MICROWIRE™ Serial Interface

National Semiconductor
Application Note 452
Abdul Aleaf
January 1992

INTRODUCTION

MICROWIRE is a simple three-wire serial communications interface. Built into COPS™, this standardized protocol handles serial communications between controller and peripheral devices. In this application note are some clarifications of MICROWIRE logical operation and of hardware and software considerations.

LOGICAL OPERATION

The MICROWIRE interface is essentially the serial I/O port on COPS microcontrollers, the SIO register in the shift register mode. SI is the shift register input, the serial input line to the microcontroller. SO is the shift register output, the serial output line to the peripherals. SK is the serial clock; data is clocked into or out of peripheral devices with this clock.

It is important to examine the logical diagram of the SIO and SK circuitry to fully understand the operation of this I/O port *(Figure 1)*.

The output at SK is a function of SYNC, EN0, CARRY, and the XAS instruction. If CARRY had been set and propagated to the SKL latch by the execution of an XAS instruction, SYNC is enabled to SK and can only be overridden by EN0 *(Figure 2)*. Trouble could arise if the user changes the state of EN0 without paying close attention to the state of the latch in the SK circuit.

If the latch is set to a logical high and the SIO register enabled as a binary counter, SK is driven high. From this state, if the SIO register is enabled as a serial shift register, SK will output the SYNC pulse immediately, without any intervening XAS instruction.

The SK clock (SYNC pulse) can be terminated by issuing an XAS instruction with CARRY = 0 *(Figure 3)*.

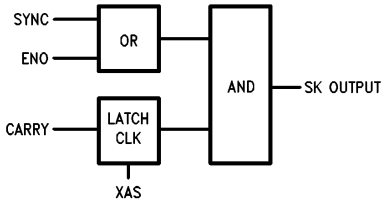


FIGURE 1. Logical Diagram of SK Circuit

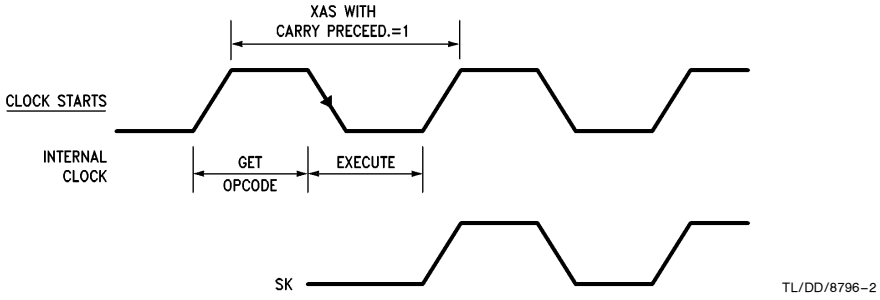


FIGURE 2. SK Clock Starts

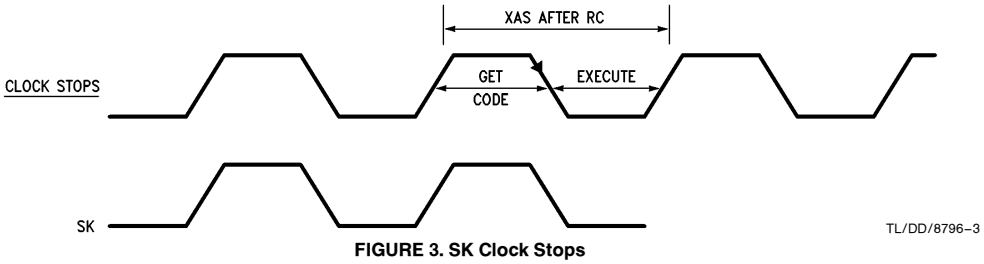


FIGURE 3. SK Clock Stops

The SIO register can be compared to four master-slave flip-flops shown in *Figure 4*. The masters are clocked by the rising edges of the internal clock. The slaves are clocked by the falling edges of the internal clock. Upon execution of an XAS, the outputs of the masters are exchanged with the contents of the accumulator (read and overdrive) in such a way that the new data are present at the inputs of the four slaves when the falling edge of the internal clock occurs. The content of the accumulator is, therefore, latched respectively in the four slave flip-flops and bit 3 appears directly on SO.

This means that:

a) SO will be shifted out upon the falling edges of SK and will be stable during rising edges of SK.

b) SI will be shifted in upon the rising edge of SK, and will be stable when executing, i.e., an XAS instruction.

The shifting function is automatically performed on each of the four instruction cycles that follow an XAS instruction *(Figure 5)*.

