Interfacing the NS32GX32 to Dynamic Memory

National Semiconductor
Application Note 629
Jim Tavakoli
June 1989

1.0 INTRODUCTION

As embedded system designs become more complex, the need intensifies for large memory subsystems that can provide high performance without contributing significantly to system cost. Dynamic memory is a relatively inexpensive solution that provides the ability to address large amounts of memory in a way that minimizes both system cost and board space.

This application note describes a dynamic random access memory (DRAM) interface to the NS32GX32 Embedded System Processor that has been optimized for both speed and design simplicity. Using 80 ns memories, this design implements burst mode with two wait states during normal cycles and one wait state during burst access. Because of the NS32GX32's general insensitivity to wait states, this two-wait state implementation allows the NS32GX32 to function at approximately 90% of its peak performance. This design can include up to two additional wait states, allowing the use of even slower memories (120 ns at 30 MHz).

This memory interface, which operates at frequencies of up to 30 MHz, includes a discrete DRAM control design and a single 32-bit DRAM bank. Either fast page mode or nibble mode DRAMs can be used.

2.0 OVERVIEW OF THE NS32GX32

The NS32GX32 is a highly integrated embedded system processor that effectively combines the most advanced features of today's microprocessors onto a single device. The NS32GX32 communicates with its environment through its parallel busses and control signals.

Among its attractive features are separate 32-bit data and address busses allowing up to 4-Gbytes of linear address space, on-chip data and instruction caches, 4-stage instruction pipeline for high performance, dynamic bus sizing for lower cost systems, and burst mode memory access.

The NS32GX32 is able to provide high performance even with slower, low cost memory because of the on-chip caches. By taking advantage of the on-chip caches' high hit-ratio, the NS32GX32's performance impact is only 4%–8% per wait state.

On each memory access, the processor initiates a memory access cycle while concurrently searching the internal caches. This reduces access time, since the memory cycle is already in process when the caches do not contain the needed information. During a read that fails to find the data in the caches (a cache miss), the memory cycle continues and the processor fetches the data from external memory.

Unless the external data is declared to be non-cacheable, the information is placed in the internal instruction or data cache for future reference. Conversely, when the instruction or data cache contains the sought information (a cache hit), the processor cancels the external memory access cycle. The NS32GX32 performs an external memory fetch in two clock cycles (assuming normal access with no wait states) and an on-chip fetch in one clock cycle.

2.1 The NS32GX32 Dynamic Memory Interface Control Signals

The following control signals are provided by the processor to simplify the DRAM interface. Refer to the NS32GX32 data sheet for further details.

/BE0–3— **Byte Enable 0–3 (Output).** These byte enable signals indicate which bytes in a double-word should be selected for transfer. During write cycles, the NS32GX32 sets BE0–3 to write to the appropriate memory bank. In the case of cacheable reads, the processor reads all bytes regardless of which byte enable signals are asserted.

/DDIN— **Data Direction (Output).** This signal connects directly to the DRAMs WE (Write Enable) pin to perform a write operation.

/BIN— **Burst In (Input).** When asserted, this input signal informs the processor that the external memory is capable of performing burst cycles.

/BOUT— **Burst Out (Output).** This signal is asserted by the NS32GX32 to initiate a burst cycle. This signal cannot be used to generate /BIN.

BCLK,/BCLK— **Bus Clock (Output).** Used to synchronize external logic and provide a processor driven output clock.

/RDY— **Ready (Input).** Used for extending current bus cycle to support slow memory and peripherals.

/CONF— **Confirm (Output).** When active, indicates that a bus cycle initiated by /ADS is valid; that is, the bus cycle has not been cancelled.

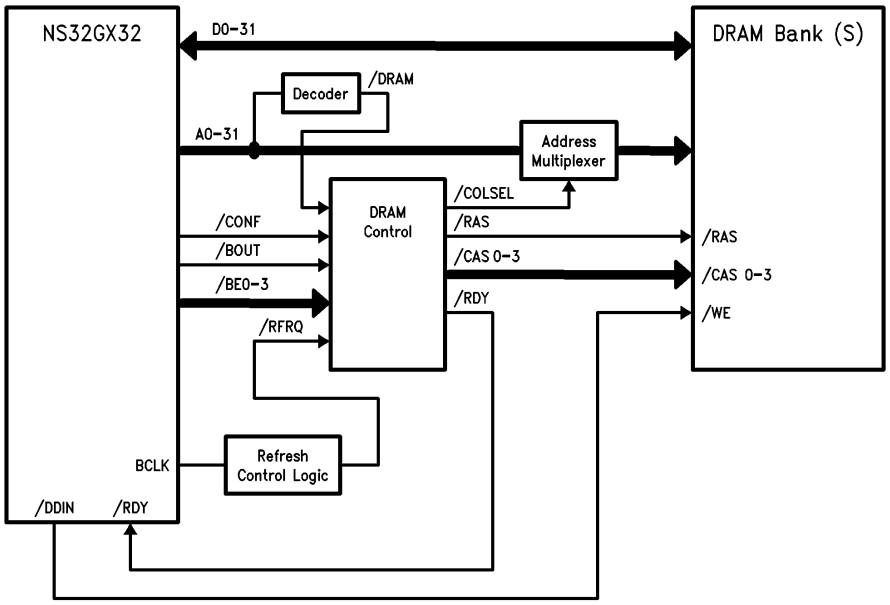
2.2 NS32GX32 DRAM Interface Overview

The DRAM interface described in this note takes advantage of many features of the NS32GX32 including Burst Mode Memory Addressing and the large address space (Refer to *Figure 1*). In performing burst cycles, the NS32GX32 allows multiple accesses by providing up to 4 sequential addresses per burst cycle.

A standard PAL device (NS PAL16R8) functions as a DRAM controller by generating /RAS (Row Address Strobe) and /CAS (Column Address Strobe) for both normal and fast access. /RFRQ (Refresh Request) is provided by external counters to request the refresh cycle from the PAL. The PAL provides pertinent signals for the /CAS before /RAS refresh technique transparently to the processor.

