National's DP8440 DRAM Controller in a 40 MHz System

National Semiconductor
Application Note 999
Sunil Aurora, Michael Schumacher
and Dan H. Thai
July 1995

A working design of National's DP8440 memory controller interfaced to the 68040 microprocessor at 40 MHz.

1.0 INTRODUCTION

1.1 DP8440 Memory Controller

The DP8440/41 is one of National's latest memory control solutions. For the designer, it presents a reliable and easy to implement solution for most DRAM controller requirements. The DP8440/41 is a general purpose programmable device, which can operate in different applications. In this design the DP8440 controls two 72-pin SIMMs, and interfaces to Motorola's 68040 microprocessor.

1.2 Design Overview

A block diagram of the demo board is shown in *Figure 1*. The board is fabricated in a standard ISA form factor, tapping power and ground from the ISA bus. While none of the other ISA signals are used in the current design, there is provision for using them through a 100-pin header placed near the ISA edge connector.

Additionally, there are two other 100-pin expansion connectors in the design. One of these connectors provides an extension of the 68040 bus, while the other connector passes the DRAM SIMM signals through which the user can add additional SIMMs on a daughter card.

The design consists of the following blocks: DRAM system, CPU, I/O, EPROM system, and the Clock generation system.

2.0 THEORY OF OPERATION

2.1 Top Level View

A top level view of the schematic *(Figure 2)* shows the interconnections between the main blocks of the design.

The following sections explain the operation of these blocks:

2.2 DRAM System

At the core of the DRAM system is the DP8440. This handles all of the overhead required in managing the two 72-pin SIMMs in the design.

2.2.1 Memory Configurations

Two 72-pin SIMM sockets form two banks of memory on the board. These sockets can be populated with either 1, 2, 4, or 8 MEG x 36 SIMMs. It is not necessary to populate both banks if a one bank system is desired. However, if two banks are used then both banks must use the same type of SIMMs. The schematic of the DRAM system is shown in *Figure 3*.

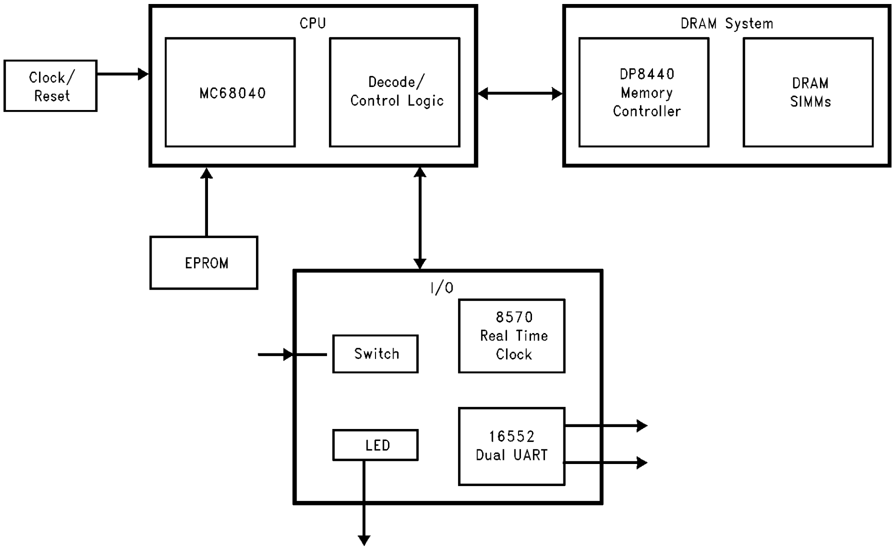
In order to utilize the various memory configurations, the DP8440 must be programmed such that the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines decode correctly, and the jumpers are set to the correct position. The jumper settings are given in Section 4.0 in Table IX, "Jumper 15 Settings: selects address input lines for the DP8440", and Table X, "Jumper 16 Settings: selects RAS line for second bank coming from 8440".

National's DP8440 DRAM Controller in a 40 MHz System

AN-999

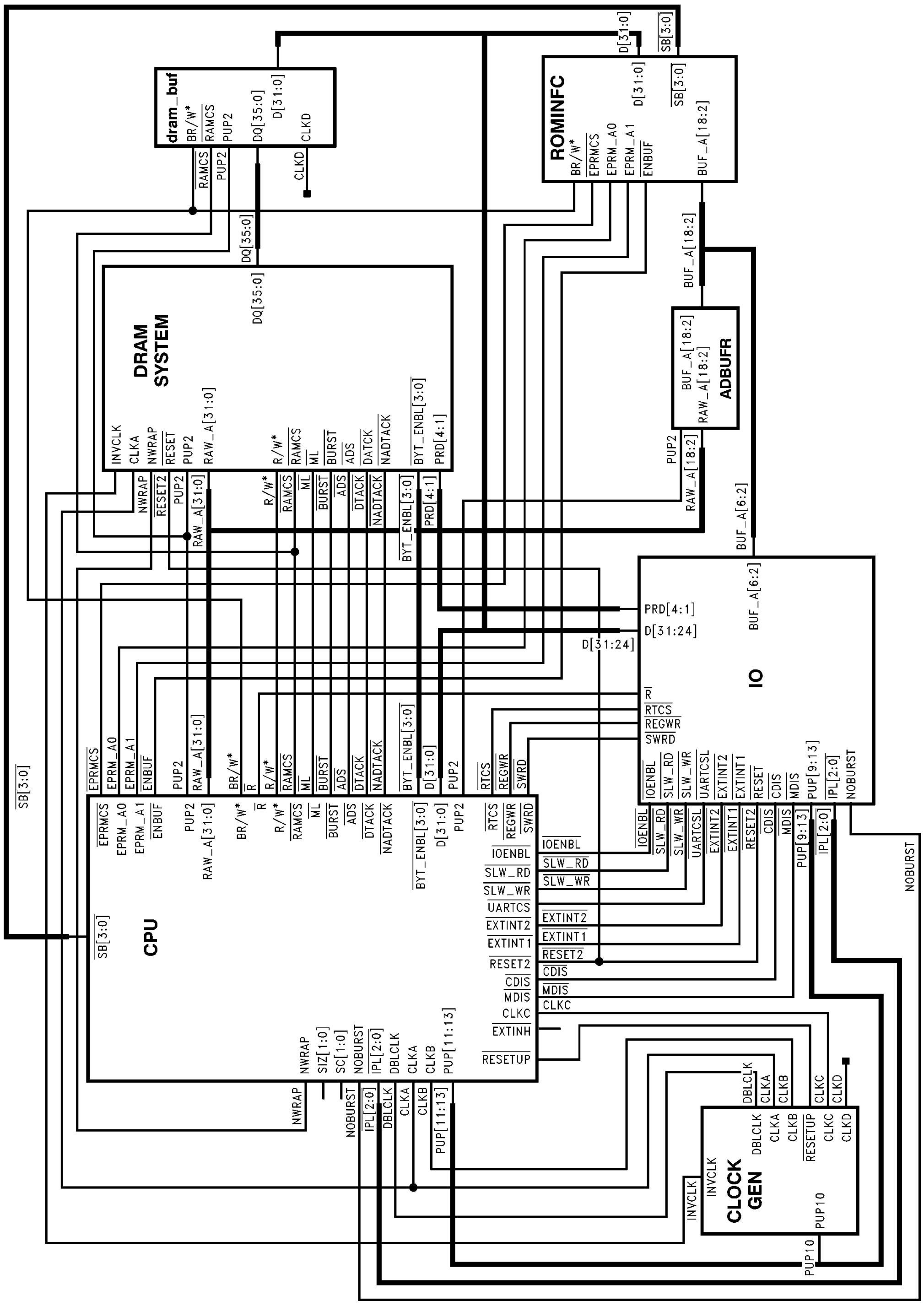
PAL® is a registered trademark of Advanced Micro Devices, Inc.
OPAL™ is a trademark of National Semiconductor Corporation.

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



TL/F/12471–1

FIGURE 1. Block Diagram



TL/F/12471–2

FIGURE 2. Top Level View

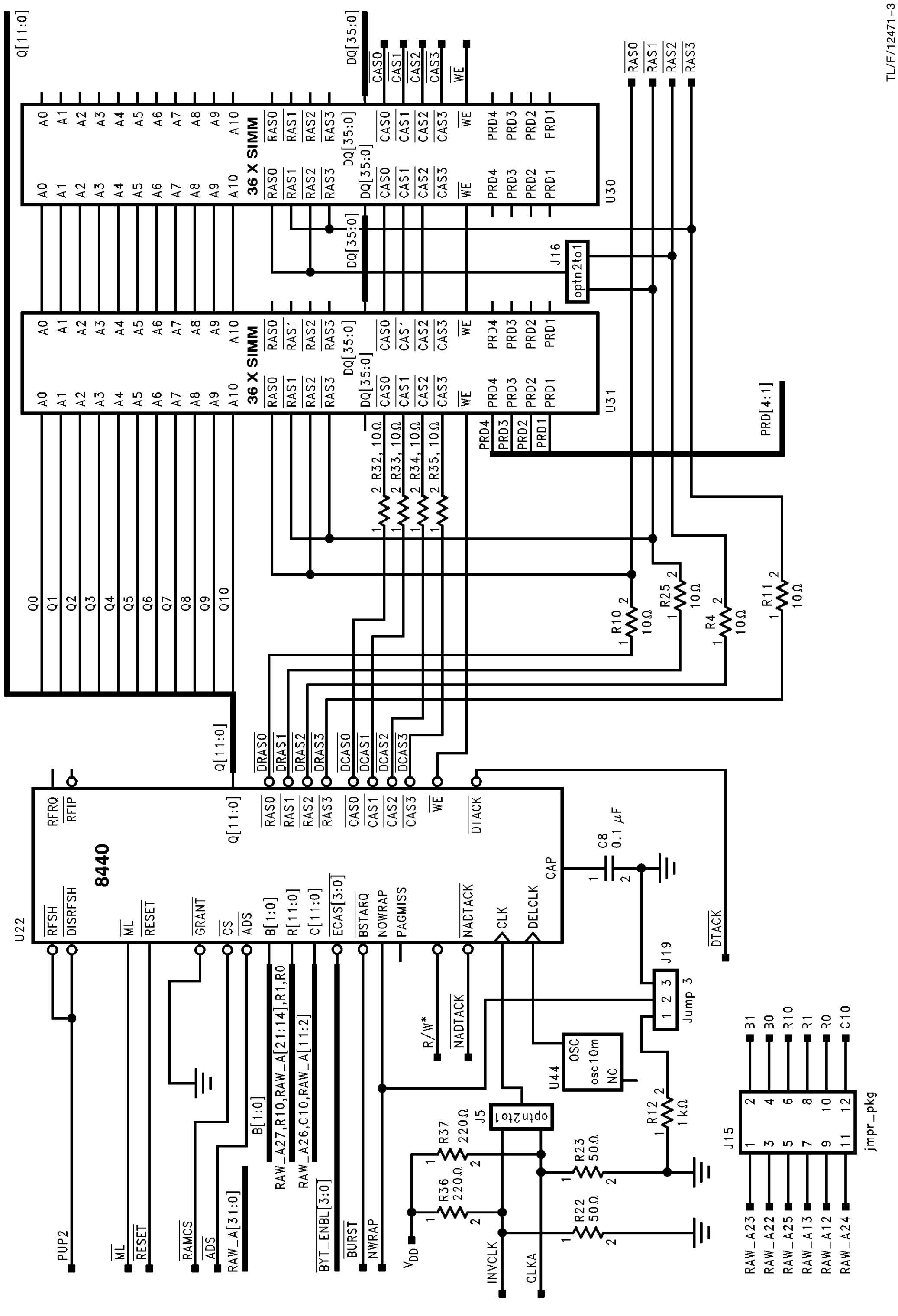


FIGURE 3. DRAM System Schematic

2.3 Programming the DP8440

The DP8440 has many different modes of operation; the desired mode must first be programmed into the DP8440 before the DRAM system becomes functional. Programming the device is accomplished by placing valid programming bits onto the programming register inputs, and then asserting Mode Load ($\overline{\text{ML}}$). This will then latch the new programming bits into the programming register and program the device. The programming register inputs are shared with the Row, Column, Bank, and $\overline{\text{ECAS}}$ inputs. Therefore, when Mode Load is asserted, these signals are used to latch data into the programming register.

Typically in a system using the DP8440, Mode Lode ($\overline{\text{ML}}$) is decoded from the high order address bits. This means there will be a ''hole'' that exists in the memory space used solely for programming the DP8440. However, this is not much of a restriction because the addressable space of the DP8440 is 256 MB, much more than is necessary in a typical system.

In this design, once the proper programming selections are determined and mapped to the corresponding address, a write is issued to this address. The address range reserved for programming the DP8440 in this design is $10000000–$1FFFFFFF.

Care should be taken when mapping the DP8440 Row, Column, and Bank inputs to the CPU address signals, as this is dependent on the setting of Jumper 15 (J15). J15 is used to select operation among the following sizes of SIMMs: 1, 2, 4 or 8 MEG x 36. See Table IX in Section 4.0 for complete jumper settings.

2.4 Clock Generation

The 68040 requires a 1 X CLK for external bus and a 2 X CLK for internal operation. A 1 X CLK is required to drive the DP8440 and state machine logic on the board. In some applications using the Page Mode feature of the DP8440 it may be advantageous to run the DP8440 on an inverted clock. This design can incorporate both an inverted and non-inverted clock by switching Jumper J5 (Table IV, Section 4.0) and changing the wait state PAL®; the PAL equations are given in Section 5.5, ''Inverse Clock Wait State Machine'', and Section 5.6, ''Positive Clock Wait State Machine''. For proper operation of the board the skew between these different clocks must be tightly controlled. To meet these clocking requirements a PLL based clock generator is needed.

The Motorola 88915PC used on this board meets these clocking requirements for operation up to 40 MHz (providing a 2 X CLK to 80 MHz). The multiple CLK outputs are distributed to balance loading. Care must be taken to follow the layout recommendations for this part to guarantee proper operation of its internal PLL.

An option to run the DP8440 on an inverted CLK is provided via Jumper J5. Running the DP8440 on an inverted CLK can be useful when using the Page mode feature. The extra time, equal to half a clock period, provides sufficient set up for the comparator without having to delay the start of the DP8440 by another clock cycle. Limiting the start up latency in this manner can improve memory system response during page misses.

At these high speeds proper termination of clocks is necessary and an option is provided for series or thevenin's termination on the board.

2.5 Buffer Selection

Programming the 68040 in the large buffer mode provides the fastest timing. In this mode the 68040 output buffers have an impedance low enough to switch a transmission line on the incident wave. A penalty of higher power consumption is incurred as terminators must be used.

In the small buffer mode the output impedance of the 68040 drivers is higher and the switching speed lower, but power consumption is reduced. This is the only option offered on the LC and EC variants of the 68040.

A jumper option block (Table II, ''Jumper3 Settings: selects buffer inputs for Interrupt Generator PAL'', Section 4.0) lets the user choose between these two modes. While the board is still operational in the small buffer mode at 40 MHz, those users desiring higher margins and operation at 40 MHz are advised to use the large buffer mode. For power conscious applications at 33 MHz and lower, the small buffer mode should be used.

2.6 Bussing Strategy

A strategy of minimizing the bus loading on the inner core is taken to allow operation at high frequency. All devices not in this critical path were moved to a peripheral bus buffered from this bus. This allows operation at 40 MHz and should hopefully take us to 66 MHz in next generation processors.

All I/O devices sit on an 8-bit bus located on an even word aligned boundary. When accessing these devices, the value of the upper 3 bytes on this bus is undefined. A 245 buffer isolates these devices from the faster 68040 bus, preventing bus contention when the recovery time of these devices may exceed the 68040 clock period.

As the 68040 does not support dynamic bus sizing to run start up code, either a 32-bit wide EPROM should be used or a byte gathering data path and state machine constructed. The second option was taken in this design.

2.7 PAL Operation

All PAL implementations are designed using National's OPAL™, a high level PAL programming language. The syntax of this programming language is straightforward and consistent with other PAL type programming languages. For further details and a listing of all the OPAL files, please see Section 5.0, ''PAL Equations''.

2.7.1 Address Decode Logic

The address decode logic is split up into two stages: the master decoder and the slave decoder. The master decoder PAL decodes the higher address bits and processor address space information to generate selects for the DRAM space, IACK and the slave decoder. For operation at 40 MHz and 33 MHz a fast 5 ns PAL must be used for the master decoder. The slave decoder generates selects for all other devices and for Mode Load which is used to program the DP8440.

2.7.2 Understanding the Address Decode

Master Decoder

The DP8440 chip select *(ramcs)* is decoded from the higher 68040 address lines, gated with *TS* and CLK to avoid false decoding starts. To maintain the chip select, a latch is implemented using a feedback term. For non-burst cycles this latch is cleared with */ta*, which is asserted following the rising edge of clock terminating the 68040 cycle. For burst cycles, *ramcs* is held through the burst cycle.

Other peripheral chip selects *(othercs)* are decoded and fed into a secondary decode logic in the Slave Decoder. Cache bursting is also inhibited for all non-dram cycles by asserting *cbi*.

Slave Decoder

The other I/O devices are decoded using *othercs* from the master decoder and the lower level bits of the 68040 decode. They are gated with *st1* from the Wait State Machine to delay the turning on of slower I/O devices. Signals *slw__rd* and *slw__wr* turn on slightly later than the main microprocessor controls.