When the SIO register is in the shift register mode (EN0 = 0), it left shifts its contents once each instruction cycle. The data present on the SI input is shifted into the least significant bit (bit 0) of the serial shift register. SO will output the most significant bit of the SIO register (bit 3) if EN3 = 1. Otherwise, SO is held low. The SK is a logic controlled clock which issues a pulse each instruction cycle. To ensure that the serial data stream is continuous, an XAS instruction must be executed every fourth time. Serial I/O timing is related to instruction cycle timing in the following way:

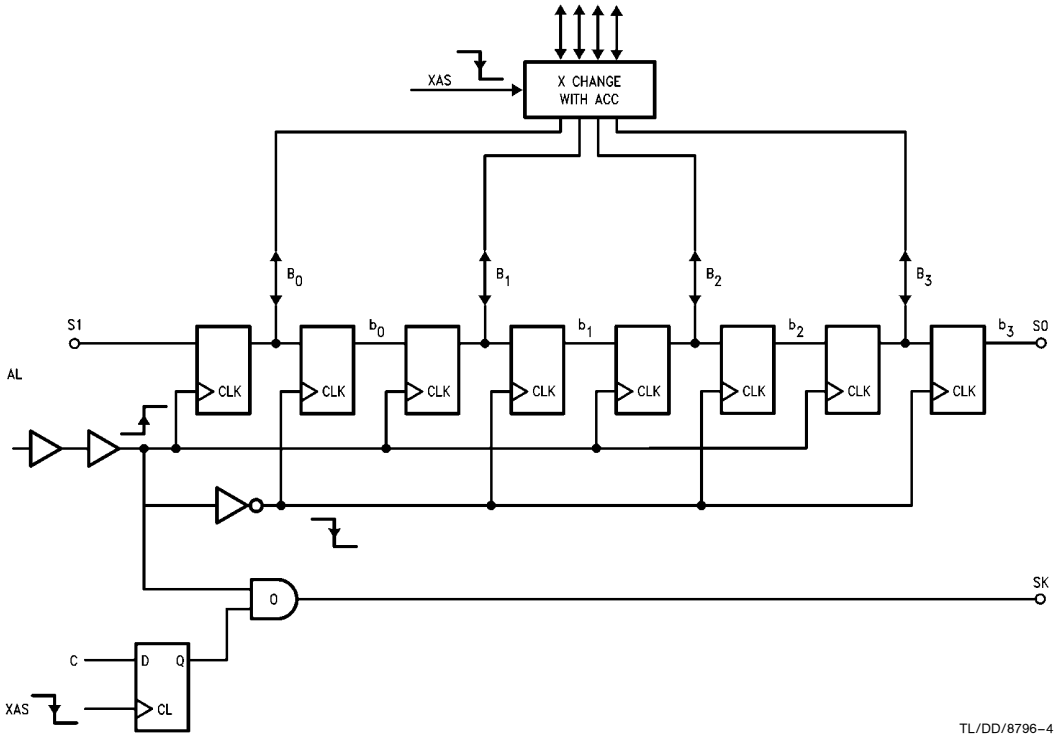


FIGURE 4

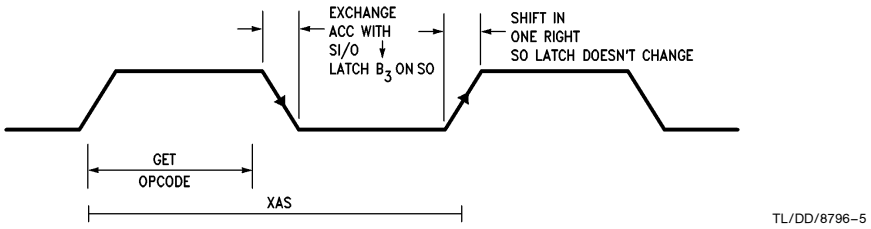
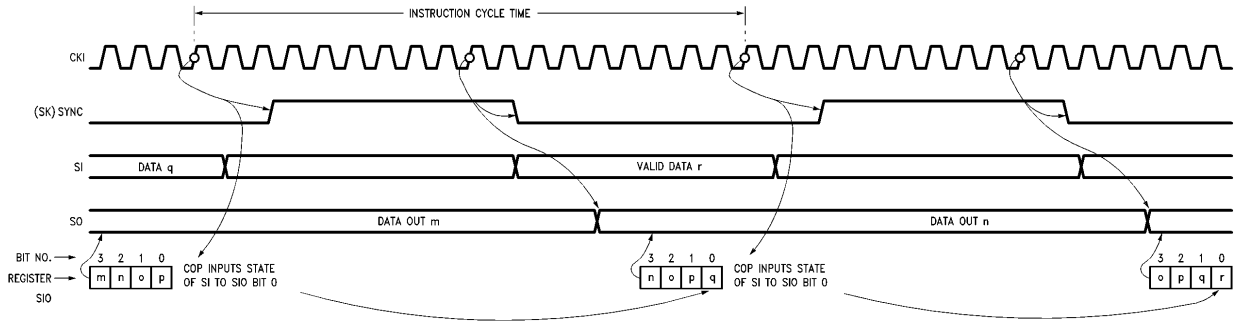
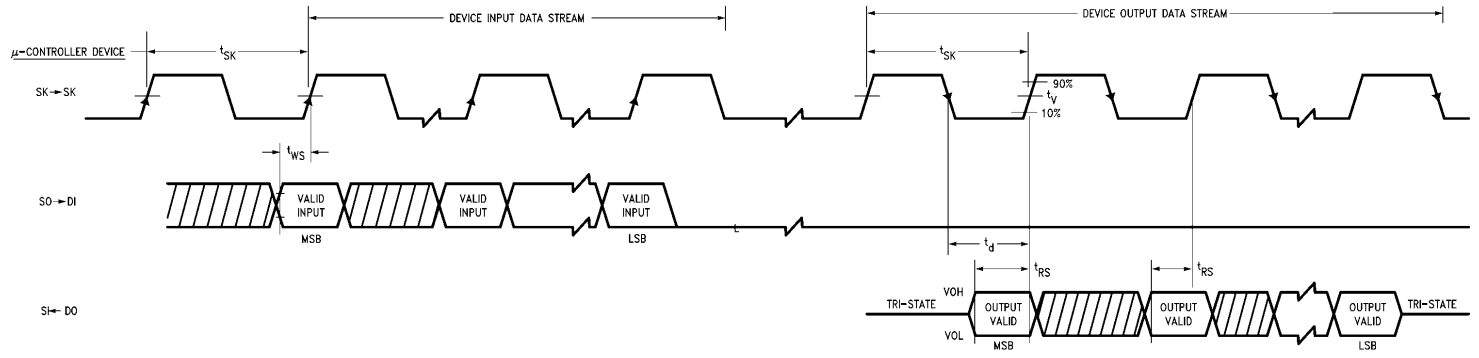


