R&S®Pulse Sequencer digital, R&S®Pulse Sequencer PDW Import Interface Interface Specification

1178390502
Version 06

ROHDE&SCHWARZ
Make ideas real

This manual describes software version V2.4 and later of the R&S®Pulse Sequencer.

Muehldorfstr. 15, 81671 Muenchen, Germany
Phone: +49 89 41 29 - 0
Email: info@rohde-schwarz.com
Internet: www.rohde-schwarz.com

1178.3905.02 | Version 06 | R&S®Pulse Sequencer digital, R&S®Pulse Sequencer

The following abbreviations are used throughout this manual: R&S®Pulse Sequencer is abbreviated as R&S Pulse Sequencer.

Contents

1 Overview

Starting at R&S Pulse Sequencer software version V1.4 the waveform object can import custom text-based PDW lists. This imported data is stored inside the repository and during scenario calculation a waveform or extended sequencer data is generated from this PDW data.

Scope

This document serves as an interface specification for the PDW list file as well as the required template for parsing the PDW list file.

We assume that you are familiar with the R&S Pulse Sequencer principles, nomenclature and user interface.

This document does not describe the operating of the software and does not substitute the R&S Pulse Sequencer user manual. This document is intended to guide you writing export filters to convert emitter database records into a format that R&S Pulse Sequencer can read.

1.1 What's new

Compared to the previous version, the following new features are described:

- PDW import interface supports import of waveforms in Rohde & Schwarz waveform format, see Chapter 3.2.3, "MOP type identifiers", on page 14 and Chapter 3.2.9, "Waveform file parameters", on page 18.

2 PDW import mechanism

R&S Pulse Sequencer uses a template-based import mechanism for the PDW import. Import templates are human readable text files that describe how information is extracted from the PDW list file. The PDW list file is also a human readable text file using one single row per PDW. The columns contain the various parameters related to the PDW.

The diagram on Figure 2-1 shows the principal concept.

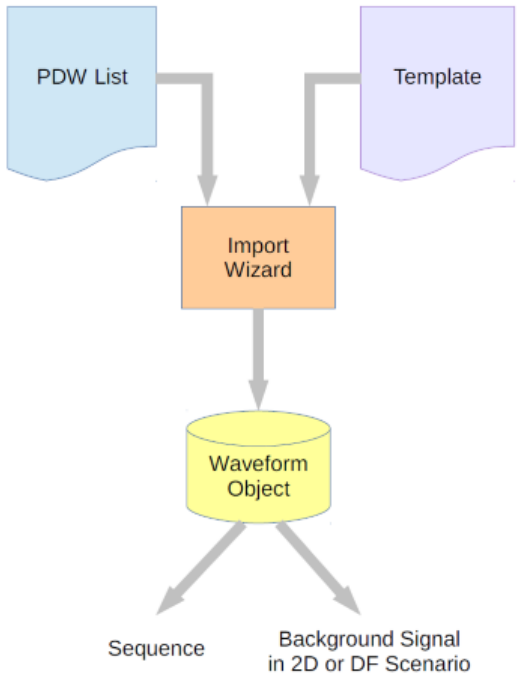

Figure 2-1: Import mechanism principle

The "PDW Data Import" wizard is a dialog that loads both files and shows their content. This dialog is also used to control the import process and accept the imported data.

The import process starts by parsing the text-based template file. Once parsing the template is complete, R&S Pulse Sequencer loads and analyzes the PDW list file. All imported data is temporarily stored in memory until you choose to store the data permanently in the repository.

Storing the data lets R&S Pulse Sequencer copy the extracted PDW data to a waveform object within the R&S Pulse Sequencer repository. The internal storage format is a proprietary binary stream. If future extensions to the data stream are required, R&S Pulse Sequencer automatically converts existing streams to the newer format.

The waveform container with the PDW information can be used as part of a sequence or as a background emitter, for example. Thus, using this signal is identical to using a captured RF signal, digital standard signals originating from R&S® WinIQSIM2™, or background emitter noise.

After the import process has completed the original template and PDW files are no longer required.

3 Template syntax

The template is a human readable text file. It describes how information is extracted from the PDW list file. This chapter deals with the syntax of the PDW template.

The following example shows a complete PDW import template as an example.