2.7.3 Byte Enable

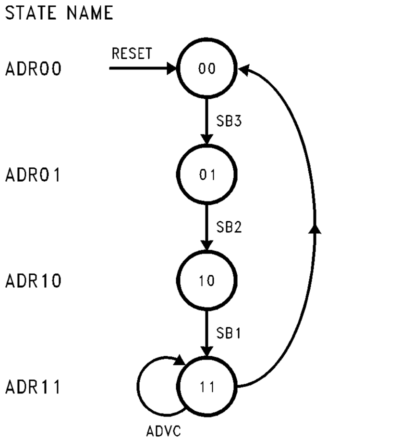
The Byte Enable PAL uses the *A[1:0]* and *SIZ[1:0]* signals of the 68040 to generate the byte enable signals for each of the four bytes. These are tied to the $\overline{\text{ECAS}}$ inputs of the DP8440.

The burst control (*burst* signal from PAL) of the DP8440 is generated from *SIZ[1:0]* signals of the 68040 and terminated before the fourth transfer acknowledge.

2.7.4 ADS Generation and EPROM Byte Gathering Machine

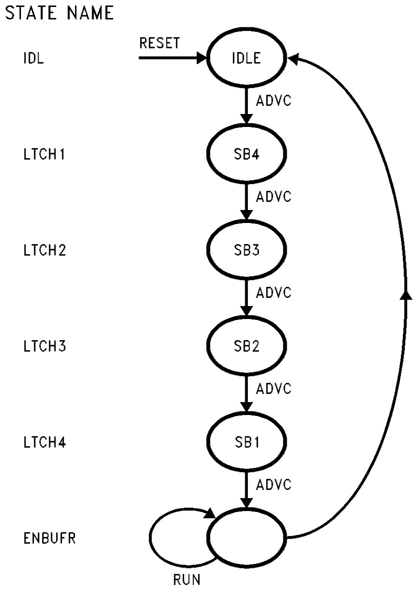
On decoding an access to the DRAM address space, $\overline{\text{ADS}}$ is asserted in the clock period following assertion of $\overline{\text{TS}}$. This provides the setup time for operation in page mode. Designs not using the page mode feature can skip this delay, whereas designs operating at higher frequencies and using the page mode feature may want to add more clock delay.

The 68040 does not support dynamic bus sizing and many users may not want to incur the cost of a 32-bit wide EPROM space. The byte gathering state machine translates each processor read into four byte accesses of the 8 bit wide EPROM. Generating the lower two address bits for the EPROM (state diagram in *Figure 4*) it assembles these bytes in four 74LS374 registers, enabling the buffers on the 68040 bus and signalling completion to the wait state machine. The state diagram for this machine is shown in *Figure 5*, “Byte Assembly Machine”.



TL/F/12471–4

FIGURE 4. Lower Bit Address Generation Machine



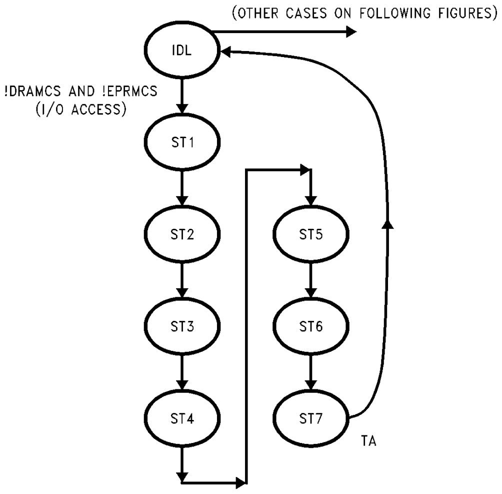
TL/F/12471–5

FIGURE 5. Byte Assembly Machine

2.7.5 Wait State Machine

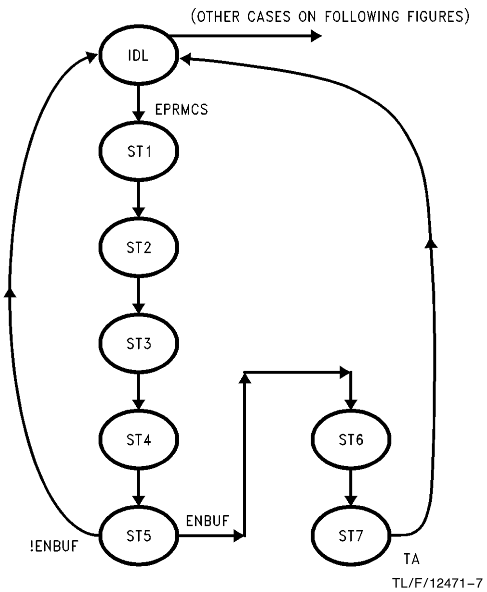
This is the main wait state machine and it follows different paths depending on whether the access is to I/O, EPROM or DRAM. An I/O access is to anything other than EPROM or DRAM; in this design that would include the dual UART, real time clock, hex switch, or LEDs. The state diagram for an I/O access is shown in *Figure 6*.

The state diagram for an EPROM access is shown in *Figure 7*. A DRAM access will follow one of two paths, depending on whether it is a burst access. For a non-burst access, the path shown in *Figure 8* will be followed, and for burst DRAM accesses, the path shown in *Figure 9* will be followed.



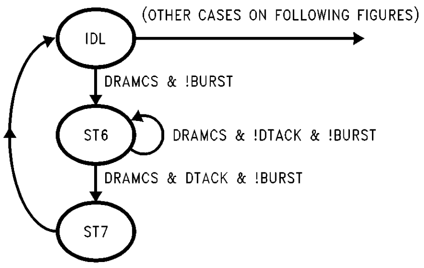
TL/F/12471–6

FIGURE 6. Wait State Machine Path for I/O Accesses



TL/F/12471–7

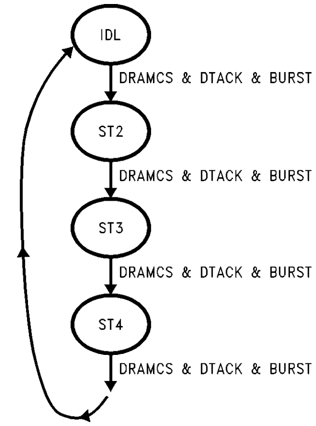
FIGURE 7. Wait State Machine Path for EPROM Accesses



TL/F/12471–8

Wait state machine does not generate *TA* for this access, *TA* is generated from *nadtack*.

FIGURE 8. Wait State Machine Path for Non-Burst DRAM Accesses



TL/F/12471–9

The principle function here is to count 4 bursts. Wait state machine does not generate *TA* for this access, *TA* is generated from *nadtack*.

FIGURE 9. Wait State Machine Path for DRAM Burst Access

2.7.6 Interrupt Encoder

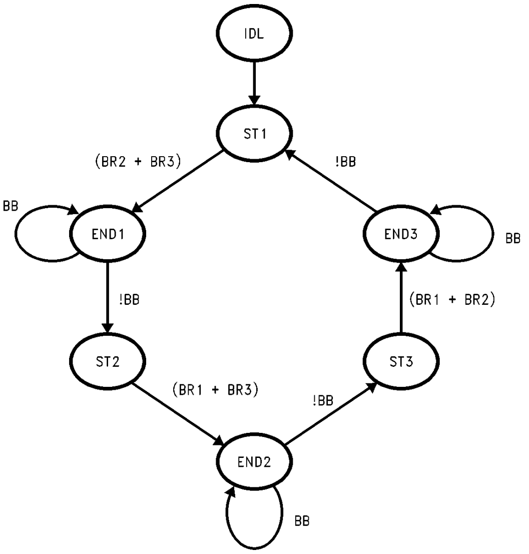
A priority encoder implemented on this device encodes the interrupts from the Real time clock and the two serial ports gating it through the IPL[2:0]. As buffer selection for the 68040 is sensed through the IPL[2:0] lines at reset time, this function is included in the encode logic.

2.7.7. Bus Arbiter

Three bus masters can take ownership in this design. The main CPU is master #1 and to request the bus it asserts the bus request signal *br1*. When this request is honored, the arbiter asserts the bus grant signal *bg1* to give ownership to master #1. The other two bus masters have access to the bus via the 100-pin expansion connector using the bus request signals (*br2* and *br3*) and the bus grant signals (*bg2* and *bg3*).

The bus arbiter is used to arbitrate between three bus masters. The arbitration scheme employed gives equal priority to each bus master in a round robin type of fashion. The state machine is shown in *Figure 10*.

In state 1 *(st1)*, master #1 is granted the bus and keeps it until a bus request comes from any of the other masters. Once this request comes, the state machine proceeds to the state *end1*. In this state, the bus is taken away from



TL/F/12471–10

FIGURE 10. State Machine for Bus Arbiter

master #1 and given to master #2. If the bus is busy (*bb* is asserted), it means master #1 is finishing up a bus access and the state machine must remain in *end1* until the bus is no longer busy. Once the bus is free, then state 2 *(st2)* can be entered.

The states *st2* and *st3* are very similar to *st1* in that they proceed to the next state only when a bus request comes from either of the other two bus masters. Likewise, *end2* and *end3* are similar to *end1* in that they only proceed to the next state when the bus is free.

2.8 I/O System

The I/O system consists of four sub-sections: the 8570 Real Time Clock, the 16552 Dual UART, a hex switch, and a LED display. A complete address map is given in Table I. For more information on the bus interconnections between the I/O system and the CPU, see Section 2.6, ‘‘Bussing Strategy’’.

TABLE I. Address Map

Addr. Range	Device	Width	Type
00000000H– 01FFFFFFH	EPROM	32 (effective)	Read Only
02000000H– 03FFFFFFH	Register	8	Write Only
04000000H– 05FFFFFFH	Switch	8	Read Only
06000000H– 06FFFFFFH	UART	8	Read/Write
08000000H– 09FFFFFFH	RTC	8	Read/Write
10000000H– 1FFFFFFFH	ML (Mode Load 8440)	*	Write Only
20000000H– 3FFFFFFFH	DRAM	32	Read/Write

2.8.1 DP8570 Timer Clock Peripheral

The clock used in this design is National Semiconductor's DP8570. For more information on this device please refer to the DP8570A datasheet. The DP8570 provides the timer functions and interrupt structure required to time events and is used in benchmarking certain sections of code. The DP8570 is mapped to the hexadecimal address range x08000000–x09FFFFFF. For detailed information on programming this part please refer to the datasheet.

The DP8570 is mapped as an 8-bit wide port on the D[31:24] data lines. Registers internal to the device are selected by changing A[6:2] bits of the microprocessor address. These address lines are connected to the A[4:0] lines on the device.

A 32.768 kHz crystal provides the time base for the oscillator and the INTR output of the DP8570 generates the RTC INTERRUPT to the 68040 on priority level 6.

Though a battery backed SRAM and Power Fail support are available on the device, they are not supported on this board. Therefore, the DP8570 must be reinitialized after each power-up and no reliance can be placed on the real time clock and calendar registers.

2.8.2 16552 Dual UART

The UART used in this design is National Semiconductor's PC16552 (dual UART). The 2 serial ports on the board are implemented through this device. Serial port 1 connects through the DB25 pin RS-232 connector and Serial port 2 connects through the DB9 pin RS-232 connector.

After power up the UARTs emulate the Industry standard 16450 UART. The user has the option of programming them in FIFO mode, in which case 16 byte FIFOs buffer data paths in the transmit and receive direction. Timing for the on chip baud rate generators is provided by a 18.432 MHz crystal.

INTR1 and INTR2 interrupts from this device generate interrupts to the 68040 on levels 4 and 5 respectively.

The general purpose multifunction output $\overline{\text{MF2}}$ is used to configure the DRAM subsystem in the NOBURST mode. When the board powers up, this is the default state. To enable bursts, the user must program the device to gate $\overline{\text{OUT2}}$ on $\overline{\text{MF2}}$; then set bit 3 ($\overline{\text{OUT2}}$) of the associated channels Modem control register to a 1.

The 16552 in this design is mapped to the address range x06000000–x06FFFFFF. For a detailed description of this part and a mapping of the internal registers please refer to the datasheet.

2.8.3 Hex Switch

The hex switch is 4 bits wide and can be set at any value from 0 hex–F hex. The switch is mapped to the address range x04000000–x05FFFFFF. When reading this switch, however, a byte wide value is returned which consists of the following:

- Four bit BCD field read from the HEX switch

Data Bus	31	30	29	28
Hex Switch	MSB	. . . .		LSB

- The four-bit presence detect bits output by the SIMM which indicate the speed of the SIMMs

Data Bus	27	26	25	24
Presence Detect	4	3	2	1

Note: Two bits of the HEX switch have been connected incorrectly. Bit 0 and Bit 2 should be switched in software to read the correct value.

2.8.4 LED display

This is an 8-bit wide register which drives the diagnostic LEDs and is mapped to the address range $02000000–03FFFFFF. The LED port is ‘‘Write only’’ and an undefined value is read from this address range.

Data Bus	31	30	29	28	27	26	25	24
LED	7	6	5	4	3	2	1	0

3.0 SOFTWARE

The software for this design falls into three separate groups: diagnostic, target, and host. The diagnostic portion was used in the early phases of making the board functional. The target software consists of the final software running on the DP8440 demo board. Finally, the host software runs on a PC, and contains the user interface to the target.

3.1 Diagnostic

The diagnostic code is written in 68040 assembly. The development cycle of the diagnostic code consisted of two phases. The first phase tested the basic functionality of each section of the design, mainly the UART, Real Time Clock, Hex Switch, LEDs, and DRAM system. The second phase consisted of more exhaustive tests, concentrating mainly on the DRAM system.

3.1.1 Phase 1

The strategy of phase 1 debugging involved writing a test suite which contained many small routines testing each section of the board. This test suite started with one simple program which read from the hex switch and then wrote the result to the LEDs. One problem immediately became evident: two bits on the hex switch had been connected incorrectly—bits 0 and 2 were switched. This problem was then reversed in software and the program was then able to verify the functionality of the hex switch and LEDs.

At this stage the DRAM system was not yet functional, precluding the use of a stack. In order to utilize common routines, much of the code uses macros and in line assembly in place of subroutine calls to avoid the use of the stack. This was done at the expense of program efficiency. However, it proved to be a valuable debugging strategy until the DRAM system became functional.

Once the hex switch and LEDs proved functional, other tests were written to test various portions of the design. Depending on the setting of the hex switch on reset, the respective test would be run. The LEDs were used in some cases to verify the success of these tests.

3.1.2 Phase 2

At this stage all sections of the design appear functional, and the stack is working properly. Subroutine calls are used throughout the program, and the diagnostic software is now a very simple monitor running on the target. A dumb terminal is used to communicate to the target via the 25-pin RS-232 serial port. The following operations can be performed on the target from the dumb terminal using a menu driven interface:

- Set Real Time clock
- Read clock
- Dump memory
- Write memory
- Normal mode DRAM test
- Burst mode DRAM test
- Page mode DRAM test
- Byte write test

3.2 Target

The target software is a simple monitor which can perform the following tasks:

- Program DP8440 in different modes
- Run Dhrystone benchmark
- Run memory move benchmark
- Write memory
- Dump memory
- DRAM test
- Real Time clock access

3.3 Host

The host software runs on a PC and communicates to the Target through the serial ports on the both PC and Target. The user interface of the host software allows the user to easily program the DP8440 in any valid mode and perform both Dhrystone and memory move benchmarks.

