*** EVALUATION ***
ANSI-C/cC++ Compiler for HC12 V-5.0.40 Build 10020, Jan 21 2010

```
    1:  /*****************************************************************************
    2:     PMF.c
    3:
    4:  *****************************************************************************/
    5:
    6:  // Include Files
    7:  #include <MC9S12E128.h>          // Derivative information
    8:  #include <stdio.h>
    9:
   10:  #include "PMF.h"
   11:  #include "ATD.h"
   12:  #include "Buffers.h"
   13:  #include "SCI0.h"
   14:  #include "Datatypes.h"
   15:  #include "IO.h"
   16:
   17:
   18:  // Constants
   19:  // One-quarter of a sine wave (90 degrees). Only first 126 values are used.
   20:  const byte Sine[128] =
   21:  {
   22:    0,  3,  5,  8, 10, 13, 15, 18, 20, 23, 25, 28, 30, 33, 35, 37,
   23:   40, 42, 45, 47, 50, 52, 55, 57, 59, 62, 64, 67, 69, 71, 74, 76,
   24:   78, 81, 83, 85, 87, 90, 92, 94, 96, 99,101,103,105,107,109,111,
   25:  113,116,118,120,122,124,126,127,129,131,133,135,137,139,141,142,
   26:  144,146,148,149,151,152,154,156,157,159,160,162,163,165,166,168,
   27:  169,170,172,173,174,175,176,178,179,180,181,182,183,184,185,186,
   28:  187,188,189,189,190,191,192,192,193,194,194,195,195,196,196,197,
   29:  197,198,198,198,199,199,199,199,200,200,200,200,200,200,200,200
   30:  };
   31:
   32:
   33:  // Global Variables
   34:  BOOL     Phase;               // Positive and negative half-cycle (2 per cycle)
   35:  BOOL     Slope;               // Positive and negative slope (2 positive and 2 negative per cycle)
   36:  word     Angle;               // Phase angle of a sign wave
   37:  byte     Amp;                 // Amplitude of sine wave generated
   38:  int      Adj;                 // Correction for zero crossing
   39:
   40:  word     SineValue;           // Value of the sine wave from table
   41:
   42:
   43:  /*****************************************************************************/
   44:  #pragma CODE_SEG NON_BANKED
   45:  interrupt void PMFA_Reload_Int(void)
   46:  {
```

*** EVALUATION ***

Function: PMFA_Reload_Int
Source : C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources\PMF.c
Options : -CPUHCS12 -Env"GENPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12;C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin;C:\Users\Public\Projects\9S12Float\CodeHC9S12\prm;C:\Users\Public\Projects\9S12Float\CodeHC9S12\cmd;C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\lib;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\src;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -Env"LIBPATH=C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -EnvOBJPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -EnvTEXTPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -Lasm=C:\Users\Public\Projects\9S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o.lst -Ms -ObjN=C:\Users\Public\Projects\9S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o -WmsgSd1106

```
  47:        if (Phase)                // Positive half-cycle
0000 fc0000       [3]     LDD   Phase
0003 273f         [3/1]   BEQ   *+65 ;abs = 0044
  48:        {
  49:            if (Slope)     // Positive slope
0005 fc0000       [3]     LDD   Slope
0008 2727         [3/1]   BEQ   *+41 ;abs = 0031
  50:            {
  51:                SineValue = Sine[Angle++];       // Get sine wave value from table
000a fe0000       [3]     LDX   Angle
000d 1901         [2]     LEAY  1,X
000f 160000       [4]     JSR   PMFA_Reload_Int:0x009c
  52:                PMFVAL0 = (Modulus/2)+(SineValue*Amp/255);       // Calculate PWM
0012 1ae0c8       [2]     LEAX  200,X
0015 7e0000       [3]     STX   _PMFVAL0
  53:                if (Angle == 125)
0018 8d007d       [2]     CPY   #125
001b 2638         [3/1]   BNE   *+58 ;abs = 0055
  54:                {
  55:                 switch(Adj)
001d fc0000       [3]     LDD   Adj
0020 83ffff       [2]     SUBD  #65535
0023 160000       [4]     JSR   _CASE_CHECKED_BYTE
0026 03           DC.B  3
0027 6b           DC.B  107
0028 06           DC.B  6
0029 03           DC.B  3
002a 4b           DC.B  75
  56:                {
  57:                        case +1:                // Add an extra angle cycle
  58:                            Angle--;        // Will do 124 over again
  59:                            Adj = 0;
  60:                            break;
  61:
  62:                        case 0:                 // In sync, no need to adjust
  63:                            Slope = 0;      // Change to negative slope
```

```
002b c7           [1]     CLRB
 64:                                              break;