Example:

```
# Pulse Sequencer Import Template Example
# 16.6.2016, Rohde&Schwarz

COMMENT          :   TEXT 'Imported from \n $PDWFILE \n $DATETIME'
START_LINE       :   STARTS_WITH \s*\d+
END_LINE         :   STARTS_WITH 'END'
COLUMN_SEPARATOR:    SPACE
DECIMAL_CHAR     :   DOT

 1 : FREQUENCY     GHz      # center frequency (carrier)
 2 : RFOFFSET      kHz      # frequency offset
 3 : WIDTH         us       # pulse width
 4 : PRI           ms       # PRI not used
 5 : PHASE         DEG      # start phase
 6 : MOP           Text     # modulation type
      'CW'      : CW
      'FMOP'    : FSK
 7 : CHIPRATE      kHz      # chip or symbol rate
 8 : STATES        1        # number of states
 9 : FSKSTEP       kHz      # FSK step size (centered around f0)
10 : CHIPCOUNT     1        # number of chips or symbols
11 : PATTERN       HEX      # bit pattern, hex encoded
12 : *                      # ignore this column
```

The template starts with a header block that contains general settings, such as the column separator or the decimal char. It also defines at which line of the PDW list file the import begins.

The subsequent block defines the individual columns of the PDW list file. A column can only contain one specific parameter, such as the pulse width or the frequency offset. It is not possible to use columns for different parameters across different pulses.

Blank lines and white spaces at the beginning of a line are ignored.

Lines starting with the # character are treated as comments. In addition, all text following the # character within a line is also treated as comment.

3.1 Header directives

The header section is always the first part of a PDW import template. It sets general parameters that concern the import.

The general syntax of all header directives is identical. An initial keyword is separated by a colon from all following parameters.

```
<Keyword> : <Parameter> [<Data>]
```

All keywords and their parameters are described in this chapter. The keywords and parameters are not case-sensitive.

3.1.1 COMMENT

The `COMMENT` directive defines how a comment can be extracted from the PDW list file. This comment is automatically copied to the comment of the waveform object in the R&S Pulse Sequencer repository.

The following parameters are supported:

- `TEXT '<String>'`
- `LINE <Line Number>`

The `TEXT` parameter directly defines a text that is copied to the comment field. In this case, no information from the PDW list file is extracted.

`LINE` specified a fixed line in the PDW file. The content of this line is used as the comment. The line number is one-based.

Example:

```
COMMENT  LINE  3
COMMENT  TEXT  'Imported on $DATETIME'
```

This example reads the text from line 3 of the PDW file into the comment field of the waveform object.

The text retrieved from the PDW list file or directly set through the `TEXT` option can contain variables that are evaluated during the PDW import process. The variable names are not case-sensitive.

$PDWFILE	PDW list filename (without path)
$PDWPATHFILE	PDW list path and filename
$DATETIME	Current date and time in short local format
$ISODATETIME	Current date and time in ISO format (YYYY-MM-DDTHH:mm:ssTZD)

3.1.2 START_LINE

This directive defines the start condition for the PDW import. R&S Pulse Sequencer starts reading PDW data from the PDW list file on the line where the start condition exactly matches.

This directive accepts the following parameters:

- `STARTS_WITH '<String>'`
- `STARTS_WITH <Regular Expression>`
- `LINE <Line Number>`

Text following the `STARTS_WITH` parameter is interpreted as case-sensitive string if it is enclosed in single quotes. In this case, the import begins on the first line that starts with the specified string.

Regular expression matching is a powerful pattern matching method. This option allows you to begin the import on a line staring with a numeric character, for example. The Chapter 5, "Regular expression matching", on page 20 describes the regular expression syntax in great detail. For example, the regular expression `\s*\d+` matches lines starting with any number of white spaces and then containing at least one numeric digit.

The `LINE` parameter sets the import line directly. The line number is one-based and includes blank lines.

Example:

```
START_LINE STARTS_WITH \s*\d+
```

This example starts the import of PDW data on the first line starting with a numeric digit. Any trailing white spaces are ignored.

3.1.3 END_LINE

This directive defines the condition for stopping the PDW import. The line identified by this directive is the first line being excluded from the PDW import.

