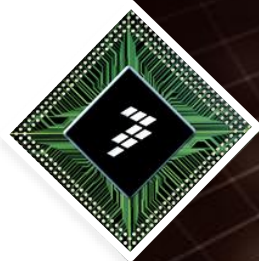




56F800EX new additional 32bit multiplication instruction test

Target on MC56F84xxx DSC

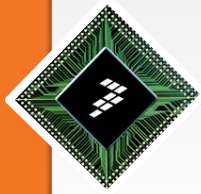
Moris Hsu
Field Application Engineer



Feb/25/2013

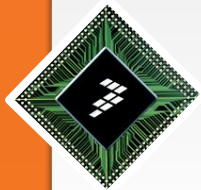
Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, C-Ware, the Energy Efficient Solutions logo, mobileGT, PowerQUICC, QorIQ, StarCore and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. BeeKit, BeeStack, ColdFire+, CoreNet, Flexis, Kinetic, MXC, Platform in a Package, Processor Expert, QorIQ Converge, Qoriva, QUICC Engine, SMARTMOS, TurboLink, VortiQa and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2011 Freescale Semiconductor, Inc.





Requirement

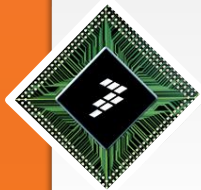
- How to use new 56F800EX additional 32bit multiplication instructions in CodeWarrior IDE?



56F800EX Core

- Core 56F800EX is a version 3 of core 56F800E.
- New Additional 32-Bit DSP56800EX Multiplication Instructions supported

Instruction	Parallel Move?	Description
IMAC32	—	Integer multiply-accumulate 32 bits
IMPY32	—	Integer multiply 32 bits x 32 bits → 32 bits
IMPY64	—	Integer multiply 32 bits x 32 bits → 64 bits
IMPY64UU	—	Unsigned integer multiply 32bits x 32 bits → 64 bits
MAC32	—	Fractional multiply-accumulate 32 bits x 32 bits → 32 bits
MPY32	—	Fractional multiply 32 bits x 32 bits → 32 bits
MPY64	—	Fractional multiply 32 bits x 32 bits → 64 bits



Test Environment

- Hardware:
 - TWR-56F8400 (MC56F84789 on board)
- Software:
 - CW for MCU V10.3
 - CW for MCU V10.5 / V10.6 (DSC compiler supports 56800EX new additional instructions natively)
 - Location of Long Long process source code: “c:\Freescale\CW for MCU V10.3\MCU\M56800E Support\runtime_56800E\I\”



CW V10.3 Configuration



Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, C-Ware, the Energy Efficient Solutions logo, mobileGT, PowerQUICC, QorIQ, StarCore and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. BeeStack, ColdFire+, CoreNet, Flexis, Kinetic, MXC, Platform in a Package, Processor Expert, QorIQ Converge, Qorivva, QUICC Engine, SMARTMOS, TurboLink, VortiQa and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2011 Freescale Semiconductor, Inc.

Compiler Setting

The screenshot shows the 'Properties for MC56F84xx_longlong_Allen_20130205' dialog box. The left pane shows a tree view with 'Settings' selected under 'C/C++ Build'. The right pane shows the 'Settings - MC56F84xx_longlong_Allen_20130205' dialog. The 'Language' category is selected in the tree view. The 'Other Flag' field contains the text '-v3 -flag slld'. A red text overlay reads 'Input -v3 -flag slld'.

Settings - MC56F84xx_longlong_Allen_20130205

- ANSI Strict
- ANSI Keywords Only
- Enums Always Int
- Use Unsigned Chars
- Require Function Prototypes
- Expand Trigraphs
- Legacy for-scoping
- Reuse Strings
- Pool Strings

Other Flag:

Input -v3 -flag slld

Compiler Setting

The screenshot shows the 'Properties for MC56F84xx_longlong_Allen_20130205' dialog box. The left sidebar contains a tree view with categories like Resource, Builders, C/C++ Build, C/C++ General, Processor Expert, and Run/Debug Settings. The 'Settings' folder under C/C++ Build is expanded. The main pane shows a tree view of settings for the DSC Compiler, with the 'Processor' folder selected and highlighted by a red box. The right pane displays the settings for the selected processor, including 'Hardware DO Loops' (set to 'No DO Loops (default)'), 'Small Program Model', 'Large Data Memory Model', 'Globals Live in Lower Memory', 'Zero-Initialized Globals Live in Data Instead of BSS', 'Segregate Data Section', 'Pad Pipeline for Debugger' (checked), 'Create Assembly Output' (checked and highlighted with a red box), and 'Generate Code for Profiling' (unchecked). Below these are dropdown menus for 'Check Inline Assembly for Pipeline' and 'Check C Source for Pipeline', both set to 'Not Detected (default)'. The dialog has 'OK' and 'Cancel' buttons at the bottom right.

