

THUNDER

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	MRCHeader Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	6
3.1.2	Member Data Documentation . . . . .	6
3.1.2.1	cella . . . . .	6
3.1.2.2	cellb . . . . .	6
3.1.2.3	dmax . . . . .	6
3.1.2.4	dmean . . . . .	6
3.1.2.5	dmin . . . . .	6
3.1.2.6	extra . . . . .	6
3.1.2.7	ispg . . . . .	7
3.1.2.8	label . . . . .	7
3.1.2.9	machst . . . . .	7
3.1.2.10	map . . . . .	7
3.1.2.11	mapc . . . . .	7
3.1.2.12	mapr . . . . .	7
3.1.2.13	maps . . . . .	7
3.1.2.14	mode . . . . .	8

3.1.2.15	<a href="#">mx</a>	8
3.1.2.16	<a href="#">my</a>	8
3.1.2.17	<a href="#">mz</a>	8
3.1.2.18	<a href="#">nlabels</a>	8
3.1.2.19	<a href="#">nsymbt</a>	8
3.1.2.20	<a href="#">nx</a>	8
3.1.2.21	<a href="#">nxstart</a>	9
3.1.2.22	<a href="#">ny</a>	9
3.1.2.23	<a href="#">nystart</a>	9
3.1.2.24	<a href="#">nz</a>	9
3.1.2.25	<a href="#">nzstart</a>	9
3.1.2.26	<a href="#">origin</a>	9
3.1.2.27	<a href="#">rms</a>	9
<b>4</b>	<b>File Documentation</b>	<b>11</b>
4.1	<a href="#">include/Geometry/Euler.h File Reference</a>	11
4.1.1	<a href="#">Detailed Description</a>	12
4.1.2	<a href="#">Function Documentation</a>	12
4.1.2.1	<a href="#">alignZ()</a>	12
4.1.2.2	<a href="#">angle()</a> [1/3]	13
4.1.2.3	<a href="#">angle()</a> [2/3]	13
4.1.2.4	<a href="#">angle()</a> [3/3]	14
4.1.2.5	<a href="#">direction()</a>	14
4.1.2.6	<a href="#">quaternion()</a> [1/3]	14
4.1.2.7	<a href="#">quaternion()</a> [2/3]	15
4.1.2.8	<a href="#">quaternion()</a> [3/3]	15
4.1.2.9	<a href="#">quaternion_conj()</a>	15
4.1.2.10	<a href="#">quaternion_mul()</a>	16
4.1.2.11	<a href="#">randRotate2D()</a>	16
4.1.2.12	<a href="#">randRotate3D()</a>	16
4.1.2.13	<a href="#">reflect3D()</a>	17
4.1.2.14	<a href="#">rotate2D()</a> [1/2]	17
4.1.2.15	<a href="#">rotate2D()</a> [2/2]	17
4.1.2.16	<a href="#">rotate3D()</a> [1/3]	18
4.1.2.17	<a href="#">rotate3D()</a> [2/3]	18
4.1.2.18	<a href="#">rotate3D()</a> [3/3]	18
4.1.2.19	<a href="#">rotate3DX()</a>	19
4.1.2.20	<a href="#">rotate3DY()</a>	19
4.1.2.21	<a href="#">rotate3DZ()</a>	19
4.1.2.22	<a href="#">swingTwist()</a>	20
4.2	<a href="#">include/Image/MRCHeader.h File Reference</a>	20
4.2.1	<a href="#">Detailed Description</a>	20
<b>Index</b>		<b>21</b>

# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">MRCHeader</a>	
The constitution of MRC main header . . . . .	<a href="#">5</a>



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

include/Geometry/ <a href="#">Euler.h</a>	
<a href="#">Euler.h</a> contains several functions, for operations of quaternions, converting between Euler angles, rotation matrices and unit quaternions and sampling rotation matrices from even distribution	11
include/Image/ <a href="#">MRCHheader.h</a>	
<a href="#">MRCHheader.h</a> contains the main header of the MRC format, including fixed format values for metadata about the images/volumes . . . . .	20





