=1            1:FFT, 2:Random Phase (see dsp.h)
iPolarization=1         Transmit polarization: 0:H, 1:V, 2:Alt, 3:H+V
iPhaseModSeq=0          See dsp.h
taskID.iSweep=0         Application sweep number
taskID.iAuxNum=0        Application auxiliary number
taskID.sTaskName=Ascope_DEFAULT    Application task name
sSiteName=RVP10        Site name of RVP10
iAqMode=161             Increments each time there is a change
iUnfoldMode=0           Dual-PRF flag, see PRF_* in dsp_lib.h
iPWidthCode=0           Pulse width index (0-3)
fPWidthUSec=1           Pulse width in microseconds
fAqClkMHz=35.9751       Acquisition clock rate
fWavelengthCM=10.7      Radar wavelength in cm
fSaturationDBM=6        Saturation power of the I & Q samples
fRangeMaskRes=125       Range mask resolution in meters
iRangeMask=33825 ...    Full range mask, up to 512 16-bit numbers
fNoiseDBm=-81.6584 -81.6584    Noise samples for the 2 channels
fNoiseStdvDB=-0.00540576 -0.00540576   Standard deviation of the noise samples
fNoiseRangeKM=525       Range at which the last noise was taken
fNoisePRFHz=250         PRF at which the last noise was taken
iGparmLatchSts=0 0      Latched status from GPARM command
iGparmImmedSts=21124 8963 771 19 0 0    Immediate status from GPARM command
iGparmDiagBits=0 0 0 0     Diagnostic results from GPARM command
sVersionString=8.04.4      Version of RVP10
fDBzCalibCx             dBZ0 for second polarization
fNoiseCalib[2]          Noise level at calibration, [2 polarizations]
fBurstCalib             Burst power at calibration
iAntStatusMask          Mask of what antenna status bits are available
fPWidthUSecPulse2       Width of pulse 2
fDBzCalibPulse2[2]      dBZ0 pulse 2 [2 polarizations]
fNoiseCalibPulse2[2]    Noise level at calibration, [2 polarizations]
fBurstCalibPulse2       Burst power at calibration
iFlags                  Bit 1 set if Hybrid Pulse recorded
fNoiseDBmPuse2[2]       Current noise level for pulse 2 [2 pols.]
rvptsPulseInfo end
```

The `rvptsPulseHdr` structure is also defined in the *`rvpts.h`* file. For example:

```
rvptsPulseHdr start iVersion=0
iFlags=3    Bit 0: N/A
               Bit 1: Gap before this pulse
               Bit 2: First pulse in trigger bank Bit 3: Last pulse in trigger
bank
               Bit 4: Trig bank (possibly unchanged) is just beginning Bit 5:
Triggers were blanked on this pulse
iMSecUTC=179        The data acquisition time ms
iTimeUTC=1071875957    The data acq. time in seconds since 1970, UTC
iBtime=2429100475      Ms time pulse inserted in API
iSysTime=45973182      Acq clock count of pulse acquisition
iPrevPRT=119917        Acq clock period from previous trigger
iNextPRT=119917     Acq clock period to next trigger
iSeqNum=287828      Sequence number of pulse in API
iAqMode=161         Acq Mode sequence number (8-bits)
iPolarBits=0        Polarization control bits
iTxPhase=182        Transmit phase 1 deg (16-bit binary angle)
iAz=16381           Azimuth=89.98 deg (16-bit binary angle)
iEl=179             Elevation=0.98 deg (16-bit binary angle)
iNumVecs=401        The number of TS samples in this pulse
iMaxVecs=401        The max possible number (1+#bins requested)
iVIQPerBin=1        1 for single polarization, 2 for dual
iTgBank=0           Trigger bank number
iTgWave=0           Trigger waveform sequence number
uiqPerm.iLong=0 0      User tag bits, permanent
uiqOnce.iLong=0 0      User tag bits, one time
RX[0].fBurstMag=3.58298e-05    Burst pulse amplitude, 1=full scale 0.446Volts
RX[0].iBurstArg=45561          Burst pulse phase difference (previous-this)
RX[1].fBurstMag=0    Second receiver burst info RX[1].iBurstArg=0
iAntStatus           Mask of current antenna status bits
iVecOffsetPulse2     Offset in the TS data to pulse 2 rvptsPulseHdr end
```

12. Technical data

12.1 Signal processing

Table 80 Signal processing

Feature	Description
Signal processor	Vaisala RVP10
Azimuth averaging	2–1024 pulses
IF digitizing	16 bits
Clutter filters	Fixed/Adaptive GMAP in frequency domain
Data outputs (8 and 16 bit)	`Ah/v`, `Azdr`, `CCOR`, `CSP`, `CSR`, `dBT`, `dBZ`, `dBZt`, `KDP`, `LDR`, `LOG`, `PHIH/V`, `PHIDP`, `PMI`, `R`, `RHOHV`, `SNR`, `SQI`, `T`, `V`, `VC`, `W`, `Z`, `ZC`, `ZDR`, `ZDRC`, `Zh`, `Zv`, `Zhv`
Optional data outputs	`HCLASS`, `I/Q`
Dual-polarization	Simultaneous (STAR), H only
Dual PRF velocity de-aliasing	2:3, 3:4, or 4:5 for 2X, 3X, or 4X velocity unfolding
Enhanced reflectivity processing (Zhv)	> 3 dB improvement detection gain
Processing modes	PPP, FFT/DFT, Random phase coding / SZ 8/64 phase coding for 2nd trip filtering and recovery
Maximum number of range bins	15,000
Minimum range resolution	Down to 15 m (accuracy of ±1.0 m)
Maximum range	Up to 1024 km

12.1.1 Processing algorithms

Table 81 Processing algorithms

Algorithm	Description
`I/Q` signal correction options	• Amplitude jitter correction based on running average of transmit power from burst pulse • Interference correction for single pulse interference • Saturation correction (3 ... 5 dB)

Algorithm	Description
Primary processing modes	• Poly-Pulse Pair (PPP) • DFT • Random or phase coded second trip echo filtering/recovery • Optional polarization with full co-variance matrix (**Zdr**, **PHIDP**, **LDR**, **RHOHV**, and so on) • Optional pulse compression
Processing options	• Azimuth averaging: 2 ... 1024 pulses • Corrections for gaseous attenuation and1/R2 • Dual-polarization: Alternating, Simultaneous, H-only, V-only • Clutter filters: Fixed/Adaptive GMAP in frequency domain, or IIR time domain • High sensitivity Rhv STAR mode processing: >3dB improvement in detectability • Pulse integration up to1024 • Range de-aliasing: random phase • Scan angle synchronization for data acquisition • Up to 4 pulse widths • Velocity de-aliasing: dual PRF velocity unfolding at 2:3, 3:4, and 4:5 for 2X, 3X, or 4X velocity unfoldings
Data outputs (8 and 16 bit)	**Zh**, **Zv**, **Zhv**, **V**, **W**, **SQI**, **ZDR**, **LDR**, **RHOHV**, **PHIDP**, and **KDP**
Optional data outputs	**HCLASS**, **I/Q**
Data quality thresholds	• Signal-to-noise ratio (SNR)—Used to reject bins having weak signals; typically applied to dBZ • Signal quality index (SQI)—Used to reject bins having incoherent signals; typically applied to mean velocity and width • Clutter-to-signal ratio (CSR)—Used to reject range bins having very strong clutter; typically applied to mean velocity, width, and dBZ • Speckle Filter—Removes single-bin targets such as aircraft or noise and fills isolated missing pixels

12.2 RVP10 Input and output summary

Table 82 I/O summary

Function	Description
AZ/EL angle input options	• Serial **AZ/EL** angle tag input using standard Vaisala RCP format • 16-bit each parallel TTL binary angles through the I/O-62 card • Synchro angle inputs through the I/O-62 card • RCP network antenna packet protocol

Function	Description
Ethernet I/O from host computer	Data output of calibrated `dBZ`, `V`, and `W` during normal operation. Full `I/Q` time series recording with a separate tsarchive utility, or through a custom application using a public API. Signal processor configuration and verification read-back is performed through the Ethernet interface.
Input from IFDR	• 32-bit floating point `I/Q` values • Optional dual-channel `I/Q` samples (for example, for polarization systems or dual-frequency systems)
Optional polarization control	RS-422 differential control for polarization switch
Trigger output	Up to 12 total triggers available on various connector pins. Triggers are programmable with respect to trigger start, trigger width, and sense (normal or inverted).

12.3 IFDR10 specifications

Table 83 IF bandpass filter

Function	Description
IF bandpass filter	Programmable digital FIR with software-selectable bandwidth, including pulse compression. Built-in filter design software with user interface.

Table 84 IF inputs

Characteristic	Description
A/D and D/A conversion	Resolution: 16 bit Sampling rate: 190 ... 240 MHz (software-selectable)
Dynamic range (dependent on matched filter)	104 dB without compression (1 µs pulse) 107 dB without compression (2 µs pulse)
IF range	10 ... 120 MHz
Impulse response	200 µs
Input signals	• IF received signal: 50 Ω, +12.0 dBm full-scale, absolute max +20 dBm • IF burst or STALO: 50 Ω, +12.0 dBm full-scale, absolute max +20 dBm • Optional reference clock: 9 – 100 MHz with 0.5 MHz resolution, 9.0 dBm ... +19.0 dBm

Characteristic	Description
Master clock jitter	< 0.5 picosec, integrated over 200 Hz ... 2 MHz offset
Multiply/accumulate cycles per second	448 billion
Pulse repetition frequency (PRF)	50 Hz ... 20 kHz, +0.1 %, continuously selectable
Saturation level (1 dB compression point)	+13.3 dBm at 50 Ω

Table 85 Digital waveform synthesis

Characteristic	Description
Analog waveform applications	• Digitally synthesized IF transmit waveform for pulse compression, frequency agility, and phase modulation applications. • Master clock or STALO signal to the radar, either phase-locked or free-running frequency.
TxDAC analog output waveform characteristics	Two independent, digitally synthesized, analog output waveforms (SMA) Max output power +10 dBm Frequency range 10 MHz ... 120 MHz

Table 86 RVP10 I/O

Characteristic	Description
Inputs (IFDR10)	Only digital inputs
Data output through Ethernet (IFDR10)	32-bit floating point I and Q values
I/O interface (IFDR10)	8 independent GPIO pairs Each pair can be configured as one differential input/output, or as two single-ended inputs/outputs (max 16 single-ended lines)
AFC output (RVP10SRV)	Serial control output Automatic 2D (time/frequency) burst pulse search and fine-tracking algorithms
IFDR10 to RVP10SRV link	Optical link (10 Gbit)

Table 87 Electrical properties of GPIO pins in single-ended input mode

Charasteristic	Description
Minimum high level input voltage	3.5 V
Maximum low level input voltage	1.5 V
Absolute maximum input voltage	5.0 V
Absolute minimum input voltage	0.0 V

Table 88 Electrical properties of GPIO pins in single-ended output mode

Charasteristic	Description
Output voltage high	5.0 V (I out = 0 mA)
	4.3 V (I out = -10 mA)
	3.6 V (I out = -20 mA)
Output voltage low	0.0 V (I out = 0 mA)
	0.7 V (I out = 10 mA)
	1.4 V (I out = 20 mA)

Table 89 Electrical properties of GPIO pins in differential mode

Characteristic	Decription
Differential input threshold voltage	-200 ... +200 mV
Absolute maximum and minimum differential input voltage	±5 V
Absolute maximum input voltage to gnd	5.0 V
Absolute minimum input voltage to gnd	0.0 V
Common mode voltage to gnd if not limited by absolute max/min input voltage	0.0 ... 5.0 V
Typical differential output voltage high	4.1 V (I out = 0 mA)
	2.8 V (I out = 10 mA)
	2.3 V (I out = 20 mA)
Typical differential output voltage low	-4.1 V (I out = 0 mA)
	-2.8 V (I out = 10 mA)
	-2.3 V (I out = 20 mA)

The differential GPIO voltage of a pin pair is defined as *the Odd pin voltage minus the Even pin voltage*. For example, if **`GPIO1 = 2 V`** and **`GPIO0 = 0.3 V`**, the differential GPIO voltage is **`2 V – 0.3 V = 1.7 V`**, and the differential pin state is true.

12.4 IFDR10 physical and environmental characteristics

Table 90 Physical and environmental characteristics

Characteristic	Description
Dimensions (w × l × h)	246 mm × 136 mm × 51 mm (9.7 in × 5.4 in × 2.0 in) With mounting brackets: 270 mm × 136 mm × 52.5 mm(10.6 in × 5.4 in × 2.1 in)
Input power	20 ... 30 VDC
Power consumption	Typical: 44 W ± 10 % Maximum: 56 W
Environment	-40 °C ... +55 °C (-40 °F ... +131°F) 0 %RH ... 95 %RH (non-condensing) with a minimum airflow of 0.6 m^3/min
Reliability (IFDR10)	MTBF > 100,000 hours (at 25 °C) < 1 hour MTTR

12.5 RVP10SRV signal processing computer specifications

Table 91 Physical and environmental characteristics

Characteristic	Description
Chassis	• 3U Short-depth rackmount chassis (depth 18.9”), Advantech HPC-7320 Chassis • Rack rail kit to accommodate 19” racks of depths 18” ... 36”
Cooling	• Fan: 2 (8 cm/57CFM) + 1 (6 cm/27.72CFM) • Air filter

Characteristic	Description

I/O	• PCI slot to allow for IO-62 card installation
	• DVD +/-RW drive
	• 2 Serial ports accessed from the back
	• 10 Gb Ethernet port accessed from the back
	• 3.0 USB ports
	  • 4 ports accessed from the back
	  • 2 ports accessed in the front
	• 2.0 USB ports, accessed from the back
	• PS/2 Single Connector with brake out cable for monitor and keyboard

Memory and processing	• 512 GB Solid State Drive configured with RAID 1 (mirrored) • 16 G of DDR4 Memory • 2 XEON E5-2609 v3 Processors with 6 Cores, 1.9G Hz clock speed, 6.4GT/s QPI Speed, 15M of Cache
Non-operating environment	Temperature: -40 ... +70° C (-40 ... +156° F) Humidity: 10 ... 95% at 60° C, non-condensing Vibration (5 ... 500Hz): 2 G
Operating environment	Temperature: 0 ... +40° C (32 ... +122° F) Humidity: 10 ... 95% at 40° C, non-condensing Vibration (5 ... 500Hz): 1 Grms Shock: 10 G (with 11 ms duration, half sine wave)
Physical characteristics	Dimensions (W * H * D): 426.4 * 132.2 * 480 mm (16.79″ * 5.2″ * 18.9″) Weight: 13.7 kg (30.1 lbs)
Power	Dual redundant power supplies auto-ranging 100 ... 240 V AC

12.6 Regulatory statements

Table 92 Regulatory statements for IFDR10

EU Directive	Standard
EMC Directive (2014/30/EU)	EN 61326-1:2013, Electrical equipment for measurement, control and laboratory use - EMC requirements - Part 1: General requirements. IFDR10 fulfills the requirements for Class B equipment.
Eye safety	Class 1 laser product, IEC/EN 60825-1:2014
Compliance marks	CE, UKCA

The following list of standards applies to the Advantech HPC-7320 server computer as manufactured. If any configurations are done to the server after manufacturing, the list may no apply.

Table 93 Regulatory statements for RVP10SRV

Item	Standard
EMC Directive (2014/30/EU)	
	IEC/EN 61000-6-4, Generic standards
	EN55011, Class A, Group 1 CISPR 32 / EN 55032, Class A
	EN 55024:2010+A1:2015 EN 55035:2017+A11:2020
	EN61000-3-2 / EN61000-3-3, harmonics and flicker
	EN61000-6-2, electromagnetic compatibility
	IEC/EN 61000-4-2, electrostatic discharge immunity
	IEC/EN 61000-4-3, radiated RF-field immunity
	IEC/EN 61000-4-4, electrical fast transient immunity
	IEC/EN 61000-4-5, surge immunity
	IEC 61000-4-6, conducted RF-field immunity
	EN 61000-4-8, testing and measurement techniques
	IEC 61000-4-11, voltage dips and short interruptions immunity
Compliance marks	CE

12.7 RVP10 spare parts

Table 94 IFDR10 spare parts

Part number	Description
IFDR10SP	Intermediate Frequency Digital Receiver without software
270934SP	SFP+ fiber optic module TX1271 for IFDR10
223150SP	Bandpass filter 30 MHz with coaxial cable
223151SP	Bandpass filter 60 MHz with coaxial cable
223152SP	Bandpass filter 57.5 MHz with coaxial cable
271595SP	Fiber optic cable, simplex SM, LC to LC, 5m

Part number	Description
260799SP	Power suppy AC/DC 24VDC for IFDR10

Table 95 RVP10SRV computer spare parts

Part number	Description
RVP10SRVSP	Signal prosessing computer
270935SP	10G optical Ethernet SFP+ module
RCP8-CPSP	IO62 panel RCP8
RVP8-IOSP	IO62 card
248382SP	Power supply for RVP10SRV

13. Writing a problem report

When troubleshooting the product, write a problem report including:

- What failed (what worked / did not work)?
- Where did it fail (location and environment)?
- When did it fail (date, immediately / after a while / periodically / randomly)?
- How many failed (only one defect / other same or similar defects / several failures in one unit)?
- What was done when the failure was noticed?
- What was connected to the product and to which connectors?
- Input power source type, voltage, and list of other items (such as lighting, heaters, and motors) that were connected to the same power output.
- Are all parts connected and grounded properly? Take a photo to help the troubleshooting.

This information is helpful if you need to contact Vaisala support.

Appendix A. IFRD10 technical drawings

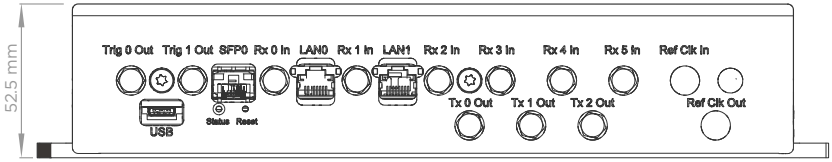


Figure 72 IFDR10 front view

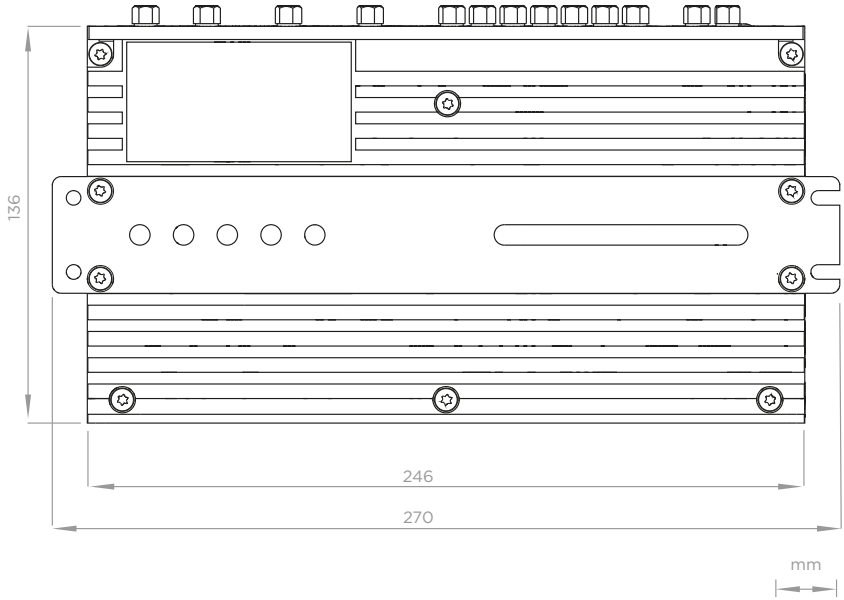


Figure 73 IFDR10 bottom view

Appendix B. Configuring softplane.conf file

B.1 Configuring the *softplane.conf* file

For manual configuration, configure the ***softplane.conf*** file to define pin–by–pin assignment of I/O functions to connectors on the I/O-62 connector panel as well as RVP901 IFDR and IFDR10.

The file is a commented plain text ASCII file. Since the RVP and RCP8 have virtually no jumpers, or wirewrap, all I/O configuration on the I/O-62 connector panel, RVP901 IFDR, and IFDR10 is done by software approach according to this file.

The file is in the ***IRIS_CONFIG*** directory, typically */etc/vaisala/irisrda* (this is the default directory that is factory installed). The empty template file is also available in the */etc/vaisala/irisrda_template* directory.

1. Login as **radarop**.
2. Type:

   ```
   $ cd /etc/vaisala/irirda
   ```

3. Launch one of the text editors provided in the system:
   - **$ gedit softplane.conf**
     This is a user-friendly editor with keyboard commands and mouse support when you are in X–Windows. It is a little easier to learn than **vi**.
   - **$ vi softplane.conf**
     This is a generic UNIX editor available on every UNIX system and familiar to many users.

B.2 *Softplane.conf* organization and syntax

The ***softplane.conf*** file defines the I/O pins on each connector on the PCI cards and connector panel, as well as RVP901 IFDR and IFDR10 connectors. There are two primary definitions for each pin:

- Physical interface: the electrical properties (RS422 output, analog input, TTL output, and so on).
- Logical interface: the internal variable name that is associated with each pin.

File syntax

- # at the beginning of a line indicates a comment.
  These are used for internal documentation. If you make changes, comment them, for example:

```
# TTL I/O on J7
 #
 # Modification by REP on 2 Apr 03
 # Added new interlock input on connector panel J7 pin07
 ...
```

> All user-defined comments will be cleared when running the command **`softplane -resave`**

- The top part of the file provides a list of internal variable names that are used to define the logical interface to the softplane. These are divided into status inputs (also called indicators) and control outputs (also called requests). For example, **`sPedAZ0`** corresponds to the LSB (lowest significant bit) of a digital azimuth angle relative to the radar pedestal. The following tables provide a summary of the available status and control variable names.
- Each definition line in the file starts with the keyword text:

```
# splConfig...
```

- The first uncommented line in the file indicates the version of the IRIS support software that was last used to machine-generate the file. This is an information-only field for traceability purposes and is not edited. For example:

```
# splConfig.sVersion = "10.0"
```

On the TTL connectors (J1, J2, J4, J5, J7), each connector must be exclusively used for **`INPUT (s vars)`** or **`OUTPUT(c vars)`**. You cannot mix these on an individual connector.

`softplane.conf` status and control signals

Signals available for use in the ***`softplane.conf`*** file are listed below. “Slow” status and control signals can be used on all hardware interfaces. “Live” signals are related to the hardware internal signals and controlled directly by the hardware itself. These can be used only on the listed devices, like IFDR10.

> The table is subject to change.

Table 96 softplane.conf control bits

softplane.conf control bits	Description
cPedAZ[15:0]	16 bits of antenna azimuth angle relative to the pedestal (fixed base system)
cPedEL[15:0]	16 bits of antenna elevation angle relative to the pedestal (fixed base system)
cEarthAZ[15:0]	16 bits of antenna azimuth angle relative to the earth (moving platform)
cEarthEL[15:0]	16 bits of antenna elevation angle relative to the earth (moving platform)
cServoPwr	To control servo power on
cCabinetRelay	To control a relay signal
cTransmitPwr	Request transmit power on
cPWidth[3:0]	Request one of four pulse widths
cvPWidth[3:0]	Request one of four pulse widths, RVP9
cTrigBlank	Trigger blanking signal
cRadiateOn	Request radiate on
cRadiateOff	Request for radiate off
cNoiseGenOn	Request Noise generator on
cSigGenOn	Request signal generator on
cSigGenCW	Request signal generator to output continuous wave
cSigGen[7:0]	Request outputs to control signal generator output level
cReset	Request a reset of external equipment
cIrisMode[2:0]	Request the application software (for example, IRIS) to switch to one of eight operating modes
cAux[80:0]	Arbitrarily assigned output requests
cStepCntAZ	
cStepCntEL	
cStepUpAZ	
cStepUpEL	
cDrcpComm[1:0]	
cDrcpActive	

softplane.conf control bits	Description
cvDAFC[23:0]	
true	Internal logic variable
false	Internal logic variable

The following table lists the *softplane.conf* status bits.

The table is subject to change

Table 97 softplane.conf status bits

softplane.conf status bits	Description
sPedAZ[15:0]	16 bits of antenna azimuth angle relative to the pedestal (fixed base system)
sPedEL[15:0]	16 bits of antenna elevation angle relative to the pedestal (fixed base system)
sServoPwr	Servo power on indicator
sLocal	Antenna local mode indicator, usually tied to an external local/remote switch
sStandby	Radar ready to radiate indicator
sLowerEL	Lower limit switch indicator
sUpperEL	Upper limit switch indicator
sProxSwAZ	
sProxSwEL	
sTransmitPwr	Transmitter cabinet power ON indicator
sTransmitLocal	Transmitter local mode indicator, usually tied to an external local/remote switch
sPWidth[3:0]	Indicator of the current pulse width
sTrigBlank	Indicator that trigger blanking is requested, usually from an external source
sRadiate	Radiate ON indicator
sAirflowFlt	Cooling airflow fault indicator
sWavegpFlt	Waveguide pressure fault indicator
sInterlockFlt	Master interlock fault indicator

softplane.conf status bits	Description
sMagCurrentFlt	Transmitter overload fault indicator
sNoiseGenOn	Indicator of noise generator being turned on
sSigGenOn	Indicator of signal generator outputting signal
sSigGenCW	Indicator of signal generator is set to output continuous wave
sSigGen[7:0]	Indicators of the signal generator output power level
sSigGenFlt	Indicator of signal generator fault
sReset	Request for reset coming from external source
sIrisMode[2:0]	Information on which operating mode is active in the application software
sDrcpComm[1:0]	
sDrcpEnable	
sAux[319:0]	Arbitrary status indicators

Each piece of hardware is identified as either in use or not in use.

