An Interface Between a 74S409 Dynamic RAM Controller and a 68000 CPU

Zwie Amitai, Glenn A. Baxter and Frank Lee

Dynamic RAMs take a leading place as a semiconductor volatile storage medium for microprocessor systems. Since dynamic RAMs have multiplexed row and column addresses special control signals are required to address a particular location in memory. Also, dynamic RAMs require a timely refresh to maintain the data stored in their cells. These special requirements bring about the need for a dynamic RAM controller which provides the control signals to access and refresh the memory.

An interface is needed between the dynamic RAM controller and the microprocessor to accommodate the specific access protocols and specific speeds of the microprocessor as well as the specific access and refresh timing of the dynamic RAMs. This interface may be implemented using Programmable Array Logic (PAL®) devices, as in the example described in this paper.

This paper presents an interface between an 8-MHz 68000 CPU and a 74S409-2 Dynamic RAM Controller/Driver driving 150-ns access time dynamic RAMs. Though the exact implementation may vary with other CPU-dynamic RAM combinations, this example may also serve as a guideline for other designs.

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2376 TWX: 910-338-2374
Monolithic Memories MMI

An Interface Between a 74S409 Dynamic RAM Controller and a 68000 CPU

Zwie Amitai, Glenn A. Baxter and Frank Lee

Dynamic Versus Static RAMS

Ever since the introduction of microprocessors, compactness has been an important issue in microcomputer design. Dynamic RAMs were introduced to reduce the chip count of a memory array as well as to reduce the chip "footprint"—the board area taken by the chip.

Static RAMs use a feedback amplifier for each bit of data to maintain the charges which represent the data. Several transistors are used for each feedback circuit, and therefore the density of static RAMs is limited by the transistor count per bit. A data bit in the dynamic RAM is stored in a capacitor and has to be regularly refreshed to compensate for charge leakage. There is only one transistor and one capacitor per bit, allowing for greater density as well as for reduced power consumption per data bit. With dynamic RAMs achieving 256K bits per chip, up to 18 address bits are required to access the data. Such a large number of address pins could result in package which takes a disproportionately large area on the printed circuit board. In order to reduce the chip footprint, dynamic RAMs are organized in matrix form, with row and column addresses multiplexed on common address lines, thus halving the number of physical address pins (see figure 1).

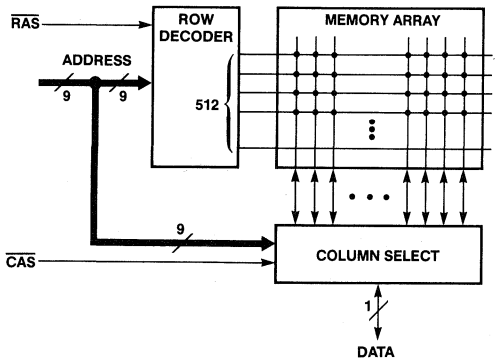


Figure 1. Dynamic RAM matrix

Special control signals are used to distinguish between row and column addresses. Dynamic RAMs need relatively elaborate control signals to orchestrate both access and refresh of stored data, and are slower than state-of-the-art static RAMs at the time of this paper's writing.

Memory Organization

A memory unit may have a number of pages (figure 2), each of which is independently selected by a chip select signal ($\overline{CSi}$). Inside a memory page there may be several banks, each holding a segment of the address space of the page. Most of the dynamic RAMs are bit sliced (i.e. each device holds one bit of data per address) and the number of devices needed to form a bank equals to the number of bits in a word of memory.

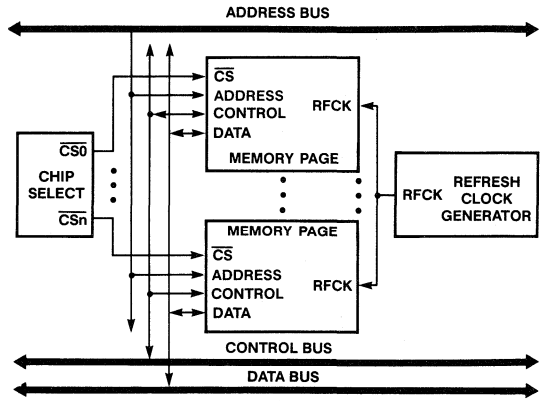


Figure 2. A memory array divided to pages

Dynamic RAM Access

A particular bit of data in the matrix is accessed in two steps (see figure 3). Firstly, the row address is presented, strobed, and latched into the row address decoder. One

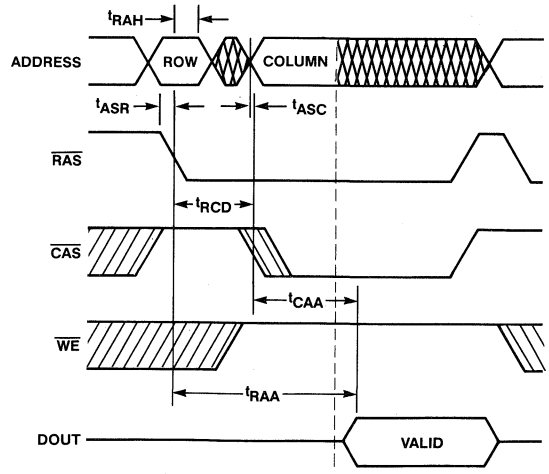


Figure 3. Read cycle timing diagram

row of the array will be activated, and all the data bits of this row will be available at the data latch. Secondly, the column address is strobed into the column decoder, and one particular column is selected. Data will be written into or read out of the particular location selected by the row and column addresses, depending on the state of the Write Enable ($\overline{WE}$) control signal. In all, three signals control the dynamic RAM: Row Address Strobe ($\overline{RAS}$), Column Address Strobe ($\overline{CAS}$), and Write Enable ($\overline{WE}$), all asserted LOW.

Refresh

In a dynamic RAM, data bits are stored as charges on capacitors. The stored charges leak slowly and have to be regularly refreshed. Refresh is accomplished by reading a bit of data and writing it back to the same location, thereby restoring the capacitor's charge. Taking advantage of its matrix organization, a dynamic RAM can be refreshed row by row; all rows must be refreshed typically once every 2 ms to maintain their data.

There are several methods for performing refresh. The most common method, which can be used for all commercially available DRAMs, requires the system to select one row of the memory array by presenting a row address at the address lines and toggling the Row Address Strobe ($\overline{RAS}$). The selected row will be activated and refreshed (see figure 4).

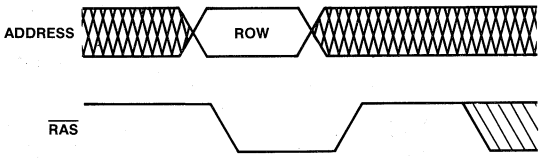


Figure 4. Refresh cycle timing

System designers use several refresh strategies for ensuring proper refresh while minimizing system slowdown. For example, a dynamic RAM organized in 128 rows has to be refreshed every 2 ms. One strategy would be to dedicate a "time slot" every 2 ms for refresh. All rows would be refreshed consecutively within this time slot, and all attempts to access the memory would be rejected. This strategy is called "Burst Refresh" and a slow-down in system performance would be expected since 3% of the time is used for refresh. "Distributed Refresh," on the other hand, is implemented by dividing the refresh interval of 2 ms into 128 refresh cycles, each of 15.6 us. One row is then refreshed each refresh cycle.

A refresh can be forced on the system in a particular time interval for both the distributed and the burst refresh. In this case, the system is denied access to the memory while refresh takes priority. A refresh can also be hidden—transparent to the system. Such a refresh takes advantage of specific knowledge about the access pattern of the system to avoid conflicts between access and refresh.

In order to perform a hidden refresh, one needs to anticipate "free" time intervals in which no memory access will be attempted by the system. Some systems have a "free" time interval every system cycle, such as the Instruction Decode in microprocessor-based systems. Performing a refresh on each system cycle is straightforward, but since dynamic RAMs consume power mainly when accessed or refreshed, overrefreshing will result in a higher power consumption. Therefore, the system designer should use only one "free" time interval per refresh cycle. For systems that do not have a "free" time interval for refresh in their system cycle, a hidden refresh is performed when the system is "looking the other way"—i.e. accessing a different page of memory or an I/O port. In such a scheme, a "Forced Refresh" backup is needed for those occasions of long continuous access of the same memory space.