FIGURE 5. XAS Sequence



TL/DD/8796–6

FIGURE 6. Serial I/O Timing



TL/DD/8796–7

FIGURE 7. MICROWIRE Serial Data Exchange Timing

To write to device: $t_{ns} > t_{setup}$

To read from device: $t_d \ll t_{SK} - t_r$; $t_{RS} > t_{SK}/4$

Where: t_{ws} is MICROWIRE write data-in (DI) setup time,

t_{setup} is device data sheet min data setup time to latch in valid data,

t_{SK} is system clock (SK) cycle time (Recommended 50% duty cycle),

t_r is rise time (10% to 70% bout) of system clock (SK),

t_d is device actual delay time before data-out (DO) valid and

t_{RS} is minimum data setup time for controller to shift-in valid data

The first clock rising edge of the instruction cycle triggers the low-to-high transition of SYNC output via SK. At this time, the processor reads the state of SI into SIO bit 0, shifting the current bits 0–2 left. Halfway through the cycle (shown in *Figure 6* as the eight clock rising edge), SK is reset low and the new SIO bit 3 is outputted via SO.

INTERFACING CONSIDERATIONS

To ensure data exchange, two aspects of interfacing have to be considered: 1) serial data exchange timing; 2) fan-out/fan-in requirements. Theoretically, infinite devices can access the same interface and be uniquely enabled sequentially in time. In practice, however, the actual number of devices that can access the same serial interface depends on the following: system data transfer rate, system supply requirement capacitive loading on SK and SO outputs, the fan-in requirements of the logic families or discrete devices to be interfaced.

HARDWARE INTERFACE

Provided an output can switch between a HIGH level and a LOW level, it must do so in a predetermined amount of time for the data transfer to occur. Since the transfer is strictly synchronous, the timing is related to the system clock (SK) *(Figure 7)*. For example, if a COPS controller outputs a value at the falling edge of the clock and is latched in by the peripheral device at the rising edge, then the following relationship has to be satisfied:

$$t_{DELAY} + t_{SETUP} \leq t_{CK}$$

where t_{CK} is the time from data output starts to switch to data being latched into the peripheral chip, t_{SETUP} is the setup time for the peripheral device where the data has to be at a valid level, and t_{DELAY} is the time for the output to read the valid level. t_{CK} is related to the system clock provided by the SK pin of the COPS controller and can be increased by increasing the COPS instruction cycle time.

The maximum t_{SETUP} is specified in the peripheral chip data sheets. The maximum t_{DELAY} allowed may then be derived from the above relationship.

Most of the delay time before the output becomes valid comes from charging the capacitive load connected to the output. Each integrated circuit pin has a maximum load of 7 pF. Other sources come from connecting wires and connection from PC boards. The total capacitive load may then be estimated. The propagation delay values given in data sheets assume particular capacitive loads (e.g. $V_{CC} = 5V$, $V_{OH} = 0.4V$, loading = 50 pF, etc.).

If the calculated load is less than the given load, those values should be used. Otherwise, a conservative estimate is to assume that the delay time is proportional to the capacitive load.

If the capacitive load is too large to satisfy the delay time criterion, then three choices are available. An external buffer may be used to drive the large load. The COPS instruction cycle may be slowed down. An external pullup resistor may be added to speed up the LOW level to HIGH level transition. The resistor will also increase the output LOW level and increase the HIGH level to LOW level transition time, but the increased time is negligible as long as the output LOW level changes by less than 0.3V. For a 100 pF load, the standard COPS controller may use a 4.7k external resistor, with the output LOW level increased by less than 0.2V. For the same load, the low power COPS controller may use a 22k resistor, with the SO and SK LOW levels increased by less than 0.1V.