```
splConfig.Io62[0].InUse = 1 (if in use)
     splConfig.Io62[0].InUse = 0 (if unused or not installed)
```

Specify the method of connecting to the I/O-62, for example:

```
splConfig.Io62[0].sExtPanel = "IO62CP"
```

The options are:

Connection type	softplane descriptor
Direct connect to I/O-62 via 62 pin connector	`DIRECT`
I/O-62 Connector panel (used for RVP8 and RCP8)	`IO62CP`
WSR88D connector panel, RVP8 portion	`RVP88D`
WSR88D connector panel, RCP8 portion	`RCP88D`

- The assignments for each connector and each pin are then made. For convenience, these are usually grouped together by connector. For example, if Pin 1 of connector J1 on the I/O-62 connector panel is assigned to be the LSB of the input azimuth angle, then:

```
# TTL/CMOS on J1
 #
splConfig.Io62[0].Opt.Cp.J1.pin01 = "sPedAZ[0]"
```

- The notation **""** indicates that no assignment is made.

```
# BNC testpoint
monitors#splConfig.Io62[0].Opt.Cp.J13_BNC = ""
```

- In the example above, the pin name is **J13_BNC**. Put a ~ in front of a logic variable to invert the variable.

```
splConfig.Io62[0].Opt.Cp.J1.pin03 = "~sPedAZ[2]"
```

B.3 Configuring *softplane* for IFDR10

Enable the RVP10 I/O in softplane if not already enabled:

```
plConfig.Rvp10.lInUse = 1
```

Resave the softplane:

```
 $ softplane -resave
```

This command populates the ***softplane.conf*** file with empty RVP10 GPIO configuration lines.

```
# -------------------- RVP10IFD #0 --------------------
#
# If you change the in-use flag, run 'softplane -resave' to rev the choices.
#
splConfig.Rvp10.lInUse = 1
# In addition to all of the standard logical signals, the
# following realtime 'live' signals may be assigned to the
# RVP10 interface pins.
#
#         Control Outputs            Status Inputs
#        -----------------          ---------------
#              tgBlanked               tgBlankReq
#          trigger[10:1]                 tgExtern
#           txPhase[7:0]
#
# Restrictions to pin assignments:
# - Pins within a pair can be only either inputs or outputs.
# - Pins within a pair can be either TTL or differential.
# - tgExtern input can be assigned to only one pin.
# - tgExtern can't be inverted. It must be configured in dspx.
splConfig.Rvp10.ifdr10.ttl[0].pin = ""
splConfig.Rvp10.ifdr10.ttl[1].pin = ""
splConfig.Rvp10.ifdr10.ttl[2].pin = ""
splConfig.Rvp10.ifdr10.ttl[3].pin = ""
splConfig.Rvp10.ifdr10.ttl[4].pin = ""
splConfig.Rvp10.ifdr10.ttl[5].pin = ""
splConfig.Rvp10.ifdr10.ttl[6].pin = ""
splConfig.Rvp10.ifdr10.ttl[7].pin = ""
splConfig.Rvp10.ifdr10.ttl[8].pin = ""
splConfig.Rvp10.ifdr10.ttl[9].pin = ""
splConfig.Rvp10.ifdr10.ttl[10].pin = ""
splConfig.Rvp10.ifdr10.ttl[11].pin = ""
splConfig.Rvp10.ifdr10.ttl[12].pin = ""
splConfig.Rvp10.ifdr10.ttl[13].pin = ""
splConfig.Rvp10.ifdr10.ttl[14].pin = ""
splConfig.Rvp10.ifdr10.ttl[15].pin = ""
# NOTE! Only even numbered differential pins can be configured!
splConfig.Rvp10.ifdr10.diff[0].pin = ""
splConfig.Rvp10.ifdr10.diff[1].pin = ""
splConfig.Rvp10.ifdr10.diff[2].pin = ""
splConfig.Rvp10.ifdr10.diff[3].pin = ""
splConfig.Rvp10.ifdr10.diff[4].pin = ""
splConfig.Rvp10.ifdr10.diff[5].pin = ""
splConfig.Rvp10.ifdr10.diff[6].pin = ""
splConfig.Rvp10.ifdr10.diff[7].pin = ""
splConfig.Rvp10.ifdr10.diff[8].pin = ""
splConfig.Rvp10.ifdr10.diff[9].pin = ""
splConfig.Rvp10.ifdr10.diff[10].pin = ""
splConfig.Rvp10.ifdr10.diff[11].pin = ""
splConfig.Rvp10.ifdr10.diff[12].pin = ""
splConfig.Rvp10.ifdr10.diff[13].pin = ""
splConfig.Rvp10.ifdr10.diff[14].pin = ""
splConfig.Rvp10.ifdr10.diff[15].pin = ""
# NOTE! Trig connectors can only be configured as outputs.
splConfig.Rvp10.ifdr10.trig[0].pin = ""
splConfig.Rvp10.ifdr10.trig[1].pin = ""
```

Table 98 softplane.conf “Live” inputs

softplane.conf “Live” inputs	Description
tgBlankReq	External request to blank triggers
tgExtern	External trigger request input

Table 99 softplane.conf “Live” outputs

softplane.conf “Live” outputs	Description
tgBlanked	Triggers are blanked
trigger[10:1]	Trigger1-10 output signal
txPhase[7:0]	TX signal phase code

When the softplane configuration is taken in use for RVP10, the configuration of the pins is forwarded to IFDR10. Based on this configuration, the directions and types of the pins are configured to the hardware, and “live” signals are routed to the selected pins inside the IFDR. The configuration of the IFDR is done through the gRPC interface using the IFDR settings in JSON format. The IFDR settings structure includes the section shown below as an example configuration.

The GPIO configuration is grouped to 8 pin pairs and 2 independent trigger I/Os. Each pin pair can be either output or input, and the pair type can be either differential (true) or single-ended (false). Within a pin pair, these parameters cannot be mixed. For example, if pin_n is output, then pin_p must also be output. Trigger I/O connectors can only be configured to be outputs.

```
"gpio_configuration": {
    "gpio_pairs": [
      {
        "output": true,
        "pair_type_differential": true,
        "pin_n": {
          "function": "TRIGGER0",
          "inverted": false,
          "variable_name": "trigger[1]"
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false,
          "variable_name": ""
        }
      },
      {
        "output": true,
        "pair_type_differential": true,
        "pin_n": {
          "function": "TRIGGER0",
          "inverted": false,
          "variable_name": "trigger[1]"
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false,
          "variable_name": ""
        }
      },
      {
        "output": true,
        "pair_type_differential": true,
        "pin_n": {
          "function": "TRIGGER0",
          "inverted": false,
          "variable_name": "trigger[1]"
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false,
          "variable_name": ""
        }
      },
      {
        "output": true,
        "pair_type_differential": true,
        "pin_n": {
          "function": "TRIGGER0",
          "inverted": false,
          "variable_name": "trigger[1]"
        },
        "pin_p": {
          "function": "EMPTY",
          "inverted": false,
          "variable_name": ""
        }
      },
      {
```

```
          "output": false,
          "pair_type_differential": false,
          "pin_n": {
            "function": "EMPTY",
            "inverted": false,
            "variable_name": ""
          },
          "pin_p": {
            "function": "EMPTY",
            "inverted": false,
            "variable_name": ""
          }
        },
        {
          "output": false,
          "pair_type_differential": false,
          "pin_n": {
            "function": "EMPTY",
            "inverted": false,
            "variable_name": ""
          },
          "pin_p": {
            "function": "EMPTY",
            "inverted": false,
            "variable_name": ""
          }
        },
        {
          "output": false,
          "pair_type_differential": false,
          "pin_n": {
            "function": "EMPTY",
            "inverted": false,
            "variable_name": ""
          },
          "pin_p": {
            "function": "EMPTY",
            "inverted": false,
            "variable_name": ""
          }
        },
        {
          "output": false,
          "pair_type_differential": false,
          "pin_n": {
            "function": "EMPTY",
            "inverted": false,
            "variable_name": ""
          },
          "pin_p": {
            "function": "EMPTY",
            "inverted": false,
            "variable_name": ""
          }
        }
      ],
      "trigger_io": [
        {
          "output": true,
          "pin": {
```

```
          "function": "TRIGGER0",
          "inverted": false,
          "variable_name": "trigger[1]"
        }
      },
      {
        "output": false,
        "pin": {
          "function": "EMPTY",
          "inverted": false,
          "variable_name": ""
        }
      }
    ]
}
```

B.4 From softplane.conf to IFDR10 settings

In normal use, the ***softplane.conf*** file is the master source for configuration. When RVP10 process is started, it calls the softplane library to load the configuration from the ***softplane.conf*** file. RVP10 will then update the IFDR settings with this configuration and save them to the IFDR10. After this, when IFDR10 starts up without RVP10 running, it will start using the correct "live" signals configurations.

B.5 Restoring the softplane.conf from IFDR10 settings

It is possible to rebuild the ***softplane.conf*** file from the IFDR settings. Do not start RVP10 before this, as it will overwrite the settings in IFDR10.

1. Read the settings from IFDR10 to a file:

   ```
   $ ifdr_get_settings > /etc/vaisala/irisrda/ifdr10_settings.json
   ```

2. Import settings to softplane:

   ```
   $ softplane –ifdr10import /etc/vaisala/irisrda/ifdr10_settings.json
   ```

Appendix C. RVP10 programming interface

C.1 RVP10 programming interface overview

RVP10 includes a set of commands that you use to set up and control the RVP10 processor for recording data.

Commands consist of an initial command word containing an opcode in the low five bits. If additional arguments are required, they are listed as `Input 1`, `Input 2`, and so on. If the command produces output, those words are listed as `Output 1`, `Output 2`, and so on.

Often each word is broken down into independent fields, each consisting of one or more bits. All data transferred to or from RVP10 are in the form of 16-bit words.

Unless otherwise noted, the command descriptions describe set bits.

C.1.1 Setting up data acquisition and processing

Although the internal RVP10 tables and parameters are set to reasonable values on power-up, you usually need to modify the default information to meet your needs.

1. On power-up, issue **IOTEST** to ensure that the interface connections are all intact.
2. Check the diagnostic result registers from **GPARM** to verify that RVP10 passes all internal checks.
3. Establish trigger and pulse width using the **SETPWF** commands.
4. Chose range bin placement and processor options using **LRMSK** and **SOPRM**.
5. Take receiver noise samples with **SNOISE**.

   The noise levels are not automatically sampled on power-up, you must issue **SNOISE** at least once.
6. If clutter filters are needed, issue **LFILT**.
7. If data rays are to be synchronized with antenna motion, issue **LSYNC** to specify a table of antenna angles.
8. When the setups are complete, issue **PROC** commands to collect, process, and output the data.

   To monitor errors detected during the execution of commands using the **GPARM** command.

C.1.2 First-in-first-out (FIFO) buffer

RVP10 contains a 4096-word first-in-first-out (FIFO) buffer through which all output data flow.

The FIFO holds each sequential word generated by RVP10 until the user is ready to accept it. When reading from the processor, the host can fall behind by as many as 4096 words before performance slows.

RVP10 writes to the FIFO at full speed as long as it is not full. Internal processing is not affected by the exact speed at which user I/O occurs.

Writing to the FIFO continues as long as the average I/O rate on, perhaps 10 ms intervals, and matches the average rate at which data are being produced.

Full output FIFO

The sequence of events changes when the FIFO is full:

1. When the processor generates the next output word, it waits in an idle loop until the user makes room in the FIFO by reading out one or more words.
2. RVP10 waits and does not proceed with its internal processing until space becomes available.

Despite the slowdown in performance, you always obtain correct data no matter how long it takes to read it. You can take advantage of this to synchronize the acquisition of data by the RVP10 with the post-processing and display of that data by the user.

In this case, RVP10 would be instructed to output data at the maximum rate, you can read these words at your maximum rate, and the overall system automatically runs at the slower of those two speeds.

Write cycle and full output FIFO

If the output FIFO is full and the RVP10 has the next word ready for output, the idle wait loop can be exited if the processor detects that the user is performing a write I/O cycle.

Since the user should have been reading data by now, the presence of a write cycle is understood to mean that a more important condition has arisen. The wait loop terminates, and RVP10 accepts the write data soon afterward. If the new data are commands, they are executed right away, but any output they try to produce may be lost in a similar manner. The processor continues to execute all commands correctly, but that their output is discarded.

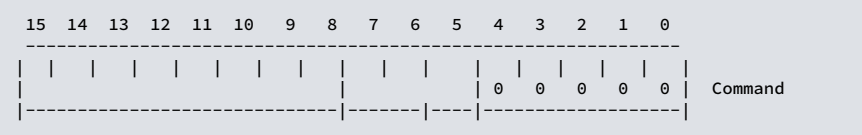
The discarded output data are not lost. The data are eventually replaced with an equal number of zeros. Each time RVP10 discards an output word, it increments an internal 24-bit count. When FIFO space becomes available, the processor replaces the missing data with zero-valued placeholders.

Writing when the FIFO is full can be useful if the new command is a **RESET** which calls for clearing the output FIFO. When the **RESET** is processed, all past and present output data are discarded, leaving the RVP10 output section empty. This is useful when the processor has pending output data that the user wants to discard.

C.2 No-Operation (**NOP**)

NOP is useful when a number of words must be flushed through RVP10 without side effects.

This single-word instruction is ignored by the signal processor.



C.3 Load Range Mask (**LRMSK**)

LRMSK informs the signal processor of the ranges at which data are to be collected.

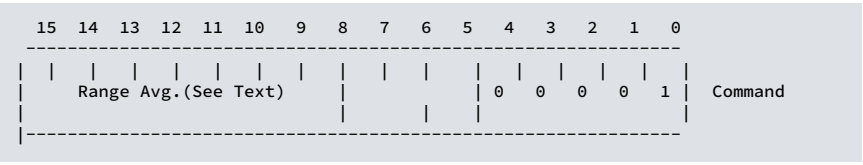
An arbitrary set of range bins are selected through an 8192- bit mask. The **N**th bit in the mask determines whether data are acquired and processed at a range equal to:

```
RES x (N-1)
```

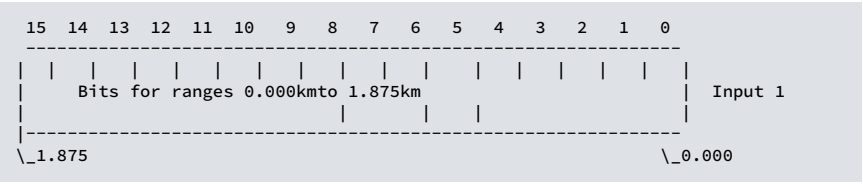
Table 100 **LRMSK** parameters

Range parameters	Description
Range resolution	Specified by a TTY setup question , in the range 25 ... 1000 meters. Any collection of ranges may be chosen from integer multiples of that distance. See **Mt<n>**— Transmit sequence #n (page 95).
Range mask	The range mask is passed to RVP10 packed in 512 16-bit words. In each group of 16: • The least significant bit of each packed word represents the nearest range. • The most significant bit represents the furthest range. According to the range bins that are selected in the mask, the signal processor computes and stores internally a range normalization table which is later used to convert receiver intensity levels into reflectivity levels in dBZ. Note that **LRMSK** implicitly specifies the number of bins to be processed and output. The maximum bin count is 4200, though depending on the computational intensity of the configuration, RVP10 may be able to compute fewer bins. If the number of bins selected in the bit mask exceeds the maximum, the trailing bins are truncated. If the new mask does not specify any active bins, then a single bin at range 0 is forced on. The default power-up mask selects 256 bins equally spaced by 1.0 km starting from 0 range.

Range parameters	Description
Range averaging	**LRMSK** determines range averaging. The upper byte of the command controls how many consecutive bins are grouped together. For example: • `0`: No averaging For example, if 100 bits are selected in the range mask and no averaging is elected. Then parameters are computed at those 100 ranges, and 100 bins of data are output. • `1`: Pairs of samples are averaged If the averaging is to 1, samples are taken at the same ranges, but pairs of bins are averaged together and only 50 ranges would result. Note that the parameters are averaged by summing the autocorrelations for each bin. The individual samples that go into each average are taken according to the bits that are set in the mask, except that they are grouped together so that only one net bin results from the several data samples. Note that the limitation of 4200 sampled ranges applies to the bin count prior to averaging. • `255`: 256 terms are summed The range normalization value associated with the averaged bin is computed according to the midpoint of the first and last sample. The command discards incompletely averaged bins. Continuing from the previous example, if the averaging were set to 2 so that triples of samples were summed, then only 33 bins would be output. This is because the 100-bit mask left a dangling 100th sample. In the extreme case where there are not enough mask bits to result in even one complete bin, the RVP10 forces the averaging to zero and turns on a single bin at zero range.



In the following examples, the ranges listed in Inputs 1 ... 512 assume a 125 km range bin.



```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|     Bits for ranges 1022.000 km to1023.875 km                 |  Input 512
|                            |       |   |                    |
|-----------------------------------------------------------------
\_1023.875                                                      \_1022.000
```

C.4 Setup operating parameters (**SOPRM**)

Use **SOPRM** to configure the signal processor. You must issue **SOPRM** when any of the parameters in the list change.

The default parameter list consists of 20, 16-bit input words. These can be followed by optional **XARG** parameters as needed.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |           |   |   |   |   |   |
|                        |NTH|           | 0   0   0   1   0 |  Command
|------------------------|---|-----------|-------------------|
```

If `Nth` is `1`, then no threshold values are set. This means the system ignores input words `4`, `5`, `6`, `7`, `11`, `12`, `13`, `14`, and `18`. This is usually used with the **THRESH** command when setting individual thresholds. See Set Individual Thresholds (**THRESH**) (page 440).

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
|                        Sample Size                            |  Input 1
|------------------------|---|-----------|-------------------|
```

The sample size is continually adjustable from 1 ... 256 pulses. In the alternating polarization mode, the sample size must be even, if an odd value is entered, it is rounded up by one.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
|  |ZNS| Polar |NHD |ASZ|16B|CMS|R2 |   |3x3|CCB|   |Lsr|Dsr|Rnv|  Input 2
|--|---|-------|----|---|---|---|---|---|---|---|---|---|---|---|-
```

Each single-bit field selects whether the given processing or threshold option is enabled (`1`) or disabled (`0`).

Table 101 **SOPRM** threshold options

Threshold option	Description
ZNS	If Rnv is 0 (no range normalization), you can set ZNS to have the dBZ and dBT outputs to be power relative to noise (P/N) rather than SNR ((PN)/N). This format is useful when collecting data that are near or below noise because there is no discontinuity at the noise level. ZNS has no effect when Rnv is set.
Polar	Configures transmit polarization and Zdr processing: 00 Fixed polarization, Horizontal 01 Fixed polarization, Vertical 10 Alternating polarization pulse-to-pulse 11 Dual simultaneous transmission
NHD	Disables the inclusion of header words in the processed data that are output by the **PROC** command. See also Configure ray header words (**CFGHDR**) (page 433).
ASZ	The Any Spectrum Size bit requests that DFT processing algorithms, clutter filters, spectral output, and so on. All operate on spectra whose size exactly matches the number of available pulses (rather than rounding the spectrum size down to the next lower power-of-two).
16B	Configures for 16-bit (rather than 8-bit) data output from the PROC command. This bit affects the single-parameter versions of Reflectivity, Velocity, Width, and Zdr data. The **PROC** command's archive format always holds 8-bit data, regardless of the 16B setting. This gives the option of extracting 8-bit and 16-bit data simultaneously from each ray.
CMS	Enables clutter microsuppression, in which individual range bins are rejected (based on excessive clutter) prior to being averaged together in range.
R2	Use 3 lag (R0/R1/R2) algorithms for width, signal power, and clutter correction.
3×3	Switches on the 3x3 2D output filter. See Speckle filters applied to the thresholded data (page 299). RVP10 automatically handles all of the pipelining overhead associated with running the 3×3 filter. Valid output data are always obtained in response to every **PROC** command.

Threshold option	Description
CCB	Circular autocorrelation bias correction. Setting this bit causes non-windowed spectra to produce autocorrelation terms that exactly match those that would be computed by traditional PPP sums, that is, with the spurious end-around term removed.
Lsr	Lsr 1D reflectivity speckle remover. When set, range speckles in dBT, dBTa, dBZ, dBZa, SNR, ZDR, LDRH, and LDRV are removed.
Dsr	Dsr 1D Doppler speckle remover. When set, range speckles in V, W, PhiDP, PhiH, PhiV, RhoHV, RhoH, RhoV, and KDP are removed.
Rnv	Range normalization of reflectivity data. This bit also enables intervening gas attenuation correction.

HClass is 1D speckle-filtered when Lsr or Dsr is set.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               Log Slope65536 * dB / (LSB)                     |  Input 3
|---------------------------------------------------------------|
```

This number defines the multiplicative constant that converts the signal power in dB to the units of the 12-bit Log of power in sample time series outputs. One fourth (1/4) of this slope is used to generate the Log of Measured Noise Level output from **GPARM** (word 6). The recommended value is 0.03 (1966). This gives a dynamic range of 122 dB in 12 bits.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|             LOG Threshold in 1/16 of dB                       |  Input 4
|---------------------------------------------------------------|
```

Reflectivity values below this level can result in thresholding of data, if the threshold control flags (see below) include LOG Noise bits. The threshold value is always non-negative. For the comparison test, see Thresholding (page 296).

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|       Clutter Correction (CCOR) Threshold in 1/16 of dB       |  Input 5
|---------------------------------------------------------------|
```

The clutter correction threshold is a bound on the computed log receiver adjustment for clutter. These corrections (in dB) are always negative. Any clutter correction which is more negative than the above value can result in thresholding of data.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                              |        SQI Threshold            |  Input 6
|------------------------------|---------------------------------|
```

The signal quality index (SQI) threshold is an unsigned binary fraction in the range 0 ... 255/256. When the SQI for a range bin falls below the stated value, it may result in thresholding of data. An analogous Polarimetric Meteo Index (PMI) can be set by the **THRESH** command. See Set Individual Thresholds (**THRESH**) (page 440).

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|      Weather Signal Power Threshold in 1/16 of dB               |  Input 7
|-----------------------------------------------------------------|
```

Weather signal power (**SIG**) is an estimate of the SNR of the weather component of the received signal. When the **SIG** falls below this comparison value, it may result in data thresholding. See Weather Signal Power (SIG threshold) (page 193).

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|          Calibration Reflectivity in 1/16 of dB                 |  Input 8
|-----------------------------------------------------------------|
```

The calibration reflectivity is referenced to 1.0 kilometers.

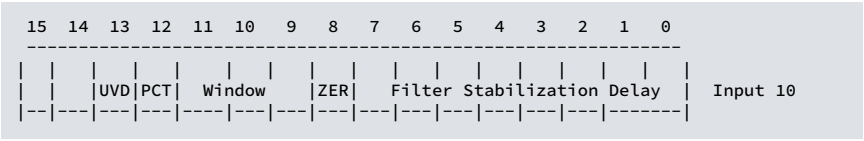
```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|              |   TopMode     |                                 |  Input 9
|--------------|---------------|---------------------------------|
```

The **TopMode** bits select the overall data acquisition and processing mode for RVP10. Although the processing algorithms used in each top level mode are different, the RVP10 command set works in a uniform way in all modes.

Table 102 **TopMode** bits

Bits	Description
0000	Polarimetric processing (PPP) mode is a combined time domain and frequency domain approach that is used primarily for dual polarization applications. Data are processed in batches of pulses. See Time series signal processing (page 168).
0001	FFT processing mode is a dedicated frequency-domain approach; data are processed in batches of pulses. See Frequency Domain Processing- Doppler power spectrum (page 171).

