Embedded in Tomorrow™

AN204957

F²MC-8FX Family 8-Bit Microcontroller MATH API Application Note

Associated Part Family: F²MC-8FX Family

This application note describes math API, which can be applied to all series F²MC-8FX 8-Bit Microcontroller.

Contents

1 Introduction

This application note describes math API, which can be applied to all series F²MC-8FX 8-Bit Microcontroller. This math APIs can calculate more efficiently. Compared to F²MC-8FX compiler's math arithmetic, only multiplication and division APIs which need to be advanced are implemented in this Math API.

2 Analysis for Cypress C Compiler Math Lib

2.1 Overview

Only multiplication and division of unsigned integer are implemented in this AN, for it is often utilized by user, so developing an optimum arithmetic can make user decrease calculation and advance system efficiency. In order to introduce the math API easily, below abbreviations will be used from here on.

- unsigned char: UChar
- unsigned int: UInt
- unsigned long: ULong
- UChar * UChar: unsigned char data multiplied by unsigned char data
- UChar * UInt: unsigned char data multiplied by unsigned int data
- UInt * UInt: unsigned int data multiplied by unsigned int data
- ULong * UChar: unsigned long data multiplied by unsigned char data
- ULong * UInt: unsigned long data multiplied by unsigned int data
- ULong * ULong: unsigned long data multiplied by unsigned long data
- UChar / UChar: unsigned char data divided by unsigned char data
- UInt / UChar: unsigned int data divided by unsigned char data
- UInt / UInt: unsigned int data divided by unsigned int data

- ULong / UChar: unsigned long data divided by unsigned char data
- ULong / UInt: unsigned long data divided by unsigned int data
- ULong / ULong: unsigned long data divided by unsigned long data

The total Math lib should cover below calculation. However only some of them need to be implemented, and some need not as its original arithmetic is already optimized.

- Multiplication:

UChar*UChar, UChar*UInt, UInt*UInt, ULong*UChar, ULong*UInt, ULong*Ulong.

- Division:

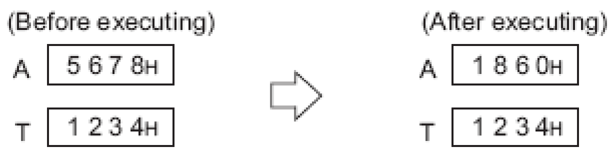
UChar/UChar, UInt/UChar, UInt/UInt, ULong/UChar, ULong/UInt, ULong/Ulong.

2.2 Instruction of Multiplication and Division

2.2.1 MULU A

This instruction performs an unsigned multiplication of AL (lower 8-bit of the accumulator) and TL (lower 8-bit of the temporary accumulator), and stores the 16-bit result in A. The contents of T (temporary accumulator) do not change. The contents of AH (higher 8-bit of the accumulator) and TH (higher 8-bit of the temporary accumulator) before execution of the instruction are not used for the operation. Figure 2-1 shows a summary of the instruction.

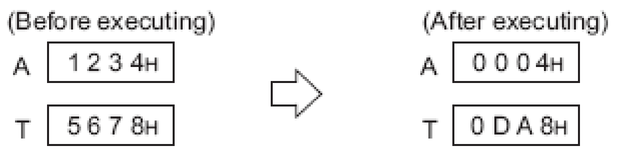
Figure 2-1. Summary of the MULU Instruction



2.2.2 DIVU A

This instruction divides the 16-bit value in T by the unsigned 16-bit value in A, and stores the 16-bit result and the 16-bit remainder in A and T respectively. When the value in A before execution of instruction is "0", the Z flag becomes "1" to indicate zero-division is executed. Figure 2-2 shows a summary of the instruction.

Figure 2-2. Summary of the DIVU Instruction



2.3 Analysis for Multiplication

2.3.1 UChar*UChar, Ulong*Ulong

The MULU means that 8bit data multiplies 8bit data, the length of result will be 16bit.

For UChar*UChar, the max product is 65025, less than 0xFFFF, MULU is enough, assembly of Cypress C compiler math lib uses MULU directly.

For ULong*ULong, the product may be more than 0xFFFF, MULU is not enough, assembly of Cypress C compiler math lib uses math lib long*long, it means 4char*4char, uses MULU 10 times. There is no redundancy MULU calculation.