Compiler Setting

Properties for MC56F84xx_longlong_Allen_20130205

Settings - MC56F84xx_longlong_Allen_20130205

Search User Paths (#include "...")

- "\${ProjDirPath}/Project_Headers"
- "\${MCUToolsBaseDir}/M56800E_Support/runtime_56800E/include"
- "\${ProjDirPath}/Generated_Code"
- "\${ProjDirPath}/Sources"

Search User Paths Recursively

Search System Paths (#include <...>)

- "\${MCUToolsBaseDir}/M56800E_Support/ms/MSL_C/MSL_Common/include"
- "\${MCUToolsBaseDir}/M56800E_Support/ms/MSL_C/DSP_56800E/inc"

Search System Paths Recursively

- "\${MCUToolsBaseDir}/M56800E_Support"

c:\Freescale\CW for MCU V10.3\MCU\M56800E
Support\ms\MSL_C\DSP_56800E\inc\

OK Cancel

Linker Setting

Properties for MC56F84xx_longlong_Allen_20130205

Settings - MC56F84xx_longlong_Allen_20130205

Configuration: MC56F84789_Internal_PFlash_SDM [Active] Manage Configurations...

Tool Settings | Build Steps | Build Artifact | Binary Parsers | Error Parsers | Build Tool Versions

- Global Settings
- DSC Linker
 - General
- DSC Compiler
 - Input
 - Access Paths
 - Warnings
 - Optimization
 - Processor
 - Language
- DSC Assembler
 - Input
 - General
 - Output
- DSC Preprocessor
 - Settings
- DSC Disassembler
 - Settings

Dead-Strip Unused Code

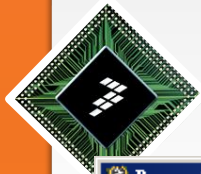
Suppress Link Warnings

Large Data Memory Model

Generates elf file for 56800EX core

Other Flags |

OK Cancel



Linker Setting

Properties for MC56F84xx_longlong_Allen_20130205

Settings - MC56F84xx_longlong_Allen_20130205

Output Type: Application

Output

- Generate Link Map
- List Unused Symbols in Map
- Show Transitive Closure in Map
- Annotate Byte Symbols in Map
- Generate ELF Symbol Table
- Sort by Address
- Generate Byte Addresses

Max S-Record Length: 80

S_Record EOL Character: DOS (\r\n)

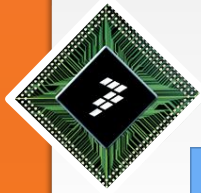
OK Cancel



CW V10.5 / V10.6 Configuration

Updated on Jun/2014





Compiler Setting

The screenshot shows the 'Properties for 84789_New Add Instruction' dialog box. The 'Settings' tab is active, displaying a tree view of compiler options. The 'Language' folder is selected. The 'Other Flag' field contains the text '-v3 -flag slld'. A red text overlay reads 'Input -v3 -flag slld'.

type filter text

- Resource
- Builders
 - C/C++ Build
 - Build Variables
 - Discovery Options
 - Environment
 - Logging
 - Settings
 - Tool Chain Editor
 - C/C++ General
 - Project References
 - Run/Debug Settings

Settings

- Global Settings
 - ANSI Strict
 - ANSI Keywords Only
 - Enums Always Int
 - Use Unsigned Chars
 - Require Function Prototypes
 - Expand Trigraphs
 - Legacy for-scoping
 - Reuse Strings
 - Pool Strings
- DSC Linker
 - Input
 - Link Order
 - General
 - Output
- DSC Compiler
 - Input
 - Access Paths
 - Warnings
 - Optimization
- Language**
- DSC Assembler
 - Input
 - General
 - Output
- DSC Preprocessor
 - Settings
- DSC Disassembler
 - Settings

Other Flag:

Input -v3 -flag slld

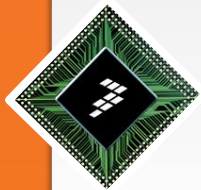
OK Cancel



Compiled C Code in CW V10.3



Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, C-Ware, the Energy Efficient Solutions logo, mobileGT, PowerQUICC, QorIQ, StarCore and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. BeeStack, ColdFire+, CoreNet, Flexis, Kinetis, MXC, Platform in a Package, Processor Expert, QorIQ Converge, Qorivva, QUICC Engine, SMARTMOS, TurboLink, VortiQa and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2011 Freescale Semiconductor, Inc.



