Using Memory-Mapped I/O with the NS32GX32 or NS32532

National Semiconductor
Application Note 592
Itzhak Nashelsky
March 1989

1.0 INTRODUCTION

The NS32GX32 and NS32532 implement memory-mapped I/O to handle peripheral I/O devices. In memory-mapped I/O, each peripheral device is mapped to a specific block of microprocessor memory. As a result, any of the processor's instructions and addressing modes can be used to communicate with the peripheral device. For example, a memory STORE command can be used to initiate an I/O write operation. Similarly, a memory LOAD command can invoke an I/O read.

The biggest advantage of memory-mapped I/O is that all I/O operations can be performed directly by a high-level language (HLL). This differs from isolated I/O systems which use assembly language subroutines for I/O. Another advantage of memory-mapped I/O is that, in virtual memory systems, the I/O space can be protected by the same memory protection mechanisms used for important blocks of microprocessor memory.

2.0 DESTRUCTIVE READING AND SIDE-EFFECTS OF WRITING

The NS32GX32 executes instructions in a heavily pipelined fashion. This greatly enhances performance since the operations of several instructions are performed simultaneously rather than sequentially. Thus, operations like fetching one instruction, reading the source operands of second instruction, calculating the results of a third instruction, and storing the results of a fourth instruction, can all occur in parallel.

However, because of the instruction pipeline and the nature of some peripheral devices, special handling is required for memory-mapped I/O. The methods implemented by the NS32GX32 handle two consequences of referencing memory-mapped I/O *destructive reading* and the *side-effects of writing*.

Destructive reading occurs when reading from a peripheral port alters the value read on the next reference to the same port or to another port in the same device. Serial communication controllers and FIFO buffers commonly operate in this manner.

Side-effects of writing occur when writing to a peripheral port alters the value read from another port of the same device. For example, before reading the counter's value from the NS32202 Interrupt Control Unit, it is first necessary to freeze the value by writing to another control register. As a side-effect, the value of the counter may be altered.

*The NS3532 contains an on-chip Memory Management Unit (MMU) and supports virtual memory, the NS32GX32 does not. This application note primarily discusses the NS32GX32 but applies to the NS32532 as well, except where noted otherwise.

3.0 MEMORY-MAPPED I/O REQUIREMENTS

The instruction pipeline of the NS32GX32 can read the source operands for one instruction while the previous instruction is executing. However, during the execution of the previous instruction, the flow of control may be altered by a trap, interrupt, or other circumstance. Because of such situations *destructive reading* of source operands before the execution of an instruction must be avoided.

The NS32GX32 can also read the source operands for one instruction before writing the results of previous instructions, unless the source operand values depend on results not yet written. Consequently, read and write references to peripheral devices that exhibit *side-effects of writing* must occur in the order dictated by the instructions.

There are two methods for handling memory-mapped I/O using the NS32532 or NS32GX32. The first method is more general and satisfies both requirements listed above and places no restriction on the location of memory-mapped peripheral devices. This method works with both the NS32GX32 and NS32532. The second method satisfies only the requirement for ''side-effects of writing'', and it restricts the location of memory-mapped I/O devices. However, this method is more efficient for devices that do not have destructive-read ports.

3.1 Method 1

The first method of handling memory-mapped I/O uses two signals: IOINH and IODEC. When the NS32GX32 generates a read bus cycle, it asserts the output signal IOINH if either of the I/O requirements listed above is not satisfied. That is, IOINH is asserted during a read bus cycle when:

- The read reference is for an instruction that may not be executed, or
- The read reference occurs while a write reference is pending for a previous instruction.

Note: When the read reference is made to a peripheral device with ''destructive-reading'' or ''side-effects of writing'' ports, the input signal IODEC must be asserted. In addition, if IOINH is active, the peripheral device must not be selected.

When the CPU detects that the IODEC input signal is active while the IOINH output signal is also active, it discards the data read during the bus cycle and serializes instruction execution. The CPU then generates the read bus cycle again (this time satisfying the requirements for memory-mapped I/O) and drives IOINH inactive. This application note contains an example using this approach which is based on the design of the National Semiconductor VME532, a VME-bus CPU board based on the NS32532.