Bits	Description
0010	Random phase processing mode. Data from first and second trips are dealiased in range based on knowledge of the radar transmitter phase. See Random phase second trip processing (page 198).
0100	DPRT-1 processing mode. The trigger generator produces alternate short and long pulses, and Doppler autocorrelations are computed using only the short pairs.
0101	DPRT-2 processing mode. The trigger generator produces alternate short and long pulses, and Doppler autocorrelations are computed using both pairs.
11XX	Reserved for custom user modes.



The RVP10 clutter filters are controlled by this word.

Table 103 RVP Clutter Filter Controls

Control	Description
Delay	This delay is introduced prior to processing the next ray of data when Dual-PRF velocity unfolding is enabled or the RVP10 has been reconfigured by user commands. The delay permits the clutter filter transients to settle down following PRF and gain switches. The value is specified as the number of pulses, and hence, the number of filter iterations, to wait.
ZER	If set, then the clutter filter's internal state variables are zeroed prior to waiting the delay time. For some signal conditions, this may give better results than allowing the filter to naturally flow into the new data.
Window	Selects the type of window that is applied to time series data prior to computing power spectra with a DFT. Choices are: • 0:Rectangular • 1:Hamming • 2:Blackman • 3:Exact Blackman • 4:VonHann
PCT	If set, RVP10 attempts to run its standard processing algorithms even when a custom trigger pattern has been selected with the **SETPWF** command.
UVD	Unfold velocities using a simple (Vhigh Vlow) algorithm, rather than the standard algorithm described in Dual-PRF unfolding (page 303) .

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|     Threshold Control Flags for UnCorrected Reflectivity      |  Input 11
|---------------------------------------------------------------|
```

These flags select which legacy threshold comparisons result in uncorrected reflectivity being accepted or rejected at each bin. There are 4 test comparisons made at each range, as described above for input words **4**, **5**, **6**, and **7**. Further quality tests such as the Polarimetric Meteo Index can be configured for each data type. See Set Individual Thresholds (**THRESH**) (page 440).

Each test either passes and produces a code of **1**, **2**, **4**, and **8** respectively, or fails and produces a code of **0**. The sum of the codes for each of the 4 tests is a number between 0 and 15, which can also be interpreted as the following four-bit binary number:

```
  3   2   1   0
-----------------
| 8 | 4 | 2 | 1 |
|---|---|---|---|
   \   \   \   \_______ LOG Threshold Passes
    \   \   \_________ CCOR Threshold Passes
     \   \_____________ SQI Threshold Passes
      \_________________ SIG Threshold Passes
```

The individual bits of the **Threshold Control Flag** word specify whether data are to be accepted (**1**) or rejected (**0**) in each of the possible combination of threshold outcomes. The pattern of bits in the flag word represents a truth table for a given logical function of the four threshold outcomes.

The following examples show values of the **Flag** word for the stated combinations of acceptance criteria:

Table 104 Example flag values with acceptance criteria combinations

Value	Criteria
FFFF	All Pass (Thresholds disabled)
0000	All Fail (No data are passed)
AAAA	LOG
8888	LOG and CSR
A0A0	LOG and SQI
8080	LOG and CSR and SQI
F0F0	SQI
FAFA	SQI or LOG
C0C0	SQI and CSR

Value	Criteria
F000	SQI and SIG
C000	SQI and SIG and CSR
FFF0	SQI or SIG
CCC0	(SQI or SIG) and CSR

One way to generate these values is to imagine four 16-bit quantities having the following names and values: LOG=AAAA, CSR=CCCC, SQI=F0F0, SIG=FF00. The flag value needed to represent a given logical combination of threshold outcomes is obtained as the result when that same logical combination is applied to these special numbers.

For example:

```
(SQI or SIG) and CSR   =   (F0F0 or FF00 ) and CCCC
                       =   (FFF0) and CCCC
                       =   CCC0
```

which corresponds with one of the examples given above.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|     Threshold Control Flags for Corrected Reflectivity        |  Input 12
|---------------------------------------------------------------|
```

See description for **Input 11**.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|     Threshold Control Flags for Velocity                      |  Input 13
|---------------------------------------------------------------|
```

See description for **Input 11**.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|     Threshold Control Flags for Width                         |  Input 14
|---------------------------------------------------------------|
```

See description for **Input 11**.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|     Additive Offset for Measured AZ Angles (Binary Angle)     |  Input 15
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|     Additive Offset for Measured EL Angles (Binary Angle)     |  Input 16
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|     Intervening Gas Attenuation Correction (dB/km)            |  Input 17
|---------------------------------------------------------------|
```

Gas attenuation correction attempts to compensate for overall (two-way) beam losses due to absorption by atmospheric gases. The correction is linear with range, and is added to the data along with range normalization. Therefore, clearing the RNV bit in Word #2 above disables the correction. Gas attenuation compensation can be turned off when RNV is on by setting a slope of 0.0 dB/km.

An attenuation of G db/km is encoded into the unsigned 16-bit word **N** as follows:

$0 \le N \le 10000 \;\; G = \frac{N}{100000}$

else

```
G = 0.1 + (N – 10000)/10000
```

This format is backward-compatible with the previous linear format for all values between 0.0 ... 0.1 dB/km and it extends the upper range of values from 0.65535 ... 5.6535. These larger attenuation corrections are needed for very short wavelength radars.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Threshold Control Flags for Differential Reflectivity(Zdr) |  Input 18
|---------------------------------------------------------------|
```

See description for **Input 11**.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Signed Zdr Calibration Offset in 1/16 dB(GDR)              |  Input 19
|---------------------------------------------------------------|
```

When differential reflectivity is computed there is a possibility that radar asymmetries introduce a bias in the **Zdr** values, that is, that **Zdr** is non-zero even when observing purely spherical targets. This calibration offset permits nulling out this effect. The GDR offset accounts for the overall **Tx/Rx** gain imbalance between the two channels of the radar.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Radar Wavelength in Thousandths of Centimeters             |  Input 20
|---------------------------------------------------------------|
```

The radar wavelength is used in the calculation of 16-bit velocity and width data, to convert from Nyquist units to absolute physical units.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Signed LDR Calibration Offset in 1/100 dB(XDR)             |  XARG 1
|---------------------------------------------------------------|
```

The XDR offset is used in the linear depolarization ratio equations, and is the differential receiver gain between the two channels. Unlike the GDR offset (used for `Zdr`), the gain difference does not depend on differential transmit power.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Range smoothing (0:None, 1:pairs, etc)                     |  XARG 2
|---------------------------------------------------------------|
```

Range smoothing can be performed on raw moment data prior to the computation of scientific parameters. The number of bins to sum together is given here. This should generally be an odd integer so that no range bias is introduced by the smoothing operation.

```
 15   14   13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |                      |                       |   |   |   |
|ENA|                      |Point Clutter Thresh   | Side Skip |  XARG 3
|-- |----------------------|-----------------------|-----------|
```

Point clutter detection is configured with this word. A bin is flagged as containing clutter if its power exceeds that of its two neighboring bins by more than the detection threshold (in decibels). Up to 7 bins may optionally be skipped on each side of the central bin prior to making these two comparisons.

`Ena`
: This bit is set to enable point clutter detection. Flag bits are reported in the `Flg` output data type of the **PROC** command.

```
 15  14   13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|ENA|ALL |                                                       |  XARG 4
|-- |----|-------------------------------------------------------|
```

Point clutter censoring is configured with this word.

`Ena`
: Set to enable point clutter censoring. Raw moment data containing point clutter are interpolated from valid signal levels on either side.

`All`
: Optionally expand the reported detection flags to show the entire replaced interval, not just the original detected bins. This gives a more honest view of the data bins that have been altered.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Bit mask of TS playback mismatch conditions to ignore       |  XARG 5
|---------------------------------------------------------------|
```

This word is a combination of `MMTS_xxx` bits, defined in *`dsp.h`*, specifying what types of mismatches are okay (do not cause an all-zero ray to be produced) during **`PROC`** command processing of timeseries data that are played back from an external source into RVP10.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Flag bits related to timeseries (playback)                  |  XARG 6
|---------------------------------------------------------------|
```

Combination of `OPTS_xxx` bits, , defined in *`dsp.h`*, which modify details of time series behavior.

```
 15   14   13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |    |   |   |   |   |   |    |   |   |   |   |   |   |     |
| 1 |    Offset in deciBels       |  Slope as 100 * dB/dB       |  XARG 7&8
|-- |-----------------------------|-------------------|---------|
```

These words allow you to set the breakpoints and slopes that modify the `LOG` threshold according to the Clutter-to-Noise ratio of the target. This makes the `LOG` threshold behave properly even as the noise floor becomes elevated due to very strong clutter targets. A value of 0 restores the RVP10 defaults from the **`Mf`** menu.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Melting Height in Meters Above/Below Mean Sea Level         |  XARG 9
|---------------------------------------------------------------|
```

During time series playback, the height recorded with the time series is used instead of this value.

The MSB is complemented in this signed number. This means that a value of `0` is the most negative value and that the burst power was unknown. This is used to compute the long term burst power calibration adjustment.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Burst power at time of calibration in 1/100 of dBm         |  XARG 10
|---------------------------------------------------------------|
```

The default (power-up) values for the above parameters are listed below. Both the scientific units and the integer-input required by the command to set up that value are given. Most of these defaults are suitable for most radars.

Table 105 Default values for melting height operating parameters

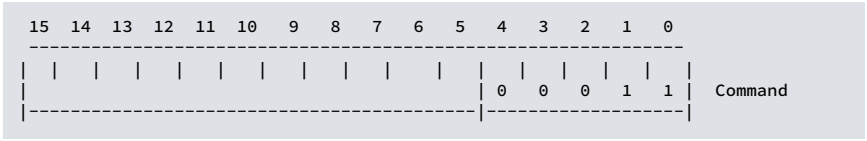
Parameter	Scientific units	Input
Sample Size	25 pulses	25
Flag Word		0007 Hex
Log Slope	0.03 dB/LSB	1966
LOG Threshold	0.5 dB	8
CCOR Threshold	25.0 dB	400
Signal Quality Index Threshold	0.5 (dimensionless)	128
SIG Threshold	10.0 dB	160
Calibration Reflectivity	22.0 dBZ	352
Gas Attenuation	0.016 dB/km	1600
Zdr Offset (GDR)	0.0 dB	0
LDR Offset (XDR)	0.0 dB	0
AGC Integration Period	8 pulses	8
Radar Wavelength	5.3 cm.	5300
Dual PRF Filter Stabilization	10 pulses	10
UnCor Refl. Thresh. Control Flag	LOG	AAAA Hex
Cor Refl. Thresh. Control Flag	LOG and CSR	8888 Hex
Velocity Thresh. Control Flag	SQI and CSR	C0C0 Hex
Width Thresh. Control Flag	SQI and CSR and SIG	C000 Hex
Zdr Refl. Thresh. Control Flag	LOG	AAAA Hex
AZ/EL Angle Offsets	0°	0000 Hex
Altitude of radar	0 meters MSL	0

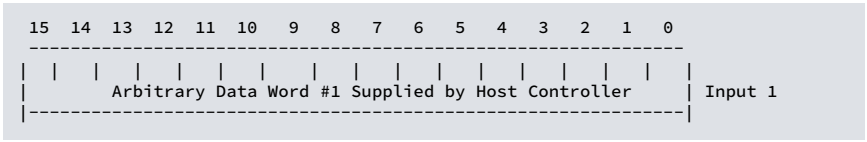
C.5 Interface Input/Output Test (**IOTEST**)

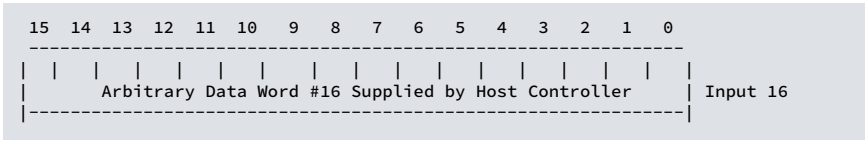
Use **IOTEST** to test the input and output data busses of the signal processor interface.

When issued, the command causes 16 words to be read from the host controller, after which the same 16 words are printed.

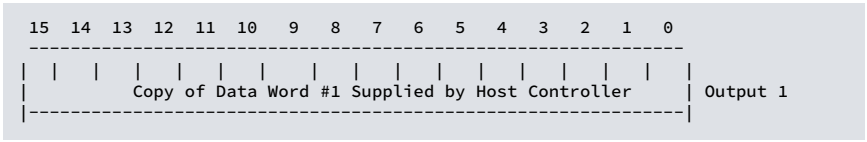
Typically, the controller supplies a "barber pole" input sequence consisting, for example, of successive powers of two. If all of the output words are correct, one may conclude that there are no malfunctioning bits in the interface hardware

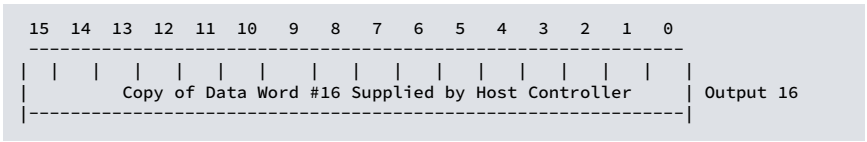






The **IOTEST** command can also process and echo up to 128 additional **XARGS** data words. See Pass auxiliary arguments to opcodes (**XARGS**) (page 431).





C.6 Interface Output Test (**OTEST**)

Use **OTEST** to test the integrity of the data being output by the signal processor. The command causes 16 words to be output consisting of successive powers of two starting from one.

By verifying whether each output word is correct, you can isolate malfunctioning bits in the interface data bus.

This test is less stringent than the input/output test **IOTEST** since the input data paths to the processor are not being checked. Typically, the **OTEST** is performed only when the **IOTEST** fails to determine whether the fault was on input or output.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                                       | 0   0   1   0   1 |  Command
|---------------------------------------|-------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   1 | Output 1
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | Output 16
|---------------------------------------------------------------|
```

C.7 Sample noise level (**SNOISE**)

Use **SNOISE** to estimate the current noise level from the receiver, so that the noise can be subtracted from subsequent measurements.

Data are sampled for 256 pulses at 256 bins, beginning at a selectable range and spaced by the range resolution at that pulse width. The internal trigger generator is temporarily set to a special noise rate (usually much lower than the operating rate) during the process. It is the user's responsibility to make sure that no returned power is present within the approximately 32 km sampling interval. In some cases, it may be necessary to raise the antenna during the noise measurement to avoid thermal noise pickup from the ground, or from weather targets.

SNOISE has the option of setting up a new sampling range and trigger generator rate each time it is called. Two bits in the command word determine which, if any, of the new values overrides the current values stored in RVP10. The power-up sampling range is 250 km (input value of 250), and the power-up trigger rate is 200 Hz (input value of 30000). These initial values persist until such time as they are altered here. Both input words must always be supplied after the command, even if the command calls for ignoring one or both of them. The

range is supplied directly in kilometers up to a maximum of 992 km. The trigger rate, resulting from a given input, is 6 MHz divided by the input value, that is, the input value is the trigger period in 0.1667 µsec increments. Note that the given rate is bounded against the minimum PRT allowed for the current radar pulse width.

The **SNOISE** command bounds the requested starting range of the noise sampling interval. This is to make sure the noise samples fit within the specified PRT, and within the range mask hardware RAM. RVP10 sets an error bit when an improper range is requested.

Reissue the **SNOISE** command periodically to compensate for drift in the RF and A/D systems.

The noise levels must be measured for the RVP10 to properly process data. This can be done by issuing the **SNOISE** at least once after power-up, or by setting the correct values for the power-up noise levels with the **Mt** setup command (see **Mt<n>**— Transmit sequence #n (page 95)). RVP10 does not automatically take a noise sample as part of its initialization procedure.

The measured offsets are stored internally for all subsequent uses in RVP10. The offset values may be inspected through the **GPARM** command, as may the current range and rate values themselves. When the range or rate are changed, the user must make sure that the new trigger rate allows at least 32 km following the new noise range. If this requirement is not met, or if other failures are detected during the noise measurement, appropriate bits are set in the**GPARM** latched status word. This word should generally be checked after **SNOISE** to make sure that everything worked properly.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|              |   Action  |Rat|Rng|        | 0   0   1   0   1 |  Command
|--------------|-----------|---|---|--------|-------------------|
```

`Rng`
: If 1, the range in input word 1 is taken as the starting noise range for this and all subsequent **SNOISE** calls.

`Rat`
: If 1, the trigger rate in input word 2 is taken as the noise rate for this and all subsequent **SNOISE** calls.

`Action`
: Specifies what action is carried out by the command.

`0`
: Compute a new noise sample based on the present IFD input signals.

`1`
: Do not compute a noise sample, but rather, read new noise values from the host computer and use them for subsequent processing. Four additional input words supply the noise information, and **GPARM** words 6, 9, and 44 .. 50 are changed to reflect the new noise settings.
If the current transmit pulse is a hybrid pulse, and XARGS arguments are supplied, then the additional arguments set the noise level for the second pulse.

2
Do not compute a noise sample, but rather, restore the powerup noise defaults.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Starting Range in km (Max 992km) of 32km Sampling Interval  |  Input 1
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|  Internal Trigger Rate (6Mhz/N) to use During Noise Sampling  |  Input 2
|---------------------------------------------------------------|
```

The following input words are optional, only if `Action=1`.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|0   0 | (MSB)        Log of Measured Noise Level(LSB)          |  Input 3
|------|-----------------------------------------------------|
```

The is the same number as **GPARM** output 6. See Initiate processing (**PROC**) (page 383) and Get processor parameters (**GPARM**) (page 399).

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|      Noise Level Standard Deviation (in 1/100 of a dB)        |  Input 4
|---------------------------------------------------------------|
```

This is the same number as the **GPARM** output 49.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|  Ratio of Horizontal/Vertical Noise Power in Hundredths of dB |  Input 5
|---------------------------------------------------------------|
```

This is the same number as the **GPARM** output 50.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               <Spare>                          |Err|Ttf|Ntg|  Input 6
|------------------------------------------------|---|---|---|
```

The following XARGS input words are optional, only if `Action=1` and hybrid pulse.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|0   0 |  (MSB) Log of Measured Noise Level, puse 2 (LSB)       |  XARG 1
|------|---------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   NOISE Level Standard Deviation, pulse 2 (in 1/100 of a dB)  |  XARG 2
|------|---------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Ratio of Horizontal/Vertical Noise, pulse 2 (in 1/100 of a dB)|  XARG 3
|------|---------------------------------------------------------|
```

These fault bits are outputted in the latched status **GPARM** word 9.

`Ntg`
: No Trigger during noise measurement.

`Ttf`
: Trigger too fast during noise measurement, that is, some of the noise sample bins were positioned past the trigger range.

`Err`
: Error detected during the **SNOISE** command.

C.8 Initiate processing (**PROC**)

The **PROC** command controls the processing and output of radar data.

PROC is a single-word command that specifies the type of processing to be performed, and the type of output to be generated.

The following table shows the modes available in the command word.

Table 106 **PROC** Modes

Mode	Description
Synchronous mode	The processor acquires, processes, and outputs one ray in response to each **PROC** command. Processing begins after each command is received.
Free running mode	A single **PROC** command is issued, and rays are continually output as fast as they can be produced and consumed. This continues until any other command is written, for example, a **NOP** can be used to terminate the free running mode with no other consequences.

Mode	Description
Time Series mode	The processor acquires, processes, and outputs one ray of time series samples in response to each **PROC** command. Data are output as 8-bit time series, 16-bit time series, or 16-bit power spectra.

Optional dual-PRF velocity unfolding is chosen by command bits 8 and 9. For Doppler data either a 2:3, 3:4, or 4:5 PRF unfolding ratio may be selected. RVP10 performs the unfolding steps internally, so mean velocity is output with respect to the larger unambiguous interval. No additional velocity processing is needed except to change the velocity scale on any generated displays.

Spectral widths are scaled consistently with respect to the higher PRF, and require no user modification before being plotted.

When unfolding is selected, the internal trigger generator automatically switches rates on alternate rays. The switch over occurs immediately after the last pulse of the current ray has been acquired; thus overlapping the internal post-processing and output time, with transmitter stabilization and data acquisition at the new rate.

Output data are selected by the upper 6 bits of the **PROC** command. Packed archive output is selected by setting the ARC bit. Individual byte or word display output is selected by setting any or all of the Z, T, V, W, Zdr, and Kdp command word bits. When more than one of these bits is set, the output array consists of all of the bins for the leftmost selected parameter, followed by all of the bins for the next selected parameter, and so on. Bits selected in XARG #1 behave the same way, except that the output order is right-to-left. Both archive and display formats can be selected simultaneously, in which case the archive format is output first, followed by whichever individual display format values were also selected. The archive format is not recommended for use with new drivers, because it can only handle four of the many possible output parameter types.

In time series mode, there are three output data formats available. For backwards compatibility, there is an 8-bit integer format, in which the 8 most significant bits from the I, Q, and LOG signals are represented in a byte. This format is not recommended, because it generally misses weak signals. Vaisala recommends the floating-point format that uses 16-bits per A/D sample. There is also a 16-bit power spectrum output that is accurate to 0.01 dB (see also **GPARM** output word #10).

In addition to the above output data, the first words of each ray optionally contain additional information about the ray. These header words are configured by the CFGHDR opcode, and are included only if the NHD (No-Headers) bit in **SOPRM** Input #2 is clear.

For example, if TAG angle headers are requested, if the ARC, Z, and V bits are all set, and if there are 100 bins selected in the current range mask, then each RVP10 output ray consists of the following:

```
1]  TAG15  TAG0      \      From Start of Acquisition
2]  TAG31  TAG16     /      Interval
3]  TAG15  TAG0      \      From End of Acquisition
4]  TAG31  TAG16     /      Interval
*   200 words of packed archive data.
*   100 words of Corrected Reflectivity data in low byte only.
*   100 words of Velocity data in low byte only.
```

The **Command** word format for Synchronous Doppler Mode is:

```
  15   14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|ARC| Z | T | V | W  |ZDR| Unfold|KDP| 0   1 | 0   0   1   1   0 |  Command
|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|-
```

The **Command** word format for Free Running Doppler Mode is:

```
  15   14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|ARC| Z | T | V | W  |ZDR| Unfold|KDP| 1   0 | 0   0   1   1   0 |  Command
|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|
```

Either of these may be augmented by an optional **XARG** word. See Pass auxiliary arguments to opcodes (**XARGS**) (page 431).

```
  15   14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |  Tx Vert  |  Tx Horz  |   |   |   |
|                   |HCL |FLG|Phi Rho Ldr|Phi Rho Ldr|SQI|RHV|PDP|  XARG1
|---|---|---|---|----|----|---|---|---|---|---|---|---|---|---|---|
```

The **Unfold** parameter selects the dual-PRF unfolding scheme:

```
00 : No Unfolding    01 : Ratio of 2:3
10 : Ratio of 3:4    11: Ratio of 4:5
```

The **ARC** selects the archive output format in which four data bytes (see 8-Bit descriptions below) are packed in two output words per bin as follows:

```
 HighByte  Low Byte
--------------------
|         |         |
|   V     |    Z    |
|         |         |
--------------------

--------------------
|         |         |
|   W     |    T    |
|         |         |
--------------------
```

The remaining data parameters are available in both 8-Bit and 16-bit formats, according to **SOPRM** `Command Input word #2` (see Setup operating parameters (**SOPRM**) (page 368)). The same **SOPRM** word configures RVP10 for single or dual polarization. The latter is required for **Kdp**, **PDP**, and **RHV** to be computed properly.

