Interfacing the NS32081 as a Floating-Point Peripheral

National Semiconductor
Application Note 383
Microprocessor Applications
Engineering
July 1986

This note is a guide for users who wish to interface the NS32081 Floating-Point Unit (FPU) as a peripheral unit to CPUs other than those of the Series 32000 family. This is not a particularly expensive procedure, but it requires some in-depth information not all of which is available in the NS32081 data sheet. Four basic topics will be covered here:

An overview of the architecture of the NS32081 as seen in a stand-alone environment.

The protocol used to sequence it through the execution of an instruction.

Special guidelines for connecting and programming the NS32081 as a peripheral component.

A sample application of these guidelines in the form of a circuit interfacing the NS32081 to the Motorola 68000 microprocessor.

References are made here to the NS32081 data sheet and the Series 32000 Instruction Set Reference Manual (Publication #420010099-001). The reader should have both these documents on hand.

1.0 Architecture Overview

1.1 REGISTER SET

The register set internal to the NS32081 FPU is shown in *Figure 1*. It consists of nine registers, each 32 bits in length:

FSR The Floating-Point Status Register. As given in the data sheet, this register holds status and mode information for the FPU. It is loaded by executing the LFSR instruction and examined using the SFSR instruction.

F0–F7 The Floating-Point Registers. Each can hold a single 32-bit single-precision floating-point value. To hold double-precision values, a register pair is referenced using the even-numbered register of the pair.

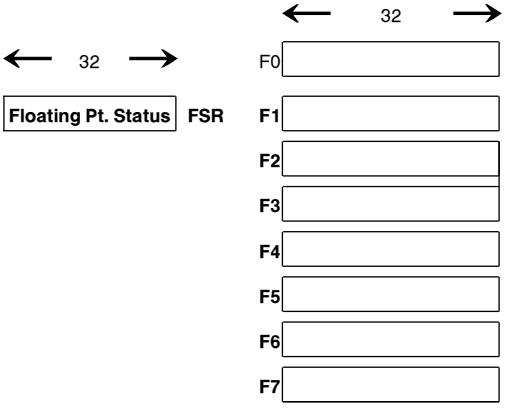


FIGURE 1. FPU Registers

Floating-point operands need not be held in registers; they may be supplied externally as part of the instruction sequence. Integer operands (appearing in conversion instructions) and values being transferred to or from the FSR must be supplied externally; they cannot be held in Floating-Point registers F0–F7.

1.2 INSTRUCTION SET AND ENCODING

The encodings used for NS32081 instructions are shown in *Figure 2*. They fall within two formats, labeled from Series 32000 tradition ''Format 9'' and ''Format 11''. These formats are distinguished by their least-significant byte (the ''ID Byte''). Execution of an FPU instruction starts by passing first the ID Byte and then the rest of the instruction (the ''Operation Word'') to the FPU.

Fields within an instruction are interpreted by the FPU in the same manner as documented in Chapter 4 of the Series 32000 Instruction Set Reference Manual, with the exception of the 5-bit General Addressing Mode fields (*gen1, gen2*). Since the FPU does not itself perform memory accesses, it does not need to use these fields for addressing calculations. The only use it makes of these fields is to determine for each operand whether the value is to be found internal to the FPU (that is, within a register F0–F7, or whether it is to be transferred to and/or from the FPU. See *Figure 3*. A value of 0–7 in a *gen* field specifies one of the Floating-Point registers F0–F7, respectively, as the location of the corresponding operand. Any greater value specifies that the operand's location is external to the FPU and that its value will be transferred as part of the protocol. Any non-floating operand is always handled by the FPU as external, regardless of the addressing mode specified in its *gen* field. It is illegal to reference an odd-numbered register for a double-precision operand. If an odd register is referenced, the results are unpredictable.

1.3 PINOUT

The FPU is packaged in a 24-pin DIP (see *Figure 4*). The pin functions can be split into two groups: those that participate in the communication protocol between the FPU and the host system, and those that reflect the familiar requirements of LSI components.

The protocol uses the following pins of the FPU:

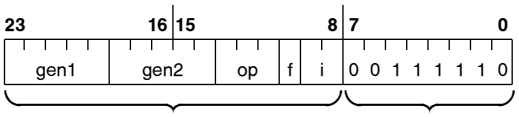
D0–D15 The 16-bit data bus. The D0 pin holds the least-significant bit of data transferred on the bus.