Besides the timing requirements, system supply and fan-out/fan-in requirements also have to be considered when interfacing with MICROWIRE. For the following discussion, we assume single supply push-pull outputs for system clock (SK) and serial output (SO), high-impedance input for serial input (SI).

To drive multi-devices on the same MICROWIRE, the output drivers of the controller need to source and sink the total maximum leakage current of all the inputs connected to it and keep the signal level within the valid logic ''1'' or logic ''0''. However, in general, different logic families have different valid ''1'' and ''0'' input voltage levels. Thus, if devices of different types are connected to the same serial interface, the output driver of the controller must satisfy all the input requirements of each device. Similarly, devices with TRI-STATE® outputs, when connected to the SI input, must satisfy the minimum valid input level of the controller and the maximum TRI-STATE leakage current of all outputs.

So, for devices that have incompatible input levels or source/sink requirements, external pull-up resistors or buffers are necessary to provide level-shifting or driving.

SOFTWARE INTERFACE

The existing MICROWIRE protocol is very flexible, basically divided into two groups:

1) 1AAA.....ADDD.....D

where leading 1 is the start bit and leading zeroes are ignored.

AAA.....A is device variable instruction/address word.

DDD.....D is variable data stream between controller and device.

2) No start bit, just bit stream, i.e., bbb.....b

where b is a variable bit stream. Thus, device has to decode various fields within the bit stream by counting exact bit position.

SERIAL I/O ROUTINES

Routines for handling serial I/O are provided below. The routines are written for 16-bit transmissions, but are trivially expandable up to 64-bit transmissions by merely changing the initial LBI instruction. The routines arbitrarily select register 0 as the I/O register. It is assumed that the external device requires a logic low chip select. It is further assumed that chip select is high and SK and SO are low on entry to the routines. The routines exit with chip select high, SK and SO low. GO is arbitrarily chosen as the chip select for the external device.

SERIAL DATA OUTPUT

This routine outputs the data under the conditions specified above. The transmitted data is preserved in the microcontroller.

```
OUT2:  LBI   0,12   ; point to start of
                      data word
       SC
       OGI   14     ; select the external
                      device
```

TABLE I. MICROWIRE Standard Family

Features		Part Number				
		DS8908	**MM545X**	**COP472-3**	**COP430 (ADC83X)**	**NM93C06A**
GENERAL						
Chip Function		AM/PM PLL	LED Display Driver	LCD Display Driver	A/D	E^2PROM
Process		ECL	NMOS	CMOS	CMOS	NMOS
V_{CC} Range		4.75V–5.25V	4.5V–11V	3.0V–5.5V	4.5V–0.3V	4.5V–5.5V
Pinout		20	40	20	8/14/20	14
HARDWARE INTERFACE						
Min V_{IH}/Max V_{IL}		2.1V/0.7V	2.2V/0.8V	$0.7V_{CC}$/0.8V	2.0V/0.8V	2.0V/0.8V
SK Clock Range		0–625 kHz	0–500 kHz	4–250 kHz	10–200 kHz	0–250 kHz
Write Data DI	Setup Min	0.3 μs	0.3 μs	1 μs	0.2 μs	0.4 μs
	Hold Min	0.8 μs	(3)	100 ns	0.2 μs	0.4 μs
Read Data Prop Delay		(Note 4)	(Note 3)	(Note 3)	(Note 3)	
Chip Enable	Setup	0.3 μs	0.4 μs	1 μs (Note 1)	0.2 μs	0.2 μs
	HOLD	0.8 μs	(Note 3)	1 μs (Note 2)	0.2 μs	0
Max Frequency Range	AM	8 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)
	FM	120 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)
SOFT						
Serial I/O Protocol		11D1...D20	1D1...D35	b1...b40	1xxx	1AA...DD
Instruction/Address Word		None	None	None	(Note 4)	(Note 4)

Note 1: Reference to SK rising edge.

Note 2: Reference to SK falling edge.

Note 3: Not defined.

Note 4: See data sheet for different modes of operation.

```
        LEI  8      ; enable shift
                      register mode
        JP   SEND2
SEND1:  XAS
SEND2:  LD          ; data output loop
        XIS
        JP   SEND1
        XAS         ; send last data
        RC
        CLRA
        NOP
        XAS         ; turn SK clock off
        OGI  15     ; deselect the device
        LEI  0      ; turn SO low
        RET
```

The code for reading serial data is almost the same as the serial output code. This should be expected because of the nature of the SIO register and the XAS instruction.

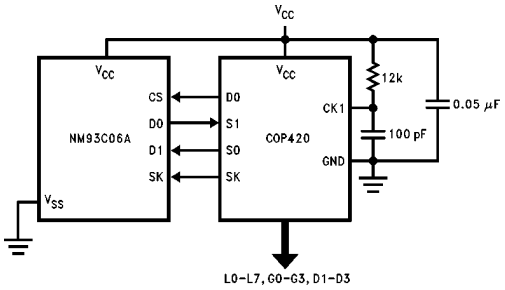
MICROWIRE STANDARD FAMILY

A whole family of off-the-shelf devices exists that is directly compatible with MICROWIRE serial data exchange standard. This allows direct interface with the COPS family of microcontrollers.

