PROGRESS REPORT

October 1, 1980 - March 31, 1981

"A Study of Real-Time Computer Graphic Display Technology for Aeronautical Applications"

NASA Research Grant NSG - 1355

S.A. Rajala

North Carolina State University
Department of Electrical Engineering
Raleigh, N.C. 27650

April 30, 1981

(NASA-CR-164221) A STUDY OF REAL-TIME COMPUTER GRAPHIC DISPLAY TECHNOLOGY FOR AERONAUTICAL APPLICATIONS Progress Report, 1 Oct. 1980 - 31 Mar. 1981 (North Carolina State Univ.) 67 p HC A04/MF A01 CSCL 09B G3/61

N81-22727

Unclas
42138

APPENDIX A

I. Introduction

The primary goal of the research conducted under this grant has been and will continue the design and implementation of hardware and software for real-time computer graphic displays for cockpits. The main emphasis of the past six month period has been the development, simulation and testing of an algorithm for anti-aliasing vector drawings.

II. Anti-aliasing of Vector Drawings

Of great interest to the users of raster graphic display devices, in the removal of the adverse effects of spatial sampling. The pseudo-anti-aliasing line drawing algorithm we propose is an extension to Bresenham's algorithm for computer control of a digital plotter [1]. While retaining the salient features of the original algorithm, the new algorithm does not reproduce a line as a sequence of disjoint line segments. The new algorithm produces a series of overlapping line segments where the display intensity shifts from one segment to the other in this overlap (transition region). True anti-aliased lines can be considered as having an overlapping behavior as well, but in these lines the rate of intensity shift and therefore the length of the overlap is a function of the slope of the true line. In this algorithm the length of the overlap and the intensity shift are essentially constants because the purpose of the transition region is an aid to the eye in integrating the segments into a single smooth line.

The anti-aliasing algorithm retains the following important features of Bresenham's algorithm:

I. Introduction

The primary goal of the research conducted under this grant has been and will continue the design and implementation of hardware and software for real-time computer graphic displays for cockpits. The main emphasis of the past six month period has been the development, simulation and testing of an algorithm for anti-aliasing vector drawings.

II. Anti-aliasing of Vector Drawings

Of great interest to the users of raster graphic display devices, in the removal of the adverse effects of spatial sampling. The pseudo-anti-aliasing line drawing algorithm we propose is an extension to Bresenham's algorithm for computer control of a digital plotter [1]. While retaining the salient features of the original algorithm, the new algorithm does not reproduce a line as a sequence of disjoint line segments. The new algorithm produces a series of overlapping line segments where the display intensity shifts from one segment to the other in this overlap (transition region). True anti-aliased lines can be considered as having an overlapping behavior as well, but in these lines the rate of intensity shift and therefore the length of the overlap is a function of the slope of the true line. In this algorithm the length of the overlap and the intensity shift are essentially constants because the purpose of the transition region is an aid to the eye in integrating the segments into a single smooth line.

The anti-aliasing algorithm retains the following important features of Bresenham's algorithm:

Implementation of a Simple Anti-aliasing Algorithm

This is a brief description of an implementation of our new anti-aliasing line plotting algorithm for a 512x512 raster display. The purpose of this document is to explain how the original Bresenham algorithm was modified for anti-aliasing.

Because the line plotting routine places pixel codes in a frame buffer, the intensity information which will be used in the discussion of the algorithm will refer to the two low order bits of each pixel. Full intensity, white, pixels will have 11 as their low order bits. Black pixels will have 00 as their low order bits, and the algorithm calls for two intensities which represent two steps from black to white. The brighter of these two is the intermediate intensity (referred to as 66% in the program comments) and has 10 as its low order bits. The last intensity is the minimum intensity (33%) which has 01 as its low order bits. This numbering scheme allows the high order bits to represent color and allows new pixels to be ORed into memory. The possibility of accidently converting an intermediate pixel to a full intensity pixel by this ORing process and the visual effect this causes is so slight, that the use of additional bits or additional code to prevent this should not be considered.

Three program variables, FULL, IMED and IMIN, set prior to time generation, usually represent the full, intermediate (66%) and minimum (33%) intersities, respectively. IMIN will in one case (covered later) be set to the intermediate (66%) value; this is the only exception.

There are three constants used in the algorithm; these should be powers of two since they are used in multiplies and should be implemented as shifts. The first constant is the number of pixels in the overlaps of the axial time segments (stairsteps) generated by the Bresenham algorithm. This lap constant (denoted LAPCON (1) in the program listings) is used for lines which are constructed primarily of axial moves. These axial lines form an angle less than $\tan^{-1}(.5)$ with a vector aligned with the M1 move and are will execute an M1 move first (they have a negative initial decision variable). The second constant, also a lap constant (LAPCON (2)), is used to set a long overlap used on axial lines which form very shallow angles with the M1 axis. These lines have long runs of M1 moves and the longer overlap enhances their appearance, giving a better approximation of the true line. The lines using the second lap constant have a large difference in their Δa and Δb values (see Bresenham). The last constant (denote RATIO, and referred to as the aspect ratio) determines how great the difference between Δa and Δb must be to use the second lap constant. The following values are normally used for these constants:

Lap constant one	=	4 (pixels)
Lap constant two	=	16 (pixels)
Aspect ratio	=	32

If these constants are changed, their relative magnitudes must remain the same, that is, RATIO must be the largest and lap constant one must be the smallest and at least equal to two.

Execution of the algorithm begins with the normal computations used in Bresenham's algorithm. The octant is established, the M1 and M2 moves are set, Δa and Δb are set, and the initial value of the decision variable, ∇_1 (referred to as delta), is computed.

The next computations are set two test variables used to position the overlaps, set the actual intensities to be used, and in some cases change the value of Bresenham's delta. The two test variables (denoted ANTI2 and ANTI1) are used to locate the transition region's (overlap's) starting and midpoints, respectively. When the algorithm initially enters the transition region, it produces the overlap by outputting a minimum intensity in the M2 direction; it makes the M1 move; and puts out an intermediate intensity. At the midpoint, an intermediate intensity is used in the M2 direction and a minimum intensity in the M1 direction. Because the transition region is divided into two equal parts, after the computation of the midpoint test value (ANTI1), the start point test value (ANTI2) is always set equal to twice the midpoint value (a shift left of one).

After the full (FULL) and intermediate (IMED) values are set using their respective parameters, the initial decision variable, delta (DELTA) is check for a non-negative value. If delta is greater than or equal to zero, then the line is of a diagonal type (it will contain only singular M1 moves, if any). For these types of lines, ANTI1 is set to its minimum value, $-2\Delta b$, and the minimum intensity is set to the value of the intermediate parameter (66%). Processing then proceeds to the setting of ANTI2 and the generation of pixels.

For axial type lines, a series of three cases are checked to set ANTI1. First Δa is checked to see if it is greater than or equal to the aspect ratio (RATIO) times Δb. If it is, then ANTI1 is set to minus the long lap constant (LAPCON(2)) times Δb. If this first test fails, then ANTI1 is set to minus the first lap constant (LAPCON(1)) time Δb; this value of ANTI1 is

ANTI1 is now compared with the initial delta computed by the Bresenham algorithm. If ANTI1 is less than delta, it is set to its minimum value, $-2\Delta b$. Now that ANTI1 is set, the minimum (IMIN) intensity is set to the value of its parameter (33%). Next, ANTI1 is added to delta (DELTA) to shift the laps in axial type lines for symmetry. After ANTI2 is set, the generation of pixels can begin.

The initial value of delta is compared to ANTI2, if it is less than ANTI2 then the first pixel of the line is output at full intensity. Otherwise, the pixel is output at the intermediate level.

A count is now initialized using the Δa value and is decremented each time M1 or M2 loop is executed. As indicated by Bresenham, this value (Δa) is the number of moves necessary to generate the line. When the count reaches zero, line generation is complete. Comparing the current position with the true line endpoint will not always work. Although the endpoint will be output by the algorithm, it may be in a lap and never actually coincide with the position datum.

Line generation begins now with the same loop (M1 or M2) that it would for Bresenham's algorithm and will proceed until the count (mentioned above) reaches zero. The M2 loop is exactly the same as it is for the Bresenham algorithm with pixels output at full intensity. The M1 loop contains the additional code for anti-aliasing.

When the M1 loop is entered, delta is compared with ANTI2. If delta is less than ANTI2, then the M1 loop performs exactly as specified by Bresenham and output a new pixel at full intensity. Since delta is usually less, the test instruction is generally the only new instruction executed. The FORTRAN listing of the algorithm uses a slightly different M1 loop to keep track of position, but the result is the same.

If delta is greater than or equal to ANTI2, it is compared with ANTI1. If it is less than ANTI1, then the following sequence takes place. First a pixel of minimum intensity is output using the current (not updated) position plus the M2 move as its location. Then the position is updated with the M1 move and usual pixel is output but with an intermediate intensity. The loop completes execution by updating delta as specified by Bresenham. If delta is greater than or equal to ANTI1, then exactly the same sequence of operations takes place, except that the first pixel output has an intermediate intensity and the second has a minimum intensity.

Reference.