$\overline{\text{SPC}}$ A dual-purpose pin, low active. $\overline{\text{SPC}}$ is pulsed low from the host system as the data strobe for bus transfers. $\overline{\text{SPC}}$ is pulsed low by the FPU to signal that it has completed the internal execution phase of an instruction.

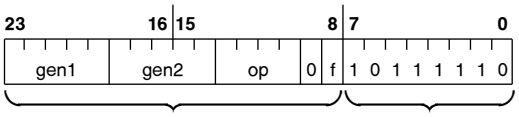
RRD-B30M105/Printed in U. S. A.

1.0 Architecture Overview (Continued)

ST0, ST1 The status code. This 2-bit value is sampled by the FPU on the falling edge of $\overline{SPC}$, and informs it of the current protocol phase. ST0 is the least-significant bit of the value. The need filled by the status code is most relevant to Series 32000-based systems, where it serves to allow retry of aborted instructions and to disambiguate the protocol when the $\overline{SPC}$ signal is bussed among multiple slave processors. In microprocessor-based peripheral applications, the status code can generally be provided from the CPU's address lines.



Format 9: LFSR/SFSR/Conversions



Format 11: Movement/Calculation

FIGURE 2. FPU Instruction Formats

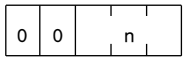
The pins providing for standard requirements are:

CLK The clock input. This is a TTL-level square wave which the FPU uses to sequence its internal calculations.

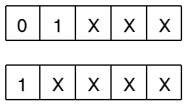
$\overline{RST}$ The reset input. This signal is used to reset the FPU's internal logic.

VCC The 5-volt positive supply.

GNDB, GNDL The grounding pins. GNDB serves as ground for the FPU's output buffers, and GNDL is used for the rest of the on-chip logic.



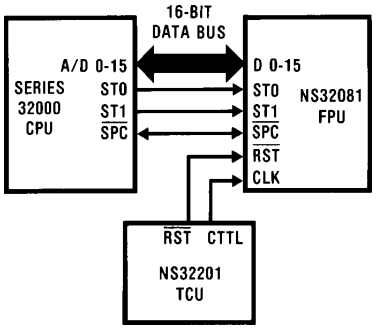
FPU Internal Register: Fn, n = 0 . . . 7
Long Floating = Even Register Only



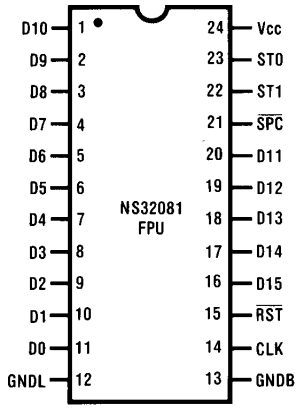
External to FPU

Note: All non-floating operands are always external.

FIGURE 3. FPU Addressing Modes



TL/EE/8388–1



Top View

TL/EE/8388–2

FIGURE 4. NS32081 FPU Connections

2.0 Protocol

The FPU requires a fixed sequence of transfers (''protocol'') in its communication with the outside world. Each step of the protocol is identified by a status code (asserted to the FPU on pins ST0 and ST1) and by its position in the sequence, as shown in *Figure 5*.

Status Combinations:
- 11: Write ID Byte
- 01: Transfer Operation/Operand
- 10: Read Status Word

Step	Status	Action
1	11	CPU sends ID Byte on least-significant byte of bus.
2	01	CPU sends Operation Word, bytes swapped on bus.
3	01	CPU sends required operands, *gen1* first, least-significant word first.
4	xx	FPU starts internal execution.
5	xx	FPU pulses $\overline{SPC}$ low.
6	10	CPU reads Status Word (Error/Comparison Result).
7	01	CPU reads result (if any), least-significant word first.

FIGURE 5. FPU Instruction Protocol

Steps 1 and 2 transfer the instruction to the FPU. Step 1 transfers the first byte of the instruction (the ID Byte) and Step 2 transfers the rest of the instruction (the Operation Word). In Step 2, the two bytes of the Operation Word must be swapped on the bus; i.e. the most-significant byte of the Operation Word must be presented on the least-significant byte of the bus.