Table 107 **PROC** 8-bit and 16-bit Data Formats

Parameter	Description	8-bit Format	16-bit Format
V	Selects radial velocity data.	Mean velocity, expressed as a fraction of the unambiguous velocity interval, is computed from the unsigned byte **N** as: $V\frac{m}{\text{sec}} = V_{Nyquist} \times \frac{(N-128)}{127.5}$ `0` Indicates velocity data is not available at this range `1` Maximum velocity towards the radar `128` Zero velocity `255` Maximum velocity away from the radar When velocity unfolding is selected, the output is still interpreted as above, except that the unambiguous interval is increased by factors of 2, 3, and 4 for 2:3, 3:4, and 4:5 unfolding.	Mean velocity in meters per second (m/s) is computed from the unsigned word **N** as: $V\frac{m}{\text{sec}} = \frac{(N-32768)}{100}$ `0` Indicates velocity data is not available at this range `1` -327.67 m/s (towards the radar) `32768` 0.00 m/s `65534` +327.66 m/s (away from the radar) `65535` Reserved Code

Parameter	Description	8-bit Format	16-bit Format
W	Selects spectral width data.	Spectral width is computed from the unsigned byte N as: $W_{Nyquist} = \frac{N}{256}$ The overall range is a fraction between 1/256 to 255/256 of the unambiguous interval. The code of zero indicates that width data was not available at this range.	Spectral width in meters per second (m/s) is computed from the unsigned word N as: $W\frac{m}{\text{sec}} = \frac{N}{100}$ The overall range is from 0.01 m/s to 655.34 m/s in one cm/s steps as follows: 0 Indicates width data is not available at this range 1 : 0.01 m/s 65534 655.34 m/s 65535 Reserved Code
Z	Selects clutter corrected reflectivity data.	The level in decibels is computed from the unsigned byte N as: $dBZ = \frac{(N-64)}{2}$ The overall range is therefore from -31.5 dBZ to +95.5 dBZ in half-dB steps as follows: 0 Indicates no reflectivity data available at this range 1 : -31.5 dBZ 64 0.0 dBZ 128 +32.0 dBZ 255 +95.5 dBZ	The level in decibels is computed from the unsigned word N as: $dBZ = \frac{(N-32768)}{100}$ 0 Indicates no reflectivity data available at this range 1 -327.67 dBZ 32768 0.00 dBZ 65534 +327.66 dBZ 65535 Reserved Code
T	Selects total reflectivity.	Same 8-bit and 16-bit coding formats as for clutter corrected reflectivity	

Parameter	Description	8-bit Format	16-bit Format
ZDR	Selects differential reflectivity data.	The level in decibels is computed from the unsigned byte N as: $dBZ = \frac{(N128)}{16}$ The overall range is from -7.935 dB to +7.935 dB in 1\16 dB steps as follows: `0` Indicates no reflectivity data available at this range `1` -7.9375 dB `128` 0.0000 dB `255` +7.9375 dB	Same as 16-bit decibel format for Z.

Parameter	Description	8-bit Format	16-bit Format
KDP	Selects dual polarization specific differential phase data.	Values are coded into an unsigned byte using a logarithmic scale. The KDP angles are multiplied by the wavelength in cm (to reduce dynamic range) and then converted to a log scale separately for both signs. The minimum value is 0.25 deg*cm/km. The maximum value is 150.0 deg*cm/km. A code of zero represents no data A code of 128 represents 0 deg*cm/km. The conversion equation for positive values (codes from 129 to 255) is: $KDP \times \lambda = 0.25$ $\times 600\left[\frac{N-129}{126}\right]$ The conversion equation for negative values (codes from 1 to 127) is: $KDP \times \lambda = -0.25$ $\times 600\left[\frac{127-N}{126}\right]$	Same as 16-bit decibel format for Z, except that the units are hundredths of degrees per kilometer. No weighting by wavelength is introduced.

Parameter	Description	8-bit Format	16-bit Format
PDP	Selects dual polarization differential phase PDP data.	The phase angle in degrees is computed on a 180° interval from the unsigned byte **N** as: $\phi DP(\mathrm{mod}\,180) = 180\frac{(N-1)}{254}$ **0** Indicates no PDP data available at this range **1** 0.00° **254** 179.29° **255** Reserved Code	The phase angle in degrees is computed on a 360° interval from the unsigned word **N** as: $\phi DP(\mathrm{mod}\,360) = 360\frac{(N-1)}{65534}$ **0** Indicates no PDP data available at this range **1** 0.00° **65534** 359.995° **65535** Reserved Code
RHV	Selects dual polarization correlation coefficient RHV data.	The correlation coefficient is computed on the interval 0.0 ... 1.0 using a square root weighting of the unsigned byte **N** as: $P_{HV} = \sqrt{\frac{(N-1)}{253}}$ **0** Indicates no RHV data available at this range **1** 0.0000 (dimensionless) **2** 0.0629 **253** 0.9980 **254** 1.0000 **255** Reserved Code	The correlation coefficient is computed on the interval 0.0 to 1.0 linearly from the unsigned word **N** as: $P_{HV} = \frac{(N-1)}{65533}$ **0** Indicates no RHV data available at this range **1** 0.0 (dimensionless) **65534** 1.0 **65535** Reserved Code
SQI	Selects Signal Quality Index data.	This dimensionless parameter uses the same 8-bit and 16-bit data formats as **RHV**.	

Parameter	Description	8-bit Format	16-bit Format
LDR	Selects Linear Depolarization Ratio, measured either on the horizontal receive channel while transmitting vertically, or on the vertical receive channel while transmitting horizontally.	The level in decibels is computed from the unsigned byte N as: `dB = -45.0 + (N-1) / 5` This spans an asymmetric interval around zero decibels, and allows for cross channel isolation as large as 45 dB. The range is from -45.0 ... dB in 0.2 dB steps as follows: `0` Indicates no LDR data available at this range `1` -45.0 dB `226` 0.0 dB `254` +5.6 dB `255` Reserved Code	Same as 16-bit decibel format for Z.
RHO	Selects Signal Quality Index data	This dimensionless parameter uses the same 8-bit and 16-bit data formats as RHV.	
PHI	Selects the cross channel differential phase.	This parameter uses the same 8-bit and 16-bit angular data formats as PDP.	
FLG	Selects flag word output.	Bits defined as follows: `0` Reflectivity obscured at this bin `1` Velocity obscured at this bin `2` Width obscured at this bin `3` Point clutter detected at this bin	

Parameter	Description	8-bit Format 16-bit Format
HCLASS	Hydrometeor Classification (HydroClass) parameter.	There are several possible classification schemes. The choice is made in the *hydroclass-*-band.conf file*, where * is C (for C-band and X-band radars) and S (for S-band radars). The legacy Meteo classifications (up to IRIS/RDA 8.12.6) are: 0 No measurement available 1 Non-meteorological target 2 Rain 3 Wet snow 4 Snow 5 Graupel 6 Hail Higher bits of the HCLASS data fields may contain results from further methods of classification. See sig_data_types.h and HCLASS data description in *IRIS Programming Guide (M212927EN)*.
SNR	Signal-to-Noise ratio on the primary (horizontal) channel.	Uses the same storage format as Z.
Ta	Total power in the alternative polarization receive channel (usually vertical).	Uses the same storage format as T.
Za	Clutter corrected reflectivity in the alternative polarization receive channel (usually vertical). Uses the same storage format as Z. The command word format for time series mode is shown below.	

Parameter	Description	8-bit Format	16-bit Format
TSOUT	Selects type of data to be output.	00 8-bit Time Series 01 Power Spectrum 10 16-bit Time Series 11 Unused	

```
 15   14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ------------------------------------------------------------------
|   |   |   |   |   |   |       |   |   |   |   |   |   |   |   |
| TSOUT |     Sub Type    |Unfold |   | 1   1 | 0   0   1   1 |0  |  Command
|---|---|-----------------|-------|---|-------|---------------|
```

When the **TSOUT** bits select **Power Spectrum** then, depending on the current major mode, a further choice may be needed to select one of several spectral view points. The following table shows the values for the random phase major mode the possible values of **Sub Type**.

Table 108 **TSOUT** Random Phase Major Mode Values

First Trip Value	First Trip Description	Second Trip Value	Second Trip Description
0	Raw First Trip	4	Raw Second Trip
1	Whitened First Trip	5	Whitened Second Trip
2	Cleaned First Trip	6	Cleaned Second Trip
3	Final First Trip	7	Final Second Trip

When the **TSOUT** bits select **Time Series** then, a further choice may be needed to select the time series from the first or second pulse when using the Hybrid Pulse Compression scheme. For the Random Phase major mode the possible values of **Sub Type** are:

0
: Main pulse time series

1
: Second pulse time series (if hybrid pulse)

When time series output is selected the output data consist either of (**3xBxN**) or (**2xBxN**) words, depending on the output format, where B is the number of bins in the current range mask and **N** is the number of pulses per ray. Data samples for each bin of pulse **#1** are output first, followed by those for each bin of pulse **#2**, and so on up to pulse **#N**. The data are output in the same time-order that they were acquired.

In the floating point format, three words are used for each bin:

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Exponent      | S |              Mantissa                 |  (I)
|------------------|---|---------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Exponent      | S |              Mantissa                 |  (Q)
|------------------|---|---------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0  0   0   0 |             Log of Power in Sample            |  (LOG)
|--------------|-----------------------------------------------|
```

To convert these legacy format floating `I` and `Q` samples to voltages:

1. Create a 12-bit signed integer in which bits 0 ... 9 are copied from the `Mantissa` field, and bits 10 and 11 are either `01` or `10` depending on whether `S` is `0` or `1`.
2. Multiply this number by 2**(exponent-40), where the exponent field is interpreted as an unsigned 5-bit integer.
3. Multiply by the maximum voltage.

The resulting value has 12-bits of precision and a dynamic range of approximately 190 dB. The large dynamic range is necessary to cover the full range of data. In summary:

$$Voltage = V_{MAX} \times \left(Sign,\ Mantissa\right) \times 2^{Exponent - 40}$$

The resulting voltage span is ± 4 × V_{MAX}. The extra factor of 4 is built into the format so that transient excursions above the full scale input voltage can be encoded properly.

A `High-SNR` packed floating format is also available that offers nearly the same dynamic range, but provides a 6 dB improvement in SNR, that is, a commensurate improvement in sub-clutter visibility of -78 dB versus -72 dB.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Exponent      | S |              Mantissa                 |  High-SNR
|------------------|---|---------------------------------------|
```

The `High-SNR` packed format is similar to the legacy packed format except that it uses one extra mantissa bit and one fewer exponent bit. The dynamic range lost in the exponent is recovered through a formatting trick known as "soft underflow", that is, the mantissa is allowed to become unnormalized when the exponent is 0.

To decode this format when the exponent is non-zero, first create a 13-bit signed integer in which bits zero through ten are copied from the Mantissa field, and bits eleven and twelve are either 01 or 10 depending on whether `S` is 0 or 1. Then, multiply this by 2**(exponent-25), where the exponent field is interpreted as an unsigned 4-bit integer.

To decode the `High-SNR` format when the exponent is 0, interpret the mantissa as a 12-bit signed integer and multiply by 2**-24.

A complete analysis of the noise properties of the floating point codes would be fairly tricky. For the `High-SNR` format, the 12-bit mantissa with hidden normalization bit vary from 2048 ... 4095. The SNR therefore varies from 66 dB ... 72 dB and we can assign a mean value of 69 dB. Another 9 dB of useful range is contained within the code as follows:

- In a floating point encoding format, the notion of fixed additive quantization noise is not really correct. For a signal having a given power, the additive noise within each instantaneous sample scales down according to the magnitude of that sample. The ensemble of noise terms thus contributes an RMS power that is smaller than the Peak-to-Noise ratio would imply. In the case of a sinusoidal input, this gives a 3 dB boost in effective SNR.
- The format, of course, also represents negative amplitudes with the same relative precision as positive values. In a fixed-point format this would add 6 dB (one more bit) to the overall dynamic range and large-signal SNR. In the floating format we really only gain 3 dB (half a bit) because the RMS noises add independently on the positive and negative excursions.
- The packed format is used to encode time series (`I`,`Q`) pairs, and it's the SNR properties of these pairs that we're really concerned about. To a first approximation, having a pair of values roughly doubles the information content and adds another 3dB to the SNR.

The last of the time series output words, the `Log of Power in Sample`, is provided for backwards compatibility. It can be calculated from the `I` and `Q` numbers. To convert to dBm it requires a slope and offset as follows:

$$dBm = P_{MAX} + Slope \times [Value - 3584]$$

where:

P_{MAX}
: +4.5 dBm for 12-bit IFDR, +6.0 dBm for 14-bit IFDR, +8.0 dBm for 16-bit IFDR

V_{MAX}
: 0.5309 V for 12-bit IFDR, 0.6310 V for 14-bit IFDR, 0.7934 for 16-bit IFDR

`Slope`
: `Log Power Slope` word 3 of the **`SOPRAM`** command. 0.03 is recommended.

For backwards compatibility, RVP10 produces a 8-bit fixed point time series format. Because of the limited dynamic range available, this only shows strong signals, and is not recommended for use. The `I`, `Q`, and `Log power` triplets are packed in two 16-bit output words as follows:

```
 HighByte    Low Byte
----------------------
|          |          |
| Q Sample | I Sample |  First Word
|          |          |
----------------------

----------------------
|          |          |
|   Zero   | Log Power|  Second Word
|          |          |
----------------------
```

The **Log Power** value is the upper 8 bits of the long format. The other numbers are produced by the equation:

$$Voltage = V_{MAX} \times \left[\frac{Sample}{128}\right]$$

When Power Spectrum output is selected, the spectrum size is chosen as the largest power of two (**N2**) that is less than or equal to the current sample size (**N**). When the sample size is not a power of two, a smaller spectrum is computed that by averaging the spectra from the first **N2** and the last **N2** points. The data format is one word/bin/pulse, in the same order as for time series output. Each word gives the spectral power in hundredths of dB, with 0 representing the level that would result from the strongest possible input signal (P_{MAX}). Thus, the spectral output terms are almost always negative.

The time series that are output by RVP10 are the filtered versions of the raw data, when available. If a non-zero time-domain clutter filter is selected at a bin, then the **I** and **Q** data for that bin show the effects of the filter. If you must observe the raw samples, make sure that no clutter filters are being applied.

In pulse pair time series mode with dual receivers, selecting (**H+V**) produces data in one of two formats according to the **Sum H+V Time Series** question in the **Mp** setup section:

- **Yes** produces summed time series from both channels, but spectra from the DSP is the averaged spectra from each channel individually.
  This allows the IRIS ascope utility to display either the spectrum-of-sum or sum-of-spectra according to whether the **Spectra from DSP** button is selected in the **Processing/Gen-Setup** window.
- **No** produces the usual (**BxN**) time series output samples, except that the first half of these samples is the first half of the **H** data in their normal order. This is followed by a zero sample if (**BxN**) is odd; followed by the first half of the **V** data, also in their normal order.

Only the first halves of the individual **H** and **V** sample arrays are output by RVP10. As an example, if you select 25 bins and 100 pulses, then the output data consists of 1250 **H** samples (from all bins in the first 50 pulses), followed by 1250 **V** samples from the exact same set of bins and pulses. This is the more useful option when custom algorithms are being run on the data from the two separate receivers.

When the number of output words is large there is a possibility that the internal buffering within RVP10 may overflow and data may be lost. Due to internal memory limitations, the product (**BxN**) must be less than 12000. A bit in the latched status word indicates when time series overflows occur. In such cases, the correct number of words are still output, but they are all 0 after the point at which overflow was detected. See Get processor parameters (**GPARM**) (page 399)

C.9 Load clutter filter flags (**LFILT**)

In the RVP10 processor, you can choose any available clutter filters independently at each selected range.

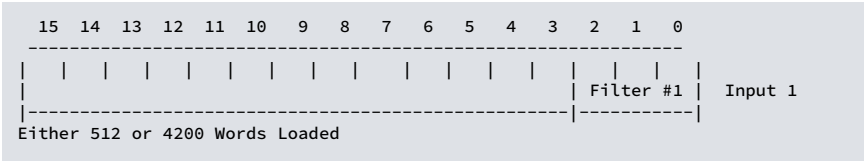
This range-dependent clutter removal is useful when the clutter characteristics vary with increasing range. Typically, clutter interference is most severe in the immediate vicinity of the radar. You can choose a highly rejective filter for near ranges, and a less rejective, or perhaps no filter, at far ranges.

Legacy version

In the legacy version of **LFILT**, the input words following the command specify the choice of filter to be applied at each of the selected range bins.

A fixed size filter table is always loaded, regardless of whether the range mask (See Load Range Mask (**LRMSK**) (page 366)) is using the full number of bins. In such cases, the later filter codes are ignored for the current range mask. However, if a longer range mask is loaded in the future, then those later codes would apply to the correspondingly numbered bins. In sum, each filter code is associated with a particular bin number, not with a particular range. The correspondence between bin numbers and ranges is made only through the range mask.

Only the low three bits are used in each word to specify the filter number. If the **L3K** bit is set in the **Command**, then 4200 words are loaded, otherwise only 512 words are loaded. In both cases the last filter choice is replicated for all bins further in range.



Enhanced version

RVP10 supports an enhanced version of the **LFILT** command that provides clutter maps, that is, much greater flexibility by allowing filter choices to depend on antenna angle as well as range. This lets you specify a 2D or even 3D table of clutter filter selections that are dynamically selected during live data processing.

RVP10 maintains an internal array of up to 1024 different filter- versus-range tables, each of which is keyed to a particular solid angle AZ/EL sector. Each enhanced **LFILT** command fills in one of these slots with a filter selection table similar to that of the legacy command, except that the number of range bins is specified explicitly and eight bits are used for filter selection rather than three. Then, for each live ray being processed, the RVP10 applies clutter filters according to the filter slot whose solid sector includes the midpoint AZ/EL of the ray. If the antenna angle of the ray does not fall within any of the defined filter sectors, then the all-pass filter (`#0`) is used at all ranges.

The low and high angle limits in each filter slot are inclusive; hence, the pair (`0x000`, `0xFFFF`) would span the full 360° circle with no gaps.

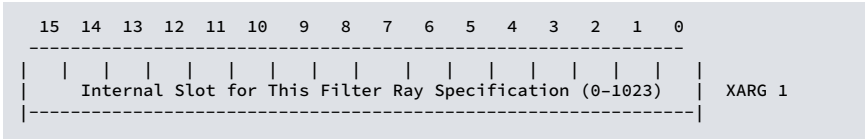
Also, the filter array can be sparse (not all slots filled in), and have overlapping sectors (in which case the highest numbered slot that spans a ray's `AZ`/`EL` midpoint is used). Choosing the highest numbered encompassing slot is a subtle but important detail that allows complex regions to be defined as a layered hierarchy of overlapping sectors. For example, if:

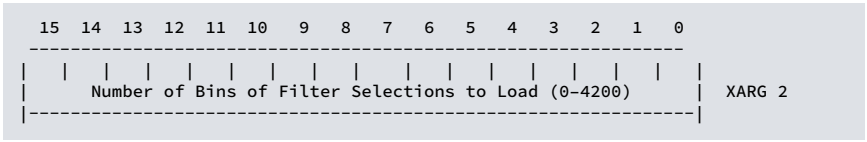
- `Slot-0` defines a default 360° filter-versus-range table.
- `Slot-1` defines special filtering within 0 ... 90° that is further modified by ° filters in a 40 ... 50° span.

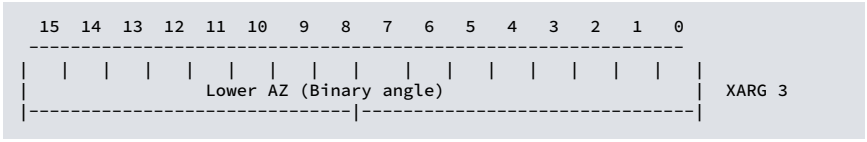
Then a 45° ray would then be filtered according to `Slot-2`, a 60° ray would use `Slot-1`, and a 100° ray would use `Slot-0`.

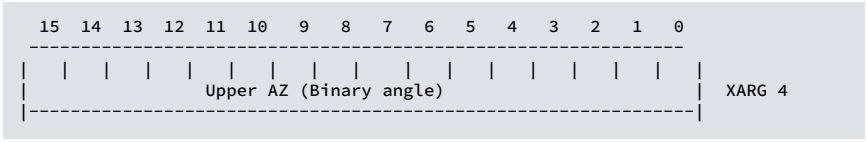
If the `CLR` bit is set in the opcode, then no additional arguments follow and the entire internal filter array (all slots) is invalidated. The result is that no clutter filter is applied to any of the processed data, regardless of range, `AZ`, or `EL`. Moreover, loading a given slot with a table consisting of 0 bins of filter data invalidates just that slot. This allows some data to be removed from the table without having to resort to a complete `CLR` operation.

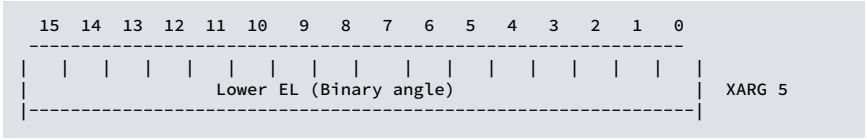
The legacy **LFILT** command is equivalent to calling the enhanced command first with the `CLR` bit set, followed by a second call that writes the legacy filter choices into slot `#0` using `AZ`/`EL` limits that cover all of space. Thus, the legacy behavior is obtained as a special case of the enhanced mechanism.

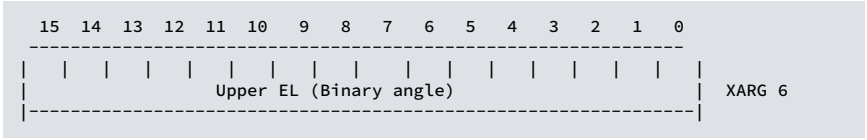


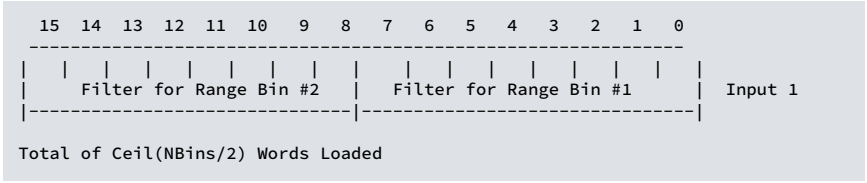












C.10 Get processor parameters (**GPARM**)

Use **GPARM** to access status information from the RVP10 processor.

64 words are always transferred. Some words are reserved for future compatibility and are read as zeros.

Table 109 RVP10 status output words

Word	Description
1	Revision/Serial number
2	Number of Range Bins
3	Current trigger period
4	Current `TAG00` - `TAG15`
5	Current `TAG16` - `TAG31`
6	Log of Measured Noise Level
7	`I` Channel DC Offset
8	`Q` Channel DC Offset
9	Latched Processor Status
10	Immediate Status Word #1
11	Diagnostic Register A
12	Diagnostic Register B
13	Number of Pulses/Ray
14	Trigger Count (Low 16-bits)
15	Trigger Count (High 8-bits)
16	No. of Properly Acquired Bins
17	No. of Properly Processed Bins
18	Immediate Status Word #2
19	Noise Range in Km
20	Noise Trigger Period
21	Pulse Width 0 min. Trig. Period
22	Pulse Width 1 min. Trig. Period
23	Pulse Width 2 min. Trig. Period
24	Pulse Width 3 min. Trig. Period
25	Pulse Width Bit Patterns
26	Current/Pulse Width
27	Current Trigger Gen. Period

Word	Description
28	Desired Trigger Gen. Period
29	PRT at Start of Last Ray
30	PRT at End of Last Ray
31	Processing/Threshold Flags
32	Log Slope
33	LOG Threshold
34	CCOR Threshold
35	SQI threshold
36	SIG Threshold for Width
37	Calibration Reflectivity
38	Reserved
39	Reserved
40	Range Averaging Choice
41	Reserved
42	Reserved
43	Header configuration of PROC data
44	I-Squared Noise (Low 16-bits)
45	I-Squared Noise (High 16-bits)
46	Q-Squared Noise (Low 16-bits)
47	Q-Squared Noise (High 16-bits)
48	Log of Measured Noise Level
49	LOG Noise Standard Deviation
50	Horizontal/Vertical Noise Ratio
51	AFC/MFC Control Value
52	Interference Filter Select
53	Interference Filter C1 Constant
54	Interference Filter C2 Constant
55	Immediate Status Word #3
56	Burst Tracking Slew
57	Polarization Algorithm Choices
58	Range Mask Spacing

Word	Description
59	Immediate Status Word #4
60	Reserved
61	Reserved
62	Reserved
63	Reserved
64	Reserved

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                                       | 0   1   0   0   1 |  Command
|---------------------------------------|-------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Rev Bits 3-0  |   Rev 6-4 |           Serial Number           |  Output 1
|---------------|-----------|-----------------------------------|
```

Shows the revision and serial numbers of the RVP10 board. This information is useful when computer software is designed to handle many signal processor revisions.

The revision number is 7 bits total, 4 of which are in the high four bits of the word for compatibility with an older format.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|        Number of Bins Currently Selected in Range Mask        |  Output 2
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   TRIGIN Current Trigger Period in 1/8km (0.83333 usec) Steps  |  Output 3
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                Current Sample of TAG bits 15-0                |  Output 4
|-------------------------------|-------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|             Current Sample of TAG bits 31-16                  |  Output 5
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0   0 | (MSB)        Log of Measured Noise Level(LSB)          |  Output 6
|-------|-----------------------------------------------------------|
```

This value is scaled four times higher than the time series LOG format (see Initiate processing (**PROC**) (page 383)). To convert to dBm, use the equation:

$$dBm = P_{MAX} + Slope \times \left[\left(\frac{Value}{4}\right) - 3584\right]$$

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|           "I" Channel Measured DC Offset                      |  Output 7
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|             "Q" Channel Measured DC Offset                    |  Output 8
|-------------------------------|-------------------------------|
```

These two words convey the measured **I** and **Q** DC offsets from the last noise sample. The output format is either signed 16-bit values in which ±32767 represent ±1.0 (legacy format), or packed time series values using the High-SNR encoding format. **Bit-9** of **GPARM** Word-59 shows which format to use.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|     Latched Status Word (Bits Cleared After Each Access)      |  Output 9
|---------------------------------------------------------------|
```

Bit 0
No Trigger during noise measurement

Bit 1
Trigger too fast during noise measurement, that is, some of the noise sample bins were positioned past the trigger range

Bit 2
No trigger during **PROC** command

Bit 3
PRT varied by more than 10 µsec within portions of a processing interval that should have been at a fixed rate.