[1] J.E. Bresenham, "Algorithm for Computer Control of a Digital Plotter," IBM System J. 4, 1965.

```
      SUBROUTINE DRAWO(STARTX,STARTY,ENDX,ENDY)
C**********************************************************************
C
C     LINE PLOTTING ROUTINE USING STANDARD BRESENHAM ALGORITHM
C             - THIS SUBROUTINE DOES NOT DO ANY ANTIALIASING AND
C             ANY COMMENTS PERTAINING TO THAT MAY BE IGNORED.  IT
C             GENERATES ONLY FULL INTENSITY PIXELS (INTENS(1)).
C
C**********************************************************************
C
C             CALL PARAMETERS
C
C     STARTX,STARTY = X & Y COORDINATES OF LINE START POINT
C     ENDX,ENDY     = X & Y CORRDINATES OF LINE END POINT
C
C**********************************************************************
      IMPLICIT INTEGER*2(A-Z)
      COMMON RATIO,LAPCON(2),INTENS(3),FRAME(512,512),DIAG
C**********************************************************************
C
C             COMMON BLOCK PARAMETERS
C
C     FRAME   = PICTURE ARRAY
C     INTENS  = INTENSITY TABLE (VALUES AS FOLLOWS)
C               1 = FULL INTENSITY PIXEL CODE
C               2 = 66% INTENSITY PIXEL CODE
C               3 = 33% INTENSITY PIXEL CODE
C     RATIO   = ASPECT RATIO FOR ANTIALIASING ROUTINE, USED TO
C               DETERMINE SHALLOW AND STEEP (NEAR 45 DEG.) LINES
C               WHICH USE A LONGER LAP (SEE LAPCON)
C     LAPCON  = PIXEL LAP CONSTANT TABLE, USED TO SET THE NUMBER
C               PIXELS IN THE TRANSITION (LAP) REGION (VALUES AS
C               FOLLOWS)
C               1 = STANDARD LAP (INTERMEDIATE SLOPE LINES)
C               2 = LONG LAP (FOR SHALLOW AND STEEP LINES)
C
C**********************************************************************


C**********************************************************************
C
C     COMPUTE DELTA X AND Y AND SETUP OCTANT
C
C**********************************************************************
      DELX = ENDX - STARTX
      DELY = ENDY - STARTY
      DELXY = IABS(DELX) - IABS(DELY)
C  OCTANT 1 OR 2
      IF (DELX GE 0  AND  DELY GE 0) THEN
              M2X = 1
              M2Y = 1
C  OCTANT 1
              IF (DELXY GE 0) THEN
                      DELA = DELX
                      DELB = DELY
```

```
                                M1X = 1
                                M1Y = 0
C   OCTANT 2
                        ELSE
                                DELA = DELY
                                DELB = DELX
                                M1X = 0
                                M1Y = 1
                        END IF
C   OCTANT 3 OR 4
                ELSE IF (DELX.LT.0 .AND. DELY.GE.0) THEN
                        M2X = -1
                        M2Y = 1
C   OCTANT 4
                        IF (DELXY.GE.0) THEN
                                DELA = -DELX
                                DELB = DELY
                                M1X = -1
                                M1Y = 0
C   OCTANT 3
                        ELSE
                                DELA = DELY
                                DELB = -DELX
                                M1X = 0
                                M1Y = 1
                        END IF
C   OCTANT 5 OR 6
                ELSE IF (DELX.LT.0 .AND. DELY.LT.0) THEN
                        M2X = -1
                        M2Y = -1
C   OCTANT 5
                        IF (DELXY.GE.0) THEN
                                DELA = -DELX
                                DELB = -DELY
                                M1X = -1
                                M1Y = 0
C   OCTANT 6
                        ELSE
                                DELA = -DELY
                                DELB = -DELX
                                M1X = 0
                                M1Y = -1
                        END IF
C   OCTANT 7 OR 8
                ELSE
                        M2X = 1
                        M2Y = -1
C   OCTANT 8
                        IF (DELXY.GE.0) THEN
                                DELA = DELX
                                DELB = -DELY
                                M1X = 1
                                M1Y = 0
C   OCTANT 7
                        ELSE
                                DELA = -DELY
                                DELB = DELX
```

```
                        M1X = 0
                        M1Y = -1
                END IF
        END IF
C*****************************************************************************
C
C       SETUP LINE DRAWING ALGORITHM
C
C*****************************************************************************
C  SET UP CONSTANTS
        DEL2B = 2*DELB
        DEL2AB = 2*(DELB - DELA)
        DELTA = DEL2B - DELA
        OLDX = STARTX
        OLDY = STARTY
        FULL = INTENS(1)


C*****************************************************************************
C
C       DIAGNOSTIC ROUTINE
C
C*****************************************************************************
        IF (DIAG.GT.0) THEN        ! IN DIAGNOSTIC MODE
                WRITE(6,2030)
2030            FORMAT(' DRAWO SUBROUTINE - STANDARD BRESENHAM')
                WRITE(6,2032) DELA,DELB,DELTA
2032            FORMAT(' A=',I4,' B=',I4,' DELTA=',I5)
        END IF
C*****************************************************************************


C*****************************************************************************
C
C       DRAW THE LINE
C
C*****************************************************************************
C  OUTPUT THE STARTING POINT
        FRAME(OLDX,OLDY) = FULL
C  DRAW THE REMAINDER OF THE LINE
100     IF (DELA.GT.0) THEN                 ! DELA = NO. OF POSITIONS IN LIN
                IF (DELTA.LT.0) THEN
C  M1 MOVE
                        OLDX = OLDX + M1X
                        OLDY = OLDY + M1Y
                        FRAME(OLDX,OLDY) = FULL
                        DELTA = DELTA + DEL2B
                ELSE
C  M2 MOVE
                        OLDX = OLDX + M2X
                        OLDY = OLDY + M2Y
                        FRAME(OLDX,OLDY) = FULL
                        DELTA = DELTA + DEL2AB
                END IF
                DELA = DELA - 1           ! DECREMENT POSITION COUNT
                GOTO 100
        END IF
```

DRAWO

```
      C
      C        LINE DRAWING COMPLETED
      C
0092           RETURN
0093           END
```

```
      SUBROUTINE DRAW1(STARTX,STARTY,ENDX,ENDY)
C*****************************************************************************
C
C       LINE PLOTTING ROUTINE USING ANTIALIASING BRESENHAM ALGORITHM
C       DEVELOPED BY E. J. DUNNING.  THIS ROUTINE USES THE SIMPLE
C       VERSION OF THE ANTIALIASING ALGORITHM WHICH ONLY CREATES
C       TRANSITION REGION LAPPING ON LINES WHICH ARE PRIMARILY
C       AXIAL IN NATURE.  THESE LINES ARE IDENTIFIED BY A NEGATIVE
C       INITIAL DELTA VALUE.  COMMENTS IN THIS ROUTINE PERTAINING
C       TO ANTIALIASING STEEP LINES MAY BE IGNORED.  THE APPEARENCE
C       OF THE STEEP LINES IS ENHANCED, HOWEVER, BY THIS ROUTINE BY
C       USING 66% INTENSITY PIXEL VALUES IN THE M1 TRANSITIONS.
C
C                               BY E. JACK DUNNING
C
C*****************************************************************************
C
C               CALL PARAMETERS
C
C       STARTX,STARTY = X & Y COORDINATES OF LINE START POINT
C       ENDX,ENDY     = X & Y CORRDINATES OF LINE END POINT
C
C*****************************************************************************
      IMPLICIT INTEGER*2(A-Z)
      COMMON RATIO,LAPCON(2),INTENS(3),FRAME(512,512),DIAG
C*****************************************************************************
C
C               COMMON BLOCK PARAMETERS
C
C       FRAME    = PICTURE ARRAY
C       INTENS   = INTENSITY TABLE (VALUES AS FOLLOWS)
C                   1 = FULL INTENSITY PIXEL CODE
C                   2 = 66% INTENSITY PIXEL CODE
C                   3 = 33% INTENSITY PIXEL CODE
C       RATIO    = ASPECT RATIO FOR ANTIALIASING ROUTINE, USED TO
C                  DETERMINE SHALLOW AND STEEP (NEAR 45 DEG.) LINES
C                  WHICH USE A LONGER LAP (SEE LAPCON)
C       LAPCON   = PIXEL LAP CONSTANT TABLE, USED TO SET THE NUMBER
C                  PIXELS IN THE TRANSITION (LAP) REGION (VALUES AS
C                  FOLLOWS)
C                   1 = STANDARD LAP (INTERMEDIATE SLOPE LINES)
C                   2 = LONG LAP (FOR SHALLOW AND STEEP LINES)
C
C*****************************************************************************


C*****************************************************************************
C
C       COMPUTE DELTA X AND Y AND SETUP OCTANT
C
C*****************************************************************************
      DELX = ENDX - STARTX
      DELY = ENDY - STARTY
      DELXY = IABS(DELX) - IABS(DELY)
C  OCTANT 1 OR 2
```

```
            IF (DELX.GE.0 .AND. DELY.GE.0) THEN
                    M2X = 1
                    M2Y = 1
C   OCTANT 1
                    IF (DELXY.GE.0) THEN
                            DELA = DELX
                            DELB = DELY
                            M1X = 1
                            M1Y = 0
C   OCTANT 2
                    ELSE
                            DELA = DELY
                            DELB = DELX
                            M1X = 0
                            M1Y = 1
                    END IF
C   OCTANT 3 OR 4
            ELSE IF (DELX.LT.0 .AND. DELY.GE.0) THEN
                    M2X = -1
                    M2Y = 1
C   OCTANT 4
                    IF (DELXY.GE.0) THEN
                            DELA = -DELX
                            DELB = DELY
                            M1X = -1
                            M1Y = 0
C   OCTANT 3
                    ELSE
                            DELA = DELY
                            DELB = -DELX
                            M1X = 0
                            M1Y = 1
                    END IF
C   OCTANT 5 OR 6
            ELSE IF (DELX.LT.0 .AND. DELY.LT.0) THEN
                    M2X = -1
                    M2Y = -1
C   OCTANT 5
                    IF (DELXY.GE.0) THEN
                            DELA = -DELX
                            DELB = -DELY
                            M1X = -1
                            M1Y = 0
C   OCTANT 6
                    ELSE
                            DELA = -DELY
                            DELB = -DELX
                            M1X = 0
                            M1Y = -1
                    END IF
C   OCTANT 7 OR 8
            ELSE
                    M2X = 1
                    M2Y = -1
C   OCTANT 8
                    IF (DELXY.GE.0) THEN
                            DELA = DELX
```

```
                        DELB = -DELY
                        M1X = 1
                        M1Y = 0
C  OCTANT 7
                   ELSE
                        DELA = -DELY
                        DELB = DELX
                        M1X = 0
                        M1Y = -1
                   END IF
              END IF
C*****************************************************************************
C
C             SET-UP LINE DRAWING ALGORITHM
C
C*****************************************************************************
C  SET-UP CONSTANTS
              DEL2B = 2*DELB
              DEL2AB = 2*(DELB - DELA)
              DELTA = DEL2B - DELA
              OLDX = STARTX
              OLDY = STARTY
C  SET-UP ANTIALIASING
              FULL = INTENS(1)
              IMED = INTENS(2)
              IF (DELTA.GE.0) THEN                ! DIAGONAL TYPE LINES
                   ANTI1 = -DEL2B                 ! DEFAULT VALUE
                   IMIN = INTENS(2)               ! SET MIN TO MED INTENSITY
                   TYPE = 3
              ELSE                                ! AXIAL TYPE LINES
                   IF ((RATIO*DELB).LE.DELA) THEN  ! SHALLOW AXIAL
                        ANTI1 = -LAPCON(2) * DELB        ! LONG LAP
                        TYPE = 1
                   ELSE                           ! STANDARD AXIAL
                        ANTI1 = -LAPCON(1) * DELB        ! STANDARD LAP
                        IF (DELTA.GE.ANTI1) ANTI1 = -DEL2B
                             ! MINIMUM TRANSITION REGION (LAP) LENG
                        TYPE = 2
                   END IF
                   IMIN = INTENS(3)               ! STANDARD MINIMUM INTENSITY
                   DELTA = DELTA + ANTI1               ! CORRECT SYMMETRY
              END IF
              ANTI2 = ANTI1 * 2


C*****************************************************************************
C
C             DIAGNOSTIC ROUTINE
C
C*****************************************************************************
              IF (DIAG.GT.0) THEN        ! IN DIAGNOSTIC MODE
                   WRITE(6,2030)
2030               FORMAT(' DRAW1 SUBROUTINE - ANTIALIAS ALG. NO. 1')
                   WRITE(6,2032) DELA,DELB,DELTA,ANTI1,ANTI2,IMED,IMIN,TY
2032               FORMAT(' A=',I4,' B=',I4,' DELTA=',I5,' A1=',I5,
     *             ' A2=',I5,' MED=',I4,' MIN=',I4,' TYPE=',I2)
              END IF
```

```
C*************************************************************************


C*************************************************************************
C
C         DRAW THE LINE
C
C*************************************************************************
C  OUTPUT THE STARTING POINT
          IF (DELTA.LT.ANTI2) THEN
                  FRAME(OLDX,OLDY) = FULL
          ELSE
                  FRAME(OLDX,OLDY) = IOR(FRAME(OLDX,OLDY),IMED)
          END IF
C  DRAW THE REMAINDER OF THE LINE
100       IF (DELA.GT.0) THEN                  ! DELA = NO. OF POSITIONS IN LI
                  IF (DELTA.LT.0) THEN
C  M1 MOVE
                          NEWX = OLDX + M1X
                          NEWY = OLDY + M1Y
                          IF (DELTA.LT.ANTI2) THEN
                                  FRAME(NEWX,NEWY) = FULL
                          ELSE IF (DELTA.LT.ANTI1) THEN
                                  FRAME(NEWX,NEWY) =
     *                                    IOR(FRAME(NEWX,NEWY),IMED)
                                  FRAME(OLDX+M2X,OLDY+M2Y) =
     *                                    IOR(FRAME(OLDX+M2X,OLDY+M2Y),IMI
                          ELSE
                                  FRAME(NEWX,NEWY) =
     *                                    IOR(FRAME(NEWX,NEWY),IMIN)
                                  FRAME(OLDX+M2X,OLDY+M2Y) =
     *                                    IOR(FRAME(OLDX+M2X,OLDY+M2Y),IME
                          END IF
                          OLDX = NEWX
                          OLDY = NEWY
                          DELTA = DELTA + DEL2B
                  ELSE
C  M2 MOVE
                          OLDX = OLDX + M2X
                          OLDY = OLDY + M2Y
                          FRAME(OLDX,OLDY) = FULL
                          DELTA = DELTA + DEL2AB
                  END IF
                  DELA = DELA - 1          ! DECREMENT POSITION COUNT
                  GOTO 100
          END IF
C
C         LINE DRAWING COMPLETED
C
          RETURN
          END
```

