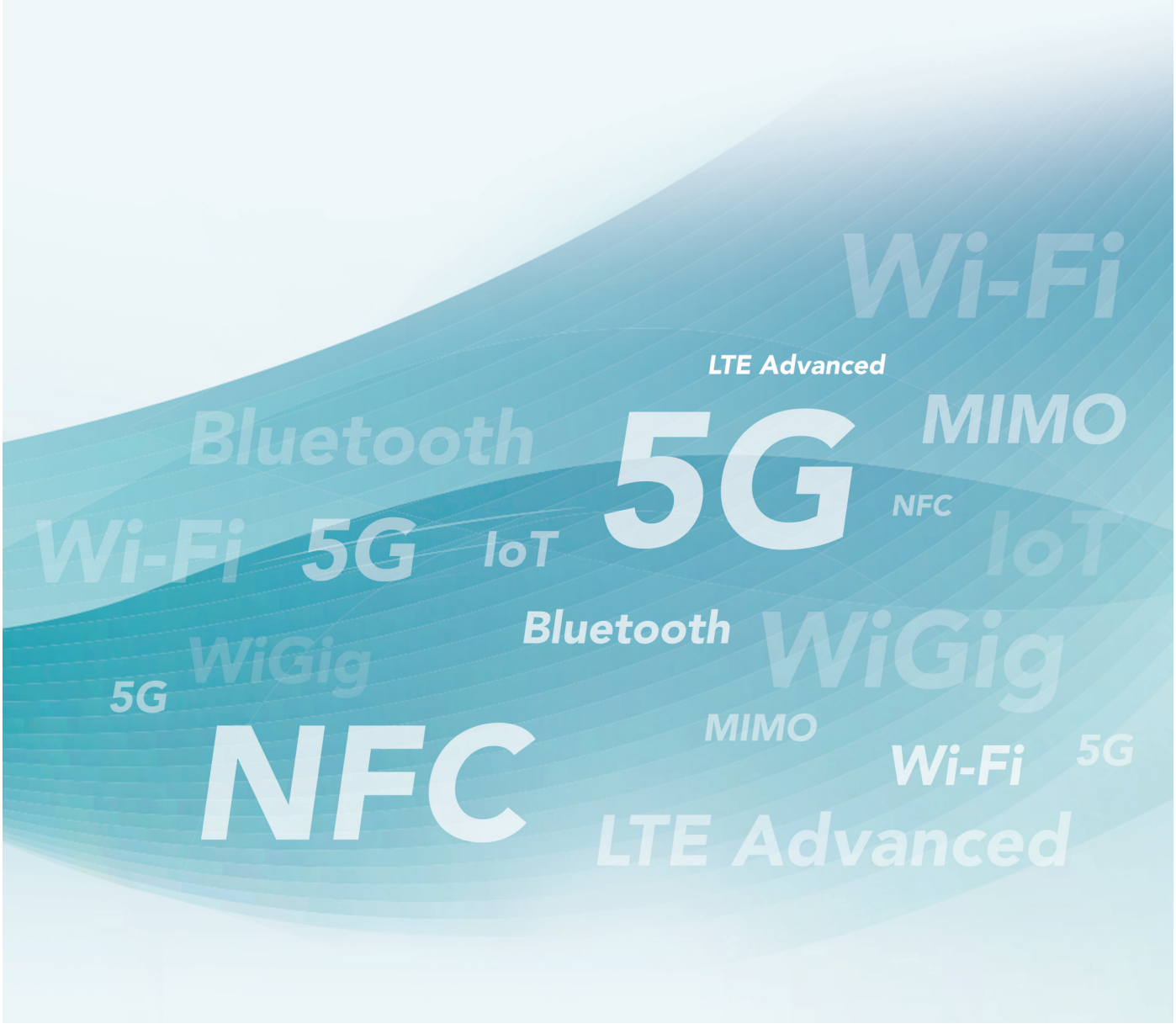
Z Series

API

Adaptive Digital Pre-Distortion API



LITEPOINT

Table of Contents

Module Index

1.1 Modules

Here is a list of all modules:

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Module Documentation

3.1 Error Codes

A reference for the status values returned by nstdpd API calls.