4.0 JUMPER SETTINGS

TABLE II. Jumper3 Settings: Selects Buffer Inputs for Interrupt Generator PAL

PAL Input	Connect Pins
SELDBBUF	1 to 2
SELADDBUF	3 to 4
SELCNTRLBUF	5 to 6
SELMDIS	7 to 8

TABLE III. Jumper 4: R/$\overline{W}$ Signal Used for ROM Emulator

Pin	Signal
1	R/$\overline{W}$ from CPU
2	Ground

TABLE IV. Jumper 5 Settings: Selects Clock Input for DP8440

Clock Type	Connect Pins
Positive CLK	2 to 3
Invert CLK	2 to 1

TABLE V. Jumper 6: 12V Supply for CPU Fan

Pin	
1	12V
2	Ground

TABLE VI. Jumper 7: Pin Expansion Connector of ISA Bus

J7 Pin Number	ISA Signal
Refer to schematics for signal connections	

TABLE VII. Jumpers 8–11 Settings: Input Select for 74ABT646

Jumper	Signal Names	Connect Pins	
J8	SBA	2 to 1	Low
		2 to 3	High
J9	CBA	2 to 1	Floating
		2 to 3	Clock D
J10	SAB	2 to 1	High
		2 to 3	Low
J11	CAB	2 to 1	Floating
		2 to 3	Low

TABLE VIII. Jumper J12 and J13: 100-Pin Expansions of CPU and DRAM Signals

Jumper	Function
J12	Expansion of 68040 bus
J13	Expansion of DRAM SIMM signals

TABLE IX. Jumper 15 Settings: Selects Address Input Lines for the DP8440

SIMM Type	Connect Signals	Connect Pins
1 MB x 36 (Requires A[22:2]) or 2 MB x 36 (Requires A[23:2])	A23 to B1 A22 to B0 A25 to R10 A13 to R1 A12 to R0 A24 to C10	1 to 2 3 to 4 5 to 6 7 to 8 9 to 10 11 to 12
4 MB x 36 (Requires A[24:2]) or 8 MB x 36 (Requires A[25:2])	A25 to B1 A24 to B0 A23 to R10 A22 to R1 A13 to R0 A12 to C10	5 to 2 11 to 4 1 to 6 3 to 8 7 to 10 9 to 12

TABLE X. Jumper 16 Settings: Selects $\overline{RAS}$ Line for Second Bank Coming from 8440

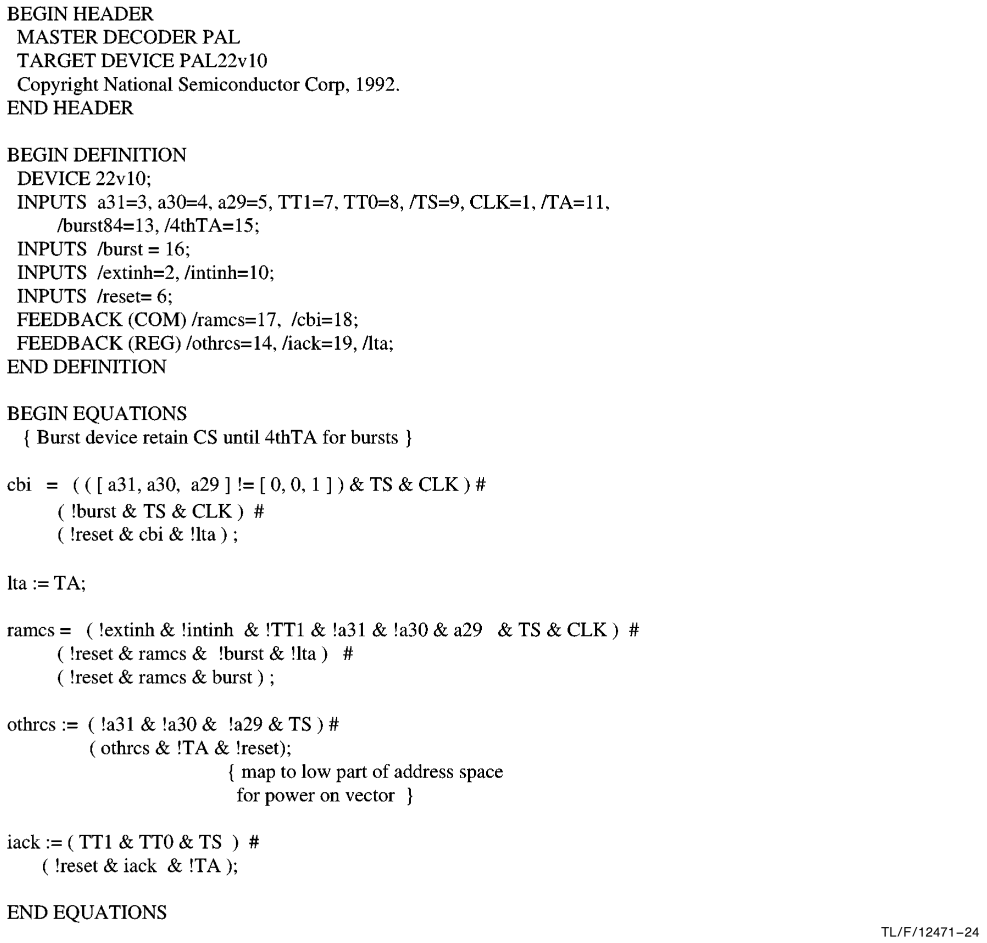
SIMM Type	1 Bank	2 Banks
1 MB x 36 or 4 MB x 36	Don't Care	$\overline{RAS1}$ Connect Pin 3 to 2
2 MB x 36 or 8 MB x 36	Don't Care	$\overline{RAS2}$ Connect Pin 1 to 2

TABLE XI. Jumper 19 Setting: Selects NOWRAP Input Line for DP8440

NOWRAP	Connect Pins	
Without Daughter Board (Not Using NOWRAP)	2 to 3	Pull-Down
With Daughter Board (Using NOWRAP)	2 to 1	Pull-Up

5.0 PAL EQUATIONS

5.1 Master Decoder



```
BEGIN HEADER
  MASTER DECODER PAL
  TARGET DEVICE PAL22v10
  Copyright National Semiconductor Corp, 1992.
END HEADER

BEGIN DEFINITION
  DEVICE 22v10;
  INPUTS  a31=3, a30=4, a29=5, TT1=7, TT0=8, /TS=9, CLK=1, /TA=11,
          /burst84=13, /4thTA=15;
  INPUTS  /burst = 16;
  INPUTS  /extinh=2, /intinh=10;
  INPUTS  /reset= 6;
  FEEDBACK (COM) /ramcs=17,  /cbi=18;
  FEEDBACK (REG) /othrcs=14, /iack=19, /lta;
END DEFINITION

BEGIN EQUATIONS
   { Burst device retain CS until 4thTA for bursts }

cbi   =   ( ( [ a31, a30,  a29 ] != [ 0, 0, 1 ] ) & TS & CLK ) #
          ( !burst & TS & CLK )  #
          ( !reset & cbi & !lta ) ;

lta := TA;

ramcs =   ( !extinh & !intinh  & !TT1 & !a31 & !a30 & a29   & TS & CLK )  #
          ( !reset & ramcs &  !burst & !lta )   #
          ( !reset & ramcs & burst ) ;

othrcs :=  ( !a31 & !a30 &  !a29 & TS ) #
              ( othrcs & !TA & !reset);
                                    { map to low part of address space
                                      for power on vector  }

iack := ( TT1 & TT0 & TS  )  #
       ( !reset & iack  & !TA );

END EQUATIONS
```

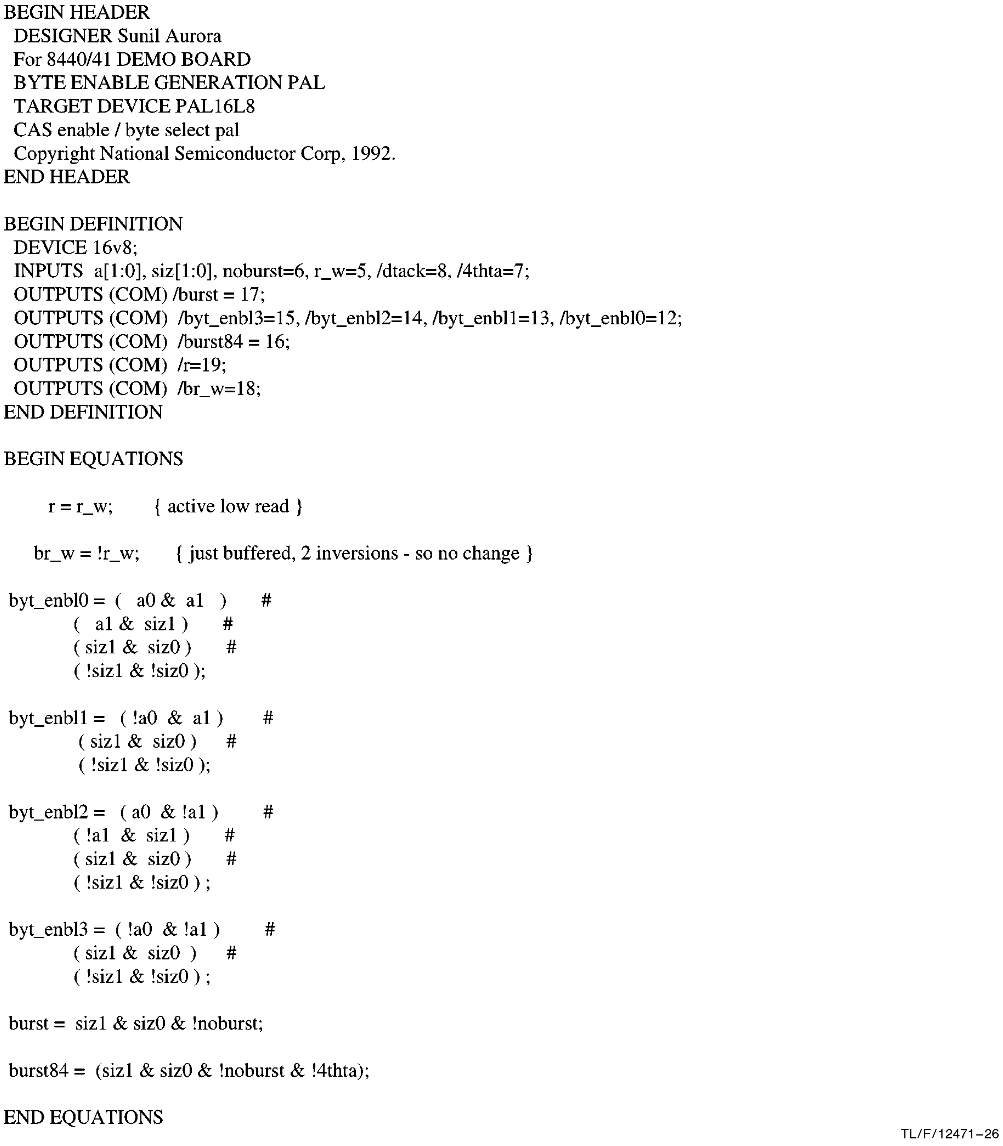
TL/F/12471-24

5.2 Slave Decoder

```
BEGIN HEADER
  DESIGNER Sunil Aurora
  For 8440/41 DEMO BOARD
  SLAVE DECODER PAL
  TARGET DEVICE 22V10
END HEADER

BEGIN DEFINITION
  DEVICE 22V10;
  INPUTS  CLK=1, /othrcs = 2, a28=3, a27=4, a26=5, a25=6,  /ta=7;
  INPUTS  /reset=9, r_w=8;
  INPUTS  /wsb2=10, /wsb1=11, /wsb0=13;
  FEEDBACK (REG) /eprmcs=23, /regwr=22, /swrd=21, /ml=20;
  FEEDBACK (REG) /uartcs=19, /rtcs=18, /slw_rd=17, /slw_wr=16, /ioenbl=15;
END DEFINITION

BEGIN EQUATIONS

st1   =  !wsb2 & !wsb1 & wsb0 ;

ioenbl :=  othrcs * /a28 * /a27 * /a26 *  a25 *  st1 * !ta  *  !reset * !r_w  # { regwr }
           othrcs * /a28 * /a27  *  a26 *  /a25 *  st1 * !ta * !reset * r_w   # { swrd }
           othrcs * /a28 * /a27 * a26 * a25 * st1 * !ta * !reset   #          {uartcs}
           othrcs * /a28 * a27 * /a26 * /a25 * st1 * !ta * !reset   #         {rtcs}
           ioenbl * !ta * !reset;

eprmcs := othrcs * /a28 * /a27 * /a26 * /a25 *  st1 * !ta * !reset   #
         eprmcs * !ta * !reset ;

regwr  := othrcs * /a28 * /a27 * /a26 *  a25 *  st1 * !ta  *  !reset * !r_w  #
          regwr * !ta * !reset ;

swrd   := othrcs * /a28 * /a27  *  a26 *  /a25 *  st1 * !ta * !reset * r_w   #
          swrd  * !ta  *  !reset ;

uartcs := othrcs * /a28 * /a27 * a26 * a25 * st1 * !ta * !reset   #
           uartcs  * !ta  * !reset ;

rtcs    := othrcs * /a28 * a27 * /a26 * /a25 * st1 * !ta * !reset   #
           rtcs  * !ta  * !reset ;

ml     := othrcs * a28 * st1 * !ta * !reset  #
           ml   * !ta * !reset ;

slw_rd := othrcs * st1 * r_w #
           slw_rd * !ta * !reset;

slw_wr := othrcs * st1 * !r_w #
           slw_wr * !ta * !reset;

END EQUATIONS
```

TL/F/12471–25

5.3 Byte Enable Logic



```
BEGIN HEADER
 DESIGNER Sunil Aurora
 For 8440/41 DEMO BOARD
 BYTE ENABLE GENERATION PAL
 TARGET DEVICE PAL16L8
 CAS enable / byte select pal
 Copyright National Semiconductor Corp, 1992.
END HEADER

BEGIN DEFINITION
 DEVICE 16v8;
 INPUTS  a[1:0], siz[1:0], noburst=6, r_w=5, /dtack=8, /4thta=7;
 OUTPUTS (COM) /burst = 17;
 OUTPUTS (COM)  /byt_enbl3=15, /byt_enbl2=14, /byt_enbl1=13, /byt_enbl0=12;
 OUTPUTS (COM)  /burst84 = 16;
 OUTPUTS (COM)  /r=19;
 OUTPUTS (COM)  /br_w=18;
END DEFINITION

BEGIN EQUATIONS

      r = r_w;      { active low read }

    br_w = !r_w;      { just buffered, 2 inversions - so no change }

 byt_enbl0 =  (   a0 &  a1   )      #
           (   a1 &  siz1 )      #
           ( siz1 &  siz0 )      #
           ( !siz1 & !siz0 );

 byt_enbl1 =  ( !a0  &  a1 )       #
            ( siz1 &  siz0 )     #
            ( !siz1 & !siz0 );

 byt_enbl2 =  ( a0  & !a1 )        #
           ( !a1  &  siz1 )      #
           ( siz1 &  siz0 )      #
           ( !siz1 & !siz0 ) ;

 byt_enbl3 =  ( !a0  & !a1 )        #
           ( siz1 &  siz0  )     #
           ( !siz1 & !siz0 ) ;

 burst =  siz1 & siz0 & !noburst;

 burst84 =  (siz1 & siz0 & !noburst & !4thta);

END EQUATIONS
```

TL/F/12471–26

5.4 EPROM State Machine

```
BEGIN HEADER
 DESIGNER Sunil Aurora
 For 8440/41 DEMO BOARD
 STATE MACHINE FOR CONTROLLING EPROM DATA BUS
 A state machine design using a 16V8 with register default set to HOLD.
 Copyright National Semiconductor Corp, 1992.
 DESCRIPTION
 This PAL consists of 2 state machines

  LATCH MACHINE which latches sequential bytes from the 8 bit wide EPROM
   to 4 latches, generating a 32 bit word in the process for the 68040.
   It then enables the output of these latches onto the 68040 data bus.
   To account for the slower access time of the EPROM, the latch machine
   increments on the input signal "advc".

 ADDRESS MACHINE
  Generates the lower 2 bits of the address required by the EPROM.
  Independent of the start of the access the machine always fetches
  in the sequence 00 01 10 11 and ends up fetching the complete 32
  bit word.
  Note the address increments after the latch phase of the earlier
  machine to guarantee hold time .
END HEADER

BEGIN DEFINITION
 DEVICE 16V8;
 INPUT clk=1, /reset=2,/oe=11, /ts=4;
 INPUT  /eprmcs=3, /wsb2=7, /wsb1=8, /wsb0=9, /ramcs=5;
 STATEBIT  /sb4=15, /sb3=14, /sb2=13, /sb1=12, /enbuf=16;
 STATE_NAMES idl=^b00000, ltch1=^b00010, ltch2=^b00100
 STATE_NAMES ltch3=^b01000, ltch4=^b10000, enbufr=^b00001;
 STATEBIT /A1=17, /A0=18;
 FEEDBACK (REG)  /ads=19;
 STATE_NAMES ADR00=^b11 , ADR01=^b10 , ADR10=^b01 , ADR11=^b00;
 SET ADR  = A[1:0];
END DEFINITION

BEGIN EQUATION
ads.c = clk;
ads.oe = oe;
enbuf.c=clk;
sb[4:1].c=clk;
enbuf.oe=oe;
sb[4:1].oe=oe;
{ Internally used expressions }
run  = eprmcs;
st5  =  ( wsb2 &  !wsb1 & wsb0 ) ;
st1  =  ( !wsb2 &  !wsb1 & !wsb0 ) ;
advc = eprmcs * st5;
{ Externally used expressions  }
ads := (ts * !reset);
END EQUATION
```