 TL/EE10455

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



TL/EE/10455–1

FIGURE 1. The interface consists of four functional blocks including a discrete DRAM controller circuitry and refresh counters.

The processor provides /DDIN to indicate the direction of data transfer, BE0–3 to indicate byte, word or double word access and /BOUT to request burst cycle.

The DRAM bank is enabled by the decoder (/DRAM). A D-type PAL16L8 is used to decode memory as well as I/O. For more details refer to Appendix B. To enhance the performance of conventional DRAMs, an early /RAS generation technique has been implemented. A simple multiplexing circuit selects between row address and column address. Fast page mode or nibble mode DRAMs with 80 ns access times are used to support processor burst cycles at 30 MHz.

2.3 Interface Features

This DRAM interface design provides the flexibility to meet various application requirements. The interface allows the NS32GX32 (running at speeds of 20 MHz, 25 MHz and 30 MHz) to communicate with DRAMs of various sizes (1–4 Mbytes) and speeds (Refer to Table I). Section 4 provides details on timing parameters and worst case calculations. The number of wait states on the various speeds of memories can be controlled through use of programmable logic. This permits customization of system timing sequences.

TABLE I. Wait States on Normal and Burst Accesses

Processor Speed	DRAM Access Time	Wait States on Normal/Burst Access
30 MHz	80, 100, 120 ns	2/1, 3/2, 4/2
25 MHz	100, 120, 150	2/1, 3/2, 4/2
20 MHz	120, 150, 200	2/1, 3/1, 3/2

The PAL state machine diagram shown in *Figure 2* implements up to two additional wait states. These can be inserted by means of jumpers at the input of the DRAM Control PAL (WSEL1 and WSEL2). Refer to Table II for details. Since DRAMs have various constraints on /RAS precharge time, an optional time delay to compensate for this is implemented through the jumper /TDSL.

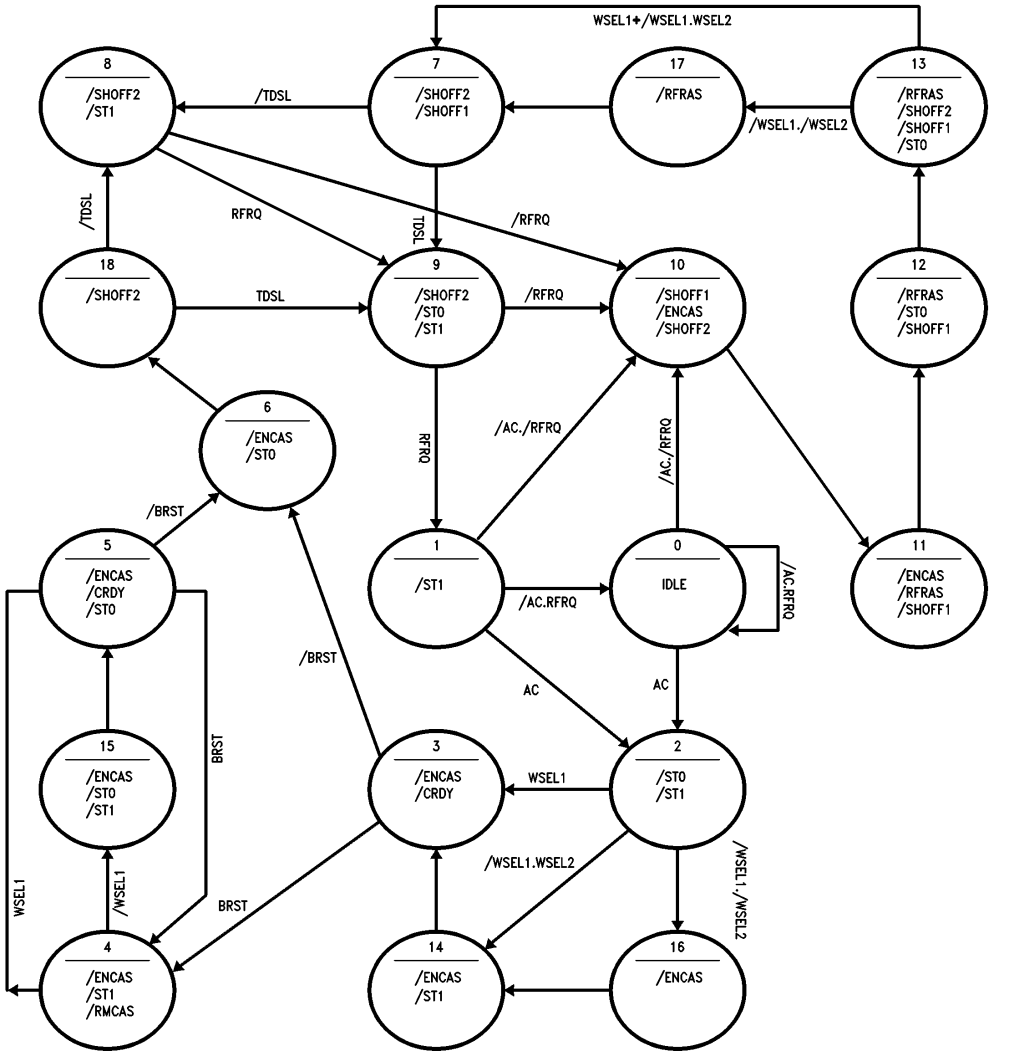
TABLE II. User Selectable Wait States

Cycle Type	WSEL1	/WSEL1. WSEL2	/WSEL1. /WSEL2
Normal Access	2	3	4
Burst Access	1	2	2
Refresh Cycle	3	3	4

WSEL1 and WSEL2 determine the number of additional wait states inserted.

Note: WSEL = Jumper Not Connected
/WSEL = Jumper Connected

This interface can take advantage of either Fast Page Mode DRAMs or Nibble Mode DRAMs. During a Fast Page access, the DRAMs receive up to 4 sequential addresses and provide up to 16 bytes of data. The DRAMs are organized in a single 32-bit bank, allowing /CAS before /RAS refresh cycles. The refresh rate is dependent on the processor clock frequency and is set by means of jumpers located at input of refresh cycle counter. The memory can be easily expanded to allow for more memory banks. In this case, staggered refresh cycles should be performed to reduce refresh rate.