Typedefs

typedef enum NST_ERROR NST_ERROR

Enumerations

enum NST_ERROR {

NST_SUCCESS,

NST_ERROR_TRUE,

NST_ERROR_NULL_POINTER,

NST_ERROR_INVALID_VALUE,

NST_ERROR_WFM_INPUT_OUT_OF_BOUNDS,

NST_ERROR_CFG_FIELD_OUT_OF_BOUNDS,

NST_ERROR_CFR_MASK_ERROR,

NST_ERROR_DPD_LEVEL_ERROR,

NST_ERROR_TRAIN_SAMPLES_ERROR,

NST_ERROR_TRAIN_WFM_SAMPLES_ERROR,

NST_ERROR_INVALID_ERRCODE,

NST_ERROR_WFM_IBW_EXCEEDS_100MHZ }

Error codes/return statuses for NSTDPD API functions.

Functions

NST_ERROR nst_dpd_error_description (NST_ERROR code, const char ** description)

Query an error code's string description.

3.1.1 Detailed Description

A reference for the status values returned by nstdpd API calls.

3.1.2 Typedef Documentation

3.1.2.1 NST_ERROR

typedef enum NST_ERROR NST_ERROR

3.1.3 Enumeration Type Documentation

3.1.3.1 NST_ERROR

enum NST_ERROR

Error codes/return statuses for NSTDPD API functions.

Enumerator

NST_SUCCESS	Successful execution.
NST_ERROR_TRUE	General error status.
NST_ERROR_NULL_POINTER	Null pointer passed as input argument.
NST_ERROR_INVALID_VALUE	Infinity, NaN or other forbidden values passed as input argument.
NST_ERROR_WFM_INPUT_OUT_OF_BOUNDS	Input waveform contains sample(s) which exceeds acceptable bounds.
NST_ERROR_CFG_FIELD_OUT_OF_BOUNDS	Configuration struct contains one or more fields which exceed acceptable bounds.
NST_ERROR_CFR_MASK_ERROR	Invalid CFR mask specified.
NST_ERROR_DPD_LEVEL_ERROR	Invalid DPD performance level specified.
NST_ERROR_TRAIN_SAMPLES_ERROR	The specified number of training samples is invalid or is greater than the number of samples in the input waveforms.
NST_ERROR_TRAIN_WFM_SAMPLES_ERROR	The input waveform has too few samples.
NST_ERROR_INVALID_ERRCODE	The specified error code is invalid.
NST_ERROR_WFM_IBW_EXCEEDS_100MHZ	The iBW of the input waveform exceeds 100MHz.

3.1.4 Function Documentation

3.1.4.1 nst_dpd_error_description()

```
NST_ERROR nst_dpd_error_description (
    NST_ERROR code,
    const char ** description )
```

Query an error code's string description.

See NST_ERROR for error codes.

Parameters

code	Pass in a valid error code from the NST_ERROR enumeration.
description	Returns a pointer to a string literal describing the error.

3.2 General Module API Calls & Definitions

General-purpose enums, functions and definitions.

Functions

NST_ERROR nst_dpd_version (float *version)

NSTDPD software module version.

3.2.1 Detailed Description

General-purpose enums, functions and definitions.

3.2.2 Function Documentation

3.2.2.1 nst_dpd_version()

NST_ERROR nst_dpd_version (

float *version)

NSTDPD software module version.

Get the version number of this release of NSTDPD.

Parameters

version	Returns the version number.

3.3 Crest-Factor Reduction (CFR)

The CFR API.

Data Structures

struct NST_CFR_CONFIG_STRUCT

CFR configuration structure.

Typedefs

typedef struct NST_CFR_CONFIG_STRUCT NST_CFR_CONFIG_STRUCT

Functions

NST_ERROR nst_dpd_cfr_get_default_config (NST_CFR_CONFIG_STRUCT *config)

Default CFR configuration.

NST_ERROR nst_dpd_cfr_apply (float wfm_in_i, float wfm_in_q, int count, float wfm_out_i, float wfm_out_q,

NST_CFR_CONFIG_STRUCT config)

Apply CFR to a waveform

3.3.1 Detailed Description

The CFR API.

Before transmitting a waveform, the user may apply CFR to the waveform in order to decrease the peak-to-average-power ratio (PAPR) of the waveform.

