

APR405

Minimal Logic DRAM Interface for the DSP56156

By Paul Robertson,
DSP Applications,
Motorola Ltd.,
East Kilbride

INTRODUCTION

Many DSP applications require large amounts of memory. Significant reductions in cost can be achieved by using dynamic RAM (DRAM) in place of fast static RAM (SRAM). However, this is always at the expense of memory access speed. This application note presents a minimal glue DRAM interface for the DSP56156 which is designed for maximum performance.

Design Overview

This application note describes the interface of the MCM54400 1Mx4 80ns DRAM to the 40MHz DSP56156.

The design differs from conventional DRAM designs in the fact that no latches are used to hold the row and column addresses during DRAM accesses. Instead, the DSP address and data lines are connected directly to the DRAM, and a single PAL16R6 is used for memory decoding, read, write and refresh control.

Although the design presented is specific to the MCM54400, it is important to note that it can easily be extended to incorporate other DRAM configurations.

DRAM Basics

The MCM54400 DRAM is a 4 Mbit part, organised as 1,048,576 4-bit locations. The ten address pins on the device are time multiplexed between the DRAM row and column addresses, producing a total of twenty address lines.

Many DRAMs support special access modes that enable improved performance in certain situations. An example of this is the 'fast page' mode for successive accesses to cells in the same row. These accesses take less time to complete than using normal random access modes. Despite the performance improvements that such methods offer, this design only implements the 'normal' random read cycle and early write cycle in order to achieve the design goal of a minimal logic DRAM interface.

DRAM cells require to be 'refreshed' periodically in order to maintain the stored information. Bits in the MCM54400 need to be refreshed every 16 milliseconds. Some DRAMs have longer refresh times (128ms is common). All the bits in a row are refreshed simultaneously when the row is addressed. The refresh procedure involves cycling through the 1024 row addresses in sequence within the specified refresh time.

The MCM54400 supports three methods of refresh: /RAS-only refresh; /CAS before /RAS refresh; and hidden refresh. The one that was chosen for this application was /CAS before /RAS refresh, for the reason that it is the simplest to implement. Only the /CAS and /RAS signals need to be periodically generated, as an internal counter on the DRAM stores the address of the next row to be refreshed.

For more detailed information on the MCM54400 the reader should refer to the device Technical Data Sheet (Ref. MCM54400/D).



Circuit Design

A circuit diagram of the DRAM interface is shown in Figure 1. It was designed to operate as an extension to the DSP56156ADM Application Development Module (ADM).

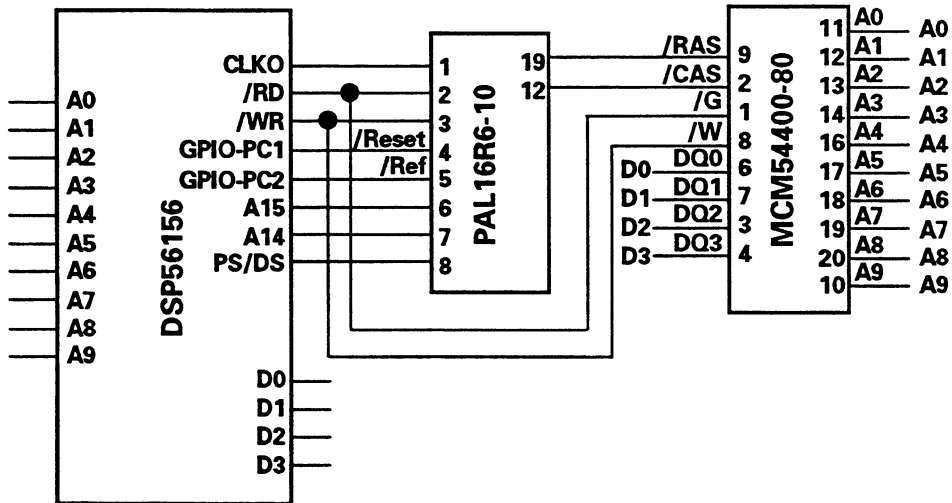
The DSP56156 address and data lines are interfaced directly to the DRAM, as are the read and write lines. The PAL16R6 generates /RAS and /CAS from the appropriate DSP signals.

The DRAM is mapped to the DSP56156 X-memory with address lines A15, A14 and PS/DS being used to define the exact location in the DSP memory space. The PS/DS signal differentiates external program memory from external data memory access, and is used to place the DRAM in X data memory. The address lines A15 and A14 reserve the address range \$8000...\$8BFF for DRAM.