32bit*32bit= 64bit with V3 function call, success

- unsigned long a, b;
- long long x = 0;

- a= 0x12345678;
- b= 0x34127856;
- Bit1_SetVal();
- **x = V3_LL_mult_int(a,b);**
- Bit1_ClrVal();

- adda #<8,SP
- move.l #12345678,X:>Fa // a= 0x12345678;
- move.l #34127856,X:>Fb // b= 0x34127856;

- move.l #e241,R0 // Bit1_SetVal();
- bfset #1,X:(R0)

- move.l X:>Fb,B // x = V3_LL_mult_int(a,b);
- move.l X:>Fa,A
- **jsr >FV3ARTIMPY64**
- move.l B10,X:(SP)
- move.l A10,X:(SP-2)
- move.l X:(SP),A
- move.l X:(SP-2),B
- move.l A10,X:(SP-4)
- move.l B10,X:(SP-6)
- move.l X:(SP-4),A
- move.l X:(SP-6),B
- move.l B10,X:>Fx
- move.l A10,X:>Fx+2

- move.l #e241,R0 // Bit1_ClrVal();
- bfcir #1,X:(R0)

32bit*32bit= 64bit with V3 function call, success

The screenshot displays the CodeWarrior Development Studio interface during a debug session. The main window shows the source code for `ProcessorExpert.c` with the following relevant lines:

```

65 #endif
66
67 #if 1
68     a = 0x12345678;
69     b = 0x34127856; // a*b should be 0x3b3f1cd1a8d4c50
70     Bit1_SetVal();
71     x = V3_LL_mult_int(a,b); // x is correct
72     Bit1_ClrVal();
73 #endif
74
75 #if 1
76     a = 0x87654321;
77     b = 0x56783412; // a*b should be 0x2dbb978ea4396c52
78     Bit1_SetVal();

```

The **Variables** window shows the state of variables `Fx`, `Fb`, and `Fa`:

Name	Value	Location
Fx	0x3b3f1cd1a8d4c50	0x000008>Data Word
Fb	0x34127856	0x000010>Data Word
Fa	0x12345678	0x000012>Data Word

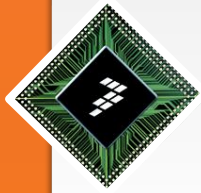
The **Disassembler** window shows the assembly instructions corresponding to the source code:

```

000226: move.l #0x00000a, A10
72      Bit1_ClrVal();
000228: move.l #0xe241, R0
00022b: bfc1r #1, X:(R0)
76      a = 0x87654321;
00022d: move.l #0x87654321, X:0x000012
77      b = 0x56783412; // a*b should be 0x2dbb978ea4396c52
000231: move.l #0x56783412, X:0x000010
78      Bit1_SetVal();
000235: move.l #0xe241, R0

```

The **Commander** window shows project creation options, and the **Problems** window is empty, indicating no errors.



64bit*64bit= 64bit, success

- long long c=0, d= 0, result = 0;
- Bit1_SetVal();
- c= 0x43218765;
- d= 0x78561234;
- **result = c * d;**
- Bit1_ClrVal();

```
• move.l  #0x241,R0      // Bit1_SetVal();
• bfc     #1,X:(R0)

• move.l  #0x43218765,B  // c= 0x43218765;
• move.l  #>>0,A        //
• move.l  B10,X:>Fc      //
• move.l  A10,X:>Fc+2    //
• move.l  #0x78561234,B  // d= 0x78561234;
• move.l  #>>0,A        //
• move.l  B10,X:>Fd      //
• move.l  A10,X:>Fd+2    //
• adda   #<8,SP
• move.l  X:>Fc,B
• move.l  X:>Fc+2,A
• move.l  B10,X:(SP-2)
• move.l  A10,X:(SP)
• move.l  X:>Fd,B
• move.l  X:>Fd+2,A
• move.l  B10,X:(SP-6)
• move.l  A10,X:(SP-4)
• jsr    >ARTMULLL64
• suba   #<8,SP
• move.l  A10,X:>Fresult // result = c * d
• move.l  B10,X:>Fresult+2 //

• move.l  #0x241,R0      // Bit1_ClrVal();
• bfc     #1,X:(R0)
```


64bit*64bit= 64bit, success

Debug - MC56F84xx_longlong_Allen_20130205/Sources/ProcessorExpert.c - CodeWarrior Development Studio