002c 2039         [3]     BRA   *+59 ;abs = 0067
 65:
 66:                                        case -1:                // Subtract one angle cycle
 67:                                              Slope = 0;        // Change to negative slope
002e c7           [1]     CLRB
 68:                                              Angle--;          // Will not do 125
 69:                                              Adj = 0;
 70:                                              break;
002f 203e         [3]     BRA   *+64 ;abs = 006f
 71:                                         }
 72:                                       }
 73:                   }
 74:                   else            // Negative slope
 75:                   {
 76:                                       SineValue = Sine[Angle--];        // Get sine wave value from table
0031 fe0000       [3]     LDX   Angle
0034 191f         [2]     LEAY  -1,X
0036 0764         [4]     BSR   *+102 ;abs = 009c
 77:                                       PMFVAL0 = (Modulus/2)+(SineValue*Amp/255);        // Calculate PWM
0038 1ae0c8       [2]     LEAX  200,X
003b 7e0000       [3]     STX   _PMFVAL0
 78:                                       if (Angle == 0)
003e 046652       [3]     TBNE  Y,*+85 ;abs = 0093
 79:                                        Phase = 0;       // Change to negative half-cycle
0041 c7           [1]     CLRB
0042 204b         [3]     BRA   *+77 ;abs = 008f
 80:                   }
 81:           }
 82:           else      // Negative half-cycle
 83:           {
 84:                   if (!Slope)     // Negative slope
0044 fc0000       [3]     LDD   Slope
0047 2638         [3/1]   BNE   *+58 ;abs = 0081
 85:                   {
 86:                                       SineValue = Sine[Angle++];        // Get sine wave value from table
0049 fe0000       [3]     LDX   Angle
004c 1901         [2]     LEAY  1,X
004e 074c         [4]     BSR   *+78 ;abs = 009c
 87:                                       PMFVAL0 = (Modulus/2)-(SineValue*Amp/255);        // Calculate PWM
0050 0762         [4]     BSR   *+100 ;abs = 00b4
 88:                                       if (Angle == 125)
0052 8d007d       [2]     CPY   #125
0055 263c         [3/1]   BNE   *+62 ;abs = 0093
 89:                                       {
 90:                                        switch(Adj)
0057 fc0000       [3]     LDD   Adj
005a 83ffff       [2]     SUBD  #65535
```

```
005d 160000       [4]     JSR   _CASE_CHECKED_BYTE
0060 03           DC.B  3
0061 31           DC.B  49
0062 0b           DC.B  11
0063 03           DC.B  3
0064 11           DC.B  17
 91:                                {
 92:                                        case +1:                // Add an extra angle cycle
 93:                                                Angle--;        // Will do 124 over again
 94:                                                Adj = 0;
 95:                                                break;
 96:
 97:                                        case 0:                 // In sync, no need to adjust
 98:                                                Slope = 1;      // Change to positive slope
0065 c601         [1]     LDAB  #1
 99:                                                break;
0067 87           [1]     CLRA
0068 7c0000       [3]     STD   Slope
006b 2026         [3]     BRA   *+40 ;abs = 0093
100:
101:                                       case -1:                 // Subtract one angle cycle
102:                                               Slope = 1;       // Change to positive slope
006d c601         [1]     LDAB  #1
103:                                               Angle--;         // Will not do 125
104:                                               Adj = 0;
105:                                               break;
006f 87           [1]     CLRA
0070 7c0000       [3]     STD   Slope
0073 fe0000       [3]     LDX   Angle
0076 09           [1]     DEX
0077 7e0000       [3]     STX   Angle
007a c7           [1]     CLRB
007b 87           [1]     CLRA
007c 7c0000       [3]     STD   Adj
007f 2012         [3]     BRA   *+20 ;abs = 0093
106:                               }
107:                              }
108:                   }
109:                   else            // Positive slope
110:                   {
111:                              SineValue = Sine[Angle--];       // Get sine wave value from table
0081 fe0000       [3]     LDX   Angle
0084 191f         [2]     LEAY  -1,X
0086 0714         [4]     BSR   *+22 ;abs = 009c
112:                              PMFVAL0 = (Modulus/2)-(SineValue*Amp/255);       // Calculate PWM