Dynamic RAM Timing Parameters

Because of the multiplexed address and the two separate control signals ($\overline{RAS}$ and $\overline{CAS}$), several timing requirements must be satisfied to ensure proper operation of the dynamic RAM. Some of these timing requirements are (see figure 3):

1. t_{ASR}—Row address setup time.
   The row address must be stable t_{ASR} before $\overline{RAS}$ is asserted LOW.

2. t_{RAH}—Row address hold time.
   The row address must be held t_{RAH} after $\overline{RAS}$ has been asserted LOW.

3. t_{ASC}—Column address setup time.
   The column address must be stable t_{ASC} before $\overline{CAS}$ is asserted LOW.

4. t_{RCDL}—$\overline{RAS}$ to $\overline{CAS}$ delay.
   The $\overline{RAS}$ to $\overline{CAS}$ delay t_{RCDL} must exceed the specified *minimum* to ensure proper operation. If the $\overline{RAS}$ to $\overline{CAS}$ delay does not exceed the specified *maximum,* then the access time from $\overline{RAS}$ will be t_{RAC} from $\overline{RAS}$ LOW, with t_{RAC} specified for the particular dynamic RAM. If the $\overline{RAS}$ to $\overline{CAS}$ delay does exceed the specified *maximum,* then the access time will be the sum of the $\overline{RAS}$ to $\overline{CAS}$ delay and t_{CAC} measured from $\overline{CAS}$ LOW.

5. t_{RAS}—$\overline{RAS}$ active period.
   $\overline{RAS}$ must be LOW for t_{RAS} or more for the access or refresh cycle to be accomplished.

6. t_{RP}—$\overline{RAS}$ precharge time
   $\overline{RAS}$ must be HIGH t_{RP} or more between consecutive cycles (refresh or access).

7. t_{CP}—$\overline{CAS}$ precharge time
   $\overline{CAS}$ must be HIGH t_{CP} before $\overline{RAS}$ can be asserted LOW for a new access or refresh cycle.

10

Dynamic RAM Controller

Special circuitry is needed to control the access and refresh of the dynamic RAMs. The functions needed from a dynamic RAM controller are:

1. Multiplexing row, column, and refresh addresses.
2. Providing refresh addresses.
3. Providing refresh timing.
4. Arbitrating between access and refresh.
5. Providing access timing.
6. Driving the DRAMs' capacitive-load inputs.
7. Interfacing to the system.

MODE	FUNCTIONS
0	Externally controlled refresh
1	Automatic forced refresh
2	Internal automatic burst refresh
3a	All-$\overline{RAS}$ automatic write
3b	Externally controlled All-$\overline{RAS}$ access
4	Externally controlled access
5	Automatic access with hidden refresh
6	Fast automatic access
7	Set end of count

Table 1. Functions of different modes of SN74S409

The 74S409 Dynamic RAM Controller/Driver

The 74S409 Dynamic RAM Controller/Driver performs all the functions needed to access and refresh the dynamic RAM, and its eight operating modes can accommodate several access and refresh schemes. The design described in this application note takes advantage of the automatic access and the hidden refresh capabilities of the 74S409. The following issues are important for a dynamic RAM controller and require further discussion:

1. Addressing capability

The maximum size of the memory space controlled by a single DRAM controller depends on the number of address lines which it can provide and the number of banks it can select. Also, the addressing capability may be limited by the dynamic RAM controller driving capability. The 74S409 can address dynamic RAMs that have 9 or less address pins, (i.e. can address up to 256K-bit DRAMs) and it can select 4 banks with its 4 $\overline{RAS}$ outputs.

2. Driving capability

Dynamic RAM inputs are capacitive loads. The number of dynamic RAMs that can be addressed and controlled directly by a dynamic RAM controller depends on its driving capability. The 74S409 can drive up to 88 dynamic RAMs organized in 4 banks, 22 dynamic RAMs per bank (16-bit wide word with 6 check bits for error detection and correction).

	MODE 5		MODE 6	
	t_{RCDL}(max)	t_{RAH}(min)	t_{RCDL}(max)	t_{RAH}(min)
409/8	125	30	105	20
409/8-2	100	20	85	12
409/8-3	145	30	120	20

Table 2. 74S409/8 timing options

3. Timing parameters

The dynamic RAM controller can provide the required delays between the signals going to the dynamic RAMs (automatic access) or can allow the system to control those delays (externally controlled access). The 74S409 has three speed versions and two speed options of the automatic access mode providing six combinations of t_{RCDL} and t_{RAH} while satisfying both t_{ASR} and t_{ASC} setup times for most dynamic RAMs (see table 2). Nevertheless, the externally controlled mode allows the system designer to meet other t_{RCDL}/t_{RAH} combinations and support special access and refresh modes such as page mode, nibble mode and others.

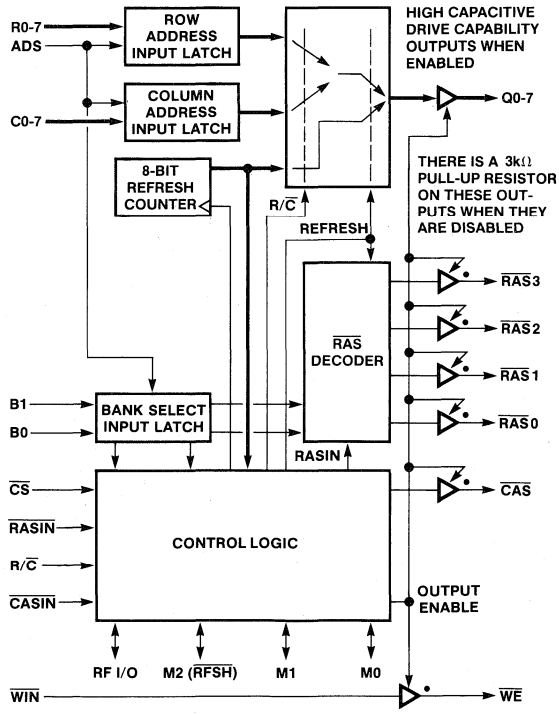


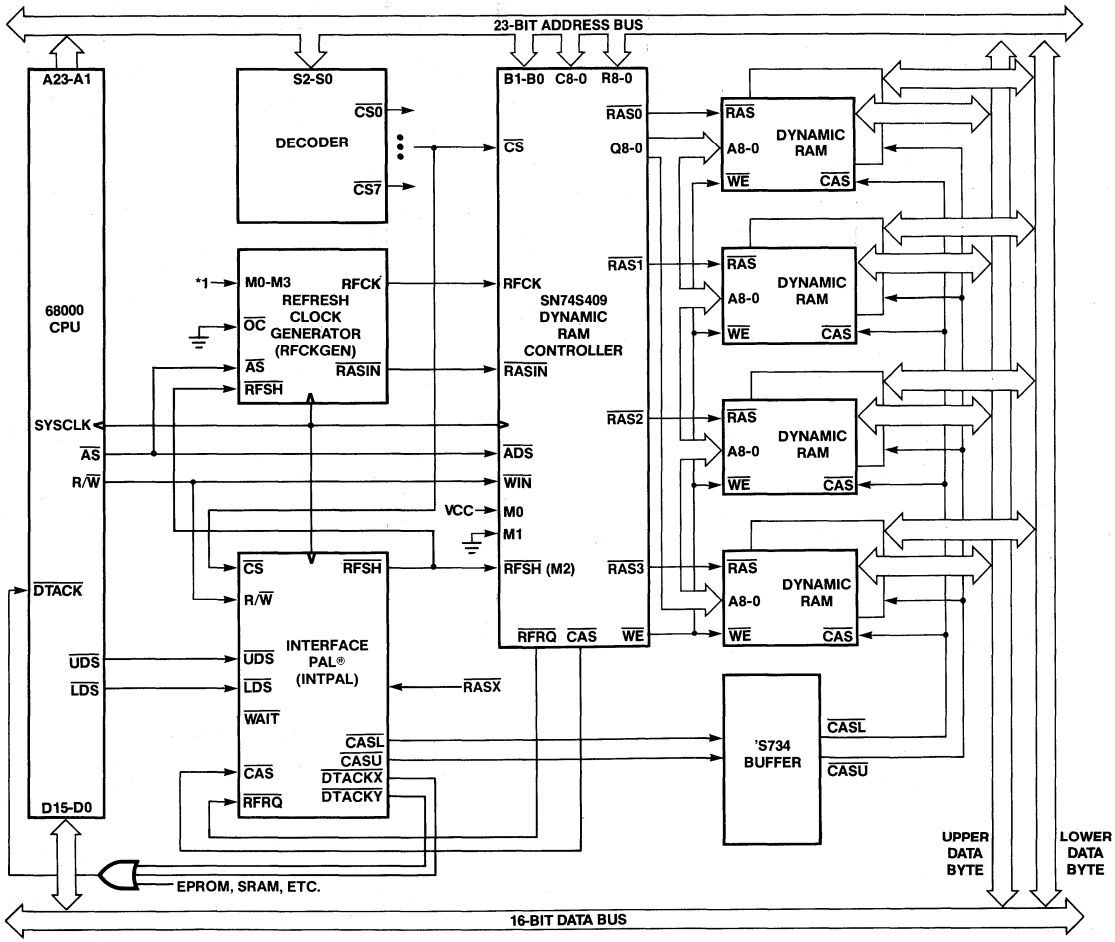
Figure 5. Block diagram of a 74S409 dynamic RAM controller