Appendix B

MULTIPLIER ACCUMULATOR CARDS AND COORDINATE TRANSFORMATIONS

Alireza F. Faryar

Sarah A. Rajala

I. INTRODUCTION

The purpose of this report is a study of four previously built Multiplier Accumulator Cards (MAC) and their application to coordinate transformations. These cards are part of a real-time raster graphic display system used for generating raster graphic algorithms to be used in the cockpits of aircrafts. In this case the MAC will be used to perform coordinate transformations, such an example is illustrated in Figures 1 and 2. For real time operation this transformation must be very fast, and in Section II, it is shown that it can be done in 6.7 micro seconds. First, however, the hardware is discussed.

A. Multiplier Accumulator Cards Hardware:

These cards are powerful processing elements each containing a fast multiplier chip, a 32 bit ALU, input-output memories, and a microprogrammed controller. A simple block diagram is shown in Figure 3. In the following sections, different parts of the card and their performance are discussed. sed.

1. Busing:

Communication with each card is provided via three different buses, Address bus, Data bus and System Function bus.

(a) Address bus:

The address bus is a 24 bit wide and is used as follows:

AB00 through AB07 provide addresses corresponding to X and Y when initial data is being written into the input memories. AB8 specifies if we are writing into X or Y memory. AB16 controls the state of the latches (B1 to B4). MPLOD-L set low lets B7, B8, C7, C8, D7 and D8 be loaded from the data bus. If every input to B12 is in the proper mode MPLOD-L will go low, when CLK goes high and if AB16 is high. AB18 and AB19 address specific cards as follows:

AB 18-19	Address	Card No.
00		0
01		1
10		2
11		3

AB21-AB23 are an enable signal to (B15)*.

(b) Data bus:

The data bus is 32 bit bus used as follows:

DB00-DB23 are used to:

- Load the counters B7, B8, C7, C8, D7 and D8 with initial addresses of X, Y, Z when a function is to be performed. A new address is provided to these counters whenever MPLOD-L goes low (as directed in (a).
- If one is writing data into input memories, X and Y, MPLOD-L is high (latches are closed) and data are carried in through A1 to A8 bus drivers to the RAMS (this time this bus is providing input data).

---

* Chip number used in wiring diagram.

If a function is performed, the 32 bit output of Z memory is written on the entire 32 bit data bus, through A1-A8. The state of bus drivers A1-A8 is controlled by RDIM-H provided from (B14). When RDIM-H is high, the output of Z RAM is written on the bus. When it is low the data or the address on the bus is being read into the card.

(c) System Function bus:

The system function bus is a six bit bus consisting of:

Reset-Clock-F3-F2-F1-F0

F0,F1,F2 initialize micro-program counter through address memory (C12). This counter provides the address to the micro-program storage ROM's.

When F3 is low:

If AB8 = 0, AB16 = 0 input is being written into X RAM.

If AB8 = 1, AB16 = 0 input Y is to be written.

If AB8 = 0 and AB16 = 1 the state is changed, RDIM-H = 0.

The address will be provided to the address counters through the data bus. CLK and RESET are provided by the main system.

2. Microprogram Counter:

This section consists of a 256 bit bipolar PROM (C12) and two synchronous four bit binary counters. The carry-look ahead circuitry of the counters has made it possible to cascade them to get an eight bit synchronous output (C14 & C15). F0, F1 and F2 of the function code will address the PROM. It's output will be an appropriate address to the micro-code memory for executing a specific function. The counters are loaded with this address and as long as MAC is in this routine

MPINC-H (RUN-H) signal (one of the outputs of micro-program memory) enables the counters to increment with $\overline{CLK}$.

3. Microprogram Memory

The microcode* is stored in four 2048 bit (256x8) bipolar PROMs in order to construct a 256x32 bit memory. Address to this memory is provided by the microprogram counter. The output is used to provide appropriate signals for every function to be executed.

4. Card Decoder

The card decoder is a three to eight line decoder multiplexer (B15). One output is used in each card to indicate if the corresponding card has been selected. The inputs to the card decoder and corresponding pin number in each card are shown below:

AB 23 22 21	AB 19 18	pin#	MAC#
011	00	7	0
011	01	9	1
011	10	10	2
011	11	11	3

For example, if AB 18 and 19 are 10 then pin number 10 of (B15) in card number 2 will go low.

5. Bus Receiver-drivers

Except for the data bus and the card busy signal, hex-inverter interface elements are used to let a MAC communicate with the system buses. For the data bus and card busy signal Tri-state quad bus transceivers have been used.

---

* Lists of the microcode are given in Appendix A.

6. Latches

All latches are Hex D-type flip-flops with clear. Four of them (B1 to B4) are used to provide either starting addresses to X, Y and Z memories, when MPLOD-L is low, or to latch the input to the counters, when the data bus carries input data to input memories (MPLOD-L high).

Latches G8 to G12, E5 to E8 and part of E12 are used to provide the outputs of the multiplier chip and Z RAM to the arithmetic logic unit, whenever appropriate. They are controlled by signals provided by the microcodes.

7. Row-counters, Column-counters and Selectors

This circuit is explained for the X input. Similar circuits are used for Y and Z memories.

(a) If data is to be written in the X input memory, the address will be ready on the address bus and data will be provided by the data bus. In this case, selectors (B5 and B6) will select the address bus as address to the X RAM. This happens when WRIM-L is low. The data on the data bus are now inputs to X and they are prohibited from appearing on inputs of counters (B7 and B8) by MPLOD-L signal being high. Instead, they are read directly from the outputs of Bus-Transceivers into the X RAM. Signals XWREN-L specify whether the data is to be written in X or Y memory.

(b) If the data is ready in Y and X RAMS and a function is to be performed, MPLOD-L will go low providing a starting address to B7 and B8 counters. WRIM-L will be high so that the output of the counters will address the X-RAM. The counter will count with XLLOD-L signal provided by microcode.

(c) To read the data out and write it on the data bus, RDIM-L will go low and so will select the address bus as address to the Z RAM. This will provide the 32 bit long output of Z RAM to the bus transceivers A1 to A8. Since TE = RDIM-H is high the outputs will be written on the data bus.

8. Memory

There are three different types of memory used in each card, X, Y and Z, all the memory is made from TTL 256 X 4 bit fully decoded random access memory (93L422). The eight bit address to each of memory is usually treated as two 4 bit fields of row and column address. For example X(7,3) is stored in location 01110011.

(a) X-input memory
This is a 256 X 16 memory. The output is always enabled. The address to it is provided either by the address bus (in write mode when XWREN-L is low), or by the address counters B7 and B8 (when executing a function, XWREN-L is high).
The input to X is provided by the data bus, when it is in read mode.
In this case the following signals are provided:

RDIM-H low

MPLOD-L high   B1 to B4 are latched

AB8 = 0 therefore XWREN-L = low

The output of this memory is connected to multiplier chip.

