vDSP One-Dimensional Fast Fourier Transforms Reference

Performance > Carbon

2009-01-06

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

vDSP One-Dimensional Fast Fourier Transforms Reference 5

Overview 5
Functions by Task 5
- Creating and Freeing FFT Setups 5
- Computing In-Place Complex FFTs 5
- Computing Out-of-Place Complex FFTs 6
- Computing In-Place Real FFTs 7
- Computing Out-of-Place Real FFTs 7

Functions 8
- vDSP_create_fftsetup 8
- vDSP_create_fftsetupD 9
- vDSP_destroy_fftsetup 10
- vDSP_destroy_fftsetupD 10
- vDSP_fft3_zop 11
- vDSP_fft3_zopD 12
- vDSP_fft5_zop 13
- vDSP_fft5_zopD 14
- vDSP_fftm_zip 15
- vDSP_fftm_zipD 16
- vDSP_fftm_zipt 17
- vDSP_fftm_ziptD 19
- vDSP_fftm_zop 20
- vDSP_fftm_zopD 21
- vDSP_fftm_zopt 23
- vDSP_fftm_zoptD 24
- vDSP_fftm_zrip 26
- vDSP_fftm_zripD 27
- vDSP_fftm_zript 28
- vDSP_fftm_zriptD 29
- vDSP_fftm_zrop 30
- vDSP_fftm_zropD 32
- vDSP_fftm_zropt 33
- vDSP_fftm_zroptD 35
- vDSP_fft_zip 36
- vDSP_fft_zipD 37
- vDSP_fft_zipt 38
- vDSP_fft_ziptD 39
- vDSP_fft_zop 40
- vDSP_fft_zopD 41
- vDSP_fft_zopt 42
- vDSP_fft_zoptD 44

vDSP_fft_zrip 45
vDSP_fft_zripD 46
vDSP_fft_zript 47
vDSP_fft_zriptD 48
vDSP_fft_zrop 49
vDSP_fft_zropD 51
vDSP_fft_zropt 52
vDSP_fft_zroptD 53

Document Revision History 55

Index 57

vDSP One-Dimensional Fast Fourier Transforms Reference

Framework: Accelerate/vecLib

Declared in vDSP.h

Overview

This document describes the C API for performing one-dimensional Fast Fourier Transforms on an input signal. It also describes the `FFTSetup` structure which you pass as a handle to the FFT functions.

Functions by Task

Creating and Freeing FFT Setups

`vDSP_create_fftsetup` (page 8)
Builds a data structure that contains precalculated data for use by single-precision FFT functions.

`vDSP_create_fftsetupD` (page 9)
Builds a data structure that contains precalculated data for use by double-precision FFT functions.

`vDSP_destroy_fftsetup` (page 10)
Frees an existing single-precision FFT data structure.

`vDSP_destroy_fftsetupD` (page 10)
Frees an existing double-precision FFT data structure.

Computing In-Place Complex FFTs

`vDSP_fft_zip` (page 36)
Computes an in-place single-precision complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zip` (page 15)
Performs the same operation as `VDSP_fft_zip` on multiple signals with a single call.

`vDSP_fft_zipD` (page 37)
Computes an in-place double-precision complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zipD` (page 16)
Performs the same operation as `VDSP_fft_zipD` on multiple signals with a single call.

`vDSP_fft_zipt` (page 38)
Computes an in-place single-precision complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse). A buffer is used for intermediate results.

`vDSP_fftm_zipt` (page 17)
Performs the same operation as `VDSP_fft_zipt` on multiple signals with a single call.

`vDSP_fft_ziptD` (page 39)
Computes an in-place double-precision complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse). A buffer is used for intermediate results.

`vDSP_fftm_ziptD` (page 19)
Performs the same operation as VDSP_fft_ziptD on multiple signals with a single call.

Computing Out-of-Place Complex FFTs

`vDSP_fft_zop` (page 40)
Computes an out-of-place single-precision complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fft_zopD` (page 41)
Computes an out-of-place double-precision complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zop` (page 20)
Performs the same operation as `VDSP_fft_zop` on multiple signals with a single call.

`vDSP_fftm_zopD` (page 21)
Performs the same operation as `VDSP_fft_zopD` on multiple signals with a single call.

`vDSP_fft_zopt` (page 42)
Computes an out-of-place single-precision complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fft_zoptD` (page 44)
Computes an out-of-place double-precision complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zopt` (page 23)
Performs the same operation as `VDSP_fft_zopt` on multiple signals with a single call.

`vDSP_fftm_zoptD` (page 24)
Performs the same operation as `VDSP_fft_zoptD` on multiple signals with a single call.

`vDSP_fft3_zop` (page 11)
Computes an out-of-place radix-3 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 3 times the power of 2 specified by parameter `log2n`; single precision.

`vDSP_fft3_zopD` (page 12)
Computes an out-of-place radix-3 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 3 times the power of 2 specified by parameter `log2n`; double precision.

`vDSP_fft5_zop` (page 13)
Computes an out-of-place radix-5 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 5 times the power of 2 specified by parameter `log2n`; single precision.

`vDSP_fft5_zopD` (page 14)
Computes an out-of-place radix-5 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 5 times the power of 2 specified by parameter `log2n`; double precision.

Computing In-Place Real FFTs

`vDSP_fft_zrip` (page 45)
Computes an in-place single-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zrip` (page 26)
Performs the same operation as `VDSP_fft_zrip` on multiple signals with a single call.

`vDSP_fft_zripD` (page 46)
Computes an in-place double-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zripD` (page 27)
Performs the same operation as `VDSP_fft_zripD` on multiple signals with a single call.

`vDSP_fft_zript` (page 47)
Computes an in-place single-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zript` (page 28)
Performs the same operation as `VDSP_fft_zript` on multiple signals with a single call.

`vDSP_fft_zriptD` (page 48)
Computes an in-place double-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zriptD` (page 29)
Performs the same operation as `VDSP_fft_zriptD` on multiple signals with a single call.

