

---

# vDSP Vector Scalar Arithmetic Operations Reference

[Performance > Carbon](#)



2007-06-15



Apple Inc.  
© 2007 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Carbon, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY**

**DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

---

## **vDSP Vector Scalar Arithmetic Operations Reference 5**

---

Overview 5

Functions by Task 5

Adding a Scalar to Elements of a Vector 5

Dividing Elements of a Vector by a Scalar 5

Multiplying Elements of a Vector by a Scalar 6

Multiplying Elements of a Vector by a Scalar, then Adding or Subtracting Another Scalar 6

Functions 6

vDSP\_sdiv 6

vDSP\_sdivD 7

vDSP\_vsadd 8

vDSP\_vsaddD 9

vDSP\_vsaddi 10

vDSP\_vsdiv 10

vDSP\_vsdivD 11

vDSP\_vsdivi 12

vDSP\_vsma 13

vDSP\_vsmaD 14

vDSP\_vsmsa 14

vDSP\_vsmsaD 15

vDSP\_vsmsb 16

vDSP\_vsmsbD 17

vDSP\_vsmul 18

vDSP\_vsmulD 19

vDSP\_zvdiv 19

vDSP\_zvdivD 20

vDSP\_zvsma 21

vDSP\_zvsmaD 22

vDSP\_zvzsm1 23

vDSP\_zvzsm1D 24

---

## **Document Revision History 27**

---

## **Index 29**

---



# vDSP Vector Scalar Arithmetic Operations Reference

---

**Framework:** Accelerate/vecLib  
**Declared in** vDSP.h

## Overview

Describes the C API for the vecLib functions that perform arithmetic operations combining a scalar with each element of a vector.

## Functions by Task

### Adding a Scalar to Elements of a Vector

- [vDSP\\_vsadd](#) (page 8)  
Vector scalar add; single precision.
- [vDSP\\_vsaddD](#) (page 9)  
Vector scalar add; double precision.
- [vDSP\\_vsaddi](#) (page 10)  
Integer vector scalar add.

### Dividing Elements of a Vector by a Scalar

- [vDSP\\_vsdiv](#) (page 10)  
Vector scalar divide; single precision.
- [vDSP\\_vsdivD](#) (page 11)  
Vector scalar divide; double precision.
- [vDSP\\_vsdivi](#) (page 12)  
Integer vector scalar divide.
- [vDSP\\_zvdiv](#) (page 19)  
Complex vector divide; single precision.
- [vDSP\\_zvdivD](#) (page 20)  
Complex vector divide; double precision.
- [vDSP\\_sdiv](#) (page 6)  
Divide scalar by vector; single precision.

[vDSP\\_sdivD](#) (page 7)

Divide scalar by vector; double precision.

## Multiplying Elements of a Vector by a Scalar

[vDSP\\_vsmA](#) (page 13)

Vector scalar multiply and vector add; single precision.

[vDSP\\_vsmAD](#) (page 14)

Vector scalar multiply and vector add; double precision.

[vDSP\\_zvsmA](#) (page 21)

Complex vector scalar multiply and add; single precision.

[vDSP\\_zvsmAD](#) (page 22)

Complex vector scalar multiply and add; double precision.

[vDSP\\_vsmu1](#) (page 18)

Multiplies vector `signal1` by scalar `signal2` and leaves the result in vector `result`; single precision.

[vDSP\\_vsmu1D](#) (page 19)

Multiplies vector `signal1` by scalar `signal2` and leaves the result in vector `result`; double precision.

[vDSP\\_zvzsm1](#) (page 23)

Complex vector multiply by complex scalar; single precision.

[vDSP\\_zvzsm1D](#) (page 24)

Complex vector multiply by complex scalar; double precision.

## Multiplying Elements of a Vector by a Scalar, then Adding or Subtracting Another Scalar

[vDSP\\_vsmsA](#) (page 14)

Vector scalar multiply and scalar add; single precision.

[vDSP\\_vsmsAD](#) (page 15)

Vector scalar multiply and scalar add; double precision.

[vDSP\\_vsmsB](#) (page 16)

Vector scalar multiply and vector subtract; single precision.

[vDSP\\_vsmsBD](#) (page 17)

Vector scalar multiply and vector subtract; double precision.