The PAL has a DRAM Refresh (/Ref) input that controls the DRAM refresh function. An external source, in this case the DSP using a general purpose I/O line and timer, enables a DRAM refresh cycle by asserting the /Ref line for a minimum period.

A Reset (/Reset) line has been included in the design to enable the circuit to be initialised to a known state after power up.

Figure 1 Circuit Diagram



DRAM Access Timing

The normal DRAM access is shown in Figure 2. The row and column addresses are latched separately by the /RAS and /CAS signals, and the access is terminated by the deassertion of /RAS and /CAS.

Figure 2 Normal DRAM Access

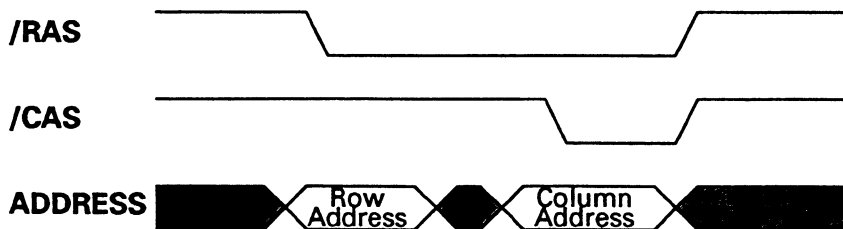


Figure 3 DRAM Read Cycle

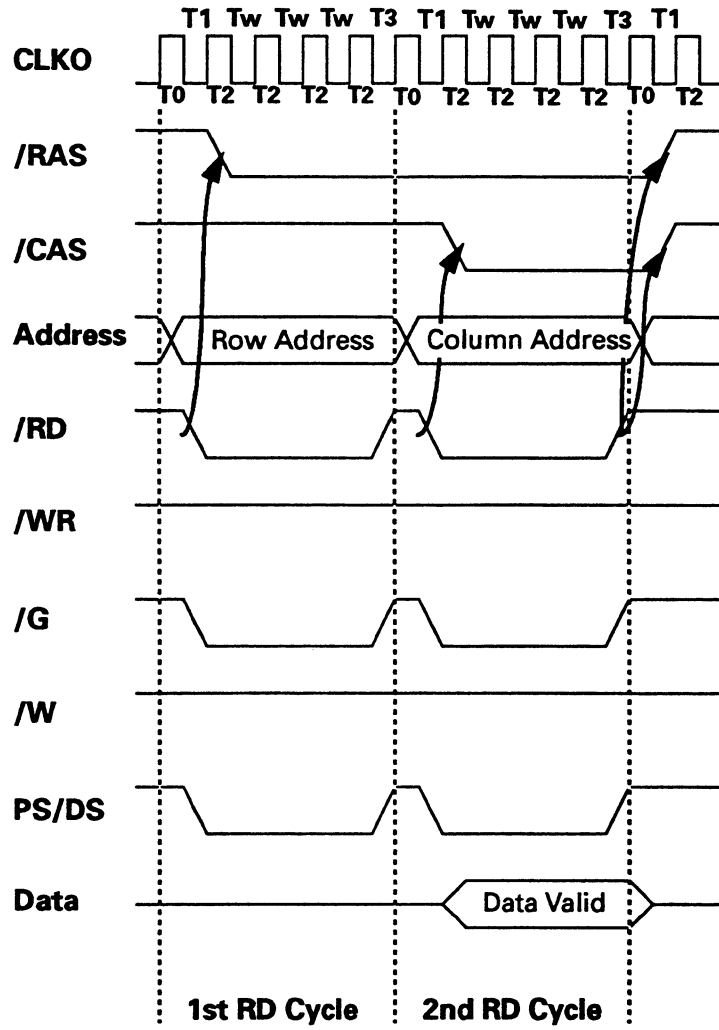


Figure 4 DRAM Write Cycle

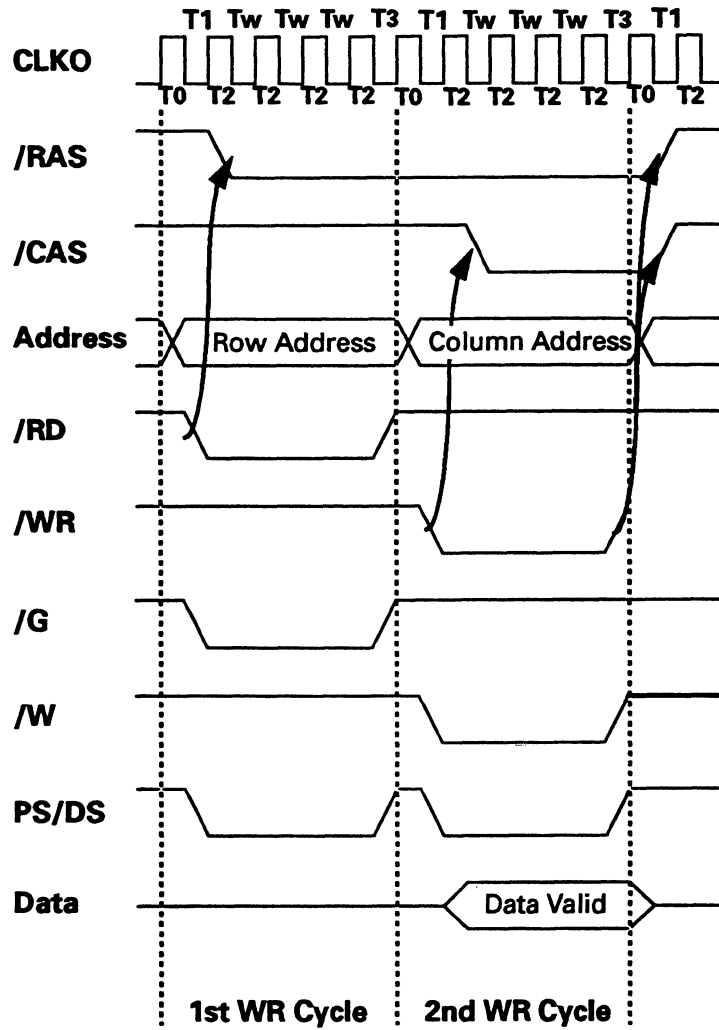
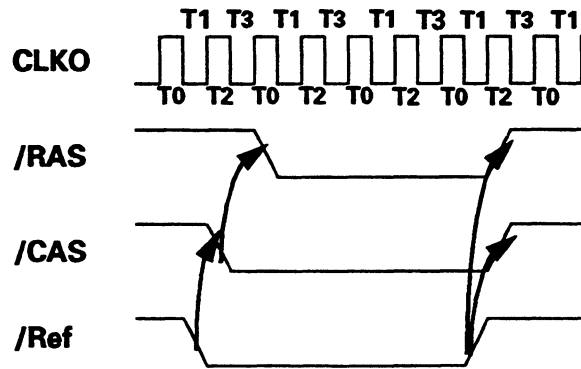


Figure 5 CAS Before RAS Refresh Cycle



Figures 3 and 4 show the DRAM read and write timing diagrams. A single DRAM read or write cycle is performed using two DSP external accesses. On the first DSP cycle the row address is output, on the second the column address is output and the data is read or written.

The DSP initiates each DRAM read or write cycle by performing a dummy read cycle. The row address is output, address lines A15 and A14 are decoded by the PAL and /RAS is asserted appropriately. The second DSP cycle determines whether the access is a read or a write cycle. The column address is output, the /RD and /WR signals are decoded by the PAL and the /CAS signal is asserted. The data is read or written during this cycle, and the transfer is terminated with the deassertion of /RAS and /CAS.

The DRAM /CAS before /RAS refresh cycle is shown in Figure 5. The refresh cycle is controlled by the /Ref input to the PAL. The /RAS and /CAS waveforms are generated from the negative and positive edges of the /Ref signal. There is a constraint on the minimum width of the /Ref pulse, which must be at least 134ns. This design uses the DSP56156 timer to generate a periodic refresh pulse every 15.5us with an assertion width of 150ns. It is important that the refresh signal does not occur during a DRAM access cycle, as in this case the refresh will be ignored. Before any read or write to DRAM the DSP examines the Timer Count Register (X:\$FFEC) to determine if a refresh will occur during the DRAM access, and if necessary delays the access until after the refresh cycle has been completed.

PAL Design

This section deals with the design of the PAL. It describes in detail how the PAL equations are derived from the timing diagrams.

The PAL used in this design is the PAL16R6 which has 6 registered and 2 combinatorial outputs. Each registered output has a D-Type flip-flop at the output stage which is clocked by an external clock signal. The DSP56156 CLKO output signal is the DSP master clock signal, and is used to clock the PAL.