Step 3 is optional and repeatable depending on the instruction. It is used to transfer to the FPU any external operands that are required by the instruction. The operand specified by *gen1* is sent first, least-significant word first, followed by the operand specified by *gen2*. If an operand is only one byte in length, it is transferred on the least-significant half of the bus.

The FPU initiates Step 4 of the protocol, internal computation, upon receiving the last external operand word or, if there are no external operands, upon receiving the Operation Word of the instruction. During this time, the data bus may be used for any purpose by the rest of the system, as long as the $\overline{SPC}$ pin is kept pulled up by a resistor and is not actively driven.

Step 5 occurs when the FPU completes the instruction. The FPU pulses the $\overline{SPC}$ pin low to acknowledge that it is ready to continue the protocol. This pulse is called the ''Done pulse''. The bus is not used during this step, and remains floating.

In Step 6, the FPU is polled by reading a Status Word. This word indicates whether an exception has been detected by the FPU. In the Compare instruction (CMPf), it also displays the relationship between the operands and serves as the result. This transfer is mandatory, regardless of whether the information presented by the FPU is intended to be used. See *Figure 3-6* of the data sheet.

Step 7 is, like Step 3, optional and repeatable depending on the instruction. Any external result of an instruction is read from the FPU in this step, least-significant word first. If the result is a 1-byte value, it is presented by the FPU on the least-significant half of the bus (D0–D7).

Note: If in Step 6 the FPU indicates that an error has occurred, it is permissible, though not necessary, to continue the protocol through Step 7. No guarantee is made regarding the validity of the value read, but continuing through Step 7 will not cause any protocol problems.

If at any time within the protocol another ID byte is sent (ST = 11), the FPU will prepare itself internally to execute another instruction, throwing away the instruction that was in progress. This is done to support the Abort with Retry feature of the Series 32000 family.

Because of this feature, however, there is an important consideration when using the FPU in systems that support multitasking: the operating system must not allow a task using the FPU to be interrupted in the middle of an instruction protocol and then transfer control to another task that is also using the FPU. The partially-executed instruction would be thrown away, leaving the first task with a garbage result when it continues. This situation can be avoided easily in software but, depending on the system, some cooperation may be required from the user program. Other solutions involving some additional hardware are also possible.

3.0 Interfacing Guidelines

There are some special interfacing considerations that are required (see *Figure 6*):

1. The edges of the $\overline{SPC}$ pulse must have a fixed relationship to the clock signal (CLK) presented to the FPU. When writing information to the FPU, the pulse must start shortly after a rising edge of CLK and end shortly after the next rising edge of CLK. Failing to do so can cause the FPU to fail, often by causing it to freeze and not generate the Done pulse. This synchronous generation of $\overline{SPC}$ is also important when reading information from the FPU, but the $\overline{SPC}$ pulse is allowed to be two clocks in width. These requirements will be expressed in future NS32081 data sheets as a minimum setup time requirement between each edge of the $\overline{SPC}$ pulse and the next rising edge of CLK, currently set at 40 nanoseconds on the basis of preliminary characterization. The propagation delay in generating $\overline{SPC}$ through a Schottky flip-flop (e.g. 74S74) and a low-power Schottky buffer (e.g. 74LS125A) is therefore acceptable at 10 MHz. LS technology is recommended for the buffer to minimize undershoot when driving $\overline{SPC}$.
2. After the FPU generates the Done pulse, it is necessary to leave the $\overline{SPC}$ pin high for an additional two cycles of CLK before performing the Read Status Word transfer.
3. After performing the Read Status Word transfer, it is necessary to wait for an additional three cycles of CLK before reading a result from the FPU.

4.0 An Interface to the MC68000 Microprocessor

4.1 HARDWARE

A block diagram of the circuitry required to interface the MC68000 MPU to the NS32081 is shown in *Figure 7*.

First the easy part. Direct connections are possible on the data bus, which is numbered compatibly (D0–D15 on both parts), the status pins ST0–ST1 (connected to address lines A4–A5 from the 68000), and the clock (CLK on both). The system reset signal ($\overline{\text{RESET}}$ to and/or from the MC68000) should be synchronized with the clock before presenting it as $\overline{\text{RST}}$ to the FPU.

All that remains to be done is to generate $\overline{\text{SPC}}$ pulses that are within specifications whenever the 68000 accesses the FPU, and to detect the Done pulse from the FPU in a manner that will allow the 68000 to poll for it.

