

# CSI NN 接口说明手册

Release v1.1.x

2022 年 01 月 18 日

## **Copyright © 2021 平头哥半导体有限公司，保留所有权利。**

本文档的产权属于平头哥半导体有限公司(下称“平头哥”)。本文档仅能分布给:(i) 拥有合法雇佣关系，并需要本文档的信息的平头哥员工，或(ii) 非平头哥组织但拥有合法合作关系，并且其需要本文档的信息的合作方。对于本文档，禁止任何在专利、版权或商业秘密过程中，授予或暗示的可以使用该文档。在没有得到平头哥半导体有限公司的书面许可前，不得复制本文档的任何部分，传播、转录、储存在检索系统中或翻译成任何语言或计算机语言。

### **商标申明**

平头哥的 LOGO 和其它所有商标归平头哥半导体有限公司及其关联公司所有，未经平头哥半导体有限公司的书面同意，任何法律实体不得使用平头哥的商标或者商业标识。

### **注意**

您购买的产品、服务或特性等应受平头哥商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，平头哥对本文档内容不做任何明示或默示的声明或保证。由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。平头哥半导体有限公司不对任何第三方使用本文档产生的损失承担任何法律责任。

## **Copyright © 2021 T-HEAD Semiconductor Co.,Ltd. All rights reserved.**

This document is the property of T-HEAD Semiconductor Co.,Ltd. This document may only be distributed to: (i) a T-HEAD party having a legitimate business need for the information contained herein, or (ii) a non-T-HEAD party having a legitimate business need for the information contained herein. No license, expressed or implied, under any patent, copyright or trade secret right is granted or implied by the conveyance of this document. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise without the prior written permission of T-HEAD Semiconductor Co.,Ltd.

### **Trademarks and Permissions**

The T-HEAD Logo and all other trademarks indicated as such herein are trademarks of Hangzhou T-HEAD Semiconductor Co.,Ltd. All other products or service names are the property of their respective owners.

### **Notice**

The purchased products, services and features are stipulated by the contract made between T-HEAD and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied. The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

平头哥半导体有限公司 T-HEAD Semiconductor Co.,LTD

地址: 杭州市余杭区向往街 1122 号欧美金融城(EFC) 英国中心西楼 T6

邮编: 311121

网址: [www.t-head.cn](http://www.t-head.cn)

# Contents:

<b>第一章 CSI NN 软件库 (DSP2 版本)</b>	<b>1</b>
1.1 简介 . . . . .	1
1.2 如何使用库 . . . . .	1
1.3 示例 . . . . .	2
<b>第二章 激活函数</b>	<b>3</b>
2.1 近似值激活函数 . . . . .	3
2.2 向量绝对值最大值 . . . . .	5
<b>第三章 卷积函数</b>	<b>6</b>
3.1 卷积 . . . . .	6
3.2 卷积 (RGB 图形版本) . . . . .	9
3.3 1x1 卷积 . . . . .	10
3.4 可分离卷积 . . . . .	11
<b>第四章 全连接层函数</b>	<b>12</b>
4.1 全连接函数 . . . . .	12
<b>第五章 池化函数</b>	<b>14</b>
5.1 最大值池化 . . . . .	14
5.2 平均值池化 . . . . .	16
<b>第六章 softmax 函数</b>	<b>17</b>
6.1 softmax . . . . .	17
<b>第七章 辅助函数</b>	<b>19</b>
7.1 向量乘法 . . . . .	19
7.2 数据转换 . . . . .	21

# 第一章 CSI NN 软件库 (DSP2 版本)

## 1.1 简介

这份手册描述的 CSI NN 软件库的 DSP2 版本, 是一些用于 CK804/E804 设备的, 使用了 DSP2 指令集加速的神经网络基本单元的集合。

库内的各个函数可以分类为:

- 激活函数
- 卷积函数
- 全连接函数
- 池化函数
- softmax 函数
- 辅助函数

库内的大多数函数都有 Q7, Q15 两种版本。

## 1.2 如何使用库

Lib 目录内提供了一些预编译的版本, 分别是:

- libcsky\_CK804e\_dsp2\_nn.a
- libcsky\_E804d\_dsp2\_nn.a

函数库的函数声明在头文件 `csky_dsp2_nnfunctions.h` 中, `csky_dsp2_nnfunctions.h` 源码放在 Include 目录中。