## Chapter 3

# Class Documentation

### 3.1 MRCHeader Struct Reference

The constitution of MRC main header.

```
#include <MRCHeader.h>
```

#### Public Attributes

- int [nx](#)
- int [ny](#)
- int [nz](#)
- int [mode](#)
- int [nxstart](#)
- int [nystart](#)
- int [nzstart](#)
- int [mx](#)
- int [my](#)
- int [mz](#)
- float [cella](#) [3]
- float [cellb](#) [3]
- int [mapc](#)
- int [mapr](#)
- int [maps](#)
- float [dmin](#)
- float [dmax](#)
- float [dmean](#)
- int [ispg](#)
- int [nsymbt](#)
- char [extra](#) [100]
- float [origin](#) [3]
- char [map](#) [4]
- int [machst](#)
- float [rms](#)
- int [nlabels](#)
- char [label](#) [10][80]

### 3.1.1 Detailed Description

The constitution of MRC main header.

The main header is limited to 1024 bytes, but includes unassigned space in anticipation of future extensions.

### 3.1.2 Member Data Documentation

#### 3.1.2.1 cella

```
float MRCHeader::cella[3]
```

cell dimensions in angstroms

#### 3.1.2.2 cellb

```
float MRCHeader::cellb[3]
```

cell angles in degrees

#### 3.1.2.3 dmax

```
float MRCHeader::dmax
```

maximum density value

#### 3.1.2.4 dmean

```
float MRCHeader::dmean
```

mean density value

#### 3.1.2.5 dmin

```
float MRCHeader::dmin
```

minimum density value

#### 3.1.2.6 extra

```
char MRCHeader::extra[100]
```

extra space used for anything - 0 by default

### 3.1.2.7 ispg

```
int MRCHHeader::ispg
```

space group number 0 or 1 (default=0)

### 3.1.2.8 label

```
char MRCHHeader::label[10][80]
```

ten 80-character text labels Symmetry records follow - if any - stored as text as in International Tables, operators separated by \* and grouped into 'lines' of 80 characters (ie. symmetry operators do not cross the ends of the 80-character 'lines' and the 'lines' do not terminate in a \*). Data records follow.

### 3.1.2.9 machst

```
int MRCHHeader::machst
```

machine stamp

### 3.1.2.10 map

```
char MRCHHeader::map[4]
```

character string 'MAP ' to identify file type

### 3.1.2.11 mapc

```
int MRCHHeader::mapc
```

axis corresp to cols (1,2,3 for X,Y,Z)

### 3.1.2.12 mapr

```
int MRCHHeader::mapr
```

axis corresp to rows (1,2,3 for X,Y,Z)

### 3.1.2.13 maps

```
int MRCHHeader::maps
```

axis corresp to sections (1,2,3 for X,Y,Z)

#### 3.1.2.14 mode

```
int MRCHheader::mode
```

MODE data type : 0 image : signed 8-bit bytes range -128 to 127 1 image : 16-bit halfwords 2 image : 32-bit reals 3 transform : complex 16-bit integers 4 transform : complex 32-bit reals 6 image : unsigned 16-bit range 0 to 65535

#### 3.1.2.15 mx

```
int MRCHheader::mx
```

number of intervals along X

#### 3.1.2.16 my

```
int MRCHheader::my
```

number of intervals along Y

#### 3.1.2.17 mz

```
int MRCHheader::mz
```

number of intervals along Z

#### 3.1.2.18 nlabels

```
int MRCHheader::nlabels
```

number of labels being used

#### 3.1.2.19 nsymbt

```
int MRCHheader::nsymbt
```

number of bytes used for symmetry data (0 or 80)

#### 3.1.2.20 nx

```
int MRCHheader::nx
```

number of columns (fastest changing in map)

#### 3.1.2.21 nxstart

```
int MRCHHeader::nxstart
```

number of first column in map (Default = 0)

#### 3.1.2.22 ny

```
int MRCHHeader::ny
```

number of rows

#### 3.1.2.23 nystart

```
int MRCHHeader::nystart
```

number of first row in map

#### 3.1.2.24 nz

```
int MRCHHeader::nz
```

number of sections (slowest changing in map)