(b) Y - input memory
This 256 X 16 RAM is addressed similar to X input memory. The only difference is that the output enable signal is not held at a fixed level. The multiplier chip MPY-16AJ (F1) receives Y inputs from the same pins where it delivers half of its output. Therefore, a signal called Tril is used to specify if the chip is reading or writing. The inverted Tril-H is used to enable the output of Y memory which is directly connected to multiplier chip. The input to Y is provided by the data bus as it was to X.

(c) Z - output memory
The Z output memory is a 256 X 32 bit random access memory. The address is provided either by the address counters (D7 & D*) when it is in reading mode, or by address bus when it is in write mode. It is in write mode when the write enable signal, ZWREN-L, is low. Its input is the output of arithmetic logic unit. The output enable signal is grounded so the output is always enabled if ZWREN-L is high. The output is provided both to the bus tranceivers (A1-A8) and the latches (E5-E8). The latter connects Z output back to the ALU.

9. Multiplier chip MPY-16J

The multiplier chip is a single chip on each MAC. This chip simply multiplies two 16 bit binary numbers and provides a 32 bit long product at its output. This is done in less than 200 n-sec.

In order to save space on the chip, one input shares pins with 16 bits of the output (LSP-out). A signal (TRIL) controls the flow of the data on these 16 bits. When Tril is high the output of the Y-RAM is provided to the multiplier and the chip will read in the data. When Tril is low the least significant protion of output will be ready at the output. The second input is provided by the X RAM. Every time a multiplication is done the output will go to the arithmetic logic unit for further arithmetic operations. Since the least significant portion of output and the Y input use the same pins, the output may not be directly connected to the arithmetic logic unit. The connection is made through a series of latches. Both multiplier chip and its latches are clocked by (MULAEN-H & MULCLK-H) provided in the microeode. For specifications see Appendix C.

10. Arithmetic Logic Unit

The ALU consists of eight ALU chips and three look-ahead carry generators (74s101 & 74s182). These two chips together can perform high speed arithmetic operations. Altogether, it is a 64 bit input-32 bit output ALU with a full carry look-ahead scheme. The inputs are provided by Z-RAM and multiplier chip. This circuit performs 16 binary arithmetic operations, with the functions and modes of operation selected through 5 inputs (one for mode of operation, 4 for function selection), which are provided in microcodes (ALUFO, 1,2,3, and ALUMO and ALUCO). The output is directly connected to the Z-RAM.

For ease of understanding a flow-chart of read and write operations is given in figure 4 and a timing diagram is shown in figure 5.

1. Subroutines

There are four different subroutines considered in a MAC. Every subroutine is called by an appropriate function code. The MAC halts after each operation and is ready for a new command. In the following routines X(0,0), Y(0,0), and Z(0,0) are all offset by the starting address.

2. Vector dot product

The following eight dot products are formed in 6.7 microseconds.

Z(0,0) = X(0,0)*Y(0,0)+X(0,1)*Y(1,0)+X(0,2)*Y(2,0)+X(0,3)*Y(3,0)

Z(1,0) = X(1,0)*Y(0,0)+X(1,1)*Y(1,0)+X(1,2)*Y(2,0)+X(1,3)*Y(3,0)

.
.
.

Z(7,0) = X(7,0)*Y(0,0)+X(7,1)*Y(1,0)+X(7,2)*Y(2,0)+X(7,3)*Y(3,0)

3. Perspective multiplication

The following sixteen pairs of products are formed in 6.7 microseconds.

Z(0,0) = X(0,0)*Y(0,0)

Z(0,1) = X(0,1)*Y(0,0)

Z(1,0) = X(1,0)*Y(1,0)

Z(1,1) = X(1,1)*Y(1,0)

-

-

-

Z(15,0) = X(15,0)*Y(15,0)

Z(15,1) = X(15,1)*Y(15,0)

4. Weighted Sum

The two following 4 X 4 weighted sums are formed in 6.7 microseconds.

$$Z(0,0) = \sum_{i=0}^{3} \sum_{j=0}^{3} X(i,j)*Y(i,j)$$

$$Z(1,0) = \sum_{i=0}^{3} \sum_{j=0}^{3} X(i,j)*Y(i-4,j)$$

5. Vector transformation

A 1X4 transformed matrix is formed by multiplying a 4X4 matrix (transformation matrix) with the 1X4 original vector, in 3.5 microseconds.

$$Z(0,0) = X(0,0)*Y(0,0)+X(0,1)*Y(1,0)+X(0,2)*Y(2,0)+X(0,3)*Y(3,0)$$

$$Z(0,1) = X(0,0)*Y(0,1)+X(0,1)*Y(1,1)+X(0,2)*Y(2,1)+X(0,3)*Y(3,1)$$

$$\vdots$$

$$Z(0,3) = X(0,0)*Y(0,3)+X(0,1)*Y(1,3)+X(0,2)*Y(2,3)+X(0,3)*Y(3,3)$$

6. Applications

As was previously mentioned one special application of these cards is the real time transformation from figure 1 to figure 2. This is done by transforming starting and ending points of each line in figure 1, using vector transformation routine. Prior to multiplication, a transformation matrix is created by concatenating a rotation matrix (figure 6.a) and a translation matrix (figure 6.b). This multiplication will transfer a point in three dimensional space to another point regarding new location and position of the aircraft. A separate routine is used to create a two-dimensional representation of the scene. Finally, the picture is drawn using a line drawing routine.

Another example, one can use a weighted summing to perform convolution. This can be done by proper selection of X(0,0), Y(0,0), Z(0,0).

IV. References

[1] S.A. Rajala, Progress Report for NASA-LRC Grant No. NSG-1355, November 1980.

[2] S.A. Rajala and E.J. Dunning, Progress Report for NASA-LRC Grant No. NSG-1355, December 1979.

[3] R.W. Stroh and J.N. England, Progress Report for NASA-LRC Grant No. NSG-1355, December 1978.

[4] J.E. Bresenham, "Algorithm for Computer Control of a Digital Plotter," IBM System J. 4, 1965.

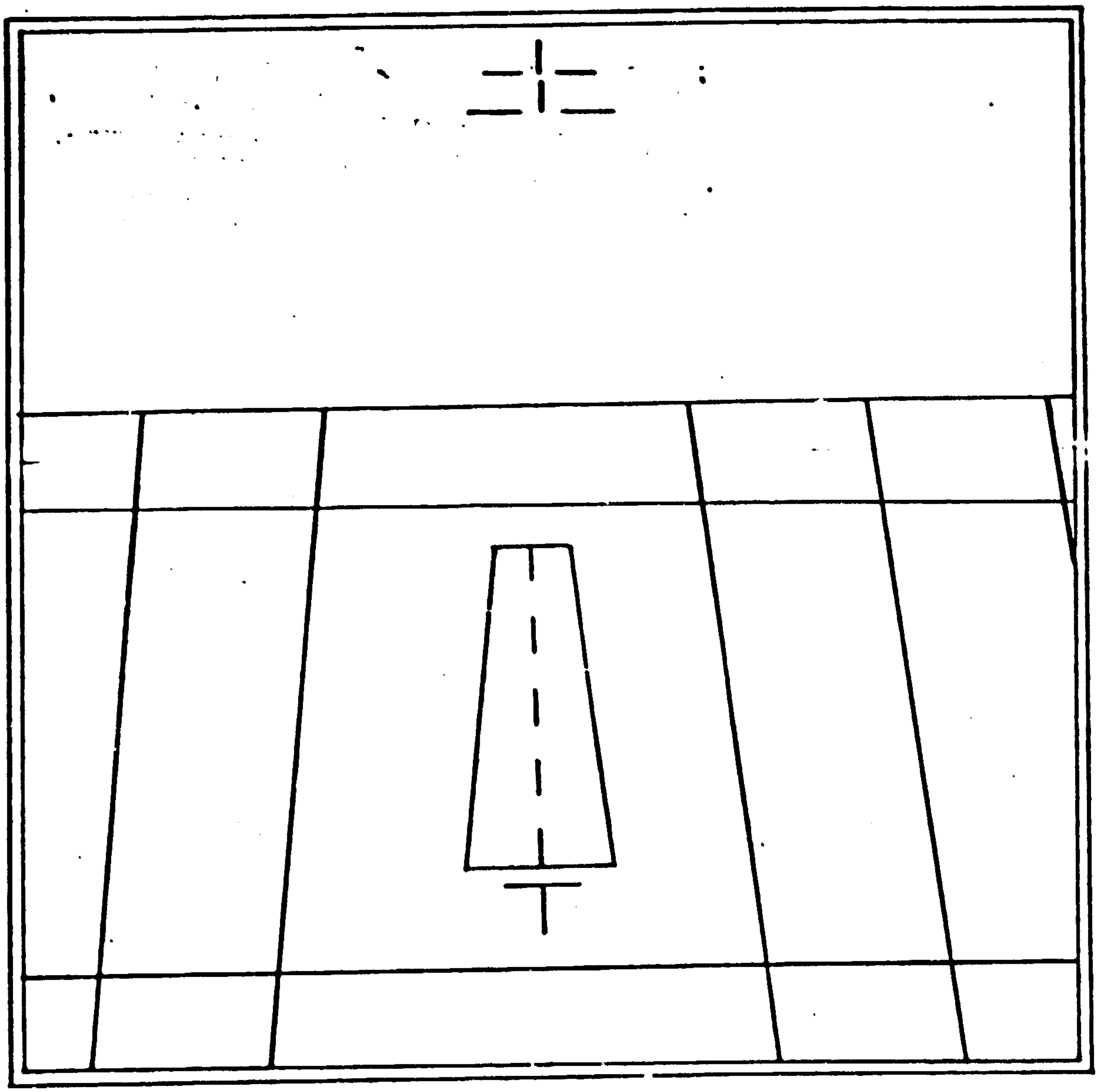


Figure 1- Example of the displayed picture by the system on board.
In this case the aircraft is in proper path for landing.