Interfacing a 68000 CPU and a Dynamic Memory Array using the 74S409-2 Dynamic RAM Controller/Driver

The design described in this paper interfaces between an 8 MHz 68000 CPU and a 74S409-2 Dynamic RAM Controller/Driver driving 150 ns (access time) dynamic RAMs. Nevertheless, parts of the design, and the design guidelines can be used for other CPU-dynamic RAM speed combinations. The interface is implemented using Programmable Array Logic (PAL®) devices, as shown in figure 6. The 74S409-2 operates in its automatic access mode (mode 5) with hidden refresh. The backup refresh is the automatic forced refresh (mode 1). The circuit presented in figure 6 was built by and tested at Monolithic Memories at room temperature at frequencies up to 12.5 MHz. The write cycle is accomplished with no wait-states, while the read cycle requires one wait-cycle (two wait-states).

A 68000 Basic System

A basic 68000 system (see Appendix) was built to support the development of the interface. The basic system includes an 8 MHz 68000 CPU, static RAM, EPROM, and associated logic. The two-stage decoding scheme, implemented by two 74F138 devices, allows the selection of a memory page of 128K words in 8 ns (one decoding level), and the selection of an 8K section in 16 ns (two decoding levels). A PAL device is used to



*1. TIED TO VCC OR GND DEPENDING ON THE REFRESH CLOCK PERIOD CHOSEN (SEE PAL SPECIFICATIONS FOR RFCKGEN).

Figure 6. An interface circuit between a 68000 CPU and a 74S409 dynamic RAM controller

provide the handshake signals to the 68000 CPU for both static RAM and EPROM. The EPROM read cycle and the static RAM write cycle are accomplished with one wait cycle (i.e. two wait-states); the static RAM read cycle requires no wait-state. The PAL specifications are listed in appendix 3.

The Interface

The interface to the dynamic RAM includes the transceivers and the decoding circuit, which serve the rest of the system as well, a Refresh Clock Generator (RFCKGEN), which divides the system clock to provide a clock for the hidden and forced refresh, and a circuit which provides the controls for the 74S409-2 as well as handshake signals for the CPU (INTPAL). Refresh generation, 74S409-2 control, and CPU handshake are all implemented in two PAL® devices. An equivalent block diagram of both circuits is shown in figure 7.

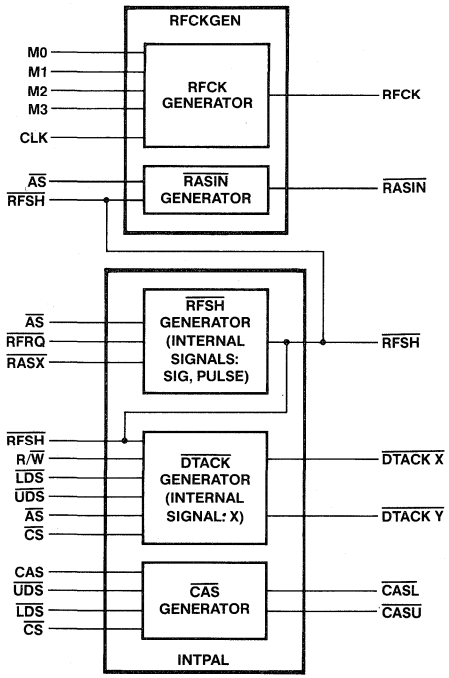


Figure 7. An equivalent block diagram for RFCKGEN and INTPAL.

Operation:

RASIN Generation

RASIN is strobed by either AS or the RFSH signal coming out of INTPAL.

Refresh Clock (RFCK)

The RFCK LOW time can be selected to allow for the longest interval needed for the forced refresh. The forced refresh interval must include the time needed for the system to complete an access cycle, the time needed for the PAL® device to switch from mode 5 (access) to mode 1 (refresh), the time needed for the actual refresh cycle, and the RAS precharge time.

RFSH Generation

When RFCK goes HIGH and no hidden refresh was performed during the RFCK LOW time, the 74S409-2 generates a refresh request (RFRQ) to signal to the system that a refresh is needed. Once AS is HIGH (i.e. no access cycle is attempted) the RFSH signal goes LOW and switches the 74S409-2 mode of operation from access to refresh. The RASIN is asserted after a delay. The internal signal SIG is generated when both RAS and RFSH are LOW and is used to extend the RFSH LOW duration for one cycle after RASIN has gone HIGH, thus delaying a pending access for one more cycle in order to satisfy RAS precharge requirements.

CAS Generation

Since the 68000 CPU has individual byte access capability, the CAS signal coming out of the 74S409-2 must be split into two signals: CASU and CASL. CASU and CASL are enabled by their respective data strobe UDS and LDS.

DTACK Generation

Two DTACK signals come out of the INTPAL: DTACKX and DTACKY. DTACKX acknowledges a read cycle and is generated so one wait cycle (two wait-states) is inserted in the memory cycle. DTACKY acknowledges a write cycle and allows no wait-states operation. The two DTACKs must be ANDed with the other DTACKs to create the 68000 DTACK signal.

Increasing the Efficiency of Dynamic RAM Hidden Refresh

One way to maximize the efficiency of a memory system is to maximize the chance of hidden refresh, thereby minimizing the number of incidents in which the system's memory access is delayed because a refresh is taking place. Hidden refresh is performed only when the system is accessing a different memory page and is not accessing this particular controller. Note that many memory accesses are sequential or at least are local, i.e. the same memory page tends to be accessed continuously for a duration of time. In such cases the dynamic RAM controller may not be able to steal any memory cycles for the hidden refresh. One way to solve this problem is to have continuous addresses distributed over different controllers by using the least significant bits of the address to select memory pages and their corresponding memory controllers.

Appendix 1. Timing Analysis

68000 Read Cycle Timing

The 68000 read cycle starts when $\overline{AS}$ is asserted LOW. If data is available no later than t_{DICL} before the negative-going clock edge of S6, then no-wait-states operation can be achieved by asserting $\overline{DTACK}$ no later than t_{ASI} before the negative-going clock edge of S4. The $\overline{AS}$ to data timing for various clock frequencies can be calculated from the equation (see table 1.1):

$\overline{AS}$ to data in valid = $2.5T - t_{DICL} - t_{CHSLn}$
(for a read cycle with no wait-states)

Where: t_{CHSLn} — Clock HIGH to $\overline{AS}$ LOW (max)
T — System clock cycle time
t_{DICL} — Data-in to clock LOW (setup time)

When the memory cycle cannot be completed within the $\overline{AS}$ LOW to data-in valid interval, wait-states are introduced to increase the available access time in increments of T (clock cycle) corresponding to minimum increments of two wait states. The introduction of wait-states is achieved by holding $\overline{DTACK}$ HIGH until after the the negative-going clock edge of S4.

68000 Write Cycle Timing

During the write cycle, data becomes available from the 68000 at:

1. t_{RLDO} after R/$\overline{W}$ goes LOW
2. t_{DOSL} before $\overline{UDS}$, or $\overline{LDS}$, or both go LOW
3. t_{CLDO} after the negative-going clock edge of S2
4. $T + t_{CLDO} - t_{CHSL}$ after $\overline{AS}$ goes LOW.