File Edit Search Project Run RTCS MQX MQX Tools PEMicro Window Help

Debug

MC56F84xx_longlong_MC56F84789_Internal_PFlash_SDM_OSJTAG [CodeWarrior Download]

DSC, MC56F84xx_longlong.elf (Suspended)

Thread [ID: 0x0] (Suspended)

2 Fmain() ProcessorExpert.c:101 C

1 Finit_56800_() 56F83x_init.asm

D:\CW Workspace V10.3\MC56F84xx_lo

intrinsic_56800EX.h ProcessorExpert.c

```
91 #endif
92
93 #if 1
94 // Bit1_ClrVal();
95 Bit1_SetVal();
96 c= 0x43218765;
97 d= 0x78561234; // c*d should be 0x1f8e4980d2429a84
98
99 result = c * d;
100 //result = (long long) (a * b);
101 Bit1_ClrVal();
102 // Bit1_SetVal(); //1.62us
103
104 #endif
```

Name	Value
"result"	0x1f8e4980d2429a84
"c"	0x43218765
"d"	0x78561234
+ Add new expression	

```
000282: move.l #0xe241,R0
000285: bfc1r #1,X:(R0)
232 for(;;) {}
000287: bra Fmain+0x7f (0x287) ; 0x000143
FCpu_Interrupt:
54 {
000288: alignsp
55 asm(DEBUGHLT); /* Halt the cc
```

Commander

Project Creation

- Import project
- Import example project
- Import MCU executable file
- New MCU project
- New MQX-Lite project

Build/Debug

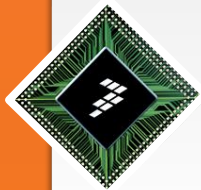
- Build (All)
- Clean (All)
- Debug

Settings

- Project s
- Build se
- Debug s

Problems Console Progress

DSC, MC56F84xx_longlong.elf



32bit*32bit= 64bit, **not** success

- unsigned long a, b;
 - long long x = 0;

 - a= 0x87654321;
 - b= 0x56783412;
 - Bit1_SetVal();
 - **x = (long long) (a * b);**
 - Bit1_ClrVal();
- move.l #\$87654321,X:>Fa // a= 0x87654321;
 - move.l #\$56783412,X:>Fb // b= 0x56783412;

 - move.l #\$e241,R0 // Bit1_SetVal();
 - bfset #1,X:(R0)

 - move.l X:>Fa,B
 - move.l X:>Fb,A
 - **impyuu A1,B0,Y**
 - **imacuu A0,B1,Y**
 - **impyuu A0,B0,B**
 - **add Y0,B**
 - **move.l #>>0,A**
 - move.l B10,X:>Fx
 - move.l A10,X:>Fx+2

 - move.l #\$e241,R0 // Bit1_ClrVal();
 - bfclr #1,X:(R0)

Only low 32bit result can be saved!



32bit*32bit= 64bit, not success

Debug - MC56F84xx_longlong_Allen_20130205/Sources/ProcessorExpert.c - CodeWarrior Development Studio

File Edit Search Project Run RTCS MQX MQX Tools PEMicro Window Help

Debug

MC56F84xx_longlong_MC56F84789_Internal_PFlash_SDM_OSJTAG [CodeWarrior Download]

DSC, MC56F84xx_longlong.elf (Suspended)

Thread [ID: 0x0] (Suspended)

2 Fmain() ProcessorExpert.c:82

1 Finit_56800_() 56F83x_init.a

D:\CW Workspace V10.3\MC56F84xx

intrinsic_56800EX.h ProcessorExpert

```

72 Bit1_ClrVal();
73 #endif
74
75 #if 1
76 a = 0x87654321;
77 b = 0x56783412; // a*b should be 0x2dbb978ea4396c52
78 Bit1_SetVal();
79 //d = a*b; // only low dword(0xa4396c52) can be saved to x
80 //x = ((long long) a) * ((long long) b);
81 x = (long long) (a * b);
82 Bit1_ClrVal();
83 #endif
84
85 #if 0

```

Name	Value	Location
Fx	0xa4396c52	0x000008*Data Word
Fb	0x56783412	0x000010*Data Word
Fa	0x87654321	0x000012*Data Word

Name	Value	Location
Fx	0xa4396c52	0x000008*Data Word
Fb	0x56783412	0x000010*Data Word
Fa	0x87654321	0x000012*Data Word

```

000247: move.l #10, X:0x000008
82 Bit1_ClrVal();
000249: move.l #0xe241, R0
00024c: bfc1r #1, X:(R0)
96 Bit1_SetVal();
00024e: move.l #0xe241, R0
000251: bfset #1, X:(R0)
97 c = 0x43218765;
000253: move.l #0x43218765, B
000256: move.l #>>0, A