0088 072a         [4]     BSR   *+44 ;abs = 00b4
113:                              if (Angle == 0)
008a 046606       [3]     TBNE  Y,*+9 ;abs = 0093
114:                               Phase = 1;      // Change to positive half-cycle
```

```
  008d c601          [1]     LDAB  #1
  008f 87            [1]     CLRA
  0090 7c0000        [3]     STD   Phase
  115:                   }
  116:         }
  117:         PMFENCA_LDOKA = 1;        // Allow PWM register load
  0093 1c000002      [4]     BSET  _PMFENCA,#2
  118:         PMFFQCA_PWMRFA = 1;       // Clear PWM Reload Flag A
  0097 1c000001      [4]     BSET  _PMFFQCA,#1
  119:  }
  009b 0b            [8]     RTI
  009c 7d0000        [3]     STY   Angle
  009f b751          [1]     TFR   X,B
  00a1 ce0000        [2]     LDX   #Sine
  00a4 e6e5          [3]     LDAB  B,X
  00a6 87            [1]     CLRA
  00a7 7c0000        [3]     STD   SineValue
  00aa b60000        [3]     LDAA  Amp
  00ad 12            [1]     MUL
  00ae ce00ff        [2]     LDX   #255
  00b1 1810          [12]    IDIV
  00b3 3d            [5]     RTS
  00b4 c6c8          [1]     LDAB  #200
  00b6 87            [1]     CLRA
  00b7 34            [2]     PSHX
  00b8 a3b1          [3]     SUBD  2,SP+
  00ba 7c0000        [3]     STD   _PMFVAL0
  00bd 3d            [5]     RTS
  120:  #pragma CODE_SEG DEFAULT
  121:
  122:
  123:  /*****************************************************************************/
  124:  #pragma CODE_SEG NON_BANKED
  125:  interrupt void PMFB_Reload_Int(void)
  126:  {
*** EVALUATION ***

Function: PMFB_Reload_Int
Source  : C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources\PMF.c
Options : -CPUHCS12 -Env"GENPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12;C:\Users\Public\Projects\9S12Float\CodeHC9S12\
bin;C:\Users\Public\Projects\9S12Float\CodeHC9S12\prm;C:\Users\Public\Projects\9S12Float\CodeHC9S12\cmd;C:\Users\Public\
Projects\9S12Float\CodeHC9S12\Sources;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\lib;C:\Program Files\
Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\src;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include"
-Env"LIBPATH=C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -EnvOBJPATH=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\bin -EnvTEXTPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -Lasm=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o.lst -Ms -ObjN=C:\Users\Public\Projects\9S12Float\CodeHC9S12\
CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o -WmsgSd1106

  127:        //PMFENCB_LDOKB = 1;      // Allow PWM register load
```

```
  128:        PMFFQCB_PWMRFB = 1;       // Clear PWM Reload Flag B
  0000 1c000001     [4]     BSET  _PMFFQCB,#1
  129:        Toggle_LED();
  0004 160000       [4]     JSR   Toggle_LED
  130:  }
  0007 0b           [8]     RTI
  131:  #pragma CODE_SEG DEFAULT
  132:
  133:
  134:  /*****************************************************************************/
  135:  #pragma CODE_SEG NON_BANKED
  136:  interrupt void PMFC_Reload_Int(void)
  137:  {
*** EVALUATION ***

Function: PMFC_Reload_Int
Source  : C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources\PMF.c
Options : -CPUHCS12 -Env"GENPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12;C:\Users\Public\Projects\9S12Float\CodeHC9S12\
bin;C:\Users\Public\Projects\9S12Float\CodeHC9S12\prm;C:\Users\Public\Projects\9S12Float\CodeHC9S12\cmd;C:\Users\Public\
Projects\9S12Float\CodeHC9S12\Sources;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\lib;C:\Program Files\
Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\src;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include"
-Env"LIBPATH=C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -EnvOBJPATH=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\bin -EnvTEXTPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -Lasm=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o.lst -Ms -ObjN=C:\Users\Public\Projects\9S12Float\CodeHC9S12\
CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o -WmsgSd1106

  138:        //PMFENCC_LDOKC = 1;      // Allow PWM register load
  139:        PMFFQCC_PWMRFC = 1;       // Clear PWM Reload Flag C
  0000 1c000001     [4]     BSET  _PMFFQCC,#1
  140:        Toggle_LED();
  0004 160000       [4]     JSR   Toggle_LED
  141:  }
  0007 0b           [8]     RTI
  142:  #pragma CODE_SEG DEFAULT
  143:
  144:
  145:
  146:  /*****************************************************************************/
  147:  void InitPMF(void)
  148:  {
*** EVALUATION ***

Function: InitPMF