TL/F/12471-27

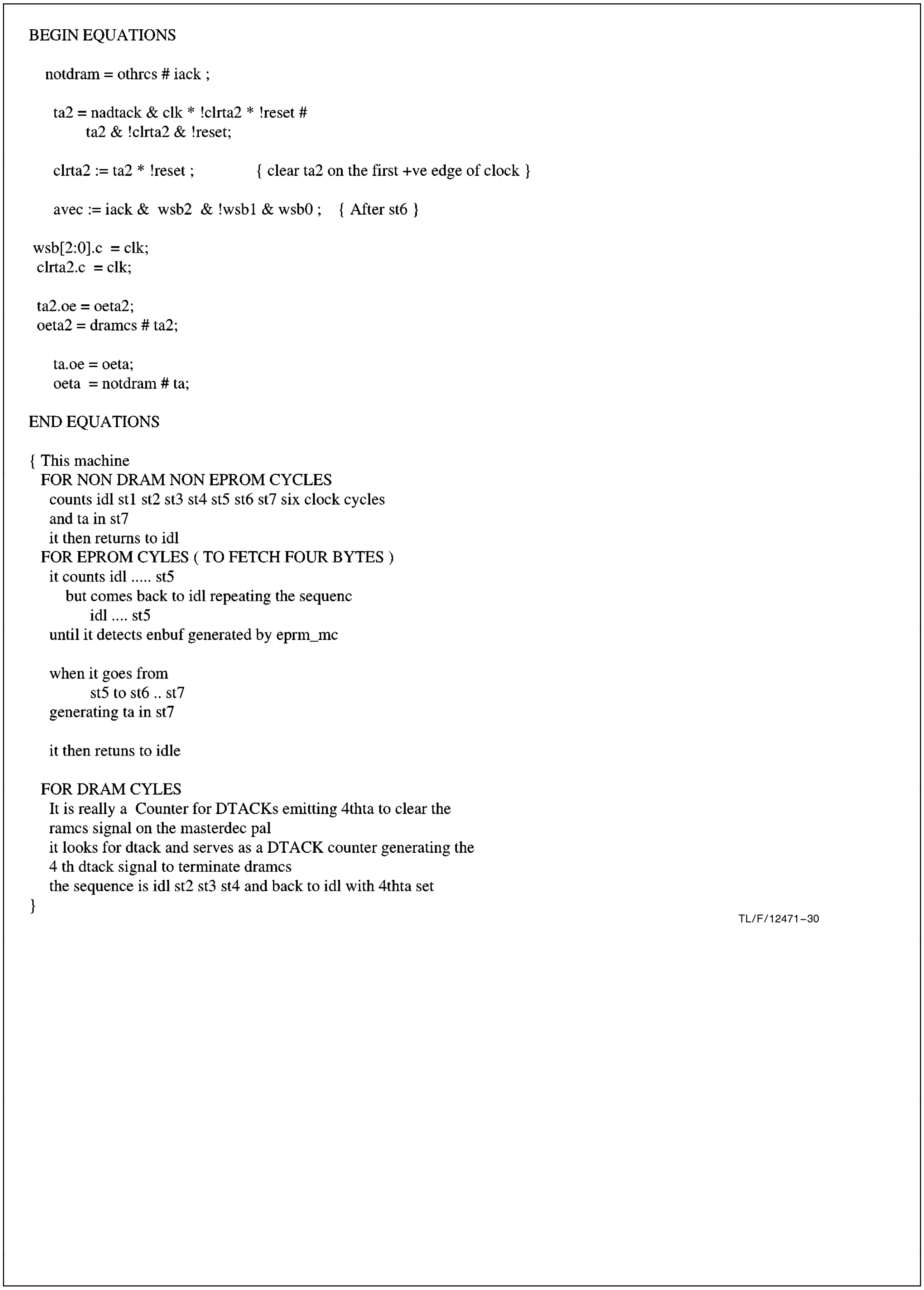
```
BEGIN STATE_DIAGRAM ( /sb1, /sb2, /sb3, /sb4, /enbuf )
  STATE ALL : IF reset THEN idl;
  STATE idl :
      if advc then ltch1      { 1st st5 }
      else idl;
  STATE ltch1 :
     if advc  then ltch2      { 2nd st5 }
     else if run  then ltch1
     else idl;
  STATE ltch2 :
     if advc  then ltch3      { 3rd st5 }
     else if run  then ltch2
     else idl;
  STATE ltch3 :
     if advc  then ltch4     { 4th st5 }
     else if run  then ltch3
     else idl;
  STATE ltch4 :
     if advc  then enbufr    { 5th st5 }
     else if run  then ltch4
     else idl;
  STATE enbufr :            { Stay here until eprmcs goes away }
     if run  then enbufr
     else idl;
  STATE UNDEFINED :
     goto idl;
END STATE_DIAGRAM

BEGIN STATE_DIAGRAM ( A1, A0 )

  STATE ALL : IF reset THEN ADR00;

  STATE ADR00 :
    IF sb3 then ADR01
    ELSE ADR00;

  STATE ADR01 :
   IF sb2 then ADR10
   ELSE ADR01;

  STATE ADR10 :
   IF sb1 then ADR11
   ELSE ADR10;

  STATE ADR11 :
   IF run then ADR11
   ELSE ADR00;

 END STATE_DIAGRAM
```

TL/F/12471–28

5.5 Inverse Clock Wait State Machine

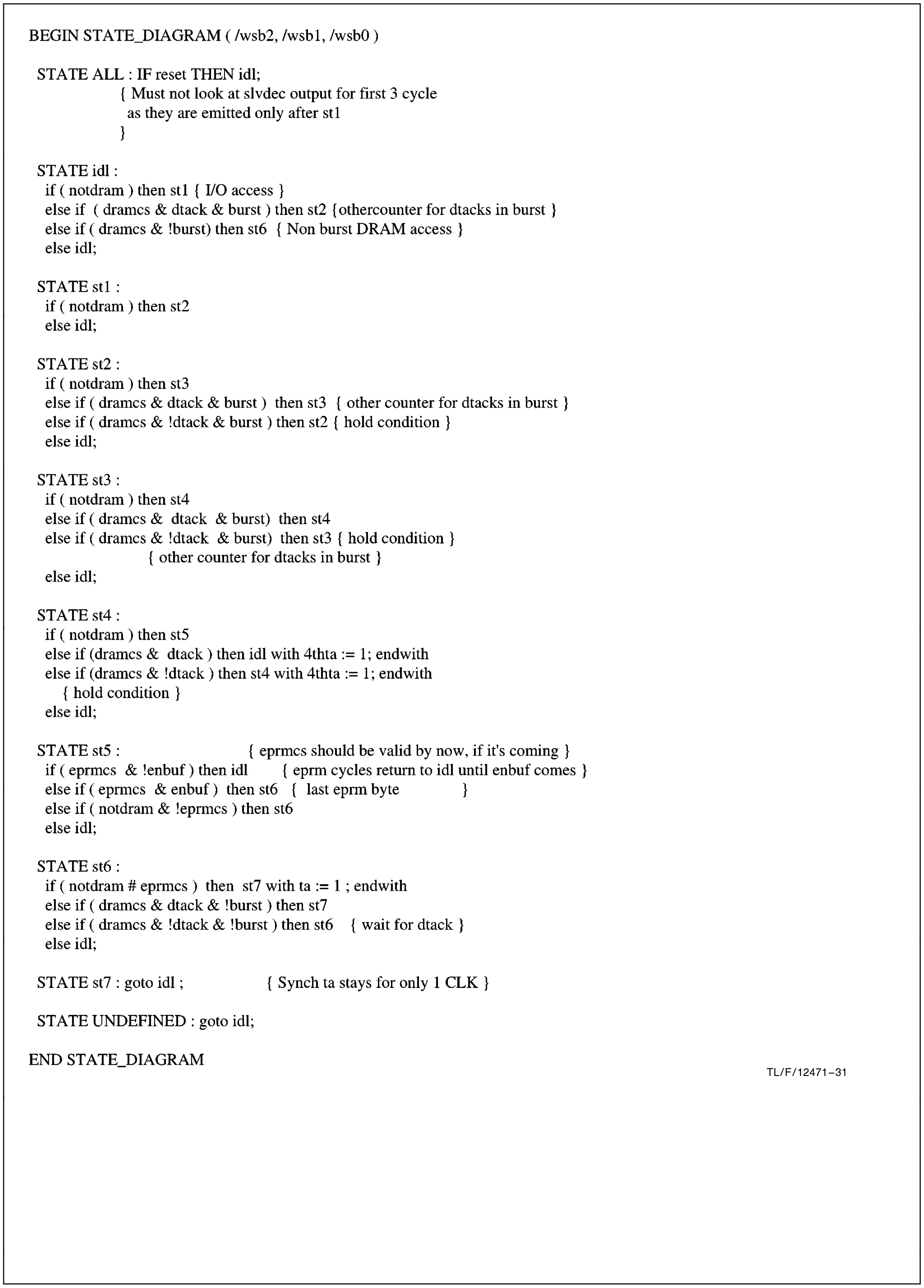
```
BEGIN HEADER
  WAIT STATE MACHINE PAL
  TARGET DEVICE PAL 22V10
  DESIGNER Sunil Aurora
  For 8440/41 DEMO BOARD
 DESCRIPTION
 Provides synchronous TA to 040
 Dtack for
         DRAM access comes from  8440
         using NADTACK and clk gating we generate TA2
         the path is combinational
         Caveats while enabling TA2 using oeta2
         EPRM |
         &    |          from state machine
         other|
 Dtack output through a latch, stays for only one clock cycle as per the new
 design convention
 Sections
 a Delay machine using  /wsb2 /wsb1  /wsb0
   /fourc is both a state and a output bit, it is used by the eprm m/c and indicates
   that four clocks have elapsed from cycle start for an access
   /fourc is emitted every time state st5 is reached
   For Eprom acesses it bounces back from idle to st5 and back to
   idle; unless enbuf is returned by eprm m/c  signalling data availability at
   the Eprom interface.
   It finally jumps to st6 where TA is set and then returns to idle.
   For non Eprom accesses it goes straight to st6.
   Delay for non-EProm acc. 5 clocks
   Delay for eprom = 4 * 4 + 1 clock
Generate wsb[2:0] bits used by eprm_mc slave decoder
 This machine counts through 5 clock ticks before returning TA
 to the 040.
END HEADER

BEGIN DEFINITION

  DEVICE 22V10;
  INPUTS  clk = 1, /enbuf=2, /reset=3;
  INPUTS  /dtack=10, /nadtack=8, /othrcs=4, /eprmcs=5, /iack=6;
  OUTPUT  /avec=23, /4thta=22;
  FEEDBACK (COM)  /ta2=17, /oeta2=15, /oeta=16;
  FEEDBACK (REG)  /ta=18,  /clrta2=14;
  INPUTS  /dramcs=7, /burst=11;  { Unused so far }
  STATEBIT /wsb2=19, /wsb1=20, /wsb0=21;
  STATE_NAMES  idl=^b000, st1=^b001, st2=^b010, st3=^b011, st4=^b100, st5=^b101,
  st6=^b110, st7=^b111;

END DEFINITION
```

TL/F/12471-29



```
BEGIN EQUATIONS

   notdram = othrcs # iack ;

     ta2 = nadtack & clk * !clrta2 * !reset #
           ta2 & !clrta2 & !reset;

     clrta2 := ta2 * !reset ;            { clear ta2 on the first +ve edge of clock }

     avec := iack &  wsb2  & !wsb1 & wsb0 ;   { After st6 }

 wsb[2:0].c  = clk;
  clrta2.c  = clk;

  ta2.oe = oeta2;
  oeta2 = dramcs # ta2;

     ta.oe = oeta;
     oeta  = notdram # ta;

END EQUATIONS

{ This machine
  FOR NON DRAM NON EPROM CYCLES
    counts idl st1 st2 st3 st4 st5 st6 st7 six clock cycles
    and ta in st7
    it then returns to idl
  FOR EPROM CYLES ( TO FETCH FOUR BYTES )
    it counts idl ..... st5
        but comes back to idl repeating the sequenc
             idl .... st5
    until it detects enbuf generated by eprm_mc

    when it goes from
             st5 to st6 .. st7
    generating ta in st7

    it then retuns to idle

  FOR DRAM CYLES
    It is really a  Counter for DTACKs emitting 4thta to clear the
    ramcs signal on the masterdec pal
    it looks for dtack and serves as a DTACK counter generating the
    4 th dtack signal to terminate dramcs
    the sequence is idl st2 st3 st4 and back to idl with 4thta set
}
```

TL/F/12471–30



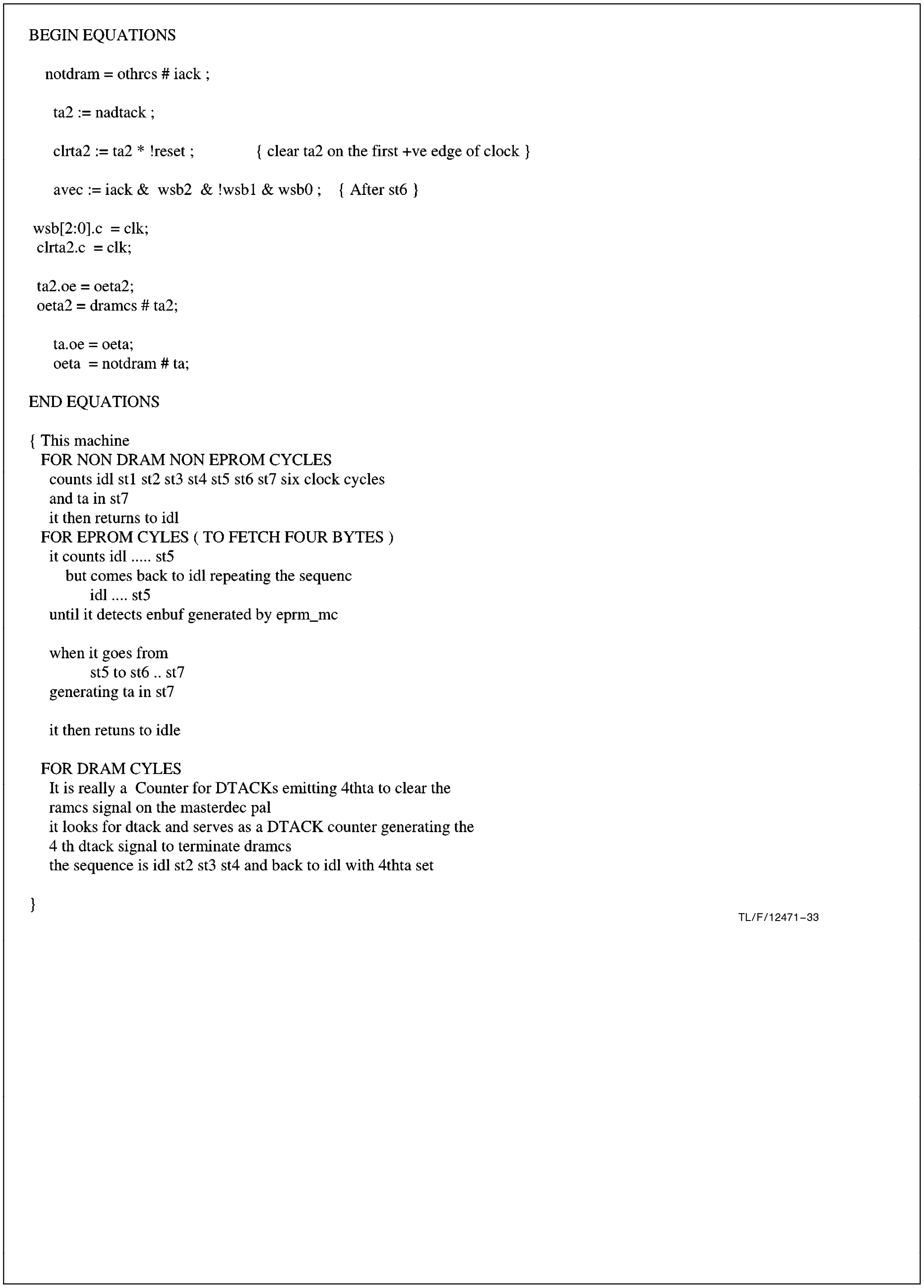
```
BEGIN STATE_DIAGRAM ( /wsb2, /wsb1, /wsb0 )

 STATE ALL : IF reset THEN idl;
                  { Must not look at slvdec output for first 3 cycle
                    as they are emitted only after st1
                  }

 STATE idl :
   if ( notdram ) then st1 { I/O access }
   else if  ( dramcs & dtack & burst ) then st2 {othercounter for dtacks in burst }
   else if ( dramcs & !burst) then st6  { Non burst DRAM access }
   else idl;

 STATE st1 :
   if ( notdram ) then st2
   else idl;

 STATE st2 :
   if ( notdram ) then st3
   else if ( dramcs & dtack & burst )  then st3  { other counter for dtacks in burst }
   else if ( dramcs & !dtack & burst ) then st2 { hold condition }
   else idl;

 STATE st3 :
   if ( notdram ) then st4
   else if ( dramcs &  dtack  & burst)  then st4
   else if ( dramcs & !dtack  & burst)  then st3 { hold condition }
                        { other counter for dtacks in burst }
   else idl;

 STATE st4 :
   if ( notdram ) then st5
   else if (dramcs &  dtack ) then idl with 4thta := 1; endwith
   else if (dramcs & !dtack ) then st4 with 4thta := 1; endwith
       { hold condition }
   else idl;

 STATE st5 :                          { eprmcs should be valid by now, if it's coming }
   if ( eprmcs  & !enbuf ) then idl        { eprm cycles return to idl until enbuf comes }
   else if ( eprmcs  & enbuf )  then st6   {   last eprm byte                }
   else if ( notdram & !eprmcs ) then st6
   else idl;

 STATE st6 :
   if ( notdram # eprmcs )  then  st7 with ta := 1 ; endwith
   else if ( dramcs & dtack & !burst ) then st7
   else if ( dramcs & !dtack & !burst ) then st6    { wait for dtack }
   else idl;

 STATE st7 : goto idl ;                 { Synch ta stays for only 1 CLK }

 STATE UNDEFINED : goto idl;

END STATE_DIAGRAM
```