Computing Out-of-Place Real FFTs

`vDSP_fft_zrop` (page 49)
Computes an out-of-place single-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zrop` (page 30)
Performs the same operation as `VDSP_fft_zrop` on multiple signals with a single call.

`vDSP_fft_zropD` (page 51)
: Computes an out-of-place double-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zropD` (page 32)
: Performs the same operation as `VDSP_fft_zropD` on multiple signals with a single call.

`vDSP_fft_zropt` (page 52)
: Computes an out-of-place single-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zropt` (page 33)
: Performs the same operation as `VDSP_fft_zropt` on multiple signals with a single call.

`vDSP_fft_zroptD` (page 53)
: Computes an out-of-place double-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

`vDSP_fftm_zroptD` (page 35)
: Performs the same operation as `VDSP_fft_zroptD` on multiple signals with a single call.

Functions

vDSP_create_fftsetup

Builds a data structure that contains precalculated data for use by single-precision FFT functions.

```
FFTSetup vDSP_create_fftsetup (vDSP_Length log2n, FFTRadix radix);
```

Parameters

`log2n`
: A base 2 exponent that represents the number of divisions of the complex unit circle and thus specifies the largest power of two that can be processed by a subsequent frequency-domain function. Parameter `log2n` must equal or exceed the largest power of 2 that any subsequent function processes using the weights array.

`radix`
: Specifies radix options. Radix 2, radix 3, and radix 5 functions are supported.

Return Value

Returns an `FFTSetup` structure for use with one-dimensional FFT functions. Returns 0 on error.

Discussion

This function allocates memory for an `FFTSetup` data structure needed by FFT functions, initializes that memory, and then returns it. Once prepared, the setup structure can be used repeatedly by FFT functions (which read the data in the structure and do not alter it) for any (power of two) length up to that specified when you created the structure.

If an application performs FFTs with diverse lengths, the calls with different lengths can share a single setup structure (created for the longest length), and this saves space over having multiple structures. However, in some cases, notably single-precision FFTs on 32-bit PowerPC, an FFT routine may perform faster if it is passed a setup structure that was created specifically for the length of the transform.

Parameter log2n is a base-two exponent and specifies that the largest transform length that can processed using the resulting setup structure is `2**log2n` (or `3*2**log2n` or `5*2**log2n` if the appropriate flags are passed, as discussed below). That is, the `log2n` parameter must equal or exceed the value passed to any subsequent FFT routine using the setup structure returned by this routine.

Parameter radix specifies radix options. Its value may be the bitwise OR of any combination of `FFT_RADIX2`, `FFT_RADIX3`, or `FFT_RADIX5`. The resulting setup structure may be used with any of the routines for which the respective flag was used. (The radix-3 and radix-5 FFT routines have "fft3" and "fft5" in their names. The radix-2 FFT routines have plain "fft" in their names.)

If zero is returned, the routine failed to allocate storage.

The setup structure is deallocated by calling `vDSP_destroy_fftsetup`.

Use `vDSP_create_fftsetup` during initialization. It is relatively slow compared to the routines that actually perform FFTs. Never use it in a part of an application that needs to be high performance.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_create_fftsetupD

Builds a data structure that contains precalculated data for use by double-precision FFT functions.

```
FFTSetupD vDSP_create_fftsetupD (vDSP_Length log2n, FFTRadix radix);
```

Parameters

`log2n`

A base 2 exponent that represents the number of divisions of the complex unit circle and thus specifies the largest power of two that can be processed by a subsequent frequency-domain function. Parameter `log2n` must equal or exceed the largest power of 2 that any subsequent function processes using the weights array.

`radix`

Specifies radix options. Radix 2, radix 3, and radix 5 functions are supported.

Return Value
Returns an `FFTSetupD` structure for use with FFT functions, or 0 if there was an error.

Discussion
This function allocates memory for an `FFTSetup` data structure needed by FFT functions, initializes that memory, and then returns it. Once prepared, the setup structure can be used repeatedly by FFT functions (which read the data in the structure and do not alter it) for any (power of two) length up to that specified when you created the structure.

If an application performs FFTs with diverse lengths, the calls with different lengths can share a single setup structure (created for the longest length), and this saves space over having multiple structures. However, in some cases, notably single-precision FFTs on 32-bit PowerPC, an FFT routine may perform faster if it is passed a setup structure that was created specifically for the length of the transform.

Parameter log2n is a base-two exponent and specifies that the largest transform length that can processed using the resulting setup structure is `2**log2n` (or `3*2**log2n` or `5*2**log2n` if the appropriate flags are passed, as discussed below). That is, the `log2n` parameter must equal or exceed the value passed to any subsequent FFT routine using the setup structure returned by this routine.

Parameter radix specifies radix options. Its value may be the bitwise OR of any combination of `FFT_RADIX2`, `FFT_RADIX3`, or `FFT_RADIX5`. The resulting setup structure may be used with any of the routines for which the respective flag was used. (The radix-3 and radix-5 FFT routines have "fft3" and "fft5" in their names. The radix-2 FFT routines have plain "fft" in their names.)

If zero is returned, the routine failed to allocate storage.

The setup structure is deallocated by calling `vDSP_destroy_fftsetupD`.

Use `vDSP_create_fftsetupD` during initialization. It is relatively slow compared to the routines that actually perform FFTs. Never use it in a part of an application that needs to be high performance.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_destroy_fftsetup

Frees an existing single-precision FFT data structure.

```
void vDSP_destroy_fftsetup (FFTSetup setup);
```

Parameters

`setup`

Identifies the weights array, and must point to a data structure previously created by `vDSP_create_fftsetup`

Discussion
`vDSP_destroy_fftsetup` frees an existing weights array. Any memory allocated for the array is released. After the `vDSP_destroy_fftsetup` function returns, the structure should not be used in any other functions.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_destroy_fftsetupD

Frees an existing double-precision FFT data structure.

```
void vDSP_destroy_fftsetupD (FFTSetupD setup);
```

Parameters

setup

Identifies the weights array, and must point to a data structure previously created by `vDSP_create_fftsetupD`.