Source  : C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources\PMF.c
Options : -CPUHCS12 -Env"GENPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12;C:\Users\Public\Projects\9S12Float\CodeHC9S12\
bin;C:\Users\Public\Projects\9S12Float\CodeHC9S12\prm;C:\Users\Public\Projects\9S12Float\CodeHC9S12\cmd;C:\Users\Public\
Projects\9S12Float\CodeHC9S12\Sources;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\lib;C:\Program Files\
Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\src;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include"
-Env"LIBPATH=C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -EnvOBJPATH=C:\Users\Public\Projects\9
```

```
S12Float\CodeHC9S12\bin -EnvTEXTPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -Lasm=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o.lst -Ms -ObjN=C:\Users\Public\Projects\9S12Float\CodeHC9S12\
CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o -WmsgSd1106

  149:  /*  */
  150:        PMFCFG0 = (0
  0000 cc5578       [2]     LDD   #21880
  0003 7b0000       [3]     STAB  _PMFCFG0
  151:                              /*| PMFCFG0_INDEPA_MASK*/       // PWM0 and PWM1 are complementart pair
  152:                              /*| PMFCFG0_INDEPB_MASK*/       // PWM2 and PWM3 are complementary pair
  153:                              /*| PMFCFG0_INDEPC_MASK*/       // PWM4 and PWM5 are complemantart pair
  154:                              | PMFCFG0_EDGEA_MASK           // PWM0 and PWM1 are edge-aligned
  155:                              | PMFCFG0_EDGEB_MASK           // PWM2 and PWM3 are edge-aligned
  156:                              | PMFCFG0_EDGEC_MASK           // PWM4 and PWM5 are edge-aligned
  157:                              | PMFCFG0_MTG_MASK                    // Multiple timebase generator
  158:                              /*| PMFCFG0_WP_MASK*/          // Write-protected regs can be written
  159:                              );
  160:
  161:        PMFCFG1   =               (0
  0006 790000       [3]     CLR   _PMFCFG1
  162:                              /*| PMFCFG1_TOPNEGA_MASK*/             // Positive PWM0 polarity
  163:                              /*| PMFCFG1_BOTNEGA_MASK*/             // Positive PWM1 polarity
  164:                              /*| PMFCFG1_TOPNEGB_MASK*/             // Positive PWM2 polarity
  165:                              /*| PMFCFG1_BOTNEGB_MASK*/             // Positive PWM3 polarity
  166:                              /*| PMFCFG1_TOPNEGC_MASK*/             // Positive PWM4 polarity
  167:                              /*| PMFCFG1_BOTNEGC_MASK*/             // Positive PWM5 polarity
  168:                              /*| PMFCFG1_ENHA_MASK*/                // Disable hardware acceleration
  169:                              );
  170:
  171:        PMFCFG2   =               (0
  0009 790000       [3]     CLR   _PMFCFG2
  172:                              /*| PMFCFG2_MSK0_MASK*/                // PWM0 is unmasked
  173:                              /*| PMFCFG2_MSK1_MASK*/                // PWM1 is unmasked
  174:                              /*| PMFCFG2_MSK2_MASK*/                // PWM2 is unmasked
  175:                              /*| PMFCFG2_MSK3_MASK*/                // PWM3 is unmasked
  176:                              /*| PMFCFG2_MSK4_MASK*/                // PWM4 is unmasked
  177:                              /*| PMFCFG2_MSK5_MASK*/                // PWM5 is unmasked
  178:                              );
  179:
  180:        PMFCFG3   =               (0
  000c 790000       [3]     CLR   _PMFCFG3
  181:                              /*| PMFCFG3_SWAPA_MASK*/               // No swap of PWM0 and PWM1
  182:                              /*| PMFCFG3_SWAPB_MASK*/               // No swap pf PWM2 and PWM3
  183:                              /*| PMFCFG3_SWAPC_MASK*/               // No swap of PWM4 and PWM5
  184:                              /*| PMFCFG3_VLMODE0_MASK*/             // Each value reg is accessed independently
  185:                              /*| PMFCFG3_VLMODE1_MASK*/
  186:                              /*| PMFCFG3_PMFFRZ_MASK*/              // PMF continues to run in freeze mode
  187:                              /*| PMFCFG3_PMFWAI_MASK*/              // PMF continues to run in wait mode
  188:                              );
```

```
189:
190:        PMFFCTL   =             (0
000f 790000       [3]     CLR   _PMFFCTL
191:                              /*| PMFFCTL_FIE0_MASK*/              // Fault 0 interrupts disabled
192:                              /*| PMFFCTL_FMODE0_MASK*/            // Manual fault clearing of fault 0