Bit 4
Error in polarization control and/or polarization status readback

Bit 5
FIFO overflow during last **PROC** command

Bit 6
Command received while waiting for output FIFO space The command was processed, but some output data has been lost (zeroed)

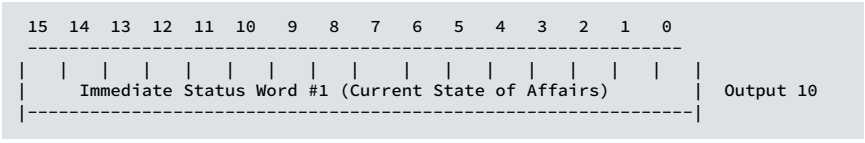
Bit 7
Error detected during last **SNOISE** command

Bit 9
Error in last Load Range Mask (**LRMSK**) command. This generally means that too many range bins were selected.

Bit 10
Error in **LSIMUL** command protocol

Bit 11
Measured phase sequence is incorrect

Bit 15
Invalid processor configuration. This bit is set if the last **PROC** command called for an illegal combination of parameters. The possible causes are:
- Spectrum size greater than 128 or less than 4
- More than 342 bins/slave in FFT modes
- (bins/slave) x (4 + sample size) exceeds 26200 in FFT modes
- (bins/slave) x (sample size) exceeds 3000 for Time Series or Spectra output
- Odd number of bins selected during fast polarization switching
- Bad combination of polarization parameters



Bit 0
No trigger, or, more than 50 ms since last trigger.

Bit 1
Error in loading trigger angle table. See Load antenna synchronization table (**LSYNC**) (page 423).

Bit 2
PWINFO command is disabled.

Bit 3
Angle sync input is BCD (Else binary angle)

Bit 4
Angle sync is on elevation axis (Else azimuth axis)

Bit 5
Angle sync is enabled

Bit 6
Angle sync allows short output rays

Bit 7
Angle sync is dynamic (else rays begin on sync angles).

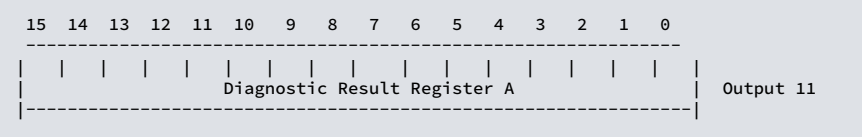
Bit 8
DSP has full IAGC hardware and firmware configuration

Bit 9
DSP supports 16-bit floating time series

Bit 11, 10
Current unfolding mode

Bit 13, 12
Number of RVP10/PROC compute processes minus one

Bit 14
DSP supports Power Spectrum output



Bit 0
Error loading config/setup files

Bit 1
Error attaching to antenna library

Bit 2
Problem when forking compute processes

Bit 3
Signals raised during startup

Bit 4
RVP running without **root** privileges

Bit 5
Problem creating daemon process

Bit 6
Inconsistent setup values detected

Bit 7
Ethernet MTU does not support requested frame size

Bit 8
Processor is running in Test/Debug mode

Bit 9

Insufficient kernel buffering for incoming UDP packets

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                 Diagnostic Result Register B                  |  Output 12
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|              Number of Pulses Being Integrated                |  Output 13
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                Trigger Count (Low 16-bits)                    |  Output 14
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                Trigger Count (high 8-bits)                    |  Output 15
|---------------------------------------------------------------|
```

The trigger count is a running tally of the number of triggers received by the RVP10 on the **TRIGIN** line. It is a full 24-bit counter.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Number of Properly Acquired Bins for Current Range Mask & PRT |  Output 16
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    No. of Valid Bins in Initial Part of Ray From Last PROC Cmd |  Output 17
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Immediate Status Word #2 (Current State of Affairs)        |  Output 18
|---------------------------------------------------------------|
```

Bit 0

Processor supports FFT algorithms

Bit 1
Processor supports Random Phase algorithms

Bit 2
Reserved (zero)

Bit 3
Processor supports DPRT-1 (dual-PRT) algorithms

On dual IFDR systems: Bits **4**, **5**, **7**, and **11** are set if either IFDR fails:

Bit 4
Unused

Bit 5
Unused

Bit 7
IFDR PLL is not locked to external user-supplied clock reference

Bits 8–10
Status of burst pulse and AFC feedback
- 1: AFC Disabled
- 2: Manual Frequency Control
- 3: No burst pulse detected
- 4: AFC is waiting for warm-up
- 5: AFC is locked
- 6: AFC is tracking

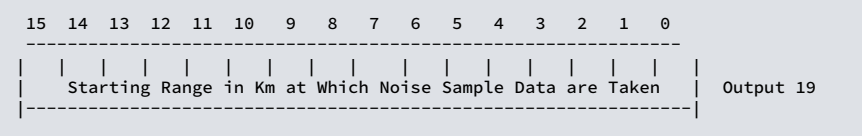
Bit 11
IFDR test switches are not in their normal operating position

Bit 12
Set according to whether the RVP10 is performing trigger blanking. This allows the host computer to decide whether to interpret the **End-TAG-0** bit in the output ray header as a blanking flag, or as a normal **TAG** line.

Bit 13
Missing signal at IFDR #**1 Burst Input**

Bit 14
Reserved (zero)

Bit 15
Set when valid burst power is detected, but the center-of-mass lies outside of the aperture sub-window that defines the portion of the pulse used for AFC analysis. This error bit flags when the burst pulse has drifted out of its optimal placement within the sampling window.



```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Trigger Period (0.16667usec Increments) During Noise Sampling  |  Output 20
|----------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Min Trig Period (0.16667usec Increments) for Pulse Width 0   |  Output 21
|----------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Min Trig Period (0.16667usec Increments) for Pulse Width 1   |  Output 22
|----------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Min Trig Period (0.16667usec Increments) for Pulse Width 2   |  Output 23
|----------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|  Min Trig Period (0.16667usec Increments) for Pulse Width 3    |  Output 24
|----------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Four 4-bit Control Bit Patterns for Each Pulse Width        |  Output 25
|----------------------------------------------------------------|
```

For definitions of these bits, see **`input word #1`** in Define pulse width control and PRT Limits (**PWINFO**) (page 420) .

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|  PWidthS      |Hyb|  Window   |     TopMode     |    PWidth     |  Output 26
|---------------|---|-----------|-----------------|---------------|
```

`PWidth`
Currently selected radar pulse width

`TopMode`
Major Mode. See **`Input #9`** in Setup operating parameters (**SOPRM**) (page 368).

`Window`
: Spectral Window Choice. See `Input #10` in Setup operating parameters (**SOPRM**) (page 368).

`PWidthS`
: Pulse width of second pulse in hybrid transmit waveform

`Hyb`
: Bit indicating second pulse in use in hybrid transmit waveform

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|  Current Trigger Generator Period (0.16667usec Increments)     |  Output 27
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Desired Trigger Generator Period (0.16667usec Increments)   |  Output 28
|---------------------------------------------------------------|
```

The desired trigger generator rate is that which was selected in the most recently issued **SETPWF** command (or power-up rate if **SETPWF** was not issued). The current rate may be different from the desired rate due to bounding against limits for the current pulse width, or being in an odd ray cycle during dual-PRT processing. The measured PRTs are forced to `0xFFFF` (the maximum unsigned value) when the external trigger is expected, but missing.

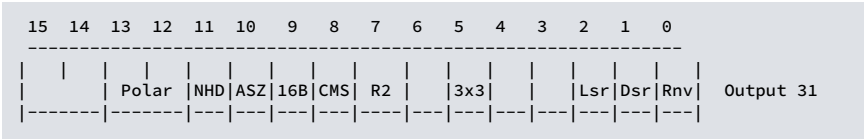
```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|      TRIGIN Period at Start of Last PROC Acquisition Time      |  Output 29
|---------------------------------------------------------------|
```

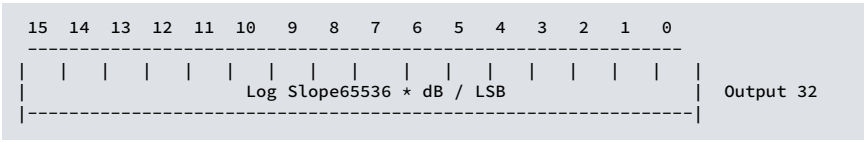
```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|       TRIGIN Period at End of Last PROC Acquisition Time       |  Output 30
|---------------------------------------------------------------|
```

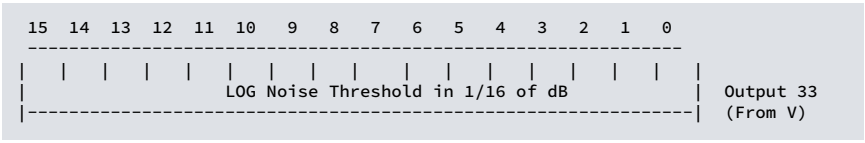
The PRTs from the start and end of the last ray are the measured values when possible, that is, when non-simulated data are being processed, and we either have an external trigger, or an internal trigger that is not in any of the dual-PRT modes. The units are the same as for the measured current trigger period in `Output #3`.

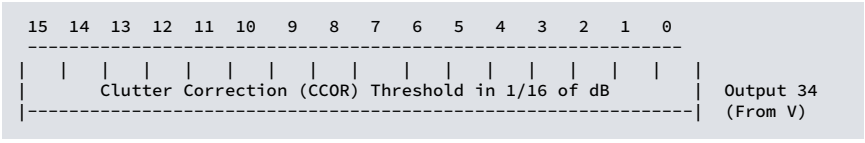
`Output 31 ... Output 37` are the current processing and threshold parameters set by **SOPRM**. See Setup operating parameters (**SOPRM**) (page 368).

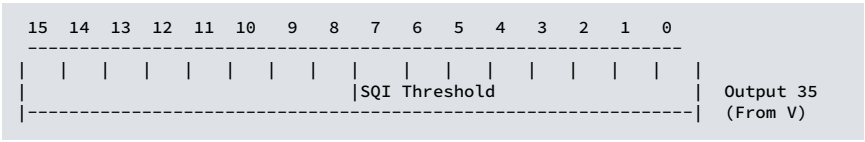
Since the threshold levels for each data parameter can be different (see Set Individual Thresholds (**THRESH**) (page 440)), words `33 ... 36` are taken from the velocity parameter.

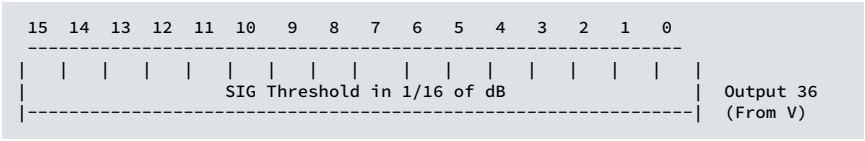

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Polar
NHD
ASZ
16B
CMS
R2
3x3
Lsr
Dsr
Rnv
Output 31

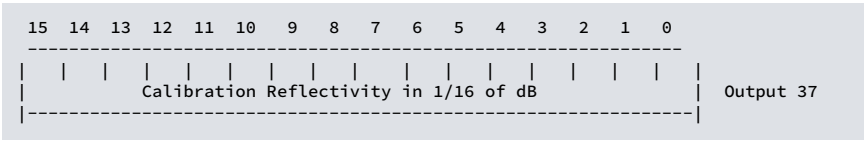

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Log Slope65536 * dB / LSB
Output 32

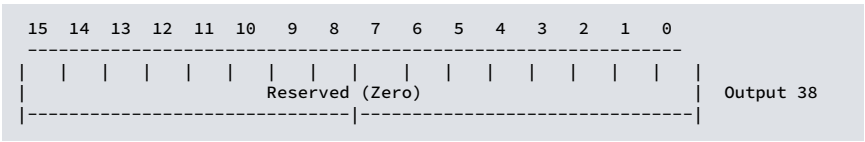

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
LOG Noise Threshold in 1/16 of dB
Output 33
(From V)

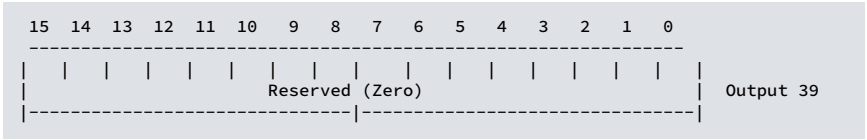

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Clutter Correction (CCOR) Threshold in 1/16 of dB
Output 34
(From V)

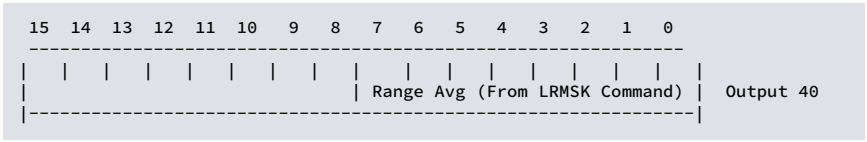

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
SQI Threshold
Output 35
(From V)

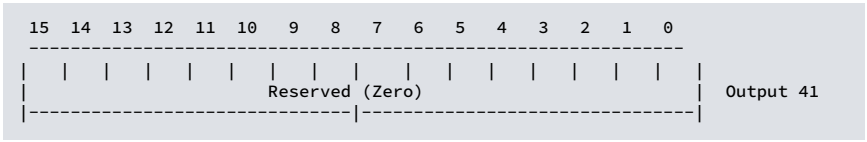

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
SIG Threshold in 1/16 of dB
Output 36
(From V)

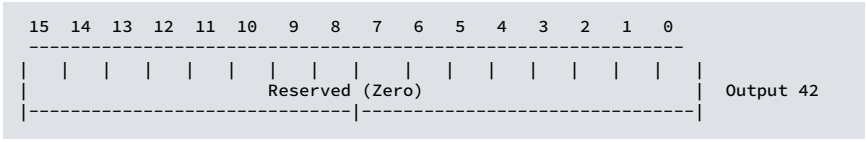

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Calibration Reflectivity in 1/16 of dB
Output 37


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved (Zero)
Output 38

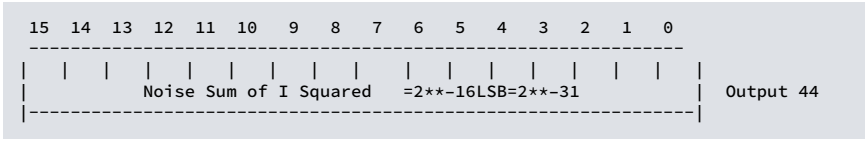

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved (Zero)
Output 39

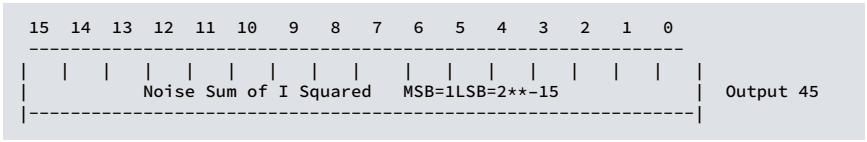

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Range Avg (From LRMSK Command)
Output 40

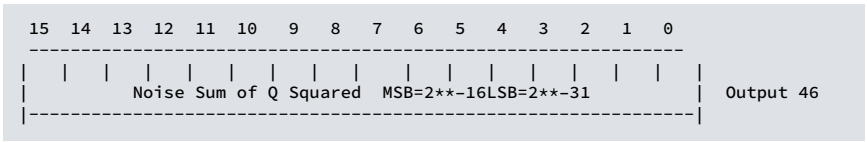

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved (Zero)
Output 41


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved (Zero)
Output 42


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Header Config of PROC data (CFGHDR Input #1)
Output 43


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Noise Sum of I Squared =2**-16LSB=2**-31
Output 44


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Noise Sum of I Squared MSB=1LSB=2**-15
Output 45


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Noise Sum of Q Squared MSB=2**-16LSB=2**-31
Output 46

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|          Noise Sum of Q Squared  MSB=1LSB=2**-15              |  Output 47
|---------------------------------------------------------------|
```

To compute the noise power in dBm from Words **44** ... 47, first calculate:

$N_I = (Word\,45) \times 2^{-15} + (Word\ 44) \times 2^{-31}$

$N_Q = (Word\,47) \times 2^{-15} + (Word\ 46) \times 2^{-31}$

from which we obtain:

$dBm = P_{MAX} + 10\log_{10}\left(N_I + N_Q\right) - 3dB$

The four integer values become rather small and severely quantized when the noise power drops to low values. Previously, these words helped balance the individual gain of the `I` and `Q` channels in RVP6 in the presence of a strong test signal. Since `I` and `Q` are inherently balanced in the RVP10, these output words are no longer of much value.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|        Log of Measured Noise Level (same as word 6)           |  Output 48
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Vert Noise Stdev in dB*10  |  Horiz Noise Stdev in dB*10    |  Output 49
|---------------------------------------------------------------|
```

The noise standard deviations for each receive channel are normalized to the mean power. The values reported here hover around 0 dB for ordinary exponentially distributed noise in which the standard deviation scales directly with the mean.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Ratio of Horizontal/Vertical Noise Power in Hundredths of dB   |  Output 50
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|           16-Bit AFC/MFC Value (-32768 through +32767)         |  Output 51
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| PhaseSeq      | IFD Sat.Power |    MinRev     |    Inter.F    |  Output 52
|---------------------------------|---------------|---------------|
```

Inter.F
Specifies which interference filter is running. Zero means "none". For information on interference filter algorithms, see Interference filter (page 161).

MinRev
Minor revision level of the RVP10 code that is currently running

IFDR Sat.Power (PMAX)
Input power required to saturate the IF-Input A/D converter for the IFDR that is currently attached

- 0: +4.5 dBm
- 1: +6.0 dBm
- 2: +8.0 dBm

PhaseSeq
Tx Phase modulation sequence. See Configure phase modulation (**CFGPHZ**) (page 438).

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Interference Filter Parameter “C1” in Hundredths of dB     |  Output 53
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Interference Filter Parameter “C2” in Hundredths of dB     |  Output 54
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|       Immediate Status Word #3 (Current State of Affairs)     |  Output 55
|---------------------------------|---------------|---------------|
```

Bit 0
Burst pulse timing adjustments can be made

Bit 1
Burst pulse frequency adjustments can be made

Bit 2
Burst pulse hunting is enabled

Bit 3
Burst pulse hunt is running right now

Bit 4
Last burst pulse hunt was unsuccessful

Bit 5
Processor supports DPRT-2 (dual-PRT) algorithms

Bit 6
Could not generate the requested phase sequence

Bit 7
Unused

Bits 8–11
User-defined Major Modes 1 ... 4 are supported

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|         Signed trigger slew in hundredths of microseconds      |  Output 56
|---------------------------------------------------------------|
```

This is the same format that is used by the **SETSLEW** command to set the current trigger slew. See Set Trigger Timing Slew (**SETSLEW**) (page 437).

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               Polarization Algorithm Choices                  |  Output 57
|---------------------------------------------------------------|
```

Bit 0
Use H transmissions for (**T**, **Z**, **V**, **W**)

Bit 1
Use V transmissions for (**T**, **Z**, **V**, **W**)

Bit 2
Use Co-Pol reception for (**T**, **Z**, **V**, **W**)

Bit 3
Use Cross-Pol reception for (**T**, **Z**, **V**, **W**)

Bit 4
Correct all polar Parameters for noise

Bit 5
Use filtered data for all polar parameters

Bit 6
Sign convention for PhiDP

Bit 7
Z and Z_{dr} are corrected for attenuation using PhiDP

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|         Range mask spacing for current pulsewidth (cm)         |  Output 58
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Immediate Status Word #4 (Current State of Affairs)        |  Output 59
|---------------------------------------------------------------|
```

Bit 0
Internal power spectra size matches sample size (else power-of-2)

Bit 1
PROC command output spectra match sample size (else power-of-2)

Bit 2
Trigger pattern has been altered to fit within the desired PRT

Bit 3
PRT has been altered to preserve the desired trigger pattern

Bit 4
Using High-SNR packed (I,Q) format

Bit 5
Trigger sequence truncated due to insufficient pattern memory

Bit 6
Time series data source is external to RVP10

Bit 7
WSR88D Batch mode is supported

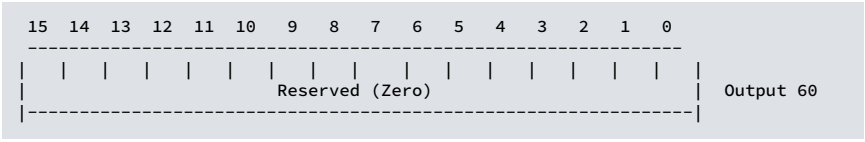
Bit 8
Major mode refuses to use external trigger

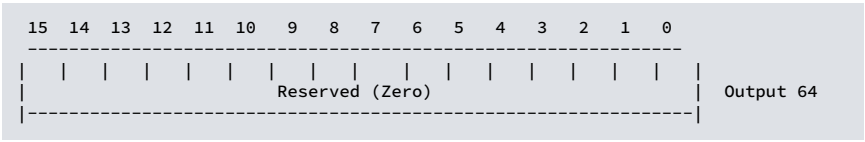
Bit 9
GPARM outputs #7 and #8 use Hi-SNR format, else linear

Bit 10
Receiver protection fault

Bit 11
IFDR dual-channel inconsistency (for example, power and/or phase out of bounds for ratio of HiGain-to-LoGain channels)

Bit 12
GPS 1-pulse-per-second input clock error





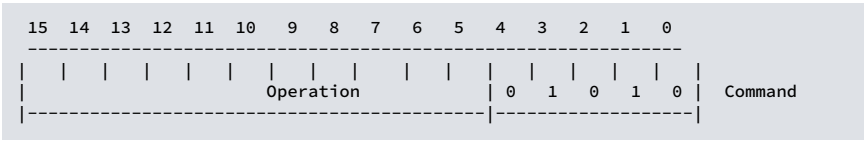
C.11 Load simulated time series data (**LSIMUL**)

LSIMUL acts as a diagnostic for proper functioning of the RVP10 algorithms. It permits arbitrary simulated data samples to be input to the processing routines, rather than sampled data from the A/D converters as is ordinarily the case. Since the properties of the simulated data are known exactly, it is possible to verify that the calculations within the RVP10 are proceeding correctly.

The **LSIMUL** command (with `Operation=1`) should be issued prior to the **PROC** command which is being tested. This enables the simulated data mode. The next **PROC** command waits for `N` (`N` = sample size) **LSIMUL** commands (with `Operation=2`) prior to outputting each ray. The arrival of any other command during that time causes the simulated data mode to be exited, and error bit `#10` is set in the **GPARM** latched status word. The error bit is also set if an **LSIMUL** command with `Operation=2` is received while simulated data mode is disabled.

You may specify a single simulated data sample for every range bin, or a pattern or simulated samples to be replicated over the range of bins. Most RVP10 algorithms are independent of range, and can be tested with identical data at every bin. Notable exceptions, however, are the "pop" clutter filter, and range bin averaging procedures.

In its full generality, the **LSIMUL** command permits independent `I` and `Q` samples to be simulated at every bin of every pulse. If this results in more host computer I/O than is practical, then specify fewer simulated bins and allow the RVP10 to replicate them internally.



The available operations are:

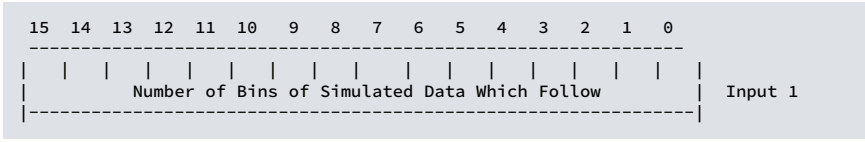
`0`
: Disable the use of simulated data. RVP10 returns to acquisition and processing of live data from the A/D converters.

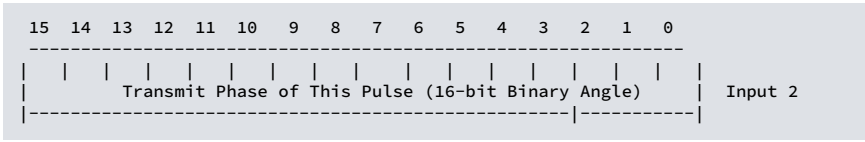
1

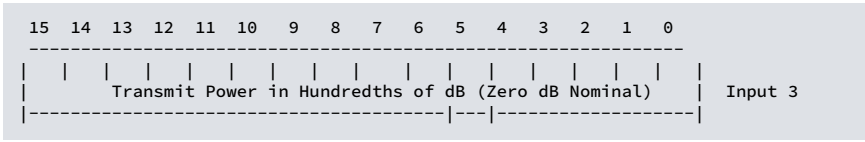
Enable processing of simulated data. Subsequent **PROC** commands use the data supplied in the next **N** (**N** = sample size) **LSIMUL** with **Operation= 2**.
The receiver noise and offset levels which are internally maintained by RVP10 are set to their special simulated values (from the **M**+ setup menu) by this command. This is because the measured offsets are not relevant to the simulated data, and must not be used in the subsequent computations. It is important to issue the **SNOISE** command before resuming the acquisition and processing of live radar data.

2 or 3

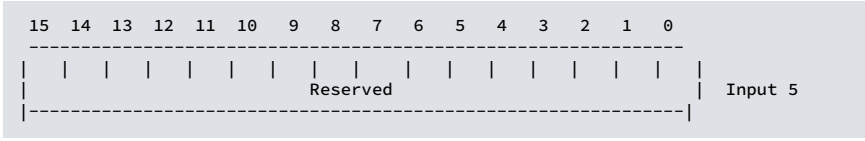
Load one pulse of data samples beginning with the following 4-word header, and continuing with an array of items each representing a single instantaneous sample of (**I**,**Q**) data. You may specify one or more bins to be loaded, and RVP10 replicates these data as necessary in order to fill out the entire count of acquired bins. If the number of bins is 0, then a zero-valued sample is applied for all channels.





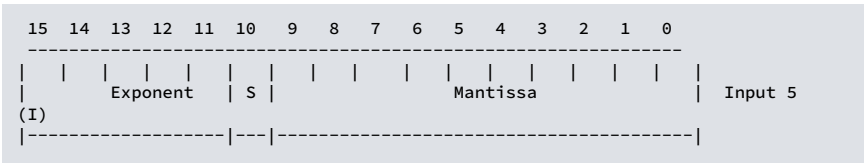


In the legacy format **#2** (RVP5-RVP10) each bin within the pulse is represented by four 16-bit fixed point words. The total number of words loaded is (**4+4B**), where **B** is the bin count specified in **Word #1**. This takes account the 4 header words, plus 4 words for every bin being defined.



In format `#3` each bin within the pulse is represented by two 16-bit floating point words having the exact same format as the packed (`I`,`Q`) time-series data that are output by the **`PROC`** command . See Initiate processing (**PROC**) (page 383).

Compared to the legacy 4-word format, this 2-word format uses half the I/O bandwidth, has superior dynamic range, and allows data to be fed back into the signal processor in their native packed format. The total number of words loaded (including the initial header section) is (`4+2B`), where `B` is the bin count specified in **`Word #1`**.



C.12 Reset (**RESET**)

The **RESET** command permits resetting either the entire RVP10 processor, or selected portions. Flags in the command word determine the action to be taken.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                     |Nv |Nse|Fif |Nv |Nv | 0   1   1   0   0 |  Command
|---------------------|---|---|----|---|---|-------------------|
```

`Nv`
: Reloads configuration from the saved nonvolatile settings.