The data stays available t_{CHDO} after the positive-going clock edge of S7 and t_{SHDO} after $\overline{AS}$ and the asserted $\overline{DS}$ go HIGH. The data is therefore available until $3T - t_{CHSLx} + t_{CHDO}$ from $\overline{AS}$ going LOW (see table 1.2).

Frequency	$\overline{AS}$ LOW to data out valid Min	Max	R/$\overline{W}$ LOW to data out valid Min	data out to $\overline{DS}$ LOW Max	data out held from $\overline{DS}$, $\overline{AS}$ HIGH Min	data out held from $\overline{AS}$ LOW Min	data out valid Min
4	260	340	55	55	60	670	330
6	177	247	35	35	40	431	184
8	135	195	30	30	30	315	120
10	100	155	20	20	20	245	90
12.5	80	135	10	15	15	185	50

Table 1.2 Data out timing relations to R/$\overline{W}$, $\overline{DS}$x, and $\overline{AS}$.

Frequency	$\overline{AS}$ LOW to data (min)
4	515 ns
6	332.5 ns
8	237.5 ns
10	185 ns
12.5	135 ns

Table 1.1 $\overline{AS}$ LOW to data valid for read cycle

Dynamic Memory Array Timing

Access Timing

When calculating the time needed for the dynamic RAM array to complete an access cycle, one must add the delays introduced by the buffers between the system and the 68000 device, by the PAL® devices that gate the signals to the dynamic RAM controller, by the dynamic RAM controller, and by the CAS generator which includes a PAL® and a 74S734 MOS driver. Figure 1.1 illustrates the delays involved in a write cycle and a read cycle for an array of 150 ns access-time dynamic RAMs. The timing analysis is done for Hitachi's HM256-15 dynamic RAM , but most critical parameters are same or very similar for all other 150 ns access time dynamic RAMs.

For the circuit described in this application $\overline{AS}$ LOW to $\overline{RAS}$ LOW is 68 ns, $\overline{AS}$ LOW to $\overline{CAS}$ LOW is 215 ns, and $\overline{AS}$ LOW to data-in valid (read) is 283 ns. Since an 8 MHz CPU is used, there are no wait-states for the write cycle and one wait-cycle (two wait-states) for the read cycle.

10

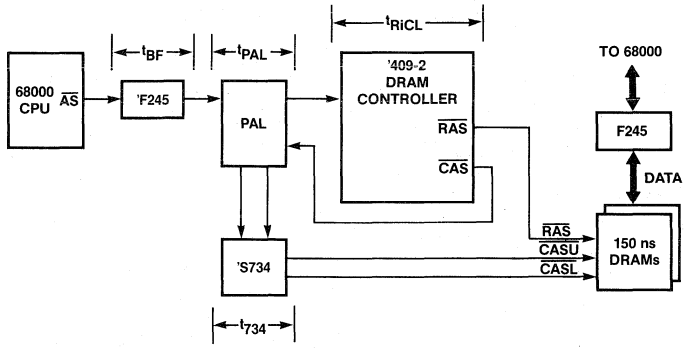


Figure 1.1 Read and write cycle delays

Precharge Timing

The dynamic RAMs require that both $\overline{RAS}$ and $\overline{CAS}$ stay deasserted (HIGH) for specific durations of time in order to allow the dynamic RAM internal circuitry to "precharge." The $\overline{RAS}$ precharge time is longer than the $\overline{CAS}$ precharge time and requires special attention. Typical $\overline{RAS}$ and $\overline{CAS}$ precharge times are shown in table 1.3.

Access Time (ns)	100	120	150	200
RAS precharge time ($\overline{RAS}$ high to $\overline{RAS}$ low)	80	90	100	120
$\overline{CAS}$ to $\overline{RAS}$ precharge ($\overline{CAS}$ high to $\overline{RAS}$ low)	10	10	10	10

Table 1.3
$\overline{RAS}$ and $\overline{CAS}$ precharge times as a function of access time.

When the 68000 $\overline{AS}$ output is connected through a buffer and a PAL® device into the $\overline{RASIN}$ input of the 74S409 dynamic RAM controller, as in the circuit described in this application, the $\overline{RAS}$ precharge time available for the dynamic RAM can be calculated from the following equation:

$$\overline{RAS}\text{ HIGH time} = t_{SH}(\text{min}) - \text{Max}(\overline{AS}\text{ LOW to }\overline{RAS}\text{ LOW delay}) + \text{Min}(\overline{AS}\text{ HIGH to }\overline{RAS}\text{ HIGH delay})$$

where t_{SH} is $\overline{AS}$ HIGH time and the $\overline{AS}$ to $\overline{RAS}$ delay consists of the buffer delay, PAL® device delay, and the 74S409 delay.

For the circuit described in this application, the $\overline{RAS}$ HIGH time exceeds the $\overline{RAS}$ precharge requirements of the 150 ns access time dynamic RAM. Whenever the $\overline{RAS}$ precharge time cannot be satisfied by the described circuit, $\overline{RAS}$ must be terminated as early as possible while holding $\overline{CAS}$ LOW. This scheme extends the time in which the data is available on the data bus by extending $\overline{CAS}$ LOW time, satisfying the $\overline{RAS}$ precharge time. The $\overline{CAS}$ HIGH to $\overline{RAS}$ LOW precharge time is much shorter, and is usually satisfied if the $\overline{CAS}$ is pulled HIGH by the positive-going edge of $\overline{AS}$.

Appendix 2: Using Externally Controlled Access (Mode 4) to Improve System Performance

The $\overline{RASIN}$ to $\overline{CAS}$ delay introduced by the 74S409 can be improved if externally controlled access (mode 4) is used instead of the automatic access (mode 5). The externally controlled access mode does not provide the hidden refresh capability, but allows the system designer to achieve shorter access times. Whenever the access time improvement allows the system to operate with less wait states than the automatic access configuration, the benefit from mode 4's tighter timing surpasses the performance improvement of the hidden refresh. The hidden refresh can also be implemented outside of the controller with some chip-count penalty. Operation with mode 4 requires two delay lines to provide the $\overline{RASIN}$-to-R/$\overline{C}$ delay and the R/$\overline{C}$-to-$\overline{CASIN}$ delay (see figure 2.1).

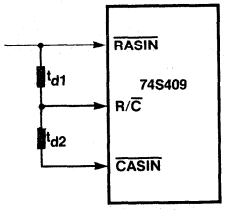


Figure 2.1. Using Two Delay Lines to Control 74S409 Output Timing in Mode 4.

In order to achieve a short access time using fast dynamic RAMs, two new parameters were specified and added to the SN74S409 data sheet. These two parameters allow the system designer to better control the critical delays relevant to the dynamic RAMs, namely the t_{RAH} (Row Address hold time) and the t_{ASC} (column address set-up to $\overline{CAS}$), thereby reducing the access time.

These parameters are:

	MIN	TYP	MAX
$t_{DIF1} = t_{RPDL} - t_{RHA}$			15 ns
$t_{DIF2} = t_{RCC} - t_{CPDL}$			15 ns

Where:

t_{RPDL} = $\overline{RASIN}$ to $\overline{RAS}$ LOW delay
t_{RHA} = row address held from column select
t_{RCC} = column select to column address
t_{CPDL} = $\overline{CASIN}$ LOW to $\overline{CAS}$ LOW delay

The system designer can now choose the delay between $\overline{RASIN}$ and R/$\overline{C}$, and the delay between R/$\overline{C}$ and $\overline{CASIN}$ using the procedure demonstrated in the following examples.

Calculating $\overline{RASIN}$ to R/$\overline{C}$ Delay (t_{d1})

t_{d1}(min) = $\overline{RASIN}$ LOW to R/$\overline{C}$ LOW delay (min)
$= t_{DIF1}$(max) + t_{RAH}(min)

Where:

t_{RAH} = Row Address hold time (Dynamic RAM parameter)

Calculating R/$\overline{C}$ to $\overline{CASIN}$ Delay (t_{d2})

t_{d2}(min) = R/$\overline{C}$ LOW to $\overline{CASIN}$ LOW delay (min)
$= t_{DIF2}$(max) + t_{ASC}(min)

Where:

t_{ASC} = Column address set-up time (Dynamic RAM parameter)

The resulting $\overline{RASIN}$ LOW to $\overline{CAS}$ LOW delay can be calculated by:

$\overline{RASIN}$ LOW to $\overline{CAS}$ LOW (max) = t_{d1}(max) + t_{d2}(max) + t_{cpdl}(max)

10

Example 1

Dynamic RAM: INMOS 2600-12 (120 ns access time)

Controller: SN74S409-2

$\overline{CAS}$ signal connected directly to the dynamic RAM.