TL/EE/10455–2

FIGURE 2. State Machine Diagram for the NS3GX32 DRAM Interface. For more details on state equations refer to Appendix A

3.0 INTERFACE LOGIC

As shown in *Figure 1*, the interface consists of four functional blocks, processor, dynamic memory, refresh logic, and DRAM control. These are discussed in the following section.

3.1 The NS32GX32 Interface

3.1.1 Burst Cycle

A burst cycle is performed to rapidly transfer instructions or data from 32-bit wide memories. A burst is composed of two parts: Normal Access (two clock cycles for the first doubleword of data) and Burst Access (one clock cycle for subsequent data). During a burst access, the processor's Bus Interface Unit forces the two least significant bits of the address lines (A0, A1) to zero and increments address bits A2, A3 to select up to 3 double words subsequent to the first address. All byte enable (BE0−3) signals are enabled during a burst mode transfer.

When using nibble mode DRAMs, the internal registers of the DRAMs will take the incoming address on the bus and increment it to provide data from up to 4 addresses. During this time, additional addresses generated by the processor are ignored. When fast page mode DRAMs are used, the processor is responsible for providing all addresses to the dynamic memory. The interface discussed here allows for both options.

The /CONF signal is used by the processor to initiate an external memory read. This signal indicates that the processor was unable to find the needed information in the caches, and therefore, it must continue with an external memory fetch. An early /RAS generation technique is implemented by means of immediate qualification of /CONF and /DRAM (DRAM chip select).

Because the processor requires two clock cycles for the initial access during burst mode, and only one clock cycle for burst accesses, /CRDY (Custom Ready) is generated from the DRAM Control PAL to be used as the processor's /RDY signal.

3.1.2 DRAM Control

The DRAM control logic is implemented by using a PAL (NS PAL16R8) device. The state machine is shown in *Figure 2*, and the PAL equations are found in Appendix A.

The PAL takes care of generating /RAS, /CAS, as well as /RFRAS (begin refresh cycle), /SHOFF1, 2 (access inhibit to distinguish between refresh cycle and memory cycle) and /CRDY to allow for slow memory. Timing delays and the number of wait states are set by the user through the jumpers WSEL1, WSEL2, and TDSL. Table II shows how WSEL1 and WSEL2 insert wait states to alter the normal execution of a bus cycle.

3.1.3 DRAM Bank

The user has a choice of three DRAM configurations as shown in Table III.

TABLE III. Memory Configurations Allowed Based on the Required Memory Size

Memory Size	Configuration
1 Mbyte	4 x 256-Kbyte
2 Mbyte	4 x 512-Kbyte
4 Mbyte	4 x 1-Mbyte

As shown in sheet 3 of the schematic, four /CAS signals are generated which are qualified with byte enable signals. Since the interface is based on a single bank of memory, only one /RAS is generated and buffered to provide adequate drive for the modules. Write Enable signals for each of the DRAM chips are easily generated by using the processor /DDIN and /BE0−3 signals.

In order to multiplex address lines for the DRAMs, a simple multiplexing circuit is implemented which selects row address at the beginning of a cycle, and then provides four incrementing column addresses for the rest of the cycle.

Although the NS32GX32 increments address lines A2 and A3 during a burst mode transfer, an external 2-bit counter is used to increment these signals. This implementation of advance address calculation allows minimization of wait states during burst accesses when fast page mode DRAMs are used.

3.1.4 Refresh Logic

Because of the internal partitioning of DRAMs, only one half the rows of DRAM memories need to be refreshed. For example, a 1 Mbit x 1 dynamic memory requires 512 rows to be refreshed every 8 ms (or each row every 15.6 μs).

The refresh logic consists of two cascaded 4-bit counters with pre-settable inputs provided at the higher order counter to allow variable refresh rates. The interface uses a ''/CAS before /RAS'' refresh technique to minimize processor bus utilization. The 16-bit address counter is implemented by counters and flip-flops to provide 512 refresh addresses and increments at the end of each refresh cycle. The /RFRQ (refresh request generated by the counters) is received by the PAL which in turn generates the required /CAS and /RAS signal.

4.0 TIMING CALCULATIONS

The following timing calculations are done for a 30 MHz processor and 80 ns DRAMs. The timing diagram shown in *Figure 3* shows a normal DRAM read-access cycle. The cycle begins with the assertion of the /RAS signal after the external memory access has been confirmed (/CONF asserted). Maximum time to /CONF low (t_{CONF}) is 8 ns after the falling edge of t1.

DRAM timing parameters for an 80 ns device:

t_{RAC} = Access Time from /RAS
t_{CAC} = Access Time from /CAS
t_{RCD} = /RAS to /CAS Delay Time
t_{AA} = Access Time from Column Address
t_{CP} = /CAS Precharge Time
t_{RASA} = R.E. T1 to /RAS Asserted
t_{CASA} = R.E. T1 to /CAS Asserted

$$t_{RASA} = t_{CONF} + T_d\ (AS27 + AS02)$$
$$= (16.5 + 8) + (5.5 + 4.5)$$
$$= 25 + 10 = 35\ ns$$

Therefore t_{RASA} is asserted as early as the rising edge of T2:

At this time, the address provided by the processor is received by the DRAMs and the multiplexing circuitry is clocked to provide column addresses for the DRAMs. The minimum address setup time before /CAS enable is zero. This allows the programmable logic to provide /ENCAS with

the rising edge of the next clock. /ENCAS is gated with /BE0–3 to provide the appropriate /CAS0–3 signals for byte, word or double-word access.

t_{CASA} = t_{ENCAS} + t_d (AS04 + AS10)
= 67 + t_d (PAL) + Td (AS04 + AS10)
= 84 ns

Thus

t_{RCD} = t_{CASA} − t_{RASA}
= 84 − 35
= 49 ns

If t_{RCD} is met, the DRAM access time is ensured. If it does not meet the DRAM specifications, the access time is dependent on t_{CAC}.