3.3.2 Typedef Documentation

3.3.2.1 NST_CFR_CONFIG_STRUCT

typedef struct NST_CFR_CONFIG_STRUCT NST_CFR_CONFIG_STRUCT

3.3.3 Function Documentation

3.3.3.1 nst_dpd_cfr_apply()

NST_ERROR nst_dpd_cfr_apply (

float wfm_in_i,

float wfm_in_q,

int count,

float wfm_out_i,

float wfm_out_q,

NST_CFR_CONFIG_STRUCT config)

Apply CFR to a waveform.

The CFR algorithm clip-and-filters the input waveform to satisfy config.targetPAPR. The input waveform must not contain Infinity or NaN samples. All fields of the CFR configuration struct must have valid settings.

See also

NST_CFR_CONFIG_STRUCT

Parameters

wfm_in_i	The I channel of the input waveform to apply CFR to.
wfm_in_q	The Q channel of the input waveform to apply CFR to.
count	The number of IQ samples in the input and output waveforms. Must be a positive number.
wfm_out _i	Returns the I channel of the waveform with CFR applied, with the same number of samples as wfm_in.
wfm_out _q	Returns the Q channel of the waveform with CFR applied, with the same number of samples as wfm_in.
config	Configures the behavior of the CFR algorithm.

3.3.3.2 nst_dpd_cfr_get_default_config()

```
NST_ERROR nst_dpd_cfr_get_default_config (
    NST_CFR_CONFIG_STRUCT * config )
```

Default CFR configuration.

Populates the CFR configuration structure with default settings.

The 'num_bands' field defaults to 0 while the fc_list & bw_list fields default to NULL pointers. Customize these fields before using this struct as an argument to any other API function.

Parameters

config	Return the default settings structure.

3.4 Adaptive Digital Predistortion (Adaptive DPD) and Training

The Adaptive DPD API The Adaptive DPD is optimized for a particular DUT through an iterative training process.

Data Structures

struct NST_DPD_CONFIG_STRUCT

DPD configuration Structure.

Typedefs

typedef enum NST_DPD_LEVEL_ENUM NST_DPD_LEVEL_ENUM

typedef enum NST_DPD_TRAINING_ENUM NST_DPD_TRAINING_ENUM

typedef struct NST_DPD_CONFIG_STRUCT NST_DPD_CONFIG_STRUCT

Enumerations

enum NST_DPD_LEVEL_ENUM { NST_DPD_LEVEL3 = 3, NST_DPD_LEVEL2 = 2, NST_DPD_LEVEL1 = 1,
NST_DPD_LEVEL0 = 0 }

DPD Performance Levels.

enum NST_DPD_TRAINING_ENUM { NST_DPD_TRAINED = 1, NST_DPD_UNTRAINED = 0 }

DPD trained or untrained.

Functions

NST_ERROR nst_dpd_get_default_config (NST_DPD_CONFIG_STRUCT *config)

Default DPD configuration.

NST_ERROR nst_dpd_apply (float wfm_in_i, float wfm_in_q, int count, float wfm_out_i, float wfm_out_q,
NST_DPD_CONFIG_STRUCT config)

Predistort a waveform.

NST_ERROR nst_dpd_train (float wfm_dpd_i, float wfm_dpd_q, int count, float wfm_Rx_i, float wfm_Rx_q,
NST_DPD_CONFIG_STRUCT config)

Train the Adaptive DPD.

NST_ERROR nst_dpd_reset_training (NST_DPD_CONFIG_STRUCT config)

Reset DPD to untrained state.

NST_ERROR nst_dpd_query_trained (NST_DPD_TRAINING_ENUM trained)

Query DPD training state.

3.4.1 Detailed Description

The Adaptive DPD API The Adaptive DPD is optimized for a particular DUT through an iterative training process.

The DPD behavior is controlled by a set of coefficients (“DPD state”) that the DLL stores in memory. The training API call updates the DPD state each time it is executed. The user must reset the DPD state at the beginning of each test.

The minimum waveform size is 10000 samples.

3.4.2 Typedef Documentation

3.4.2.1 NST_DPD_CONFIG_STRUCT

typedef struct NST_DPD_CONFIG_STRUCT NST_DPD_CONFIG_STRUCT

3.4.2.2 NST_DPD_LEVEL_ENUM

typedef enum NST_DPD_LEVEL_ENUM NST_DPD_LEVEL_ENUM

3.4.2.3 NST_DPD_TRAINING_ENUM

typedef enum NST_DPD_TRAINING_ENUM NST_DPD_TRAINING_ENUM

3.4.3 Enumeration Type Documentation

3.4.3.1 NST_DPD_LEVEL_ENUM

enum NST_DPD_LEVEL_ENUM

DPD Performance Levels.