Table I provides a summary of the existing devices and their functions and specifications.

TYPICAL APPLICATION

Figure 8 shows pin connection involved in interfacing an E^{2}PROM with the COP420 microcontroller.

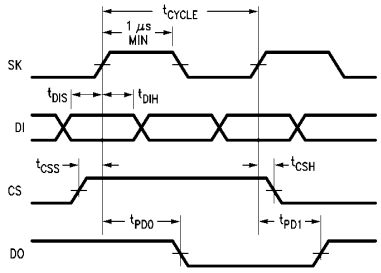


TL/DD/8796–8

FIGURE 8. NM93C06A COP420 Interface

The following points have to be considered:

1. For NM93C06A the SK clock frequency should be in the 0 kHz–250 kHz range. This is easily achieved with COP420 running at 4 µs–10 µs instruction cycle time (SK period is the COP420 instruction cycle time). Since the minimum SK clock high time is greater than 1 µs, the duty cycle is not a critical factor as long as the frequency does not exceed the 250 kHz max.
2. CS low period following an E/W instruction must not exceed 30 ms maximum. It should be set at typical or minimum spec of 10 ms. This is easily done in software using the SKT timer on COP420.



TL/DD/8796–9

FIGURE 9. NM93C06A Timing

3. As shown in WRITE timing diagram, the start bit on DI must be set by a "0" to "1" transition following a CS enable ("0" to "1") when executing any instruction. One CS enable transition can only execute one instruction.
4. In the read mode, following an instruction and data train, the DI can be a "don't care," while the data is being outputted, i.e., for the next 17 bits or clocks. The same is true for other instructions after the instruction and data has been fed in.
5. The data-out train starts with a dummy bit 0 and is terminated by chip deselect. Any extra SK cycle after 16 bits is not essential. If CS is held on after all 16 of the data bits have been outputted, the DO will output the state of DI until another CS LO to HI transition starts a new instruction cycle.
6. After a read cycle, the CS must be brought low for one SK clock cycle before another instruction cycle starts.

INSTRUCTION SET

Commands	Opcode	Comments
READ	10000A3A2A1A0	Read Register 0–15
WRITE	11000A3A2A1A0	Write Register 0–15
ERASE	10100A3A2A1A0	Erase Register 0–15
EWEN	111000 0 0 1	Write/Erase Enable
ENDS	111000 0 1 0	Write/Erase Disable
***WRAL	111000 1 0 0	Write All Registers
ERAL	111000 1 0 1	Erase All Registers

All commands, data in, and data out are shifted in/out on rising edge of SK clock.

Write/erase is then done by pulsing CS low for 10 ms.

All instructions are initiated by a LO-HI transition on CS followed by a LO-HI transition on DI.

READ— After read command is shifted in DI becomes don't care and data can be read out on data out, starting with dummy bit zero.

WRITE— Write command shifted in followed by data in (16 bits) then CS pulsed low for 10 ms minimum.

ERASE

ERASE ALL—Command shifted in followed by CS low.