Dynamic RAM critical parameters:

t_{RAH} = 12 (min)
t_{RCD} = 17 (min), 40 (max)
t_{ASC} = −5 (min)

Note that CAS can go low as soon as 5 ns before the column address is stable.

t_{d1}(min) = $\overline{RASIN}$ LOW to R/$\overline{C}$ LOW delay (min) = 13 + 12 = 25 (min)

t_{d2}(min) = R/$\overline{C}$ LOW to $\overline{CASIN}$ LOW delay (min) = 13 − 5 = 10 (min)

$\overline{RASIN}$ LOW to $\overline{CAS}$ LOW (max)
$= t_{d1} + t_{d2} + t_{cpdl}$(max)
= 25 + 10 + 58 = 93 (max)

Example 2

Dynamic RAM : HM256-15 (150 ns access time)

Controller : SN74S409-2

(With $\overline{CAS}$ splitting scheme using a PAL® device and 'S374 MOS driver)

Dynamic RAM critical parameters:

$t_{RAH} = 15$ (min)
$t_{RCD} = 25$ (min), 75 (max)
$t_{ASC} = 0$ (min)

PAL® delay: 25 ns(max), 0 ns(min)
'S734 delay : 30 ns(max), 8 ns(min) (at 250 pF load)

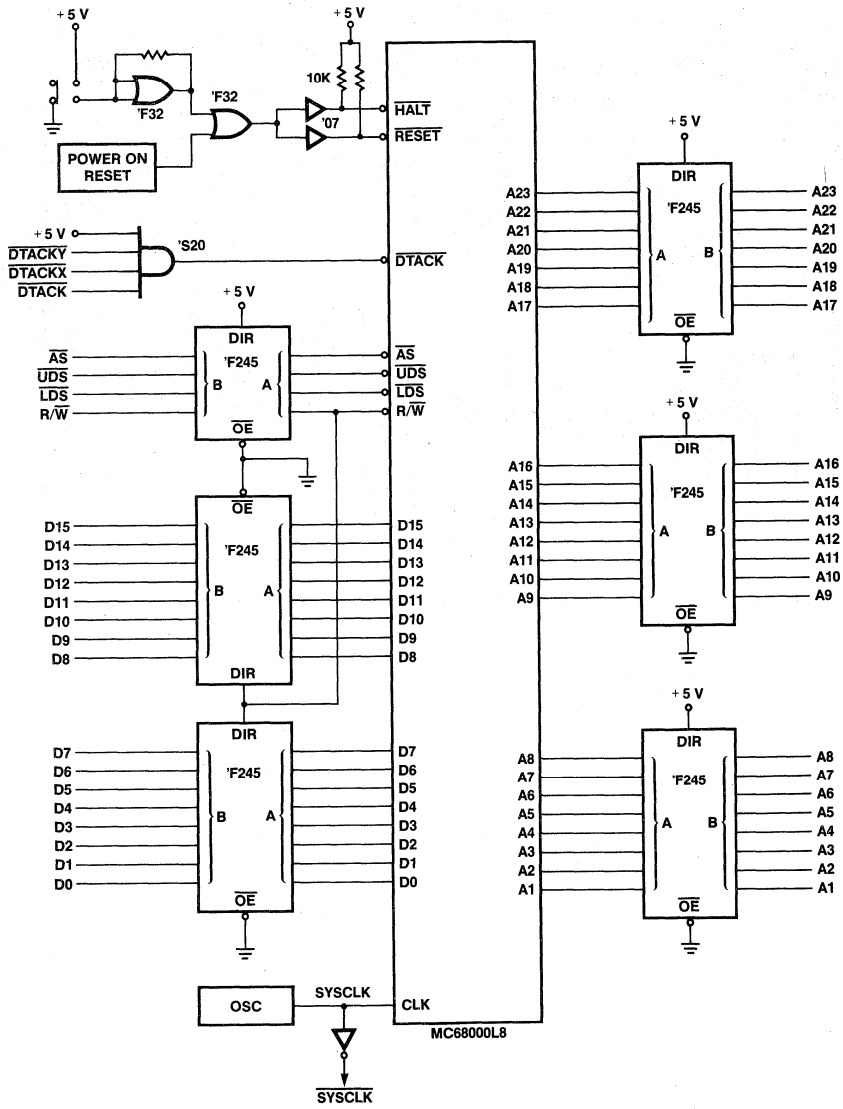
t_{d1}(min) = $\overline{RASIN}$ LOW to R/$\overline{C}$ LOW delay (min)
$= t_{DIF1}(max) + t_{RAH}(min)$
$= 15 + 15 = 30(min)$

t_{d2}(min) = R/$\overline{C}$ LOW to $\overline{CASIN}$ LOW delay (min)
$= t_{DIF2}(max) - t_{PAL}(min) - t_{374}(min) + t_{ASC}(min)$
$= 15 - 0 - 8 + 0 = 7(min)$

$\overline{RASIN}$ LOW to $\overline{CAS}$ LOW (max) = $t_{d1}(max) +$
$t_{d2}(max) + t_{cpdl}(max) + t_{PAL}(max) + t_{374}(max)$
$= 30 + 7 + 58 + 25 + 30 = 150$ (max)

A 35 ns improvement in the $\overline{RASIN}$ to $\overline{CAS}$ delay is achieved for the configuration described above. In this particular application, this improvement does not eliminate the need for two wait-states, but using faster dynamic RAMs and a faster $\overline{CAS}$ splitting scheme can further reduce the $\overline{AS}$ to data time (read) by more than 35 ns thus eliminating the need for wait-states.

Appendix 3: Circuit Schematics and PAL® Specifications



NOTES: 1. Use 'F' parts where stated
2. BGACK, BR, VPA, BERA, IPLO-2 all tied to VCC via 10K Ω resistor