```

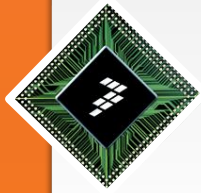
Commander

- Project Creation
 - Import project
 - Import example project
 - Import MCU executable file
 - New MCU project
 - New MQX-Lite project
- Build/Debug
 - Build (All)
 - Clean (All)
 - Debug
- Settings
 - Project s
 - Build se
 - Debug s

Problems Console Progress

DSC, MC56F84xx_longlong.elf

Writable Smart Insert 82 : 1



32bit*32bit= 64bit, success

- unsigned long a, b;
- long long x = 0;

- a= 0x87654321;
- b= 0x56783412;
- Bit1_SetVal();
- **x = ((long long) a) * ((long long) b);**
- Bit1_ClrVal();

**Compare with 64bit*64bit=64bit,
this will save some instructions,
due to there is no variable
initiation actions.**

- move.l #\$87654321,X:>Fa // a= 0x87654321;
- move.l #\$56783412,X:>Fb // b= 0x56783412;

- move.l #\$e241,R0 // Bit1_SetVal();
- bfcset #1,X:(R0)

- adda #<8,SP
- move.l X:>Fa,B
- move.l #>>0,A
- move.l B10,X:(SP-2)
- move.l A10,X:(SP)
- move.l X:>Fb,B
- move.l #>>0,A
- move.l B10,X:(SP-6)
- move.l A10,X:(SP-4)
- jsr >ARTMULLL64
- suba #<8,SP
- move.l A10,X:>Fb
- move.l B10,X:>Fb+2 // x = ((long long) a) * ((long long) b);

- move.l #\$e241,R0 // Bit1_ClrVal();
- bfcclr #1,X:(R0)



32bit*32bit= 64bit, success

Debug - MC56F84xx_longlong_Allen_20130205/Sources/ProcessorExpert.c - CodeWarrior Development Studio

File Edit Search Project Run RTCS MQX MQX Tools PEMicro Window Help

Debug Variables Registers Breakpoints Memory Modules Memory Brows Expressions

Name	Value	Location
Fx	0x2dbb978ea4396c52	0x000008>Data Word
Fb	0x56783412	0x000010>Data Word
Fa	0x87654321	0x000012>Data Word

MC56F84xx_longlong_MC56F84789_Internal_PFlash_SDM_OSJTAG [CodeWarrior Download]

DSC, MC56F84xx_longlong.elf (Suspended)

Thread [ID: 0x0] (Suspended)

- 2 Fmain() ProcessorExpert.c:81 0x000:
- 1 Finit_56800_() 56F83x_init.asm:148

D:\CW Workspace V10.3\MC56F84xx_longlo

intrinsic_56800EX.h ProcessorExpert.c

```

70 Bit1_SetVal();
71 x = V3_LL_mult_int(a,b); /
72 Bit1_clrVal();
73 #endif
74
75 #if 1
76 a= 0x87654321;
77 b= 0x56783412; // a*b should be 0x2dbb978ea4396c52
78 Bit1_SetVal();
79 //d = a*b; // only low dword(0xa4396c52) can be saved to x
80 x = ((long long) a) * ((long long) b);
81 Bit1_clrVal();
82 #endif
83

```

81 Bit1_clrVal();

```

000250: move.l #0xe241,R0
000253: bfc1r #1,X:(R0)
95 Bit1_SetVal();
000255: move.l #0xe241,R0
000258: bfset #1,X:(R0)
96 c= 0x43218765;
00025a: move.l #0x43218765,B
00025d: move.l #>>0,A

```

Commander Build/Debug Settings Problems Console Progress

Project Creation Build/Debug Settings

- Import project
- Import example project
- Import MCU executable file
- New MCU project
- New MQX-Lite project
- Build (All)
- Clean (All)
- Debug
- Project s
- Build se
- Debug s