A detailed timing analysis of the DRAM interface, in terms of maximum and minimum assertion and deassertion times, and appropriate set up and hold times for each signal, identifies the time critical parts of the interface. These are:

- The minimum assertion times of /CAS and /RAS.

- The valid data set up and hold times with respect to /CAS.

- The minimum deassertion time of /RAS between successive DRAM accesses.

These determine the minimum number of wait states which are needed by the DSP.

Once the signal timings and relationships of the interface are determined, the next stage in the design is to work out how to derive the PAL signals /RAS and /CAS with respect to the DSP56156 master clock (CLKO), taking into account the PAL logic propagation delays.

State Machine

The state machine design approach is used to define the PAL. The arrows in the DRAM Read and Write Cycle timing diagrams show the signal transitions which define each part of the /RAS and /CAS waveforms.

Figure 6 shows the State Diagram of the system. This section describes each state in detail.

IDLE

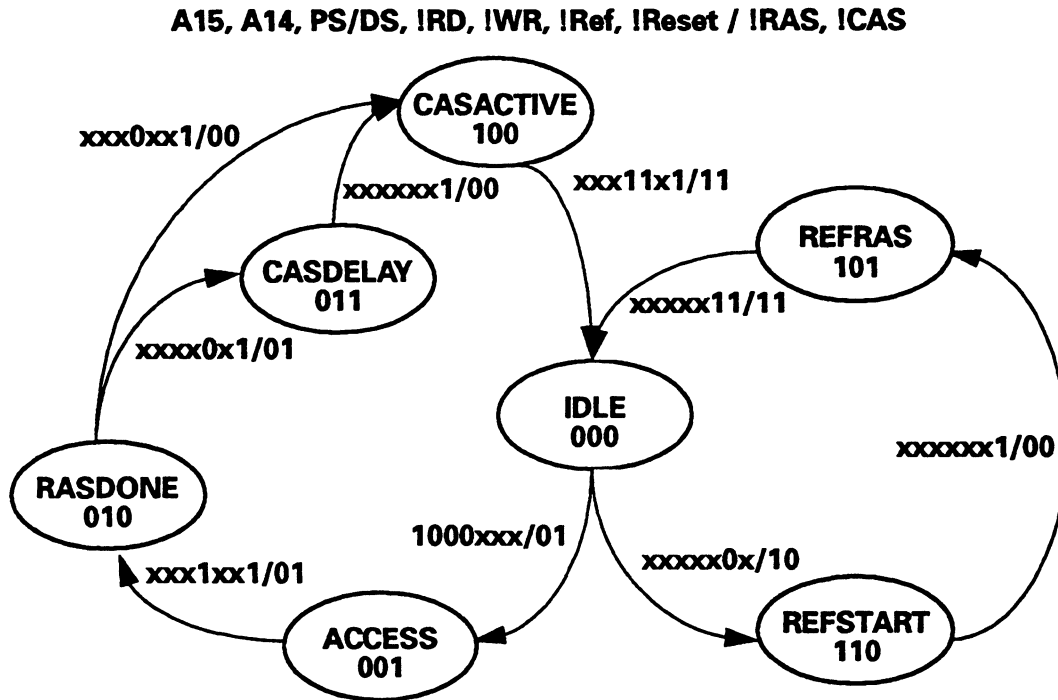
The IDLE state is the rest state of the system, i.e. when the DRAM is not being accessed or refreshed. When the circuit is reset it enters the IDLE state.

IDLE also defines the termination of a DRAM access cycle.

/RAS - Deasserted, High

/CAS - Deasserted, High

Figure 6 PAL State Diagram



ACCESS

A DRAM read or write cycle is initiated by the DSP performing a dummy read cycle to an address within the DRAM address space. The lower 10 bits of the address (A9..A0) contain the DRAM row address of the cell to be accessed and A14, A15, PS/DS and /RD are decoded by the PAL to define a valid DRAM access.

IF A15&!A14&!PS/DS&!IRD THEN DRAM Access Cycle

/RAS - Asserted, Low

/CAS - Deasserted, High

RASDONE

This state defines the end of the first half of the DRAM access cycle. It is entered when the /RD signal is deasserted, indicating the end of the initial dummy read cycle.

/RAS - Asserted, Low

/CAS - Deasserted, High

The state machine will remain in this state until the DSP initiates another read or write cycle.