在应用程序中包括 `csky_dsp2_nnfunctions.h` 头文件, 就可以直接调用 NN 库函数; 链接的时候指定应用对应的库版本, 就可以将库函数链接进应用程序。

**注意** 原来以 `csky_dsp2_` 开头的命名方式将被以 `csi_` 开头的方式取代, 并且, 不同版本的库都会使用这种命名方式。  
因此, 推荐使用以 `csi` 的开头的命名方式,

以 `csky_dsp2_` 开头的命名方式在下一个版本不再支持。

## 1.3 示例

CSI SDK 里面带了一些示例演示如何使用库函数。

# 第二章 激活函数

## 2.1 近似值激活函数

### 2.1.1 函数

- `csky_dsp2_activations_direct_q15` : Q15 近似值激活函数。
- `csky_dsp2_activations_direct_q7` : Q7 近似值激活函数。

### 2.1.2 简要说明

用查表的方式求取激活函数的近似值。

支持 sigmod 和 tanh，根据参数可选值 CSKY\_SIGMOID 和 CSKY\_TANH 区分。

为 Q7, Q15 每种类型都提供了不同的函数。

### 2.1.3 函数说明

#### 2.1.3.1 `csky_dsp2_activations_direct_q15`

```
void csky_dsp2_activations_direct_q15 (q15_t *data, uint16_t size, uint16_t int_width,  
→ csky_nn_activation_type type)
```

**参数:**

\*`data`: 指向输入的数值  
`size`: 输入元素的个数  
`int_width`: 每个定点数元素的整数部分位数  
`type`: 选择要使用的激活函数

**返回值:**

无

**缩放和溢出时的行为:**

函数实现上使用了查表的方式求取激活函数的近似值。

函数的输入假定数值小于等于 3。大于 3 的输入值在意义不大，可以直接用饱和近似求解。

### 2.1.3.2 csky\_dsp2\_activations\_direct\_q7

```
void csky_dsp2_activations_direct_q7 (q7_t *data, uint16_t size, uint16_t int_width,  
→csky_nn_activation_type type)
```

参数:

\*data: 指向输入的数值  
size: 输入元素的个数  
int\_width: 每个定点数元素的整数部分位数  
type: 选择要使用的激活函数

返回值:

无

缩放和溢出时的行为:

函数实现上使用了查表的方式求取激活函数的近似值。  
函数的输入假定数值小于等于 3。大于 3 的输入值在意义不大，可以直接用饱和近似求解。

## 2.2 向量绝对值最大值

### 2.2.1 函数

- `csky_dsp2_relu_q15` : Q15 relu。
- `csky_dsp2_relu_q7` : Q7 relu。

### 2.2.2 简要说明

求解输入的 relu 值。

为 Q7, Q15 每种类型都提供了不同的函数。

### 2.2.3 函数说明

#### 2.2.3.1 `csky_dsp2_relu_q15`

```
void csky_dsp2_relu_q15 (q15_t *data, uint16_t size)
```

参数:

`*data`: 指向输入元素  
`size`: 输入的元素个数

返回值:

无

#### 2.2.3.2 `csky_dsp2_relu_q7`

```
void csky_dsp2_relu_q7 (q7_t *data, uint16_t size)
```

参数:

`*data`: 指向输入元素  
`size`: 输入的元素个数

返回值:

无

# 第三章 卷积函数

这部分提供了卷积层所需要的基本函数。

实现上，卷积基本分为了 im2col 和 GEMM 两步。

im2col 将需要的数据展开成列，然后卷积操作就转换成了矩阵与矩阵的乘法（GEMM）操作。

为了减少内存占用，实际实现上，im2col 每次只展开部分数据，在 GEMM 计算完成后再展开后续。

## 3.1 卷积

### 3.1.1 函数

- `csky_dsp2_convolve_HWC_q15_basic` : Q15 卷积
- `csky_dsp2_convolve_HWC_q7_basic` : Q7 卷积