So for UChar*UChar, Ulong*Ulong, the assembly is simple, the RAM/ROM/MCLK usage is the least, if adopting math API, 4 more steps are needed:

1. Store parameters in stack
2. Save registers in stack
3. Save result
4. Restore registers

The RAM/ROM/MCLK usage of math API is more than Cypress C compiler math lib, so it needn't to adopt math API.

2.3.2 UChar*UInt, UInt*UInt, ULong*UChar, ULong*UInt

For UChar*UInt, UInt*UInt, ULong*UChar, ULong*UInt, ULong*ULong, the product may be more than 0xFFFF, MULU is not enough.

In C code, if use UChar*UInt or UInt*UInt directly, the product is not correct, because the assembly calculates it by using Fujitsu C compiler math lib int*int, and then throw away the high byte.

For example,

unsigned char multiplicator_char = 0xFF;

unsigned int multiplicand_int = 0xFFFF;

unsigned long product_char_int;

void main(void)

product_char_int = multiplicator* multiplicand_int;

The product_char_int is 0xFF01, but correct product_char_int should be 0xFEFF01

In C code, if use (unsighed long)UcharMulUInt or (unsighed long)UInt*UInt, the product is correct, because the assembly calculates it by using Cypress C compiler math lib long*long, and then can save the high byte.

For example,

unsigned char multiplicator_char = 0xFF;

unsigned int multiplicand_int = 0xFFFF;

unsigned long product_char_int;

void main(void)

product_char_int = (unsigned long)multiplicator* multiplicand_int;

The product_char_int is 0xFEFF01.

In C code, use ULong*UChar, ULong*UInt directly, the product is correct (ignore the high bytes which is more than 0xFFFFFFFF), because the assembly calculates it by using Cypress C compiler math lib long*long.

But when using Cypress C compiler math lib long*long, it means 4char*4char, uses MULU 10 times.

Actually, For UChar*UInt, 1char*2char is enough,

For UInt*UInt, 2char*2char is enough,

For ULong*UChar, 4char*1char is enough,

For ULong*UInt, 4char*2char is enough,

Some MULU calculations are redundancy, because the product is 0, so math API can just calculate which the product is not 0, the target of math API is to solve this problem

2.4 Analysis for Division

2.4.1 UChar/UChar, UInt/UChar, UInt/UInt, ULong/Ulong

The DIVU means that 16bit data divided by 16bit data, the quotient is 16bit data, and the remainder is 16 bit data (ignore the remainder in math API).

For UChar/UChar, UInt/UChar, UInt/UInt, DIVU is enough, assembly of Cypress C compiler math lib uses DIVU directly.

For ULong/ULong, DIVU is not enough, assembly of Fujitsu C compiler math lib uses math lib long/long. There is no redundancy MULU calculation.

The assembly is simply, the RAM/ROM/MCLK usage is the least, if adopting math API, 4 more steps are needed:

1. Store parameters in stack
2. Save registers in stack
3. Save result
4. Restore registers

The RAM/ROM/MCLK usage of math API is more than Cypress C compiler math lib, so it needn't to adopt math API.

2.4.2 ULong/UChar, ULong/UInt

In C code, use ULong/UChar, ULong/UInt directly, the assembly calculates it by using Cypress C compiler math lib long/long.

But Cypress C compiler math lib long/long, will check if it's ULong/UChar, ULong/UInt or Ulong/ULong, and then select related code for ULong/UChar, ULong/UInt, it means RAM/ROM/MCLK usage is more than it's needed actually.

But math API can calculate directly, because it's known if ULong/UChar, ULong/Uint, so it needs to adopt math API.

3 Cypress Math API

3.1 Multiplication API

3.1.1 API for UChar*UInt

Name	unsigned long UCharMulUInt(unsigned char mul1, unsigned int mul2)
Parameter	unsigned char mul1, unsigned int mul2
Return	unsigned long

Description: unsigned char mul1 multiplies unsigned int mul2

Sample:

```
unsigned char    multiplicator_char = 0xFF;
unsigned int     multiplicand_int = 0xFFFF;
unsigned long product_char_int;
void main(void)
product_char_int = UCharMulUInt(multiplicator_char,multiplicand_int);
```

3.1.2 API for UInt*UInt

Name	unsigned long UIntMulUInt(unsigned int mul1, unsigned int mul2)
Parameter	unsigned int mul1, unsigned int mul2
Return	unsigned long

Description: unsigned int mul1 multiplies unsigned int mul2

Sample

```
unsigned int     multiplicator_int = 0xFFFF;
unsigned int     multiplicand_int = 0xFFFF;
unsigned long product_int_int;
void main(void)
product_int_int = UIntMulUInt(multiplicator_int, multiplicand_int);
```