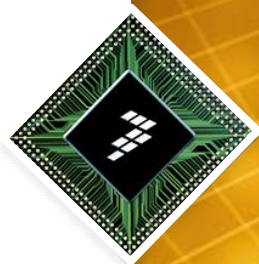
DSC, MC56F84xx_longlong.elf

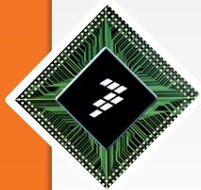
Writable Smart Insert 81 : 1



Compiled C Code in CW V10.5 / V10.6

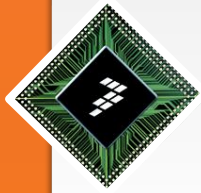
Updated on Jun/2014





Signed 32bit*32bit= 64bit, success

- **long c=0, d= 0;**
- **long long result= 0;**
- c= 0x43218765;
- d= 0x78561234;
- result = **c * d;**
- move.l #0x43218765,X:0x000000
- move.l #0x78561234,X:0x000000
- move.l X:0x000000,B ; Fc
- move.l X:0x000000,A ; Fd
- **impy64 A,B,A**
- move.l Y,X:0x000000 ; Fresult
- move.l A10,X:0x000000 ; Fresult+2



Signed 32bit*32bit= 64bit, success

Debug - 84789_New Add Instruction/Sources/main.c - CodeWarrior Development Studio

File Edit Source Refactor Search Project RTCS MQX MQXTools Run Window Help

Quick Access C/C++ Debug

Debug 84789_New Add Instruction_FLASH_SDM_FSL USB TAP [CodeWarrior]
DSC, 84789_NewAddInstruction.elf (Suspended)
Thread [ID: 0x0] (Suspended)
2 Fmain() main.c:79 0x000281
1 Finit_MC56F847xx_ISR_HW_RESET() MC56F847xx_init.asm:14
D:\Workspace_CW V10.5\84789_New Add Instruction\FLASH_SDM\

main.c

```
printf("\n\n... program done.\n");  
c = 0x43218765;  
d = 0x78561234; // c*d should be  
result = c * d;  
a = 0x87654321;  
b = 0x56783412; // a*b should be 0x2dbb978ea4396c52
```

Variables

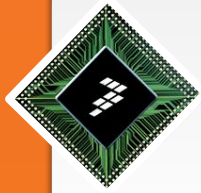
Name	Value
X:Y "c"	0x43218765
X:Y "d"	0x78561234
X:Y "result"	0x1f8e4980d2429a84
X:Y "a"	0x00000000
X:Y "b"	0x00000000
X:Y "v"	0x0000000000000000

Registers

Name	Value
X:Y "c"	0x43218765
X:Y "d"	0x78561234
X:Y "result"	0x1f8e4980d2429a84

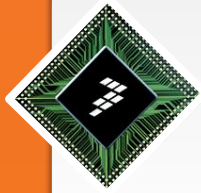
Console

```
DSC, 84789_NewAddInstruction.elf  
Array = 4 6 7 1 2 3 4 12 4 5  
Array = 1 2 3 4 4 4 5 6 7 12  
... program done.
```

Unsigned 32bit*32bit= 64bit, success

- `unsigned long a=0, b= 0;`
- `long long x= 0;`
- `a= 0x87654321;`
- `b= 0x56783412;`
- `x = a * b;`
- `move.l #0x87654321,X:0x000000`
- `move.l #0x56783412,X:0x000000`
- `move.l X:0x000000,B ; Fa`
- `move.l X:0x000000,A ; Fb`
- `impy64uu A,B,A`
- `move.l Y,X:0x000000 ; Fx`
- `move.l A10,X:0x000000 ; Fx+2`



Unsigned 32bit*32bit= 64bit, success

Debug - 84789_New Add Instruction/Sources/main.c - CodeWarrior Development Studio

File Edit Source Refactor Search Project RTCS MQX MQX Tools Run Window Help

Quick Access C/C++ Debug

Debug 84789_New Add Instruction_FLASH_SDM_FSL USB TAP [CodeWarrior]
DSC, 84789_NewAddInstruction.elf (Suspended)
Thread [ID: 0x0] (Suspended)
2 Fmain() main.c:91 0x000292
1 Finit_MC56F847xx_ISR_HW_RESET() MC56F847xx_init.asm:14
D:\Workspace_CW V10.5\84789_New Add Instruction\FLASH_SDM\

main.c

```
c = 0x43218765;  
d = 0x78561234; // c*d should be  
  
result = c * d;  
  
a = 0x87654321;  
b = 0x56783412; // a*b should be 0  
  
x = a * b;  
  
/*
```

Variables Expressions Registers Memory Modules

Name	Value
"result"	0x1f8e4980d2429a84
"a"	0x87654321
"b"	0x56783412
"x"	0x2dbb978ea4396c52

Commander

- Project Creation
- Import project
- Import example project
- Import MCU executable file
- Build
- Build
- Clear
- Debug