3.2 Method 2

The second method for handling memory-mapped I/O uses a dedicated 8 Mbyte region of memory (FF000000 to FFFFFFFF, inclusive).* The processor treats all references to this region of memory in a special manner:

- While a write to a location in this region is pending, reads from locations in the same region are delayed. However, reads from locations with addresses lower than FF000000 may occur.
- Similarly, reads from locations in the region may occur while writes to locations outside of the region are pending.

Note: The CPU may assert IOINH even when the reference is within the dedicated region.

*In the case of the NS32532 this region is in virtual memory.

RRD-B30M75/Printed in U. S. A.

4.0 EXAMPLE OF MEMORY INTERFACE USING METHOD 1

4.1 General Design Guidelines

When designing an I/O subsystem around the NS32GX32, the following should be observed:

- Pre-read operations should be inhibited if ''destructive-read'' I/O devices are being used (e.g., FIFO-contained UARTS).
- In such cases, ~IODEC and ~IOINH signals are used to cancel I/O accesses. For more information, refer to the NS32GX32 Data Sheet.
- At I/O access completion, the I/O subsystem must fulfill the address hold-time required by most of the programmable devices, even though the CPU has already progressed to the next bus cycle.

  This hold time can be provided by:

  1. Latching the address applied to the I/O subsystem, and keeping it valid for a couple of clock cycles after each I/O access; and/or
  2. Not allowing the CPU to initiate a new I/O access until the hold time interval is completed.

4.2 Implementation Guidelines

The following describes a typical I/O block, emphasizing the interface with the CPU and the I/O state machine responsible for generation of control signals.

4.2.1 CPU Interface

Refer to Appendix B for a schematic of the interface to the NS32GX32. The main control signals that interface the CPU block and the I/O block are listed below. Note that the direction of a signal refers to the I/O block.

~IODEC: This input from the main address decoder indicates a request for an I/O access.

~RDY: This TRI-STATE® output is enabled during I/O access, asserting wait states to the CPU until access is completed.

~IOINH: When asserted by the CPU, this input indicates that the bus cycle is a pre-read. In such cases, the bus-cycle is cancelled by asserting ~RDY without starting an I/O access.

~RDYT1: This input is a time frame signal from the main address decoder which enables ~RDY during I/O operation.

BCLK: This is the CPU clock.

~BRST1: This input is the system reset, used to initiate the I/O subsystem.

~BMT: This input from the CPU indicates a beginning of a confirmed bus-cycle.

~BDDIN: An incoming signal from the CPU indicating the direction of the bus-cycle.

In addition, partial address and data buses are used for I/O device selection and bi-directional data transfer.

4.2.2 Hardware Description

Three PALs contain all the logic required to operate the I/O block. They are as follows:

RDYGEN: U3 generates ~RDY signal towards the CPU, and starts I/O operation.

IOSMG: U4 generates all the control signals and timing frames, synchronous to system clock.

IOADEC: U5 selects the required I/O device according to some low-order address bits.

In addition, two address latches (U1 and U2) are used to latch the address until the end of the transaction, and one bi-directional data buffer (U6) handles data transfers with the CPU.

4.2.3 Block Operation

The following describes the block operation for the VME532 I/O subsystem.

When I/O is decoded on T1, ~IODEC is activated. If ~IOINH is inactive, ~RDY is deasserted to keep the CPU in wait states. ~IOST is activated, triggering the state machine IOSMG.

The data buffers are enabled by ~IOST, with the direction controlled by ~BDDIN.