Figure 3.1 68000-409 INTERFACE CIRCUIT PART I

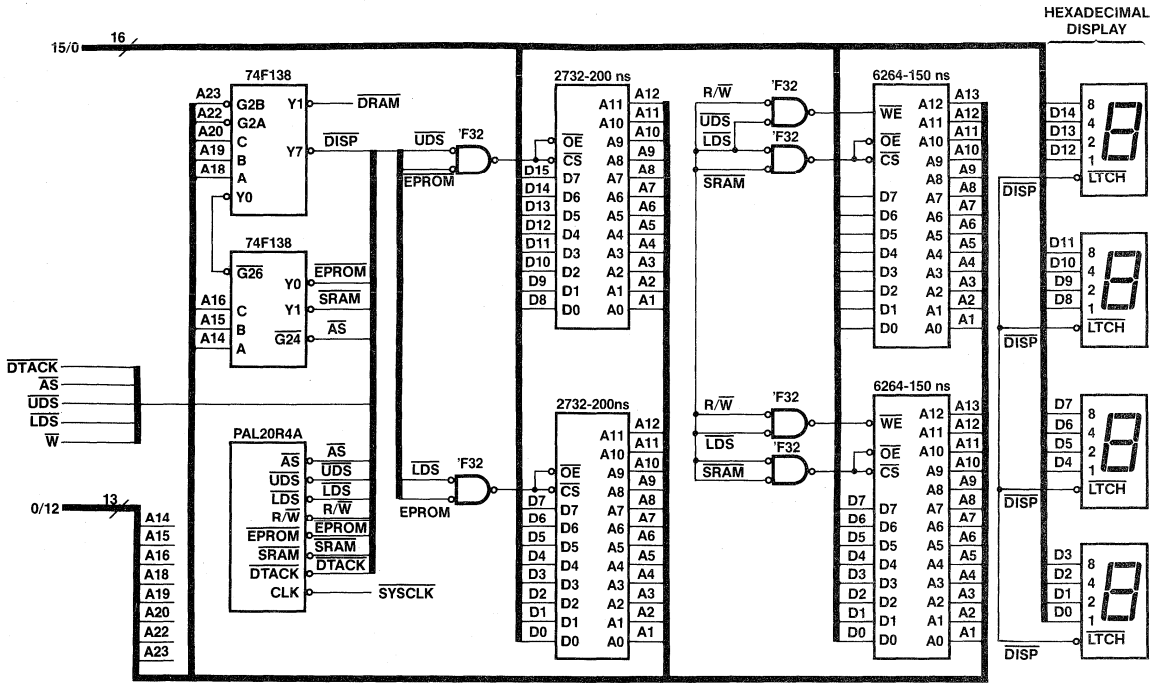


Figure 3.2. 68000-409 Interface Circuit Part II

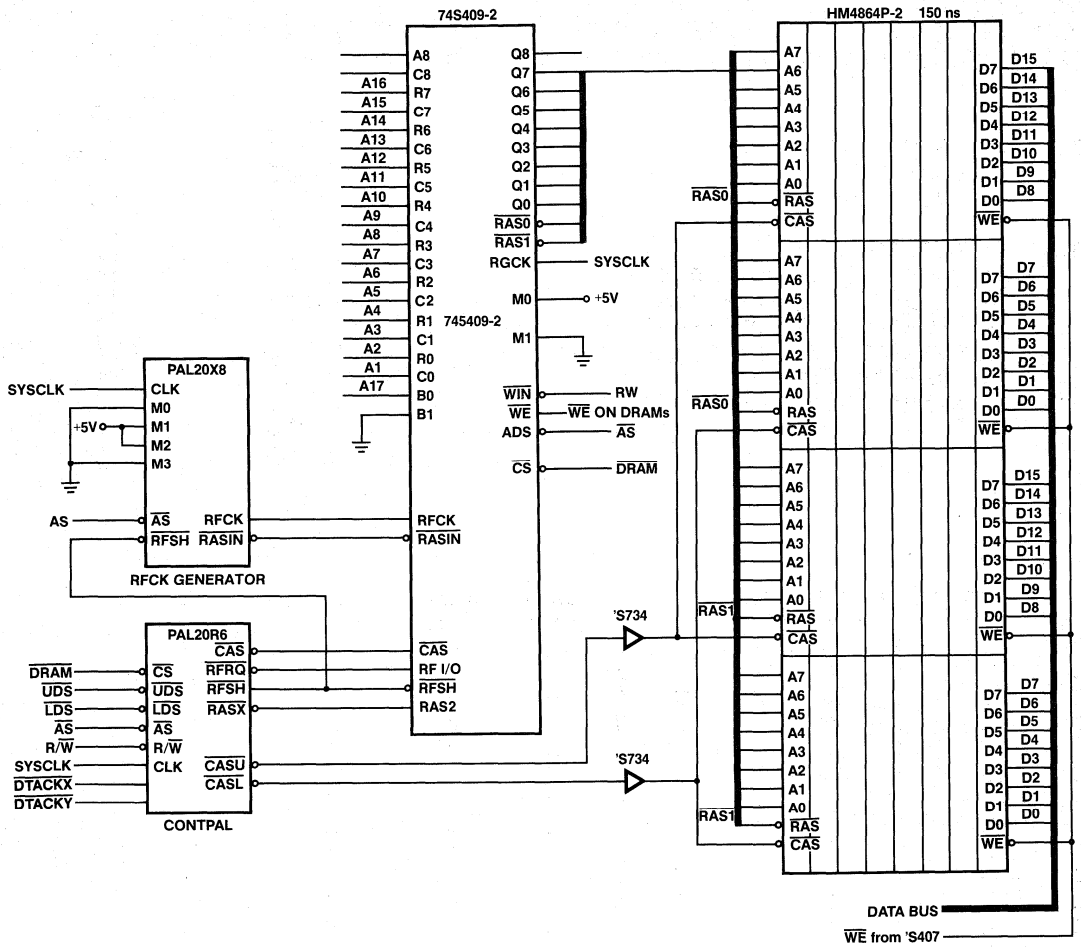
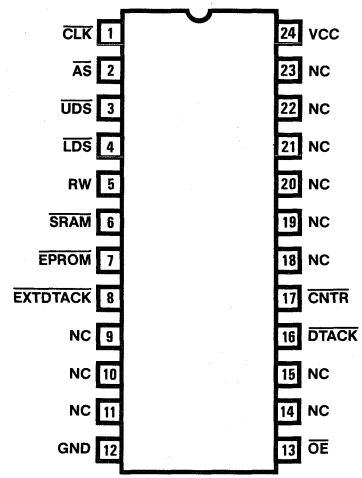


Figure 3.3 68000-409 interface circuit part III

```
PAL20R4A                                        68000L8/68000L10 INF'C PAL
68K.1                                             GLENN BAXTER  29 JUN 84
DTACK GENERATION
MMI SUNNYVALE, CALIFORNIA
/CLK /AS /UDS /LDS RW /SRAM /EPROM /EXTDTACK NC NC NC GND
/OE NC NC /DTACK /CNTR NC NC NC NC NC NC VCC


CNTR := AS*UDS*LDS*/CNTR
      + UDS*CNTR
      + LDS*/CNTR

DTACK = EXTDTACK
      + CNTR*AS*UDS*RW*EPROM
      + CNTR*AS*LDS*RW*EPROM
      + CNTR*AS*UDS*/RW*SRAM
      + CNTR*AS*LDS*/RW*SRAM
      + SRAM*RW*UDS
      + SRAM*RW*LDS
```

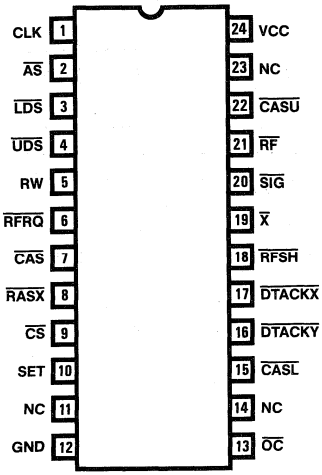


FUNCTION TABLE

/CLK	/OE	/CNTR	/AS	/UDS	/LDS	RW	/SRAM	/EPROM	/EXTDTACK	/DTACK
C	H	Z	X	X	X	X	H	H	L	L
C	H	Z	X	X	X	X	H	H	H	H
C	L	X	X	X	X	X	H	H	L	L
C	L	X	X	X	X	X	H	H	H	H
C	L	H	X	X	X	X	H	X	H	H
C	L	X	L	L	L	H	H	L	H	L
C	L	X	L	L	H	H	H	L	H	L
C	L	X	L	H	L	H	H	L	H	L
C	L	X	L	H	H	H	H	L	H	H
C	L	X	X	X	X	L	H	L	H	H
C	L	X	H	H	H	X	H	L	H	H
C	L	X	X	L	L	H	L	H	H	L
C	L	X	X	L	H	H	L	H	H	L
C	L	X	X	H	L	H	L	H	H	L
C	L	X	X	H	H	H	L	H	H	H
C	L	H	X	X	X	L	L	H	H	H
C	L	X	H	X	X	L	L	H	H	H
C	L	X	L	L	L	L	L	H	H	L
C	L	X	L	L	H	L	L	H	H	L
C	L	X	L	H	L	L	L	H	H	L
C	L	X	L	H	H	L	L	H	H	H

DESCRIPTION:
THIS PAL PRODUCES 0 WAIT-STATES IF A SRAM READ IS TAKING PLACE. IT INSERTS 1 WAIT-STATES IF EITHER A SRAM WRITE OR AN EPROM READ IS TAKKING PLACE. THE PAL ALSO HAS AN OR'ED INPUT FOR EXTERNAL DTACK (DELAYED 25 NS).

```
PAL20R6                                          PAL DESIGN SPECIFICATIONS
CONTPAL                                  GLENN A. BAXTER     15 AUG, 1984
RFSH AND DTACK GENERATOR
MMI SUNNYVALE, CALIFORNIA
CLK /AS /LDS /UDS RW /RFRQ /CAS /RASX /CS SET NC GND
/OC NC /CASL /DTACKY /DTACKX /RFSH /X /SIG /RF /CASU NC VCC


CASL = CAS * LDS
CASU = CAS * UDS

RFSH     := /AS * RFRQ
          + RF * RFSH
          + SIG * RFSH

SIG      := RASX * RFSH

RF       := RFRQ


DTACKX   := X * AS * RW * CS * LDS
          +  X * AS * RW * CS * UDS

X        := /RFSH * AS * RW * CS * LDS
          +  /RFSH * AS * RW * CS * UDS

DTACKY   := /RFSH * AS * /RW * CS * LDS
          +  /RFSH * AS * /RW * CS * UDS
```



10

DESCRIPTION:

THIS PAL WILL GENERATE FIVE BASIC SIGNALS:

/RFSH - A SIGNAL TO THE ´S409 TO PERFORM A FORCED REFRESH. THIS SIGNAL IS GENERATED BY TWO OTHER SIGNALS /SIG AND /PULSE.