193:                              /*| PMFFCTL_FIE1_MASK*/              // Fault 1 interrupts disabled
194:                              /*| PMFFCTL_FMODE1_MASK*/            // Manual fault clearing of fault 1
195:                              /*| PMFFCTL_FIE2_MASK*/              // Fault 2 interrupts disabled
196:                              /*| PMFFCTL_FMODE2_MASK*/            // Manual fault clearing of fault 2
197:                              /*| PMFFCTL_FIE3_MASK*/              // Fault 3 interrupts disabled
198:                              /*| PMFFCTL_FMODE3_MASK*/            // Manual fault clearing of fault 3
199:                              );
200:
201:        PMFFPIN   =             (0
0012 790000       [3]     CLR   _PMFFPIN
202:                              /*| PMFFPIN_FPINE0_MASK*/            // Fault 0 pin disabled
203:                              /*| PMFFPIN_FPINE1_MASK*/            // Fault 1 pin disabled
204:                              /*| PMFFPIN_FPINE2_MASK*/            // Fault 2 pin disabled
205:                              /*| PMFFPIN_FPINE3_MASK*/            // Fault 3 pin disabled
206:                              );
207:
208:        PMFFSTA   =             (0
0015 7a0000       [3]     STAA  _PMFFSTA
209:                              | PMFFSTA_FFLAG0_MASK               // Clear all interrupt flags
210:                              | PMFFSTA_FFLAG1_MASK
211:                              | PMFFSTA_FFLAG2_MASK
212:                              | PMFFSTA_FFLAG3_MASK
213:                              );
214:
215:        PMFQSMP   =             (0
0018 790000       [3]     CLR   _PMFQSMP
216:                              /*| PMFQSMP_QSMP00_MASK*/                  // 1 sample
217:                              /*| PMFQSMP_QSMP01_MASK*/
218:                              /*| PMFQSMP_QSMP10_MASK*/                  // 1 sample
219:                              /*| PMFQSMP_QSMP11_MASK*/
220:                              /*| PMFQSMP_QSMP20_MASK*/                  // 1 sample
221:                              /*| PMFQSMP_QSMP21_MASK*/
222:                              /*| PMFQSMP_QSMP30_MASK*/                  // 1 sample
223:                              /*| PMFQSMP_QSMP31_MASK*/
224:                              );
225:
226:  // PMF Disable Mapping A Register (PMFDMPA)
227:        PMFDMPA   =             (0
001b 790000       [3]     CLR   _PMFDMPA
228:                              /*| PMFDMPA_DMP00_MASK*/              // Disable PWM0 fault 0
229:                              /*| PMFDMPA_DMP01_MASK*/              // Disable PWM0 fault 1
230:                              /*| PMFDMPA_DMP02_MASK*/              // Disable PWM0 fault 2
231:                              /*| PMFDMPA_DMP03_MASK*/              // Disable PWM0 fault 3
232:                              /*| PMFDMPA_DMP10_MASK*/              // Disable PWM1 fault 0
```

```
                                    /*| PMFDMPA_DMP11_MASK*/               // Disable PWM1 fault 1
                                    /*| PMFDMPA_DMP12_MASK*/               // Disable PWM1 fault 2
                                    /*| PMFDMPA_DMP13_MASK*/               // Disable PWM1 fault 3
                                    );

 // PMF Disable Mapping B Register (PMFDMPB)
        PMFDMPB   =                 (0
001e 790000       [3]     CLR   _PMFDMPB
                                    /*| PMFDMPB_DMP20_MASK*/               // Disable PWM2 fault 0
                                    /*| PMFDMPB_DMP21_MASK*/               // Disable PWM2 fault 1
                                    /*| PMFDMPB_DMP22_MASK*/               // Disable PWM2 fault 2
                                    /*| PMFDMPB_DMP23_MASK*/               // Disable PWM2 fault 3
                                    /*| PMFDMPB_DMP30_MASK*/               // Disable PWM3 fault 0
                                    /*| PMFDMPB_DMP31_MASK*/               // Disable PWM3 fault 1
                                    /*| PMFDMPB_DMP32_MASK*/               // Disable PWM3 fault 2
                                    /*| PMFDMPB_DMP33_MASK*/               // Disable PWM3 fault 3
                                    );

 // PMF Disable Mapping C Register (PMFDMPC)
        PMFDMPC   =                 (0
0021 790000       [3]     CLR   _PMFDMPC
                                    /*| PMFDMPC_DMP40_MASK*/               // Disable PWM4 fault 0
                                    /*| PMFDMPC_DMP41_MASK*/               // Disable PWM4 fault 1
                                    /*| PMFDMPC_DMP42_MASK*/               // Disable PWM4 fault 2