#### 3.1.2.25 nzstart

```
int MRCHHeader::nzstart
```

number of first section in map

#### 3.1.2.26 origin

```
float MRCHHeader::origin[3]
```

origin in X,Y,Z used for transforms

#### 3.1.2.27 rms

```
float MRCHHeader::rms
```

rms deviation of map from mean density

The documentation for this struct was generated from the following file:

- include/Image/[MRCHHeader.h](#)



## Chapter 4

# File Documentation

### 4.1 include/Geometry/Euler.h File Reference

[Euler.h](#) contains several functions, for operations of quaternions, converting between Euler angles, rotation matrices and unit quaternions and sampling rotation matrices from even distribution.

```
#include <cmath>
#include <gsl/gsl_math.h>
#include "Macro.h"
#include "Typedef.h"
#include "Precision.h"
#include "Random.h"
#include "Functions.h"
```

#### Functions

- void [quaternion\\_mul](#) (dvec4 &dst, const dvec4 &a, const dvec4 &b)  
*Calculate the product of two quaternions  $\mathbf{q}_1$  and  $\mathbf{q}_2$ .*
- dvec4 [quaternion\\_conj](#) (const dvec4 &quat)  
*Calculate the conjugate quaternion of a quaternion.*
- void [angle](#) (double &phi, double &theta, const dvec3 &src)  
*Calculate  $\phi$  and  $\theta$  given a certain direction  $\mathbf{v}$ .*
- void [angle](#) (double &phi, double &theta, double &psi, const dmat33 &src)  
*Calculate  $\phi$ ,  $\theta$  and  $\psi$  of the rotation represented by the rotation matrix  $\mathbf{R}$ .*
- void [angle](#) (double &phi, double &theta, double &psi, const dvec4 &src)  
*Calculate  $\phi$ ,  $\theta$  and  $\psi$  of the rotation represented by the unit quaternion  $\mathbf{q}$ .*
- void [quaternion](#) (dvec4 &dst, const double phi, const double theta, const double psi)  
*Calculate the unit quaternion  $\mathbf{q}$  for representing the rotation, given 3 Euler angles  $\phi$ ,  $\theta$  and  $\psi$ .*
- void [quaternion](#) (dvec4 &dst, const double phi, const dvec3 &axis)  
*Calculate the unit quaternion  $\mathbf{q}$  for representing the rotation, given the rotation axis  $\mathbf{r}$  and the rotation angle around this axis  $\phi$ .*
- void [quaternion](#) (dvec4 &dst, const dmat33 &src)  
*Calculate the unit quaternion  $\mathbf{q}$  for representing the rotation, given the rotation matrix  $\mathbf{R}$ .*
- void [rotate2D](#) (dmat22 &dst, const dvec2 &vec)  
*Calculate the rotation matrix (2D)  $\mathbf{R}$ , which rotates the unit vector  $\mathbf{v}_0 = \{1, 0\}$  to the given unit vector  $\mathbf{v}$ .*