This directive accepts the following parameters:

- `END_OF_FILE or EOF`
- `STARTS_WITH '<String>'`
- `STARTS_WITH <Regular Expression>`
- `LINE <Line Number>`

Except for the parameter `END_OF_FILE` (or `EOF`), the syntax of this directive is identical to `START_LINE`.

`EOF` indicates that the import continues until the end of the PDW list file is reached.

3.1.4 COLUMN_SEPARATOR

The column separator defines the character that is used as the delimiter between individual columns of a PDW file. Multiple delimiting characters in a sequence are treated as one delimiter.

This directive accepts the following parameters:

- `SPACE`
- `SEMICOLON`
- `COMMA`

The space character (0x20) is the default delimiter if this directive is omitted.

3.1.5 DECIMAL_CHAR

The decimal character applies to all number conversions.

This directive accepts the following parameters:

- `DOT`
- `COMMA`

The `DOT` is the default decimal character if this option is omitted.

3.2 Column definitions

The column definitions specify which information is provided in the columns of the PDW list file. A column can only be used for one specific parameter.

The general syntax for the column definition in the PDW file is as follows:

```
<row> : <type tag> <unit>
```

The `<row>` is a one-based number. A PDW template can only contain one single description for each row number.

The `<type tag>` denominates the information provided in that specific row. A detailed list of all available type tags is provided in Chapter 3.2.1, "General type tags", on page 11.

The `<unit>` can be used to translate a numeric value, e.g. from us to seconds. The unit applies to all values in a column.

3.2.1 General type tags

Table 3-1 lists all general type tags. These type tags are valid for all pulses and do not depend on other parameters, such as `MOP`.

Table 3-1: General type tags

Type Tag	Units	Meaning
*		Ignore column
TOA	TIME MSEPOCH	Time of arrival
FREQUENCY	HZ, KHZ, MHZ, GHZ	Absolute frequency
RFOFFSET	Hz	Frequency offset
TOFFSET	S, MS, US, NS	Additional timing offset
WIDTH	S, MS US, NS	Pulse width, 0%/100%
LEVELOFFSET	DB	Level offset
PHASE	DEG	Phase at beginning
PRI	S, MS US, NS	Pulse repetition interval
MOP	Text	Text defining a modulation on pulse
VALUES	Text	Custom text data for MOP = CPH and MOP = PLFM
MARKER	Text	Binary Marker Flags 1,2,3,4 Hex coded

Providing timing information is mandatory. Thus, timing can either be provided by a TOA column or by a combination of PRI and WIDTH.

The pulse width (WIDTH) column is mandatory. Omitting this column causes an error during the import process.

All other information can be omitted. In this case, the values are assumed to be zero and the pulse to be unmodulated (CW).

The frequency of each PDW can be specified as an absolute frequency or as a frequency offset:

- If no frequency is defined (both tags are omitted), it is assumed that all pulses are generated on the center frequency.
- If no center frequency is specified, R&S Pulse Sequencer uses the frequency set in the emitter of the scenario instead.
- If only the absolute frequencies are set, than the software calculates the center frequency based on the absolute frequency values of all PDWs and also the frequency offsets for each PDW.
- If the frequency offsets are defined, these values are used; center frequency is not calculated.

Example:

For three PDWs with FREQUENCY = 3 GHz, 3.1 GHz and 3.2 GHz, the center frequency is 3.1 GHz. The offsets per PDW are -0.1 GHz, 0 GHz and 0.1 GHz.

3.2.2 TOA tag

The TOA data can be provided in two different formats, see:

- "TIME format" on page 13
- "SEPOCH and MSEPOCH format" on page 14

The absolute time information of the PDW data is removed during import. Instead the first pulse is always set to T = 0 and all following pulses use the time difference related to this pulse.

As an alternative to providing TOA directly it is also possible to use a combination of WIDTH and PRI. In this case, the time starts at T = 0 and the start time of the current pulse is derived from the start time and PRI of the previous pulse.

TOA has priority over PRI. Hence, if both values are provided, then PRI is ignored.

TIME format

Using the TIME format requires an additional line in the template immediately following the column definition for the TOA column. This line contains the detailed format specification of the date/time format.

The following example demonstrates how to define the TOA in TIME format.

Example:

```
1 : TOA     TIME          # TOA provided in TIME format
      'hh:mm:ss.zzzzzzzzz'
2 : RFOFFSET   MHz         # Frequency offset
3 : WIDTH      us             # Pulse width
```

The following characters are supported in the date/time string. The characters are case-sensitive.