3.1.3 API for ULong*UChar

Name	unsigned long ULongMulUChar(unsigned long mul1, unsigned char mul2)
Parameter	unsigned long mul1, unsigned char mul2
Return	unsigned long

Description: unsigned long mul1 multiplies unsigned char mul2

Sample

```
unsigned long     multiplicator_long = 0xFFFFFFFF;
unsigned char     multiplicand_char = 0xFF;
unsigned long product_long_char;
void main(void)
product_long_char = ULongMulUChar(multiplicator_long, multiplicand_char);
```

3.1.4 API for ULong*UInt

Name	unsigned long ULongMulUInt(unsigned long mul1, unsigned int mul2)
Parameter	unsigned long mul1, unsigned int mul2
Return	unsigned long

Description: unsigned long mul1 multiplies unsigned int mul2

Sample

```
unsigned long     multiplicator_long = 0xFFFFFFFF;
unsigned int      multiplicand_int = 0xFFFF;
unsigned long product_long_int;
void main(void)
product_long_int = ULongMulUint(multiplicator_long, multiplicand_int);
```

3.2 Division API

3.2.1 API for ULong/UChar

Name	unsigned long ULongDivUChar(unsigned long dividend, unsigned char divisor)
Parameter	unsigned long dividend, unsigned char divisor
Return	unsigned long

Description: unsigned long dividend divides unsigned char divisor

Sample

```
unsigned long     dividend_long = 0xFFFFFFFF;
unsigned char     divisor_char = 0xFF;
unsigned long quotient_long_char;
void main(void)
quotient_long_char = ULongDivUChar(dividend_long, divisor_char);
```

3.2.2 API for ULong/UInt

Name	unsigned long ULongDivUInt(unsigned long dividend, unsigned int divisor)
Parameter	unsigned long dividend, unsigned int divisor
Return	unsigned long

Description: unsigned long dividend divides unsigned int divisor

Sample

```
unsigned long      dividend_long = 0xFFFFFFFF;
unsigned int       divisor_int = 0xFFFF;
unsigned long quotient_long_int;
void main(void)
quotient_long_int = ULongDivUInt(dividend_long, divisor_int);
```

4 Cypress Math API Performance

Performance for math API is shown in Table 4-1.

Table 4-1. Performance Compare between Math API and C Compiler Math Lib

	C compiler Math Lib				Math API			
Calculation	**Times**	**MCLK**	**ROM**	**RAM**	**Times**	**MCLK**	**ROM**	**RAM**
UChar*Uint	10	252	127	10	2	87	40	8
UInt*Uint	10	252	127	10	4	168	88	10
ULong*Uchar	10	252	127	10	4	157	79	12
ULong*Uint	10	252	127	10	7	212	107	12
ULong/Uchar	0	1614	238	12	0	1327	80	12
ULong/Uint	0	1614	238	12	0	1320	76	12

