**Freescale Semiconductor**
Application Note

Document Number:AN4699
Rev. 1.0, 4/2013

# Data Structures for Matrix and Vector Algebra

**by: Mark Pedley**

**Contents**



## 1 Introduction

This Application Note documents the matrix and vector algebra functions in Freescale's Xtrinsic eCompass and Magnetic Calibration software provided under license at www.freescale.com/ecompass. This Application Note is part of the technical documentation for that software and its use and distribution are controlled by the license agreement.

## 2 Matrices and vectors in the C language

### 2.1 Matrices

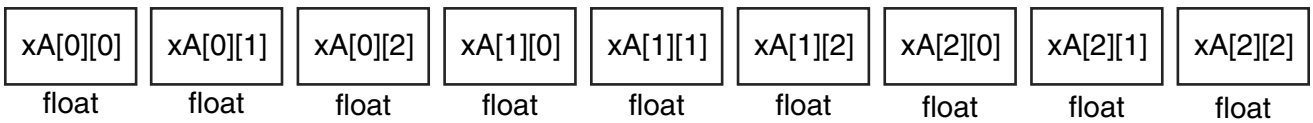
Matrices can be represented in the C language as either conventional two-dimensional arrays or as an array of pointers to one-dimensional arrays.

The first usage would be typified by the declaration:

```
float A[3][3];
```

The array A is stored in sequential 4-byte memory blocks with code automatically generated by the compiler to calculate the offset of the specific element `A[row][column]` . The floating

point number stored at `A[row][column]` is accessed by computing the offset (3*row+column) times 4 bytes and then adding this offset to the base address `A`.

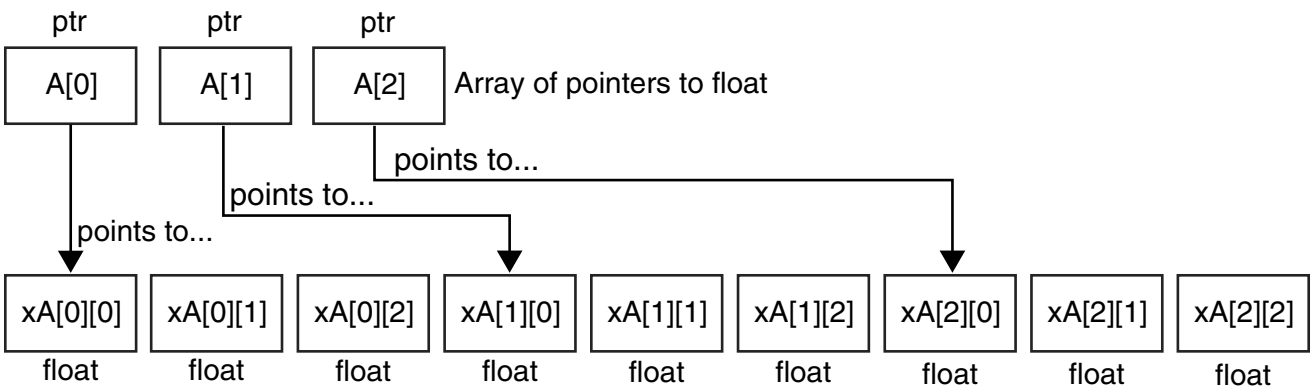



There is a significant drawback to this approach which is that the C compiler must know the number of columns in the matrix to compute the offset. The makes it difficult to write general matrix functions in the C programming language which can be passed matrices of variable dimensions.

The workaround used in the Xtrinsic software is to access matrix elements via an array of pointers to the first elements of the matrix rows. This allows the use of variable size matrices but has the small drawback of increasing memory usage by requiring the creation of the array of 4-byte pointers to the matrix rows. The initialization code then looks like:

```
float  xA[3][3], *A[3];
for  (i = 0; i < 3; i++)
  A[i] = xA[i];
```

Here the declaration `xA[3][3]` creates the 3x3 matrix `xA` with the required storage for its elements. The loop over matrix rows `A[i] = xA[i]` assigns the elements of the array of pointers `A` to the first elements of the matrix rows A. From then onwards, the matrix is only accessed via the array of pointers `A`. The resulting code to access element `[i][j]` via the construct `A[i][j]` looks like conventional C matrix code but *under the hood* is implemented using an array of pointers to the matrix rows.




## 2.2 Vectors

In contrast, the C language can handle variable length vectors since the vector name is a pointer to the first element. However, the Xtrinsic software also handles vectors as a special case of matrices with just one column in order to allow the use of the general purpose matrix algebra functions. The drawback is that memory requirement for floating point vectors is doubled since both the vector and the array of 4-byte pointers to the vector rows are required.

An example line of vector assignment in the Xtrinsic software is:

```
ftmpA7x1[6][0] = 1.0F;
```

## 2.3 Example function fmatrixAeqBxC

The matrix multiplication function below exemplifies this approach. Matrices or vectors of arbitrary dimensions are accessed using arrays of pointers to pointers to float (`float **A`, `float **B`, `float **C`) which point to the rows of the actual underlying matrices containing the data elements.

```
// function calculates the matrix product A = B x C
void fmatrixAeqBxC(float **A, float **B, float **C, int32 rB, int32 cBrC, int32 cC)
{
  // rB = rows in B
  // cBrC = columns in B = rows in C
  // cC = columns in C
  // A has dimension rB rows x cC columns

  int32 i, j, k;  // counters

  for (i = 0; i < rB; i++)
  {
    for (j = 0; j < cC; j++)
    {
      A[i][j] = 0.0F;
      for (k = 0; k < cBrC; k++)
        A[i][j] += B[i][k] * C[k][j];
    }
  }
  return;
}
```

## *How to Reach Us:*


**Home Page:**
www.freescale.com

**Web Support:**
http://www.freescale.com/support

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com





Document Number: AN4699
Rev. 1.0, 4/2013