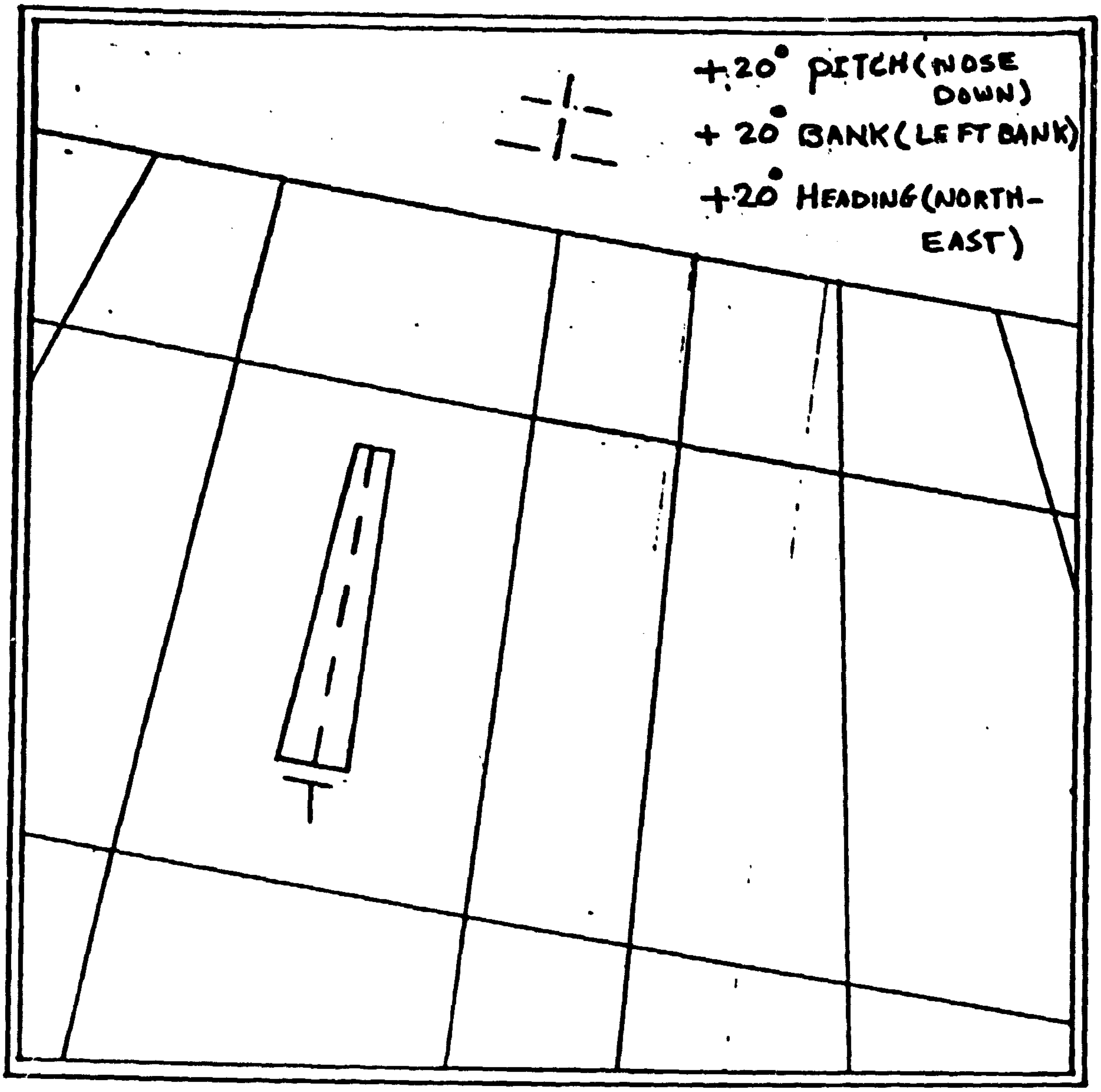


Figure 2- When the aircraft is not in proper path for landing.

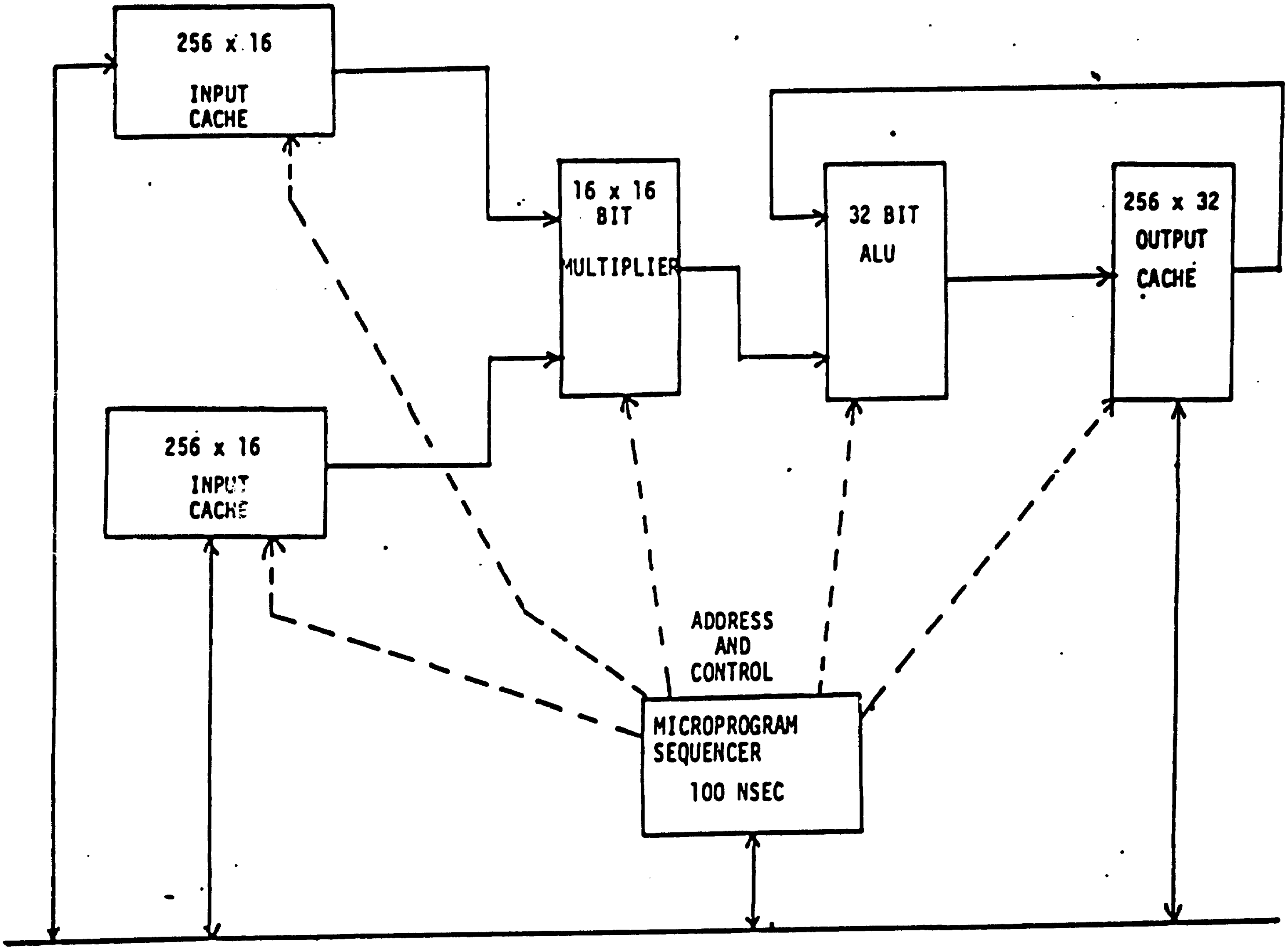
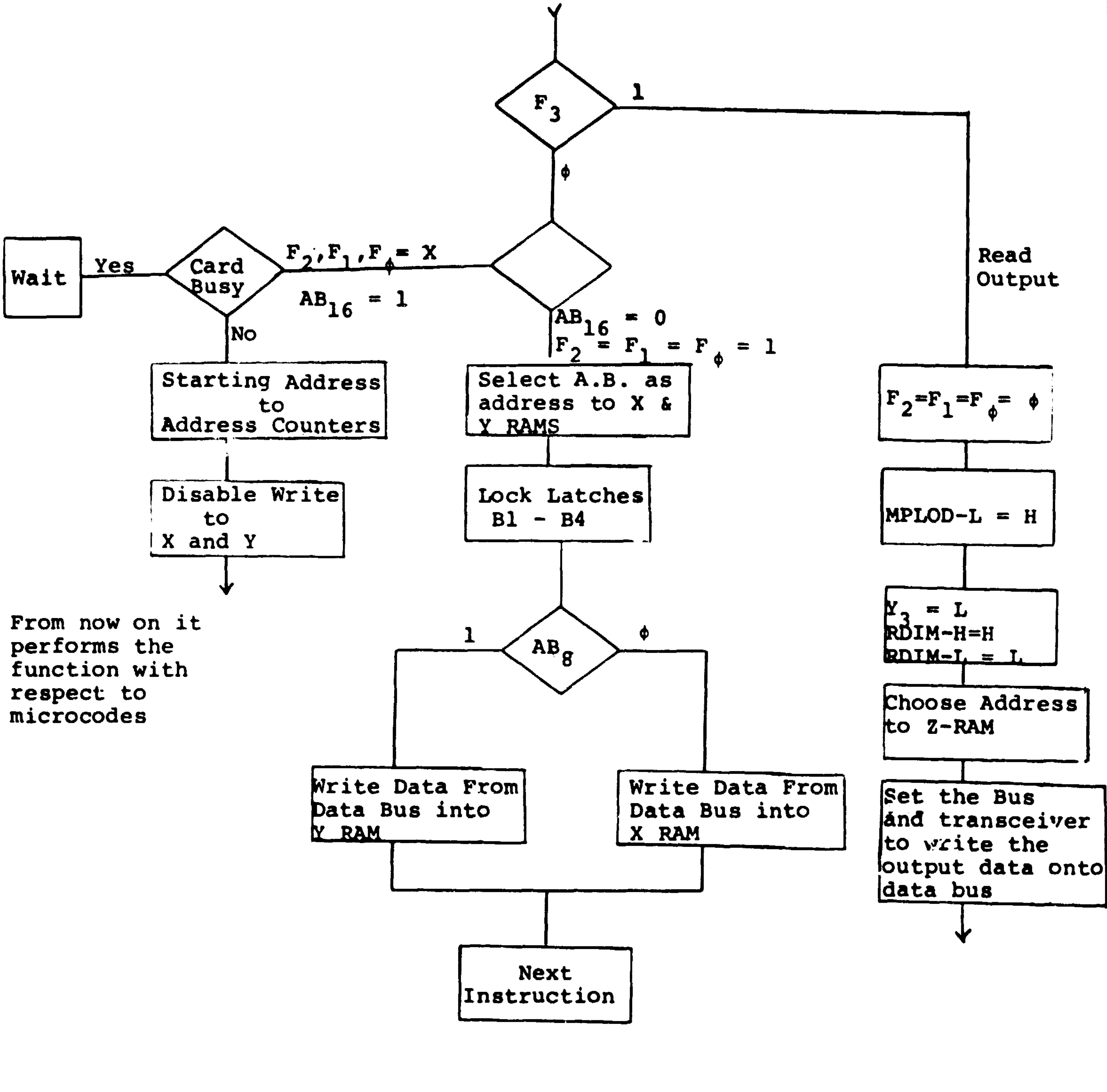