Note: Times: how many times math API uses MULU/DIVU

MCLK: how many machine clock used by math API

ROM: how many bytes ROM used by math API

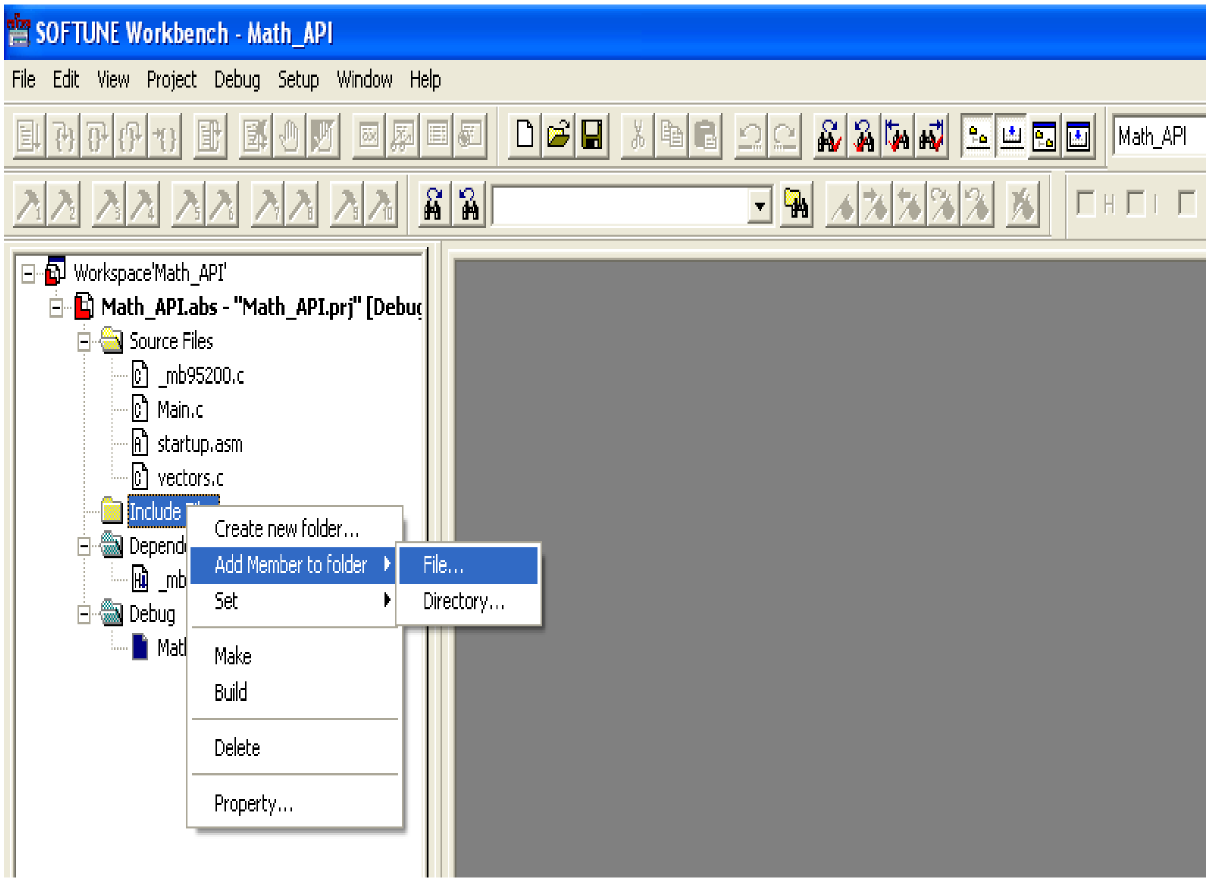
RAM: how many bytes RAM (stack) used by math API

5 How to use Cypress Math API Lib

5.1 Add Cypress Math_API.lib to User's Project

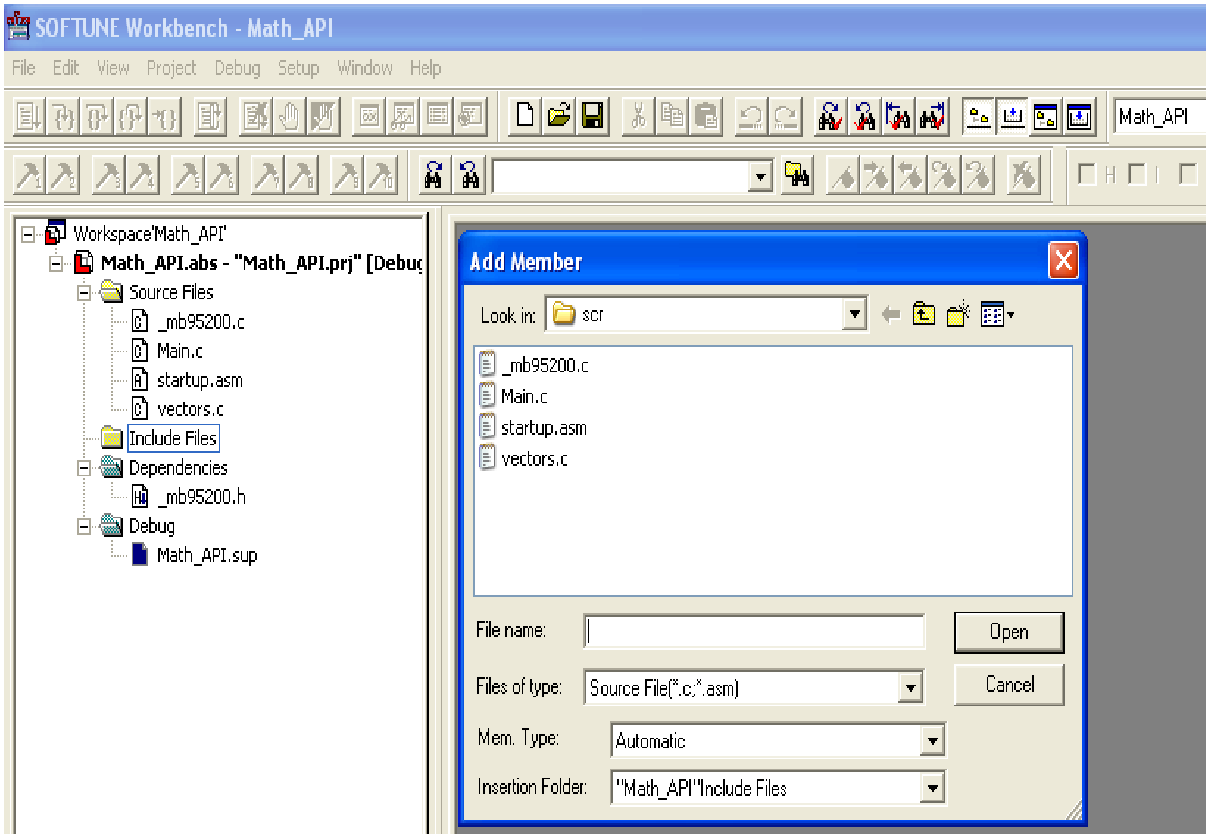
In Softune, right click **Include Files**, select **Add member to folder→File** from shortcut menu.