## Functions

### **vDSP\_sdiv**

Divide scalar by vector; single precision.

```
void vDSP_sdiv (float * A,
               float * B,
               vDSP_Stride J,
               float * C,
               vDSP_Stride K,
               vDSP_Length N);
```

**Parameters**

<i>A</i>	Single-precision real input scalar
<i>B</i>	Single-precision real input vector
<i>J</i>	Stride for <i>B</i>
<i>C</i>	Single-precision real output vector
<i>K</i>	Stride for <i>C</i>
<i>N</i>	Count

**Discussion**

This performs the operation

$$C_{nK} = \frac{A}{B_{nJ}}, \quad n = \{0, N-1\}$$

Divides scalar *A* by each element of vector *B*, storing the results in vector *C*.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_sdivD**

Divide scalar by vector; double precision.

```
void vDSP_sdivD (double * A,
                double * B,
                vDSP_Stride J,
                double * C,
                vDSP_Stride K,
                vDSP_Length N);
```

**Parameters**

<i>A</i>	Double-precision real input scalar
<i>B</i>	Double-precision real input vector

<i>J</i>	Stride for B
<i>C</i>	Double-precision real output vector
<i>K</i>	Stride for C
<i>N</i>	Count

**Discussion**

This performs the operation

$$C_{nK} = \frac{A}{B_{nJ}}, \quad n = \{0, N-1\}$$

Divides scalar A by each element of vector B, storing the results in vector C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsadd**

Vector scalar add; single precision.

```
void vDSP_vsadd (float * A,
                vDSP_Stride I,
                float * B,
                float * C,
                vDSP_Stride K,
                vDSP_Length N);
```

**Parameters**

<i>A</i>	Single-precision real input vector
<i>I</i>	Stride for A
<i>B</i>	Single-precision real input scalar
<i>C</i>	Single-precision real output vector
<i>K</i>	Stride for C
<i>N</i>	Count

**Discussion**

Performs the operation



$$C_{nK} = A_{nI} + B \quad n = \{0, N-1\}$$

Adds scalar *B* to each element of vector *A* and stores the result in the corresponding element of vector *C*.

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

vDSP.h

### vDSP\_vsaddD

Vector scalar add; double precision.

```
void vDSP_vsaddD (double * A,
                 vDSP_Stride I,
                 double * B,
                 double * C,
                 vDSP_Stride K,
                 vDSP_Length N);
```

#### Parameters

<i>A</i>	Double-precision real input vector
<i>I</i>	Stride for <i>A</i>
<i>B</i>	Double-precision real input scalar
<i>C</i>	Double-precision real output vector
<i>K</i>	Stride for <i>C</i>
<i>N</i>	Count

#### Discussion

Performs the operation

$$C_{nK} = A_{nI} + B \quad n = \{0, N-1\}$$

Adds scalar *B* to each element of vector *A* and stores the result in the corresponding element of vector *C*.

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

vDSP.h

**vDSP\_vsaddi**

Integer vector scalar add.

```
void vDSP_vsaddi (int * A,
                 vDSP_Stride I,
                 int * B,
                 int * C,
                 vDSP_Stride K,
                 vDSP_Length N);
```

**Parameters**

<i>A</i>	Integer input vector
<i>I</i>	Stride for A
<i>B</i>	Integer input scalar
<i>C</i>	Integer output vector
<i>K</i>	Stride for C
<i>N</i>	Count

**Discussion**

Performs the operation

$$C_{nK} = A_{nI} + B \quad n = \{0, N-1\}$$

Adds scalar *B* to each element of vector *A* and stores the result in the corresponding element of vector *C*.**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsdiv**

Vector scalar divide; single precision.

```
void vDSP_vsdiv (float * A,
                vDSP_Stride I,
                float * B,
                float * C,
                vDSP_Stride K,
                vDSP_Length N);
```

**Parameters**

<i>A</i>	Single-precision real input vector
----------	------------------------------------

<i>I</i>	Stride for A
<i>B</i>	Single-precision real input scalar
<i>C</i>	Single-precision real output vector
<i>K</i>	Stride for C
<i>N</i>	Count

**Discussion**

Performs the operation

$$C_{nK} = \frac{A_{nI}}{B} \quad n = \{0, N-1\}$$

Divides each element of vector A by scalar B and stores the result in the corresponding element of vector C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsdivD**

Vector scalar divide; double precision.

```
void vDSP_vsdivD (double * A,
                 vDSP_Stride I,
                 double * B,
                 double * C,
                 vDSP_Stride K,
                 vDSP_Length N);
```