The approach selected for generating $\overline{\text{SPC}}$ pulses uses an address decoder that recognizes two separate address spaces; one to transfer information to or from the FPU ($\overline{\text{XFER}}$), and one to poll for the Done pulse ($\overline{\text{POLL}}$).

The 68000 signals $\overline{\text{AS}}$ (Address Strobe) and R/$\overline{\text{W}}$ (Read / not Write) are used to generate $\overline{\text{SPC}}$ timing.

Figure 8 shows the timing generated when the 68000 is writing to the FPU. The $\overline{\text{SPC}}$ pin is kept floating (held high by a pullup resistor) until bus state S4, at which point it is pulled low. On the next rising edge of CLK, $\overline{\text{SPC}}$ is actively pulled high, and is set floating afterward. It is not simply allowed to float high, as the resulting rise time can be unacceptable at speeds above about 4 MHz. A timing chain, required due to the 10-MHz 68000's treatment of its $\overline{\text{AS}}$ strobe, generates the signals TA, TB and TC, from which the $\overline{\text{SPC}}$ signal's state and enable are controlled.

Figure 9 shows the $\overline{\text{SPC}}$ timing for reading from the FPU. The basic difference is that $\overline{\text{SPC}}$ remains active for two clocks, so that the FPU holds data on the bus until it is sampled by the 68000. Again, $\overline{\text{SPC}}$ is actively driven high before being released.

Note: Although $\overline{\text{SPC}}$ must be driven high before being released, it must not be actively driven for more than two clocks after the trailing edge of $\overline{\text{SPC}}$. This is because the FPU can respond as quickly as three clocks after that edge with a Done pulse.

A simpler scheme in which the $\overline{\text{SPC}}$ pulse is identical for both reading and writing (1-clock wide always, but starting ½ clock later with CLK into the FPU inverted) was considered, but was rejected because the data hold time presented by the 68000 on a Write cycle would be inadequate at 10 MHz.

Any $\overline{\text{SPC}}$ pulse appearing while the $\overline{\text{XFER}}$ Select signal is inactive is interpreted as a Done pulse, which is latched in a flip-flop within the Done Detector block. When the 68000 performs a Read cycle from the address that generates the $\overline{\text{POLL}}$ select signal, the contents of the flip-flop are placed on data bus bit D15. Since this is the sign bit of a 16-bit value, the 68000 can perform a fast test of the bit using a MOVE.W instruction and a conditional branch (BPL) to wait for the FPU.

The schematic for the $\overline{\text{SPC}}$ generator and the Done pulse detector is given in *Figures 10a* and *10b*. The flip-flop labeled SPC generates the edges of the $\overline{\text{SPC}}$ pulse (on the signal $\overline{\text{SPCT}}$). The timing chain (TA, TB) provides the enable control to the buffer driving $\overline{\text{SPC}}$ to the FPU, as well as the signal to terminate the $\overline{\text{SPC}}$ pulse (either TB or TC, depending on the direction of the data transfer). Note that the timing chain assumes a full-speed memory cycle of four clocks in accessing the FPU, and will fail otherwise. The circuit generating the Data Acknowledge signal to the 68000 (DTACK, not shown) must guarantee this. In any system that must use a longer access, some modification to the timing chain will be necessary.

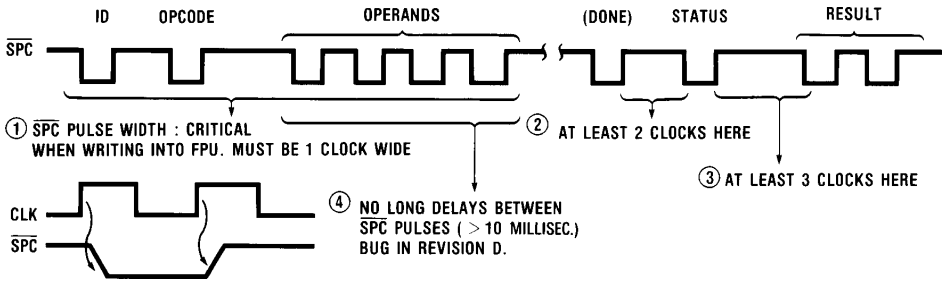
The flip-flop labeled DONE (*Figure 10b*) is the Done pulse detector. It is cleared by performing a data transfer into the FPU and is set by a Done pulse on $\overline{\text{SPC}}$. A buffer, enabled by the $\overline{\text{POLL}}$ select signal, connects its output to data bus bit 15.