/RAS and /CAS are concurrently asserted for two clock cycles so that data in the DRAMs is accessed early enough to meet the processor data setup time.

t_{BCP} = Bus Clock Period
t_{DIS} = Data Setup Time
N = Number of Wait States

The number of wait states required for accessing slower DRAMs can be calculated from the following equations:

To determine t_{RAC} during a normal access:

t_{RAC} ≤ t_{BCP} × (N+2) − (t_{CONF}) − t_d(AS02 + AS27) − t_{DIS} − (t_d (AS245)
≤ 33 × (N+2) − 24 − 10 − 9 − 7

Assuming two wait states (N = 2), t_{RAC} ≤ 83 ns.

To determine t_{CAC} during a normal access:

t_{CAC} ≤ t_{BCP}× (N+2) − (t_{ENCAS}) − t_{DIS} − t_d (AS04 + AS10) − t_d(AS245)
≤ 33 × (N+2) − 75 − 9 − 9 − 7

Assuming two wait states (N = 2), t_{CAC} ≤ 33 ns.

During a normal read or write, /RMCAS (Remove CAS) is always inactive. However, this signal plays an important role during burst cycles as shown in *Figure 4*. During a burst cycle, /CAS must be brought high in order to strobe in the new column address. Furthermore, the /CAS precharge time must be met before /CAS may go low again. This is the main reason why /RMCAS and /ENCAS are used together to toggle /CAS during burst cycles. Once the cycle is completed, the state machine provides /RAS precharge time and initializes the DRAMs for another access or a refresh cycle. *Figure 4* shows the sequence of signals during a burst cycle.

To determine the worst case margins during a burst access, two other parameters should be analyzed. The first calculation relates to the TAA (access time from column address) of fast access DRAM. Using an advance address calculation implementation:

t_{ABv} = Burst Address Valid
= t_{CRDY} + t_{BCP} + t_{AS08} + t_{AS04} + t_{AS374} + t_{163} + t_{AS258} + t_{AS1034}
= 75 + 16 + 4.5 + 4.5 + 8 + 11 + 7 + 5
= 3 ns before rising edge of T_B

Therefore

t_{AA} < 3 + t_{BCP} × (N+1) − t_{AS245} − t_{DIS}
< 54 ns

During burst access, the /CAS signal is removed as soon as possible to allow for /CAS precharge time. The following equation may be used to calculate the worst /CAS margin for t_{CP} (/CAS precharge time) during burst access.

t_{CP} < t_{BCL}

It is important to make certain the timing parameters of DRAMs present values within the range shown.

5.0 SUMMARY

Embedded system applications such as high resolution graphics, page oriented printing, and data communications place growing demands on embedded processor throughput. The overall throughput of embedded systems often depends on the performance of the memory subsystem. Because of cost constraints, these subsystems cannot be composed of expensive high speed Static RAMs.

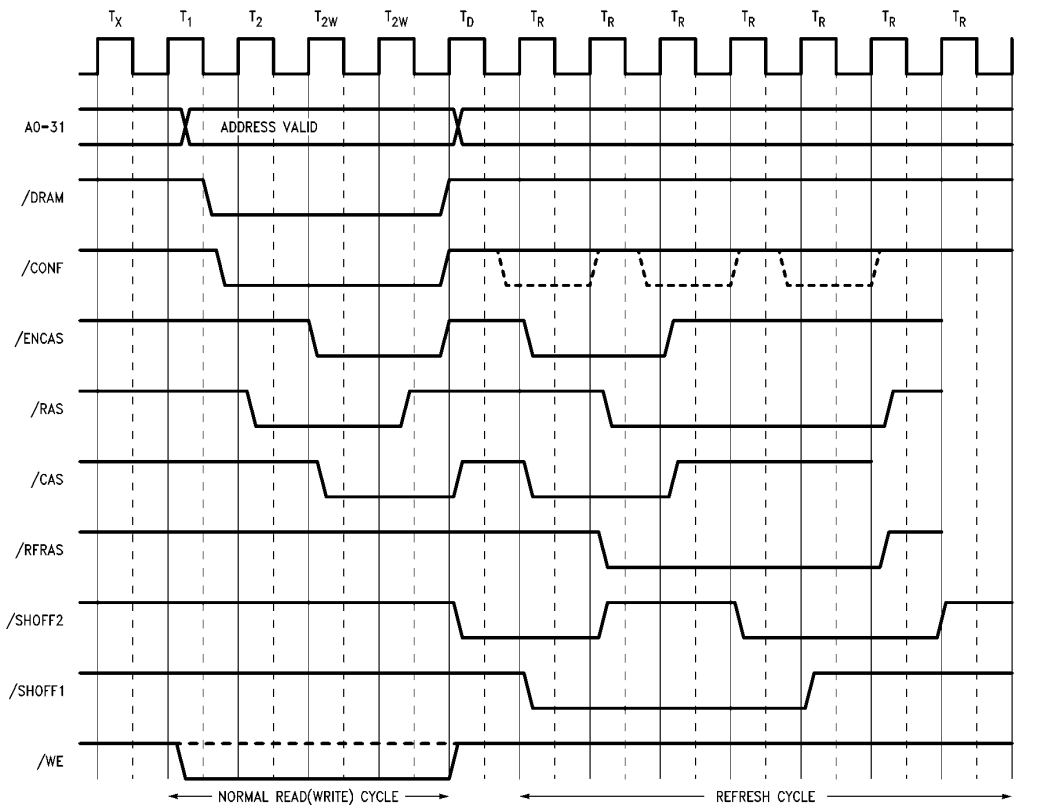
The design discussed in this application note is a high performance design which at the same time allows for lower system cost. It uses 80 ns DRAMs to support processor burst cycles at 30 MHz. It achieves normal memory cycles in four clock cycles (two clocks for normal access and two wait states) and it completes a burst mode transfer of 4 double-words in 10 clock cycles. The programmable DRAM control PAL can be implemented to vary the number of wait states and other timing features to suit various application needs.