`Nse`
: Reset the receiver noise levels to the power-up default value for all pulsewidths as defined in the **Mt** setup questions. See **`Mt<n>`**— Transmit sequence #n (page 95).

`Fif`
: Remove any data currently in the output FIFOs. This permits flushing output data that was left from a previous command, so that new output can be read from scratch. See First-in-first-out (FIFO) buffer (page 365).

C.13 Define trigger generator waveforms (**`TRIGWF`**)

NOTICE! Do not use **`TRIGWF`** in any new code applications that drive RVP10. Use the interactive trigger setup procedure to define RVP10 triggers and timing. See Burst pulse timing plot (Pb) (page 113).

`TRIGWF` is obsolete. It is included for backward compatibility with RVP6. The code is disabled by default.

RVP10 has a built-in trigger generator that can synthesize 6 independent digital output waveforms, each having arbitrary shape and being active anywhere in a window centered around zero-range.

The trigger outputs can be defined by a 2048-word by 6-bit table which is loaded from the user computer. The patterns are automatically read from the table and output to the 6 trigger lines during each radar pulse.

The outputs can be used for transmitter triggers, scope triggers, range strobes, PLL gates, and so on. The writable waveform table is unique, in that the detailed timing of trigger and related control signals can be easily adjusted in software, without having to resort to reprogramming **`PROM`**s. This makes it possible for user software to edit the trigger timing interactively.

Trigger waveforms are loaded using the **`TRIGWF`** command. Four bits in the command word (**`PW0 ... PW3`**) select which pulse widths receive the new waveforms. On power-up, the pulse widths are initialized to user-selected waveforms.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               |PW3|PW2|PW1|PW0|           | 0   1   1   0   1 |  Command
|---------------|---|---|---|---|-----------|-------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| H |               | Transition Point of POLAR0 Control Line   |  Input 1
|---|---------------|-------------------------------------------|
```

The **H** bit defines the sense of the control line when horizontal polarization is selected.

Inputs **2 ... 2049**, indicate bits output every 7.195 MHz and **Input 1025** corresponds to the time at which data at range zero are sampled.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                                       | Bits for -142.22 usec |  Input 2
|---------------------------------------|-----------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                                       |  Bits for zero range  |  Input 1025
|---------------------------------------|-----------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                                       | Bits for +142.08 usec |  Input 2049
|---------------------------------------|-----------------------|
```

C.14 Define pulse width control and PRT Limits (**PWINFO**)

RVP10 can control the radar transmitter's pulse width and corresponding receiver bandwidth.

There are 16 pulse/bandwidth codes, numbered **0** ... **15**. The association between codes and pulse widths is determined by the needs and capabilities of each radar. In some cases, code **0** can represent 0.25 microsecond pulse width, and in other cases it can represent 2.0 microseconds, and some radars may use all sixteen codes while others provide fewer options.

The **PWINFO** command defines what happens for each pulse width code. **SETPWF** defines selects which code is used.

The **PWINFO** command loads four codes at a time according to the `UpperPW` bits: `00` loads codes `0` … `3`, `01` loads codes `4` … `7`, and so on.

RVP10 drives four TTL output lines (**PWBW0 – 3**) which control the radar pulse/bandwidth hardware. Typically this control is through relays or solid-state switches in the transmitter and receiver. The user decides what state the four lines assume for each pulse width code using word #1 following the command, which contains four codes packed into one 16-bit word. The power-up default is to drive output line **N** low for a code of **N**, keeping all other lines high (Input of `7BDE Hex`). The flexibility in defining the output bits usually makes the radar hardware connections very simple. For example, if pulse width selection relied on choosing one of four relays, then each **PWBWn** line could serve directly as a relay driver using the default pattern.

For each pulse width there is a corresponding minimum trigger PRT permitted. This bound limits the transmitter duty cycle to a safe value under all conditions. **PWINFO** sets up these minimum PRTs using words `2` … `5` following the command. The maximum frequency of the internal trigger generator is then constrained at each pulse width to the indicated rate. This protection applies at all times, that is, during noise sampling, during ray processing, and during the standby time between rays. The default PRT bounds are 2000, 1000, 750, and 500 Hertz (Inputs of 3000, 6000, 8000, and 12000). If your radar does not use all of the pulse width codes, it is still a good idea to set the unused PRT limits to reasonable values. This way protection is still provided if that **SETPWF** accidentally selects one of the unused states. If the internal trigger generator is not being used, then the PRT limits no longer affect the trigger rate and transmitter protection becomes the responsibility of the user hardware.

You can turn off the pulse/bandwidth mechanism by setting the bit patterns and PRT limits all to the same value.

The **PWINFO** command can be disabled (for transmitter safety), so that PRT limits cannot accidentally be changed by the host computer. When this is done, RVP10 still reads the five input words, but no changes are made to the pulse width and PRT information. The command I/O behaves the same way, whether enabled or disabled.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                      |UpperPW|           | 0   1   1   1   1 |  Command
|----------------------|---|---|-----------|-------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Bits for PW 3 | Bits for PW 2 | Bits for PW 1  | Bits for PW 0 |  Input 1
|---------------|---------------|----------------|---------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Min Trig Period (0.16667usec Increments) for Pulse Width 0   |  Input 2
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Min Trig Period (0.16667usec Increments) for Pulse Width 1   |  Input 3
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Min Trig Period (0.16667usec Increments) for Pulse Width 2   |  Input 4
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   Min Trig Period (0.16667usec Increments) for Pulse Width 3   |  Input 5
|---------------------------------------------------------------|
```

C.15 Set pulse width and PRF (**SETPWF**)

SETPWF selects the pulse width and trigger rate.

A 4-bit pulse width code is passed in bits (**13**, **12**, **9**, **8**) of the command word, and selects one of 16 pulse widths as described under **PWINFO**. The new radar PRT is passed in **word #1**. For all processing modes that use a fixed trigger rate, this value defines the trigger period that is output at all times except during noise measurements. For Dual-PRF applications, this word defines the short period (high PRF) rate. The long period is internally computed as either 3/2, 4/3, or 5/4 the short period, and the trigger generator alternates between the short and long rates on each successive ray.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|       |UpperPW| (Rsv) |LowerPW|           | 1   0   0   0   0 |  Command
|---------------|-------|---|---|-----------|-------------------|
```

`UpperPW`
: Upper two bits of overall 4-bit pulse width selection

`LowerPW`
: Lower two bits of overall 4-bit pulse width selection

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|    Desired Trigger Generator Period (0.16667usec Increments)   |  Input 1
|---------------------------------------------------------------|
```

When **Input #1** is zero, the arguments take on an alternate form that allows an array of **N** (up to 64) trigger periods to be specified, and also gives much finer time resolution in the choice of each period. The **XARGS** command is first used to load an array of **N** 32-bit words that define the trigger period(s) in nanoseconds. RVP10 generates triggers with shapes (relative starts and widths) are identical for each pulse, but whose periods follow the selected sequence. Trigger patterns such as these are intended to support research customers who use the real-time (**I**,**Q**) data stream directly.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|       Lower 16-Bits of 32-Bit Trigger Period in Nanoseconds    |  XARG 1
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|       Upper 16-Bits of 32-Bit Trigger Period in Nanoseconds    |  XARG 2
|---------------------------------------------------------------|
```

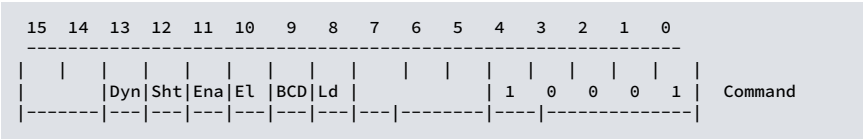
C.16 Load antenna synchronization table (**LSYNC**)

RVP10 can operate in a mode where radar data are acquired in synchronization with the antenna motion along the azimuth or elevation axis. This means that the user computer does not need to separately monitor the antenna angles and request each data ray individually.

To use this mode, **TAG0–15** must be wired to receive azimuth angles and **TAG15–31** must be wired to receive elevation. Angle input may be 16-bit binary angles or 4-digit BCD. This synchronization mode is the only one that ascribes meaning to TAG inputs (usually they are only passed on to the user computer as ancillary information).

Antenna synchronization is done using a table of trigger angles. This table, which contains 3 … 4096 angles, defines the angle boundaries for each processed ray. The trigger angles do not need to be uniformly spaced nor must they span the full 360° rotation. This gives flexibility in the choice of angles. For example, if local obstructions cause shadows in the radar image, then those regions can be skipped by omitting table entries in their vicinity. Likewise, as the antenna rotates, data can be acquired within one or more sectors by specifying the appropriate sets of contiguous bearings at the desired angular resolution. On power-up, the angle table is initialized to 360 values corresponding to half-integer-valued degrees from 0.5 … 359.5°.

The synchronization algorithm works automatically with clockwise or counterclockwise antenna rotation, and can tolerate any sequence of changes in direction, for example, if the antenna scans a sector, or turns erratically. The trigger angles do not have to be hit exactly to start each new ray – the antenna only needs to move across them. This minimizes the possibility of losing data due to missing codes in the angle encoders. RVP10 automatically produces an output ray after one second of waiting, even if no trigger angles have been crossed. This prevents time-outs with the host computer when the antenna is not moving.



`Dyn`
: If set, the endpoints of each ray are synchronized. Their angular width can vary by what is required to collect the **SOPRM** number of pulses.
If clear, the angular width of each ray is fixed (between successive table entries) by adjusting the pulse count and reinterpreting the **SOPRM** sample size as a maximum value.

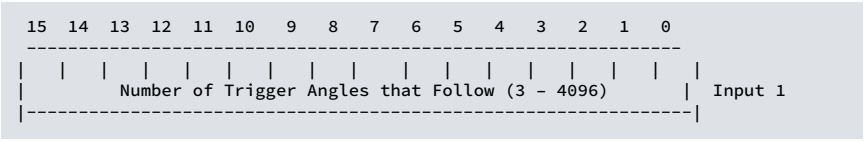
`Sht`
: Synchronized rays ordinarily are the full width of successive table entries (in the static case) or the full requested pulse count (dynamic case). This bit modifies the behavior in both modes to allow short rays to be produced, where the pulse count is less than expected due to encountering a feature in the CPI (usually a trigger transition) that would normally result in the entire ray being discarded.
When this bit is set, the user's code must check the pulse count that went into each ray and manually discard those that are too short to contain useful data.

`Ena`
: Enables antenna synchronization.

`El`
: Synchronization is based on `TAG1531` (Elevation) inputs. Otherwise, `TAG015` (Azimuth) is used.

`BCD`
: Specifies that TAG angle input is in the form of 4-digit Binary Coded Decimal. Otherwise, a 16-bit binary angle is assumed.

`Ld`
: Indicates that a new table size and array of values follow the command.
If `Ld = 0`, then **LSYNC** is a one-word command only. Otherwise, the following words are used to load the new table:



```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                          (Ignored)                            |  Input 2
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|          First Trigger Angle (16-Bit Binary Angle)            |  Input 3
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|          Last Trigger Angle (16-Bit Binary Angle)             |  Input 3
|---------------------------------------------------------------|
```

You must configure the system to use the synchronization mode.

1. Use the **LSYNC** command to load the trigger angle table.
   a. Choose the number of table entries.
   b. Write the required number of words to RVP10.

      Supply the angles clockwise in a strictly increasing order. They must neither reach nor pass 0° by the table's end. The first value may be 0.
      Use binary angle representation, where Bit 15 represents 180°, Bit 14 represents 90°, and so on.

   c. Set the **Ld** bit command word to indicate that a new table size and set of angles are being loaded.
   d. Set a flag bit to be set if errors are detected when loading the table of angles.

   See Get processor parameters (**GPARM**) (page 399).

2. Enable synchronized operation by setting the **Ena** command bit. Set or clear, `EL` and `BCD` according to your needs.

   These bits may be used independently of reloading the table values. Thus, antenna synchronization may be turned on and off without having to reload the table each time. If there are errors when the table was last loaded, the processor ignores the `Ena` bit and synchronization is forced off.
   Once enabled, **PROC** commands are issued in the usual manner to acquire and process the radar data. Either the single-cycle or free-run **PROC** mode may be used. Data collection proceeds as usual, except that the rays are automatically aligned with the trigger angles. The angle sync algorithm is dynamic and works as follows:
   a. Each ray begins immediately upon the user's request, or upon completion of the previous ray when in continuous processing mode.
   b. At the start of the ray, RVP10 finds the pair of sync angles that enclose the previous trigger angle.
   c. The current ray then runs until the antenna passes outside of either limit, at which point processing for that ray is terminated.
   d. Once this happens, a new trigger angle is assigned based on which limit was crossed.
3. In the `Sample Size` field of the **SOPRM** command, specify the maximum number of pulses present in each ray during angle syncing .

   This is the number of pulses that used when `Dyn=1`.
   When `Dyn=0`, the number of pulses may be less if a trigger angle is crossed before the full pulse count can be accumulated

C.17 Set or Clear User LED (**SLED**)

SLED turns the red user LED on and off under program control. The LED is on during the initial running of internal diagnostics, and then remains off unless changed by this command.

Note that the red LED can be configured to serve as an internal activity indicator (see chapter *TTY Non-volatile Setups*), in which case this command has no effect.

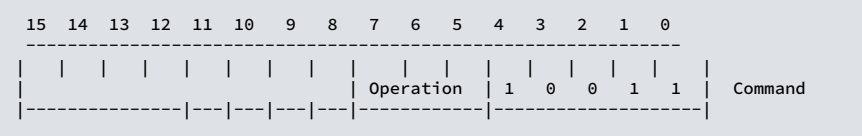
```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                          |LED|           | 1   0   0   1  0  |  Command
|--------------------------|---|-----------|-------------------|
```

C.18 TTY operation (**TTYOP**)

TTYOP controls the TTY chat mode interface to the host computer. The command can simulate the typing of characters on the RVP10 setup TTY.

Characters entered in this manner are indistinguishable from those typed on the TTY. Whatever you can do in the TTY, you can also do with this command.

RVP10 sends all TTY output to whichever stream (TTY, or host computer) provided the most recent input character. This command is also used to monitor the graphical data from the special scope plotting modes.



The operation codes are as follows:

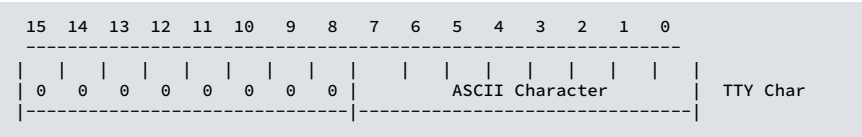
`0`
: Sends the ASCII character in the upper byte of the word to the RVP10 as if it had been typed on the setup TTY keyboard.

`1`
: Allow scope plotting data to be output when a plot is being drawn. All relevant status and data words are output once upon each receipt of this command. Subsequently, status and data is output only when a change has taken place.

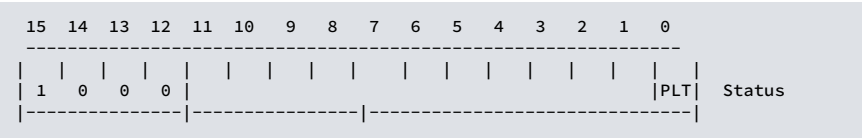
`2`
: Disable the scope plotting output data.

Any of the following types of data may be output by RVP10 while the TTY monitor is running. The order of arrival of each data type is indeterminate, but all multi-word sequences are always be output as contiguous words.

Individual TTY characters generated by RVP10 are output in the low byte of the word, with the upper byte set to zeros.

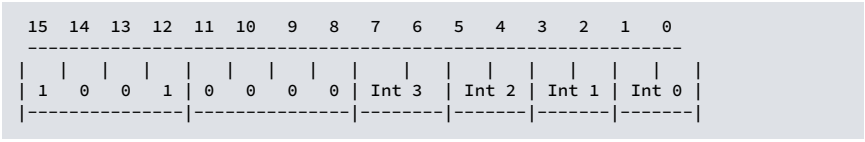


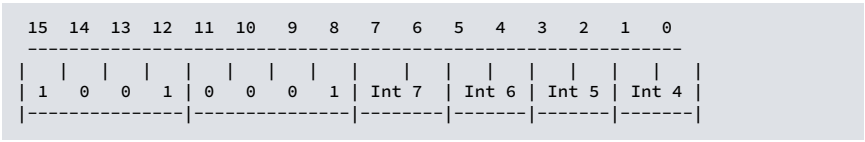
The status of the plotting modes is given in the following word.

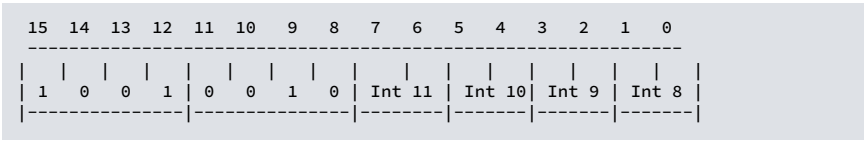


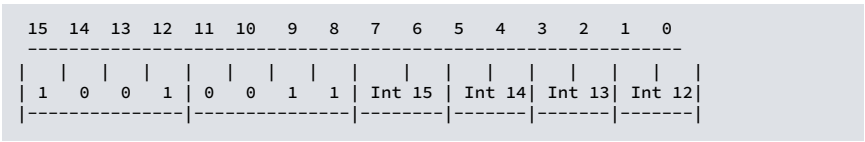
`PLT` Indicates that a scope plot is being drawn now.

The 2-bit intensities of each of 16 possible strokes of data is given in the following 4-word sequence. An intensity of `0` represents OFF; `1`, `2`, and `3` are successively brighter.

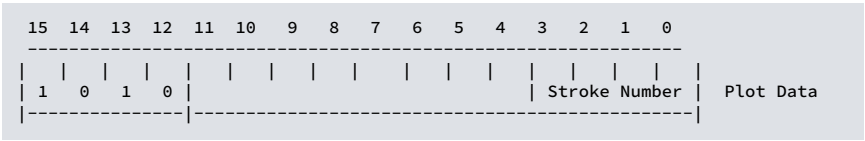


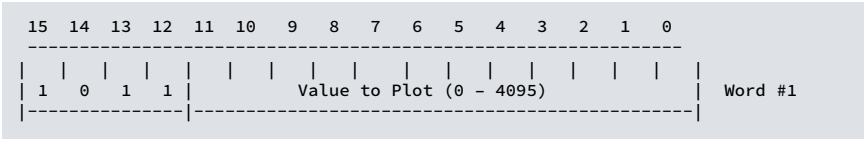


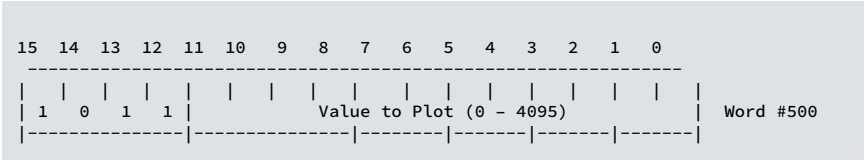




The data for each stroke of the plot is given by the following sequence of 501 words.







C.19 Load custom range normalization (**LDRNV**)

Reflectivities computed by RVP10 are usually corrected for range effects by adding an offset in decibels equal to 20 `log(R/1 km)`, where `R` is the range in kilometers. This correction is based on a simple filled beam geometry, and is sufficiently accurate for most meteorological observations.

The **LDRNV** command is for applications that require an alternate custom range correction, for example, if an external user-supplied STC waveform drives the radar receiver's LNA.

LDRNV loads a 251-word custom correction table holding values in hundredths of decibels over 5 decades of `log(range)` from 0.01km ... 1000km. There are 50 table entries per decade of range. The range in kilometers corresponding to an input word `#N` is `10[{{N-1}<devide>50}- 2]`, and the default correction table (automatically used on power-up) is `40(N - 101)`.

The table values are stored and interpolated when RVP10 loads a new range mask. Custom values for the user ranges are then computed. See Load Range Mask (**LRMSK**) (page 366).

The **LDRNV** command must be issued only once, before choosing the working set of range bins.