CASDELAY

This state is entered if the DRAM access is a write cycle. The DRAM write cycle requires the assertion of /CAS to be delayed to fit into the DSP data valid window. The CASDELAY state delays the assertion of /CAS by one clock cycle.

/RAS - Asserted, Low

/CAS - Deasserted, High

CASACTIVE

This state defines when /CAS is asserted. The state is entered either on

- the assertion of /RD during a DRAM read cycle, or

- after exiting the CASDELAY state.

/RAS - Asserted, Low

/CAS - Asserted, Low

The DRAM read or write cycle is terminated when /RD and /WR are deasserted, and the system then returns to the IDLE state.

REFSTART

This state defines the start of the DRAM refresh cycle. The condition for entering this state is the assertion of /Ref.

/RAS - Deasserted, High

/CAS - Asserted, Low

REFRAS

This state is entered immediately from REFSTART and there are no conditions. The state defines the assertion of /RAS.

/RAS - Asserted, Low

/CAS - Asserted, Low

The refresh cycle is terminated by the deassertion of /Ref, and the system returns to the IDLE state.

PAL Equations

The PAL source file is shown in Figure 7.

One interesting problem encountered during the design relates to the state outputs Q0, Q1, and Q2. The PAL16R6 allows 8 product terms for each registered output, and 7 for each combinatorial output. When an initial set of PAL equations was compiled, the number of product terms generated for the state outputs exceeded the allowed maximum. This basically meant the equations could not be realised on the PAL. However, changing the state outputs from positive logic to negative logic reduced the number of product terms to less than 8, thus allowing the design to fit into a single PAL.

Figure 7 PAL Source File

```
Name      DRAMPAL;
Partno    none;
Revision  1.0;
Date      17/6/92;
Designer  P. Robertson;
Company   Motorola;
Location  DSP Applications;
Assembly  None;
Device    p16r6;

/*****
/*      This is the source file for the DRAM Decode PAL      */
*****/
/*****
/* Allowable Target Device Types : PAL16R6 or equivalent */
*****/

/** Inputs **/

Pin 1      = Clock;
Pin 2      = !RD;
Pin 3      = !WR;
Pin 4      = !Reset;
Pin 5      = !Ref;          /* DRAM Refresh Line */
Pin 6      = A15;
Pin 7      = A14;
Pin 8      = PSDS;

/** Outputs **/

Pin 19     = !RAS;
Pin 12     = !CAS;
Pin 17     = !Q2;
Pin 16     = !Q1;
Pin 15     = !Q0;

/** Declarations and Intermediate Variable Definitions **/

FIELD      State=[Q2..Q0];
$DEFINE    IDLE          'b'000
$DEFINE    ACCESS        'b'001
$DEFINE    RASDONE       'b'010
$DEFINE    CASDELAY      'b'011
$DEFINE    CASACTIVE     'b'100
$DEFINE    REFSTART      'b'101
$DEFINE    REFRAS        'b'110
$DEFINE    UNUSE1        'b'111

dram      = A15&!A14&!PSDS&RD;

/** Logic Equations **/

SEQUENCE          State(

PRESENT          IDLE
IF Ref NEXT REFSTART;
IF dram NEXT ACCESS;
DEFAULT NEXT IDLE;

PRESENT ACCESS
OUT RAS;
IF !RD NEXT RASDONE;
IF Reset NEXT IDLE;
```



```

DEFAULT NEXT ACCESS;

PRESENT RASDONE
OUT RAS;
IF RD NEXT CASACTIVE;
IF WR NEXT CASDELAY;
IF Reset NEXT IDLE;
DEFAULT NEXT RASDONE;

PRESENT CASDELAY
OUT RAS;
IF Reset NEXT IDLE;
NEXT CASACTIVE;

PRESENT CASACTIVE
OUT RAS OUT CAS;
IF !RD&!WR NEXT IDLE;
IF Reset NEXT IDLE;
DEFAULT NEXT CASACTIVE;

PRESENT REFSTART
OUT CAS;
IF Reset NEXT IDLE;
NEXT REFRAS;

PRESENT REFRAS
OUT RAS OUT CAS;
IF !Ref NEXT IDLE;
IF Reset NEXT IDLE;
DEFAULT NEXT REFRAS;

PRESENT UNUSE1
NEXT IDLE;
}

```

Hardware Considerations

The power surge generated when the DRAM turns on during an access can cause problems, such as glitches on the control lines. Therefore, it is important to provide good power and ground connections to the DRAM. These should be decoupled using a suitable by-pass capacitor placed as close to the DRAM as possible. A 0.1uF tantalum should be sufficient.