                                    /*| PMFDMPC_DMP43_MASK*/               // Disable PWM4 fault 3
                                    /*| PMFDMPC_DMP50_MASK*/               // Disable PWM5 fault 0
                                    /*| PMFDMPC_DMP51_MASK*/               // Disable PWM5 fault 1
                                    /*| PMFDMPC_DMP52_MASK*/               // Disable PWM5 fault 2
                                    /*| PMFDMPC_DMP53_MASK*/               // Disable PWM5 fault 3
                                    );

 // PMF Output Control Bit Register (PMFOUTC)
        PMFOUTC   =                 (0
0024 790000       [3]     CLR   _PMFOUTC
                                    /*| PMFOUTC_OUTCTL0_MASK*/             // PWM pin software control disabled
                                    /*| PMFOUTC_OUTCTL1_MASK*/             // PWM pin software control disabled
                                    /*| PMFOUTC_OUTCTL2_MASK*/             // PWM pin software control disabled
                                    /*| PMFOUTC_OUTCTL3_MASK*/             // PWM pin software control disabled
                                    /*| PMFOUTC_OUTCTL4_MASK*/             // PWM pin software control disabled
                                    /*| PMFOUTC_OUTCTL5_MASK*/             // PWM pin software control disabled
                                    );
 // PMF Output Control Bit Register (PMFOUTB)
        PMFOUTB   =                 (0
0027 790000       [3]     CLR   _PMFOUTB
                                    /*| PMFOUTB_OUT0_MASK*/                // No direct control of pins
                                    /*| PMFOUTB_OUT1_MASK*/
                                    /*| PMFOUTB_OUT2_MASK*/
                                    /*| PMFOUTB_OUT3_MASK*/
                                    /*| PMFOUTB_OUT4_MASK*/
```

```
278:                                  /*| PMFOUTB_OUT5_MASK*/
279:                                  );
280:
281:  // PMF Deadtime Sample Register (PMFDTMS)
282:        // PMFDTMS is a read only register
283:
284:  // PMF Correction Control Register (PMFCCTL)
285:        PMFCCTL   =               (0
002a 790000       [3]     CLR   _PMFCCTL
286:                                  /*| PMFCCTL_IPOLA_MASK*/               // PMF Value 0 register in next PWM cycle
287:                                  /*| PMFCCTL_IPOLB_MASK*/               // PMF Value 2 register in next PWM cycle
288:                                  /*| PMFCCTL_IPOLC_MASK*/               // PMF Value 4 register in next PWM cycle
289:                                  /*| PMFCCTL_ISENS0_MASK*/              // No correction
290:                                  /*| PMFCCTL_ISENS1_MASK*/
291:                                  );
292:
293:
294:
295:  // PMF Frequency Control A Register (PMFFQCA)
296:  // Prescaler A
297:        PMFFQCA_PRSCA = 1;        // PWM clock frequency is f(bus)/2
002d f60000       [3]     LDAB  _PMFFQCA
0030 c4f9         [1]     ANDB  #249
0032 ca02         [1]     ORAB  #2
0034 7b0000       [3]     STAB  _PMFFQCA
298:  // Load frequency
299:        PMFFQCA_LDFQA = 0;        // Reload every PWM opportunity
0037 1d0000f0     [4]     BCLR  _PMFFQCA,#240
300:
301:  // PMF Counter A Register (PMFCNTA)
302:        // PMFCNTA is a read only register
303:
304:  // PMF Counter Modulo A Register (PMFMODA)
305:        PMFMODA = Modulus;        // See PMF.h
003b cc0190       [2]     LDD   #400
003e 7c0000       [3]     STD   _PMFMODA
306:
307:  // PMF Deadtime A Register (PMFDTMA)
308:        PMFDTMA = 12;                     // Insert this many f(bus) cycles for dead time
0041 ce000c       [2]     LDX   #12
0044 7e0000       [3]     STX   _PMFDTMA
309:
310:  //  PMF Value 0 Register (PMFVAL0)
311:        PMFVAL0   =               Modulus/2;       // Equal positive and negative periods produce 0 volts
0047 49           [1]     LSRD
0048 7c0000       [3]     STD   _PMFVAL0
312:
313:  // PMF Enable Control A Register (PMFENCA)
314:        PMFENCA_LDOKA = 1;        // Load prescaler A, modulus A, and PWM0-1
```

```
004b 1c000002     [4]     BSET  _PMFENCA,#2
315:        PMFENCA_PWMENA = 1;        // Enable PWM generator A and PWM0-1
004f 1c000080     [4]     BSET  _PMFENCA,#128
316:
317:        Phase = TRUE;                       // 1 = top half, 0 = bottom half
0053 c601         [1]     LDAB  #1
0055 7c0000       [3]     STD   Phase
318:        Slope = TRUE;                       // 1 = going up, 0 = going down
0058 7c0000       [3]     STD   Slope
319:        Angle = 0;                                // Start at zero address is sine table
005b c7           [1]     CLRB
005c 7c0000       [3]     STD   Angle