TL/F/12471–31

5.6 Positive Clock Wait State Machine

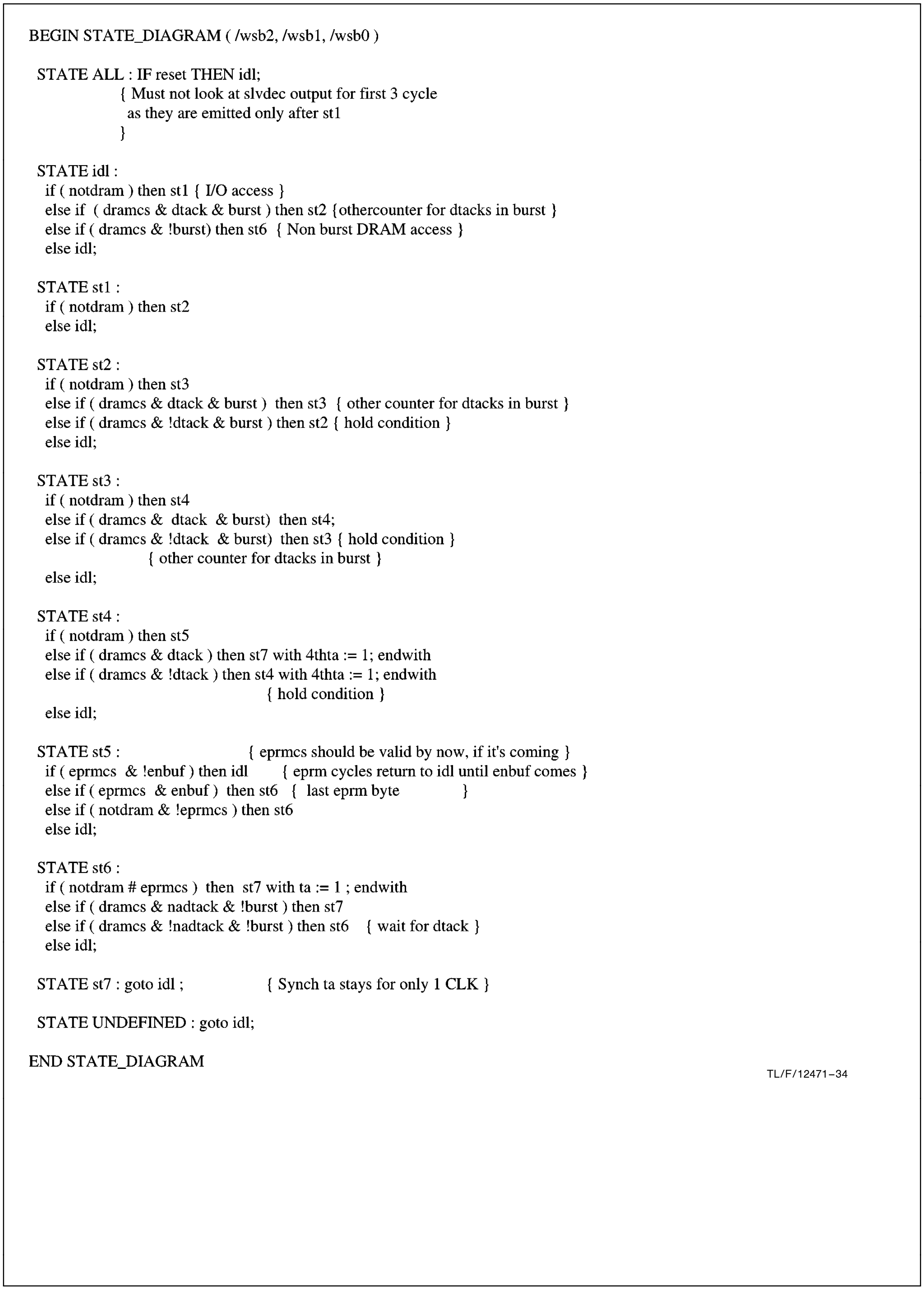
```
BEGIN HEADER
 WAIT STATE MACHINE PAL
 TARGET DEVICE PAL 22V10
 DESIGNER Sunil Aurora
 For 8440/41 DEMO BOARD
DESCRIPTION
Dtack for
        DRAM access comes from  8440
        using NADTACK and clk gating we generate TA2
        the path is combinational
        Caveats while enabling TA2 using oeta2
        EPRM |
        &   |           from state machine
        other|
Dtack output through a latch, stays for only one clock cycle as per the new
design convention
Sections
a Delay machine using  /wsb2 /wsb1  /wsb0
  /fourc is both a state and a output bit, it is used by the eprm m/c and indicates
  that four clocks have elapsed from cycle start for an access
  /fourc is emitted every time state st5 is reached
  For Eprom acesses it bounces back from idle to st5 and back to
  idle; unless enbuf is returned by eprm m/c  signalling data availability at
  the Eprom interface.
  It finally jumps to st6 where TA is set and then returns to idle.
  For non Eprom accesses it goes straight to st6.
  Delay for non-EProm acc. 5 clocks
  Delay for eprom = 4 * 4 + 1 clock
Generate wsb[2:0] bits used by eprm_mc slave decoder
 This machine counts through 5 clock ticks before returning TA
 to the 040.
END HEADER

BEGIN DEFINITION
 DEVICE 22V10;
 INPUTS clk = 1, /enbuf=2, /reset=3;
 INPUTS /dtack=10, /nadtack=8, /othrcs=4, /eprmcs=5, /iack=6;
 OUTPUT  /4thta=22, /avec=23;
 FEEDBACK (COM)  /oeta2=15, /oeta=16;
 FEEDBACK (REG)  /ta2=17, /ta=18,  /clrta2=14;
 INPUT /dramcs=7, /burst=11;  { Unused so far }
 STATEBIT /wsb2=19, /wsb1=20, /wsb0=21;
 STATE_NAMES  idl=^b000, st1=^b001, st2=^b010, st3=^b011, st4=^b100, st5=^b101,
 st6=^b110, st7=^b111;
END DEFINITION
```

TL/F/12471-32



```
BEGIN EQUATIONS

   notdram = othrcs # iack ;

     ta2 := nadtack ;

     clrta2 := ta2 * !reset ;           { clear ta2 on the first +ve edge of clock }

     avec := iack &  wsb2  & !wsb1 & wsb0 ;   { After st6 }

 wsb[2:0].c  = clk;
  clrta2.c  = clk;

  ta2.oe = oeta2;
  oeta2 = dramcs # ta2;

     ta.oe = oeta;
     oeta  = notdram # ta;

END EQUATIONS

{ This machine
   FOR NON DRAM NON EPROM CYCLES
     counts idl st1 st2 st3 st4 st5 st6 st7 six clock cycles
     and ta in st7
     it then returns to idl
   FOR EPROM CYLES ( TO FETCH FOUR BYTES )
     it counts idl ..... st5
         but comes back to idl repeating the sequenc
               idl .... st5
     until it detects enbuf generated by eprm_mc

     when it goes from
               st5 to st6 .. st7
     generating ta in st7

     it then retuns to idle

   FOR DRAM CYLES
     It is really a  Counter for DTACKs emitting 4thta to clear the
     ramcs signal on the masterdec pal
     it looks for dtack and serves as a DTACK counter generating the
     4 th dtack signal to terminate dramcs
     the sequence is idl st2 st3 st4 and back to idl with 4thta set

}
```

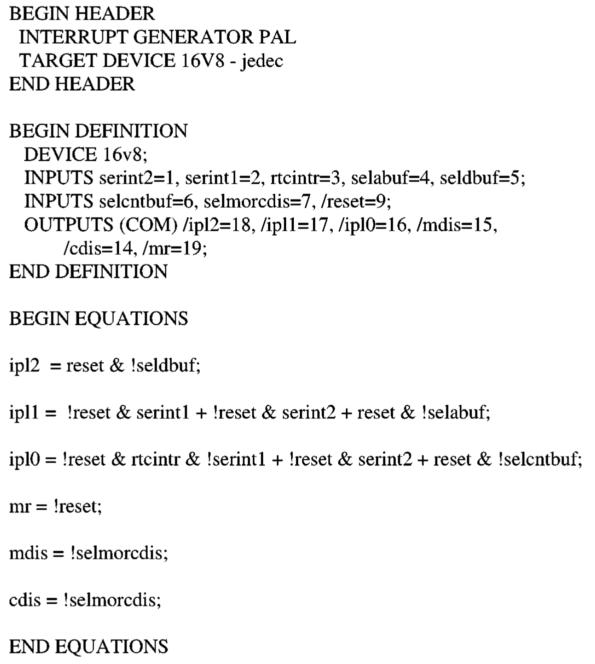
TL/F/12471–33



```
BEGIN STATE_DIAGRAM ( /wsb2, /wsb1, /wsb0 )

 STATE ALL : IF reset THEN idl;
                  { Must not look at slvdec output for first 3 cycle
                    as they are emitted only after st1
                  }

 STATE idl :
  if ( notdram ) then st1 { I/O access }
  else if  ( dramcs & dtack & burst ) then st2 {othercounter for dtacks in burst }
  else if ( dramcs & !burst) then st6  { Non burst DRAM access }
  else idl;

 STATE st1 :
  if ( notdram ) then st2
  else idl;

 STATE st2 :
  if ( notdram ) then st3
  else if ( dramcs & dtack & burst )  then st3  { other counter for dtacks in burst }
  else if ( dramcs & !dtack & burst ) then st2 { hold condition }
  else idl;

 STATE st3 :
  if ( notdram ) then st4
  else if ( dramcs &  dtack  & burst)  then st4;
  else if ( dramcs & !dtack  & burst)  then st3 { hold condition }
                        { other counter for dtacks in burst }
  else idl;

 STATE st4 :
  if ( notdram ) then st5
  else if ( dramcs & dtack ) then st7 with 4thta := 1; endwith
  else if ( dramcs & !dtack ) then st4 with 4thta := 1; endwith
                                                   { hold condition }
  else idl;

 STATE st5 :                            { eprmcs should be valid by now, if it's coming }
  if ( eprmcs  & !enbuf ) then idl        { eprm cycles return to idl until enbuf comes }
  else if ( eprmcs  & enbuf )  then st6   {   last eprm byte              }
  else if ( notdram & !eprmcs ) then st6
  else idl;

 STATE st6 :
  if ( notdram # eprmcs )  then  st7 with ta := 1 ; endwith
  else if ( dramcs & nadtack & !burst ) then st7
  else if ( dramcs & !nadtack & !burst ) then st6    { wait for dtack }
  else idl;

 STATE st7 : goto idl ;                   { Synch ta stays for only 1 CLK }

 STATE UNDEFINED : goto idl;

END STATE_DIAGRAM
```

TL/F/12471–34

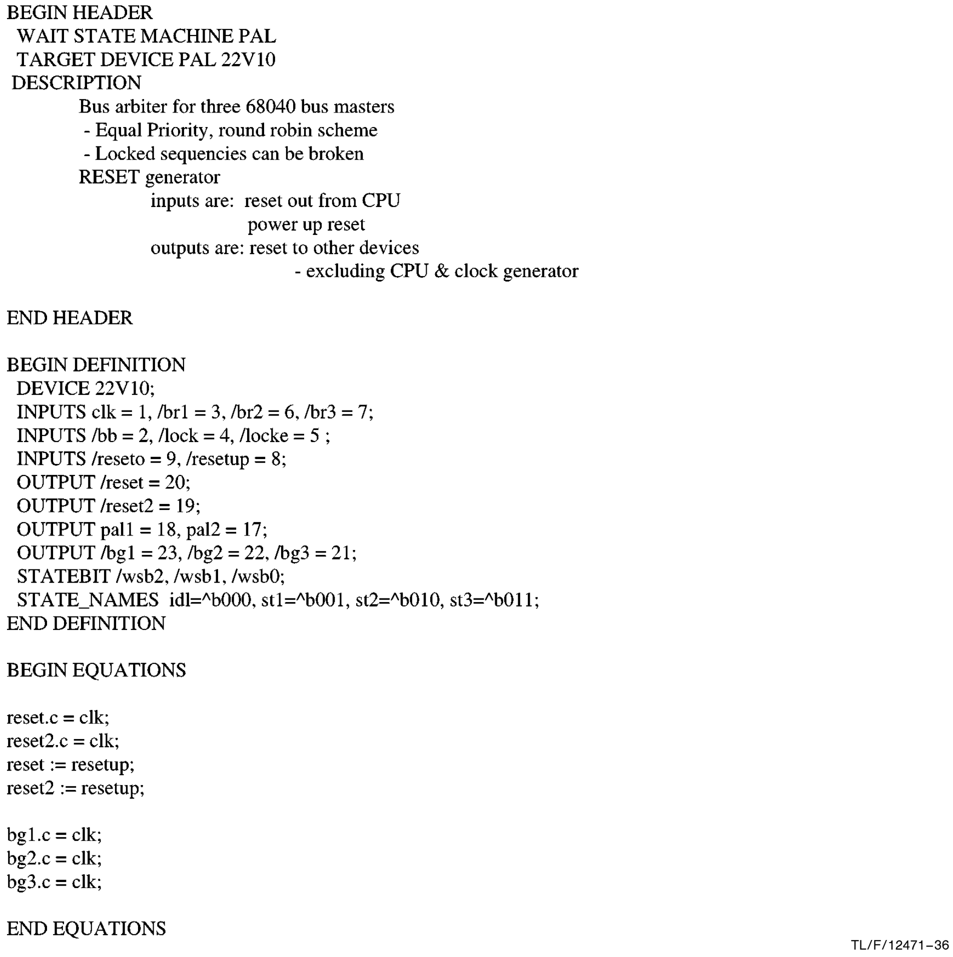
5.7 Interrupt Encoder



```
BEGIN HEADER
  INTERRUPT GENERATOR PAL
  TARGET DEVICE 16V8 - jedec
END HEADER

BEGIN DEFINITION
   DEVICE 16v8;
   INPUTS serint2=1, serint1=2, rtcintr=3, selabuf=4, seldbuf=5;
   INPUTS selcntbuf=6, selmorcdis=7, /reset=9;
   OUTPUTS (COM) /ipl2=18, /ipl1=17, /ipl0=16, /mdis=15,
           /cdis=14, /mr=19;
END DEFINITION

BEGIN EQUATIONS

ipl2  = reset & !seldbuf;

ipl1 =  !reset & serint1 + !reset & serint2 + reset & !selabuf;

ipl0 = !reset & rtcintr & !serint1 + !reset & serint2 + reset & !selcntbuf;

mr = !reset;

mdis = !selmorcdis;

cdis = !selmorcdis;

END EQUATIONS
```

TL/F/12471–35

5.8 Bus Arbiter



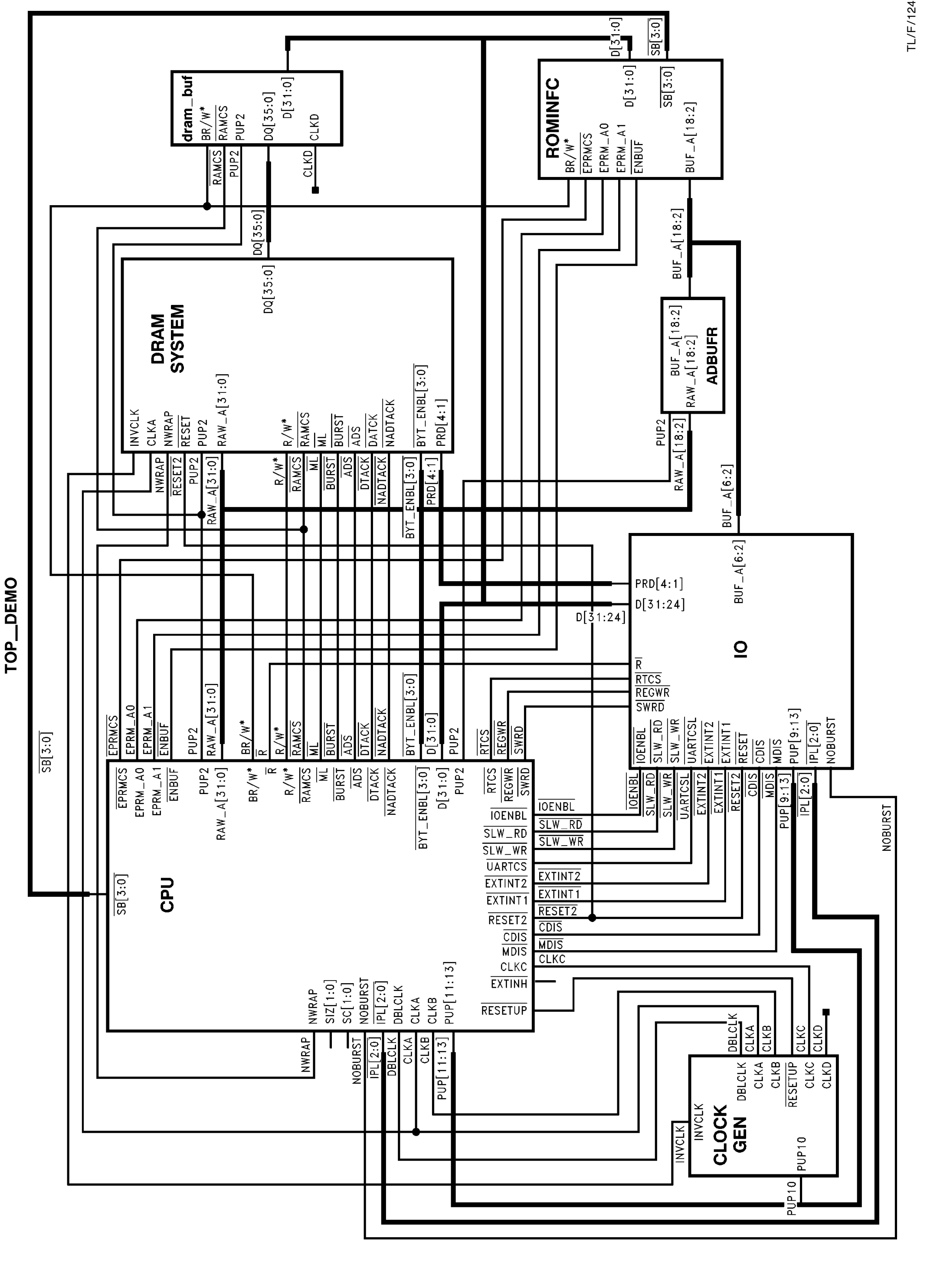
```
BEGIN HEADER
  WAIT STATE MACHINE PAL
  TARGET DEVICE PAL 22V10
 DESCRIPTION
        Bus arbiter for three 68040 bus masters
         - Equal Priority, round robin scheme
         - Locked sequencies can be broken
        RESET generator
                inputs are:  reset out from CPU
                             power up reset
                outputs are: reset to other devices
                                  - excluding CPU & clock generator

END HEADER

BEGIN DEFINITION
  DEVICE 22V10;
  INPUTS clk = 1, /br1 = 3, /br2 = 6, /br3 = 7;
  INPUTS /bb = 2, /lock = 4, /locke = 5 ;
  INPUTS /reseto = 9, /resetup = 8;
  OUTPUT /reset = 20;
  OUTPUT /reset2 = 19;
  OUTPUT pal1 = 18, pal2 = 17;
  OUTPUT /bg1 = 23, /bg2 = 22, /bg3 = 21;
  STATEBIT /wsb2, /wsb1, /wsb0;
  STATE_NAMES  idl=^b000, st1=^b001, st2=^b010, st3=^b011;
END DEFINITION

BEGIN EQUATIONS

reset.c = clk;
reset2.c = clk;
reset := resetup;
reset2 := resetup;

bg1.c = clk;
bg2.c = clk;
bg3.c = clk;

END EQUATIONS
```