4.2 SOFTWARE

Some notes on programming the FPU in a 68000 environment:

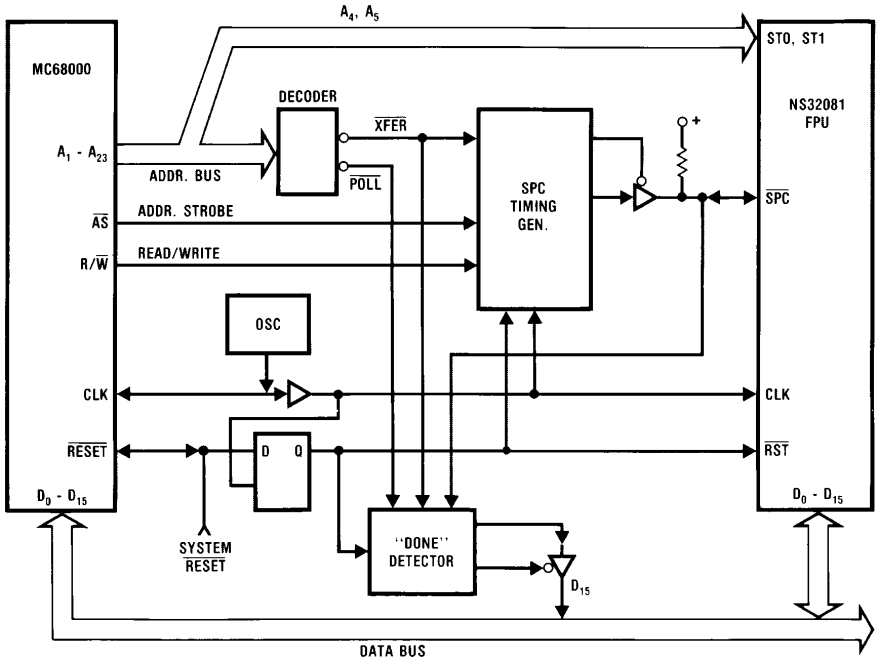
1. The byte addressing convention in the 68000 differs from that of the Series 32000 family. In particular, a byte with an even address is transferred on the most-significant half of the bus by the 68000, but the FPU expects to see it on the least-significant byte. When transferring a single byte to or from the FPU, either do so with an odd address specified, or transfer the byte as the least-significant half of a 16-bit value at an even address.
2. The 68000 transfers 32-bit operands by sending the most-significant 16 bits first. The FPU expects values to be transferred in the opposite order. Make certain that operands are transferred in the correct order (the 68000 SWAP instruction can be helpful for this).

A sample program that sequences the FPU through the execution of an ADDF instruction is listed in *Figure 11*. As this example is intended for clarity rather than efficiency, improvements are possible. The $\overline{\text{XFER}}$ select is assumed to be generated by addresses of the form 06xxxx (hex) and the $\overline{\text{POLL}}$ select is assumed to be generated by addresses of the form 07xxxx.