REFERENCES

NS32GX32 High-Performance 32-Bit Embedded System Processor Data Sheet

Advanced Peripherals:

DRAM Management Handbook



TL/EE/10455–3

FIGURE 3. /SHOFF1 and /SHOFF2 are Used to Distinguish between a Normal Cycle and a Refresh Cycle

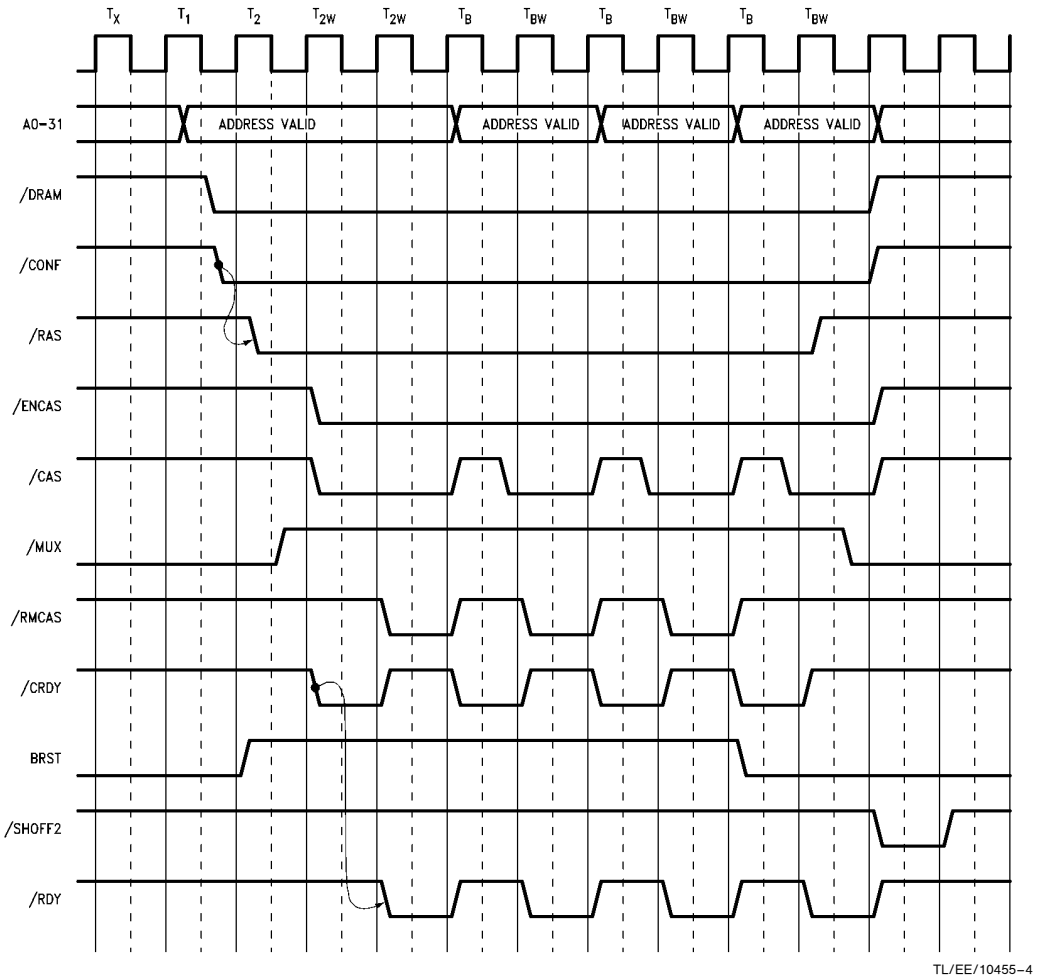
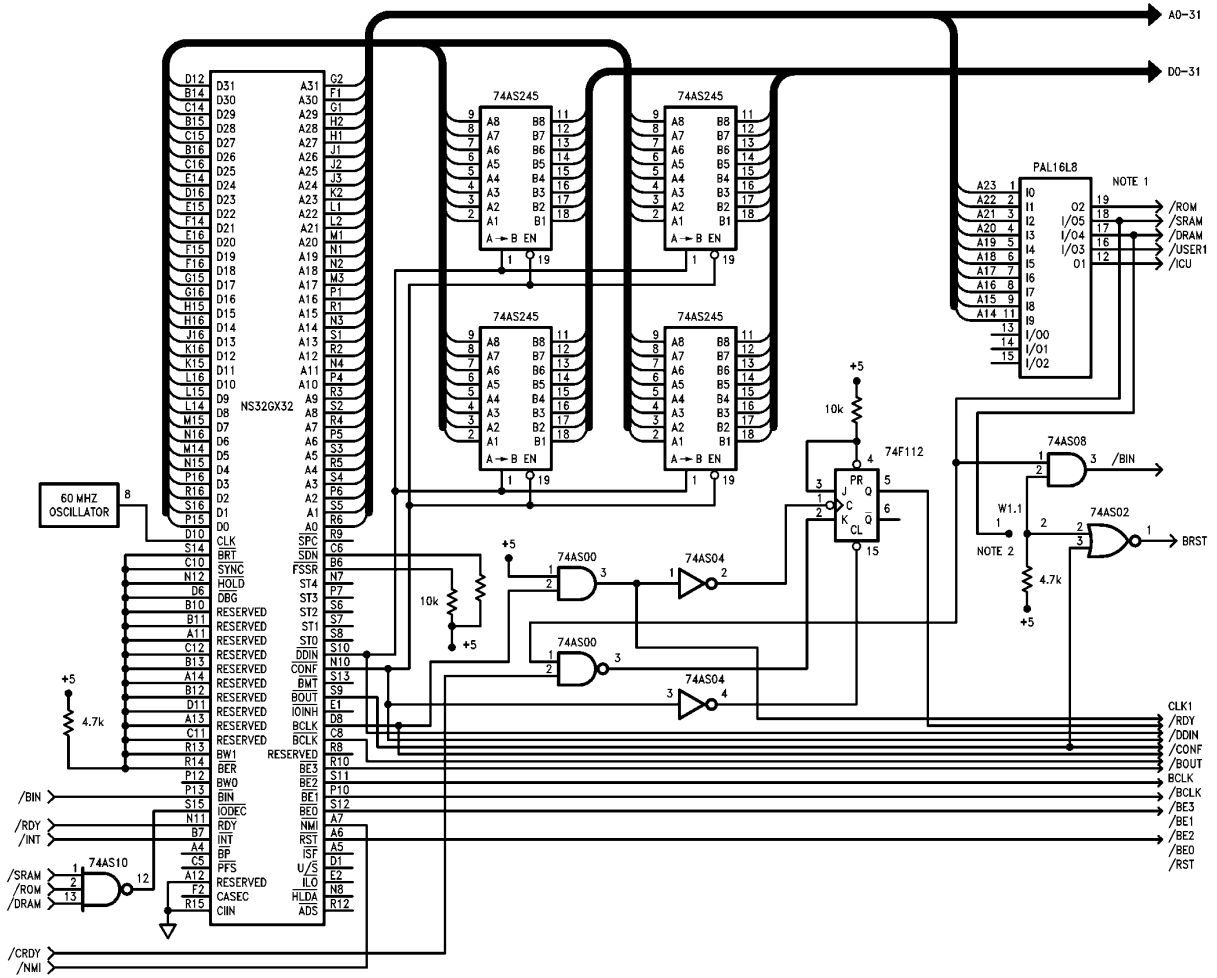