Console Problems

DSC, 84789_NewAddInstruction.elf

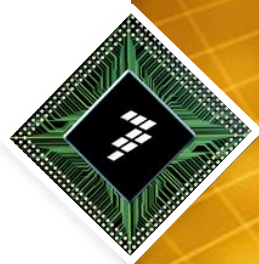
```
Array = 4 6 7 1 2 3 4 12 4 5  
Array = 1 2 3 4 4 4 5 6 7 12  
... program done.
```

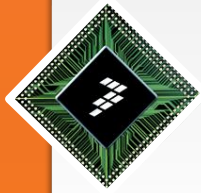
1:7



Long Long(LL) Calculation Function Call

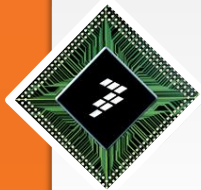
File Location: "c:\Freescale\CW for MCU V10.3\MCUM56800E
Support\runtime_56800E\IARTMULLL64.asm"





V3ARTIMPY64 source code – New Multiplication

```
• ;=====
• ; FUNCTION: V3ARTIMPY64
• ; DESCRIPTION:          56800EX long long multiplication.(56800EX instructions intrinsic functions support)
• ;                      Multiply two 32-bit integer values generating a signed 64-bit integer result.
• ;                      56800EX instruction:          IMPY64 FF1,FF1,FF:Y - Integer Multiply 32x32 = 64 bit; save higher (in Acc),
• lower (in Y)
• ;
• V3ARTIMPY64:
• FV3ARTIMPY64:
•     adda #<2,SP
•     move.l C10,X:(SP)
•
•     impy64 A,B,C
•
•     adda #<4,SP
•     move.l C10,X:(SP)
•     move.l Y,X:(SP-2)
•     move.l X:(SP),B
•     move.l X:(SP-2),A
•     suba #4,SP
•
•     move.l X:(SP)-,C
•     rts
• end_FV3ARTIMPY64:
```



ARTMULLL64 source code – Old Multiplication

- ;=====
- ;=====
- ; FUNCTION: ARTMULLL64
- ; DESCRIPTION: Long long multiplication.
- ;
- ; INPUT: a = x, b = y
- ; OUTPUT: a = result = x * y
- ;
- **ARTMULLL64:**
- FARTMULLL64:
- ;ARTMULULL64:
- ;FARTMULULL64:

- adda #<2,SP
- move.l C10,X:(SP)+
- move.l D10,X:(SP)

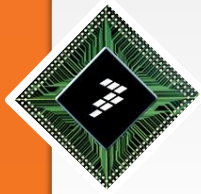
- adda #<14,SP
- move.l X:(SP-22),A ; 1
- ;move.l X:(SP-20),C ; 0
- move.l X:(SP-26),B ; 3
- ;move.l X:(SP-24),D ; 2

- impyuu A0,B0,D
- impyuu A1,B0,Y
- move.w B1,B0
- impyuu A0,B0,C
- add C,Y
- move.w Y1,B0
- move.w #\$0,B1
- jcc L1
- move.w #\$1,B1
- L1:
- move.w #\$1,X:(SP-2) ; to keep carry flag - clear up part of D
- tfr Y0,Y
- add D,Y
- tfr Y,D
- jcs L2
- move.w #\$0,X:(SP-2) ; to keep carry flag - set up part of D - used in the end test



ARTMULLL64 source code – Old Multiplication(cont'd)

- L2:
- `move.l D10,X:(SP-4) ; d5`
- `move.l B10,X:(SP-8) ; d14`
- `move.l X:(SP-22),A ; 1`
- `move.l X:(SP-24),B ; 2`
- `impyuu A0,B0,C ; d10`
- `move.l X:(SP-20),A ; 0`
- `move.l X:(SP-26),B ; 3`
- `impyuu B0,A0,D ; d15`
- `add C,D ; d15`
- `tfr D,Y`
- `move.l X:(SP-22),A ; 1`
- `move.l X:(SP-26),B ; 3`
- `move.w A1,A0`
- `imacuu A0,B1,Y`
- `move.l X:(SP-8),C`
- `add C,Y ; d15`
- `move.l Y,X:(SP-8)`
- `move.l X:(SP-22),B ; 1`
- `move.l X:(SP-24),A ; 2`
- `impysu A1,B0,Y`
- `imacuu A0,B1,Y`
- `move.l X:(SP-20),B ; 0`
- `move.l X:(SP-26),A ; 3`
- `tfr A,C`
- `imacuu B0,C1,Y`
- `move.l X:(SP-26),A ; 3`
- `imacus A0,B1,Y`
- `asll.l #$10,Y`
- `move.l X:(SP-8),D`
- `bfclr #$01,SR`
- `add D,Y ; Y = d4`
- `move.l X:(SP-4),A`
- `move.w X:(SP-2),B0`