Discussion

`vDSP_destroy_fftsetupD` frees an existing weights array. Any memory allocated for the array is released. After the `vDSP_destroy_fftsetupD` function returns, the structure should not be used in any other functions.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fft3_zop

Computes an out-of-place radix-3 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 3 times the power of 2 specified by parameter `log2n`; single precision.

```
void vDSP_fft3_zop (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
DSPSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Length log2n,
FFTDirection flag);
```

Parameters

setup

Use `vDSP_create_fftsetup`, to initialize this function. `FFT_RADIX3` must be specified in the call to `vDSP_create_fftsetup`. `setup` is preserved for reuse.

signal

A complex vector signal input.

signalStride

Specifies an address stride through the input vector `signal`. To process every element of the vector, specify 1 for parameter `signalStride`; to process every other element, specify 2. The value of `signalStride` should be 1 for best performance.

result

The complex vector signal output.

resultStride

Specifies an address stride for the result. The value of `resultStride` should be 1 for best performance.

log2n

The base 2 exponent of the number of elements to process in a single input signal. `log2n` must be between 3 and 15, inclusive.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

Where $N = 3(2^m)$ for Radix-3 functions, and $N = 5(2^m)$ for Radix-5 functions

If F = 1 $C_m = \text{RDFT}(A_m) \cdot 2$ If F = -1 $C_m = \text{IDFT}(A_m) \cdot N$ m = {0, N-1}

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also the FFT Limitations sections in the Target chapters of the Developer's Guide.

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft3_zopD

Computes an out-of-place radix-3 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 3 times the power of 2 specified by parameter `log2n`; double precision.

```
void vDSP_fft3_zopD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride K,
DSPDoubleSplitComplex * result,
vDSP_Stride L,
vDSP_Length log2n,
FFTDirection flag);
```

Parameters

setup

Use `vDSP_create_fftsetupD`, to initialize this function. `FFT_RADIX3` must be specified in the call to `vDSP_create_fftsetupD`. `setup` is preserved for reuse.

signal

A complex vector input.

K

Specifies an address stride through the input vector `signal`. To process every element of the vector, specify 1 for parameter `K`; to process every other element, specify 2. The value of `K` should be 1 for best performance.

result

The complex vector result.

L

Specifies an address stride for the result. The value of `L` should be 1 for best performance.

log2n

The base 2 exponent of the number of elements to process in a single input signal. `log2n` must be between 3 and 15, inclusive.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

Where $N = 3(2^m)$ for Radix-3 functions, and $N = 5(2^m)$ for Radix-5 functions

If F = 1 $C_m = \text{RDFT}(A_m) \cdot 2$ If F = -1 $C_m = \text{IDFT}(A_m) \cdot N$ m = {0, N-1}

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also the FFT Limitations sections in the Target chapters of the Developer's Guide.

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft5_zop

Computes an out-of-place radix-5 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 5 times the power of 2 specified by parameter `log2n`; single precision.

```
void vDSP_fft5_zop (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
DSPSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Length log2n,
FFTDirection flag);
```

Parameters

setup

Use `vDSP_create_fftsetup`, to initialize this function. `FFT_RADIX5` must be specified in the call to `vDSP_create_fftsetup`. `setup` is preserved for reuse.

signal

A complex vector signal input.

signalStride

Specifies an address stride through the input vector `signal`. To process every element of the vector, specify 1 for parameter `signalStride`; to process every other element, specify 2. The value of `signalStride` should be 1 for best performance.

result

The complex vector signal output.

resultStride

Specifies an address stride for the result. The value of `resultStride` should be 1 for best performance.

log2n

The base 2 exponent of the number of elements to process in a single input signal. `log2n` must be between 3 and 15, inclusive.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

Where $N = 3(2^m)$ for Radix-3 functions, and $N = 5(2^m)$ for Radix-5 functions

If F = 1 $C_m = \text{RDFT}(A_m) \cdot 2$ If F = -1 $C_m = \text{IDFT}(A_m) \cdot N$ $m = \{0, N-1\}$

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also the FFT Limitations sections in the Target chapters of the Developer's Guide.

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft5_zopD

Computes an out-of-place radix-5 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 5 times the power of 2 specified by parameter `log2n`; double precision.

```
void vDSP_fft5_zopD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride K,
DSPDoubleSplitComplex * result,
vDSP_Stride L,
vDSP_Length log2n,
FFTDirection flag);
```

Parameters

setup

Use `vDSP_create_fftsetupD`, to initialize this function. `FFT_RADIX5` must be specified in the call to `vDSP_create_fftsetupD`. `setup` is preserved for reuse.

signal

A complex vector input.

K

Specifies an address stride through the input vector `signal`. To process every element of the vector, specify 1 for parameter `K`; to process every other element, specify 2.

result

The complex vector result.

L

Specifies an address stride for the result.

log2n

The base 2 exponent of the number of elements to process in a single input signal.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

This performs the operation

Where $N = 3(2^m)$ for Radix-3 functions, and $N = 5(2^m)$ for Radix-5 functions

If F = 1 $C_m = \text{RDFT}(A_m) \cdot 2$ If F = -1 $C_m = \text{IDFT}(A_m) \cdot N$ m = {0, N-1}

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also the FFT Limitations sections in the Target chapters of the Developer's Guide.