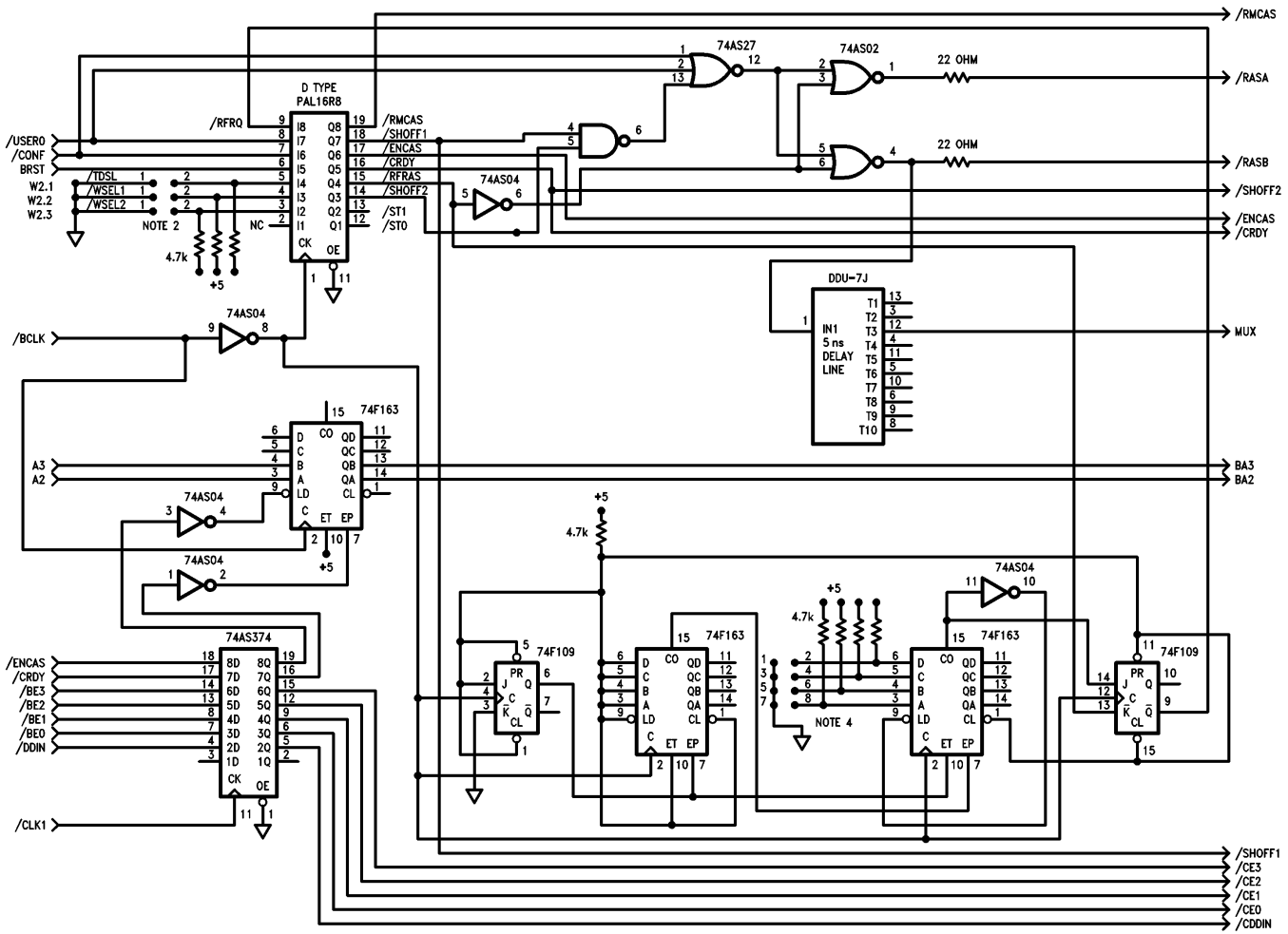
FIGURE 4. In a burst access cycle, the processor is capable of fetching up to 16 bytes of data from Fast Page Mode or Nibble Mode DRAMs



