

MatrixLib Memory Layout and Alignment

C NeonF32x4

- register : float32x4_t
- width = 128 bits

maps_to = Vec4f
maps_to = Quatf
maps_to = Mat row (4 floats)

NEON operations:
vld1q_f32() - load
vst1q_f32() - store
vaddq_f32() - add
vmulq_f32() - multiply

C PodProperties

- trivially_copyable : true

Vec3f
Mat3f
Mat4f
Quatf

Compatible with:
memcpy()
DMA transfers
C interfaces
binary serialization

C AlignmentProperties

- alignas(16)

Benefits:
SIMD efficiency
cache alignment
hardware acceleration
atomic operations

C Vec3f

- data[0] : float // offset 0-3
- data[1] : float // offset 4-7
- data[2] : float // offset 8-11
- padding : float // offset 12-15

size = 16 bytes
alignment = 16
simd_compatible = true

Vec3f is 16 bytes and aligned to 16.
Padding ensures SIMD-friendly layout.

C Mat3f

- data[0] : float // row 0 col 0
- data[1] : float // row 0 col 1
- data[2] : float // row 0 col 2
- data[3] : float // row 1 col 0
- data[4] : float // row 1 col 1
- data[5] : float // row 1 col 2
- data[6] : float // row 2 col 0
- data[7] : float // row 2 col 1
- data[8] : float // row 2 col 2
- padding[3] : float // offsets 36-47

size = 48 bytes
alignment = 16
storage = row-major

Row-major storage.
Total size: 48 bytes (36 data + 12 pad).
element[r][c] = data[r * 3 + c].

C Quatf

- x : float // offset 0-3
- y : float // offset 4-7
- z : float // offset 8-11
- w : float // offset 12-15

size = 16 bytes
alignment = 16
simd_compatible = true

Layout: [x, y, z, w].
SIMD compatible (NEON 128-bit).
vec() returns Vec3f(x, y, z).

C Mat4f

- row0[4] : float // offsets 0-15
- row1[4] : float // offsets 16-31
- row2[4] : float // offsets 32-47
- row3[4] : float // offsets 48-63

size = 64 bytes
alignment = 16
simd_compatible = true

Row-major 4x4 matrix.
Each row is 16 bytes (4 floats).
Perfect fit for SIMD operations.

C CmsisMatF32

- type : arm_matrix_instance_f32
- storage : float32_t array

maps_to = Mat<float,R,C>
maps_to = SquareMat<float,N>

CMSIS-DSP operations:
arm_mat_mult_f32()
arm_mat_inverse_f32()
arm_add_f32()
arm_dot_prod_f32()