Adaptive DPD offers four tiers of DPD performance, which are enumerated here.

Linearizer performance & execution time increase for higher performance levels.

Enumerator

NST_DPD_LEVEL3	DPD Level 3.
NST_DPD_LEVEL2	DPD Level 2.
NST_DPD_LEVEL1	DPD Level 1.
NST_DPD_LEVEL0 DPD	Level 0.

3.4.3.2 NST_DPD_TRAINING_ENUM

enum NST_DPD_TRAINING_ENUM

DPD trained or untrained.

Enumerator

NST_DPD_TRAINED	The DPD state has been modified by the training algorithm.
NST_DPD_UNTRAINED	The DPD state has been reset to its default value. In this state the DPD output matches its input.

3.4.4 Function Documentation

3.4.4.1 nst_dpd_apply()

```
NST_ERROR nst_dpd_apply (
    float  wfm_in_i,
    float  wfm_in_q,
    int count,
    float  wfm_out_i,
    float  wfm_out_q,
    NST_DPD_CONFIG_STRUCT config )
```

Predistort a waveform.

The input waveform must not contain Infinity or NaN samples. The magnitudes of the complex waveform samples must not exceed config.abs_vsg_max. All fields of the DPD configuration struct must have valid settings.

See also
NST_DPD_CONFIG_STRUCT

Parameters

wfm_in_i	The I channel of the input waveform to apply DPD to.
wfm_in_q	The Q channel of the input waveform to apply DPD to.
count	The number of IQ samples in the input and output waveforms. Must be greater than or equal to 10000.
wfm_out_i	Returns the I channel of the waveform with DPD applied, with the same number of samples as wfm_in.
wfm_out_q	Returns the Q channel of the waveform with DPD applied, with the same number of samples as wfm_in.
config	Configures the behavior of the training algorithm.

3.4.4.2 nst_dpd_get_default_config()

```
NST_ERROR nst_dpd_get_default_config (
    NST_CFR_CONFIG_STRUCT * config )
```

Default DPD configuration.

Populates the DPD configuration structure with default settings.

Parameters

config	Return the default settings structure.

3.4.4.3 nst_dpd_query_trained()

```
NST_ERROR nst_dpd_query_trained (
    NST_DPD_TRAINING_ENUM * trained )
```

Query DPD training state.

Indicates whether the DPD algorithm has been trained, or is in an untrained state.

Parameters

trained	Returns the training state of the DPD algorithm.

See also
nst_dpd_reset_training

3.4.4.4 nst_dpd_reset_training()

```
NST_ERROR nst_dpd_reset_training (
    NST_DPD_CONFIG_STRUCT * config )
```

Reset DPD to untrained state.

This call (1) resets the DPD state to its default (untrained) value, and (2) sets the DPD performance level to the value of config.lvl. The DPD performance level must be valid.

3.4.4.5 nst_dpd_train()

```
NST_ERROR nst_dpd_train (
    float  wfm_dpd_i,
    float  wfm_dpd_q,
    int count,
    float  wfm_Rx_i,
    float  wfm_Rx_q,
    NST_DPD_CONFIG_STRUCT config )
```

Train the Adaptive DPD.

Improve DPD linearization by comparing a predistorted waveform to the baseband DUT output waveform that results from applying that predistorted waveform to the DUT.

This function updates the DPD state in memory; it has no return values. The input waveforms must not contain Infinity or NaN samples. For the wfm_dpd argument, the magnitudes of the complex waveform samples must not exceed config.abs_vsg_max. For the wfm_Rx argument, the magnitudes of the complex waveform samples must not exceed config.abs_vsa_max. All fields of the DPD configuration struct must have valid settings. config.training_samples must be at least 5000.

See also
NST_DPD_CONFIG_STRUCT

Parameters

wfm_dpd_i	The I channel of the predistorted waveform.
wfm_dpd_q	The Q channel of the predistorted waveform.
count	The number of IQ samples in the predistorted waveform and the baseband DUT output waveform. Must greater than or equal to 10000.
wfm_Rx_i	The I channel of the baseband DUT output waveform, with the same number of samples as wfm_in.
wfm_Rx_q	The Q channel of the baseband DUT output waveform, with the same number of samples as wfm_in.
config	Configures the behavior of the training algorithm.