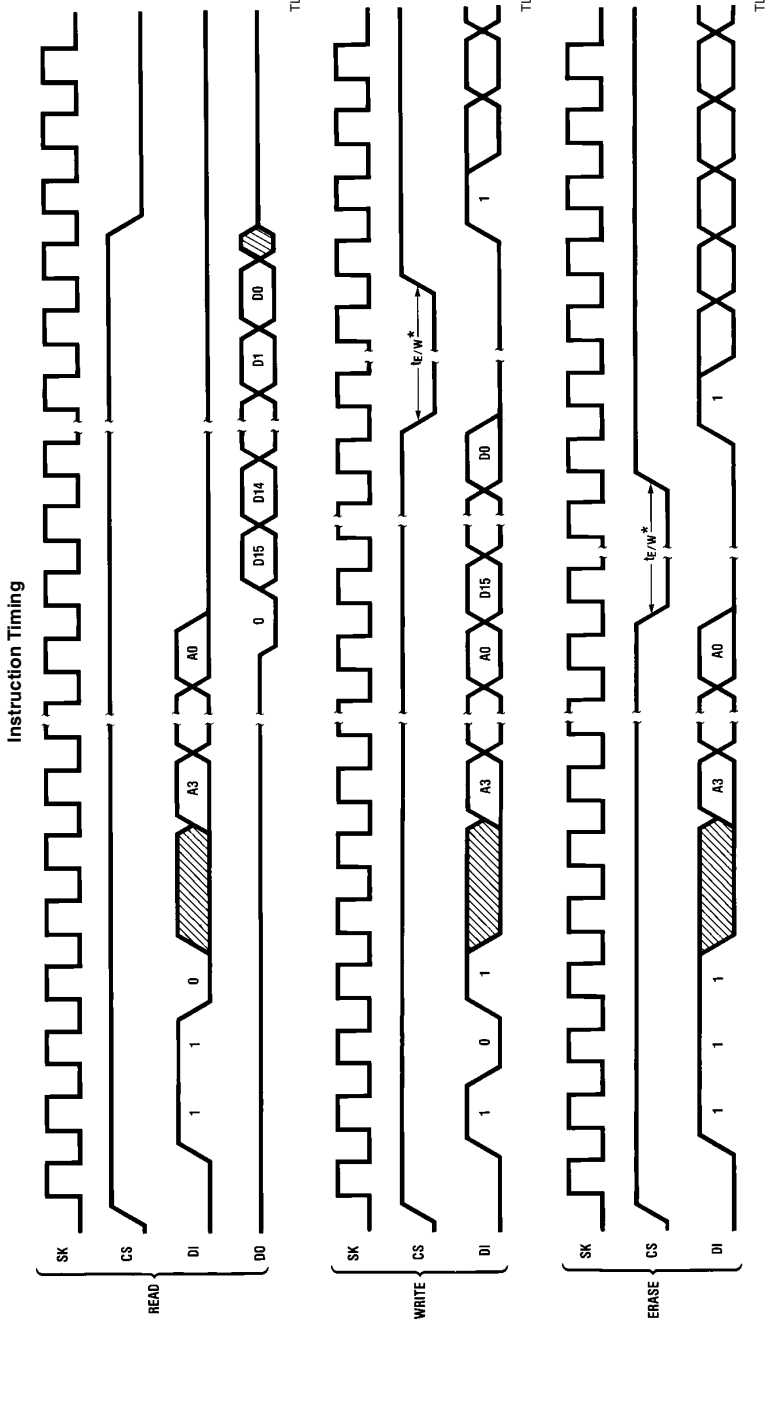
WRITE ALL—Pulsing CS low for 10 ms.

WRITE

ENABLE/DISABLE—Command shifted in.

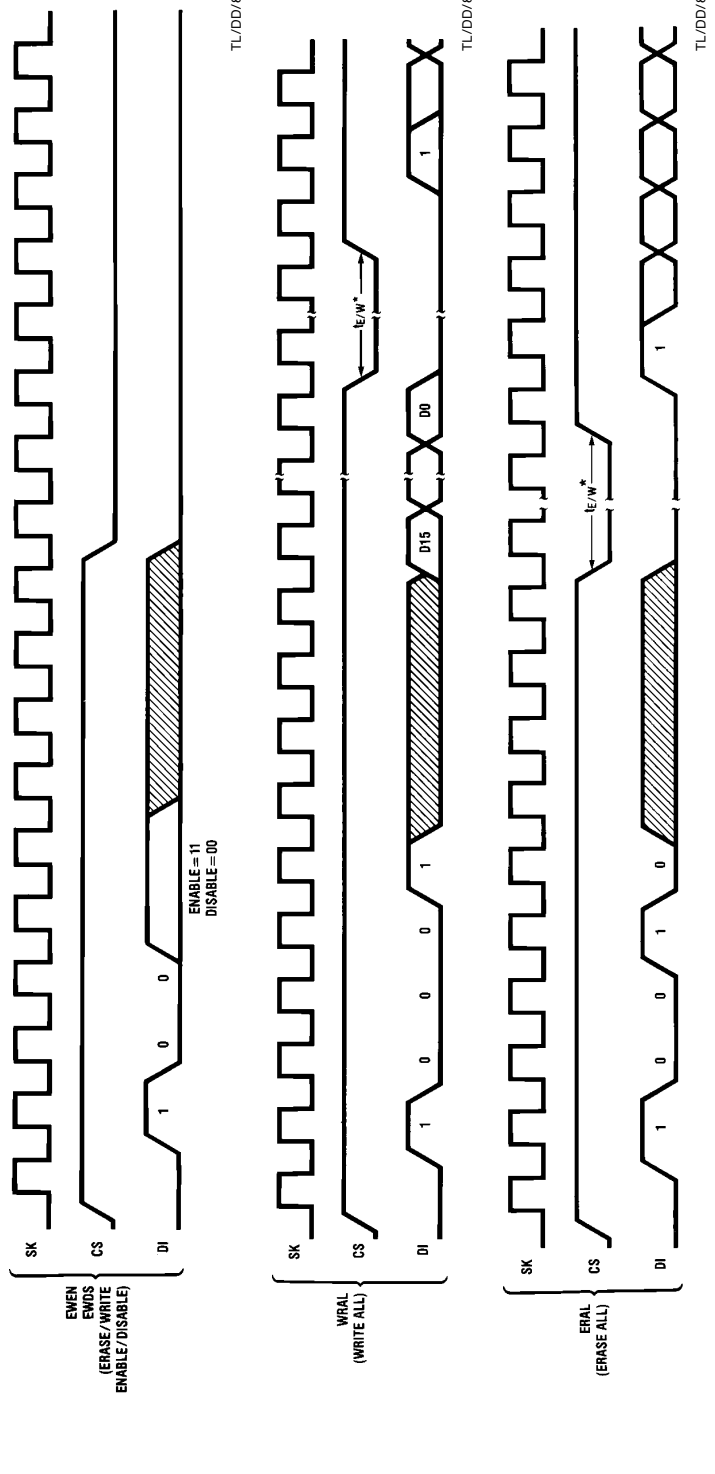
***(This instruction is not speced on Data sheet.)