Character	Meaning
d	day
M	month
y	year
h	hour
m	minute
s	second
z	milliseconds

The length of the milliseconds field is usually 3 digits long and permit values from 000 to 999. R&S Pulse Sequencer uses an extended field where microseconds and nanoseconds can be added without using a decimal point or any extra character.

In the above example, the character z is used nine times which means that the last digit has nanoseconds resolution.

SEPOCH and MSEPOCH format

(seconds/milliseconds since start of epoch)

Using these time formats does not require an extra line in the template. Both formats expect the timestamp as floating point number in seconds/milliseconds since `1970-01-01T00:00:00.0000 UTC`.

3.2.3 MOP type identifiers

The `MOP` entry always requires the unit to be set to `TEXT`. This unit type instructs R&S Pulse Sequencer to convert a custom text field into an internal `MOP` type. Using `TEXT` requires a translation block in the template file immediately following the line defining the `TEXT` columns. This translation block uses the general format:

`'<text>' : <id>`

Table 3-2 lists the supported `MOP` types (`<id>`).

Table 3-2: Supported MOP tags

MOP Tag	Meaning
`CW`	Unmodulated
`AM` `FM`	Amplitude modulation Frequency modulation

`ASK`
`FSK`
`PSK`
	Amplitude shift keying
	Frequency shift keying
	Phase shift keying

`LFM`
`NLFM`
`TFM`
`PLFM`
	Linear frequency modulation (chirp)
	Non-linear chirp
	Triangular chirp
	Piecewise linear chirp

`BKR3`
`BKR4A`
`BKR4B`
`BKR5`
`BKR7`
`BKR11`
`BKR13`
`BKR15`
	Barker pulse

`CPH`	Custom phase
`WVFILE`	Requires R&S Pulse Sequencer Waveform file

The following is an example of a `MOP` entry in the PDW template file.

Example:

```
5 :    MOP TEXT
            'UNM'  : CW
            'LIN'  : LFM
            'TRI'  : TFM
            'SLAW' : NLFM
```

This example defines column 5 of the PDW file to contain text that specifies the `MOP` type. The text `'UNM'` translates into the `MOP` type CW which means an unmodulated pulse is generated. `'LIN'` creates a linear frequency modulated chirp. `'TRI'` creates a triangular chirp. `'SLAW'` is used to create a non-linear chirp with quadratic and cubic term.

Each `MOP` type requires a specific set of parameters, see:

- Chapter 3.2.4, "AM/FM specific parameters", on page 15
- Chapter 3.2.5, "Chirp specific parameters", on page 15
- Chapter 3.2.6, "ASK, FSK, PSK specific parameters", on page 16
- Chapter 3.2.7, "Barker specific parameters", on page 18
- Chapter 3.2.8, "Custom phase specific parameters", on page 18
- Chapter 3.2.9, "Waveform file parameters", on page 18

3.2.4 AM/FM specific parameters

Table 3-3 describes parameters that are related to the AM or FM modulation.

Table 3-3: AM/FM specific parameters

Type Tag	Units	Meaning
`MODFREQ`	HZ, KHZ MHZ, GHZ	AM, FM modulation frequency
`AMDEPTH`	PERCENT	AM modulation depth
`FMDEVIATION`	HZ, KHZ, MHZ, GHZ	FM deviation

The above parameters are mandatory if AM or FM is used. All pulses not using this `MOP` type can set the column value in the PDW list file to `'N/A'`.

3.2.5 Chirp specific parameters

Table 3-4 describes parameters that only concern chirped pulse modulation schemes (LFM, NLFM, TFM, see Table 3-2).

Table 3-4: Chirp-specific parameters

Type Tag	Units	Meaning
CHRATE	HZ/S, KHZ/S HZ/MS, HZ/US, KHZ/US	LFM, TFM chirp rate
NLFM_LIN	1	NLFM linear factor
NLFM_Q	1	NLFM quadratic factor
NLFM_C	1	NLFM cubic factor
VALUES	-	Value triples separated by "|"

The parameter CHRATE is mandatory for all linear chirps. A positive value indicates a chirp sweeping from a lower to a higher frequency.