320:        //Amp = 250;                        // Maximum amplitude for test
321:        Amp = 0;                            // Starting amplitude is zero
005f 7b0000       [3]     STAB  Amp
322:
323:        PMFENCA_PWMRIEA = 1;     // Enable PWM A interrupts
0062 1c000001     [4]     BSET  _PMFENCA,#1
324:
325:
326:
327:  // PMF Frequency Control B Register (PMFFQCB)
328:        PMFFQCB   =               0;
0066 7b0000       [3]     STAB  _PMFFQCB
329:  // PMF Counter B Register (PMFCNTB)
330:        // PMFCNTB is a read only register
331:  // PMF Counter Modulo B Register (PMFMODB)
332:        PMFMODB   =               0;
0069 7c0000       [3]     STD   _PMFMODB
333:  // PMF Deadtime B Register (PMFDTMB)
334:        PMFDTMB   =               0;
006c 7c0000       [3]     STD   _PMFDTMB
335:  // PMF Enable Control B Register (PMFENCB)
336:        PMFENCB   =               0;
006f 7b0000       [3]     STAB  _PMFENCB
337:
338:
339:
340:  // PMF Frequency Control C Register (PMFFQCC)
341:        PMFFQCC   =               0;
0072 7b0000       [3]     STAB  _PMFFQCC
342:  // PMF Counter C Register (PMFCNTC)
343:        // PMFCNTC is a read only register
344:  // PMF Counter Modulo C Register (PMFMODC)
345:        PMFMODC   =               0;
0075 7c0000       [3]     STD   _PMFMODC
346:  // PMF Deadtime C Register (PMFDTMC)
347:        PMFDTMC   =               0;
0078 7c0000       [3]     STD   _PMFDTMC
```

```
  348:  // PMF Enable Control C Register (PMFENCC)
  349:        PMFENCC   =              0;
  007b 7b0000       [3]     STAB  _PMFENCC
  350:
  351:  // After this instruction, write-protected registers can't be modified
  352:        PMFCFG0_WP = 1;
  007e 1c000080     [4]     BSET  _PMFCFG0,#128
  353:
  354:  /*
  355:        EnableHBridge();         // Turn on H-bridge
  356:  */
  357:
  358:  }
  0082 3d           [5]     RTS
  359:
  360:
  361:  /*************************************************************************/
  362:  void IncAmplitude(void)
  363:  {
*** EVALUATION ***

Function: IncAmplitude
Source  : C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources\PMF.c
Options : -CPUHCS12 -Env"GENPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12;C:\Users\Public\Projects\9S12Float\CodeHC9S12\
bin;C:\Users\Public\Projects\9S12Float\CodeHC9S12\prm;C:\Users\Public\Projects\9S12Float\CodeHC9S12\cmd;C:\Users\Public\
Projects\9S12Float\CodeHC9S12\Sources;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\lib;C:\Program Files\
Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\src;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include"
-Env"LIBPATH=C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -EnvOBJPATH=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\bin -EnvTEXTPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -Lasm=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o.lst -Ms -ObjN=C:\Users\Public\Projects\9S12Float\CodeHC9S12\
CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o -WmsgSd1106

  364:        if (Amp < 254)
  0000 f60000       [3]     LDAB  Amp
  0003 c1fe         [1]     CMPB  #254
  0005 2403         [3/1]   BCC   *+5 ;abs = 000a
  365:                Amp++;
  0007 720000       [4]     INC   Amp
  366:  }
  000a 3d           [5]     RTS
  367:
  368:
  369:  /*************************************************************************/
  370:  void DecAmplitude(void)
  371:  {
*** EVALUATION ***

Function: DecAmplitude
Source  : C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources\PMF.c
```

```
Options : -CPUHCS12 -Env"GENPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12;C:\Users\Public\Projects\9S12Float\CodeHC9S12\
bin;C:\Users\Public\Projects\9S12Float\CodeHC9S12\prm;C:\Users\Public\Projects\9S12Float\CodeHC9S12\cmd;C:\Users\Public\
Projects\9S12Float\CodeHC9S12\Sources;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\lib;C:\Program Files\
Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\src;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include"
-Env"LIBPATH=C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -EnvOBJPATH=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\bin -EnvTEXTPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -Lasm=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o.lst -Ms -ObjN=C:\Users\Public\Projects\9S12Float\CodeHC9S12\
CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o -WmsgSd1106

  372:        if (Amp > 0)
  0000 f60000       [3]     LDAB  Amp
  0003 2703         [3/1]   BEQ   *+5 ;abs = 0008
  373:                  Amp--;
  0005 730000       [4]     DEC   Amp
  374:  }