**Parameters**

<i>A</i>	Double-precision real input vector
<i>I</i>	Stride for A
<i>B</i>	Double-precision real input scalar
<i>C</i>	Double-precision real output vector
<i>K</i>	Stride for C
<i>N</i>	Count

**Discussion**

Performs the operation

$$C_{nK} = \frac{A_{nI}}{B} \quad n = \{0, N-1\}$$

Divides each element of vector *A* by scalar *B* and stores the result in the corresponding element of vector *C*.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsdivi**

Integer vector scalar divide.

```
void vDSP_vsdivi (int * A,
                 vDSP_Stride I,
                 int * B,
                 int * C,
                 vDSP_Stride K,
                 vDSP_Length N);
```

**Parameters**

<i>A</i>	Integer input vector
<i>I</i>	Stride for <i>A</i>
<i>B</i>	Integer input scalar
<i>C</i>	Integer output vector
<i>K</i>	Stride for <i>C</i>
<i>N</i>	Count

**Discussion**

Performs the operation

$$C_{nK} = \frac{A_{nI}}{B} \quad n = \{0, N-1\}$$

Divides each element of vector *A* by scalar *B* and stores the result in the corresponding element of vector *C*.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsma**

Vector scalar multiply and vector add; single precision.

```
void vDSP_vsma (const float * A,
               vDSP_Stride I,
               const float * B,
               const float * C,
               vDSP_Stride K,
               float * D,
               vDSP_Stride L,
               vDSP_Length N);
```

**Parameters**

<i>A</i>	Single-precision real input vector
<i>I</i>	Stride for A
<i>B</i>	Single-precision real input scalar
<i>C</i>	Single-precision real input vector
<i>K</i>	Stride for C
<i>D</i>	Single-precision real output vector
<i>L</i>	Stride for D
<i>N</i>	Count

**Discussion**

Performs the operation

$$D_{nM} = A_{nI} \cdot B + C_{nK} \quad n = \{0, N-1\}$$

Multiplies vector A by scalar B and then adds the products to vector C. Results are stored in vector D.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsmaD**

Vector scalar multiply and vector add; double precision.

```
void vDSP_vsmaD (const double * A,
                vDSP_Stride I,
                const double * B,
                const double * C,
                vDSP_Stride K,
                double * D,
                vDSP_Stride L,
                vDSP_Length N);
```

**Parameters**

<i>A</i>	Double-precision real input vector
<i>I</i>	Stride for A
<i>B</i>	Double-precision real input scalar
<i>C</i>	Double-precision real input vector
<i>K</i>	Stride for C
<i>D</i>	Double-precision real output vector
<i>L</i>	Stride for D
<i>N</i>	Count

**Discussion**

Performs the operation

$$D_{nM} = A_{nI} \cdot B + C_{nK} \quad n = \{0, N-1\}$$

Multiplies vector A by scalar B and then adds the products to vector C. Results are stored in vector D.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsmsa**

Vector scalar multiply and scalar add; single precision.

```
void vDSP_vsmsa (float * A,
                vDSP_Stride I,
                float * B,
                float * C,
                float * D,
                vDSP_Stride L,
                vDSP_Length N);
```

**Parameters**

<i>A</i>	Single-precision real input vector
<i>I</i>	Stride for A
<i>B</i>	Single-precision real input scalar
<i>C</i>	Single-precision real input scalar
<i>D</i>	Single-precision real output vector
<i>L</i>	Stride for D
<i>N</i>	Count

**Discussion**

Performs the operation

$$D_{nM} = A_{nI} \cdot B + C \quad n = \{0, N-1\}$$

Multiplies vector A by scalar B and then adds scalar C to each product. Results are stored in vector D.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsmsaD**

Vector scalar multiply and scalar add; double precision.

```
void vDSP_vsmsaD (double * A,
                 vDSP_Stride I,
                 double * B,
                 double * C,
                 double * D,
                 vDSP_Stride L,
                 vDSP_Length N);
```

**Parameters**

<i>A</i>	Double-precision real input vector
<i>I</i>	Stride for A
<i>B</i>	Double-precision real input scalar
<i>C</i>	Double-precision real input scalar
<i>D</i>	Double-precision real output vector
<i>L</i>	Stride for D
<i>N</i>	Count

**Discussion**

Performs the operation

$$D_{nM} = A_{nI} \cdot B + C \quad n = \{0, N-1\}$$

Multiplies vector A by scalar B and then adds scalar C to each product. Results are stored in vector D.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsmsb**