### 3.1.2 简要说明

通用的卷积实现，适应各种张量大小和卷积核。

加速版本需要调整输入。

为 Q15 和 Q7 类型都提供了不同的函数。

### 3.1.3 函数说明

#### 3.1.3.1 `csky_dsp2_convolve_HWC_q15_basic`

```
void csky_dsp2_convolve_HWC_q15_basic (const q15_t *Im_in, const uint16_t dim_im_in,
const uint16_t ch_im_in, const q15_t *wt, const uint16_t ch_im_out, const uint16_t_
dim_kernel, const uint16_t padding, const uint16_t stride, const q15_t *bias, const_
uint16_t bias_shift, const uint16_t out_shift, q15_t *Im_out, const uint16_t dim_im_
out, q15_t *bufferA)
```

参数:

\*Im\_in: 指向输入的张量  
dim\_im\_in: 输入张量的维度  
ch\_im\_in: 输入张量的通道数量  
\*wt: 指向卷积核  
ch\_im\_out: 输出张量的通道数  
dim\_kernel: 卷积核大小  
padding: 填充的大小  
stride: 步进的大小  
\*bias: 偏差值  
bias\_shift: 偏差值需要的左移位数  
out\_shift: 输出结果需要的右移位数  
\*Im\_out: 指向输出张量  
dim\_im\_out: 输出张量的维度  
\*bufferA: 临时区间

返回值:

无

临时区间大小:

需要的临时区间大小:  $2 * \text{ch\_im\_in} * \text{dim\_kernel} * \text{dim\_kernel}$

### 3.1.3.2 csky\_dsp2\_convolve\_HWC\_q7\_basic

```
void csky_dsp2_convolve_HWC_q7_basic (const q7_t *Im_in, const uint16_t dim_im_in,
const uint16_t ch_im_in, const q7_t *wt, const uint16_t ch_im_out, const uint16_t dim_
kernel, const uint16_t padding, const uint16_t stride, const q7_t *bias, const_
uint16_t bias_shift, const uint16_t out_shift, q7_t *Im_out, const uint16_t dim_im_
out, q15_t *bufferA)
```

参数:

\*Im\_in: 指向输入的张量  
dim\_im\_in: 输入张量的维度  
ch\_im\_in: 输入张量的通道数量  
\*wt: 指向卷积核  
ch\_im\_out: 输出张量的通道数  
dim\_kernel: 卷积核大小  
padding: 填充的大小  
stride: 步进的大小  
\*bias: 偏差值  
bias\_shift: 偏差值需要的左移位数  
out\_shift: 输出结果需要的右移位数

\*Im\_out: 指向输出张量

dim\_im\_out: 输出张量的维度

\*bufferA: 临时区间

**返回值:**

无

**临时区间大小:**

需要的临时区间大小:  $2 * \text{ch\_im\_in} * \text{dim\_kernel} * \text{dim\_kernel}$

## 3.2 卷积 (RGB 图形版本)

### 3.2.1 函数

- `csky_dsp2_convolve_HWC_q7_RGB`: Q7 卷积, RGB 图形版本

### 3.2.2 简要说明

函数是为处理 RGB 图形卷积的优化实现, 可以用在 CNN 等网络的第一层。

暂时只支持 q7 版本。

### 3.2.3 函数说明

#### 3.2.3.1 `csky_dsp2_convolve_HWC_q7_RGB`

```
void csky_dsp2_convolve_HWC_q7_RGB (const q15_t *Im_in, const uint16_t dim_im_in,  
const q15_t *wt, const uint16_t ch_im_out, const uint16_t dim_kernel, const uint16_t  
padding, const uint16_t stride, const q15_t *bias, const uint16_t bias_shift, const  
uint16_t out_shift, q15_t *Im_out, const uint16_t dim_im_out, q15_t *bufferA)
```

参数:

\*`Im_in`: 指向输入的张量  
`dim_im_in`: 输入张量的维度  
`*wt`: 指向卷积核  
`ch_im_out`: 输出张量的通道数  
`dim_kernel`: 卷积核大小  
`padding`: 填充的大小  
`stride`: 步进的大小  
`*bias`: 偏差值  
`bias_shift`: 偏差值需要的左移位数  
`out_shift`: 输出结果需要的右移位数  
`*Im_out`: 指向输出张量  
`dim_im_out`: 输出张量的维度  
`*bufferA`: 临时区间

返回值:

无

临时区间大小:

需要的临时区间大小:  $2 * \text{ch\_im\_in} * \text{dim\_kernel} * \text{dim\_kernel}$

## 3.3 1x1 卷积

### 3.3.1 函数

- `csky_dsp2_convolve_1x1_HWC_q7_fast` : Q7 1x1 卷积快速版本

### 3.3.2 简要说明

函数是为 1x1 卷积的优化实现，可以用在 MobileNets 等网络。

暂时只支持 q7 版本的 1x1 卷积。

### 3.3.3 函数说明

#### 3.3.3.1 `csky_dsp2_convolve_1x1_HWC_q7_fast`

```
void csky_dsp2_convolve_1x1_HWC_q7_fast (const q7_t *Im_in, const uint16_t dim_im_in_x, const uint16_t dim_im_in_y, const uint16_t ch_im_in, const q7_t *wt, const uint16_t ch_im_out, const q7_t *bias, const uint16_t bias_shift, const uint16_t out_shift, const q7_t *Im_out, const uint16_t dim_im_out_x, const uint16_t dim_im_out_y, q15_t *bufferA)
```

**参数:**

\*Im\_in: 指向输入的张量  
dim\_im\_in\_x: 输入张量的 x 维度  
dim\_im\_in\_y: 输入张量的 y 维度  
ch\_im\_in: 输入张量的通道数量  
\*wt: 指向卷积核  
ch\_im\_out: 输出张量的通道数  
\*bias: 偏差值  
bias\_shift: 偏差值需要的左移位数  
out\_shift: 输出结果需要的右移位数  
\*Im\_out: 指向输出张量  
dim\_im\_out\_x: 输出张量的 x 维度  
dim\_im\_out\_y: 输出张量的 y 维度  
\*bufferA: 临时区间

**返回值:**

无

**限制:**

函数的实现有如下限制：ch\_im\_in 必须是 4 的倍数，ch\_im\_out 必须是 2 的倍数。

## 3.4 可分离卷积

### 3.4.1 函数

- `csky_DSP2_DEPTHWISE_SEPARABLE_CONV_HWC_Q7`: Q7 可分离卷积

### 3.4.2 简要说明

暂时只支持 q7 版本的可分离卷积。

### 3.4.3 函数说明

#### 3.4.3.1 `csky_DSP2_DEPTHWISE_SEPARABLE_CONV_HWC_Q7`

```
void csky_DSP2_DEPTHWISE_SEPARABLE_CONV_HWC_Q7 (const q7_t *Im_in, const uint16_t dim_im_in, const uint16_t ch_im_in, const q7_t *wt, const uint16_t ch_im_out, const uint16_t dim_kernel, const uint16_t padding, const uint16_t stride, const q7_t *bias, const uint16_t bias_shift, const uint16_t out_shift, q7_t *Im_out, const uint16_t dim_im_out, q15_t *bufferA)
```

参数:

\*`Im_in`: 指向输入的张量  
`dim_im_in`: 输入张量的维度  
`ch_im_in`: 输入张量的通道数量  
\*`wt`: 指向卷积核  
`ch_im_out`: 输出张量的通道数  
`dim_kernel`: 卷积核大小  
`padding`: 填充的大小  
`stride`: 步进的大小  
\*`bias`: 偏差值  
`bias_shift`: 偏差值需要的左移位数  
`out_shift`: 输出结果需要的右移位数  
\*`Im_out`: 指向输出张量  
`dim_im_out`: 输出张量的维度  
\*`bufferA`: 临时区间

返回值:

无

临时区间大小:

需要的临时区间大小:  $2 * \text{ch\_im\_in} * \text{dim\_kernel} * \text{dim\_kernel}$

# 第四章 全连接层函数

## 4.1 全连接函数

### 4.1.1 函数

- `csky_dsp2_fully_connected_mat_q7_vec_q15`: Q7 矩阵和 Q15 向量全连接
- `csky_dsp2_fully_connected_q15`: Q15 全连接
- `csky_dsp2_fully_connected_q7`: Q7 全连接