Note 1: DRAM space is implemented as /USER0

Note 2: Jumper W1.1 installed enables burst

NS32GX32 DRAM Interface

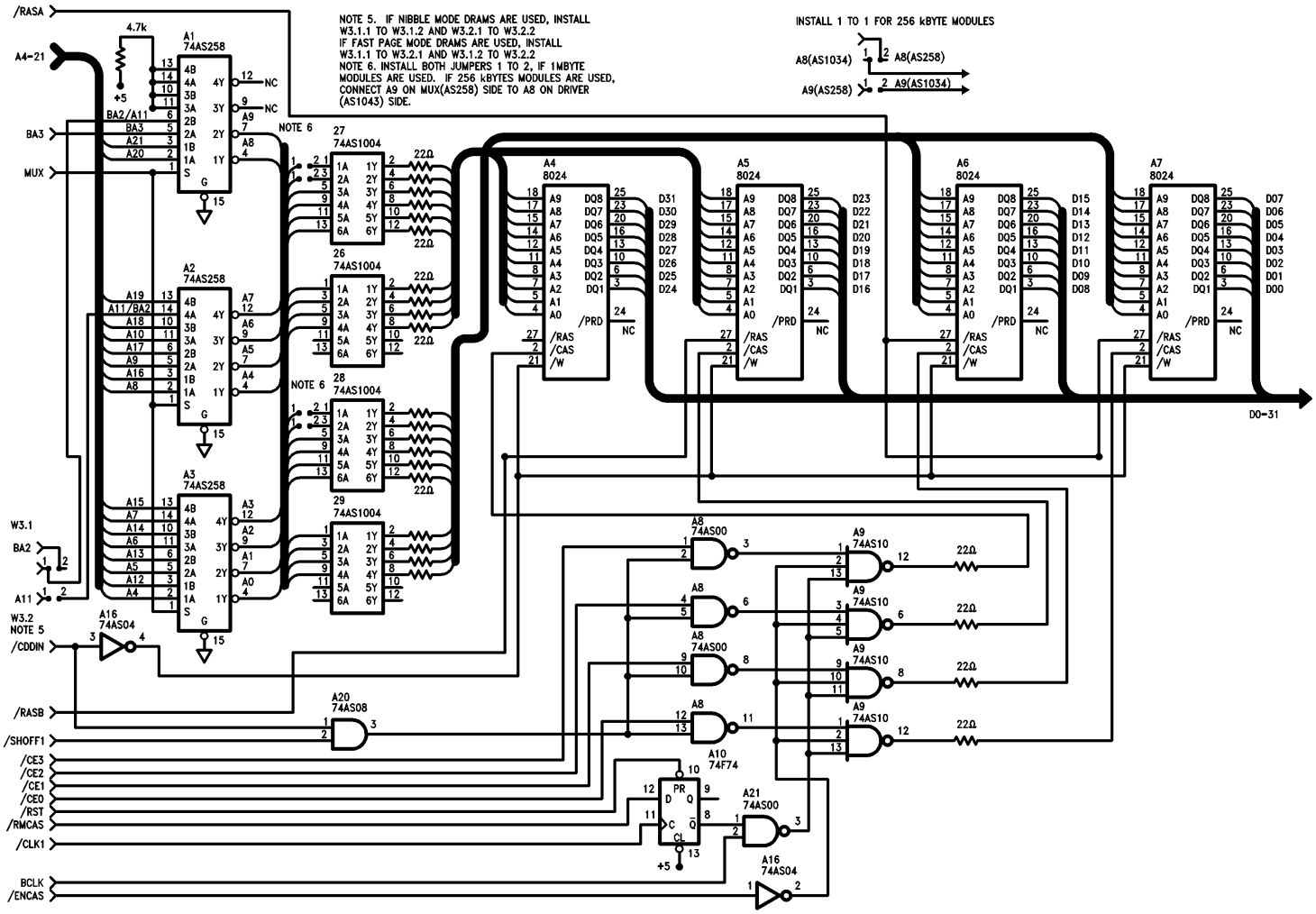


TL/EE/10455–6

Note 3: Jumper 2.1 allows for /RAS precharge compensation. Jumper 2.2 allows for one additional wait state. Jumper 2.3 allows for two additional wait states.

Note 4: Jumper 2.4 determines refresh rate depending on clock frequency.

NS32GX32 DRAM Interface (Continued)



NS32GX32 DRAM Interface (Continued)

APPENDIX A. STATE MACHINE EQUATIONS

```
               PARTNO    520162;
               NAME      STATE_MACHINE;
               DATE      02/10/89;
               REV        A;
               DESIGNER  JIM TAVAKOLI;
               COMPANY   NATIONAL SEMICONDUCTOR CORP.;
               ASSEMBLY  NS32GX32;
               LOCATION  U3.23;

/*************************************************************************/
/******STATE MACHINE DEVICE FOR NS32GX32 DRAM INTERFACE*****************/
/*************************************************************************/
/* THIS DEVICE GENERATES SIGNALS REQUIRED FOR CONTROLLING A              */
/* SINGLE 32-BIT BANK OF DYNAMIC RAMS. THE DEVICE ALLOWS                 */
/* DRAM CONTROLLER IMPLEMENTATION WITH WAIT SELECT OPTIONS               */
/*************************************************************************/
/* Allowable Target Device Types: PAL16R8                                */
/*************************************************************************/
/**    Inputs   **/

PIN    1       =BCLK     ;       /*  GX32/ BUS CLOCK                    */
PIN    3       =!WSEL2   ;       /*  ADDS TWO ADDITIONAL WAITS          */
PIN    4       =!WSEL1   ;       /*  ADDS ONE ADDITIONAL WAIT           */
PIN    5       =!TDSL    ;       /*  CREATES CYCLE DELAY                */
PIN    6       =BRST     ;       /*  BURST ENABLE                       */
PIN    7       =!CONF    ;       /*  MEMORY CYCLE CONFIRMATION          */
PIN    8       =!USER0   ;       /*  DRAM ADDRESS CHIP SELECT           */
PIN    9       =!RERQ    ;       /*  REFRESH REQUEST                    */
PIN    11      =!OE      ;       /*  DEVICE OUTPUT ENABLE               */

/**    Outputs   **/

PIN    12      =!ST0     ;       /*  DISTINGUISHING STATES BETWEEN      */
PIN    13      =!ST1     ;       /*  REFRESH RAS AND ACCESS RAS         */
PIN    14      =!SHOFF2  ;       /*  ACCESS SHUT-OFF DUE TO DELAY       */
PIN    15      =!RFRAS   ;       /*  RAS ENABLE FOR REFRESH CYCLE       */
PIN    16      =!CRDY    ;       /*  CUSTOM READY FOR SLOW ACCESSES     */
PIN    17      =!ENCAS   ;       /*  CAS ENABLE                         */
PIN    18      =!SHOFF1  ;       /*  ACCESS SHUT-OFF DUE TO REFRESH     */
PIN    19      =!RMCAS   ;       /*  CAS REMOVAL FOR FAST ACCESS        */

/*     STATE 0 (/ST0) AND STATE 1 (ST1) ARE USED TO DISTINGUISH         */
/*     BETWEEN INTERNAL OPERATIONS OF THE DEVICE. THESE TWO SIGNALS     */
/*     ARE NOT IMPLEMENTED AS OUTPUT SIGNALS.                           */
```

