M4 DSP Matrix Functions

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code uses the M4 DSP to calculate Matrix Functions
BSP Version	M480 Series BSP CMSIS V3.04.000
Hardware	NuMaker-PFM-M487 Ver 3.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Function Description

1.1 Introduction

This sample code uses the CMSIS DSP library to calculate matrix functions, including

1. Matrix Addition
2. Matrix Subtraction
3. Matrix Multiplication
4. Matrix Inverse
5. Matrix Scale
6. Matrix Transpose

User can use these functions to implement mathematical equations. The sample code compares the difference between using the DSP calculation time and calculates the efficiency improvement ratio.

1.2 Principle

The CMSIS DSP library provides calculation of matrix functions to reduce computation time. User need to define matrix variables with `arm_matrix_instance_f32`, and then initial matrix setting with `arm_mat_init_f32()`. Its definition is as shown in table：

`arm_mat_init_f32(arm_matrix_instance_f32 *S, uint16_t nRows, uint16_t nColumns, float32_t *pData)`		
Parameters：	`*S`	[in, out] points to an instance of the floating-point matrix structure
	`nRows`	[in] number of rows in the matrix
	`nColumns`	[in] number of columns in the matrix
	`*pData`	[in] points to the matrix data array
Returns：		none

`arm_mat_add_f32(const arm_matrix_instance_f32 *pSrcA, const arm_matrix_instance_f32 *pSrcB, arm_matrix_instance_f32 *pDst)`		
Parameters：	`*pSrcA`	[in] points to first input matrix structure
	`*pSrcB`	[in] points to second input matrix structure
	`*pDst`	[out] points to output matrix structure

Returns：	ARM_MATH_SUCCESS : Operation successful ARM_MATH_SIZE_MISMATCH : Matrix size check failed

`arm_mat_sub_f32(const arm_matrix_instance_f32 *pSrcA, const arm_matrix_instance_f32 *pSrcB, arm_matrix_instance_f32 *pDst)`		
Parameters：	`*pSrcA`	[in] points to the first input matrix structure
	`*pSrcB`	[in] points to the second input matrix structure
	`*pDst`	[out] points to output matrix structure
Returns：		ARM_MATH_SUCCESS : Operation successful ARM_MATH_SIZE_MISMATCH : Matrix size check failed

`arm_mat_mult_f32(const arm_matrix_instance_f32 *pSrcA, const arm_matrix_instance_f32 *pSrcB, arm_matrix_instance_f32 *pDst)`		
Parameters：	`*pSrcA`	[in] points to the first input matrix structure
	`*pSrcB`	[in] points to the second input matrix structure
	`*pDst`	[out] points to output matrix structure
Returns：		ARM_MATH_SUCCESS : Operation successful ARM_MATH_SIZE_MISMATCH : Matrix size check failed

`arm_mat_inverse_f32(const arm_matrix_instance_f32 *pSrc, arm_matrix_instance_f32 *pDst)`		
Parameters：	`*pSrc`	[in] points to input matrix structure
	`*pDst`	[out] points to output matrix structure
Returns：		ARM_MATH_SUCCESS : Operation successful ARM_MATH_SIZE_MISMATCH : Matrix size check failed ARM_MATH_SINGULAR : Input matrix is found to be singular (non-invertible)

arm_mat_scale_f32(const arm_matrix_instance_f32 *pSrc, float32_t scale, arm_matrix_instance_f32 *pDst)		
Parameters：	*pSrc	[in] points to input matrix
	scale	[in] scale factor to be applied
	*pDst	[out] points to output matrix structure
Returns：		ARM_MATH_SUCCESS : Operation successful ARM_MATH_SIZE_MISMATCH : Matrix size check failed

arm_mat_trans_f32(const arm_matrix_instance_f32 *pSrc, arm_matrix_instance_f32 *pDst)		
Parameters：	*pSrc	[in] points to input matrix
	*pDst	[out] points to output matrix
Returns：		ARM_MATH_SUCCESS : Operation successful ARM_MATH_SIZE_MISMATCH : Matrix size check failed

1.3 Demo Result

```
+----------------------------------------+
|     DSP Matrix functions Sample Code   |
+----------------------------------------+

Calculating time with DSP instruction is 0.040667 ms

Calculating time without DSP instruction is 0.243083 ms

Efficiency increase rate is 5.98
```