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fftm_zip

Performs the same operation as `VDSP_fft_zip` on multiple signals with a single call.

```
void vDSP_fftm_zip (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function vDSP_create_fftsetup (page 8). The value supplied as parameter log2n of the earlier call to the setup function must equal or exceed the value supplied as parameter log2n of this transform function.

signal

A complex vector that stores the input and output signal.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter signalStride; to process every other element, specify 2.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter log2n.

numFFT

The number of signals.

flag

A forward/inverse directional flag, which must specify FFT_FORWARD for a forward transform or FFT_INVERSE for an inverse transform.

Discussion

This function allows you to perform Discrete Fourier Transforms on multiple input signals using a single call. They will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride.

The functions compute in-place complex discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_fftm_zipD

Performs the same operation as VDSP_fft_zipD on multiple signals with a single call.

```
void vDSP_fftm_zipD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetupD`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input and output signal.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter signalStride; to process every other element, specify 2.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`. The value of `log2n` must be between 2 and 12, inclusive.

numFFT

The number of signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

This function allows you to perform Discrete Fourier Transforms on multiple signals at once, using a single call. It will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride.

The function computes in-place complex discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fftm_zipt

Performs the same operation as `VDSP_fft_zipt` on multiple signals with a single call.

```
void vDSP_fftm_zipt (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPSplitComplex * temp,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input and output signal.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter signalStride; to process every other element, specify 2. The value of signalStride should be 1 for best performance.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

temp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `temp.realp` and `temp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`. The value of `log2n` must be between 2 and 12, inclusive.

numFFT

The number of different input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

The function allows you to perform Fourier transforms on a number of different input signals at once, using a single call. It can be used for efficient processing of small input signals (less than 512 points). It will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input/output vector, the parameter `signal`.

The function computes in-place complex discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fftm_ziptD

Performs the same operation as VDSP_fft_ziptD on multiple signals with a single call.

```
void vDSP_fftm_ziptD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPDoubleSplitComplex * temp,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetupD`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input and output signal.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter signalStride; to process every other element, specify 2. The value of signalStride should be 1 for best performance.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

temp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `temp.realp` and `temp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`. The value of `log2n` must be between 2 and 12, inclusive.

numFFT

The number of different input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
The function allows you to perform Fourier transforms on a number of different input signals at once, using a single call. It can be used for efficient processing of small input signals (less than 512 points). It will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input/output vector, the parameter `signal`.

The function computes in-place complex discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fftm_zop

Performs the same operation as `VDSP_fft_zop` on multiple signals with a single call.

```
void vDSP_fftm_zop (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Stride rfftStride,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup
: Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup`. The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal
: A complex vector that stores the input signal.

signalStride
: Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter signalStride; to process every other element, specify 2. The value of signalStride should be 1 for best performance.

fftStride
: The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

result
: The complex vector signal output.

resultStride

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `resultStride` should be 1 for best performance.

rfftStride

The number of elements between the first element of one result vector and the next in the output vector `result`.

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`.

numFFT

The number of input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

The function allows you to perform Discrete Fourier transforms on a number of different input signals at once, using a single call. It can be used for efficient processing of small input signals (less than 512 points). It will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input vector, `signal`, and single output vector, `result`.

The function computes out-of-place complex discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fftm_zopD

Performs the same operation as `VDSP_fft_zopD` on multiple signals with a single call.

```
void vDSP_fftm_zopD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPDoubleSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Stride rfftStride,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup` or `vDSP_create_fftsetupD`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter signalStride; to process every other element, specify 2. The value of signalStride should be 1 for best performance.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

result

The complex vector signal output.

resultStride

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `resultStride` should be 1 for best performance.

rfftStride

The number of elements between the first element of one result vector and the next in the output vector `result`.

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`. The value of `log2n` must be between 2 and 12, inclusive.

numFFT

The number of different input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

The function allows you to perform Fourier transforms on a number of different input signals at once, using a single call. It can be used for efficient processing of small input signals (less than 512 points). It will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input/output vector, the parameter `signal`.

The function computes out-of-place complex discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fftm_zopt

Performs the same operation as `VDSP_fft_zopt` on multiple signals with a single call.

```
void vDSP_fftm_zopt (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Stride rfftStride,
DSPSplitComplex * temp,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter signalStride; to process every other element, specify 2. The value of signalStride should be 1 for best performance.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

result

The complex vector signal output.

resultStride

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `resultStride` should be 1 for best performance.

rfftStride

The number of elements between the first element of one result vector and the next in the output vector `result`.

temp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `temp.realp` and `temp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`. The value of `log2n` must be between 2 and 12, inclusive.

numFFT

The number of different input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

The function allows you to perform Fourier transforms on a number of different input signals at once, using a single call. It can be used for efficient processing of small input signals (less than 512 points). It will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input/output vector, the parameter `signal`.

The function computes out-of-place complex discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fftm_zoptD

Performs the same operation as `VDSP_fft_zoptD` on multiple signals with a single call.

```
void vDSP_fftm_zoptD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPDoubleSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Stride rfftStride,
DSPDoubleSplitComplex * temp,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetupD`. The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter signalStride; to process every other element, specify 2. The value of signalStride should be 1 for best performance.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

result

The complex vector signal output.

resultStride

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2.

rfftStride

The number of elements between the first element of one result vector and the next in the output vector `result`.

temp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `temp.realp` and `temp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`. The value of `log2n` must be between 2 and 12, inclusive.

numFFT

The number of different input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
The function allows you to perform Fourier transforms on a number of different input signals at once, using a single call. It can be used for efficient processing of small input signals (less than 512 points). It will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input/output vector, the parameter `signal`.

The function computes out-of-place complex discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fftm_zrip

Performs the same operation as `VDSP_fft_zrip` on multiple signals with a single call.

```
void vDSP_fftm_zrip (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup
Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup` (page 8). The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal
A complex vector signal input.

signalStride
Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter `signalStride`; to process every other element, specify 2.

fftStride
The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

log2n
The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`.

numFFT
The number of input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
The function allows you to perform Discrete Fourier Transforms on multiple signals using a single call. They will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride.

The functions compute in-place real Discrete Fourier Transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fftm_zripD

Performs the same operation as `VDSP_fft_zripD` on multiple signals with a single call.