Performance Evaluation

The minimum number of wait states required for a DRAM access cycle is 3. This is illustrated in Figures 3 and 4. The number of wait states was calculated assuming an 80ns DRAM and a DSP core frequency of 40MHz.

The interface supports a maximum data transfer rate of 1.18Mbytes/s.

This data rate can be achieved if the accesses are made to a single row or column. In this case the end of row and column checks can be omitted.

The bandwidth of the DRAM interface could be further increased by modifying the design to support the Fast Page Mode.

Further Design Ideas

In a typical end application the DRAM refresh control would not be done by the DSP. Instead a host MCU or timer circuit would be used to generate !Ref. This would allow the DSP to be placed in a low power STOP mode when not being used. Taking the refresh control off the DSP creates the problem of having a refresh cycle occurring during a DRAM access, or vice-versa. This is avoided in the current design by looking at the Timer Counter Register before performing a DRAM access, and delaying the access if a refresh cycle is about to occur. A possible solution when an external device is responsible for the DRAM refresh would be to get the controlling device to generate two signals, !Ref and 'Refresh Imminent'. The 'Refresh Imminent' signal would be an input to the DSP and would indicate when a refresh was taking place or about to take place, allowing the DSP to appropriately delay a DRAM access during a refresh cycle.

Software

The program listings for the individual modules of the DRAM interface are shown in Figures 8 - 12.

DRAM Initialisation

Figure 8 shows the equates, reserved memory locations and initialisation code for the DRAM interface.

Figure 8 DRAM Initialisation

```
;*****
; Equates
;*****
ipr      equ    $FFDF      ;Interrupt Priority Register
pcc      equ    $FFC1      ;Port C Control Register
pcddr    equ    $FFC3      ;Port C Data Direction Register
pcd      equ    $FFE3      ;Port C Data Register
tpr      equ    $FFEF      ;Timer Pre-load Register
tcpr     equ    $FFEE      ;Timer Compare Register
tctr     equ    $FFED      ;Timer Count Register
tcr      equ    $FFEC      ;Timer Control Register
bcr      equ    $FFDE      ;Bus Control Register

row_strt equ    $8000      ;DRAM Row start address
col_strt equ    $8000      ;DRAM Column start address
row_fin  equ    $8400      ;DRAM Row finish address
col_fin  equ    $8400      ;DRAM Column finish address
ctr_cmp  equ    40         ;Counter Compare Value - For Refresh
mask     equ    $000F      ;Mask for 4-bit data

;*****
; Reserved X-Memory
;*****
        org     x:$0
wr_row   ds     1          ;Current DRAM Write Row Address
wr_col   ds     1          ;Current DRAM Write Column Address
rd_row   ds     1          ;Current DRAM Read Row Address
rd_col   ds     1          ;Current DRAM Read Column Address
data     ds     1          ;DRAM Data
```

```

;*****
; Interrupt Vectors
;*****
        org     p:$0
        jmp     start           ;Reset Vector

        org     p:$20         ;Timer Overflow
        jsr     refsh          ;Jump to DRAM Refresh Routine

;*****
; Main Program
;*****
        org p:$40

start
        ; Initialise Port C      ;PC1 is DRAM Reset line - !Reset
                                   ;PC2 is DRAM Refresh line - !Ref
        move    #>0,r0
        move    r0,x:pcc        ;make Port C GPIO
        move    #>$F,r0         ;
        move    r0,x:pcddr      ;Make PC0,1,2,3 outputs
        move    r0,x:pcd        ;Initialise outputs high

        ;Reset DRAM circuit to initialise to IDLE state
        bfclr   #>$2,x:pcd      ;Assert !Reset
        nop
        nop
        nop
        bfset   #>$2,x:pcd      ;Deassert !Reset

        ;Initialise DRAM - On power up an initial pause of 100us is
        ;required followed by a minimum of 8 active cycles of /RAS.
        ;It is assumed that the time for the DSP to exit Reset and bootload
        ;program is sufficient to meet the pause of 100us. 8 refresh cycles
        ;are performed to meet the 2nd requirement.
        ;Note: This requirement is documented in the DRAM Data Sheet.
        do      #8,end_init
        bfclr   #>$4,x:pcd      ;Assert !Ref
        nop
        ;Needs to be asserted for a 133ns min.
        bfset   #>$4,x:pcd      ;Deassert !Ref
        nop
        nop
end_init

        ;Initialise Data Values
        move    #$407F,y1       ;Insert 3 wait states
        move    y1,x:bcr

        ;Initialise Timer - On chip timer is used for DRAM refresh
        move    #$0200,a         ;Initialise Timer Control Register
        move    a,x:tcr
        move    #311,a           ;Initialise Timer Preload Register
        move    a,x:tpr          ; 311, 15.5us @ 40MHz
        move    #3C000,a
        move    a,x:ipr          ;Enable Timer Interrupts
        bfset   #$8000,x:tcr     ;TE=1, Enable Timer
        movec   #0,sr           ;Unmask Interrupt