  0008 3d           [5]     RTS
  375:
  376:
  377:  /*************************************************************************/
  378:  void DisplayAmplitude(void)
  379:  {
*** EVALUATION ***

Function: DisplayAmplitude
Source  : C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources\PMF.c
Options : -CPUHCS12 -Env"GENPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12;C:\Users\Public\Projects\9S12Float\CodeHC9S12\
bin;C:\Users\Public\Projects\9S12Float\CodeHC9S12\prm;C:\Users\Public\Projects\9S12Float\CodeHC9S12\cmd;C:\Users\Public\
Projects\9S12Float\CodeHC9S12\Sources;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\lib;C:\Program Files\
Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\src;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include"
-Env"LIBPATH=C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -EnvOBJPATH=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\bin -EnvTEXTPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -Lasm=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o.lst -Ms -ObjN=C:\Users\Public\Projects\9S12Float\CodeHC9S12\
CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o -WmsgSd1106

  0000 3b           [2]     PSHD
  380:        char* BufNum;
  381:
  382:        BufNum = GetBuffer();     // Get a free buffer
  0001 160000       [4]     JSR   GetBuffer
  383:        (void)sprintf(BufNum,"Amplitude: %d\r\n",Amp);
  0004 b745         [1]     TFR   D,X
  0006 f60000       [3]     LDAB  Amp
  0009 87           [1]     CLRA
  000a 3b           [2]     PSHD
  000b cc0000       [2]     LDD   #"Amplitude: %d\015\012"
  000e 3b           [2]     PSHD
  000f 34           [2]     PSHX
  0010 6e86         [2]     STX   6,SP
  0012 160000       [4]     JSR   sprintf
```

```
  384:        SCI0_PutQueue(BufNum);
  0015 eca5          [3]     LDD   6,+SP
  0017 160000        [4]     JSR   SCI0_PutQueue
  385:  }
  001a 3a            [3]     PULD
  001b 3d            [5]     RTS
  386:
  387:
  388:  /*****************************************************************************/
  389:  void EnableHBridge(void)
  390:  {
*** EVALUATION ***

Function: EnableHBridge
Source  : C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources\PMF.c
Options : -CPUHCS12 -Env"GENPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12;C:\Users\Public\Projects\9S12Float\CodeHC9S12\
bin;C:\Users\Public\Projects\9S12Float\CodeHC9S12\prm;C:\Users\Public\Projects\9S12Float\CodeHC9S12\cmd;C:\Users\Public\
Projects\9S12Float\CodeHC9S12\Sources;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\lib;C:\Program Files\
Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\src;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include"
-Env"LIBPATH=C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -EnvOBJPATH=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\bin -EnvTEXTPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -Lasm=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o.lst -Ms -ObjN=C:\Users\Public\Projects\9S12Float\CodeHC9S12\
CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o -WmsgSd1106

  391:        PTP_PTP2 = 0;                     // Enable H-bridge drivers
  0000 1d000004      [4]     BCLR  _PTP,#4
  392:  }
  0004 3d            [5]     RTS
  393:
  394:
  395:  /*****************************************************************************/
  396:  void DisableHBridge(void)
  397:  {
*** EVALUATION ***

Function: DisableHBridge
Source  : C:\Users\Public\Projects\9S12Float\CodeHC9S12\Sources\PMF.c
Options : -CPUHCS12 -Env"GENPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12;C:\Users\Public\Projects\9S12Float\CodeHC9S12\
bin;C:\Users\Public\Projects\9S12Float\CodeHC9S12\prm;C:\Users\Public\Projects\9S12Float\CodeHC9S12\cmd;C:\Users\Public\
Projects\9S12Float\CodeHC9S12\Sources;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\lib;C:\Program Files\
Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\src;C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include"
-Env"LIBPATH=C:\Program Files\Freescale\CodeWarrior for S12(X) V5.0\lib\HC12c\include" -EnvOBJPATH=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\bin -EnvTEXTPATH=C:\Users\Public\Projects\9S12Float\CodeHC9S12\bin -Lasm=C:\Users\Public\Projects\9
S12Float\CodeHC9S12\CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o.lst -Ms -ObjN=C:\Users\Public\Projects\9S12Float\CodeHC9S12\
CodeHC9S12_Data\Standard\ObjectCode\PMF.c.o -WmsgSd1106

  398:        PTP_PTP2 = 1;                     // Disable H-bridge drivers
  0000 1c000004      [4]     BSET  _PTP,#4
  399:  }
```

```
  0004 3d           [5]     RTS
  400:
```