```
void vDSP_fftm_zripD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetupD` (page 9). The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter `signalStride`; to process every other element, specify 2.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`.

numFFT

The number of input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

The functions allow you to perform Fourier transforms on a number of different input signals at once, using a single call. They can be used for efficient processing of small input signals (less than 512 points). They will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride.

The functions compute in-place real discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fftm_zript

Performs the same operation as `VDSP_fft_zript` on multiple signals with a single call.

```
void vDSP_fftm_zript (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPSplitComplex * temp,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup` (page 8). The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter `signalStride`; to process every other element, specify 2.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

temp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `temp.realp` and `temp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`.

numFFT

The number of different input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

The functions allow you to perform Fourier transforms on a number of different input signals at once, using a single call. They can be used for efficient processing of small input signals (less than 512 points). They will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride.

The functions compute in-place real Discrete Fourier Transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fftm_zriptD

Performs the same operation as `VDSP_fft_zriptD` on multiple signals with a single call.

```
void vDSP_fftm_zriptD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPDoubleSplitComplex * temp,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetupD` (page 9). The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through the input signals. To process every element of each signal, specify 1 for parameter signalStride; to process every other element, specify 2.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

temp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `temp.realp` and `temp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process in a single input signal. For example, to process 512 elements, specify 9 for parameter `log2n`.

numFFT

The number of input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

The functions allow you to perform Fourier transforms on a number of different input signals at once, using a single call. They can be used for efficient processing of small input signals (less than 512 points). They will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride.

The functions compute in-place real Discrete Fourier Transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fftm_zrop

Performs the same operation as `VDSP_fft_zrop` on multiple signals with a single call.

```
void vDSP_fftm_zrop (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Stride rfftStride,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through input signals . To process every element of each signal, specify a stride of 1; to process every other element, specify 2.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

result

The complex vector signal output.

resultStride

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2.

rfftStride

The number of elements between the first element of one result vector and the next in the output vector `result`.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

numFFT

The number of input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

This function allows you to perform Discrete Fourier Transforms on multiple input signals using a single call. They can be used for efficient processing of small input signals (less than 512 points). They will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input vector, `signal`, and a single output vector, `result`.

The functions compute out-of-place real Discrete Fourier Transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fftm_zropD

Performs the same operation as `VDSP_fft_zropD` on multiple signals with a single call.

```
void vDSP_fftm_zropD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPDoubleSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Stride rfftStride,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

`setup`

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetupD`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

`signal`

A complex vector signal input.

`signalStride`

Specifies an address stride through input signals . To process every element of each signal, specify a stride of 1; to process every other element, specify 2. The value of `signalStride` should be 1 for best performance.

`fftStride`

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

`result`

The complex vector signal output.

`resultStride`

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `resultStride` should be 1 for best performance.

`rfftStride`

The number of elements between the first element of one result vector and the next in the output vector `result`.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

numFFT

The number of input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

This function allows you to perform Discrete Fourier Transforms on multiple input signals using a single call. They can be used for efficient processing of small input signals (less than 512 points). They will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input vector, the parameter `signal`.

The functions compute out-of-place real Discrete Fourier Transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fftm_zropt

Performs the same operation as `VDSP_fft_zropt` on multiple signals with a single call.

```
void vDSP_fftm_zropt (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Stride rfftStride,
DSPSplitComplex * temp,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through input signals . To process every element of each signal, specify a stride of 1; to process every other element, specify 2. The value of `signalStride` should be 1 for best performance.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

result

The complex vector signal output.

resultStride

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `resultStride` should be 1 for best performance.

rfftStride

The number of elements between the first element of one result vector and the next in the output vector `result`.

temp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `temp.realp` and `temp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

numFFT

The number of input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

The functions allow you to perform Fourier transforms on a number of different input signals at once, using a single call. They can be used for efficient processing of small input signals (less than 512 points). They will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input vector, the parameter `signal`.

The functions compute out-of-place real discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetup" (page 8), "vDSP_destroy_fftsetup" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fftm_zroptD

Performs the same operation as `VDSP_fft_zroptD` on multiple signals with a single call.

```
void vDSP_fftm_zroptD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
vDSP_Stride fftStride,
DSPDoubleSplitComplex * result,
vDSP_Stride resultStride,
vDSP_Stride rfftStride,
DSPDoubleSplitComplex * temp,
vDSP_Length log2n,
vDSP_Length numFFT,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetupD`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through input signals . To process every element of each signal, specify a stride of 1; to process every other element, specify 2. The value of `signalStride` should be 1 for best performance.

fftStride

The number of elements between the first element of one input signal and the first element of the next (which is also to length of each input signal, measured in elements).

result

The complex vector signal output.

resultStride

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `resultStride` should be 1 for best performance.

rfftStride

The number of elements between the first element of one result vector and the next in the output vector `result`.

temp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `temp.realp` and `temp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

numFFT

The number of different input signals.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

The functions allow you to perform Fourier transforms on a number of different input signals at once, using a single call. They can be used for efficient processing of small input signals (less than 512 points). They will work for input signals of 4 points or greater. Each of the input signals processed by a given call must have the same length and address stride. The input signals are concatenated into a single input vector, the parameter `signal`.

The functions compute out-of-place real discrete Fourier transforms of the input signals, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

See also functions "vDSP_create_fftsetupD" (page 9), "vDSP_destroy_fftsetupD" (page 10), and Chapter 2, "Using Fourier Transforms."