- void `rotate2D` (dmat22 &dst, const double phi)  
*Calculate the rotation matrix (2D)  $\mathbf{R}$ , given the rotation angle  $\phi$ .*
- void `direction` (dvec3 &dst, const double phi, const double theta)  
*Calculate the unit direction vector  $\mathbf{v}$ , given the rotation angle  $\phi$  and  $\theta$ .*
- void `rotate3D` (dmat33 &dst, const double phi, const double theta, const double psi)  
*Calculate the rotation matrix  $\mathbf{R}$ , given the rotation angle  $\phi$ ,  $\theta$  and  $\psi$ .*
- void `rotate3D` (dmat33 &dst, const dvec4 &src)  
*Calculate the rotation matrix  $\mathbf{R}$ , given the unit quaternion  $\mathbf{q}$  which represents this rotation.*
- void `rotate3DX` (dmat33 &dst, const double phi)  
*Calculate the rotation matrix  $\mathbf{R}$  which represents the rotation along X-axis with rotation angle  $\phi$ .*
- void `rotate3DY` (dmat33 &dst, const double phi)  
*Calculate the rotation matrix  $\mathbf{R}$  which represents the rotation along Y-axis with rotation angle  $\phi$ .*
- void `rotate3DZ` (dmat33 &dst, const double phi)  
*Calculate the rotation matrix  $\mathbf{R}$  which represents the rotation along Z-axis with rotation angle  $\phi$ .*
- void `alignZ` (dmat33 &dst, const dvec3 &vec)  
*Calculate the rotation matrix  $\mathbf{R}$  which aligns a direction vector  $\mathbf{v}$  to Z-axis.*
- void `rotate3D` (dmat33 &dst, const double phi, const dvec3 &axis)  
*Calculate the rotation matrix  $\mathbf{R}$  which represents the rotation along the axis  $\mathbf{v}$  with rotation angle  $\phi$ .*
- void `reflect3D` (dmat33 &dst, const dvec3 &plane)  
*Calculate the transformation matrix  $\mathbf{M}$  of reflection against a certian plane, which is represented by its normal vector  $\mathbf{n}$ .*
- void `swingTwist` (dvec4 &swing, dvec4 &twist, const dvec4 &src, const dvec3 &vec)  
*Calculate the two quaternions  $\mathbf{q}_s$  and  $\mathbf{q}_t$ , which represent swing and twist along axis  $\mathbf{v}$  respectively, representing the rotation represented by quaternion  $\mathbf{q}$ .*
- void `randRotate2D` (dmat22 &rot)  
*Sample a 2D rotation matrix  $\mathbf{R}$  from even distribution.*
- void `randRotate3D` (dmat33 &rot)  
*Sample a 3D rotation matrix  $\mathbf{R}$  from even distribution.*

#### 4.1.1 Detailed Description

`Euler.h` contains several functions, for operations of quaternions, converting between Euler angles, rotation matrices and unit quaternions and sampling rotation matrices from even distribution.

Quaternions are a number system that extends the complex numbers. Unit quaternions provide a convenient mathematical notation for representing rotations of objects in 3D. Compared to Euler angles, they are simpler to compose and avoid the problem of gimbal lock. Compared to rotation matrices, they are more compact and more efficient. Moreover, unlike Euler angles, unit quaternions do not rely on the choosing and order of the rotation axes.

To be noticed, Euler angles in this file follow the standard of ZXZ Euler system. In other words, Euler angle set  $\{\phi, \theta, \psi\}$  stands for rotating along Z axis with  $\phi$ , followed by rotating along X axis with  $\theta$ , and followed by rotating along Z axis with  $\psi$ .

#### 4.1.2 Function Documentation

##### 4.1.2.1 alignZ()

```
void alignZ (
    dmat33 & dst,
    const dvec3 & vec )
```

Calculate the rotation matrix  $\mathbf{R}$  which aligns a direction vector  $\mathbf{v}$  to Z-axis.



## Parameters

out	<i>dst</i>	<b>R</b>
in	<i>vec</i>	<b>v</b>

4.1.2.2 `angle()` [1/3]

```
void angle (
    double & phi,
    double & theta,
    const dvec3 & src )
```

Calculate  $\phi$  and  $\theta$  given a certain direction **v**.

**v** must be a unit vector. Output value  $\phi$  ranges  $[0, 2\pi)$ , and  $\theta$  ranges  $[0, \pi]$ .

## Parameters

out	<i>phi</i>	$\phi$
out	<i>theta</i>	$\theta$
in	<i>src</i>	<b>v</b>

4.1.2.3 `angle()` [2/3]

```
void angle (
    double & phi,
    double & theta,
    double & psi,
    const dmat33 & src )
```

Calculate  $\phi$ ,  $\theta$  and  $\psi$  of the rotation represented by the rotation matrix **R**.

**R** must be an orthogonal matrix and determinant of which equals to 1. In other words,  $RR^T = I$  and  $\det A = 1$ . Output value  $\phi$  ranges  $[0, 2\pi)$ ,  $\theta$  ranges  $[0, \pi]$ , and  $\psi$  ranges  $[0, 2\pi)$ .

## Parameters

out	<i>phi</i>	$\phi$
out	<i>theta</i>	$\theta$
out	<i>psi</i>	$\psi$
in	<i>src</i>	<b>R</b>

#### 4.1.2.4 angle() [3/3]

```
void angle (
    double & phi,
    double & theta,
    double & psi,
    const dvec4 & src )
```

Calculate  $\phi$ ,  $\theta$  and  $\psi$  of the rotation represented by the unit quaternion  $\mathbf{q}$ .