TIMING WAVEFORMS



*$t_{E/W}$ measured to rising edge of SK or CS, whichever occurs last.

TIMING WAVEFORMS (Continued)



*$t_{E/W}$ measured to rising edge of SK or CS, whichever occurs last.

I/O ROUTINE TO EVALUATE COP494

```
 1                      .TITLE      E494,  "I/O ROUTINE TO EVALUATE COP494"
 2     01A4             .CHIP       420
 3     0000             .PAGE       0
 4
 5              ;THIS IS I/O ROUTINE TO EVALUATE COP494
 6              ;
 7              ;
 8
 9              ;RAM VARIABLES DECLARATIONS:
10     000E             COMMAND = 0, 14          ;494 8BITS INST/ADDR WORD
11     001C             RWDATA  = 1, 12          ;494 16BITS R/W DATA BUFFER
12
13 000 00       PON:    CLRA                     ;POWER-ON INIT
14 001 32               RC                       ;RESET SK CLOCK
15 002 4F               XAS
16 003 3F       CLRAM:  LBI     3, 0             ;CLEAR RAM FROM 7, 0 TO 0, 15
17 004 00       CLR:    CLRA                     ;
18 005 04               XIS                      ;
19 006 C4               JP      CLR              ;CONTI CLEAR REG
20 007 12               XABR                     ;(A) TO BR
21 008 5F               AISC    15               ;REG 0 CLEARED?
22 009 600F     DONE:   JMP     C494DR           ;Y, DONE CLEAR RAM, CALL 494 D
23 00B 12               XABR                     ;N, DEC BR
24 00C C4               JP      CLR              ;CONTI CLEAR REG TILL DONE
25 00D 44               NOP
26 00E 44               NOP
27
28              ;***    START 494 DRIVER SAMPLE CALLING SEQUENCE   ***
29
30              C494DR:                          ;INIT CALLING SEQUENCE
31 00F 3350             OGI     0                ;GO=L TO DESELECT 494
32 011 3368             LEI     8                ;ENABLE SIO AS S.R.
33              ERASE:
34 013 0D               LBI     COMMAND          ;PRELOAD 494 ERASE REG A3-A0
35 014 7C               STII    0C               ;PRELOAD 494 ERASE INST
36 015 70               STII    0                ;SELECT REG A3-A0
37 016 690E             JSR     WI4P4            ;SEND IT
38              WEEN:
39 01B 0D               LBI     COMMAND          ;LOAD 494 WHEN REG A3-A0
40 019 73               STII    3                ;PRELOAD 494 WREN INST
41 01A 70               STII    0                ;SELECT REG A3-A0
42 01B 690E             JSR     WI494            ;SEND IT
43              WRITE:
44 01D 0D               LBI     COMMAND          ;PRELOAD WR REG A3-A0
45 01E 74               STII    4                ;PRELOAD 494 WRITE INST
46 01F 70               STII    0                ;SELECT REG A3-A0
47 020 1B               LBI     RWDATA           ;PRELOAD 494 SAMPLE WRITE DATA
48 021 75               STII    5
49 022 7A               STII    ON
50 023 75               STII    5
```