Figure 5-1. Select File



You will find you can't find the Math_API.lib.

Figure 5-2. Open Add Member Window



File name:

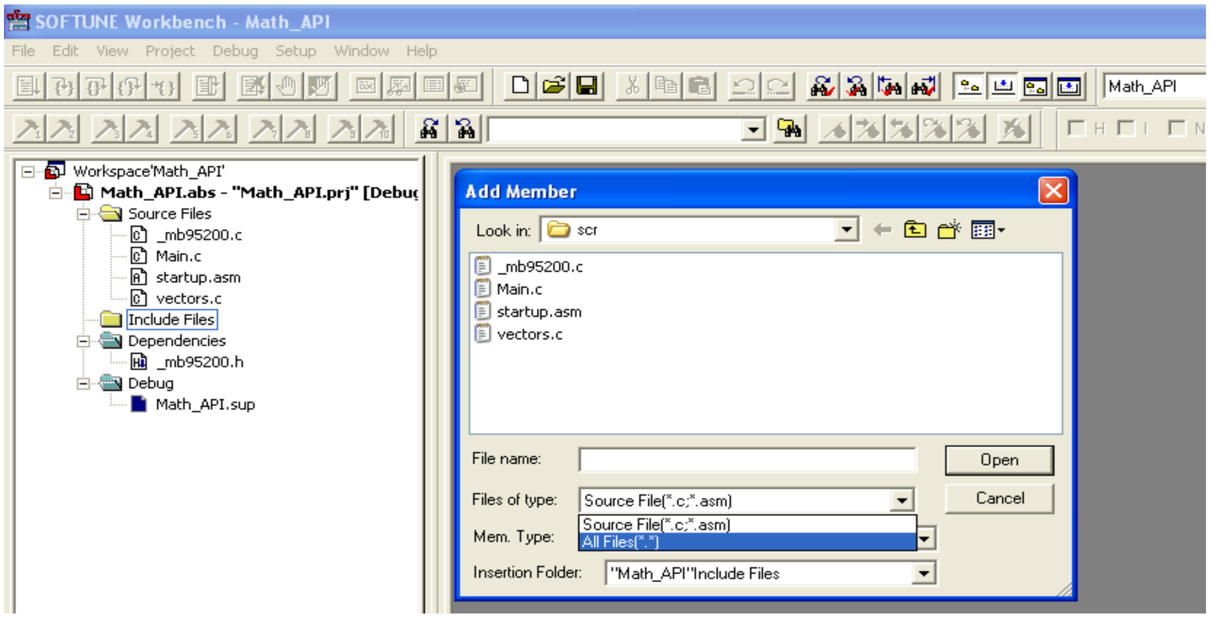
Files of type: Source File(*.c;*.asm)

Mem. Type: Automatic

Insertion Folder: "Math_API"Include Files

In window of **Add Member**, select **All Files** in **Files of type**, then you will find the Math_API.lib.