```

DRAM Read

The listing for the DRAM read cycle is shown in Figure 9.

The subroutine reads a 16-bit word from DRAM and stores it at location X:DATA in DSP memory. The code has been written to enable a block transfer of data from DRAM to be easily implemented.

The pointer registers R1 and R2 hold the row and column address of the DRAM cell to be accessed. The address of the first nibble of the 16-bit data word to be read is loaded into R1 and R2 at the start of the subroutine. A Do Loop is used to read the 4 separate nibbles which are packed to form the 16-bit data word. Before the actual DRAM read cycle is performed interrupts are disabled (except the Timer interrupt) and the Refresh Counter is checked, (see Section 10.4). The reason for disabling interrupts is to ensure that the instructions that perform the DRAM access occur consecutively without interruption.

The Read Cycle is performed after returning from the Check Refresh Counter subroutine. The row address is output followed by the column address. The data is read into B1 during the output of the column address. An important point to highlight is the NOP instruction inserted between the output of the row address and column address. It has been commented with the statement 'If not using CLK0 insert NOP - asynchronous clocks'. If the PAL is being clocked using the CLK0 signal from the DSP then this NOP instruction is not needed. However, if the clock is taken direct from the crystal, for example, then a NOP needs to be inserted. The reason is that the DSP internal clock and the PAL clock are asynchronous.

After completion of the Read Cycle the interrupts are enabled and the register holding the received data word (B accumulator) is shifted right 4 bits. Once all 4 nibbles have been received the 16-bit data word is stored in DSP memory.

The last part of the routine supports block reads from DRAM. The value of the row address pointer R1 is saved, this holds the row address of the start of the next 16-bit word in DRAM. Before the address is saved it is checked to see if the last row location has been reached. If it has, the column address is incremented and the row address is reset to the start of the next row.

Figure 9 DRAM Read.

```

;*****
; DRAM Read
;*****
;*****
; Registers corrupted - R1, R2, N2, M1, M2, X1, Y1, A, B

;This routine reads 16-bit words from DRAM. The data is read
;4-bits at a time and packed to form a 16-bit word.
;*****

dram_rd    move    x:rd_row,r1      ;Pointer for Last Row Address
           move    x:rd_col,r2     ;Pointer for Read Column address
           move    #1,n2
           move    #mask,y1        ;Get 4-bit mask - this masks out bits
           move    #$fff,m1        ;4..15
           move    m1,m2

           ori     #$02,mr         ;Disable interrupts e.g Level 1
           jsr    ref_chk          ;Check Refresh Counter
           clr    b                ;Clear received word register
           do     #4,end_wrd       ;Loop to receive 4-bits at a time

read       move    x:(r1)+,x1      ;Output Row Address
           ;nop                    ;** If Not Using CLK0 insert NOP
           ;- asynchronous clocks! **
           move    x:(r2),b1       ;Output Column Address and Read
           ;Data

end_wrd    asr4    b                ;Shift right 1 nibble (4-bits)

           move    b0,x:data       ;Store received word
           andi   #$fc,mr         ;Enable interrupts

;***** Check if end of Row reached - If reading a block of data *****
           move    r1,a            ;Get Row address counter

           move    #row_fin,x1     ;Compare value to test if the end
           cmp    x1,a            ;of a Row has been reached
           jeq    new_coll

           move    r1,x:rd_row     ;Save Row Address

           jmp    rd_end

new_coll   move    #row_strt,a     ;Set Row Address to start of Row
           move    a,x:rd_row

           rnd    a    (r2)+n2     ;Increment Column Counter

;***** Check if end of Column reached - If reading a block of data *****
           move    r2,a            ;Get Column Address counter
           move    #col_fin-1,x1   ;Compare if last column
           cmp    x1,a            ;has been reached
           jne    stor            ;If it hasn't store next column address

           move    x:rec_col_strt,a ; else reset column address

stor       move    a,x:rd_col     ;Store Column Address

rd_end    rts

```

DRAM Write

The listing for the DRAM write cycle is shown in Figure 10.