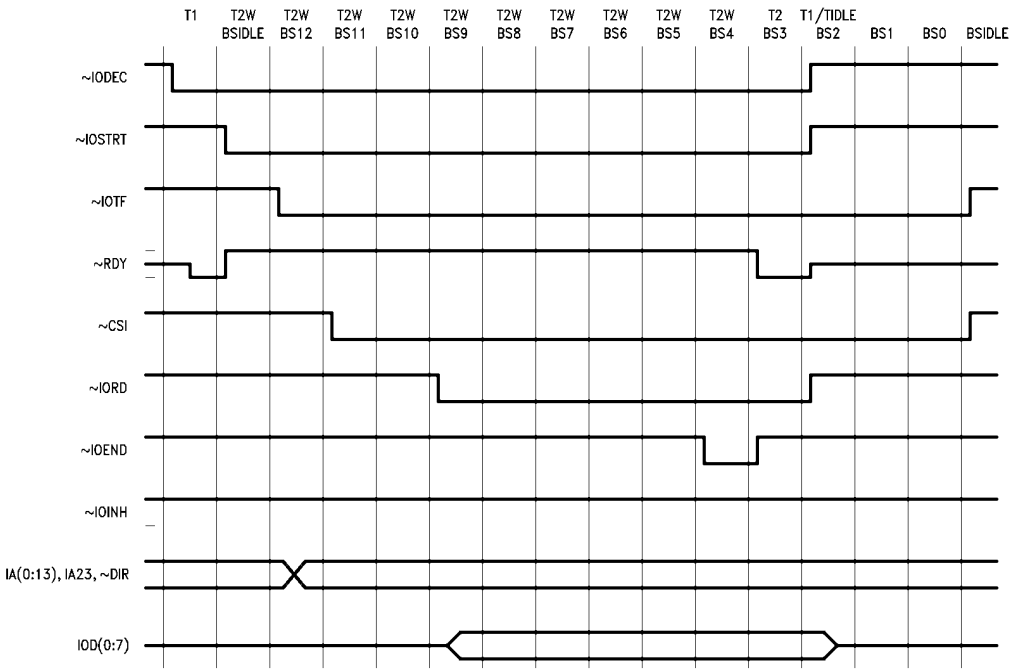
The I/O state machine is a simple 4-bit counter, activating some control signals until terminal count, when ~IOEND pulse is generated.

~IOEND pulse indicates I/O completion and causes ~RDY assertion.

Figure 1 shows the timing waveforms of a read transaction. For write transactions, ~IOEND is delayed to BS2 to fulfill data hold time requirements with respect to ~IOWR.

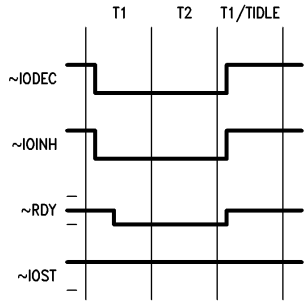
IOADEC PAL generates select signals to a variety of I/O devices: PROM, UART, seven-segment write-only latch, ready-only dip-switch buffer and an Interrupt Control Unit (ICU) the NS32202.

A. Non-Destructive Read



TL/EE/10360–1

B. Destructive Read



TL/EE/10360–2

FIGURE 1

Note: IOSMG can be optimized to various access times of the I/O devices, using feed-back signals from IOADEC. Those signals are used to skip some states of the counter when high-speed devices are available. IOSMG is optimized for 150 ns PROM devices, with hardware preparation for ICU and UART optimization.

Since ~RDY is a TRI-STATE signal, other participants may connect on the ~RDY line directly. The hand-over between ~RDY users is taking place during T1, when the NS32GX32 applies the new address.

Note: During hand-over, both participants should drive ~RDY low, in order to prevent contention.