##### Parameters

out	<i>phi</i>	$\phi$
out	<i>theta</i>	$\theta$
out	<i>psi</i>	$\psi$
in	<i>src</i>	$\mathbf{q}$

#### 4.1.2.5 direction()

```
void direction (
    dvec3 & dst,
    const double phi,
    const double theta )
```

Calculate the unit direction vector  $\mathbf{v}$ , given the rotation angle  $\phi$  and  $\theta$ .

##### Parameters

out	<i>dst</i>	$\mathbf{v}$
in	<i>phi</i>	$\phi$
in	<i>theta</i>	$\theta$

#### 4.1.2.6 quaternion() [1/3]

```
void quaternion (
    dvec4 & dst,
    const double phi,
    const double theta,
    const double psi )
```

Calculate the unit quaternion  $\mathbf{q}$  for representing the rotation, given 3 Euler angles  $\phi$ ,  $\theta$  and  $\psi$ .

##### Parameters

out	<i>dst</i>	$\mathbf{q}$
in	<i>phi</i>	$\phi$
in	<i>theta</i>	$\theta$
in	<i>psi</i>	$\psi$

## 4.1.2.7 quaternion() [2/3]

```
void quaternion (
    dvec4 & dst,
    const double phi,
    const dvec3 & axis )
```

Calculate the unit quaternion **q** for representing the rotation, given the rotation axis **r** and the rotation angle around this axis  $\phi$ .

This rotation axis **r** must be a unit vector, while the rotation angle  $\phi$  ranges  $(-\infty, +\infty)$ .

## Parameters

out	<i>dst</i>	<b>q</b>
in	<i>phi</i>	$\phi$
in	<i>axis</i>	<b>r</b>

## 4.1.2.8 quaternion() [3/3]

```
void quaternion (
    dvec4 & dst,
    const dmat33 & src )
```

Calculate the unit quaternion **q** for representing the rotation, given the rotation matrix **R**.

## Parameters

out	<i>dst</i>	<b>q</b>
in	<i>src</i>	<b>R</b>

## 4.1.2.9 quaternion\_conj()

```
dvec4 quaternion_conj (
    const dvec4 & quat )
```

Calculate the conjugate quaternion of a quaternion.

## Returns

the conjugate quaternion

## Parameters

in	<i>quat</i>	a quaternion
----	-------------	--------------

## 4.1.2.10 quaternion\_mul()

```
void quaternion_mul (
    dvec4 & dst,
    const dvec4 & a,
    const dvec4 & b )
```

Calculate the product of two quaternions  $\mathbf{q}_1$  and  $\mathbf{q}_2$ .

Assuming that  $\mathbf{q}_1 = (w_1, x_1, y_1, z_1)$  and  $\mathbf{q}_2 = (w_2, x_2, y_2, z_2)$ , the product can be calculated as

$$\begin{pmatrix} w_1 \\ x_1 \\ y_1 \\ z_1 \end{pmatrix} \times \begin{pmatrix} w_2 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\ w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2 \\ w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2 \\ w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2 \end{pmatrix}$$

## Parameters

out	<i>dst</i>	product, a quaternion
in	<i>a</i>	left multiplier, $\mathbf{q}_1$
in	<i>b</i>	right multiplier, $\mathbf{q}_2$

## 4.1.2.11 randRotate2D()

```
void randRotate2D (
    dmat22 & rot )
```

Sample a 2D rotation matrix  $\mathbf{R}$  from even distribution.

## Parameters

out	<i>rot</i>	$\mathbf{R}$
-----	------------	--------------

## 4.1.2.12 randRotate3D()

```
void randRotate3D (
    dmat33 & rot )
```

Sample a 3D rotation matrix  $\mathbf{R}$  from even distribution.

## Parameters

out	<i>rot</i>	<b>R</b>
-----	------------	----------

## 4.1.2.13 reflect3D()

```
void reflect3D (
    dmat33 & dst,
    const dvec3 & plane )
```

Calculate the transformation matrix **M** of reflection against a certian plane, which is represented by its normal vector **n**.