Figure 5-3. Select File Type



File name:

Files of type: Source File(*.c;*.asm)

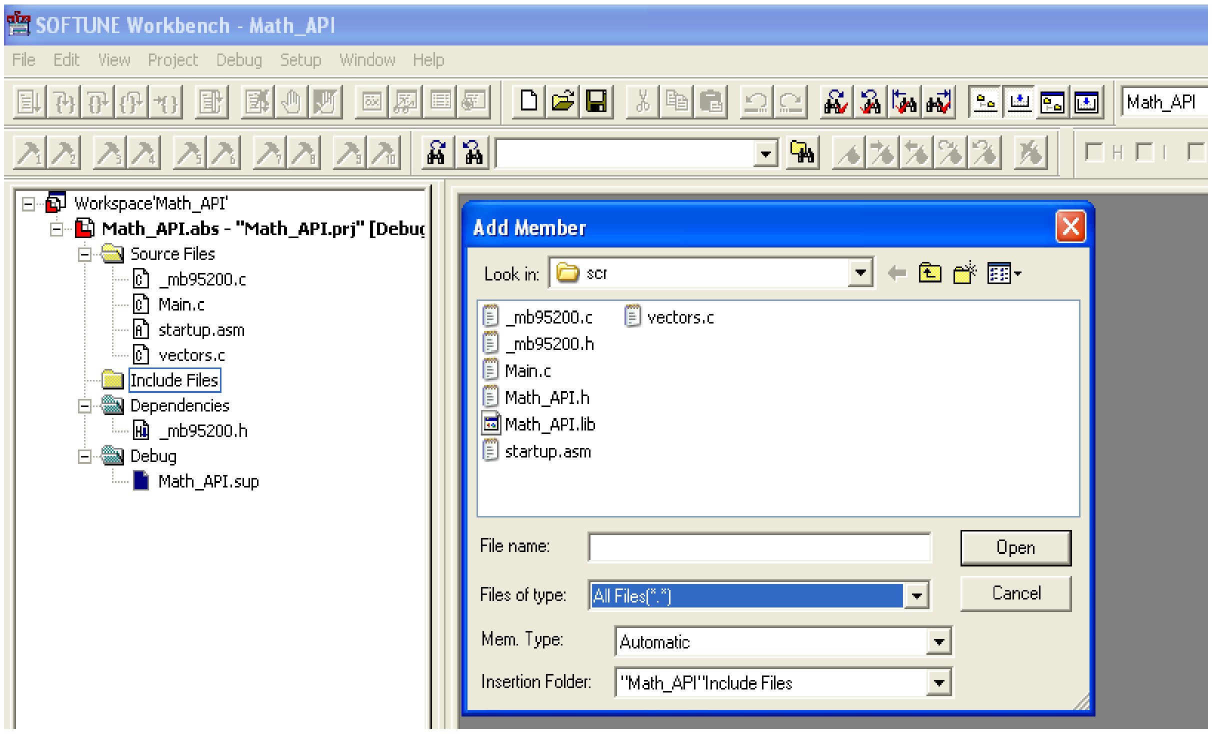
Source File(*.c;*.asm)

All Files(*.*)

Mem. Type:

Insertion Folder: "Math_API"Include Files

Figure 5-4: Show Math_API.lib



Look in: scr

_mb95200.c, vectors.c, _mb95200.h, Main.c, Math_API.h, Math_API.lib, startup.asm

File name:

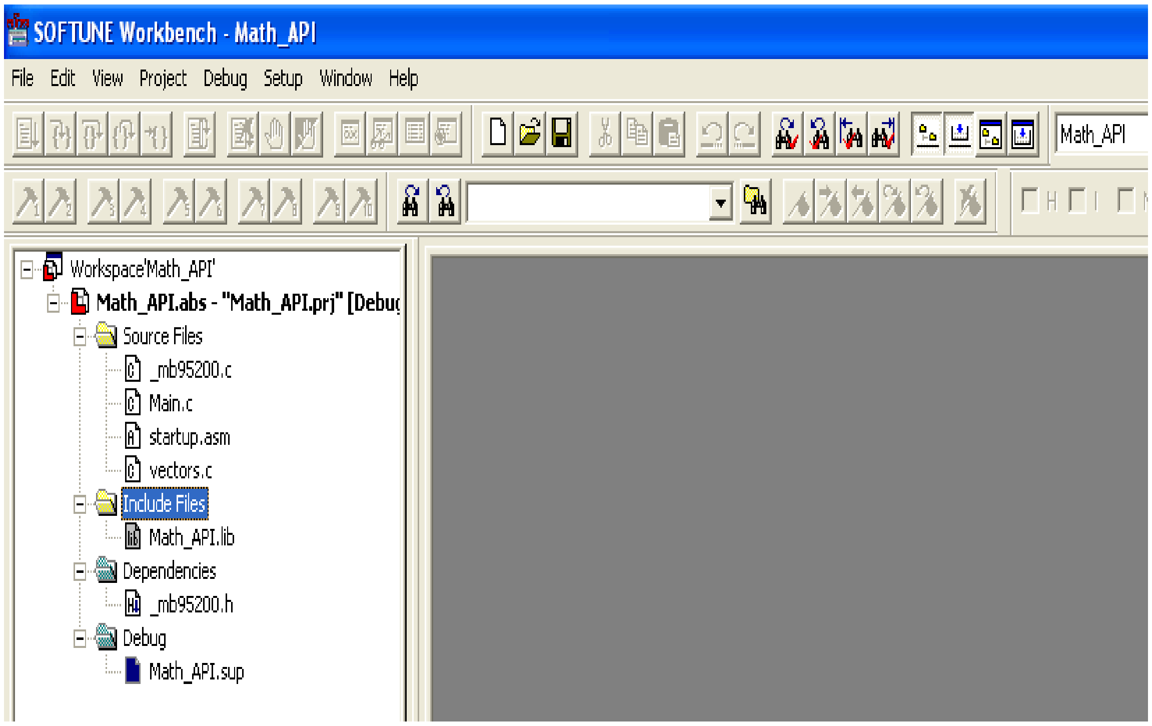
Files of type: All Files(*.*)

Mem. Type: Automatic

Insertion Folder: "Math_API"Include Files

In **Add Member**, select **All Files**, double click **Math_API.lib**. The Math_API.lib will be added to Source Files.

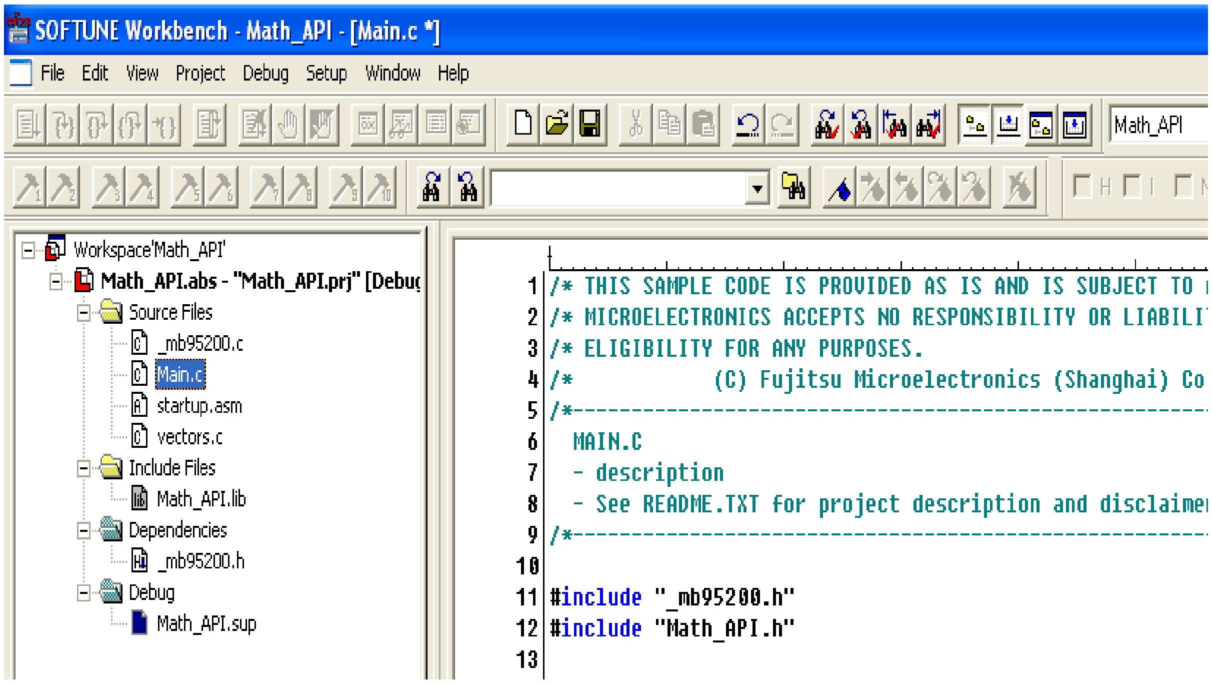
Figure 5-5. Add Math_API.lib



5.2 Add "#include " Math_API.h" "in c file

Add '#include "Math_API.h "' in c file, such as in "main.c ".