### 4.1.2 简要说明

全连接层本质上是矩阵与向量相乘并加上偏差的计算。矩阵是权重，而输入的向量则是来自激活层的输出。

支持的矩阵和向量的数据类型的组合有：8 位和 8 位，16 位和 16 位，8 位和 16 位。

基础的函数使用通常的 GEMV。使用优化的接口需要使用特别调整之后的权重参数。

### 4.1.3 函数说明

#### 4.1.3.1 `csky_dsp2_fully_connected_mat_q7_vec_q15`

```
void csky_dsp2_fully_connected_mat_q7_vec_q15 (const q15_t *pv, const q7_t *pm, const
    uint16_t dim_vec, const uint16_t num_of_rows, const uint16_t bias_shift, const
    uint16_t out_shift, const q7_t *bias, q15_t *pout)
```

**参数:**

- \*pv: 指向输入的向量
- \*pm: 指向输入的矩阵
- dim\_vec: 向量的长度
- num\_of\_rows: 矩阵的行数
- bias\_shift: 偏差的左移位数
- out\_shift: 结果的右移位数
- \*bias: 指向偏差
- \*pout: 指向输出向量

返回值:

无

#### 4.1.3.2 csky\_dsp2\_fully\_connected\_q15

```
void csky_dsp2_fully_connected_q15 (const q15_t *pv, const q15_t *pm, const uint16_t dim_vec, const uint16_t num_of_rows, const uint16_t bias_shift, const uint16_t out_shift, const q15_t *bias, q15_t *pout)
```

参数:

- \*pv: 指向输入的向量
- \*pm: 指向输入的矩阵
- dim\_vec: 向量的长度
- num\_of\_rows: 矩阵的行数
- bias\_shift: 偏差的左移位数
- out\_shift: 结果的右移位数
- \*bias: 指向偏差
- \*pout: 指向输出向量

返回值:

无

#### 4.1.3.3 csky\_dsp2\_fully\_connected\_q7

```
void csky_dsp2_fully_connected_q7 (const q7_t *pv, const q7_t *pm, const uint16_t dim_vec, const uint16_t num_of_rows, const uint16_t bias_shift, const uint16_t out_shift, const q15_t *bias, q7_t *pout)
```

参数:

- \*pv: 指向输入的向量
- \*pm: 指向输入的矩阵
- dim\_vec: 向量的长度
- num\_of\_rows: 矩阵的行数
- bias\_shift: 偏差的左移位数
- out\_shift: 结果的右移位数
- \*bias: 指向偏差
- \*pout: 指向输出向量

返回值:

无

# 第五章 池化函数

支持最大值池化和平均值池化。

## 5.1 最大值池化

### 5.1.1 函数

- `csky_dsp2_maxpool_q7_HWC` : Q7 最大值池化

### 5.1.2 简要说明

暂时只支持 q7 版本。

### 5.1.3 函数说明

#### 5.1.3.1 `csky_dsp2_maxpool_q7_HWC`

```
void csky_dsp2_maxpool_q7_HWC (q7_t *Im_in, const uint16_t dim_im_in, const uint16_t_
↪ch_im_in, const uint16_t dim_kernel, const uint16_t padding, const uint16_t stride,_
↪const uint16_t dim_im_out, q7_t *Im_out)
```

参数:

\*`Im_in`: 指向输入的张量  
`dim_im_in`: 输入张量的维度  
`ch_im_in`: 输入张量的通道数  
`dim_kernel`: 核的大小  
`padding`: 填充的大小  
`stride`: 步进的大小  
`dim_im_out`: 输出张量的维度  
\*`Im_out`: 指向输出张量

返回值:

无

注意:

计算过程中会破坏输入值。

## 5.2 平均值池化

### 5.2.1 函数

- `csky_dsp2_avepool_q7_HWC` : Q7 平均池化

### 5.2.2 简要说明

暂时只支持 q7 版本。

### 5.2.3 函数说明

#### 5.2.3.1 `csky_dsp2_avepool_q7_HWC`

```
void csky_dsp2_avepool_q7_HWC (q7_t *Im_in, const uint16_t dim_im_in, const uint16_t_
→ch_im_in, const uint16_t dim_kernel, const uint16_t padding, const uint16_t stride,_
→const uint16_t dim_im_out, q7_t *bufferA, q7_t *Im_out)
```