2 Code Description

Use CMSIS DSP library to calculate matrix functions：

```
/* Initialize A Matrix Instance with numRows, numCols and data array(A_f32) */
arm_mat_init_f32(&A, numRows, numCols, A_f32);

/* Initialize Matrix Instance AT with numRows, numCols and data array(AT_f32) */
arm_mat_init_f32(&AT, numRows, numCols, AT_f32);

/* calculation of A transpose with DSP instruction */
arm_mat_trans_f32(&A, &AT);

/* Initialize ATMA Matrix Instance with numRows, numCols and data array(ATMA_f32) */
arm_mat_init_f32(&ATMA, numRows, numCols, ATMA_f32);

/* calculation of AT Multiply with A with DSP instruction */
arm_mat_mult_f32(&AT, &A, &ATMA);

/* Initialize ATMAI Matrix Instance with numRows, numCols and data array(ATMAI_f32) */
arm_mat_init_f32(&ATMAI, numRows, numCols, ATMAI_f32);

/* calculation of Inverse ((Transpose(A) * A) with DSP instruction */
arm_mat_inverse_f32(&A, &ATMAI);
```

And calculation matrix functions without DSP：

```
/* Matrix Inverse (number of rows) */
void MatrixInverse(int n)
{
    uint32_t i,j,k;
    double t;
    for(i=0; i<n; ++i) {
        for(j=0; j<n; ++j) {
            A1_f32[i][n+j] = 0;
        }
    }
    for(i=0; i<n; ++i) {
        A1_f32[i][n+i] = 1;
    }
    for(i=0; i<n; ++i) {
```

```
        if(A1_f32[i][i] == 0)
            for(j=i+1; j<n; ++j)
                if(A1_f32[j][i] != 0) {
                    for(k=i; k<n*2; ++k)
                        swap(&A1_f32[i][k], &A1_f32[j][k]);
                    break;
                }
        /* if Matrix inverse not exist*/
        if(A1_f32[i][i] == 0) break;

        t = A1_f32[i][i];
        for(k=i; k<n*2; ++k) {
            A1_f32[i][k] /= t;
        }
        for(j=0; j<n; ++j) {
            if(i != j && A1_f32[j][i] != 0) {
                double t = A1_f32[j][i];
                for(k=i; k<n*2; ++k) {
                    A1_f32[j][k] -= A1_f32[i][k] * t;
                }
            }
        }
    }
}
/* Matrix Transpose (number of rows, number of columns) */
void MatrixTranspose(int m, int n)
{
    uint32_t i, j;
    for(i=0; i<n; i++) {
        for(j=0; j<m; j++) {
            resultT[i][j] = A1_f32[j][i];
        }
    }
}
/* Matrix Multiplication (number of rows, number of columns) */
void MatrixMult(int m, int n, int p)
{
    uint32_t i, j, k;
    for(i=0; i<m; i++) {
        for(j=0; j<p; j++) {
            resultM[i][j] = 0;
```

```
            for(k=0; k<n; k++) {
                resultM[i][j] = resultM[i][j] + resultT[i][k] * A1_f32[k][j];
            }
        }
    }
}
```

Change the counter to time, where the timer clock source is HXT 12MHz：

```
/* Calculate the time, timer clock source is 12M, unit is ms */
DSPCalTime = (DSPCalTime/12000000) * 1000;
CalTime = (CalTime/12000000)* 1000;
```

3 Software and Hardware Environment

Software environment

- BSP version
  - M480 Series BSP CMSIS V3.04.000
- IDE version
  - Keil uVersion 5.26

Hardware environment

- Circuit components
  - NuMaker-PFM-M487 or other M480 Development Board
- Diagram

M480's UART0_RX (PB.12) and UART0_TX (PB.13) are connected to Nu-Link Me to print the message. Set COM port and Baud. The number of the COM Port can be found in the device manager "NuBridge Virtual Com Port (COMX)" and Baud is set to 115200.

UART0_RX (PB.12)	↔	TX
UART0_TX (PB.13)	↔	RX
M480		Nu-Link-Me

4 Directory Information

EC_M480_DSP_Matrix_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click DSP_Matrix.uvproj
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Jun. 25, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.