APPENDIX A. STATE MACHINE EQUATIONS (Continued)

```
/**    Logic Equations   **/
S0     =  !RMCAS & !SHOFF1 & !ENCAS & !CRDY & !RFRAS & !SHOFF2 & !ST1 & !ST0;
S1     =  !RMCAS & !SHOFF1 & !ENCAS & !CRDY & !RFRAS & !SHOFF2 &  ST1 & !ST0;
S2     =  !RMCAS & !SHOFF1 & !ENCAS & !CRDY & !RFRAS & !SHOFF2 &  ST1 &  ST0;
S3     =  !RMCAS & !SHOFF1 &  ENCAS &  CRDY & !RFRAS & !SHOFF2 & !ST1 & !ST0;
S4     =   RMCAS & !SHOFF1 &  ENCAS & !CRDY & !RFRAS & !SHOFF2 &  ST1 & !ST0;
S5     =  !RMCAS & !SHOFF1 &  ENCAS &  CRDY & !RFRAS & !SHOFF2 & !ST1 &  ST0;
S6     =  !RMCAS & !SHOFF1 &  ENCAS & !CRDY & !RFRAS &  SHOFF2 & !ST1 &  ST0;
S7     =  !RMCAS &  SHOFF1 & !ENCAS & !CRDY & !RFRAS &  SHOFF2 & !ST1 & !ST0;
S8     =  !RMCAS & !SHOFF1 & !ENCAS & !CRDY & !RFRAS &  SHOFF2 &  ST1 & !ST0;
S9     =  !RMCAS & !SHOFF1 & !ENCAS & !CRDY & !RFRAS &  SHOFF2 &  ST1 &  ST0;
S10    =  !RMCAS &  SHOFF1 &  ENCAS & !CRDY & !RFRAS &  SHOFF2 & !ST1 & !ST0;
S11    =  !RMCAS &  SHOFF1 &  ENCAS & !CRDY &  RFRAS & !SHOFF2 & !ST1 & !ST0;
S12    =  !RMCAS &  SHOFF1 & !ENCAS & !CRDY &  RFRAS & !SHOFF2 & !ST1 &  ST0;
S13    =  !RMCAS &  SHOFF1 & !ENCAS & !CRDY &  RFRAS &  SHOFF2 & !ST1 &  ST0;
S14    =  !RMCAS & !SHOFF1 &  ENCAS & !CRDY & !RFRAS & !SHOFF2 &  ST1 & !ST0;
S15    =  !RMCAS & !SHOFF1 &  ENCAS & !CRDY & !RFRAS & !SHOFF2 &  ST1 &  ST0;
S16    =  !RMCAS & !SHOFF1 &  ENCAS & !CRDY & !RFRAS & !SHOFF2 & !ST1 & !ST0;
S17    =  !RMCAS & !SHOFF1 & !ENCAS & !CRDY &  RFRAS &  SHOFF2 & !ST1 & !ST0;
S18    =  !RMCAS & !SHOFF1 & !ENCAS & !CRDY & !RFRAS &  SHOFF2 & !ST1 & !ST0;

/**    STATE EQUATIONS   **/

RMCAS.D  = (S3 # S5) & BRST ;
SHOFF1.D = S11 # (S13 # S12) # (S0 # S1) & RFRQ & !USER0 # S17
         # (S8 # S9) & RFRQ # (S0 # S1) & RFRQ & !CONF # S10;
ENCAS.D  = (S14 # S4) # (S2 # S15) # (S9 # S8) & RFRQ # (S3 # S5)
         # (S0 # S1) & RFRQ & !CONF # (S0 # S1) & RFRQ & !USER0
         # (S16 # S10) ;
CRDY.D   = S14 # (S2 # S4) & !WSEL1 # S15 ;
RFRAS.D  S10 # S11 # S12 # S13 & WSEL1 & WSEL2 ;
SHOFF2.D = (S18 # S7) # (S8 # S9)&RFRQ # S8 & !RFRQ # S17 # (S12 # S13)
         # (S0 # S1)& RFRQ & !CONF # (S0 # S1) & RFRQ & !USER0 # S6 ;
ST1.D    = (S0 # S1) & USER0 & CONF # (S18 # S7) # S9 # S8 & !RFRQ
         # S2 & WSEL1 & !WSEL2 # S4 & WSEL1 # S16 # (S3 # S5) & BRST;
ST0.D    = S11 # S4 # (S0 # S1) & USER0 & CONF # (S18 # S7) & !TDSL
         # S8 & !RFRQ # S12 # S15 # (S3 # S5) & !BRST ;
```

APPENDIX B.

A typical decoding Pal used for supporting memory and I/O devices interfaced to NS32GX32

```
PAL16L8
PART NO. 52016
DECODER
NSC, SANTA CLARA, CA
A23, A22, A20, A19, A18, A17, A16, A15, GND A8 IOSEL1 OVER IOSEL0
ICU USER1 DRAM SRAM ROM VCC
/ROM = A23*A22*A21*A20*/A19*/A18 + /A23*/A22*/A21*/A20*/A19*/A18*OVER
/SRAM = /A23*/A22*/A21*/A20*/A19*/A18*OVER
/DRAM = A23*/A22
/USER1 = A23*A22*/A21
/ICU = A23*A22*A21*A20*A19*A18*A17*A16*A15*A8
/IOSEL0 = A23*A22*A21*A20*A19*A18*A17*/A16*/A15
/IOSEL1 = A23*A22*A21*A20*A19*A18*A17*/A16*/A15
```

Lit. # 100629

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