The linear intervening gas attenuation correction (see Setup operating parameters (**SOPRM**) (page 368)) is always added to the reflectivity data, regardless of whether default or custom range normalization is enabled. If this is undesirable, set the intervening gas slope to 0.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                                           | 1   0   1   0   1 |  Command
|-------------------------------------------|-------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|  Signed Hundredths dB of Range Normalization for Range 0.01 km |  Input 1
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|  Signed Hundredths dB of Range Normalization for Range 1000 km |  Input 251
|---------------------------------------------------------------|
```

C.20 Read back internal tables and parameters (**RBACK**)

RBACK permits some RVP10 internal tables to be read back for confirmation and diagnostic purposes. This command is not generally used during normal data acquisition and processing.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|              Data to Show                 | 1   0   1   1   0 |  Command
|---------------|---|---|---|---|-----------|-------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                Number of words to output                      |  Input 1
|-------------------|-----------------------------------------|
```

Table 110 Data returned by RBACK

Data number	Description
0	Full operational parameter table from last **SOPRM** command.
1	Ray history array consisting of six words per ray for the last 40 rays (in reverse time order) that were processed. Each six- word group holds: • Number of samples that went into the ray • Time since the last ray (in tenths of ms) • Ending azimuth TAG bits • Ending elevation TAG bits • Starting azimuth TAG bits • Starting elevation TAG bits
2	Angle sync table from last **LDSYNC** command.
3	Reserved (was **AGC** table)
4	Filter selection array from the last **LFILT** command. This returns the filter selection codes, one per word, for all range bins described by filter slot #0.
5	Reserved (was **STC** table)
6	Custom range normalization from last **LDRNV** command.
7	Samples of the TAG input lines at 4 ms intervals. The sampling begins at the moment the **RBACK** command is received, and continues until the output count is reached. Each 32-bit sample is output as a pair of 16-bit words: • Azimuth (TAG bits 0–15) • Elevation (TAG bits 16–31)
8	Doppler clutter filter coefficients (Same format as for **LFCOEFS** command)
9	Reserved (was **LOG** clutter filter coefficients)
10	Range mask spacing in cm for each pulse width
11	Current value of UIQ bits from **Set/Clr** all prior operations

Data number	Description
12	Individual threshold configuration for each data type. This allows read back of the threshold table set with the **THRESH** command. See Set Individual Thresholds (**THRESH**) (page 440). Outputs 7 words per data type. Datatypes in the order specified in the selection mask.
13	Extended parameter information defined in `struct dspExParmIO`.
14	Minimum and maximum values of the optional I/O-62 A/D converter, sampled over at least one complete pulse period. 16-bit signed outputs represent the full range of the A/D converter. When the RVP98D backpanel is connected, the first Min/Max pair samples the `LOGVideo` input, the second pair samples the `CathodePulse` input, and the third and fourth pairs sample internal levels.
15	Returns an array of `struct RVP9SpecFiltIO` structures for each of the non-zero clutter filter definitions, beginning with #1. This is the same format used by the **LFSPECS** command to define each clutter filter. The order is as defined in the `PPRMS_N_* #defines` in `rvp9.h`.
16	Returns the identifiers of the currently active HydroClass classifier algorithms. The identifiers are encoded in `sig_data_types.h`.
17	Returns the nickname of the currently active `HydroClass` configuration. The nickname can be used to label the possible customizations made to echo identification settings in the *`hydroclass-*-band.conf`* file. It is impractical to save all modified parameters as metadata. The string of 16 characters is reported in a sequence of 8 words, each reporting 2 characters.

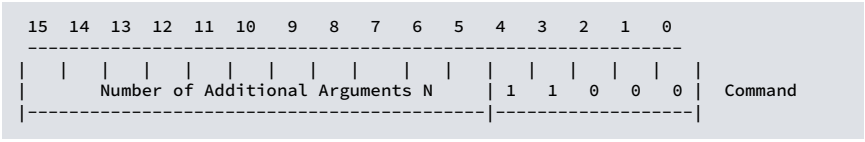
C.21 Pass auxiliary arguments to opcodes (**XARGS**)

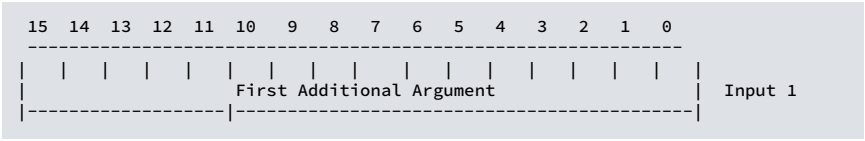
XARGS provides a backward compatible mechanism for supplying additional (optional) arguments to other opcodes. The command may be used freely in the RVP10 instruction stream, even if the opcode being modified does not expect any optional arguments. **XARGS** is a **NOP** in that case.

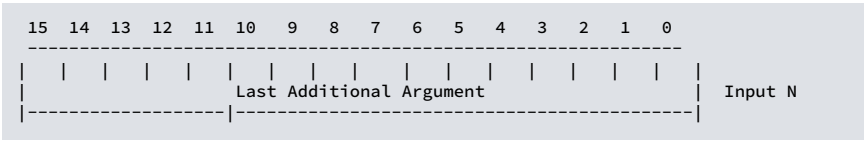
To supply optional arguments to another opcode **OP**, the **XARGS** command is first executed with the additional argument count encoded in its upper 11-bits. This is followed by the array of 0 ... 2047 additional arguments. At this point the **XARGS** command finishes and the **OP** command is fetched as the next instruction. **OP** executes normally, except that the additional arguments from **XARGS** can be picked up after its own input list has been read to completion.

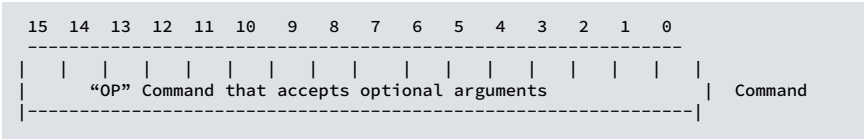
XARGS affects only the opcode that immediately follows it. The entire list of optional arguments is discarded after **OP** executes, even if **OP** did not use some or all of the list.

However, if **OP** is another **XARGS** command, then the additional arguments that it supplies are appended to the first set. In this way, **XARGS** can supply an arbitrarily large number of additional arguments.



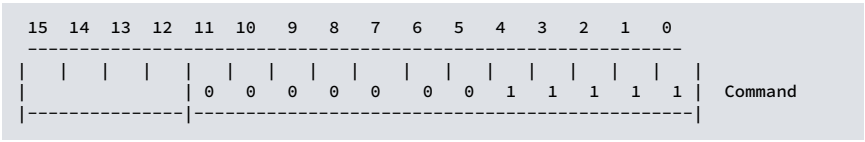


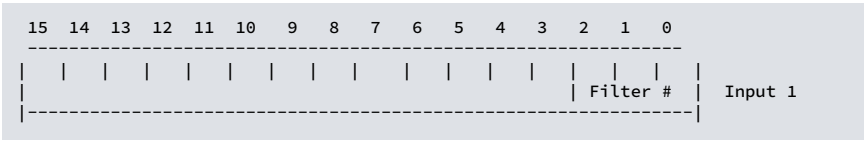




C.22 Load clutter filter specifications (**LFSPECS**)

RVP10 allows 7 different clutter filters (plus the fixed all-pass filter) to be resident at once, so that an appropriate filter can be selected and applied to each processed ray based on Range, Azimuth, and/or Elevation. The **LFSPECS** command allows this suite of filters to be redefined on the fly.





```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                          Filter Type                          |  Input 1
|---------------------------------------------------------------|
 Variable number of XARGS depending on Filter Type
```

The **`Filter #`** tells which filter definition slot is being modified, and the **`Filter Type`** tells the type of clutter filter to construct.

The command is followed by additional **XARGs** that give the specific filter parameters. Beginning with the **`Filter Type`**, the complete **XARG** list is a **`struct RVP10SpecFiltIO`** (See **`include/RVP10.h`**) for each of the following filter types.

`Type:0 SPFILT_FIXED Fixed Width Spectral Filter`

Legacy clutter filter inherited from RVP6/7, specified by a width parameter telling how many points to remove (center zero velocity point, plus one side), plus an **`Edge Points`** parameter telling how many points to minimize on each side of the gap to compute the end points of a linear interpolation to fill the gap.

`Type:1 SPFILT_VARIABLE Variable Width Spectral Filter`

Similar to the fixed width filter except that the width parameter is interpreted as a minimum width, and a third parameter indicates the maximum width. The clutter gap width is dynamically determined at each bin based on the slopes of the spectral terms. Linear interpolation of the gap (based on **`Edge Points`**) is the same as above.

`Type:2 SPFILT_VARLSQ Variable Width / Quadratic interpolation`

Similar to the variable width filter except that quadratic gap interpolation is used. This filter is experimental and should not be used.

`Type:3 SPFILT_GMAP Gaussian Model Adaptive Processing Spectral Filter`

This is the most advanced clutter filter of RVP10, combining the best techniques for determining the clutter gap width and restoring whatever low-velocity spectral points are removed.

This filter is characterized by a single parameter, which is the assumed clutter width expressed as a physical velocity.

C.23 Configure ray header words (**CFGHDR**)

The processed data that is output by the **PROC** command may contain optional header words that give additional information about each ray. This command configures the set of words that makeup each header.

There are (up to) 32 different choices of words or groups of words to include, as indicated by the bit mask following the command. Setting a bit requests that those words be included in the header, and be placed in the order implied by the sequence of the bits. Leaving all bits clear suppresses the header; though this can also be done without changing the configuration in the **NHD** (No-Headers) bit in **SOPRM Input #2**.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               | 0   0   0   0   0   1   0   1   1   1   1   1 |  Command
|---------------|---|---|---|---|------------|-------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                   |TID|PBN|SYT| MMT|UTC|Flg|Gpm|Tim|Pul|PRT|Tag|  Input 1
|-------------------|-----------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                                                               |  Input 2
|-------------------|-----------------------------------------------|
```

`Tag`

Four words containing two 32-Bit TAG samples, one from the beginning and one from the end of the ray:

Word #1	**TAG150**	Start of Ray
Word #2	**TAG3116**	Start of Ray
Word #3	**TAG150**	End of Ray
Word #4	**TAG3116**	End of Ray

- When RVP10 operates in dual PRF mode, bit zero of the start TAG word is replaced with a flag indicating that the ray's PRF was low (**0**) or high (**1**).
- When trigger blanking is enabled, bit zero of the end TAG word is replaced with a flag indicating that the trigger was blanked (**0**) or normal (**1**). Note that the data within a ray are considered to be invalid if any of the pulses that were used to compute the ray were blanked.

RVP10 outputs all zeroed data when a ray contains any blanked pulses.

`PRT`

PRT (Pulse Repetition Time) measured at the end of the ray.
Same format as **GPARM Word #30**.
The measured PRTs are forced to **0xFFFF** (the maximum unsigned value) when the external trigger is expected but missing.

`Pul`

Number of pulses used to compute the ray.

`Tim`
: Milliseconds universal time (0999), sampled at the end of the ray.

`Gpm`
: **`GPARM`**. Sends a copy of the 64-word **`GPARM`** output with each ray.

`Flg`
: Ray Flag word:
  - **`Bit 0`**: Dual PRF is in the low PRF state
  - **`Bit 1`**: Trigger is blanked for this ray
  - **`Bit 2`**: This ray is from one of the PRFSECT special sectors
  - **`Bits46`**: Tells which PRFSECT when Bit-2 is set

`UTC`
: 3-word universal time, sampled at the beginning of the ray:
  - Word 1: Milliseconds (0999)
  - Word 2: Low 16-bits of 32-bit UTC time
  - Word 3: High 16-bits of 32-bit UTC time

`MMT`
: Mis-matched timeseries bits (playback versus RVP10 configuration). See the **`MMTS_*`** **`flags in dsp.h`**.

`SYT`
: IFDR system clock time at the beginning of the ray:
  - Word 1: Low 16-bits of 32-bit clock counter
  - Word 2: High 16-bits of 32-bit clock counter

`PBN`
: Timeseries playback version number

`TID`
: Task ID encoded as **`struct RVP10TaskID_IO`** (14 words total)

`PedINU`
: Auxiliary pedestal and INU information encoded as struct **`RVP10AuxPedINU`**, giving full angle and position information for moving platform systems (18 words total).

C.24 Configure interference filter (**`CFGINTF`**)

RVP10 can optionally apply an interference filter to its incoming (**`I`**,**`Q`**) data stream, with the goal of rejecting occasional and sparse interference from other (usually man-made) signal sources.

The **`CFGINTF`** command is used to choose which filtering algorithm is applied, and to configure its operation with additional **`XARGS`** parameters. See Pass auxiliary arguments to opcodes (**`XARGS`**) (page 431).

If the **`XARGS`** are not supplied, the filter parameters retain their previous values. **`CFGINTF`** with no **`XARGS`** can be used to turn the interference filters On/Off without making any other changes to their threshold constants. Likewise, if only **`XARGS`** is supplied, then that single threshold value is used for both **`C1`** and **`C2`**.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Filter        | 0   0   0   0   0   1   1   1   1   1   1   1 |  Command
|---------------|-----------------------------------------------|
```

`Filter` chooses which interference algorithm should be run. See Interference filter (page 161).

- 0: None (Interference filtering is disabled)
- 1: `Alg.1` (Traditional JMA Algorithm)
- 2: `Alg.2` (`Alg.1` optimized for additive interference)
- 3: `Alg.3` (`Alg.2` with better statistics)

Vaisala recommends `Alg.3` for general operational use. The other algorithms are included mostly for historical reasons.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|      Threshold Parameter C1 in Hundredths of decibels         |  XARG 1
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|      Threshold Parameter C2 in Hundredths of decibels         |  XARG 2
|---------------------------------------------------------------|
```

C.25 Set AFC level (**SETAFC**)

SETAFC sets the AFC level to a given value.

The signed 16-bit span is identical to `GPARM Output #51` which shows the present AFC level, that is, corresponding to the -100 … +100% AFC range that is defined in the **Mb** menu. See **Mb**— Burst pulse and AFC (page 86).

RVP10 automatically converts the new level to the configured analog or digital AFC output format. The exception is for the Motor/Integrator type of AFC loop, for which this command does nothing.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               | 0   0   0   0   1   0   1   1   1   1   1   1 |  Command
|---------------|-----------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|      16-Bit AFC/MFC Value (-32768 through +32767)              |  Input 1
|-------------------|-----------------------------------------|
```

C.26 Set Trigger Timing Slew (**SETSLEW**)

The **Mt** menu allows you to select a subset of triggers that can be slewed left and right to place the burst pulse accurately at range zero.

SETSLEW allows you to manually set the present amount of slew. The input argument is in hundredths of microseconds, that is, ranging from - 327.68 .. +327.67 µsec. The permitted span is ±20 µsec.

This is the same format used in **GPARM Output #56** which shows the present slew value.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               | 0   0   0   0    1   1   0   1   1   1   1   1 |  Command
|---------------|-----------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|        Signed trigger slew in hundredths of microseconds      |  Input 1
|---------------------------------------------------------------|
```

C.27 Hunt for burst pulse (**BPHUNT**)

The **BPHUNT** command allows the host computer to initiate hunt mode when it knows or can sense that a burst pulse should be present.

BPHUNT starts the internal procedure to hunt for a missing burst pulse when we are uncertain of both its time and frequency.

Depending on how the hunting process has been configured in the **Mb** menu, the procedure may take several seconds to complete. The RVP10 host computer interface remains functional during this time, but any acquired data is questionable.

GPARM status bits in **word #55** indicate when the hunt procedure is running, and whether it has completed successfully.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|          |Now| 0   0   0   0    1   1   1   1   1   1   1   1 |  Command
|----------|---|---|---|---|---|-----------|-------------------|
```

Now forces the hunt procedure to start even if the burst pulse is already present.

Normally the procedure only starts when the burst pulse is missing at the time **BPHUNT** is given.

C.28 Configure phase modulation (**CFGPHZ**)

CFGPHZ configures RVP10 phase control output lines, which determine the relative phase of each transmitted pulse.

In some cases the chosen phase sequence has side effects elsewhere in the processor, for example, different algorithms may be used in **Random Phase** mode according to the transmit sequence that is requested.

Some phase sequences chosen by **CFGPHZ** also expect additional arguments to have been supplied by the **XARGS** command.

Phase sequences are expressed as a list of **N** 16-bit binary angles representing the desired phase sequence. The sequence is assumed to be periodic with period **N**.

The **Mz** command defines the correspondence between phase codes and phase angles.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |  PhSeq    | 0   0   0   1   0   0   0   1   1   1   1   1 |  Command
|---|-----------|-----------------------------------------------|
```

PhSeq=0
Selects **No Modulation**. RVP10 outputs a constant default phase request as defined in the **Mz** menu.

PhSeq=1
Selects a **Random Phase** sequence. This is also the default phase modulation that is output following power-up. From the set of valid phase codes that are defined in the **Mz** setup section, a random code is automatically chosen for each pulse. Each code has an equal probability of being chosen each time, and the choice is independent of any previous state. No **XARGS** words accompany this command.

PhSeq=2
Selects a **User Defined** sequence. If no **XARGS** have been supplied, RVP10 outputs the default idle phase that is defined in **Mz**. If **XARGS** are supplied, then they are interpreted as a sequence of 16-bit binary angles.
RVP10 makes the best match between each desired angle and the closest realizable angle that the phase modulation hardware can produce. The maximum length of the sequence is 1024 pulses.

`PhSeq=3`

Selects the `SZ(8/64)` sequence.

This is a systematic code[1] that separates and recovers first and second trip echoes in `Random Phase` mode. It usually performs better than a truly random transmit sequence, especially when the processing interval is fairly short (as little as 32-pulses). With no **`XARGS`**, RVP10 automatically generates the phase sequence using the closest realizable angles that the phase modulation hardware can produce. This is the recommended way to invoke `SZ(8/64)` coding. You may also supply your own 32-pulse angle sequence.

C.29 Set user IQ bits (**`UIQBITS`**)

`UIQBITS` loads user-specified bits that are included with the pulse headers in the RVP10 `TimeSeries` API data stream.

The permanent `Set/Clr` bits are updated in the signal processor and retain their value from the last time they were defined. These bits are then repeated in all pulse headers. The **`ONCE`** bits are transitory and appear in only one pulse header each time they are set.

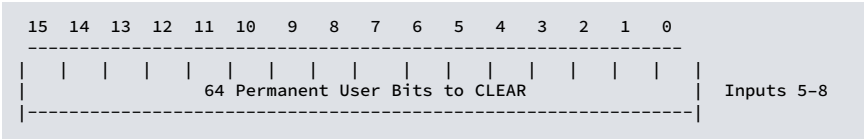
A `FIFO` history of the permanent bits is maintained so that the bits can be associated with the data being acquired right now as the **`UIQBITS`** opcode is executed. Each 16-bit command arg specifies bits to `Set/Clr` in successive bytes of the structure. This allows user code to safely change some bits without affecting others.

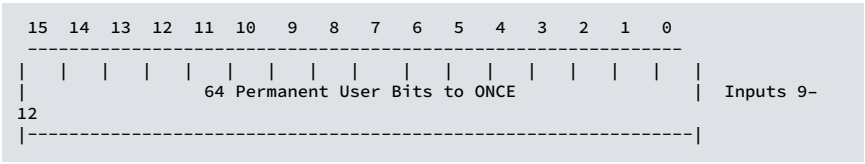
The user bits from separate calls are never be collapsed into a single pulse header, even if the header and bit times indicate that they could. This means that each **`UIQBITS`** opcode always result in at least one pulse header being tagged with exactly that data. This is generally what you want, since no other exact outcome could be guaranteed based on time-of-arrival alone.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               | 0   0   0   1     0   0   1   1   1   1   1   1 |  Command
|---------------|-------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                 64 Permanent User Bits to SET                 |  Inputs 1–4
|---------------------------------------------------------------|
```

1) *Sachidananda, M., D.S. Zrni, and R.J. Doviak, 1997: Signal Design and Processing Techniques for WSR–88D Ambiguity Resolution. National Severe Storms Laboratory Report, Part 1, Norman, OK, 100 pp.*



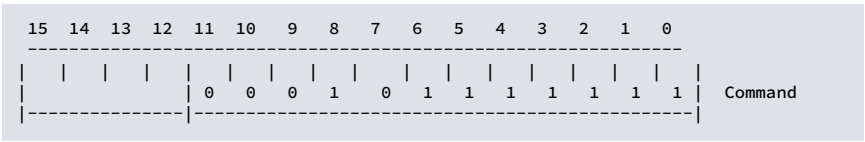


C.30 Set Individual Thresholds (**THRESH**)

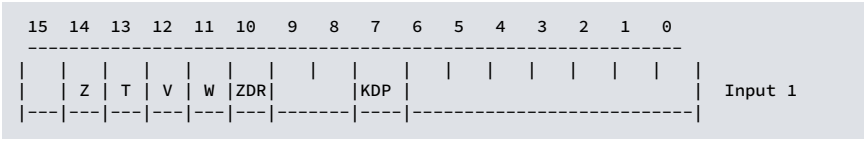
The **SOPRM** command allows you to configure 4 threshold numbers used by all data types, and to select the threshold control flags for 5 of the data types. See Setup operating parameters (**SOPRM**) (page 368).

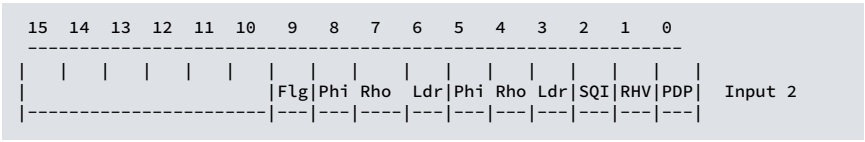
Use the **THRESH** command if you wish to apply different threshold numbers to different data types. You can individually set the thresholds and mask used for each data type, or for groups of data types.

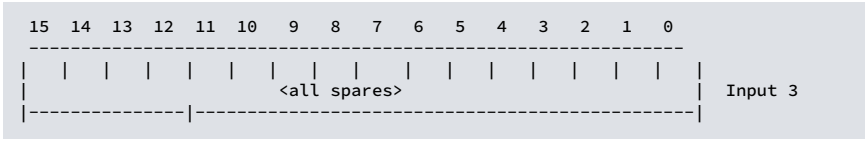
The **GPARM** command reads out the threshold numbers set for velocity. To read back the numbers for each data type use the **RBACK** command. See Read back internal tables and parameters (**RBACK**) (page 429).

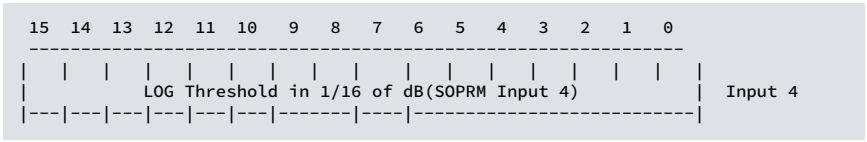


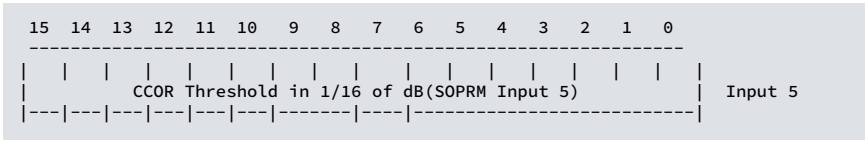
The first 3 words supply a mask that indicates which data types are being set:

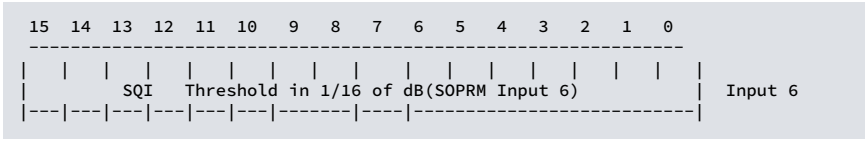


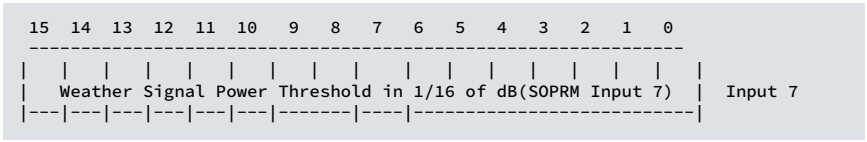


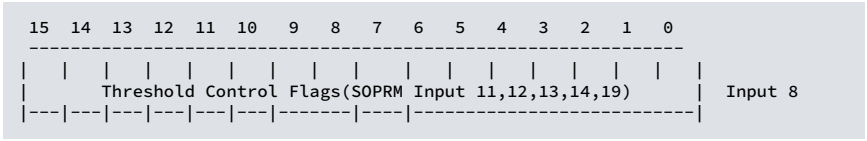

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
<all spares>
Input 3

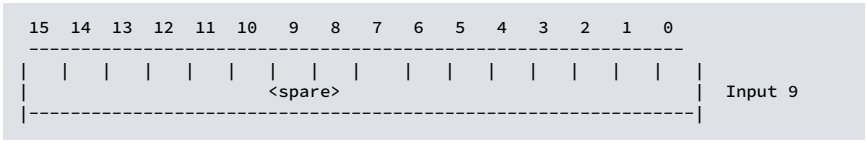

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
LOG Threshold in 1/16 of dB(SOPRM Input 4)
Input 4

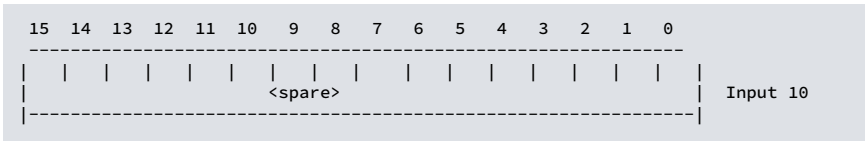

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
CCOR Threshold in 1/16 of dB(SOPRM Input 5)
Input 5


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
SQI Threshold in 1/16 of dB(SOPRM Input 6)
Input 6


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Weather Signal Power Threshold in 1/16 of dB(SOPRM Input 7)
Input 7


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Threshold Control Flags(SOPRM Input 11,12,13,14,19)
Input 8


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
<spare>
Input 9


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
<spare>
Input 10

C.31 Set task identification information (**TASKID**)

TASKID allows you to name the (I,Q) data that are currently being acquired by RVP10.