## Parameters

out	<i>dst</i>	<b>M</b>
in	<i>plane</i>	<b>n</b>

## 4.1.2.14 rotate2D() [1/2]

```
void rotate2D (
    dmat22 & dst,
    const dvec2 & vec )
```

Calculate the rotation matrix (2D) **R**, which rotates the unit vector  $\mathbf{v}_0 = \{1, 0\}$  to the given unit vector **v**.

## Parameters

out	<i>dst</i>	<b>R</b>
in	<i>vec</i>	<b>v</b>

## 4.1.2.15 rotate2D() [2/2]

```
void rotate2D (
    dmat22 & dst,
    const double phi )
```

Calculate the rotation matrix (2D) **R**, given the rotation angle  $\phi$ .

## Parameters

out	<i>dst</i>	<b>R</b>
in	<i>phi</i>	$\phi$

**4.1.2.16 rotate3D()** [1/3]

```
void rotate3D (
    dmat33 & dst,
    const double phi,
    const double theta,
    const double psi )
```

Calculate the rotation matrix **R**, given the rotation angle  $\phi$ ,  $\theta$  and  $\psi$ .

**Parameters**

out	<i>dst</i>	<b>R</b>
in	<i>phi</i>	$\phi$
in	<i>theta</i>	$\theta$
in	<i>psi</i>	$\psi$

**4.1.2.17 rotate3D()** [2/3]

```
void rotate3D (
    dmat33 & dst,
    const dvec4 & src )
```

Calculate the rotation matrix **R**, given the unit quaternion **q** which represents this rotation.

**Parameters**

out	<i>dst</i>	<b>R</b>
in	<i>src</i>	<b>q</b>

**4.1.2.18 rotate3D()** [3/3]

```
void rotate3D (
    dmat33 & dst,
    const double phi,
    const dvec3 & axis )
```

Calculate the rotation matrix **R** which represents the rotation along the axis **v** with rotation angle  $\phi$ .

**Parameters**

out	<i>dst</i>	<b>R</b>
in	<i>phi</i>	$\phi$
in	<i>axis</i>	<b>v</b>

#### 4.1.2.19 rotate3DX()

```
void rotate3DX (
    dmat33 & dst,
    const double phi )
```

Calculate the rotation matrix **R** which represents the rotation along X-axis with rotation angle  $\phi$ .

##### Parameters

out	<i>dst</i>	<b>R</b>
in	<i>phi</i>	$\phi$

#### 4.1.2.20 rotate3DY()

```
void rotate3DY (
    dmat33 & dst,
    const double phi )
```

Calculate the rotation matrix **R** which represents the rotation along Y-axis with rotation angle  $\phi$ .

##### Parameters

out	<i>dst</i>	<b>R</b>
in	<i>phi</i>	$\phi$

#### 4.1.2.21 rotate3DZ()

```
void rotate3DZ (
    dmat33 & dst,
    const double phi )
```

Calculate the rotation matrix **R** which represents the rotation along Z-axis with rotation angle  $\phi$ .

##### Parameters

out	<i>dst</i>	<b>R</b>
in	<i>phi</i>	$\phi$

#### 4.1.2.22 swingTwist()

```
void swingTwist (
    dvec4 & swing,
    dvec4 & twist,
    const dvec4 & src,
    const dvec3 & vec )
```

Calculate the two quaternions  $\mathbf{q}_s$  and  $\mathbf{q}_t$ , which represent swing and twist along axis  $\mathbf{v}$  respectively, representing the rotation represented by quaternion  $\mathbf{q}$ .

##### Parameters

out	<i>swing</i>	$\mathbf{q}_s$
out	<i>twist</i>	$\mathbf{q}_t$
in	<i>src</i>	$\mathbf{q}$
in	<i>vec</i>	$\mathbf{v}$

## 4.2 include/Image/MRCHeader.h File Reference

[MRCHeader.h](#) contains the main header of the MRC format, including fixed format values for metadata about the images/volumes.

### Classes

- struct [MRCHeader](#)  
*The constitution of MRC main header.*

#### 4.2.1 Detailed Description

[MRCHeader.h](#) contains the main header of the MRC format, including fixed format values for metadata about the images/volumes.

