~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MOTOROLA MICROPROCESSOR & MEMORY TECHNOLOGY GROUP
M68000 Hi-Performance Microprocessor Division
M68060 Software Package
Production Release P1.00 -- October 10, 1994


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~



68060 FLOATING-POINT SOFTWARE PACKAGE (Library version)
------------------------------------------------------

The file fplsp.sa contains the "Library version" of the
68060SP Floating-Point Software Package. The routines
included in this module can be used to emulate the
FP instructions not implemented in 68060 hardware. These
instructions normally take exception vector #11
"FP Unimplemented Instruction".

By re-compiling a program that uses these instructions, and
making subroutine calls in place of the unimplemented
instructions, a program can avoid the overhead associated
with taking the exception.

Release file format:
--------------------
The file fplsp.sa is essentially a hexadecimal image of the
release package. This is the ONLY format which will be supported.
The hex image was created by assembling the source code and
then converting the resulting binary output image into an
ASCII text file. The hexadecimal numbers are listed
using the Motorola Assembly Syntax assembler directive "dc.l"
(define constant longword). The file can be converted to other
assembly syntaxes by using any word processor with a global
search and replace function.

To assist in assembling and linking this module with other modules,

the installer should add a symbolic label to the top of the file.
This will allow calling routines to access the entry points
of this package.

The source code fplsp.s has also been included but only for
documentation purposes.

Release file structure:
-----------------------
The file fplsp.sa contains an "Entry-Point" section and a
code section. The FPLSP has no "Call-Out" section. The first section
is the "Entry-Point" section. In order to access a function in the
package, a program must "bsr" or "jsr" to the location listed
below in "68060FPLSP entry points" that corresponds to the desired
function. A branch instruction located at the selected entry point
within the package will then enter the correct emulation code routine.

The entry point addresses at the beginning of the package will remain
fixed so that a program calling the routines will not have to be
re-compiled with every new 68060FPLSP release.

There are 3 entry-points for each instruction type: single precision,
double precision, and extended precision.

As an example, the "fsin" library instruction can be passed an
extended precision operand if program executes:



```
# fsin.x fp0

      fmovm.x     &0x01,-(%sp)      # pass operand on stack
      bsr.l _060FPLSP_TOP+0x1a8 # branch to fsin routine
      add.l &0xc,%sp  # clear operand from stack
```



Upon return, fp0 holds the correct result. The FPSR is
set correctly. The FPCR is unchanged. The FPIAR is undefined.



Another example. This time, a dyadic operation:

```
# frem.s %fp1,%fp0

      fmov.s      %fp1,-(%sp) # pass src operand
      fmov.s      %fp0,-(%sp) # pass dst operand
      bsr.l _060FPLSP_TOP+0x168 # branch to frem routine
      addq.l      &0x8,%sp    # clear operands from stack
```

Again, the result is returned in fp0. Note that BOTH operands
are passed in single precision format.

Exception reporting:
--------------------
The package takes exceptions according to the FPCR value upon subroutine
entry. If an exception should be reported, then the package forces
this exception using implemented floating-point instructions.
For example, if the instruction being emulated should cause a
floating-point Operand Error exception, then the library routine
executes an FMUL of a zero and an infinity to force the OPERR
exception. Although the FPIAR will be undefined for the enabled
Operand Error exception handler, the user will at least be able
to record that the event occurred.

Miscellaneous:
--------------
The package does not attempt to correctly emulate instructions
with Signalling NAN inputs. Use of SNANs should be avoided with

this package.

The fabs/fadd/fdiv/fint/fintrz/fmul/fneg/fsqrt/fsub entry points
are provided for the convenience of older compilers that make
subroutine calls for all fp instructions. The code does NOT emulate
the instruction but rather simply executes it.