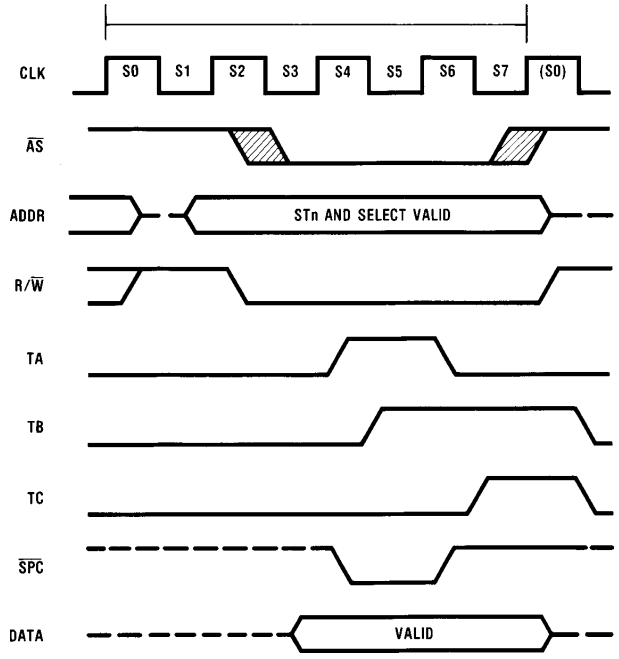
TL/EE/8388–3

FIGURE 6. Interfacing to FPU: Cautions



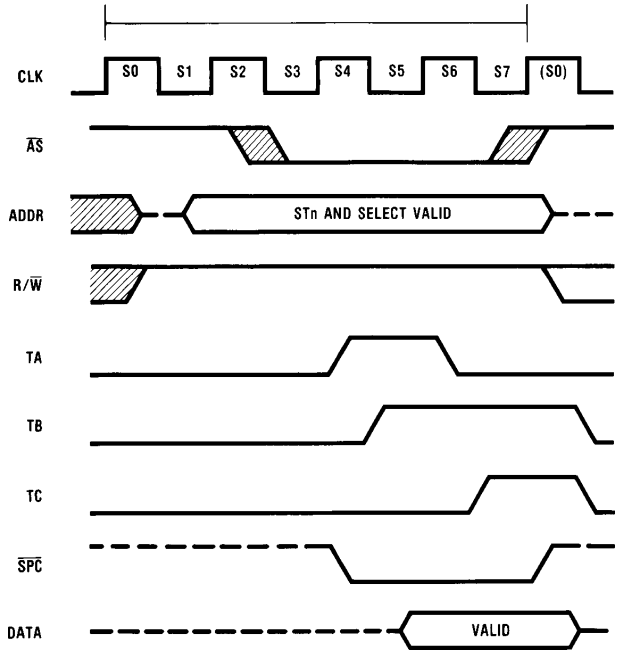
TL/EE/8388–4

FIGURE 7. 68000-32081 Interface Block Diagram



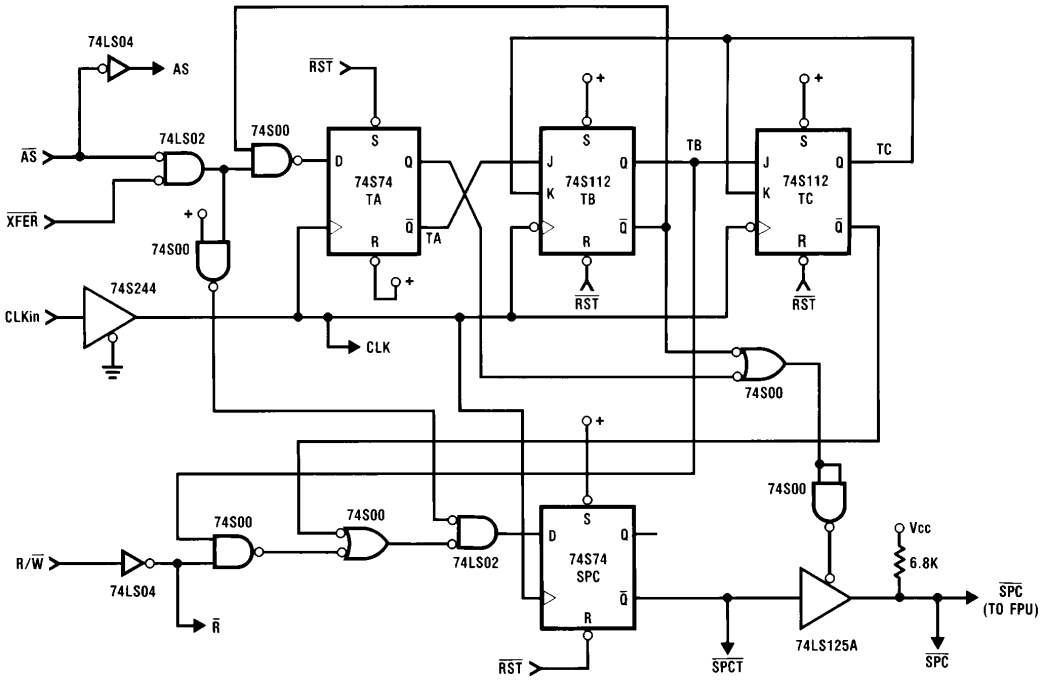
TL/EE/8388–5

FIGURE 8. 68000 Write to FPU



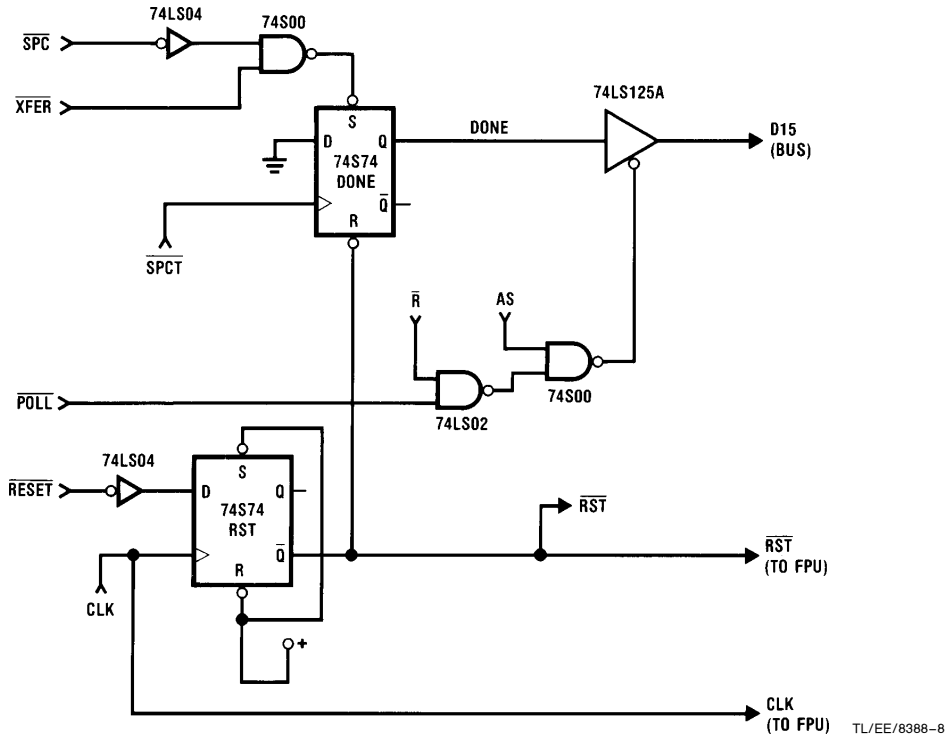
TL/EE/8388–6

FIGURE 9. 68000 Read from FPU



TL/EE/8388–7

FIGURE 10a. Schematic: $\overline{\text{SPC}}$ Timing Generator



TL/EE/8388–8

FIGURE 10b. Schematic: DONE Detector and $\overline{\text{RESET}}$ Synchronizer

```
*     Register Contents:
*
*     A0 = 00070000     Address of DONE flip-flop.
*     A1 = 00060010     Address for ST=1 transfer (Transfer Operand).
*     A2 = 00060020     Address for ST=2 transfer (Read Status Word).
*     A3 = 00060030     Address for ST=3 transfer (Broadcast ID).
*
*     D0 = 000000BE     ID byte for ADDF instruction.
*     D1 = 00000184     Operation Word for ADDF. (Note bytes swapped.)
*     D2 = 3F800000     First operand = 1.0.
*     D3 = 3F800000     Second operand = 1.0.
*     D4                Receives Status Word from FPU.
*     D5                Receives result from FPU.
*     D7                Scratch register (for DONE bit test).
*
START MOVE.W  D0,(A3)         Send ID byte.
      MOVE.W  D1,(A1)         Send Operation Word.
      SWAP    D2              Send operands. The swapping
      MOVE.L  D2,(A1)          is included because the
      SWAP    D2               FPU expects the least-
      SWAP    D3               significant word first.
      MOVE.L  D3,(A1)          (Can be avoided, with care.)
      SWAP    D3
*
POLL  MOVE.W  (A0),D7         Check the DONE flip-flop,
      BPL     POLL             loop until FPU is finished.
*                              (DONE bit is sign bit, tested
*                              by the MOVE instruction.)
*
      MOVE.W  (A2),D4         Read Status Word.
      MOVE.L  (A1),D5         Read result.
      SWAP    D5              Swap halves of result.
```

FIGURE 11. Single-Precision Addition (Demo Routine)

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408