TL/F/12471–36

```
BEGIN STATE_DIAGRAM ( /wsb2, /wsb1, /wsb0 )
 STATE idl :
  bg1 := 1;              {Give priority to CPU #1}
  bg2 := 0;
  bg3 := 0;
  if br1 then st1
  else if br2 then st2
  else if br3 then st3
  else idl;
 STATE st1 :             {This state is started with bg1 asserted}
  bg1 := 1;
  bg2 := 0;              {Grant bus to CPU #1}
  bg3 := 0;
  if (!br2 & !br3) then st1 {stay here until a request from
                              another master comes}
  else end1;
 STATE end1 :
  bg1 := 0;
  bg2 := 1;              {Grant the bus to master #2}
  bg3 := 0;
  if bb then end1 {stay here until bus access completes}
  else st2;
 STATE st2 :             {This state is started with bg2 asserted}
  bg1 := 0;
  bg2 := 1;              {Grant bus to CPU #2}
  bg3 := 0;
  if (!br1 & !br3) then st2 {stay here until a request from
                              another master comes}
  else end2;
 STATE end2  :
  bg1 := 0;
  bg2 := 0;              {Grant the bus to master #3}
  bg3 := 1;
  if bb then end2 {stay here until bus access completes}
  else st3;
 STATE st3 :             {This state is started with bg3 asserted}
  bg1 := 0;
  bg2 := 0;
  bg3 := 1;
  if (!br1 & !br2) then st3 {stay here until a request from
                              another master comes}
  else end3;
 STATE end3 :
  bg1 := 1;
  bg2 := 0;              {Grant the bus to master #1}
  bg3 := 0;
  if bb then end3 {stay here until bus access completes}
  else st1;
 STATE UNDEFINED : goto idl;

END STATE_DIAGRAM
```

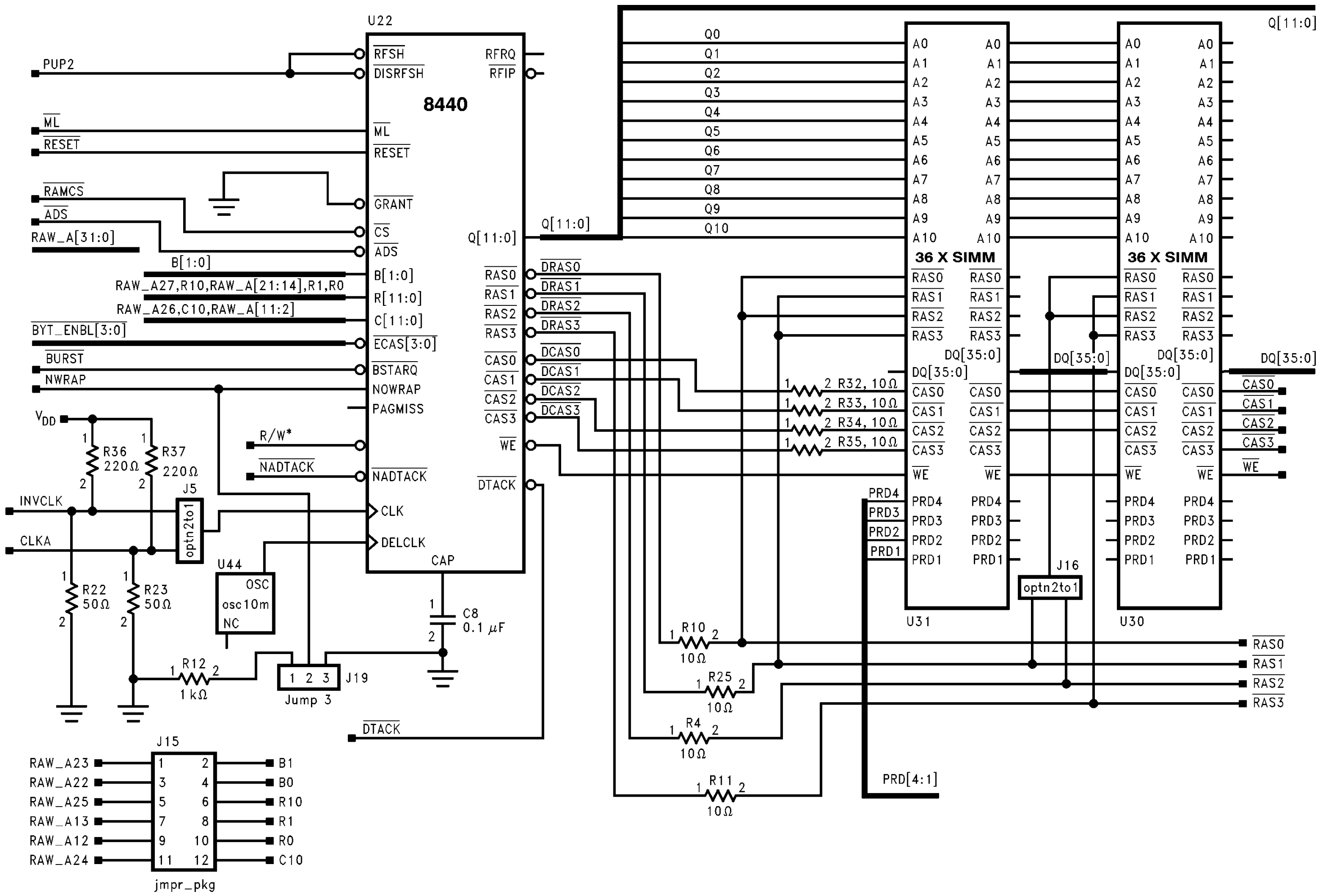
TL/F/12471–37

6.0 SCHEMATICS



TL/F/12471–2

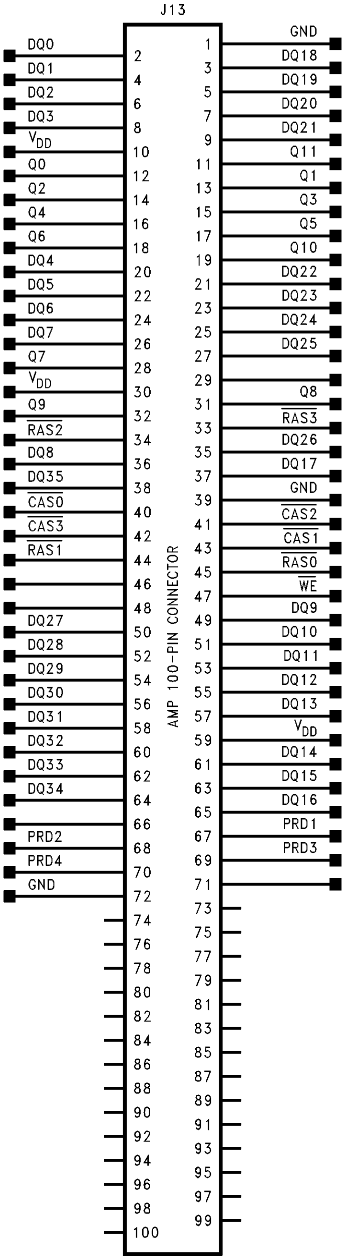
DRAM System (1/2)



TL/F/12471–3

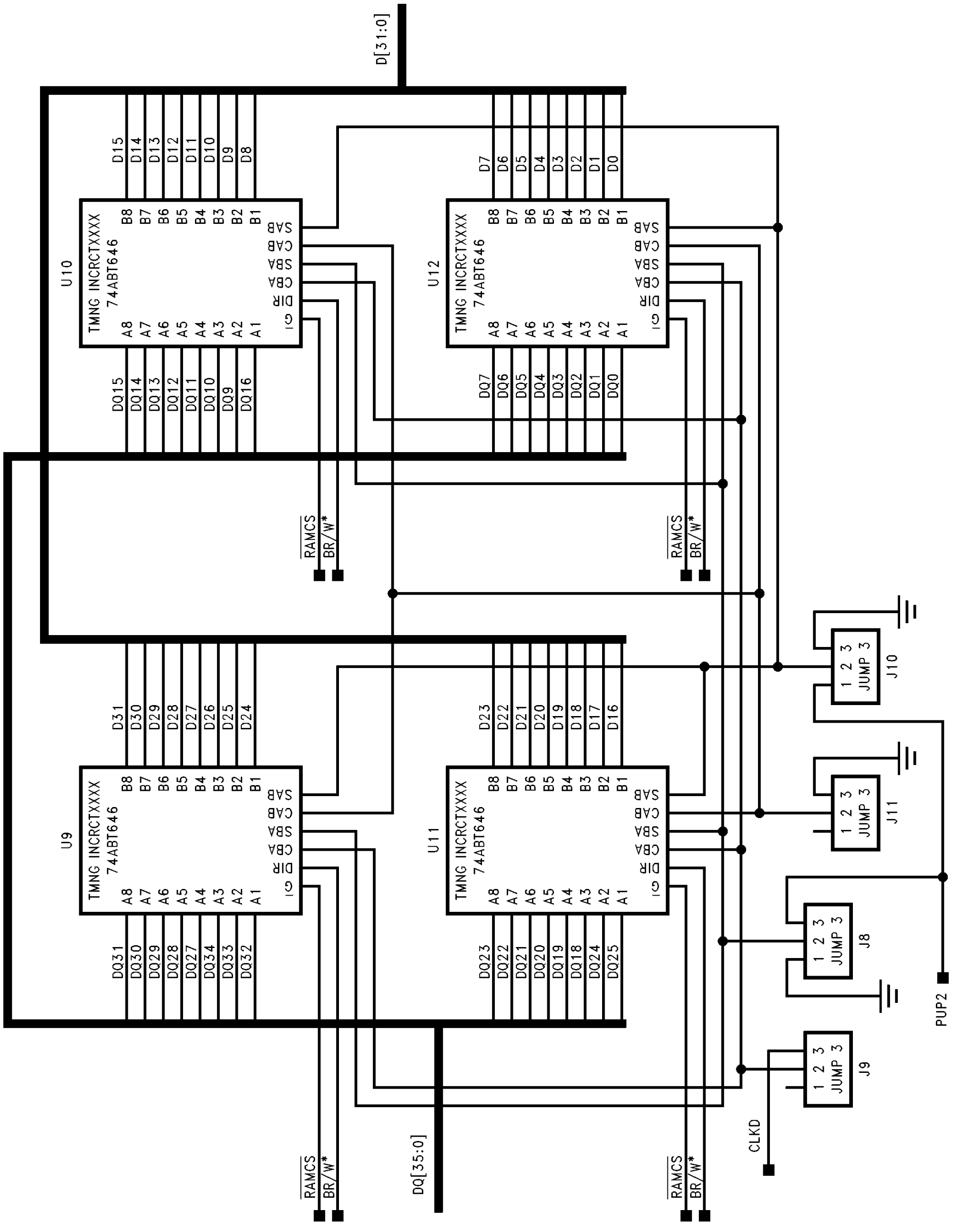
Option for 1, 2, 4, 8M x 36 SIMMs

DRAM System (2/2)
100-Pin Connector from SIMMs



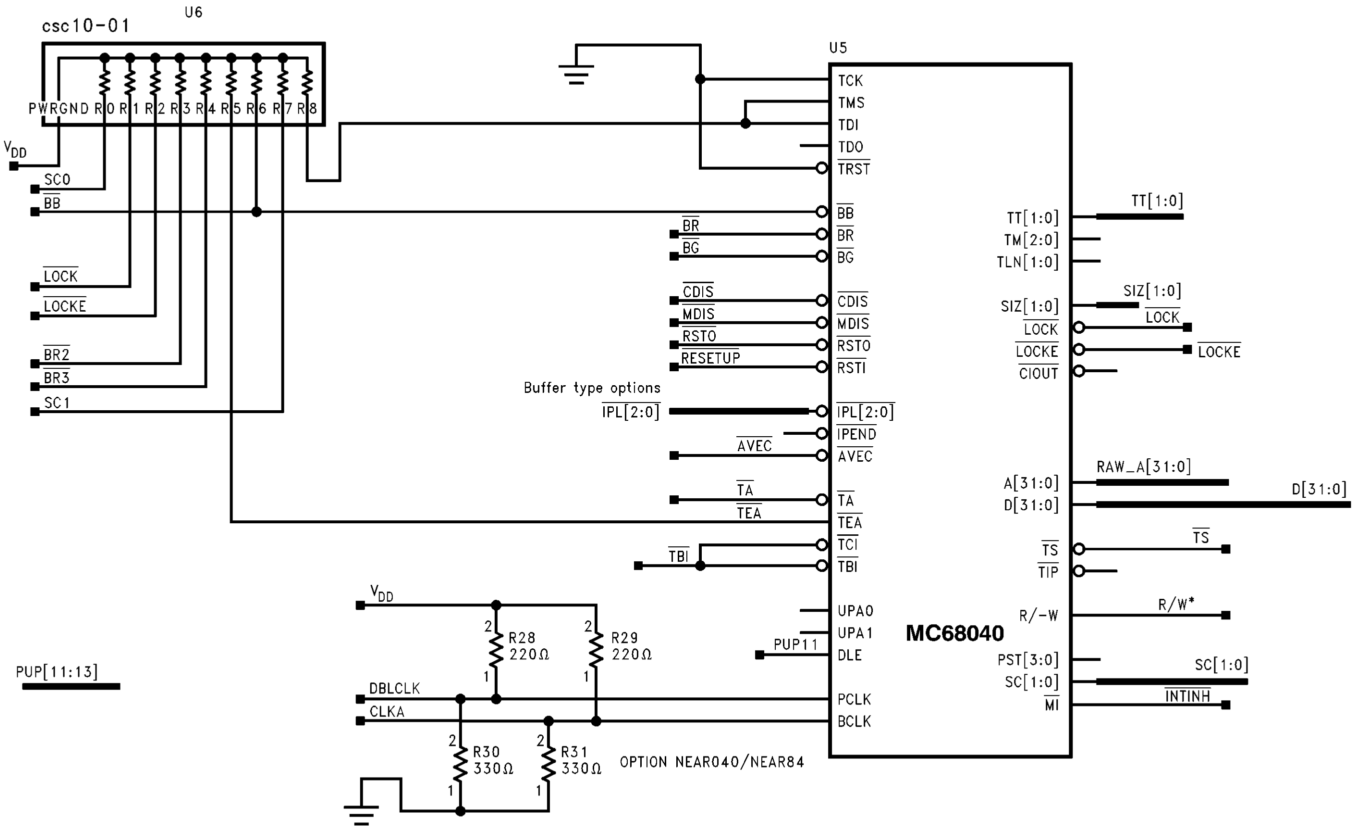
TL/F/12471–11

DRAM__BUF


D[31:0]
DQ[35:0]
U10
U12
U9
U11
TMNG INCRCTXXXX
74ABT646
A8 A7 A6 A5 A4 A3 A2 A1
B8 B7 B6 B5 B4 B3 B2 B1
SAB
CAB
SBA
CBA
DIR
G̅
D15 D14 D13 D12 D11 D10 D9 D8
D7 D6 D5 D4 D3 D2 D1 D0
D31 D30 D29 D28 D27 D26 D25 D24
D23 D22 D21 D20 D19 D18 D17 D16
DQ15 DQ14 DQ13 DQ12 DQ11 DQ10 DQ9 DQ16
DQ7 DQ6 DQ5 DQ4 DQ3 DQ2 DQ1 DQ0
DQ31 DQ30 DQ29 DQ28 DQ27 DQ34 DQ33 DQ32
DQ23 DQ22 DQ21 DQ20 DQ19 DQ18 DQ24 DQ25
RAMCS
BR/W*
1 2 3
JUMP 3
J10
J11
J8
J9
PUP2
CLKD