The parameters NLFM_LIN, NLFM_Q and NLFM_C are mandatory for all non-linear NLFM chirps. The frequency within the chirp is calculated by the following equation.

$f(x) = NLFM_{LIN}*x + NLFM_Q*x^2 + NLFM_C*x^3$, with $x = -1$ to $+1$

where $x = -1 .0 + 2.0*dPFract$ and $dPFract = CurrentSample*(1/TotalSamples) - 1$

The parameter VALUES is mandatory for the piecewise linear chirp PLFM.

It is a sequence of triples separated by "|", where each triple describes a part of the piecewise linear chirp. A triple is a comma-separated sequence of values, describing the duration, the rate and the frequency offset in each part of the linear chirp. The syntax is as follows:

<duration#1>,<rate#1>,<offset#1>|...|*<duration#N>,<rate#N>,<offset#N>*, where:

- N is the number of parts of the piecewise chirp
- Triple parameters are defined in Table 3-5.

Table 3-5: PLFM-specific parameters

Triple parameters	Units	Value range	Meaning
<duration>	%	0 to 100	Duration of the chirp part
<rate, us>	Hz	- 1 GHz to 1 GHz	
<offset>	Hz	- 1 GHz to 1 GHz	Offset during the chirp part

Example:

10,4e6,10e6|20,1e6,20e6|30,1.5e6,30e6

3.2.6 ASK, FSK, PSK specific parameters

Table 3-6 describes parameters that only affect the ASK, FSK and PSK modulation schemes.

Table 3-6: ASK, FSK, PSK-specific parameters

Type Tag	Units	Meaning
CHIPRATE	HZ, KHZ MHZ, GHZ	Chip or symbol rate
STATES	1	Number of states, e.g. levels, frequencies, phases
CHIPCOUNT	1	Number of chips or symbols
PATTERN	HEX BINARY	Data bits
ASKSTEP	DB	ASK level steps
FSKSTEP	HZ, KHZ, MHZ, GHZ	FSK frequency steps
PSKSTEP	DEG	PSK phase steps

The parameters CHIPRATE, STATES, CHIPCOUNT and PATTERN are common to all MOP types.

CHIPRATE in Hz specifies the rate at which symbols occur within the pulse. This parameter is mandatory.

STATES describe how many different amplitude, frequency or phase values an individual symbol can have. If this parameter is omitted, a value of two is assumed. The maximum permissible values for this parameter are 16.

CHIPCOUNT is the number of symbols that occur within the pulse. If this number is omitted, a value of two is assumed. The maximum chip count is 4096.

The PATTERN parameter contains the data used for the MOP. In HEX format, an 8 bytes hexadecimal value is used to provide the data for up to 16 symbols with 16 states each. Each character of the hexadecimal value describes one single symbol. If this value is omitted, an alternating sequence of zeros and ones is assumed. The maximum permissible value for each digit depends on the number of states defined for the MOP. The pattern data can also be provided in binary format in which case the string must contain a series of ASCII ones and zeros.

ASKSTEP defines a step size for the ASK in dB. The resulting level attenuation depends on the symbol value according to the following equation.

$Level_{dB} = -SymVal * ASKSTEP_{dB}$

Where *SymVal* = 0 to (STATES - 1).

FSKSTEP defines the frequency step size for the FSK modulation. The individual steps of this modulation are centered symmetrically around the center frequency. For example, a binary FSK uses -FSKSTEP/2 and +FSKSTEP/2 as the frequency for symbol zero and symbol one.

PSKSTEP provides the step size for the PSK modulation. The individual phase steps are centered on 0 degrees phase shift. Thus, a phase step of 180 degrees generates a

value of -90 degrees and +90 degrees. If this parameter is omitted, a value of 180 degrees is assumed.

3.2.7 Barker specific parameters

This modulation scheme does not use any specific parameters.

3.2.8 Custom phase specific parameters

Table 3-7 describes parameters that only affect the CPH modulation scheme.

Table 3-7: CPH-specific parameters

Type Tag	Units	Meaning
VALUES	DEG	Comma-separated phase values with value range: -180 deg to 180 deg

The intervals the phases are used are distributed equidistant where the interval duration is equal to the pulse width divided by the number phase values.

Example:

30,40,135,120

3.2.9 Waveform file parameters

Requires R&S Pulse Sequencer.

Table 3-8 describes parameters that only affect waveform files.