APPENDIX A—PAL LISTINGS

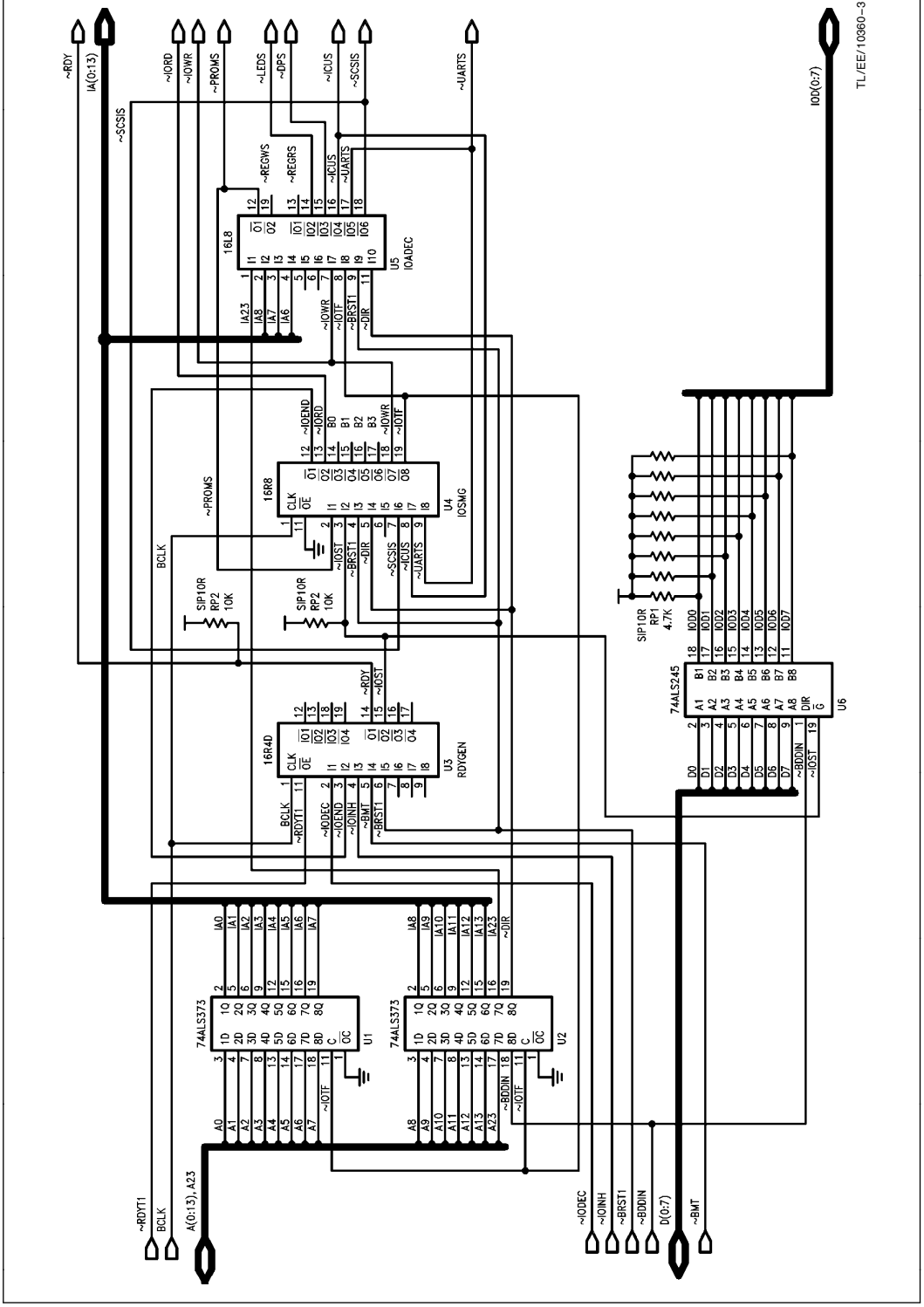
```
module rdygen
flag '-r3', '-t2', '-q2', '-s'
title 'ready generator       rdygen        izik nashelsky'
'Declarations
   RDYGEN            device 'P16R4';
CLK,OE                    pin 1,11;
pio,ioend,ioinh,bmt,brst  pin 2,3,4,5,6;
rdy,iostrt                pin 14,15;
   x,ck,h,l = .X.,.C., 1, 0;
equations
!iostrt := !bmt & !pio & brst & ioinh
        # !iostrt & rdy & brst;
!rdy    := !brst                    "catch on reset
        # !rdy & (!(!bmt & !pio & ioinh))
                 "keep active till io cycle starts (if pre-read
                 "                   - remain active
        # rdy & !ioend; "catch again when end of io cycle.
test_vectors (
[CLK,pio,ioend,ioinh,bmt,brst] → [rdy,iostrt])
[ck, x, x, x, x, 0] → [0, 1] ; "reset - initiate rdy
[ck, 0, 1, 1, 0, 1] → [1, 0] ; "start io cycle
[ck, 1, 0, 1, 1, 1] → [0, 0] ; "catch rdy again
[ck, 1, 1, 1, 1, 1] → [0, 1] ; "release iostrt
[ck, 0, 1, 0, 0, 1] → [0, 1] ; "pre-read - keep rdy active

end rdygen
```