ARTMULLL64 source code – Old Multiplication(cont'd)

- `bftsth #0001,B0`
- `jcc L3`
- `move.l #1,C`
- `add C,Y`
- `L3:`
- `tfr Y,B`
- `sxt.l B`
- `suba #<14,SP`

- `move.l X:(SP)-,D`
- `move.l X:(SP)-,C`

- `rts`

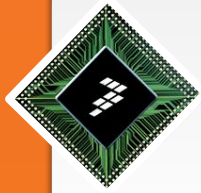
- `end_FARTMULLL64:`
- `;end_FARTMULULL64:`



Execution time in CW V10.3

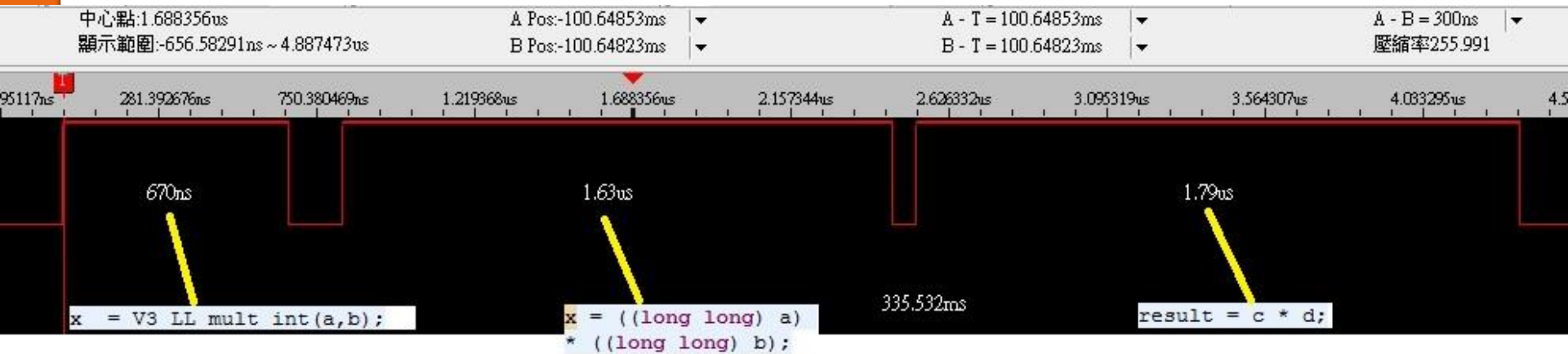


Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, C-Ware, the Energy Efficient Solutions logo, mobileGT, PowerQUICC, QorIQ, StarCore and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. BeeStack, ColdFire+, CoreNet, Flexis, Kinetic, MXC, Platform in a Package, Processor Expert, QorIQ Converge, Qorivva, QUICC Engine, SMARTMOS, TurboLink, VortiQa and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2011 Freescale Semiconductor, Inc.



Execution time

- It takes **670ns** to run “FV3ARTIMPY64” for 32bit*32bit=64bit.
- It takes **1.63us** to run “ARTMULLL64” for 32bit*32bit= 64bit.
- It takes **1.79us** to run “ARTMULLL64” for 64bit*64bit= 64bit.

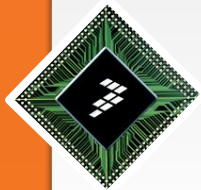




Conclusion



Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, C-Ware, the Energy Efficient Solutions logo, mobileGT, PowerQUICC, QorIQ, StarCore and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. BeeStack, ColdFire+, CoreNet, Flexis, Kinetis, MXC, Platform in a Package, Processor Expert, QorIQ Converge, Qorivva, QUICC Engine, SMARTMOS, TurboLink, VortiQa and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2011 Freescale Semiconductor, Inc.



Conclusion

- To use new version of LL calculation function call, refer to “C:\Freescale\CW for MCU V10.3\MCU\M56800E Support\ms\MSL_C\DSP_56800E\inc\intrinsics_56800EX.h”
 - The multiplication calculation which uses new additional instruction does more simple than the old one
- To achieve 64bit result calculation, the multiplicand and multiplier must be 64bit if no V3 LL calculation function call be used.
 - Compiler & Linker options must be configured first.
- If data type of multiplicand and multiplier are not 64bit, data type conversion is required.
 - Note: found this C code can save some instructions comparing with 64bit*64bit= 64bit, due to there is no variable initiation.
- The DSC compiler in CW V10.5 and V10.6 does support 56800EX new additional instructions natively. (Updated on Jun/2014)