Vector scalar multiply and vector subtract; single precision.



```
void vDSP_vsmsb (float * A,
                vDSP_Stride I,
                float * B,
                float * C,
                vDSP_Stride K,
                float * D,
                vDSP_Stride L,
                vDSP_Length N);
```

**Parameters**

<i>A</i>	Single-precision real input vector
<i>I</i>	Stride for A
<i>B</i>	Single-precision real input scalar
<i>C</i>	Single-precision real input vector
<i>K</i>	Stride for C
<i>D</i>	Single-precision real output vector
<i>L</i>	Stride for D
<i>N</i>	Count

**Discussion**

Performs the operation

$$D_{nM} = A_{nI} \cdot B - C_{nK} \quad n = \{0, N-1\}$$

Multiplies vector A by scalar B and then subtracts vector C from the products. Results are stored in vector D.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsmsbD**

Vector scalar multiply and vector subtract; double precision.

```
void vDSP_vsmsbD (double * A,
                 vDSP_Stride I,
                 double * B,
                 double * C,
                 vDSP_Stride K,
                 double * D,
                 vDSP_Stride L,
                 vDSP_Length N);
```

**Parameters**

<i>A</i>	Double-precision real input vector
<i>I</i>	Stride for A
<i>B</i>	Double-precision real input scalar
<i>C</i>	Double-precision real input vector
<i>K</i>	Stride for C
<i>D</i>	Double-precision real output vector
<i>L</i>	Stride for D
<i>N</i>	Count

**Discussion**

Performs the operation

$$D_{nM} = A_{nI} \cdot B - C_{nK} \quad n = \{0, N-1\}$$

Multiplies vector *A* by scalar *B* and then subtracts vector *C* from the products. Results are stored in vector *D*.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsmul**

Multiplies vector *signal1* by scalar *signal2* and leaves the result in vector *result*; single precision.

```
void vDSP_vsmul (const float input1[],
                vDSP_Stride stride1,
                const float * input2,
                float result[],
                vDSP_Stride strideResult,
                vDSP_Length size);
```

**Discussion**

This performs the operation

$$C_{nK} = A_{nI} \cdot B \quad n = \{0, N-1\}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_vsmulD**

Multiplies vector `signal1` by scalar `signal2` and leaves the result in vector `result`; double precision.

```
void vDSP_vsmulD (const double input1[],
                 vDSP_Stride stride1,
                 const double * input2,
                 double result[],
                 vDSP_Stride strideResult,
                 vDSP_Length size);
```

**Discussion**

This performs the operation

$$C_{nK} = A_{nI} \cdot B \quad n = \{0, N-1\}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zvdiv**

Complex vector divide; single precision.

```
void vDSP_zvdiv (DSPSplitComplex * A,
                vDSP_Stride I,
                DSPSplitComplex * B,
                vDSP_Stride J,
                DSPSplitComplex * C,
                vDSP_Stride K,
                vDSP_Length N);
```

**Parameters**

<i>A</i>	Single-precision complex input vector
<i>I</i>	Stride for A
<i>B</i>	Single-precision complex input vector
<i>J</i>	Stride for B
<i>C</i>	Single-precision complex output vector
<i>K</i>	Stride for C
<i>N</i>	Count

**Discussion**

Divides vector B by vector A.

$$C_{nK} = \frac{B_{nJ}}{A_{nI}} \quad n = \{0, N-1\}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zvdivD**

Complex vector divide; double precision.

```
void vDSP_zvdivD (DSPDoubleSplitComplex * A,
                 vDSP_Stride I,
                 DSPDoubleSplitComplex * B,
                 vDSP_Stride J,
                 DSPDoubleSplitComplex * C,
                 vDSP_Stride K,
                 vDSP_Length N);
```

**Parameters**

<i>A</i>	Double-precision complex input vector
<i>I</i>	Stride for A
<i>B</i>	Double-precision complex input vector
<i>J</i>	Stride for B
<i>C</i>	Double-precision complex output vector
<i>K</i>	Stride for C
<i>N</i>	Count

**Discussion**

Divides vector B by vector A.

$$C_{nK} = \frac{B_{nJ}}{A_{nI}} \quad n = \{0, N-1\}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zvsma**