MRC is a file format that has become industry standard in cryo-electron microscopy (cryoEM) and electron tomography (ET), where the result of the technique is a three-dimensional grid of voxels each with a value corresponding to electron density or electric potential.

Reference: Cheng, Anchi; Henderson, Richard; Mastronarde, David; Ludtke, Steven J.; Schoenmakers, Remco H.M.; Short, Judith; Marabini, Roberto; Dallakyan, Sargis; Agard, David; Winn, Martyn (November 2015). "MR↔C2014: Extensions to the MRC format header for electron cryo-microscopy and tomography". *Journal of Structural Biology*. 192 (2): 146–150. doi:10.1016/j.jsb.2015.04.002.



# Index

alignZ  
    Euler.h, [12](#)

angle  
    Euler.h, [13](#)

cella  
    MRCHeader, [6](#)

cellb  
    MRCHeader, [6](#)

direction  
    Euler.h, [14](#)

dmax  
    MRCHeader, [6](#)

dmean  
    MRCHeader, [6](#)

dmin  
    MRCHeader, [6](#)

Euler.h  
    alignZ, [12](#)  
    angle, [13](#)  
    direction, [14](#)  
    quaternion, [14](#), [15](#)  
    quaternion\_conj, [15](#)  
    quaternion\_mul, [16](#)  
    randRotate2D, [16](#)  
    randRotate3D, [16](#)  
    reflect3D, [17](#)  
    rotate2D, [17](#)  
    rotate3DX, [19](#)  
    rotate3DY, [19](#)  
    rotate3DZ, [19](#)  
    rotate3D, [18](#)  
    swingTwist, [19](#)

extra  
    MRCHeader, [6](#)

include/Geometry/Euler.h, [11](#)

include/Image/MRCHeader.h, [20](#)

ispg  
    MRCHeader, [6](#)

label  
    MRCHeader, [7](#)

MRCHeader, [5](#)  
    cella, [6](#)  
    cellb, [6](#)  
    dmax, [6](#)  
    dmean, [6](#)  
    dmin, [6](#)  
    extra, [6](#)  
    ispg, [6](#)  
    label, [7](#)  
    machst, [7](#)  
    map, [7](#)  
    mapc, [7](#)  
    mapr, [7](#)  
    maps, [7](#)  
    mode, [7](#)  
    mx, [8](#)  
    my, [8](#)  
    mz, [8](#)  
    nlabels, [8](#)  
    nsymbt, [8](#)  
    nx, [8](#)  
    nxstart, [8](#)  
    ny, [9](#)  
    nystart, [9](#)  
    nz, [9](#)  
    nzstart, [9](#)  
    origin, [9](#)  
    rms, [9](#)  
    machst  
        MRCHeader, [7](#)  
    map  
        MRCHeader, [7](#)  
    mapc  
        MRCHeader, [7](#)  
    mapr  
        MRCHeader, [7](#)  
    maps  
        MRCHeader, [7](#)  
    mode  
        MRCHeader, [7](#)  
    mx  
        MRCHeader, [8](#)  
    my  
        MRCHeader, [8](#)  
    mz  
        MRCHeader, [8](#)  
    nlabels  
        MRCHeader, [8](#)  
    nsymbt  
        MRCHeader, [8](#)  
    nx  
        MRCHeader, [8](#)  
    nxstart  
        MRCHeader, [8](#)

- ny
  - MRCHeader, 9
- nystart
  - MRCHeader, 9
- nz
  - MRCHeader, 9
- nzstart
  - MRCHeader, 9
- origin
  - MRCHeader, 9
- quaternion
  - Euler.h, 14, 15
- quaternion\_conj
  - Euler.h, 15
- quaternion\_mul
  - Euler.h, 16
- randRotate2D
  - Euler.h, 16
- randRotate3D
  - Euler.h, 16
- reflect3D
  - Euler.h, 17
- rms
  - MRCHeader, 9
- rotate2D
  - Euler.h, 17
- rotate3DX
  - Euler.h, 19
- rotate3DY
  - Euler.h, 19
- rotate3DZ
  - Euler.h, 19
- rotate3D
  - Euler.h, 18
- swingTwist
  - Euler.h, 19