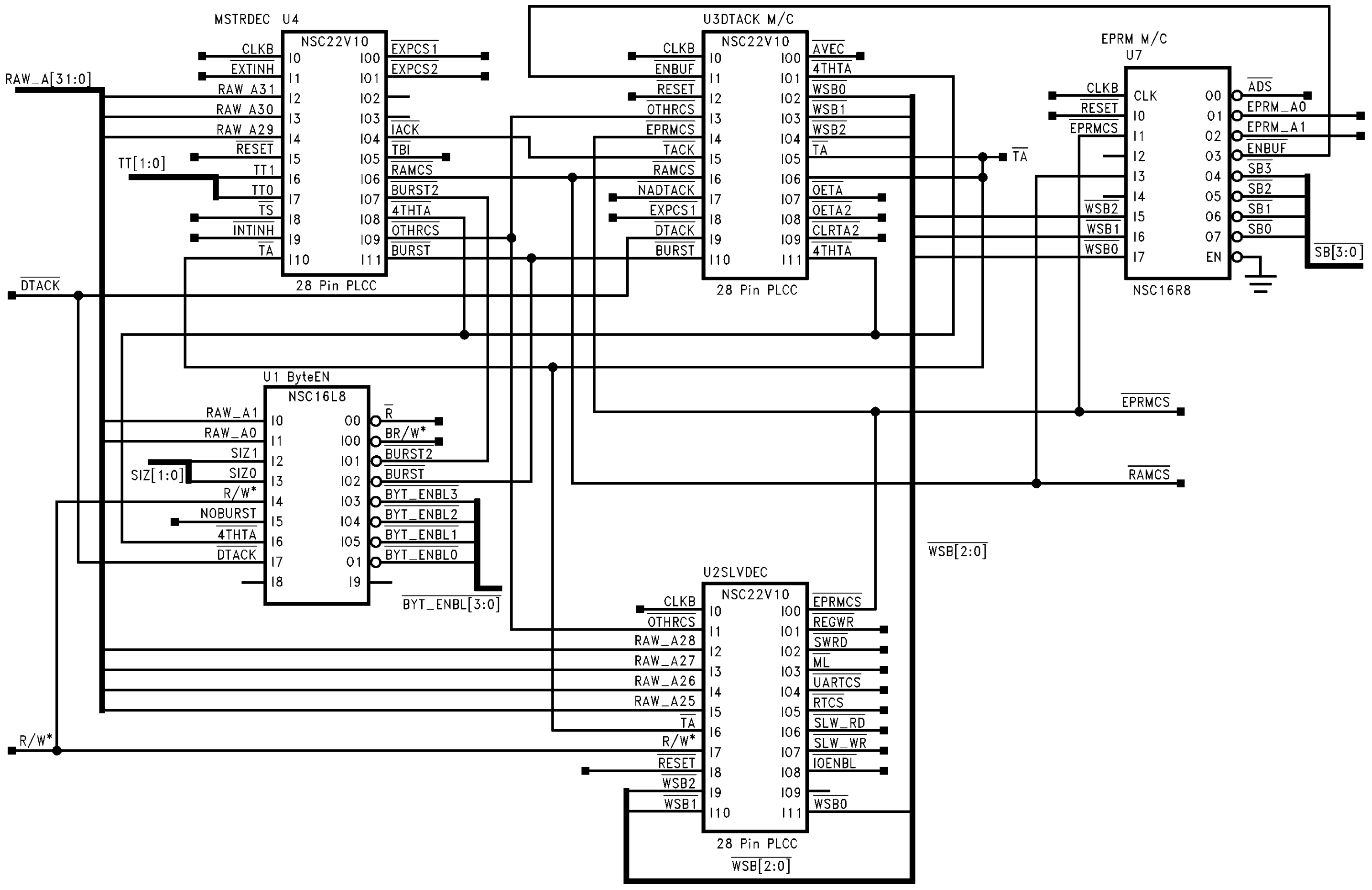
TL/F/12471-12

Processor (1/5)

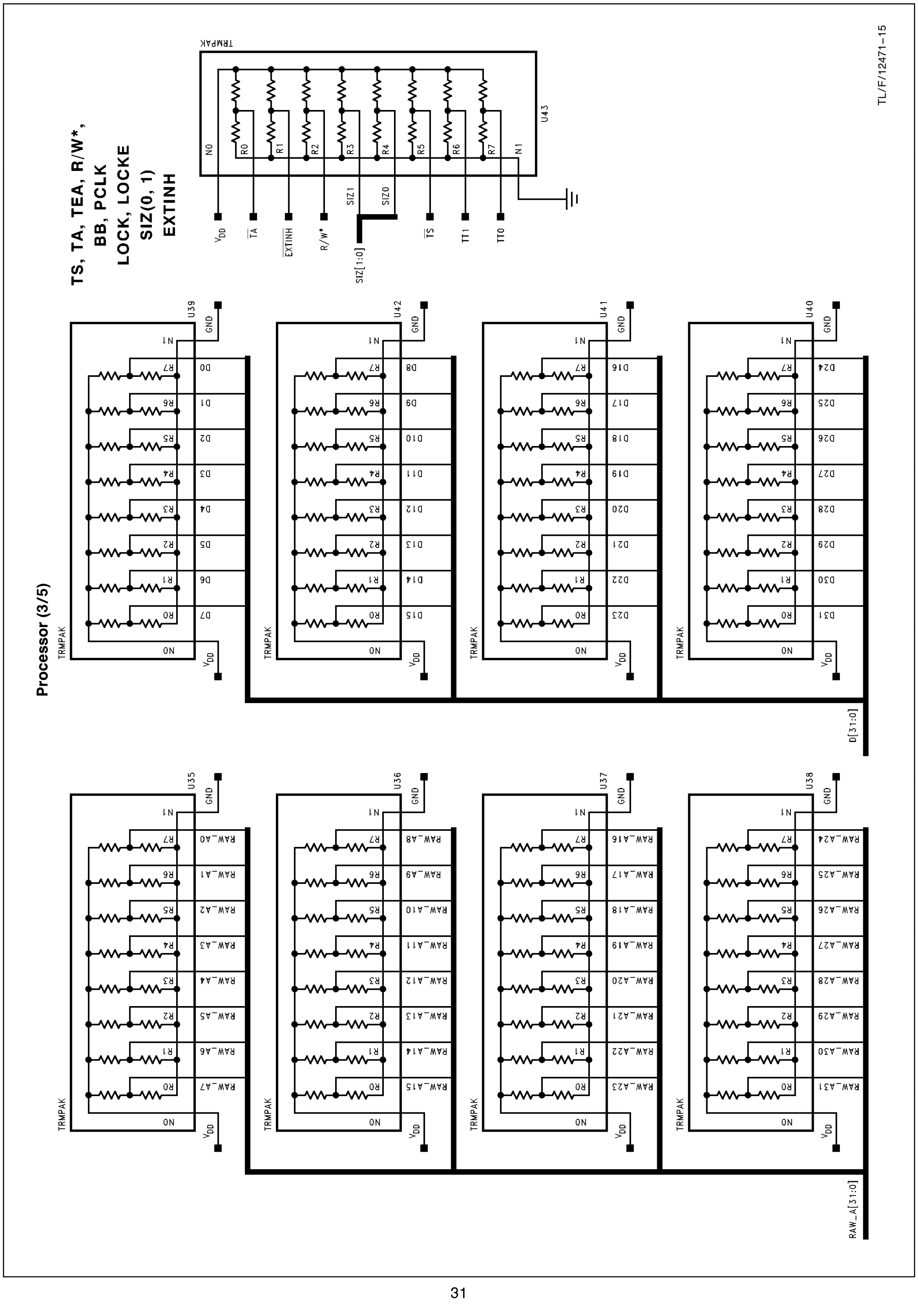


TL/F/12471–13

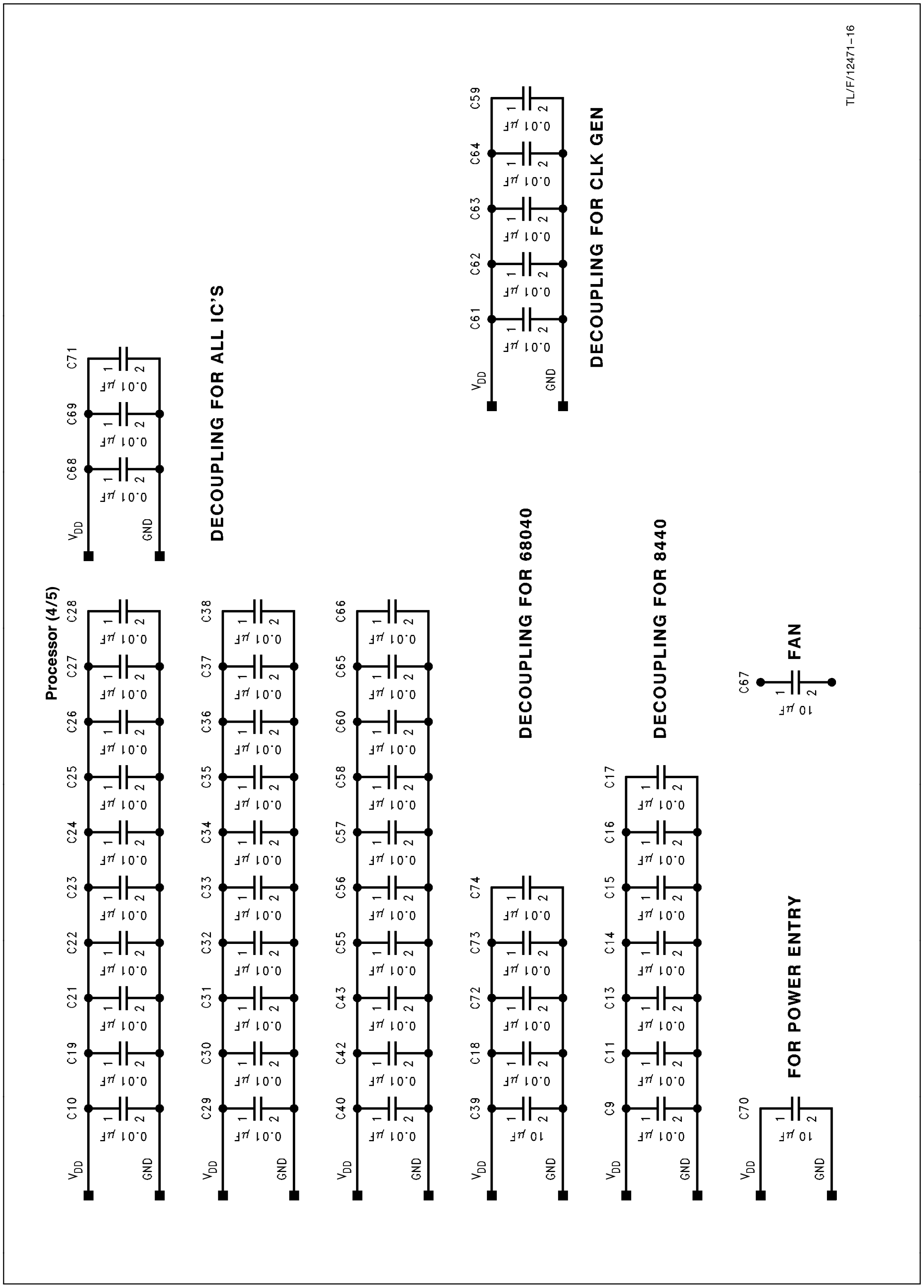
Processor (2/5)



TL/F/12471–14



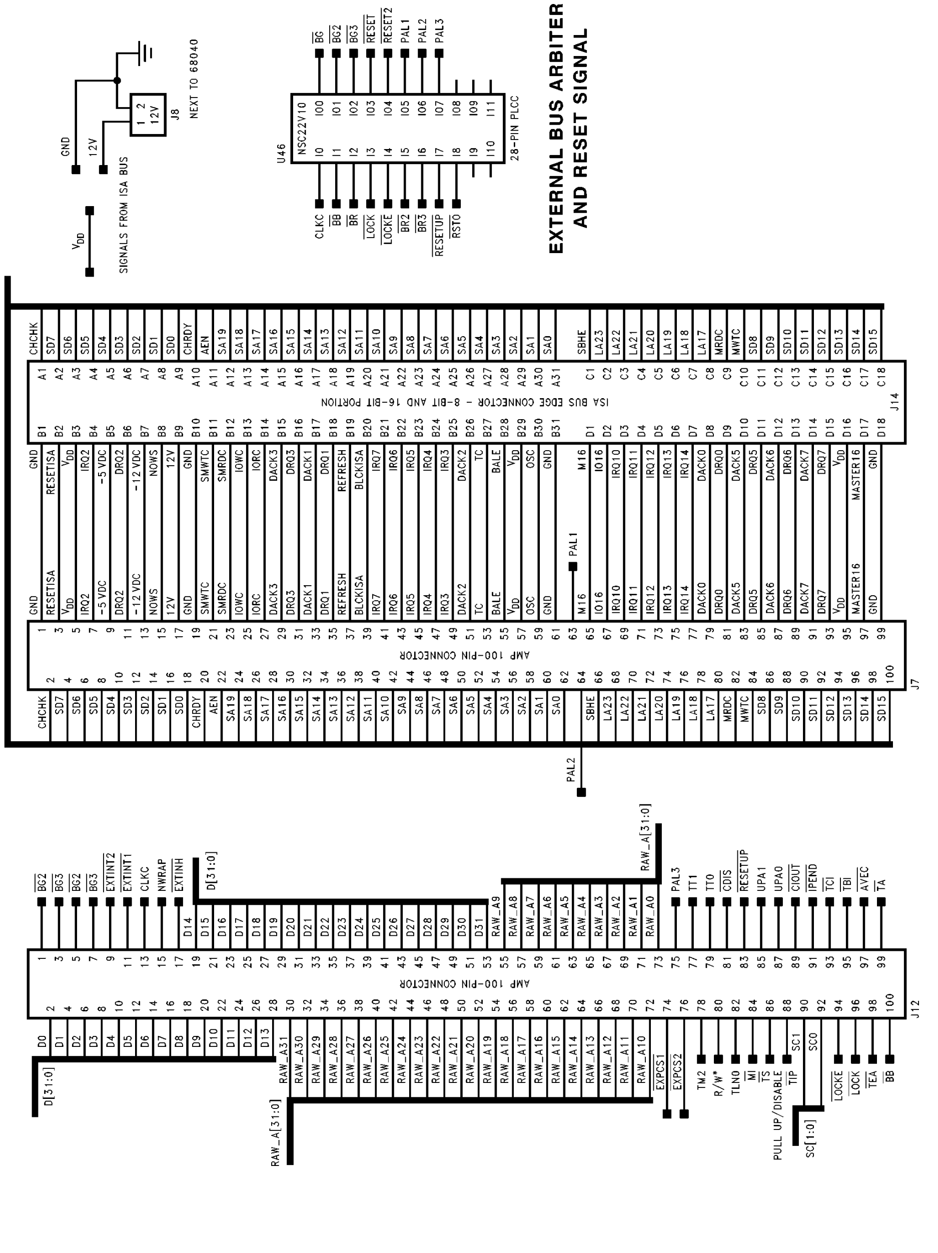
Processor (3/5)


Processor (4/5)
VDD
GND
C10
C19
C21
C22
C23
C24
C25
C26
C27
C28
C29
C30
C31
C32
C33
C34
C35
C36
C37
C38
C40
C42
C43
C55
C56
C57
C58
C60
C65
C66
0.01 µF
1
2
C68
C69
C71
DECOUPLING FOR ALL IC'S
C39
C18
C72
C73
C74
10 µF
DECOUPLING FOR 68040
C61
C62
C63
C64
C59
DECOUPLING FOR CLK GEN
C9
C11
C13
C14
C15
C16
C17
DECOUPLING FOR 8440
C67
FAN
C70
FOR POWER ENTRY