I/O ROUTINE TO EVALUATE COP494 (Continued)

```
 51 024 7A             STII    0A
 52 025 6900           JSR     WD494             ;SEND THEM TO 494
 53          READ:
 54 027 0D             LBI     COMMAND           ;PRELOAD READ REG A3-A0
 55 028 78             STII    8                 ;PRELOAD 494 READ INST
 56 029 70             STII    0                 ;SELECT REG A3-A0
 57 02A 6908           JSR     RD494             ;READ 494 DATA BACK VIA SI
 58 02C 44             NOP
 59 02D 44             NOP
 60
 61     0080           .PAGE   2                 ;SUBROUTINE PAGE
 62 080 32   SETUP:    RC                        ;RESET SK BEFORE SELECT 494
 63 081 4F             XAS
 64 082 3351           0GI     1                 ;G0=1 TO SELECT 494
 65 084 00             CLRA                      ;ENSURE SO=L BEFORE GEN START B
 66 085 22             SC                        ;
 67 086 4F             XAS                       ;TURN ON SK CLOCK
 68 087 00             CLRA                      ;GENERATE 494 START BIT
 69 088 51             AISC    1                 ;
 70 089 22             SC                        ;
 71 08A 4F             XAS                       ;SEND IT AS MSB VIA SO
 72 08B 0D             LBI     COMMAND           ;FETCH 1ST INST/ADDR WORD
 73 08C 05             LD
 74 08D 44             NOP                       ;
 75 08E 4F             XAS                       ;SEND IT (MSB OF INST FIRST)
 76 08F 0E             LBI     COMMAND+1         ;FETCH 2ND INST/ADDR NIBBLE
 77 090 05             LD
 78 091 44             NOP
 79 092 4F             XAS                       ;SEND IT
 80 093 1B             LBI     RWDATA            ;POINT TO READ/WRITE DATA BUFFER
 81 094 48             RET                       ;RET OF SETUP
 82
 83 095 00   TWEDLY:   CLRA                      ;VPP WIDTH, TWE>20MS @ 4Us/INST
 84 096 5B   TWECONT:  AISC    11                ;5 SKT LOOPS?
 85 097 99             JP      . + 2             ;N,CONTI
 86 098 48   TWEDONE:  RET                       ;Y,DONE
 87 099 41             SKT                       ;
 88 09A 99             JP      . -1              ;
 89 09B 96             JP      TWECONT           ;CONTI TWE TIME
 90
 91 09C 48   RET;      RET                       ;2 CYCLES DELAY
 92
 93     0100           .PAGE   4                 ;
 94
 95          ;***      START 494 I/O DRIVER SUBROUTINE    ***
 96
 97 100 80   WD494:    JSRP    SETUP             ;ENTRY TO WRITE 494 REG A3-A0
 98 101 05   RWLOOP:   LD                        ;R/W 494 16 DATA BITS
 99 102 4F             XAS                       ;
100 103 04             XIS                       ;
```

I/O ROUTINE TO EVALUATE COP494 (Continued)

```
101 104 C1           JP    RWLOOP
102 105 3350         OGI   0         ;DESELECT 494 AFTER R/W DATA
103 107 D1           JP    FINI      ;
104 108 80    RD494: JSRP  SETUP     ;ENTRY TO RD 494 REG A3-A0
105 109 00           CLRA            ;FINISH SEND OUT A3-A0 VIA SO
106 10A 44           NOP             ;
107 10B 44           NOP             ;WAIT 1BIT TIME FOR VALID D15
108 10C 44           NOP
109 10D C1           JP    RWLOOP      ;
110 10E 80    WI494: JSRP  SETUP     ;ENTRY TO WRITE INST TO 494
111 10F 00           CLRA            ;ENSURE SO = L
112 110 4F           XAS             ;
113 111 00    FINI:  CLRA            ;ENSURE SO = L BETWEEN INST
114 112 3350         OGI   0         ;DESELECT 494 BETWEEN INST WRIT
115 114 32           RC              ;
116 115 4F           XAS             ;TURN OFF SK CLOCK
117 116 95           JSRP  TWEDLY    ;DELAY TWE >20MS TO PULSE VPP=21
118 117 48           RET             ;RET OF WD494 OR RD494 OR WI494
119
120                  .END
```

SOFTWARE DEBUG OF SERIAL REGISTER FUNCTIONS

In order to understand the method of software debug when dealing with the SIO register, one must first become familiar with the method in which the COPS MOLE™ (Development System) BREAKPOINT and TRACE operations are carried out. Once these operations are explained, the difficulties which could arise when interrogating the status of the SIO register should become apparent.

SERIAL OUT DURING BREAKPOINT

When the MOLE BREAKPOINTs, the COPS user program execution is stopped and execution of a monitor-type program, within the COP device, is started. At no time does the COP part ''idle.'' The monitor program loads the development system with the information contained in the COP registers.

Note also that single-step is simply a BREAKPOINT on every instruction.

If the COP chip is BREAKPOINTed while a serial function is in progress, the contents of the SIO register will be destroyed.

By the time the monitor program dumps the SIO register to the MOLE, the contents of the SIO register will have been written over by clocking in SI. To inspect the SIO register using BREAKPOINT, an XAS must be executed prior to BREAKPOINT; therefore, the SIO register will be saved in the accumulator.

An even more severe consequence is that the monitor program executes an XAS instruction to get the contents of the SIO register to the MOLE. Therefore, the SK latch is dependent on the state of the CARRY prior to the BREAKPOINT. In order to guarantee the integrity of the SIO register, one must carefully choose the position of the BREAKPOINT address.

As can be seen, it is impossible to single-step or BREAKPOINT through a serial operation in the SIO register.

SERIAL OUT DURING TRACE

In the TRACE mode, the user's program execution is never stopped. This mode is a real-time description of the program counter and the external event lines; therefore, the four external event lines can be used as logic analyzers to monitor the state of any input or output on the COPS device. The external event lines must be tied to the I/O which is to be monitored.

The state of these I/O (external event lines) is displayed along with the TRACE information. The safest way to monitor the real-time state of SO is to use the TRACE function in conjunction with the external event lines.

CONCLUSIONS

National's super-sensible MICROWIRE serial data exchange standard allows interfacing to any number of specialized peripherals using an absolute minimum number of valuable I/O pins; this leaves more I/O lines available for system interfacing and may permit the COPS controller to be packaged in a smaller package.

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

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