Table 3-8: Waveform file specific parameters

Type Tag	File extension	Meaning
VALUES	*.wv	File path and file name of the waveform file

The recorded or pre-calculated IQ segment data are imported and replayed by the PDW .

Example:

C:\waveforms\1.wv

4 PDW file syntax

The general format of a PDW file is human readable ASCII text. The content of the file is interpreted by a text-based template file.

The following example shows a PDW file containing seven pulses.

Example:

```
# Pulse Sequencer PDW File
# FSK example

RF    Offset  PW    PRI Phase MOP   Rate  States Step   Syms Data
GHz   kHz     µs    ms   Deg        kHz          kHz    Hex Value
-------------------------------------------------------------------------------------------
 3.0  0.000   50.0  2.0     0.0  CW      N/A   N/A  N/A     N/A N/A
 3.0  0.000  160.0  2.0     0.0  FMOP  100.0     2  2000.0   16 0x1010101010101010
 3.0  0.000  160.0  2.0     0.0  FMOP  100.0     4  1000.0   16 0x1230001111001230
 3.0  0.000   80.0  2.0     0.0  FMOP  100.0     8   250.0    8 0x12345670
 3.0  0.000   40.0  2.0     0.0  FMOP  100.0    16   500.0    4 0xFCA5
 3.0  0.000  160.0  2.0     0.0  FMOP  100.0    16   100.0   16 0x0123456789ABCDEF
 3.0  0.000   50.0  2.0     0.0  CW      N/A   N/A    N/A   N/A  N/A
 END

Pulses ........................... 5 FSK modulated, 2 CW
Last TOA ......................... 12 ms
Center frequency ................. 3 GHz
Bandwidth ........................ 0 for the frequency hops
Level range ...................... 0 dB
```

All PDW parameters are provided in individual columns. The delimiting character as well as the decimal character are defined in the template.

Columns must provide an entry in each line. If a value is not needed for a pulse, the text `'N/A'` must be used. This text entirely avoids parsing the value at this position. This is useful if a parameter is not needed for a specific pulse entry. If a value is provided the import wizard also performs a range check. Thus, simply setting an unused parameter to zero is not always an option.

Pulses must not overlap in time. If this condition is detected the import stops and an error is generated.

5 Regular expression matching

This section describes the most important aspects of the regular expression pattern matching. For more detailed description, refer to the QRegExp class documentation of QT 5 (http://doc.qt.io/qt-5/).

Single characters

.	The dot matches any character
c	A character represents itself unless it has a special regular expression meaning
\c	A character that follows a backslash matches the character itself
\n	Matches the ASCII line feed
\r	Matches the ASCII carriage return
\t	Matches the ASCII horizontal tab
\d	Matches a decimal digit (e.g. numbers, -, +)
\D	Matches a non-digit
\s	Matches a white space character
\S	Matches a non-whitespace character
\w	Matches a word character
\W	Matches a non-word character

Sets of characters

A set of characters can be defined by enclosing the individual characters in square brackets. Writing `[abcx]` therefore matches the characters a, b, c and x.

A caret (^) negates the character set if it occurs as the first character in the square brackets. A dash (-) indicates a range of characters.

Example:

[a-z]	Matches the characters a,b,c to z
[^abc]	Matches anything but a,b,c
[0-9]	Matches the numbers 0 to 9, this is equal to \d

Quantifiers

The characters or character sets explained above are by default treated as single character. Quantifiers following a character define if characters occur more than once.

?	Matches zero or one occurrence, e.g. '`xyz?`' matches *xy* and *xyz*
+	Matches one or more occurrences, e.g. `[0-9]+` matches any length positive integer number

*	Matches zero or more occurrences
{n}	Matches exactly `n` occurrences
{n,}	Matches at least `n` occurrences
{,m}	Matches at most `m` occurrences
{n,m}	Matches at least `n` and at most `m` occurrences

To apply a quantifier to more than just the preceding character, use parentheses to group characters together in an expression. For example, use `(xyz)+` to match one or more occurrences of *xyz*.

R&S Pulse Sequencer automatically extends your regular expression (`<user-specific_regexp>`) if it is used with the `STARTS_WITH` parameter. The complete regular expression is:

`^<user_regexp>.*`

The caret marks the beginning of the string. The `.*` allows any number character following.