/CASL - THIS IS THE LOWER BYTE COLUMN ADDRESS STROBE. IT MUST BE BUFFERED EXTERNALLY BY `S734 BEFORE GOING TO THE DRAMS.

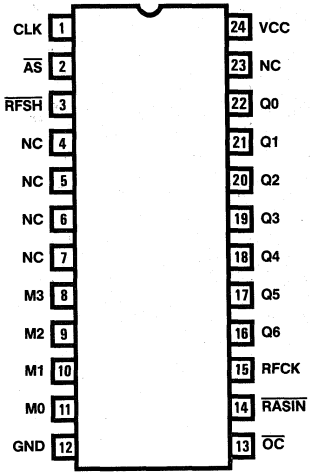
/CASU - THIS IS THE UPPER BYTE COLUMN ADDRESS STROBE. IT MUST BE BUFFERED EXTERNALLY BY `S734 BEFORE GOING TO THE DRAMS.

/DTACKY - THIS IS THE DATA TRANSFER ACKNOWLEDGE SUGNAL FOR A DRAM WRITE. IT ALLOWS ZERO-WAIT-STATES WRITE CYCLE. THIS SIGNAL MUST BE OR`ED WITH THE REST OF THE DTACK SIGNALS TO PRODUCE THE FINAL DTACK TO THE 68000.

/DTACKX - THIS IS THE DATA TRANSFER ACKNOWLEDGE SUGNAL FOR A DRAM READ. IT ALLOWS ONE-WAIT-CYCLE (TWO-WAIT-STATES) WRITE CYCLE. THIS SIGNAL MUST BE OR`ED WITH THE REST OF THE DTACK SIGNALS TO PRODUCE THE FINAL DTACK TO THE 68000.

```
PAL24 V1.7F - PAL20R6  -  RFSH AND DTACK GENERATOR

                 11 1111 1111 2222 2222 2233 3333 3333
   0123 4567 8901 2345 6789 0123 4567 8901 2345 6789

 8 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
 9 ---- ---- -X-- ---- ---- -X-- ---- ---- ---- ---- CAS*UDS

16 ---- ---- ---- ---- -X-- ---- ---- ---- ---- ---- RFRQ

24 ---- ---- ---- ---- ---- ---X -X-- ---- ---- ---- RASX*RFSH

32 -X-- -X-- ---- X--- ---- --X- ---- -X-- ---- ---- /RFSH*AS*RW*CS*LDS
33 -X-- ---- -X-- X--- ---- --X- ---- -X-- ---- ---- /RFSH*AS*RW*CS*UDS

40 X--- ---- ---- ---- -X-- ---- ---- ---- ---- ---- /AS*RFRQ
41 ---- ---- ---X ---- ---- ---X ---- ---- ---- ---- RF*RFSH
42 ---- ---- ---- ---X ---- ---X ---- ---- ---- ---- SIG*RFSH

48 -X-- -X-- ---- X--- ---X ---- ---- -X-- ---- ---- X*AS*RW*CS*LDS
49 -X-- ---- -X-- X--- ---X ---- ---- -X-- ---- ---- X*AS*RW*CS*UDS

56 -X-- -X-- ---- -X-- ---- --X- ---- -X-- ---- ---- /RFSH*AS*/RW*CS*LDS
57 -X-- ---- -X-- -X-- ---- --X- ---- -X-- ---- ---- /RFSH*AS*/RW*CS*UDS

64 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
65 ---- -X-- ---- ---- ---- -X-- ---- ---- ---- ---- CAS*LDS


LEGEND:  X : FUSE NOT BLOWN (L,N,0)    - : FUSE BLOWN    (H,P,1)
         0 : PHANTOM FUSE   (L,N,0)    O : PHANTOM FUSE (H,P,1)

NUMBER OF FUSES BLOWN =  557
```

```
PAL20X8                                          PAL DESIGN SPECIFICATION
RFCKGEN                                          GLENN A. BAXTER  08/20/84
REFRESH CLOCK GENERATOR
MMI SANTA CLARA, CALIFORNIA
CLK /AS /RFSH NC NC NC NC M3 M2 M1 M0 GND
/OC /RASIN RFCK Q6 Q5 Q4 Q3 Q2 Q1 Q0 NC VCC
```




```
RASIN = AS * /RFSH

/Q0     := Q0

/Q1     := /Q1 :+: Q0

/Q2     := /Q2 :+: Q0 * /Q1

/Q3     := /Q3 :+: /Q0 * /Q1 * /Q2

/Q4     := /Q4 :+: /Q0 * /Q1 * /Q2 * /Q3

/Q5     := /Q5 :+: /Q0 * /Q1 * /Q2 * /Q3 * /Q4

/Q6     := /Q6 :+: /Q0 * /Q1 * /Q2 * /Q3 * /Q4 * /Q5

/RFCK   :=  M3 * /Q6 * Q5 + M2 * /Q6 * /Q5 * Q4
        :+: M1 * /Q6 * /Q5 * /Q4 * Q3 + M0 * /Q6 * /Q5 /Q4 * /Q3 * Q2
```

10

DESCRIPTION

THIS PAL GNERATES TWO SIGNALS... THE FIRST IS /RASIN. THIS OUTPUT GOES TO THE ´S409 AND IS USED AS THE RAS INPUT. THE SECOND SIGNAL WHOSE DURATION IS 128 SYSTEM CLOCK CYCLES. THE TIME OF RFCK BEING LOW IS DETERMINED BY THE M0-M3 INPUTS (SEE TABLE BELOW). THE HIGH TIME IS OBVIOUSLY:

TIME HIGH = TOTAL CYCLE TIME - LOW TIME

= 128 CYCLES - LOW TIME

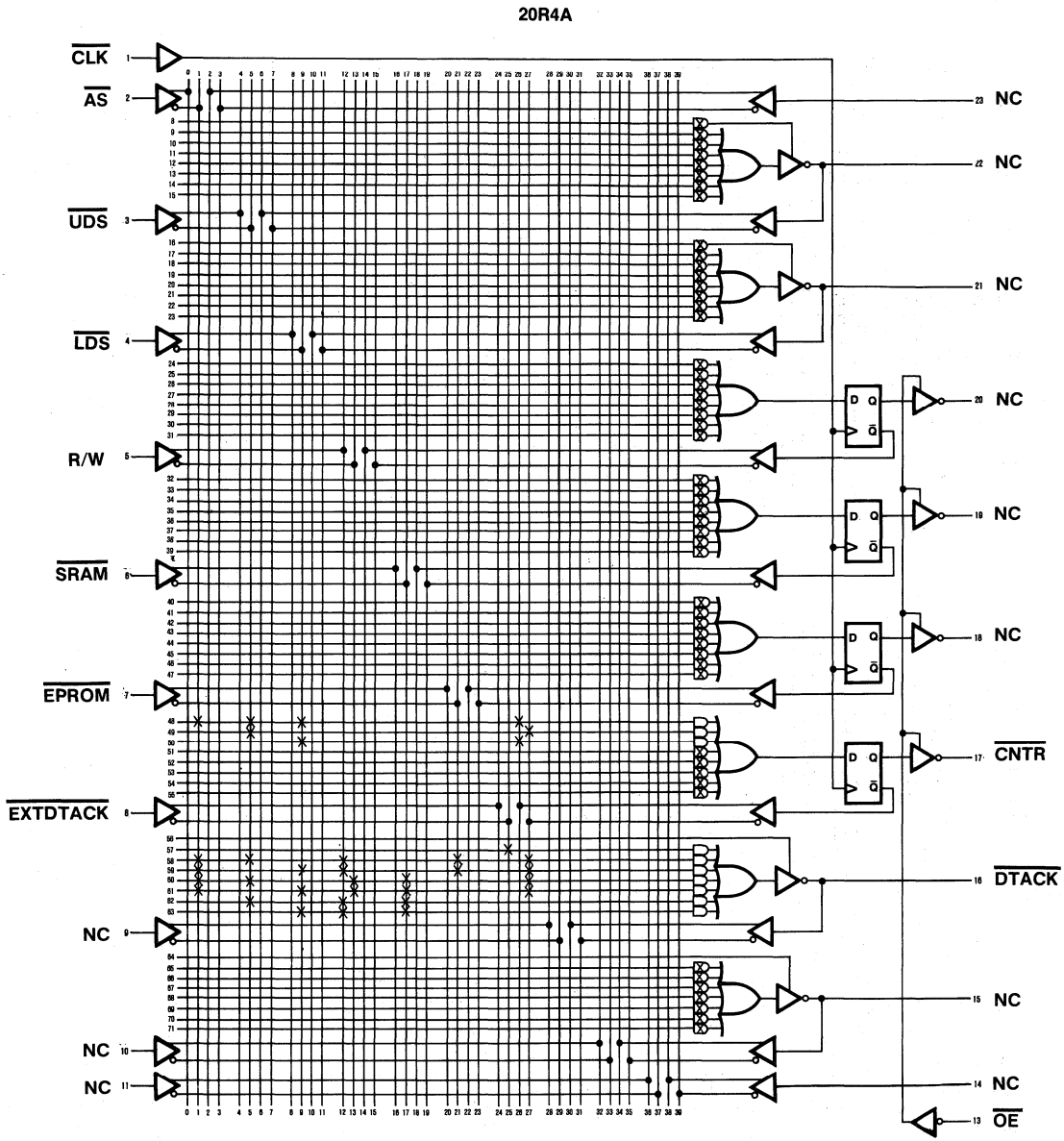
```
PAL24 V1.7F - PAL20X8  -  REFRESH CLOCK GENERATOR

                 11 1111 1111 2222 2222 2233 3333 3333
   0123 4567 8901 2345 6789 0123 4567 8901 2345 6789

 8 ---- --X- ---- ---- ---- ---- ---- ---- ---- ---- Q0

16 ---- ---- ---X ---- ---- ---- ---- ---- ---- ---- /Q1
18 ---- --X- ---- ---- ---- ---- ---- ---- ---- ---- Q0

24 ---- ---- ---- ---X ---- ---- ---- ---- ---- ---- /Q2
26 ---- --X- ---X ---- ---- ---- ---- ---- ---- ---- Q0*/Q1

32 ---- ---- ---- ---- ---X ---- ---- ---- ---- ---- /Q3
34 ---- ---X ---X ---X ---- ---- ---- ---- ---- ---- /Q0*/Q1*/Q2

40 ---- ---- ---- ---- ---- ---X ---- ---- ---- ---- /Q4
42 ---- ---X ---X ---X ---X ---- ---- ---- ---- ---- /Q0*/Q1*/Q2*/Q3

48 ---- ---- ---- ---- ---- ---- ---X ---- ---- ---- /Q5
50 ---- ---X ---X ---X ---X ---X ---- ---- ---- ---- /Q0*/Q1*/Q2*/Q3*/Q4

56 ---- ---- ---- ---- ---- ---- ---- ---X ---- ---- /Q6
58 ---- ---X ---X ---X ---X ---X ---X ---- ---- ---- /Q0*/Q1*/Q2*/Q3*/Q4*

64 ---- ---- ---- ---- ---- ---- X-X- ---X ---- ---- M3*/Q6*Q5
65 ---- ---- ---- ---- ---- --X- ---X X--X ---- ---- M2*/Q6*/Q5*Q4
66 ---- ---- ---- ---- --X- ---X ---X ---X X--- ---- M1*/Q6*/Q5*/Q4*Q3
67 ---- ---- ---- --X- ---X ---X ---X ---X ---- X--- M0*/Q6*/Q5*/Q4*/Q3*Q

72 ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
73 -X-- X--- ---- ---- ---- ---- ---- ---- ---- ---- AS*/RFSH

LEGEND:  X : FUSE NOT BLOWN (L,N,0)    - : FUSE BLOWN    (H,P,1)
         0 : PHANTOM FUSE   (L,N,0)    O : PHANTOM FUSE  (H,P,1)

NUMBER OF FUSES BLOWN =  712
```

3210	RFCK LOW DURATION (CYCLES)
0000	0
0001	4
0010	8
0011	12
0100	16
0101	20*
0110	24
0111	28
1000	32
1001	36*
1010	40*
1011	44*
1100	48
1101	52*
1110	56
1111	60

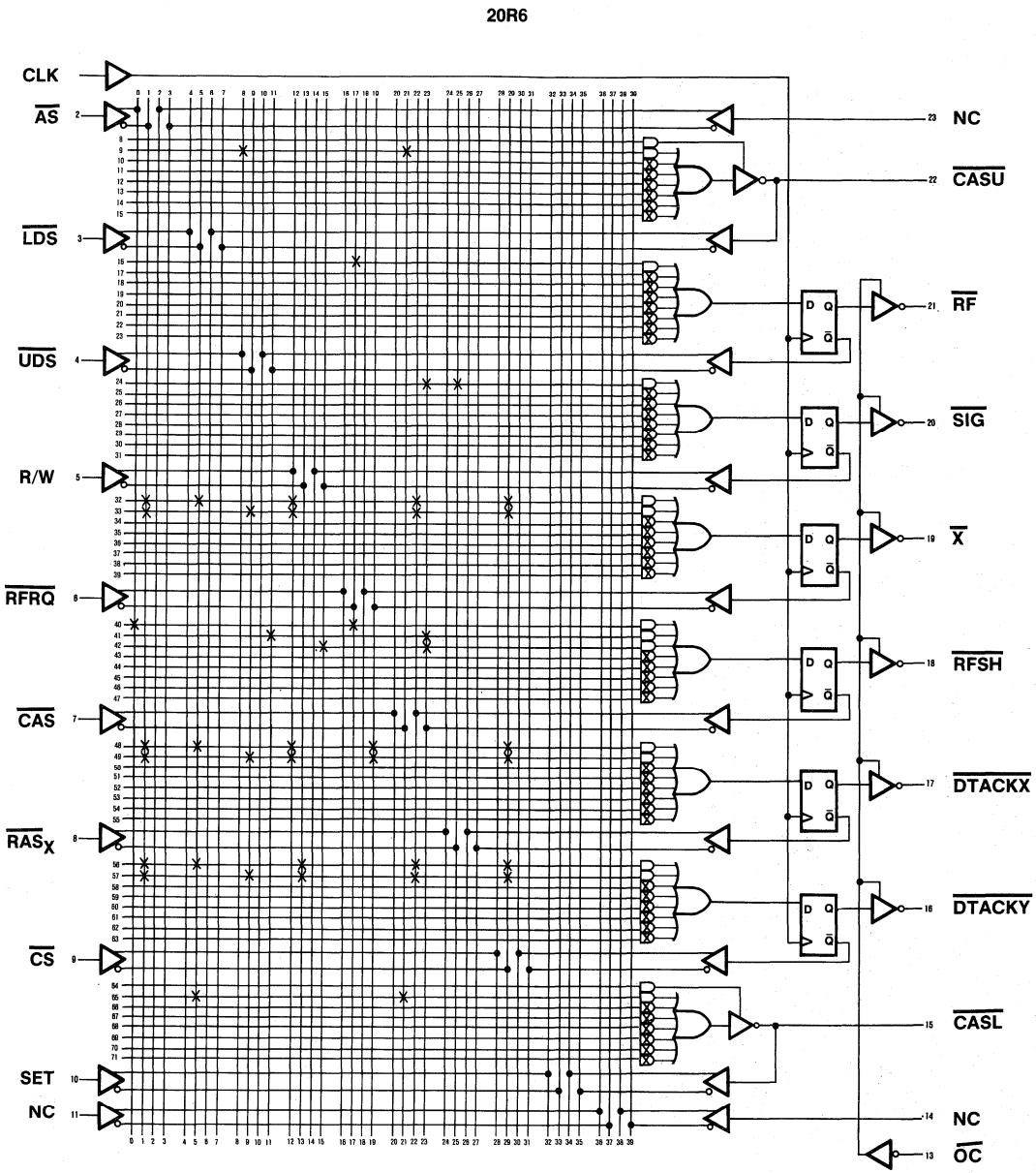
* Not allowed due to bad waveforms.

DTACK Generator

Logic Diagram PAL20R4A



Refresh Generator



Refresh Clock Generator

Logic Diagram PAL20X8