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fft_zip

Computes an in-place single-precision complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zip (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride stride,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup` .The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input and output signal.

stride

Specifies an address stride through the input/output vector signal. To process every element of the vector, specify 1 for parameter `stride`; to process every other element, specify 2.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

If F = 1 $C_m = \text{FDFT}(C_m)$ If F = -1 $C_m = \text{IDFT}(C_m) \cdot N$ m = {0, N-1}

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also functions "vDSP_create_fftsetup" (page 8) and "vDSP_destroy_fftsetup" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zipD

Computes an in-place double-precision complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zipD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride stride,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetupD` (page 9). The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input and output signal.

stride

Specifies an address stride through the input/output vector signal. To process every element of the vector, specify 1 for parameter `stride`; to process every other element, specify 2. The value of `stride` should be 1 for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

If F = 1 $C_m = \mathrm{FDFT}(C_m)$ If F = -1 $C_m = \mathrm{IDFT}(C_m) \cdot N$ m = {0, N-1}

$$\mathrm{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \mathrm{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also functions "vDSP_create_fftsetupD" (page 9) and "vDSP_destroy_fftsetupD" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zipt

Computes an in-place single-precision complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse). A buffer is used for intermediate results.

```
void vDSP_fft_zipt (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride stride,
DSPSplitComplex * bufferTemp,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetup`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input and output signal.

stride

Specifies an address stride through the input/output vector signal. To process every element of the vector, specify 1 for parameter `stride`; to process every other element, specify 2.

bufferTemp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `bufferTemp.realp` and `bufferTemp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

If F = 1 $C_m = \text{FDFT}(C_m)$ If F = -1 $C_m = \text{IDFT}(C_m) \cdot N$ m = {0, N-1}

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also functions "vDSP_create_fftsetup" (page 8) and "vDSP_destroy_fftsetup" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_ziptD

Computes an in-place double-precision complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse). A buffer is used for intermediate results.

```
void vDSP_fft_ziptD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride stride,
DSPDoubleSplitComplex * bufferTemp,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to the FFT weights array function `vDSP_create_fftsetupD`.The value supplied as parameter `log2n` of the earlier call to the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input and output signal.

stride

Specifies an address stride through the input/output vector signal. To process every element of the vector, specify 1 for parameter `stride`; to process every other element, specify 2.

bufferTemp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, `bufferTemp.realp` and `bufferTemp.imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

If F = 1 $C_m = \text{FDFT}(C_m)$ If F = -1 $C_m = \text{IDFT}(C_m) \cdot N$ m = {0, N-1}

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also functions "vDSP_create_fftsetupD" (page 9) and "vDSP_destroy_fftsetupD" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zop

Computes an out-of-place single-precision complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zop (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
DSPSplitComplex * result,
vDSP_Stride strideResult,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetup`. The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input and output signal.

signalStride

Specifies an address stride through input vector signal. Parameter `strideResult` specifies an address stride through output vector result. Thus, to process every element, specify a signalStride of 1; to process every other element, specify 2.

result

The complex vector signal output.

strideResult

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

$$\text{If F = 1} \quad C_m = \text{FDFT}(A_m) \qquad \text{If F = -1} \quad C_m = \text{IDFT}(A_m) \cdot N \qquad \text{m} = \{0, \text{N-1}\}$$

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also functions "vDSP_create_fftsetup" (page 8) and "vDSP_destroy_fftsetup" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zopD

Computes an out-of-place double-precision complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zopD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
DSPDoubleSplitComplex * result,
vDSP_Stride strideResult,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetup` or `vDSP_create_fftsetupD`. The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input signal.

signalStride

Specifies an address stride through input vector signal. Parameter `strideResult` specifies an address stride through output vector result. Thus, to process every element, specify a `signalStride` of 1; to process every other element, specify 2. The values of `signalStride` and `strideResult` should be 1 for best performance.

result

The complex vector signal output.

strideResult

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `strideResult` should be 1 for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

This performs the operation

If F = 1 $C_m = \mathrm{FDFT}(A_m)$ If F = -1 $C_m = \mathrm{IDFT}(A_m) \cdot N$ m = {0, N-1}

$$\mathrm{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \mathrm{IDFT}(X_m) = \frac{1}{N}\sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also functions "vDSP_create_fftsetupD" (page 9) and "vDSP_destroy_fftsetupD" (page 10).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fft_zopt

Computes an out-of-place single-precision complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zopt (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
DSPSplitComplex * result,
vDSP_Stride strideResult,
DSPSplitComplex * bufferTemp,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetup`. The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector that stores the input and output signal.

signalStride

Specifies an address stride through input vector signal. Parameter `strideResult` specifies an address stride through output vector result. Thus, to process every element, specify a signalStride of 1; to process every other element, specify 2. The values of signalStride and `strideResult` should be 1 for best performance.

result

The complex vector signal output.

strideResult

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2.

bufferTemp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, tempBuffer . `realp` and tempBuffer . `imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

This performs the operation

$$\text{If F = 1} \quad C_m = \text{FDFT}(A_m) \qquad \text{If F = -1} \quad C_m = \text{IDFT}(A_m) \cdot N \qquad \text{m = \{0, N-1\}}$$

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also functions "vDSP_create_fftsetup" (page 8) and "vDSP_destroy_fftsetup" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zoptD

Computes an out-of-place double-precision complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zoptD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
DSPDoubleSplitComplex * result,
vDSP_Stride strideResult,
DSPDoubleSplitComplex * bufferTemp,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetupD`. The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through input vector signal. Parameter `strideResult` specifies an address stride through output vector result. Thus, to process every element, specify a signalStride of 1; to process every other element, specify 2. The values of signalStride and `strideResult` should be 1 for best performance.

result

The complex vector signal output.

strideResult

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2.

bufferTemp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, tempBuffer.`realp` and tempBuffer.`imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

If F = 1 $C_m = \mathrm{FDFT}(A_m)$ If F = -1 $C_m = \mathrm{IDFT}(A_m) \cdot N$ m = {0, N-1}

$$\mathrm{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \mathrm{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

See also functions "vDSP_create_fftsetupD" (page 9) and "vDSP_destroy_fftsetupD" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zrip

Computes an in-place single-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zrip (FFTSetup setup,
DSPSplitComplex * C,
vDSP_Stride K,
vDSP_Length log2n,
FFTDirection F);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetup` (page 8). The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

C

A complex input/output vector.