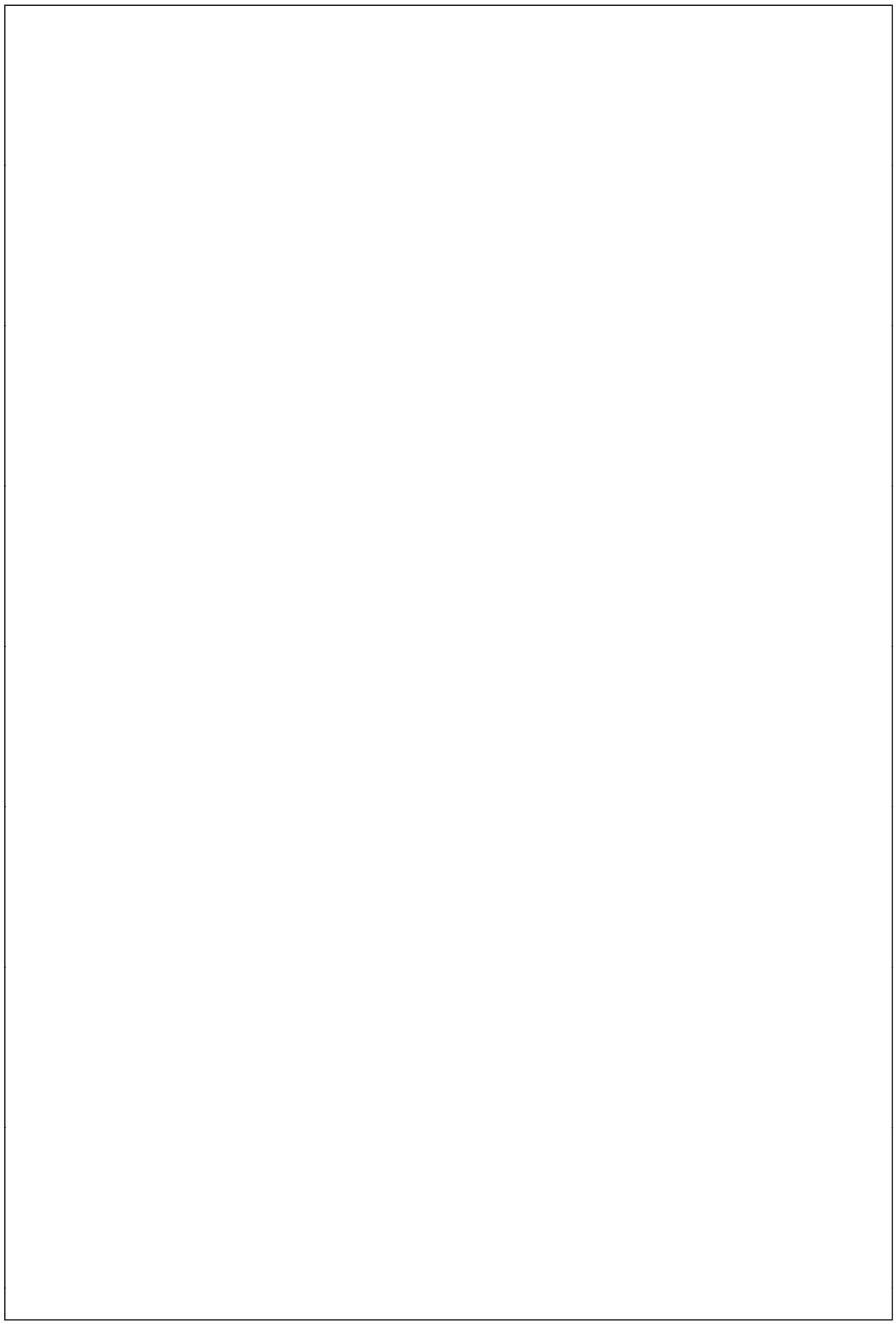
```
module iosmg
flag '-r3', '-t2', '-s', '-q2'
title 'io state machine       iosmg        izik nashelsky'
"Declarations
  IOSMG           device 'P16R8';
  CLK             pin 1;
  OE              pin 11;
"inputs:
  nBRST, nDIR     pin 4,5;
  nPROMS          pin 2;
  hIOSTRT         pin 3;
  nICUS, nDUARTS  pin 8,9;       "Preparations for optimization
"outputs:
  B3,B2,B1,B0     pin 17,16,15,14;     "Internal State.
  nIOTF,nIOEND    pin 19,12;         "IO Time Frame & End Pulse.
  nIOWR,nIORD     pin 18,13;        "IO Write & Read.
  nSCSIS          pin 7;
RST,DIR,STRT,C = nBRST,nDIR,nIOSTRT,CLK;
PROMS,IORD,IOWR,IOEND = nPROMS,nIORD,nIOWR,nIOEND;
c,x = .c.,.x.;
"State description of counter Bi
  BSi = [B3,B2,B1,B0];
  BSidle = 15;  BS11 = 11;  BS07 = 07;  BS03 = 03;
  BS14   = 14;  BS10 = 10;  BS06 = 06;  BS02 = 02;
  BS13   = 13;  BS09 = 09;  BS05 = 05;  BS01 = 01;
  BS12   = 12;  BS08 = 08;  BS04 = 04;  BS00 = 00;
State_Diagram BSi
"================
state BSidle:    case
                   (nBRST & nIOTF & !nIOSTRT):    BS12;
                 endcase;
state BS12:      goto BS11;
state BS11:      case
                   (!nPROMS):                     BS06;
                   ( nPROMS):                     BS10;
                 endcase;
state BS10:      goto BS09;
state BS09:      goto BS08;
state BS08:      goto BS07;
state BS07:      goto BS06;
state BS06:      goto BS05;
state BS05:      goto BS04;
state BS04:      goto BS03;
state BS03:      goto BS02;
state BS02:      goto BS01;
state BS01:      goto BS00;
state BS00:      goto BSidle;
```

```
Equations
"========
!nIOTF := (BSi == BSidle) & (!nIOSTRT & nIOTF & nBRST) "IO Time Frame.
       # (BSi != BS00) & (BSi != BSidle) & (!nIOTF & nBRST);
!nIORD := nBRST & !nDIR & (BSi >= BS04) & ( BSi <= BS10);
!nIOWR := nBRST & nDIR & (BSi >= BS04) & (BSi <= BS10);