Figure 3- Multiplier Accumulator Card block diagram



From now on it performs the function with respect to microcodes

TIMING

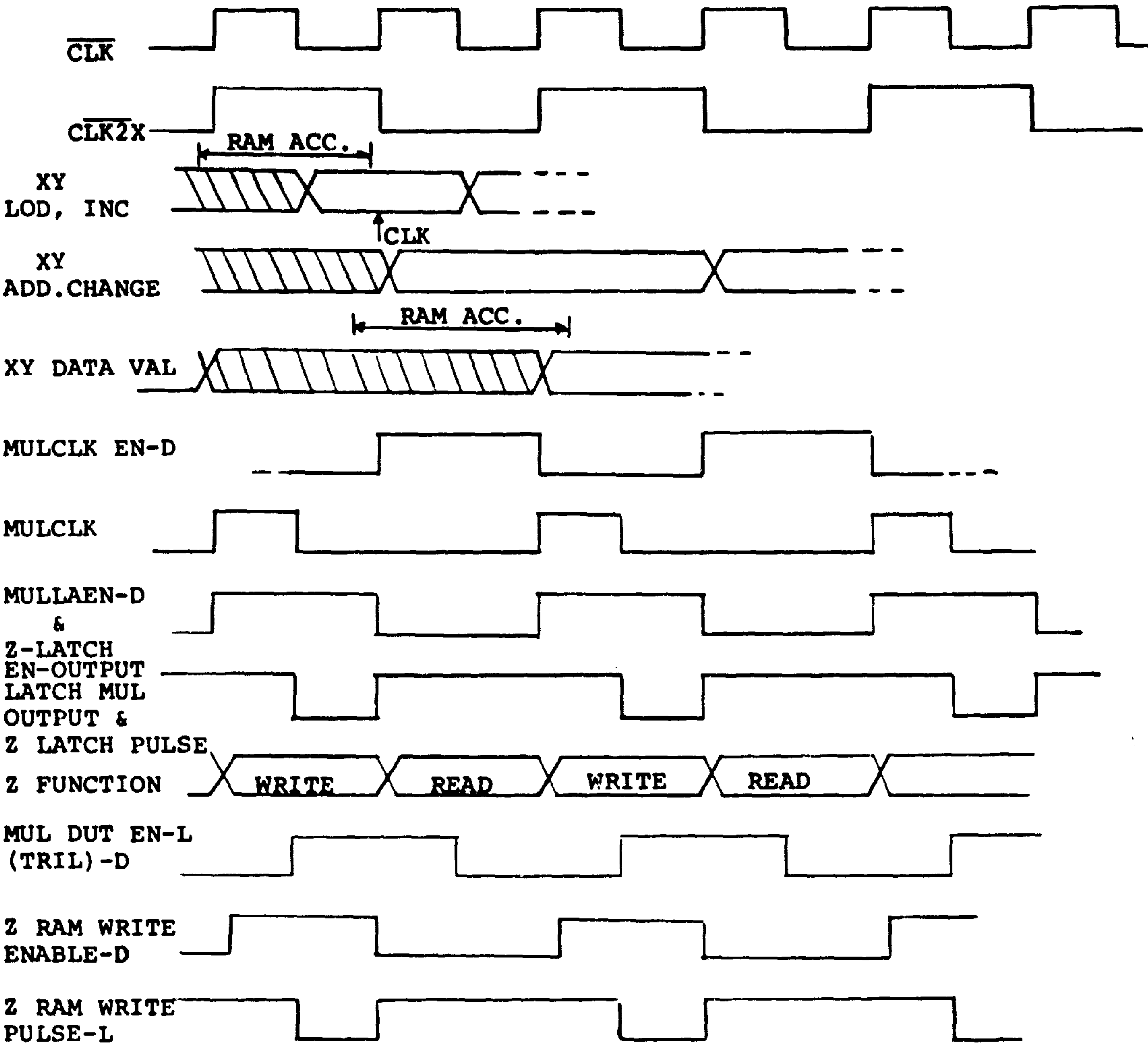


Figure 5 Timing Diagram

Cos H Cos B + Sin H Sin P Sin B	-Cos H Sin B + Sin H Sin P Cos B	Sin H Cos P	0
Cos P Sin B	Cos P Cos B	-Sin P	0
-Sin H Cos B + Cos H Sin P Sin B	Sin H Sin B + Cos H Sin P	Cos H Cos P	0
0	0	0	1

P= Pitch B= Bank H= Heading

Figure 6.a - Rotation matrix

1	0	0	0
0	1	0	0
0	0	1	0
X	Y	Z	1

Figure 6.b - Translation matrix

Appendix B.1

Microcode Table

TRAN... 1 x 1 MULTIPLIER MICRO CODE FORM F-0000 (1)

		34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULAEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	
	XLINC-H	0	0	1	0	1	0	1	0	0	0	1	0	1	
	XLLOD-L	0	1	1	1	1	1	1	1	0	1	1	1	1	
	XHINC-H	0	0	0	0	0	0	0	0	1	0	0	0	0	
	XHLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
	YLLOD-L	0	1	1	1	1	1	1	1	0	1	1	1	1	
	YHINC-H	0	0	1	0	1	0	1	0	0	0	1	0	1	
	YHLOD-L	0	1	1	1	1	1	1	1	0	1	1	1	1	
	ZLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZLLOD-L	0	1	1	1	1	1	1	1					1	
	ZHINC-H	0	0	0	0	0	0	0	0	0	0	0	1	0	
	ZHLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	0	0	0	0	0	1 P	1 P	1 P	1 P	1 P	1 P	0 A	0 A	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	0	0	0	0	0	1	1	1	1	1	1	0	0	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	0	0	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	0	0	0	0	0	1	0	1	0	1	0	1	0	MSB
	DRESS	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	

ORIGINAL PAGE IS OF POOR QUALITY

ORIGINAL PAGE IS OF POOR QUALITY

TRANSFORM

MULTIPLIER MICRO CODE FORM

	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULA EN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
	XLINC-H	0	1	0	0	0	1	0	1	0	1	0	0	0	1	
	XLLOD-L	1	1	1	0	1	1	1	1	1	1	1	0	1	1	
	XH INC-H	0	0	0	1	0	0	0	0	0	0	0	1	0	0	
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YL INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
	YLLOD-L	1	1	1	0	1	1	1	1	1	1	1	0	1	1	
	YH INC-H	0	1	0	0	0	1	0	1	0	1	0	0	0	1	
	YH LOD-L	1	1	1	0	1	1	1	1	1	1	1	0	1	1	
	ZL INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZH INC-H	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	1	1	1	1	1	1	0	0	1	1	1	1	1	1	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	1	1	1	1	1	1	0	0	1	1	1	1	1	1	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	MSB
	2E35	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	

1X1 TRANSFORM

MULTIPLY MICROCODE FORM

	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULA EN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
	XLINC-H	0	1	0	1	0	0	0	1	0	1	0	1	0	0	
	XLLOD-L	1	1	1	1	1	0	1	1	1	1	1	1	1	0	
	XH INC-H	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
	XH LOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YL INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
	YLLOD-L	1	1	1	1	1	0	1	1	1	1	1	1	1	0	
	YH INC-H	0	1	0	1	0	0	0	1	0	1	0	1	0	0	
	YH LOD-L	1	1	1	1	1	0	1	1	1	1	1	1	1	0	
	ZL INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZH INC-H	1	0	0	0	0	0	0	0	1	0	0	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	0	0	1	1	1	1	1	1	0	0	1	1	1	1	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	0	0	1	1	1	1	1	1	0	0	1	1	1	1	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	MSB
	ESS	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	

1X1 TRANSFORM

MULTIPLIER MICRO CODE FORM

(4)

		5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULA EN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
	XLINC-H	0	1	0	1	0	1	0	0	0	1	0	1	0	1	
	XLLOD-L	1	1	1	1	1	1	1	0	1	1	1	1	1	1	
	XH INC-H	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YL INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
	YLLOD-L	1	1	1	1	1	1	1	0	1	1	1	1	1	1	
	YHINC-H	0	1	0	1	0	1	0	0	0	1	0	1	0	1	
	YHLOD-L	1	1	1	1	1	1	1	0	1	1	1	1	1	1	
	ZLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZHINC-H	0	0	1	0	0	0	0	0	0	0	1	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	1	1	0	0	1	1	1	1	1	1	0	0	1	1	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	1	1	0	0	1	1	1	1	1	1	0	0	1	1	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	MSB
	(ADD)RESS	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	

TRANSFORM

MULTIPLIER MICRO CODE FORM

5

		6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77		
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	0		LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0		
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0		
(1D)	MULAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	0		
	XLINC-H	0	0	0	1	0	1	0	1	0	0	0	0	0		
	XLLOD-L	1	0	1	1	1	1	1	1	1	1	1	1	0		
	XHINC-H	0	1	0	0	0	0	0	0	0	0	0	0	0		
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0		MSB
	YLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0		LSB
	YLLOD-L	1	0	1	1	1	1	1	1	1	1	1	1	0		
	YHINC-H	0	0	0	1	0	1	0	1	0	0	0	0	0		
	YHLOD-L	1	0	1	1	1	1	1	1	1	1	1	1	0		
	ZLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0		
	ZLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0		
	ZHINC-H	0	0	0	0	1	0	0	0	0	0	0	0	0		
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0		MSB
(1D)	ALUFØ	1	1	1	1	0	0	1	1	1	1	1	[illegible]	0		LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0		
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0		
(1D)	ALUF3	1	1	1	1	0	0	1	1	1	1	1	1	0		
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	0		
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0		
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0		
(1D)	ZLAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	0		MSB
	RESS	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77		