K

Specifies an address stride through the input/output vector. To process every element of the vector, specify 1 for parameter `signalStride`; to process every other element, specify 2.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

F

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
Forward transforms read real input and write packed complex output. You can find more details on the packing format in vDSP Library. Inverse transforms read packed complex input and write real output. As a result of packing the frequency-domain data, time-domain data and its equivalent frequency-domain data have the same storage requirements.

If F = 1 $C_m = \text{RDFT}(C_m) \cdot 2$ If F = -1 $C_m = \text{IDFT}(C_m) \cdot N$ m = {0, N-1}

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

Real data is stored in split complex form, with odd reals stored on the imaginary side of the split complex form and even reals in stored on the real side.

See also functions "vDSP_create_fftsetup" (page 8) and "vDSP_destroy_fftsetup" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zripD

Computes an in-place double-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zripD (FFTSetupD setup,
DSPDoubleSplitComplex * C,
vDSP_Stride K,
vDSP_Length log2n,
FFTDirection F);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetupD` (page 9). The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

C

A complex vector input.

K

Specifies an address stride through the input/output vector . To process every element of the vector, specify 1 for parameter `stride`; to process every other element, specify 2.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

F

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
Forward transforms read real input and write packed complex output. Inverse transforms read packed complex input and write real output. As a result of packing the frequency-domain data, time-domain data and its equivalent frequency-domain data have the same storage requirements.

If F = 1 $C_m = \text{RDFT}(C_m) \cdot 2$ If F = -1 $C_m = \text{IDFT}(C_m) \cdot N$ $m = \{0, N-1\}$

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

Real data is stored in split complex form, with odd reals stored on the imaginary side of the split complex form and even reals in stored on the real side.

See also functions "vDSP_create_fftsetupD" (page 9) and "vDSP_destroy_fftsetupD" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zript

Computes an in-place single-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zript (FFTSetup setup,
DSPSplitComplex * C,
vDSP_Stride K,
DSPSplitComplex * bufferTemp,
vDSP_Length log2n,
FFTDirection F);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetup` (page 8). The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

C

A complex vector input.

K

Specifies an address stride through the input/output vector . To process every element of the vector, specify 1 for parameter signalStride; to process every other element, specify 2.

bufferTemp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, tempBuffer . `realp` and tempBuffer . `imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

F

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
Forward transforms read real input and write packed complex output. Inverse transforms read packed complex input and write real output. As a result of packing the frequency-domain data, time-domain data and its equivalent frequency-domain data have the same storage requirements.

If F = 1 $C_m = \text{RDFT}(C_m) \cdot 2$ If F = -1 $C_m = \text{IDFT}(C_m) \cdot N$ m = {0, N-1}

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N}\sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

Real data is stored in split complex form, with odd reals stored on the imaginary side of the split complex form and even reals in stored on the real side.

See also functions "vDSP_create_fftsetup" (page 8) and "vDSP_destroy_fftsetup" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zriptD

Computes an in-place double-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zriptD (FFTSetupD setup,
DSPDoubleSplitComplex * C,
vDSP_Stride K,
DSPDoubleSplitComplex * bufferTemp,
vDSP_Length log2n,
FFTDirection F);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetupD` (page 9). The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

C

A complex vector input.

K

Specifies an address stride through the input/output vector . To process every element of the vector, specify 1 for parameter `signalStride`; to process every other element, specify 2.

bufferTemp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, tempBuffer.`realp` and tempBuffer.`imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

F

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

Forward transforms read real input and write packed complex output. Inverse transforms read packed complex input and write real output. As a result of packing the frequency-domain data, time-domain data and its equivalent frequency-domain data have the same storage requirements.

$$\text{If F = 1} \quad C_m = \text{RDFT}(C_m) \cdot 2 \qquad \text{If F = -1} \quad C_m = \text{IDFT}(C_m) \cdot N \qquad \text{m} = \{0, \text{N-1}\}$$

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

Real data is stored in split complex form, with odd reals stored on the imaginary side of the split complex form and even reals in stored on the real side.

See also functions "vDSP_create_fftsetupD" (page 9) and "vDSP_destroy_fftsetupD" (page 10).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_fft_zrop

Computes an out-of-place single-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zrop (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
DSPSplitComplex * result,
vDSP_Stride strideResult,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetup`. The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through input vector signal. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `signalStride` should be 1 for best performance.

result

The complex vector signal output.

strideResult

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

This performs the operation

If F = 1 $C_m = \text{RDFT}(A_m) \cdot 2$ If F = -1 $C_m = \text{IDFT}(A_m) \cdot N$ $m = \{0, N\text{-}1\}$

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

Forward transforms read real input and write packed complex output (see vDSP Library for details on the packing format). Inverse transforms read packed complex input and write real output. As a result of packing the frequency-domain data, time-domain data and its equivalent frequency-domain data have the same storage requirements.

See also functions "vDSP_create_fftsetup" (page 8) and "vDSP_destroy_fftsetup" (page 10).

Availability

Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zropD

Computes an out-of-place double-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zropD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
DSPDoubleSplitComplex * result,
vDSP_Stride strideResult,
vDSP_Length log2n,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetupD`. The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through input vector signal. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `signalStride` should be 1 for best performance.

result

The complex vector signal output.

strideResult

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `strideResult` should be 1 for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

If F = 1 $C_m = \text{RDFT}(A_m) \cdot 2$ If F = -1 $C_m = \text{IDFT}(A_m) \cdot N$ m = {0, N-1}

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

Forward transforms read real input and write packed complex output. Inverse transforms read packed complex input and write real output. As a result of packing the frequency-domain data, time-domain data and its equivalent frequency-domain data have the same storage requirements. Real data is stored in split complex form, with odd reals stored on the imaginary side of the split complex form and even reals in stored on the real side.