**参数:**

\*`Im_in`: 指向输入的张量  
`dim_im_in`: 输入张量的维度  
`ch_im_in`: 输入张量的通道数  
`dim_kernel`: 核的大小  
`padding`: 填充的大小  
`stride`: 步进的大小  
`dim_im_out`: 输出张量的维度  
`*bufferA`: 临时区间  
\*`Im_out`: 指向输出张量

**返回值:**

无

**临时区间大小:**

需要的临时区间大小:  $2 * \text{dim\_im\_out} * \text{ch\_im\_in}$

**注意:**

计算过程中会破坏输入值。

# 第六章 softmax 函数

## 6.1 softmax

### 6.1.1 函数

- *csky\_dsp2\_softmax\_q15* : Q15 softmax
- *csky\_dsp2\_softmax\_q7* : Q7 softmax

### 6.1.2 简要说明

相比通常的 softmax，函数使用了 2 的指数替代 e 的指数。

为 Q15 和 Q7 数据类型分别提供不同的函数。

### 6.1.3 函数说明

#### 6.1.3.1 *csky\_dsp2\_softmax\_q15*

```
void csky_dsp2_softmax_q15 (const q15_t *vec_in, const uint16_t dim_vec, q15_t *p_out)
```

参数:

\*vec\_in: 指向输入向量  
dim\_vec: 输入向量的维度  
\*p\_out: 返回的向量

返回值:

无

#### 6.1.3.2 *csky\_dsp2\_softmax\_q7*

```
void csky_dsp2_softmax_q7 (const q7_t *vec_in, const uint16_t dim_vec, q7_t *p_out)
```

参数:

\*vec\_in: 指向输入向量

dim\_vec: 输入向量的维度

\*p\_out: 返回的向量

返回值:

无

# 第七章 辅助函数

## 7.1 向量乘法

### 7.1.1 函数

- `csky_dsp2_nn_mult_q15`: Q15 向量相乘
- `csky_dsp2_nn_mult_q7`: Q7 向量相乘

### 7.1.2 简要说明

计算两个向量的相乘。  
为 Q15 和 Q7 数据类型分别提供不同的函数。

### 7.1.3 函数说明

#### 7.1.3.1 `csky_dsp2_nn_mult_q15`

```
void csky_dsp2_nn_mult_q15 (q15_t *pSrcA, q15_t *pSrcB, q15_t *pDst, const uint16_t ↴  
    ↴out_shift, uint32_t blockSize)
```

参数:

\*pSrcA: 指向输入向量 A  
\*pSrcB: 指向输入向量 B  
\*pDst: 指向输出向量  
out\_shift: 输出向量的右移位数  
blockSize: 输入向量的元素数量

返回值:

无

### 7.1.3.2 csky\_dsp2\_nn\_mult\_q7

```
void csky_dsp2_nn_mult_q7 (q7_t *pSrcA, q7_t *pSrcB, q7_t *pDst, const uint16_t out_
↪shift, uint32_t blockSize)
```

参数:

- \*pSrcA: 指向输入向量 A
- \*pSrcB: 指向输入向量 B
- \*pDst: 指向输出向量
- out\_shift: 输出向量的右移位数
- blockSize: 输入向量的元素数量

返回值:

无

## 7.2 数据转换

### 7.2.1 函数

- `csky_dsp2_q7_to_q15_no_shift`: 将 Q7 向量转换为 Q15 向量
- `csky_dsp2_q7_to_q15_reordered_no_shift`: 将 Q7 向量转换为 Q15 向量，并且重排

### 7.2.2 简要说明

处理网络中需要的类型转换。

### 7.2.3 函数说明

#### 7.2.3.1 `csky_dsp2_q7_to_q15_no_shift`

```
void csky_dsp2_q7_to_q15_no_shift (const q7_t *pSrc, q15_t *pDst, uint32_t blockSize)
```

参数:

- \*`pSrc`: 指向输入向量
- \*`pDst`: 指向输出向量
- `blockSize`: 输出向量的元素数量

返回值:

无

#### 7.2.3.2 `csky_dsp2_q7_to_q15_reordered_no_shift`

```
void csky_dsp2_q7_to_q15_reordered_no_shift (const q7_t *pSrc, q15_t *pDst, uint32_t  
    ↴blockSize)
```

参数:

- \*`pSrc`: 指向输入向量
- \*`pDst`: 指向输出向量
- `blockSize`: 输出向量的元素数量

返回值:

无