1X1 TRANSFORMATION

[X Y Z W] A B C D = YB+ZC+WD

$X_2 Y_2 Z_2 W_2$

PTS. X HI	X LO	XFM Y HI	Y LO	OUTPUT Z HI	Z LO	ALU
LD	LD	LD	LD	LD	LD	XY
	INC	INC				PLUS
	INC	INC				PLUS
	INC	INC				PLUS
INC	LD	LD	LD	INC		XY
	INC	INC				PLUS
	INC	INC				PLUS
	INC	INC				PLUS
INC	LD	LD	LD	INC		XY
	INC	INC				PLUS
	INC	INC				PLUS
	INC	INC				PLUS
INC	LD	LD	LD	INC		XY
	INC	INC				PLUS
	INC	INC				PLUS
	INC	INC				PLUS
INC	LD	LD	LD	INC		XY
	INC	INC				PLUS
	INC	INC				PLUS
	INC	INC				PLUS
INC	LD	LD	LD	INC		XY
	INC	INC				PLUS
	INC	INC				PLUS
	INC	INC				PLUS
INC	LD	LD	LD	INC		XY
	INC	INC				PLUS
	INC	INC				PLUS
	INC	INC				PLUS
INC	LD	LD	LD	INC		XY
	INC	INC				PLUS
	INC	INC				PLUS
	INC	INC				PLUS

STOP

PERSPECTIVE

MULTIPLIER MICRO CODE Form

F = 0001

①

		0				1				2				3	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULAEN-H	0	0	0	1	0	1	0	1	0	1	0	1	0	D13
	XLINC-H	0	0	1	0	0	0	1	0	0	0	1	0	0	
	XLLOD-L	0	1	1	1	0	1	1	1	0	1	1	1	0	
	XHINC-H	0	0	0	0	1	0	0	0	1	0	0	0	1	
	XHLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
	YLLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	
	YHINC-H	0	0	0	0	1	0	0	0	1	0	0	0	1	
	YHLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	E13
	ZLINC-H	0	0	0	0	0	1	0	0	0	1	0	0	0	
	ZLLOD-L	0	1	1	1	1	1	1	0	1	1	1	0	1	
	ZHINC-H	0	0	0	0	0	0	0	1	0	0	0	1	0	
	ZHLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	0	0	0	0	0	0	0	0	0	0	0	0	0	E13
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	0	0	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	0	0	0	0	0	0	0	0	0	0	0	0	0	MSB
	DRESS	78	79	7A	7B	7C	7D	7E	7F	125 80	81	82	83	84	

PERSPECTIVE

MULTIPLIER MICRO CODE FORM ②

3 4 5 6

ORIGINAL PAGE IS OF POOR QUALITY

	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
	XLINC-H	0	1	0	0	0	1	0	0	0	1	0	0	0	1	
	XLLOD-L	1	1	1	0	1	1	1	0	1	1	1	0	1	1	
	XHINC-H	0	0	0	1	0	0	0	1	0	0	0	1	0	0	
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
	YLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	YHINC-H	0	0	0	1	0	0	0	1	0	0	0	1	0	0	
	YHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZLINC-H	1	0	0	0	1	0	0	0	1	0	0	0	1	0	
	ZLLOD-L	1	1	0	1	1	1	0	1	1	1	0	1	1	1	
	ZHINC-H	0	0	1	0	0	0	1	0	0	0	1	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MSB
	DRESS	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	

PERSPECTIVE MULTIPLIER CODE FORM

3

7 8 9 10

		93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1/2D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULA EN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	D13
	XLINC-H	0	0	0	1	0	0	0	1	0	0	0	1	0	0	
	XLLOD-L	1	0	1	1	1	0	1	1	1	0	1	1	1	0	
	XH INC-H	0	1	0	0	0	1	0	0	0	1	0	0	0	1	
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YL INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
	YLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	YH INC-H	0	1	0	0	0	1	0	0	0	1	0	0	0	1	
	YH LOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	E13
	ZL INC-H	0	0	1	0	0	0	1	0	0	0	1	0	0	0	
	ZLLOD-L	0	1	1	1	0	1	1	1	0	1	1	1	0	1	
	ZH INC-H	1	0	0	0	1	0	0	0	1	0	0	0	1	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	G13
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MSB
	DRESS	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	100 A0	

PERSPECTIVE MULTIPLIER MICRO CODE FORM

		10				11				12				13		
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
	XLINC-H	0	1	0	0	0	1	0	0	0	1	0	0	0	1	
	XLLOD-L	1	1	1	0	1	1	1	0	1	1	1	0	1	1	
	XHINC-H	0	0	0	1	0	0	0	1	0	0	0	1	0	0	
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
	YLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	YHINC-H	0	0	0	1	0	0	0	1	0	0	0	1	0	0	
	YHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZLINC-H	1	0	0	0	1	0	0	0	1	0	0	0	1	0	
	ZLLOD-L	1	1	0	1	1	1	0	1	1	1	0	1	1	1	
	ZHINC-H	0	0	1	0	0	0	1	0	0	0	1	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MSB
	ADDRESS	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	

PERSPECTIVE

MULTIPLIER MICRO CODE FORM

14 15 5

		AF	176 B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB			
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	0			LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0			
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0			
(1D)	MULA EN-H	1	0	1	0	1	0	1	0	1	0	1	0	0			
	XLINC-H	0	0	0	1	0	0	0	1	0	0	0	0	0			
	XLLOD-L	1	0	1	1	1	0	1	1	1	1	1	1	0			
	XH INC-H	0	1	0	0	0	1	0	0	0	0	0	0	0			
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0			MSB
	YL INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0			LSB
	YLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0			
	YH INC-H	0	1	0	0	0	1	0	0	0	0	0	0	0			
	YH LOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0			
	ZL INC-H	0	0	1	0	0	0	1	0	0	0	1	0	0			
	ZLLOD-L	0	1	1	1	0	1	1	1	0	1	1	1	0			
	ZH INC-H	1	0	0	0	1	0	0	0	1	0	0	0	0			
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0			MSB
(1D)	ALUFØ	0	0	0	0	0	0	0	0	0	0	0	0	0			LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0			
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0			
(1D)	ALUF3	0	0	0	0	0	0	0	0	0	0	0	0	0			
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	0			
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0			
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0			
(1D)	ZLAEN-H	0	0	0	0	0	0	0	0	0	0	0	0	0			MSB
	DRESS	AF	176 B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB			

PERSPECTIVE

XY W
XY W
XY W
W

X HI	X LO	Y HI	Y LO	Z HI	Z LO	ALU XY
LD	LD	LD	LD	LD	LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
INC	INC LD	INC		INC	INC LD	
	INC				INC	

STOP

EVALUATION | MULTIPLIER MICRO CODE | F =

Φ 1 (1)

Selector Signals of 74163

		BC	BD	BE	BF	C0	C1	C2	C3	C4	C5	C6	C7	C8	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULA EN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	
	XLINC-H	0	0	1	0	1	0	1	0	0	0	1	0	1	
	XLLOD-L	0	1	1	1	1	1	1	1	0	1	1	1	1	
	XH INC-H	0	0	0	0	0	0	0	0	1	0	0	0	0	
	XHLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YL INC-H	0	0	1	0	1	0	1	0	0	0	1	0	1	LSB
	YLLOD-L	0	1	1	1	1	1	1	1	0	1	1	1	1	
	YH INC-H	0	0	0	0	0	0	0	0	1	0	0	0	0	
	YH LOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	
	ZL INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZLLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	
	ZH INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZHLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	0	0	0	0	0	1	1	1	1	1	1	1	1	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	0	0	0	0	0	1	1	1	1	1	1	1	1	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	0	0	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	0	0	0	0	0	1	0	1	0	1	0	1	0	MSB
	...RESS	BC	BD	BE	BF	C0	C1	C2	C3	C4	C5	C6	C7	C8	

EVALUATION | MULTIPLIER MICRO CODE 2 3 (2)

	Signal	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
	XLINC-H	0	1	0	0	0	1	0	1	0	1	0	0	0	1	
	XLLOD-L	1	1	1	0	1	1	1	1	1	1	1	0	1	1	
	XHINC-H	0	0	0	1	0	0	0	0	0	0	0	1	0	0	
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YLINC-H	0	1	0	0	0	1	0	1	0	1	0	0	0	1	LSB
	YLLOD-L	1	1	1	0	1	1	1	1	1	1	1	0	1	1	
	YHINC-H	0	0	0	1	0	0	0	0	0	0	0	1	0	0	
	YHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZHINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	MSB
	ADRS	C9	CA	CB	CC	CD	CE	CF	205 DØ	D1	D2	D3	D4	D5	D6	

EVALUATION 7

MULTIPLIER MICROCODE FORM

	Signal	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	0	LSB
(1/2D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	
(1 1/2D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	
(1D)	MULAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	0	
	XLINC-H	0	0	0	1	0	1	0	1	0	0	0	0	0	
	XLLOD-L	1	0	1	1	1	1	1	1	1	1	1	1	0	
	XHINC-H	0	1	0	0	0	0	0	0	0	0	0	0	0	
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0	MSB
	YLINC-H	0	0	0	1	0	1	0	1	0	0	0	0	0	LSB
	YLLOD-L	1	0	1	1	1	1	1	1	1	1	1	1	0	
	YHINC-H	0	1	0	0	0	0	0	0	0	0	0	0	0	
	YHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0	
	ZLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0	
	ZHINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	0	MSB
(1D)	ALUFØ	1	1	1	1	1	1	1	1	1	1	1	1	0	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	1	1	1	1	[illegible]	1	1	1	1	1	1	1	0	
(1D)	ALUCØ-L	1	1	1	1	1	[illegible]	[illegible]	[illegible]	1	1	1	1	0	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	
(1D)	ZLAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	0	MSB