Figure 5-6. Add '#include " Math_API.h ""' in " main.c '

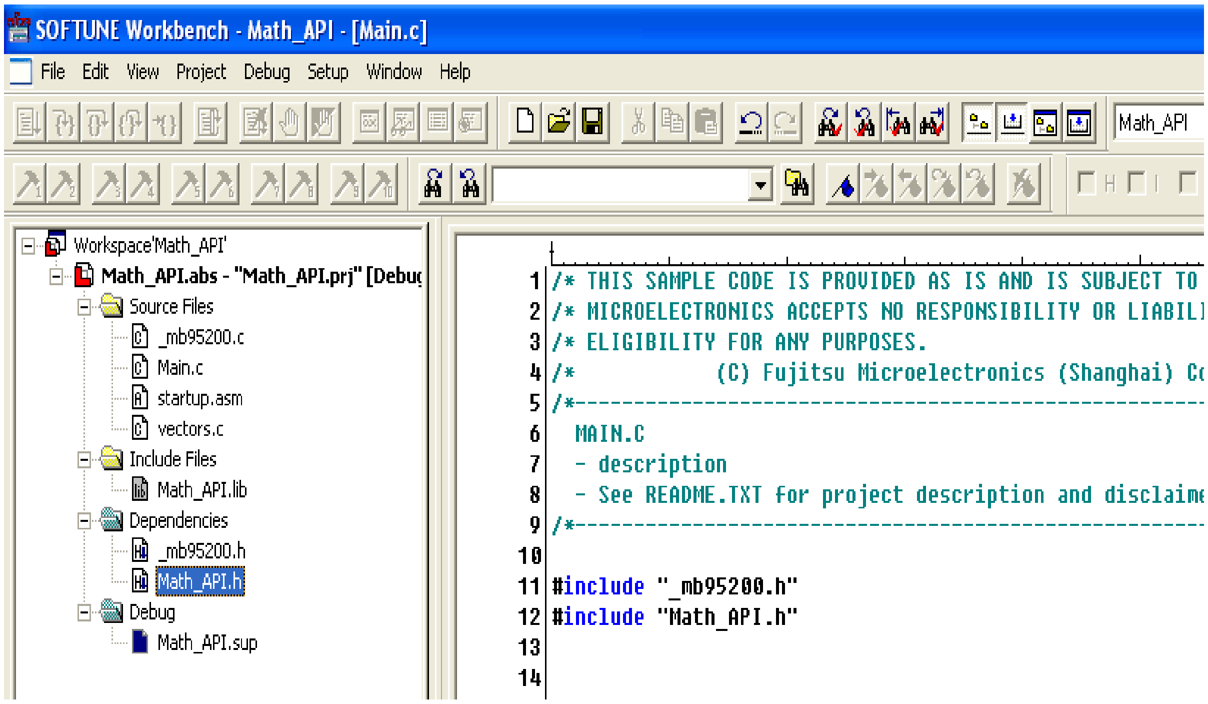


```
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO 
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILI
/* ELIGIBILITY FOR ANY PURPOSES.
/*           (C) Fujitsu Microelectronics (Shanghai) Co
/*-------------------------------------------------------
  MAIN.C
  - description
  - See README.TXT for project description and disclaime
/*-------------------------------------------------------

#include "_mb95200.h"
#include "Math_API.h"

```

Then compile the whole project, "Math.h" will link Math_API.lib to c file, so that user program can use API functions which are in Math_API.lib.



```
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILI
/* ELIGIBILITY FOR ANY PURPOSES.
/*           (C) Fujitsu Microelectronics (Shanghai) Co
/*-------------------------------------------------------
  MAIN.C
  - description
  - See README.TXT for project description and disclaime
/*-------------------------------------------------------

#include "_mb95200.h"
#include "Math_API.h"


```

6 Additional Information

Please contact your local support team for any technical question.

Document History

Document Title: AN204957 - F²MC-8FX Family 8-bit Microcontroller MATH API Application Note

Document Number: 002-04957

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	—	HUAL	12/07/2009	First draft
			12/22/2009	Add content
			12/30/2009	update section 2, 3
			01/04/2010	update API table in section 3 update API Performance table and description in section 3
			01/05/2010	update format
*A	5267480	HUAL	06/23/2016	Migrated Spansion Application Note “MCU-AN-500073-E-14” to Cypress format.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com