Data Structure Documentation

4.1 NST_CFR_CONFIG_STRUCT Struct Reference

CFR configuration structure.

Data Fields

int num_bands
The number of frequency bands to apply CFR to.

float * fc_list
The center frequency (MHz) of each band.

float * bw_list
The width of each frequency band (MHz).

float offset
The frequency-domain baseband offset of the input waveform (MHz).

float f_sample
The sampling rate (MHz) of the input waveform.

float targetPAPR
The desired PAPR (dB) of the CFR output.

4.1.1 Detailed Description

CFR configuration structure.

Dictates the behavior the NSTDPD CFR algorithm. All fields must have non-Infinity, non-NaN values.

The num_bands, fc_list, and bw_list fields describe the spectral mask of the input waveform as a list of carrier bands with a certain bandwidths and center frequencies. If the offset field is nonzero, the mask described by num_bands, fc_list and bw_list will be frequency-shifted by that value. The overal spectrum mask described by the first four fields of this struct must be fully contained within the range $\left[-\frac{f\_sample}{2}, +\frac{f\_sample}{2}\right]$

4.1.2 Field Documentation

4.1.2.1 bw_list

float bw_list

The width of each frequency band (MHz).

4.1.2.2 f_sample

float f_sample

The sampling rate (MHz) of the input waveform.

Must be a positive value.

4.1.2.3 fc_list

float fc_list

The center frequency (MHz) of each band.

4.1.2.4 num_bands

int num_bands

The number of frequency bands to apply CFR to.
Must be a non-negative value.

4.1.2.5 offset

float offset

The frequency-domain baseband offset of the input waveform (MHz).

4.1.2.6 targetPAPR

float targetPAPR

The desired PAPR (dB) of the CFR output.
Must be a non-negative value.

4.2 NST_DPD_CONFIG_STRUCT Struct Reference

DPD configuration structure.

Data Fields

NST_DPD_LEVEL_ENUM lvl
Configures the performance level of the DPD algorithm.

float rho
The robustness coefficient for DPD training.

float abs_vsg_max
The absolute value of the endpoints of the VSG full-scale range, [-abs_vsg_max,abs_vsg_max].

int training_samples
The number of samples to use when training the DPD.

float f_sample
The sampling rate (MHz) of the input waveform

4.2.1 Detailed Description

DPD configuration Structure.
Dictates the behavior of the DPD/Training algorithms.

4.2.2 Field Documentation

4.2.2.1 abs_vsg_max

float abs_vsg_max

The absolute value of the endpoints of the VSG full-scale range, [-abs_vsg_max,abs_vsg_max].
Must be a positive value.

4.2.2.2 f_sample

float f_sample

The sampling rate (MHz) of the input waveform.
Must be a positive value.

4.2.2.3 lvl

NST_DPD_LEVEL_ENUM lvl

Configures the performance level of the DPD algorithm.

4.2.2.4 rho

float rho

The robustness coefficient for DPD training.
Must be a positive value.

4.2.2.5 training_samples

int training_samples

The number of samples to use when training the DPD.
Must be a positive value less than the number of samples in the waveform.

Index

RESTRICTED RIGHTS LEGEND

DISCLAIMER

LitePoint Corporation makes no representations or warranties with respect to the contents of this manual or of the associated LitePoint Corporation products, and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. LitePoint Corporation shall under no circumstances be liable for incidental or consequential damages or related expenses resulting from the use of this product, even if it has been notified of the possibility of such damages.

If you find errors or problems with this documentation, please notify LitePoint Corporation at the address listed below. LitePoint Corporation does not guarantee that this document is error-free. LitePoint Corporation reserves the right to make changes in specifications and other information contained in this document without prior notice.

TRADEMARKS

LitePoint and the LitePoint logo are registered trademarks of LitePoint Corporation. zSeries is a trademark of LitePoint Corporation. All other trademarks or registered trademarks are owned by their respective owners.

CONTACT INFORMATION

LitePoint Corporation
575 Maude Court
Sunnyvale, CA 94085-2803
United States of America

+1.866.363.1911
+1.408.456.5000

LITEPOINT TECHNICAL SUPPORT

www.litepoint.com/support

Doc: 1075-5159-001
July 2017 Rev 1