!nIOEND := (BSi == BS05) & (nPROMS & nBRST & !nDIR) "No proms, early IOEND.
        # (BSi == BS02) & (!nPROMS & nBRST & !nDIR) "Proms, late IOEND.
        # (BSi == BS03) & (nDIR);        "Write, asure data hold.

test_vectors
"READ, not PROM
(     [C,RST,STRT,DIR,PROMS,nSCSIS] → [nIOTF,IORD,IOWR,IOEND,BSi])
"01"  [c, 0, x, x, x, 1] → [1, 1, 1, 1, 15];
"02"  [c, 1, 1, 1, 1, 1] → [1, 1, 1, 1, 15];
"03"  [c, 1, 0, 0, 1, 1] → [0, 1, 1, 1, 12];
"04"  [c, 1, 0, 0, 1, 1] → [0, 1, 1, 1, 11];
"05"  [c, 1, 0, 0, 1, 1] → [0, 1, 1, 1, 10];
"06"  [c, 1, 0, 0, 1, 1] → [0, 0, 1, 1, 09];
"07"  [c, 1, 0, 0, 1, 1] → [0, 0, 1, 1, 08];
"08"  [c, 1, 0, 0, 1, 1] → [0, 0, 1, 1, 07];
"09"  [c, 1, 0, 0, 1, 1] → [0, 0, 1, 1, 06];
"10"  [c, 1, 0, 0, 1, 1] → [0, 0, 1, 1, 05];
"11"  [c, 1, 0, 0, 1, 1] → [0, 0, 1, 0, 04];
"13"  [c, 1, 0, 0, 1, 1] → [0, 0, 1, 1, 03];
"14"  [c, 1, 1, 0, 1, 1] → [0, 1, 1, 1, 02];
"15"  [c, 1, 1, 0, 1, 1] → [0, 1, 1, 1, 01];
"16"  [c, 1, 1, 0, 1, 1] → [0, 1, 1, 1, 00];
"17"  [c, 1, 1, 0, 1, 1] → [1, 1, 1, 1, 15];
test_vectors "READ, PROM
(     [C,RST,STRT,DIR,PROMS,nSCSIS] → [nIOTF,IOEND,BSi])
"18"  [c, 1, 1, 1, 1, 1] → [1, 1, 15];
"19"  [c, 1, 1, 1, 1, 1] → [1, 1, 15];
"20"  [c, 1, 0, 0, 1, 1] → [0, 1, 12];
"21"  [c, 1, 0, 0, 0, 1] → [0, 1, 11];
"22"  [c, 1, 1, 0, 0, 1] → [0, 1, 06];
"23"  [c, 1, 1, 0, 0, 1] → [0, 1, 05];
"24"  [c, 1, 1, 0, 0, 1] → [0, 1, 04];
"25"  [c, 1, 1, 0, 0, 1] → [0, 1, 03];
"26"  [c, 1, 1, 0, 0, 1] → [0, 1, 02];
"27"  [c, 1, 1, 0, 0, 1] → [0, 0, 01];
"28"  [c, 1, 1, 0, 0, 1] → [0, 1, 00];
"29"  [c, 1, 1, 0, 0, 1] → [1, 1, 15];

end iosmg
```

```
module ioadec
flag '-r3', '-t2', '-s', '-q2'
title 'IO address decoder       ioadec        izik nashelsky'
"Declarations
  IOADEC        device 'P16L8';
"inputs:
  A23,A8,A7,A6     pin 1,2,3,4;
  nIOTF, nDIR         pin 8,11;
  nBRST,nIOWR      pin 9,7;
"outputs:
  nPROMS,nICUS     pin 12,16; "PROM & ICU Selects.
  nDUARTS          pin 17;   "SIO Select.
  nDPS             pin 15; "Dip Switch Select.
  nREGRS,nREGWS    pin 13,19; "Board Cont Reg RD & WR.
  nMBWS            pin 18; "mail box write.
  nMBRS            pin 14; "mail box read.
x = .x.;
TF,D,R,WR = nIOTF,nDIR,nBRST,nIOWR;
ADDR = [A23,x,x,x,x,x,x,x,x,x,x,x,x,x,A8,A7,A6,x,x,x,x,x,x];

"***********************************************
"*  Note: Double-Mapping between m.s.b. D0-D7 (BW=8)
"*        and m.s.b. D8-DF (BW=16) according to A27
"***********************************************
Equations
"========
!nPROMS = !nIOTF & !nDIR & (ADDR >= ^h000000) & (ADDR <= ^h7FFFFF:);
!nICUS = !nIOTF & (ADDR == ^h800E00);
!nMBWS = !nIOTF & (ADDR == ^h800E80) & !nIOWR;
!nMBRS = !nIOTF & (ADDR == ^h800E80) & !nDIR;
!nDPS = !nIOTF & (ADDR == ^h800EC0) & !nDIR;
!nDUARTS = !nIOTF & (ADDR == ^h800F80);
!nREGRS = !nIOTF & (ADDR == ^h800FC0) & !nDIR;
!nREGWS = !nIOTF & !nIOWR & (ADDR == ^h800FC0) # !nBRST;
test_vectors(
  [ADDR,TF,D,R,WR] -> [nPROMS,nICUS,nDUARTS,nREGRS,nREGWS,nDPS,nMBWS])
"lRST" [^h0000000, 1, x, 1, x] -> [1, 1, 1, 1, 1, 1, 1];
"ROM" [^h0123F12, 0, 0, 1, 1] -> [0, 1, 1, 1, 1, 1, 1];
"ICU" [^h0800E00, 0, x, 1, x] -> [1, 0, 1, 1, 1, 1, 1];
"SIO" [^h0800F80, 0, x, 1, x] -> [1, 1, 0, 1, 1, 1, 1];
"RGR" [^h0800FC0, 0, 0, 1, 1] -> [1, 1, 1, 0, 1, 1, 1];
"RGW" [^h0800FC0, 0, 1, 1, 0] -> [1, 1, 1, 1, 0, 1, 1];
"DPS" [^h0800EC0, 0, 0, 1, 1] -> [1, 1, 1, 1, 1, 0, 1];
"MBW" [^h0800E80, 0, 1, 1, 0] -> [1, 1, 1, 1, 1, 1, 0];

end ioadec
```

APPENDIX B—SCHEMATIC
INTERFACE TO THE NS32GX32/NS32532





Lit. # 100592

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
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical
Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

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 Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998