This naming information then becomes associated with these data, and is available in the pulse information structures (`struct RVP10PulseInfo`) that are read from the `Timeseries` API. The `iAqMode` field of the pulse headers (`struct RVP10PulseHdr`) are incremented each time a **TASKID** opcode is received, but the continuous flow of (I,Q) data from the RVP10/Rx card(s) are disturbed.

The **TASKID** command defines a 16-character Null-terminated name, along with a 16-bit sweep number and 16-bit auxiliary (user defined) number.

You may use all 16 characters of the name, as it is stored internally in 17 slots.

The `Sweep Number` and `Scan Geometry` (one of `SCAN_xxx parameters`) should be filled in with values best approximating those notions. `Auxiliary Number` may be filled in with any value that you find meaningful.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               | 0   0   0   1    0   1   1   1   1   1   1   1 |  Command
|---------------|-----------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                       Sweep Number                              |  Input 1
|-----------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|             Auxiliary (User Defied) Number                      |  Input 2
|-----------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                    Name Byte-2|Name Byte-1                     |  Input 3
|-------------------------------|-------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                               |Scan Geometry                  |  XARG 1
|-------------------------------|-------------------------------|
```

C.32 Define PRF Pie Slices (**PRFSECT**)

This command supplements the **SETPWF** command and allows an alternate trigger PRF to be generated within prescribed AZ/EL sectors. See Set pulse width and PRF (**SETPWF**) (page 422).

Up to 8 trigger sectors can be defined by invoking **PRFSECT** for each separate region. The trigger pattern then automatically changes when the antenna enters any of these regions, but the timeseries data remains continuous and uninterrupted throughout each change. **PRFSECT** allows a complete volume scan to run with PRFs that have been optimized to the radar echoes in all directions. Note the following caveats should be observed:

- Dual-PRF unfolding cannot work properly at PRF sector boundaries. We recommend not using Dual-PRF and **PRFSECT** at the same time.
- When PRF sectors are used in conjunction with angle sync'ing, it is best to set the PRF sector boundaries at the midpoint between individual sync angles. This prevents the PRF seam from bobbling between two adjacent sync angles.

The **SETPWF** opcode deletes any alternate sectors that have been setup so far. Thus, you can only use the **PRFSECT** command after **SETPWF** has established the pulse width and default trigger rate for the entire AZ/EL scan volume.

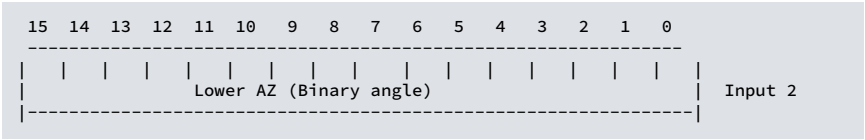
```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               | 0   0   0   1   1   0   0   1   1   1   1   1 |  Command
|---------------|-----------------------------------------------|
```

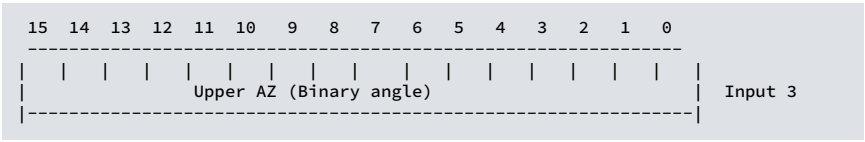
```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                                                   |Sector     |  Input 1
|---------------------------------------------------|-----------|
```

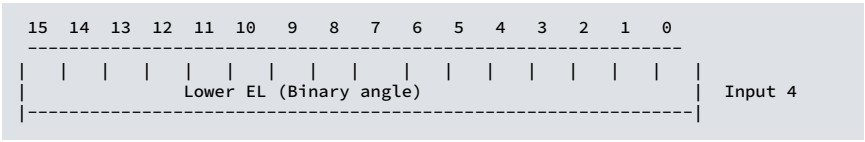
`Sector` selects which sector number is defined to have an alternate trigger pattern. This is an arbitrary index from 0 … 7.

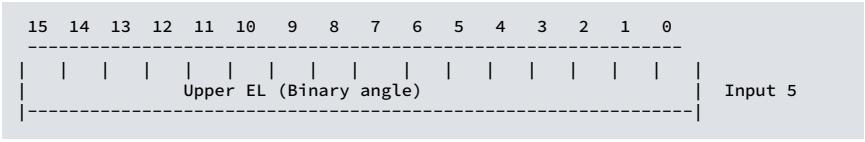
The following 4 arguments define a solid sector in azimuth and elevation within which the alternate trigger pattern is used in preference to the default (**SETPWF**) pattern. When the current `AZ/EL` angle pair is contained in more than one defined sector, the trigger pattern from the lower numbered sector is used.

The sector bounds are inclusive, that is, they include the upper/lower `AZ/EL` boundaries themselves. This convention makes it simpler to define several contiguous regions without generating slivers in between.

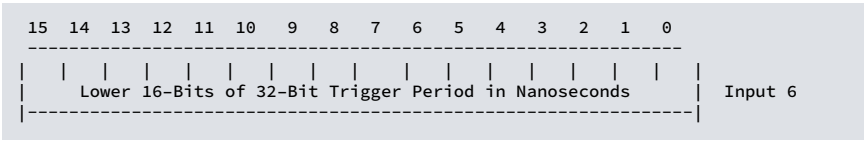


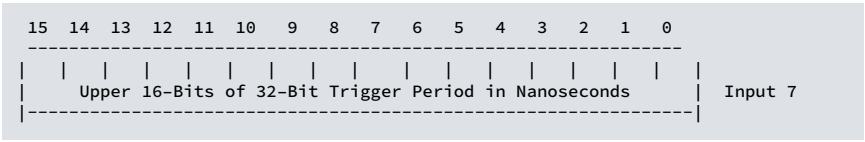






The following arguments specify a trigger period in the same manner as the optional form of the **SETPWF** command.



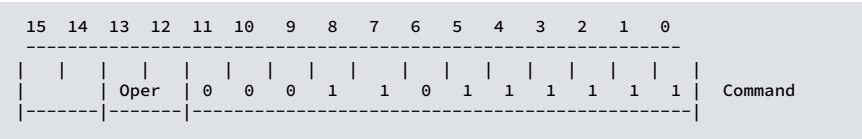


C.33 Configure target simulator (**TARGSIM**)

RVP10 contains a built-in target simulator tool that can test and debug processing algorithms that work with multiple trip returns. Several real physical targets can be simulated, each having a range span measured in kilometers, a Doppler shift in Hertz, and an echo power relative to the saturation level of the receiver. The echoes are placed in range exactly according to how they have been illuminated by whatever sequence of pulses have been transmitted so far. Multiple trip returns and range folding are all modeled correctly.

The target simulator can be used with both live and simulated (**I**,**Q**) data. See Load simulated time series data (**LSIMUL**) (page 416).

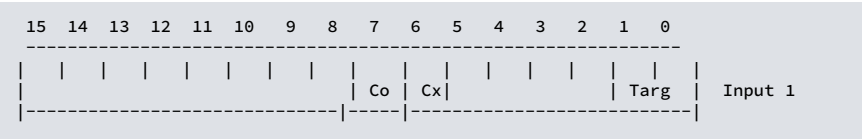
In the simulated case, you can overlay simulated physical targets on top of real physical targets from the radar receiver.



`Oper=0`
Disable all target simulation activity (no additional arguments)

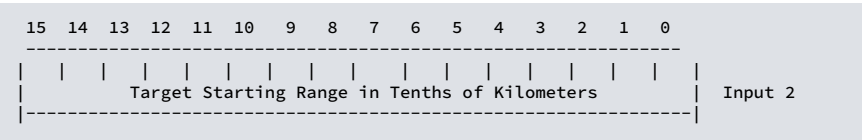
`Oper=1`
Enable simulation of all defined targets (no additional arguments)

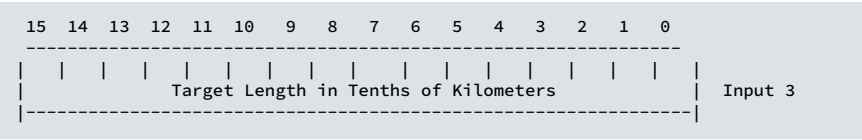
`Oper=2`
Define a new simulated target (arguments list follows)

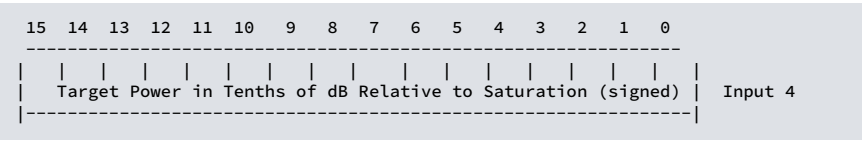


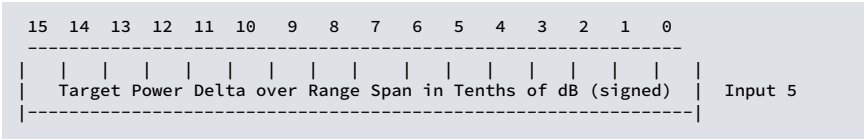
`Targ`
Which target is being defined

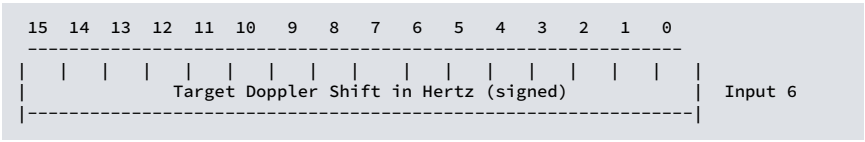
`Co/Cx`
Place simulated target in Co-Pol and/or Cross-Pol Rx channels

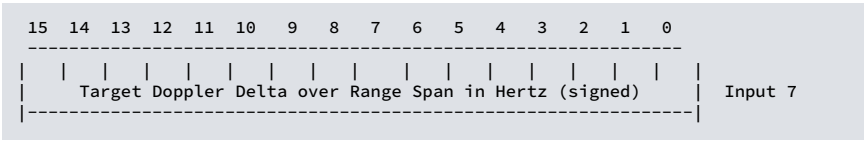


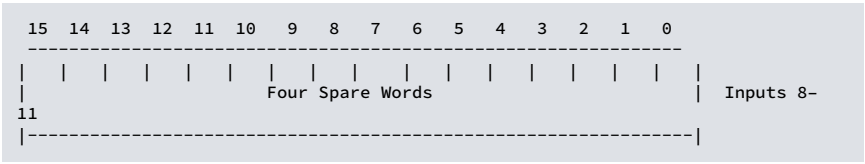






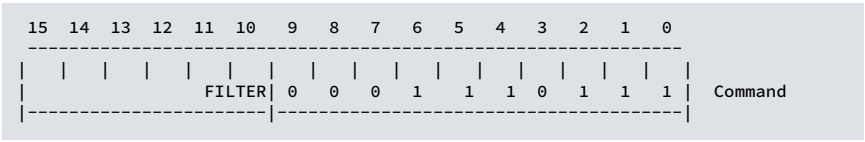


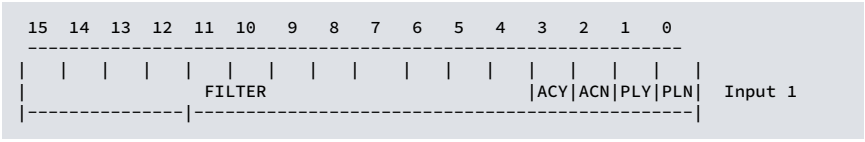




C.34 Set burst pulse processing options (**BPOPTS**)

Use this command to set burst pulse processing options.





PLY/N
Affects whether RVP10 phase locks its (**I**,**Q**) data to the measured burst pulse. The **PLY** and **PLN** bits force **Yes** and **No** responses. If both bits are clear or both bits are set, no change is made.

`ACY/N`
: Affects whether RVP10 applies pulse-to-pulse amplitude correction to its (**I**,**Q**) data. The **ACY** and **ACN** bits force **Yes** and **No** responses. If both bits are clear or both bits are set, no change is made.

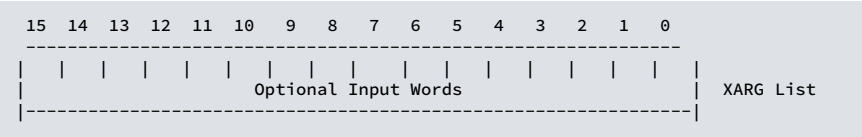
C.35 Custom User Opcode (**USRINTR** and **USRCONT**)

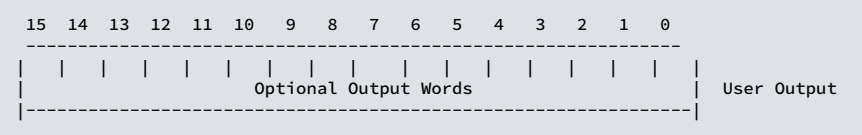
These opcodes are part of the open software extensions that allow you to define custom opcodes for each major mode of operation.

Arguments may be passed into a custom opcode handler as an **XARG** list. An optional array of words returned from that handler appears after the command executes.

`UserBits`
: Four additional bits defined by the user to help subdivide the opcode functions.

`CON`
: If set, the **I**,**Q** data acquisition thread proceeds continuously while the opcode is executed.
If clear, the **I**,**Q** stream is interrupted before handling the call.





C.36 Load melting layer specification (**MLSPEC**)

RVP10 allows you to get melting layer-specific information, including the melting layer height and the uncertainties, from the latest melting layer product created in IRIS.

The melting layer specifications are set as a function of range, azimuth, and elevation based on the information coming from the configuration of the task and either the melting layer product parameters used in the latest melting product created or the melting layer product default parameters.

RVP10 can use spatially variable melting layer (ML) altitudes, which may be preloaded for each interval of data processing (**PROC**). The ML altitudes are referenced to the Mean Sea Level (MSL), and estimate the top of ML.

The input data are the maps of melting layer altitudes projected into sweep cones of the data sweeps, to be carried out by the **PROC** command.

The command is an extended **OpCommand** and is defined as follows:

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|               | 0   0   0   1    0   0   0   1   1   1   1   1 |  Command
|---------------|-----------------------|---|-------------------|
```

The lowest 5 bits correspond to the default **0X1F** for extended **opcode** commands and the 7 middle bits are the value for the **XOP_MLSPEC** command (**0X10**).

The application software provides input data by, for example, converting the IRIS standard cartesian MLHGT products to the curved Earth coordinates of the sweeps configured for the radar task. Typically, a map for a collection of sweeps (a volume) is uploaded.

The application software adjusts the spatial resolution of the map by down scaling of gates so that it fits in the buffer managed in RDA:

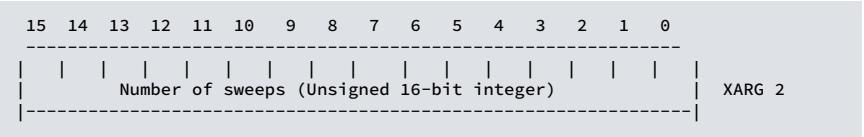
```
DPOLAPP_MAX_ML_SWEEPS*DPOLAPP_MAX_ML_RAYS*DPO LAPP_MAX_ML_BINS
```

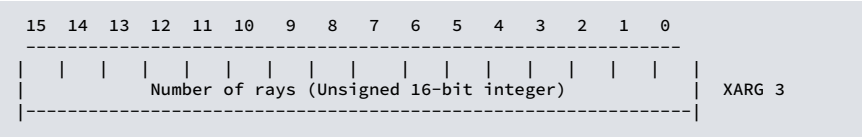
The command also checks the maximum number of sweeps and rays (**DPOLAPP_MAX_ML_SWEEPS = 30** and **DPOLAPP_MAX_ML_RAYS=90**). If the buffer does not exist or exceeds its limits, it is flushed.

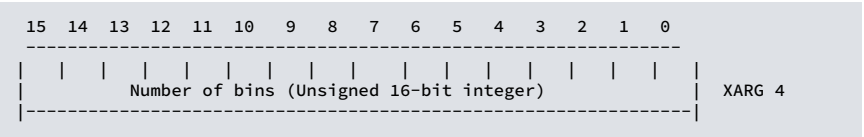
RVP10 supports a command that provides a way to feedback the melting layer information from IRIS to the RDA. The melting layer altitude information is specified for each antenna angle and range. This way, the melting layer height can be used in different live data processing applications such as **HydroClass**.

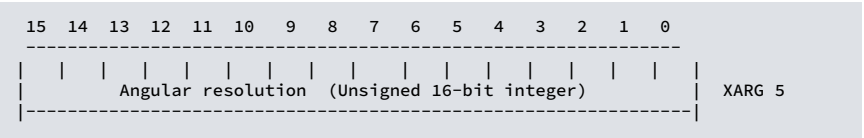
RVP10 maintains an internal array of up to 1024 different filter versus- range tables, each of which is keyed to a particular angle (**EL**, for PPIs and **AZ** for RHIs). Each **XOP_MLSPEC** command uploads the complete sweep. Then, for each live bin in every processed ray processed, RVP10 obtains the melting layer at the midpoint **AZ**/**EL** of the ray.

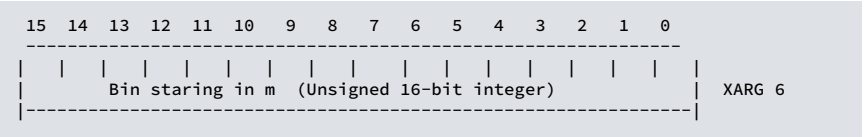
```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|           Scan Type (Unsigned 16-bit integer)                 |  XARG 1
|---------------------------------------------------------------|
```

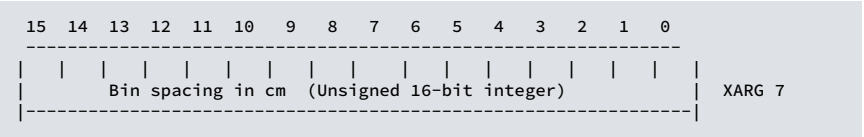

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Number of sweeps (Unsigned 16-bit integer)
XARG 2

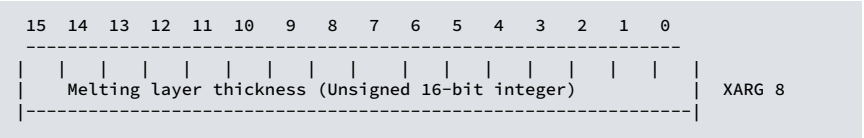

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Number of rays (Unsigned 16-bit integer)
XARG 3

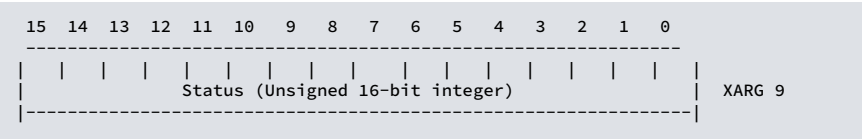

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Number of bins (Unsigned 16-bit integer)
XARG 4


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Angular resolution (Unsigned 16-bit integer)
XARG 5


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bin staring in m (Unsigned 16-bit integer)
XARG 6


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bin spacing in cm (Unsigned 16-bit integer)
XARG 7


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Melting layer thickness (Unsigned 16-bit integer)
XARG 8


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Status (Unsigned 16-bit integer)
XARG 9

The **XARGS** statements allow flexibility for backward compatibility. They include the current task and the product configuration settings. The command uploads the melting layer height and thickness for every volume.

`Scan type`
The melting layer information can be uploaded to the RDA for both RHI and PPI scans.

`Number of sweeps`
The minimum between the number of sweep of the running task and the `DPOLAPP_MAX_ML_SWEEPS`.

`Number of rays`
The number of rays is calculated as $\#rays = \frac{360000}{ires}$

where `iRES` is the desired angular resolution expressed as an integer number of thousands of degrees.

`Number of bins`
The number of output bins.

`Angular resolution`
The binary angle of the corresponding angle in degrees, $\theta = \frac{360}{nMLrays}$

`Bin starting`
Range of the first bin in centimeters.

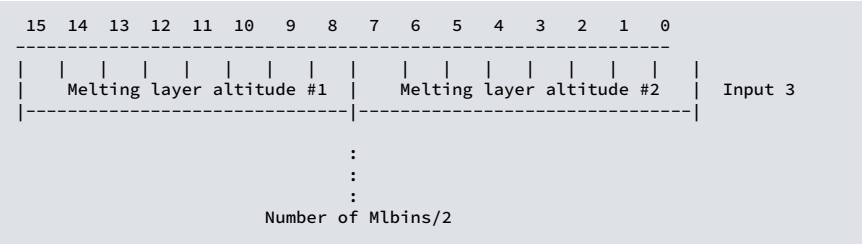
`Bin Spacing`
The product of the number of output bins by the step divided by the number of bins.

`Melting layer thickness`
The user setting of the melting layer thickness in the product configuration.

`Status`
Indicates the status of the DSP features.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|        Static elevation angle (16-bit binary angle)           |  Input 1
|---------------------------------------------------------------|
```

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                Ray angle (16-bit binary angle)                |  Input 2
|---------------------------------------------------------------|
```


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Melting layer altitude #1
Melting layer altitude #2
Input 3
Number of Mlbins/2

Appendix D. Recycling instruction

These recycling instructions guide you on the end-of-life treatment of this Vaisala product. As waste regulations and infrastructure vary in each country, these instructions only indicate the different components to be separated and common ways to handle them. Always follow local requirements when disposing of the product. Vaisala encourages to use the best available recycling practices to minimize related environmental impacts.

Vaisala is committed to meeting the requirements of the EU Waste Electrical and Electronic Equipment (WEEE) Directive. This directive aims to minimize the impact of electrical and electronic goods on the environment, by increasing reuse and recycling, and reducing the amount of WEEE going to landfill. This symbol indicates that the product should be collected separately from other waste streams and treated appropriately.

For recycling the RVP10SRV server, see manufacturer's documentation.

IFDR10 includes an aluminium chassis, which must be disposed of as metal waste, and a circuit board that must be disposed of as electrical/electronic waste.

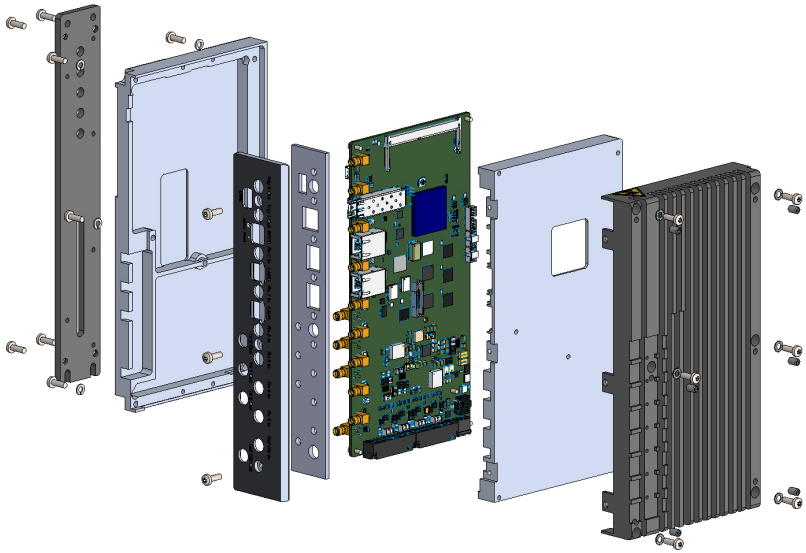


Figure 74 IFDR10 exploded view

Index

I

W

Z

Warranty

For standard warranty terms and conditions, see vaisala.com/warranty.
Please observe that any such warranty may not be valid in case of damage due to normal wear and tear, exceptional operating conditions, negligent handling or installation, or unauthorized modifications. Please see the applicable supply contract or Conditions of Sale for details of the warranty for each product.

Technical support

Contact Vaisala technical support at helpdesk@vaisala.com. Provide at least the following supporting information as applicable:

- Product name, model, and serial number
- Software/Firmware version
- Name and location of the installation site
- Name and contact information of a technical person who can provide further information on the problem

For more information, see vaisala.com/support.

VAISALA