68060FPLSP entry points:
------------------------
_060FPLSP_TOP:
0x000:  _060LSP__facoss_
0x008:  _060LSP__facosd_
0x010:  _060LSP__facosx_
0x018:  _060LSP__fasins_
0x020:  _060LSP__fasind_
0x028:  _060LSP__fasinx_
0x030:  _060LSP__fatans_
0x038:  _060LSP__fatand_
0x040:  _060LSP__fatanx_
0x048:  _060LSP__fatanhs_
0x050:  _060LSP__fatanhd_
0x058:  _060LSP__fatanhx_
0x060:  _060LSP__fcoss_
0x068:  _060LSP__fcosd_
0x070:  _060LSP__fcosx_
0x078:  _060LSP__fcoshs_
0x080:  _060LSP__fcoshd_
0x088:  _060LSP__fcoshx_
0x090:  _060LSP__fetoxs_
0x098:  _060LSP__fetoxd_
0x0a0:  _060LSP__fetoxx_
0x0a8:  _060LSP__fetoxm1s_
0x0b0:  _060LSP__fetoxm1d_
0x0b8:  _060LSP__fetoxm1x_
0x0c0:  _060LSP__fgetexps_
0x0c8:  _060LSP__fgetexpd_
0x0d0:  _060LSP__fgetexpx_
0x0d8:  _060LSP__fgetmans_
0x0e0:  _060LSP__fgetmand_
0x0e8:  _060LSP__fgetmanx_
0x0f0:  _060LSP__flog10s_
0x0f8:  _060LSP__flog10d_
0x100:  _060LSP__flog10x_
0x108:  _060LSP__flog2s_
0x110:  _060LSP__flog2d_
0x118:  _060LSP__flog2x_
0x120:  _060LSP__flogns_
0x128:  _060LSP__flognd_
0x130:  _060LSP__flognx_
0x138:  _060LSP__flognp1s_
0x140:  _060LSP__flognp1d_
0x148:  _060LSP__flognp1x_
0x150:  _060LSP__fmods_
0x158:  _060LSP__fmodd_
0x160:  _060LSP__fmodx_
0x168:  _060LSP__frems_
0x170:  _060LSP__fremd_
0x178:  _060LSP__fremx_
0x180:  _060LSP__fscales_
0x188:  _060LSP__fscaled_
0x190:  _060LSP__fscalex_
0x198:  _060LSP__fsins_
0x1a0:  _060LSP__fsind_
0x1a8:  _060LSP__fsinx_

```
0x1b0:      _060LSP__fsincoss_
0x1b8:      _060LSP__fsincosd_
0x1c0:      _060LSP__fsincosx_
0x1c8:      _060LSP__fsinhs_
0x1d0:      _060LSP__fsinhd_
0x1d8:      _060LSP__fsinhx_
0x1e0:      _060LSP__ftans_
0x1e8:      _060LSP__ftand_
0x1f0:      _060LSP__ftanx_
0x1f8:      _060LSP__ftanhs_
0x200:      _060LSP__ftanhd_
0x208:      _060LSP__ftanhx_
0x210:      _060LSP__ftentoxs_
0x218:      _060LSP__ftentoxd_
0x220:      _060LSP__ftentoxx_
0x228:      _060LSP__ftwotoxs_
0x230:      _060LSP__ftwotoxd_
0x238:      _060LSP__ftwotoxx_

0x240:      _060LSP__fabss_
0x248:      _060LSP__fabsd_
0x250:      _060LSP__fabsx_
0x258:      _060LSP__fadds_
0x260:      _060LSP__faddd_
0x268:      _060LSP__faddx_
0x270:      _060LSP__fdivs_
0x278:      _060LSP__fdivd_
0x280:      _060LSP__fdivx_
0x288:      _060LSP__fints_
0x290:      _060LSP__fintd_
0x298:      _060LSP__fintx_
0x2a0:      _060LSP__fintrzs_
0x2a8:      _060LSP__fintrzd_
0x2b0:      _060LSP__fintrzx_
0x2b8:      _060LSP__fmuls_
0x2c0:      _060LSP__fmuld_
0x2c8:      _060LSP__fmulx_
0x2d0:      _060LSP__fnegs_
0x2d8:      _060LSP__fnegd_
0x2e0:      _060LSP__fnegx_
0x2e8:      _060LSP__fsqrts_
0x2f0:      _060LSP__fsqrtd_
0x2f8:      _060LSP__fsqrtx_
0x300:      _060LSP__fsubs_
0x308:      _060LSP__fsubd_
0x310:      _060LSP__fsubx_
```