The subroutine writes a 16-bit word to DRAM from location X:DATA in DSP memory. The code has been written to enable a block transfer of data to DRAM to be easily implemented.

The structure of the DRAM write cycle is identical to the read cycle.

Figure 10 DRAM Write.

```
;*****
; DRAM Write
;*****

;*****
; Registers corrupted - R1, R2, N2, M1, M2, X1, Y1, A, B
;*****

;This routine writes 16-bit data to DRAM. The data words are
;unpacked and written a nibble at a time (4-bits).
;*****

dram_wr      move    x:wr_row,r1      ;Get present Write Row Address
             move    x:wr_col,r2      ; and Column Address
             move    #1,n2
             move    #$fff,m1
             move    m1,m2

             ori     #$02,mr          ;Disable interrupts e.g Level 1
             jsr    ref_chk           ;Check Refresh Counter
             move    x:data,b         ;Get 16-bit data to write to DRAM
             do     #4,end_word       ;Loop to transmit 4-bits at a time

write        move    b,x1             ;Data to be written
             move    x:(r1)+,y1       ;Output Row Address
             ;nop                      ;** If Not using CLKO insert NOP
             ;- asynchronous clocks! **
             move    x1,x:(r2)        ;Output Column Address and Write
             ;Data

end_word     asr4    b                ;Shift to get next nibble

             andi   #$fc,mr          ;Enable interrupts

;***** Check if end of Row reached - If reading a block of data *****
             move   r1,a              ;Get Row address counter

             move   #row_fin,x1       ;Compare value to test if the end
             cmp    x1,a              ;of a Row has been reached
             jeq    new_col
             move   r1,x:wr_row       ;Save Row Address

             jmp    wr_end

;***** Check if end of Column reached - If reading a block of data *****
```

```

new_col
    move    #row_strt,a
    move    a,x:wr_row      ;Set Row Address to start of Row

    rnd a   (r2)+n2        ;Increment Column Counter

    move    r2,a            ;Get Column Address counter
    move    #col_fin-1,x1  ;Compare if last column
    cmp     x1,a            ;has been reached.
    jne     store          ;If it hasn't store next column address

    move    #row_fin-1,a    ;Save Last DRAM Write Row and
    move    a,x:last_row1  ; Column Address
    move    #col_fin-1,a
    move    a,x:last_coll

    move    x:rec_col_strt,a ; else reset column address

store    move    a,x:wr_col      ;Store Column Address.

wr_end
    rts

```

Refresh Counter Compare

Before a DRAM access is performed, the Timer Count Register is examined to check if a DRAM refresh cycle is about to occur. If a refresh cycle is imminent, the program flow is kept within the subroutine until the refresh has finished.

Figure 11 Refresh Counter Compare.

```

;*****
; Refresh Counter Compare
;*****
;This routine fetches the Timer Count Register (tctr) and compares
;it with a test value. This is to ensure there is enough time to
;perform a DRAM Read or Write cycle before the next Refresh.
ref_chk  move    x:tctr,x1      ;Get contents of Timer Count Register
         move    #ctr_cmp,a    ;Get compare value
         cmp     x1,a            ;and compare with tctr
         jge    ref_chk        ;Keep comparing until refresh o.k
         rts

```

DRAM Refresh


The DRAM refresh cycle is implemented by asserting the !Ref GPIO pin for a minimum of 133ns.

Figure 12 DRAM Refresh

```

;*****
; DRAM Refresh Routine
;*****
refsh    bfcclr  #>$4,x:pcd      ;Assert !Ref
         nop     ;Needs to be asserted for a 133ns min
         bfset  #>$4,x:pcd      ;Deassert !Ref
         rti

```

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:

USA/EUROPE: Motorola Literature Distribution;
P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, Toshikatsu Otsuki,
6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-3521-8315

MFAX: RMFAX0@email.sps.mot.com -TOUCHTONE (602) 244-6609
INTERNET: <http://Design-NET.com>

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298



MOTOROLA

APR405/D