Complex vector scalar multiply and add; single precision.

```
void vDSP_zvsmA (DSPSplitComplex * A,
vDSP_Stride I,
DSPSplitComplex * B,
DSPSplitComplex * C,
vDSP_Stride K,
DSPSplitComplex * D,
vDSP_Stride L,
vDSP_Length N);
```

**Parameters**

<i>A</i>	Single-precision complex input vector
<i>I</i>	Stride for A
<i>B</i>	Single-precision complex input scalar
<i>C</i>	Single-precision real input vector
<i>K</i>	Stride for C
<i>D</i>	Single-precision real output vector
<i>L</i>	Stride for D
<i>N</i>	Count

**Discussion**

Multiplies vector A by scalar B and add the products to vector C. The result is stored in vector D.

$$D_{nL} = A_{nI}B + C_{nK}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zvsmAD**

Complex vector scalar multiply and add; double precision.

```
void vDSP_zvsmad (DSPDoubleSplitComplex * A,
                 vDSP_Stride I,
                 DSPDoubleSplitComplex * B,
                 DSPDoubleSplitComplex * C,
                 vDSP_Stride K,
                 DSPDoubleSplitComplex * D,
                 vDSP_Stride L,
                 vDSP_Length N);
```

**Parameters**

<i>A</i>	Double-precision complex input vector
<i>I</i>	Stride for A
<i>B</i>	Double-precision complex input scalar
<i>C</i>	Double-precision real input vector
<i>K</i>	Stride for C
<i>D</i>	Double-precision real output vector
<i>L</i>	Stride for C
<i>N</i>	Count

**Discussion**

Multiplies vector A by scalar B and add the products to vector C. The result is stored in vector D.

$$D_{nL} = A_{nI}B + C_{nK}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zvzsmi**

Complex vector multiply by complex scalar; single precision.

```
void vDSP_zvzsm1 (DSPSplitComplex * A,
                 vDSP_Stride I,
                 DSPSplitComplex * B,
                 DSPSplitComplex * C,
                 vDSP_Stride K,
                 vDSP_Length N);
```

**Parameters**

<i>A</i>	Single-precision complex input vector
<i>I</i>	Stride for A
<i>B</i>	Single-precision complex input scalar
<i>C</i>	Single-precision complex output vector
<i>K</i>	Stride for C
<i>N</i>	Count

**Discussion**

This performs the operation

$$C_{nK} = A_{nI}B$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zvzsm1D**

Complex vector multiply by complex scalar; double precision.

```
void vDSP_zvzsm1 (DSPSplitComplex * A,
                 vDSP_Stride I,
                 DSPSplitComplex * B,
                 DSPSplitComplex * C,
                 vDSP_Stride K,
                 vDSP_Length N);
```

**Parameters**

<i>A</i>	Double-precision complex input vector
<i>I</i>	Stride for A
<i>B</i>	Double-precision complex input scalar



*C*  
Double-precision complex output vector

*K*  
Stride for *C*

*N*  
Count

**Discussion**

This performs the operation

$$C_{nK} = A_{nI}B$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h



# Document Revision History

---

This table describes the changes to *vDSP Vector Scalar Arithmetic Operations Reference*.

Date	Notes
2007-06-15	New document that describes the C API for the vDSP functions that perform arithmetic operations combining a scalar with each element of a vector

## REVISION HISTORY

### Document Revision History

# Index

---

## V

---

[vDSP\\_svdiv function](#) 6  
[vDSP\\_svdivD function](#) 7  
[vDSP\\_vsadd function](#) 8  
[vDSP\\_vsaddD function](#) 9  
[vDSP\\_vsaddi function](#) 10  
[vDSP\\_vsdiv function](#) 10  
[vDSP\\_vsdivD function](#) 11  
[vDSP\\_vsdivi function](#) 12  
[vDSP\\_vsmma function](#) 13  
[vDSP\\_vsmmaD function](#) 14  
[vDSP\\_vmsma function](#) 14  
[vDSP\\_vmsmaD function](#) 15  
[vDSP\\_vmsmb function](#) 16  
[vDSP\\_vmsmbD function](#) 17  
[vDSP\\_vsmul function](#) 18  
[vDSP\\_vsmulD function](#) 19  
[vDSP\\_zvdiv function](#) 19  
[vDSP\\_zvdivD function](#) 20  
[vDSP\\_zvsma function](#) 21  
[vDSP\\_zvsmaD function](#) 22  
[vDSP\\_zvzsm1 function](#) 23  
[vDSP\\_zvzsm1D function](#) 24