TL/F/12471-16

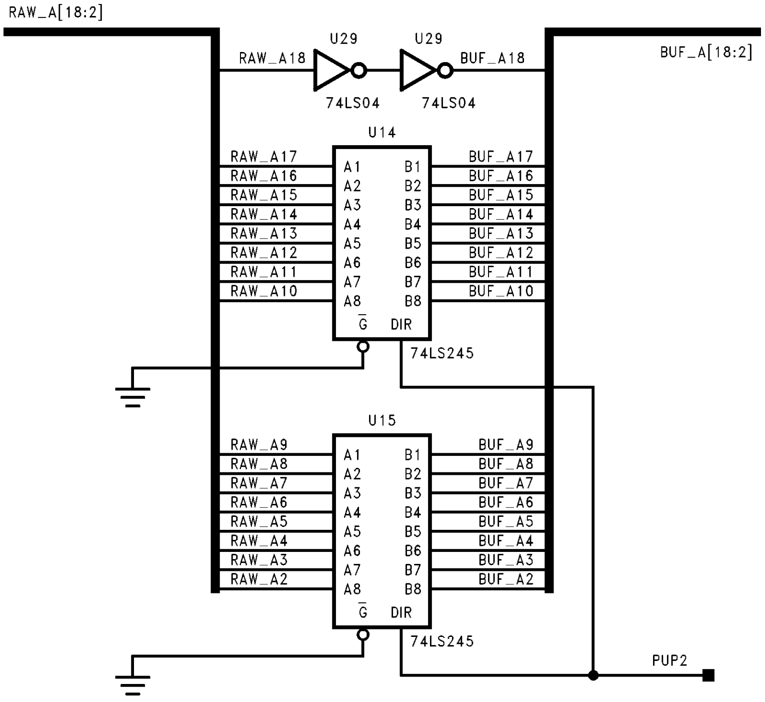
Processor (5/5)
CNNCTR and Arbiter

CHCHK, SD[15:0], CHRDY, AEN, SA[19:0], SBHE, LA[23:17], MRDC, MWTC, V_{DD}, GND, 12V

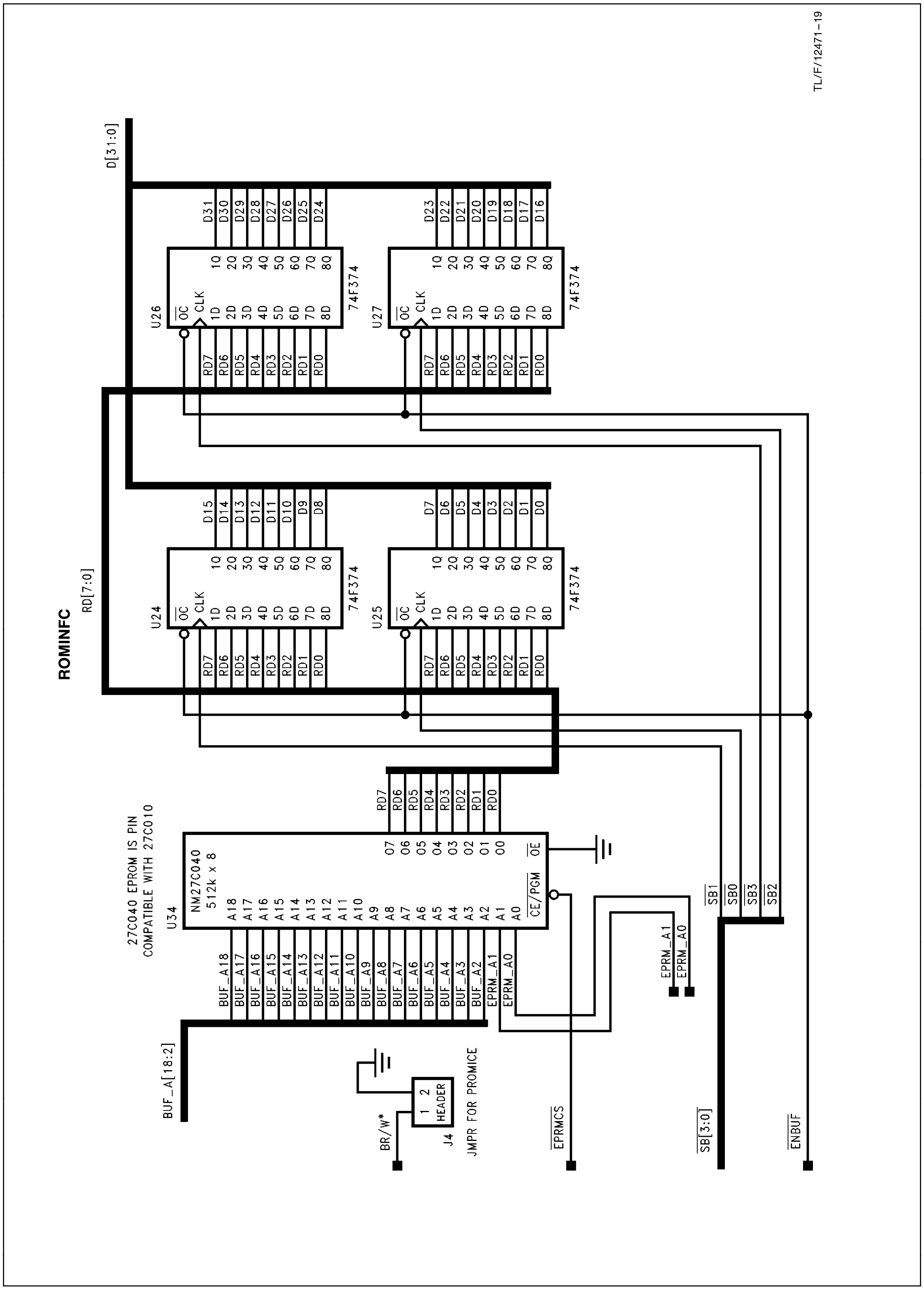


TL/F/12471–17

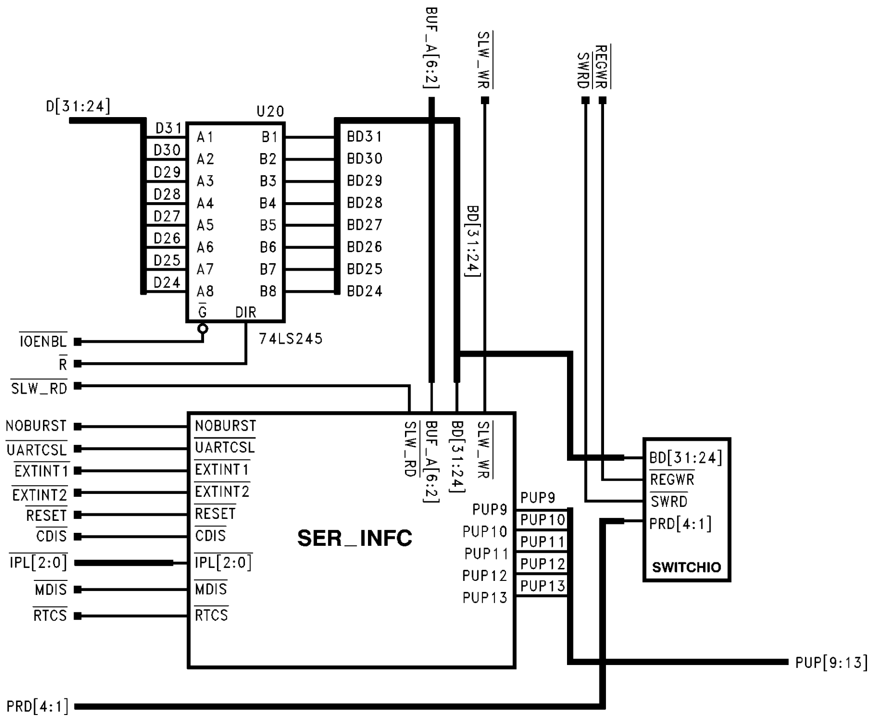
ADBUFR



TL/F/12471–18

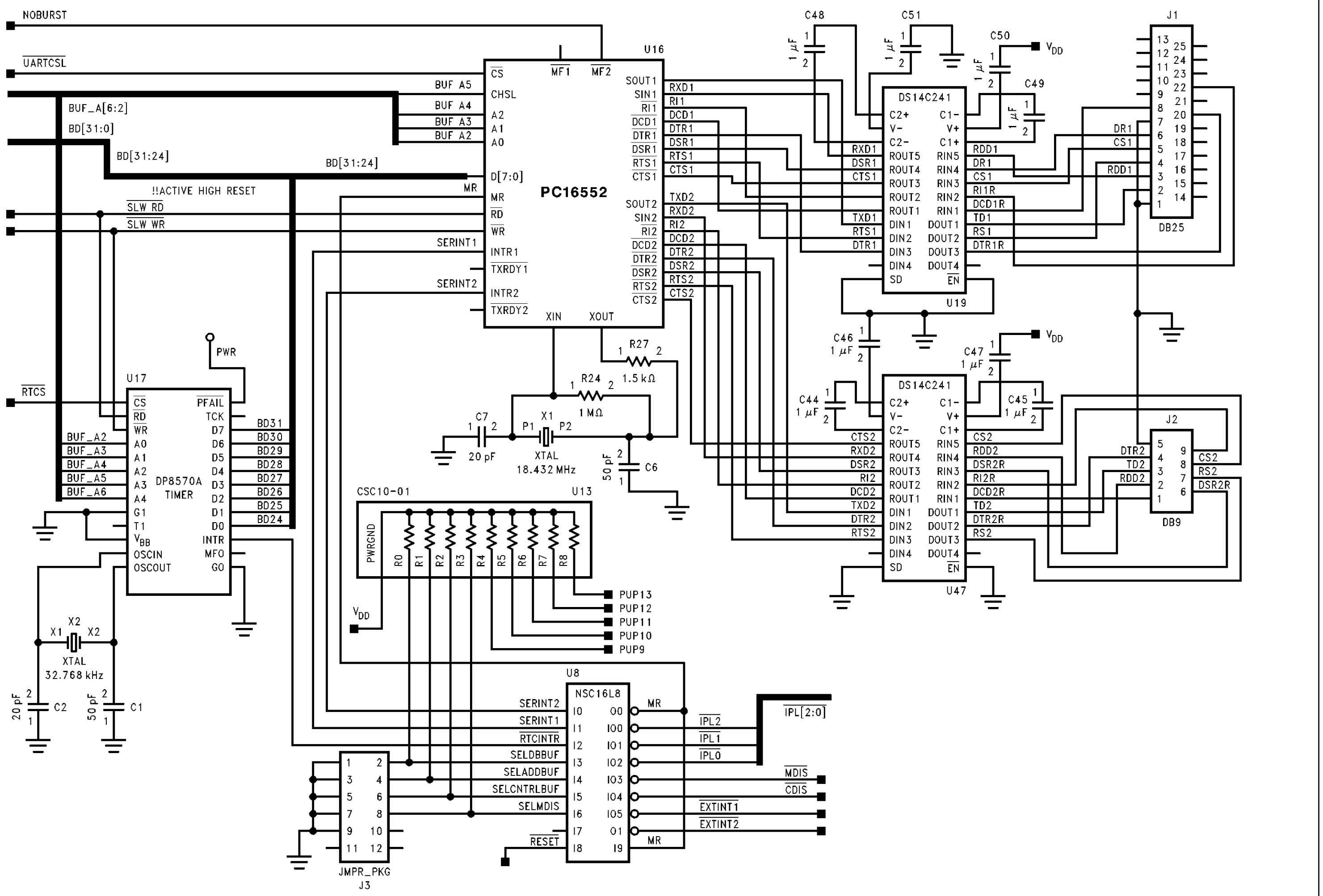

ROMINFC
RD[7:0]
D[31:0]
U26
U27
U24
U25
74F374
OC
CLK
1D 2D 3D 4D 5D 6D 7D 8D
1Q 2Q 3Q 4Q 5Q 6Q 7Q 8Q
RD7 RD6 RD5 RD4 RD3 RD2 RD1 RD0
D31 D30 D29 D28 D27 D26 D25 D24
D23 D22 D21 D20 D19 D18 D17 D16
D15 D14 D13 D12 D11 D10 D9 D8
D7 D6 D5 D4 D3 D2 D1 D0
27C040 EPROM IS PIN
COMPATIBLE WITH 27C010
U34
NM27C040
512k x 8
A18 A17 A16 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
O7 O6 O5 O4 O3 O2 O1 O0
OE
CE/PGM
BUF_A18 BUF_A17 BUF_A16 BUF_A15 BUF_A14 BUF_A13 BUF_A12 BUF_A11 BUF_A10 BUF_A9 BUF_A8 BUF_A7 BUF_A6 BUF_A5 BUF_A4 BUF_A3 BUF_A2
EPRM_A1
EPRM_A0
BUF_A[18:2]
BR/W*
1 2
HEADER
J4
JMPR FOR PROMICE
EPRMCS
SB1
SB0
SB3
SB2
SB[3:0]
ENBUF
TL/F/12471-19

IO



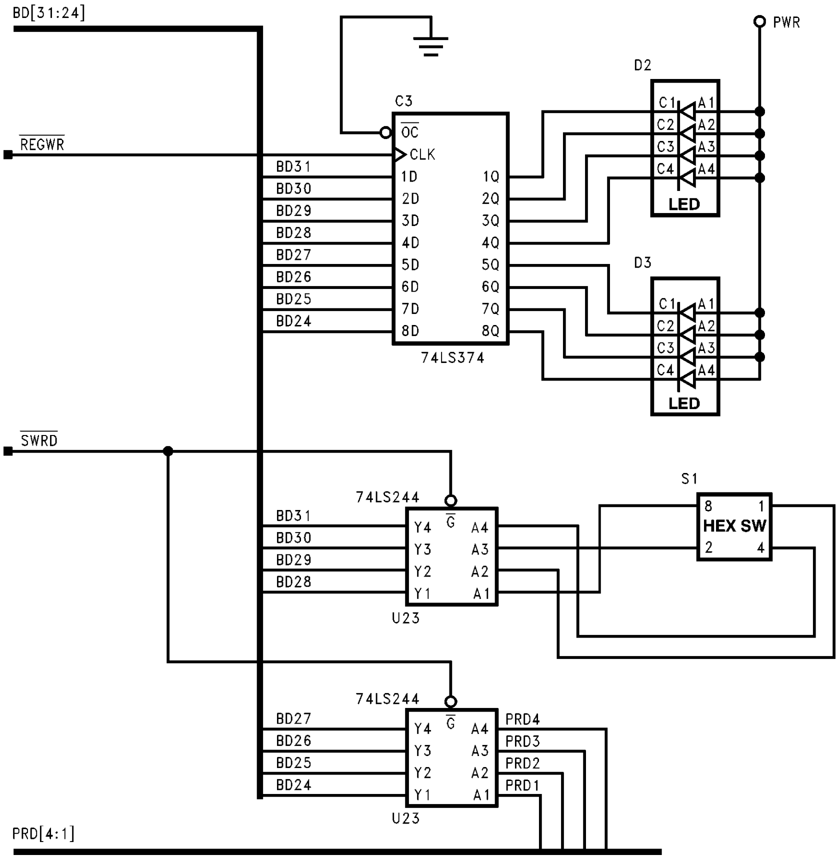
TL/F/12471–20

Serial INFC



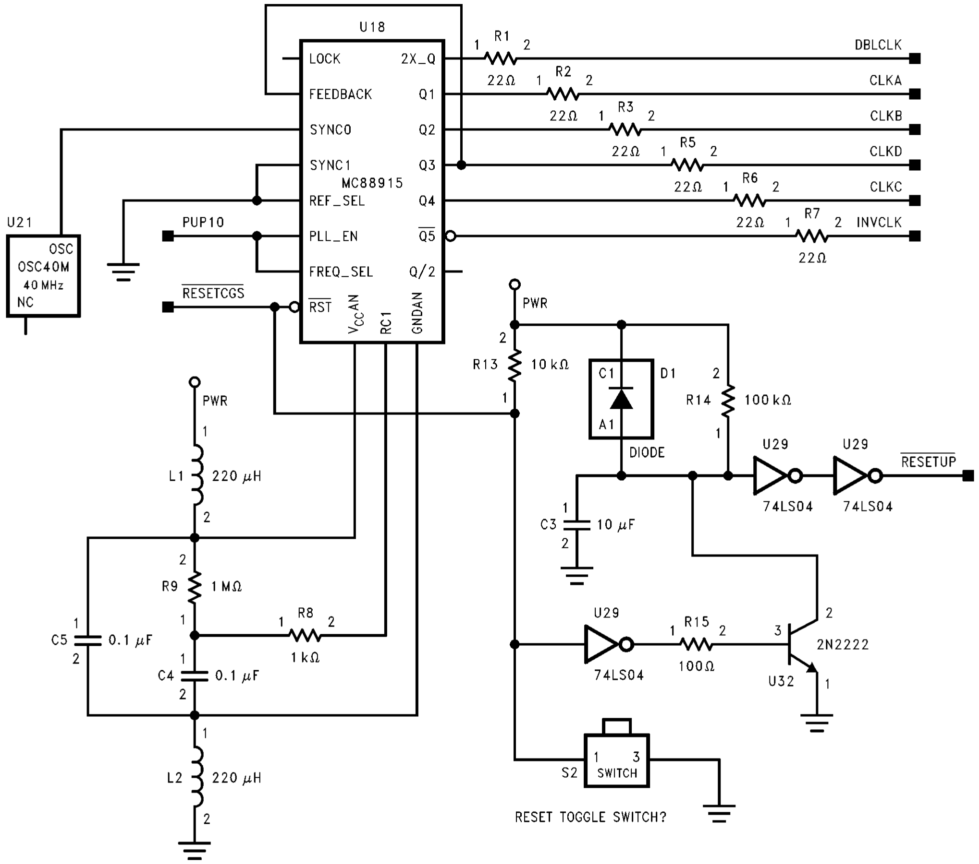
TL/F/12471–21

SWTCHIO



TL/F/12471–22

CLK__GEN



TL/F/12471–23

National's DP8440 DRAM Controller in a 40 MHz System

AN-999

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

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.