See also functions "vDSP_create_fftsetupD" (page 9) and "vDSP_destroy_fftsetupD" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zropt

Computes an out-of-place single-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zropt (FFTSetup setup,
DSPSplitComplex * signal,
vDSP_Stride signalStride,
DSPSplitComplex * result,
vDSP_Stride strideResult,
DSPSplitComplex * bufferTemp,
vDSP_Length log2n,
FFTDirection direction);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetup`. The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through input vector signal. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `signalStride` should be 1 for best performance.

result

The complex vector signal output.

strideResult

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `strideResult` should be 1 for best performance.

bufferTemp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, tempBuffer.`realp` and tempBuffer.`imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

direction

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion
This performs the operation

$$\text{If F = 1} \quad C_m = \text{RDFT}(A_m) \cdot 2 \qquad \text{If F = -1} \quad C_m = \text{IDFT}(A_m) \cdot N \qquad \text{m} = \{0, \text{N-1}\}$$

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

Forward transforms read real input and write packed complex output. Inverse transforms read packed complex input and write real output. As a result of packing the frequency-domain data, time-domain data and its equivalent frequency-domain data have the same storage requirements. Real data is stored in split complex form, with odd reals stored on the imaginary side of the split complex form and even reals in stored on the real side.

See also functions "vDSP_create_fftsetup" (page 8) and "vDSP_destroy_fftsetup" (page 10).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_fft_zroptD

Computes an out-of-place double-precision real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

```
void vDSP_fft_zroptD (FFTSetupD setup,
DSPDoubleSplitComplex * signal,
vDSP_Stride signalStride,
DSPDoubleSplitComplex * result,
vDSP_Stride strideResult,
DSPDoubleSplitComplex * bufferTemp,
vDSP_Length log2n,
FFTDirection flag);
```

Parameters

setup

Points to a structure initialized by a prior call to FFT weights array function `vDSP_create_fftsetupD`. The value supplied as parameter `log2n` of the setup function must equal or exceed the value supplied as parameter `log2n` of this transform function.

signal

A complex vector signal input.

signalStride

Specifies an address stride through input vector signal. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `signalStride` should be 1 for best performance.

result

The complex vector signal output.

strideResult

Specifies an address stride through output vector result. Thus, to process every element, specify a stride of 1; to process every other element, specify 2. The value of `strideResult` should be 1 for best performance.

bufferTemp

A temporary vector used for storing interim results. The size of temporary memory for each part (real and imaginary) is the lower value of 4*n or 16k for best performance. Or you can simply pass the buffer of size 2^(log2n) for each part (real and imaginary). If possible, tempBuffer.`realp` and tempBuffer.`imagp` should be 32-byte aligned for best performance.

log2n

The base 2 exponent of the number of elements to process. For example, to process 1024 elements, specify 10 for parameter `log2n`.

flag

A forward/inverse directional flag, which must specify `FFT_FORWARD` for a forward transform or `FFT_INVERSE` for an inverse transform.

Discussion

This performs the operation

$$\text{If F = 1} \quad C_m = \text{RDFT}(A_m) \cdot 2 \qquad \text{If F = -1} \quad C_m = \text{IDFT}(A_m) \cdot N \qquad \text{m} = \{0, \text{N-1}\}$$

$$\text{FDFT}(X_m) = \sum_{n=0}^{N-1} X_n \cdot e^{(-j2\pi nm)/N} \qquad \text{IDFT}(X_m) = \frac{1}{N} \sum_{n=0}^{N-1} X_n \cdot e^{(j2\pi nm)/N}$$

Forward transforms read real input and write packed complex output. Inverse transforms read packed complex input and write real output. As a result of packing the frequency-domain data, time-domain data and its equivalent frequency-domain data have the same storage requirements. Real data is stored in split complex form, with odd reals stored on the imaginary side of the split complex form and even reals in stored on the real side.

See also functions "vDSP_create_fftsetupD" (page 9) and "vDSP_destroy_fftsetupD" (page 10).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

Document Revision History

This table describes the changes to *vDSP One-Dimensional Fast Fourier Transforms Reference*.

Date	Notes
2009-01-06	TBD
2008-11-19	Renamed function parameters to match formulae. Added corrections to Discussion sections.
2008-06-09	Improved discussion section and added return type information for vDSP_create_fftsetup and vDSP_create_fftsetupD.
2007-06-15	New document that describes the C API for the vDSP functionality for one-dimensional Fast Fourier Transforms.

Document Revision History

Index

V

vDSP_create_fftsetup function 8
vDSP_create_fftsetupD function 9
vDSP_destroy_fftsetup function 10
vDSP_destroy_fftsetupD function 10
vDSP_fft3_zop function 11
vDSP_fft3_zopD function 12
vDSP_fft5_zop function 13
vDSP_fft5_zopD function 14
vDSP_fftm_zip function 15
vDSP_fftm_zipD function 16
vDSP_fftm_zipt function 17
vDSP_fftm_ziptD function 19
vDSP_fftm_zop function 20
vDSP_fftm_zopD function 21
vDSP_fftm_zopt function 23
vDSP_fftm_zoptD function 24
vDSP_fftm_zrip function 26
vDSP_fftm_zripD function 27
vDSP_fftm_zript function 28
vDSP_fftm_zriptD function 29
vDSP_fftm_zrop function 30
vDSP_fftm_zropD function 32
vDSP_fftm_zropt function 33
vDSP_fftm_zroptD function 35
vDSP_fft_zip function 36
vDSP_fft_zipD function 37
vDSP_fft_zipt function 38
vDSP_fft_ziptD function 39
vDSP_fft_zop function 40
vDSP_fft_zopD function 41
vDSP_fft_zopt function 42
vDSP_fft_zoptD function 44
vDSP_fft_zrip function 45
vDSP_fft_zripD function 46
vDSP_fft_zript function 47
vDSP_fft_zriptD function 48
vDSP_fft_zrop function 49
vDSP_fft_zropD function 51
vDSP_fft_zropt function 52
vDSP_fft_zroptD function 53