EVALUATION

MULTIPLIER MICRO CODE ocm

4

		E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
	XLINC-H	0	1	0	1	0	1	0	0	0	1	0	1	0	1	
	XLLOD-L	1	1	1	1	1	1	1	0	1	1	1	1	1	1	
	XHINC-H	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YLINC-H	0	1	0	1	0	1	0	0	0	1	0	1	0	1	LSB
	YLLOD-L	1	1	1	1	1	1	1	0	1	1	1	1	1	1	
	YHINC-H	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
	YHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZHINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	MSB
	ES	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	240 F0	F1	F2	

LUATION | MULTIPLIER MICRO CODE FORM 4 | 5 (5)

RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D) TRILEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D) MULCLKEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D) MULAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
XLINC-H	0	1	0	1	0	0	0	1	0	1	0	1	0	0	
XLLOD-L	1	1	1	1	1	0	1	1	1	1	1	1	1	0	
XHINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
XHLOD-L	1	1	1	1	1	0	1	1	1	1	1	1	1	1	MSB
YLINC-H	0	1	0	1	0	0	0	1	0	1	0	1	0	0	LSB
YLLOD-L	1	1	1	1	1	0	1	1	1	1	1	1	1	0	
YHINC-H	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
YHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
ZLINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ZLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
ZHINC-H	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D) ALUFØ	1	1	1	1	1	1	1	1	1	1	0	0	1	1	LSB
(1D) ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D) ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D) ALUF3	1	1	1	1	1	1	1	1	1	1	0	0	1	1	
(1D) ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	[illegible]	[illegible]	
(1D) ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D) ZWRITE-H	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D) ZLAEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	MSB
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	

ORIGINAL PAGE IS OF POOR QUALITY

CONTROL PT. X | CO-EFF Y | OUTPUT Z | ALU

X HI	X LO	Y HI	Y LO	Z HI	Z LO	ALU
LD	LD	LD	LD	LD	LD	XY PLUS
	INC		INC			
	INC		INC			
	INC		INC			
INC	LD	INC	LD			
	INC		INC			
	INC		INC			
	INC		INC			
INC	LD	INC	LD			
	INC		INC			
	INC		INC			
	INC		INC			
INC	LD	INC	LD			
	INC		INC			
	INC		INC			
	INC		INC			
LD	LD	INC	LD	INC		XY PLUS
	INC		INC			
	INC		INC			
	INC		INC			
INC	LD	INC	LD			
	INC		INC			
	INC		INC			
	INC		INC			
INC	LD	INC	LD			
	INC		INC			
	INC		INC			
	INC		INC			
INC	LD	INC	LD			
	INC		INC			
	INC		INC			
	INC		INC			

END

$X_1, X_2, X_3, X_4, Y_1, Y_2, Y_3, Y_4, Z_1, Z_2, Z_3, Z_4, W_1, W_2, W_3, W_4$

X_5, X_6, X_7, X_8

OK but depends on

X X X X ...
X X X X
X X X X
X X X X
.
.

not

XYZW XYZW
XYZW XYZW
. .
. .

4x4 TRANSFORM | MULTIPLIER MICRO CODE | F= 0111 | 1

		10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
(1½D)	MULCLKEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
(1D)	MULA EN-H	0	0	0	1	0	1	0	1	0	1	0	1	0	1	
	XLINC-H	0	0	1	0	1	0	1	0	0	0	1	0	1	0	
	XLLOD-L	0	1	1	1	1	1	1	1	0	1	1	1	1	1	
	XH INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	XHLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YL INC-H	0	0	0	0	0	0	0	0	1	0	0	0	0	0	LSB
	YLLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
	YH INC-H	0	0	1	0	1	0	1	0	0	0	1	0	1	0	
	YH LOD-L	0	1	1	1	1	1	1	1	0	1	1	1	1	1	
	ZL INC-H	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
	ZLLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZH INC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZHLOD-L	0	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	0	0	0	0	0	1	1	1	1	1	1	0	0	1	LSB
(1D)	ALUF1	0	0	0	0	0	0 PLUS	0 PLUS	0 PLUS	0 PLUS	0 PLUS	0 PLUS	0	0	0 PLUS	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	0	0	0	0	0	1	1	1	1	1	1	0	0	1	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALU MO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	Z WRITE-H	0	0	0	0	1	0	1	0	1	0	1	0	1	0	
(1D)	ZLAEN-H	0	0	0	1	0	1	0	1	0	1	0	1	0	1	MSB
	ADDRESS	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	

4X4 TRANSFORM | MULTIPLIER MICRO C FORM

(V)

		1E	1F	20	21	22	23	24	25	26	27	28	29	2A	
	RUN-H	1	1	1	1	1	1	1	1	1	1	1	1	1	LSB
(1½D)	TRILEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1½D)	MULCLKEN-H	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	MULAEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	D13
	XLINC-H	1	0	0	0	1	0	1	0	1	0	0	0	1	
	XLLOD-L	1	1	0	1	1	1	1	1	1	1	0	1	1	
	XHINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
	XHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
	YLINC-H	0	0	1	0	0	0	0	0	0	0	1	0	0	LSB
	YLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	
	YHINC-H	1	0	0	0	1	0	1	0	1	0	0	0	1	
	YHLOD-L	1	1	0	1	1	1	1	1	1	1	0	1	1	E13
	ZLINC-H	0	0	0	0	0	1	0	0	0	0	0	0	0	
	ZLLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	
	ZHINC-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	1	1	1	1	1	MSB
(1D)	ALUFØ	1	1	1	1	1	0	0	1	1	1	1	1	1	LSB
		PLUS	PLUS	PLUS	PLUS	PLUS	A	A	ACUS	PLUS	PLUS	PLUS	PLUS	PLUS	
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	1	1	1	1	1	0	0	1	1	1	1	1	1	G13
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	1	1	1	1	1	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	1	0	1	0	1	0	1	0	1	0	1	0	1	
(1D)	ZLAEN-H	0	1	0	1	0	1	0	1	0	1	0	1	0	MSB
	DRESS	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	

4x4 TRANSFORM · MULTIPLIER MICRO CODE F..

	Signal	2B	2C	2D	2E	2F	30	31	32	33	
	RUN-H	1	1	1	1	1	1	1	1	0	LSB
(1½D)	TRILEN-H	0	1	0	1	0	1	0	1	0	
(1½D)	MULCLKEN-H	0	1	0	1	0	1	0	0	0	
(1D)	MULA EN-H	1	0	1	0	1	0	1	0	0	
	XLINC-H	0	1	0	1	0	0	0	0	0	
	XLLOD-L	1	1	1	1	1	1	1	1	0	
	XH INC-H	0	0	0	0	0	0	0	0	0	
	XHLOD-L	1	1	1	1	1	1	1	1	0	MSB
	YL INC-H	0	0	0	0	0	0	0	0	0	LSB
	YLLOD-L	1	1	1	1	1	1	1	1	0	
	YH INC-H	0	1	0	1	0	0	0	0	0	
	YH LOD-L	1	1	1	1	1	1	1	1	0	
	ZL INC-H	1	0	0	0	0	0	0	0	0	
	ZLLOD-L	1	1	1	1	1	1	1	1	0	
	ZH INC-H	0	0	0	0	0	0	0	0	0	
	ZHLOD-L	1	1	1	1	1	1	1	1	0	MSB
(1D)	ALUFØ	0	0	1	1	1	1	1	1	0	LSB
(1D)	ALUF1	0	0	0	0	0	0	0	0	0	
(1D)	ALUF2	0	0	0	0	0	0	0	0	0	
(1D)	ALUF3	0	0	1	1	1	1	1	1	0	
(1D)	ALUCØ-L	1	1	1	1	1	1	1	1	0	
(1D)	ALUMO	0	0	0	0	0	0	0	0	0	
(1D)	ZWRITE-H	0	1	0	1	0	1	0	1	0	
(1D)	ZLAEN-H	1	0	1	0	1	0	1	0	0	MSB
	ADDRESS	2B	2C	2D	2E	2F	30	31	32	33	

MAT [illegible]

X		Y		Z		ALU
HI	LOW	HI	LOW	HI	LOW	F=000
LD	LD	LD	LD	LD	LD	XY
—	INC	INC	—	—	—	ADD
—	INC	INC	—	—	—	ADD
—	INC	INC	—	—	—	ADD
—	LD	LD	INC	—	INC	XY
—	INC	INC	—	—	—	ADD
—	INC	INC	—	—	—	ADD
—	INC	INC	—	—	—	ADD
—	LD	LD	INC	—	INC	XY
—	INC	INC	—	—	—	ADD
—	INC	INC	—	—	—	ADD
—	INC	INC	—	—	—	ADD
—	LD	LD	INC	—	INC	XY
—	INC	INC	—	—	—	ADD
—	INC	INC	—	—	—	ADD
—	INC	INC	—	—	—	ADD

32+ LOCS

Appendix B.2

Multiplier Chip Specification

MPY-16AJ

16 x 16 BIT PARALLEL MULTIPLIER

See Electronic Design 14, Sept. 13, 1978 p. 98
TRW also has applications notes on this device also —

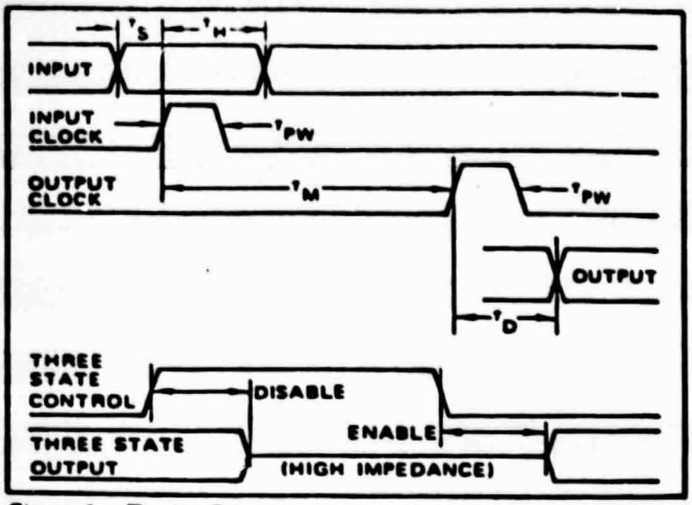


Figure 1. Timing Diagram

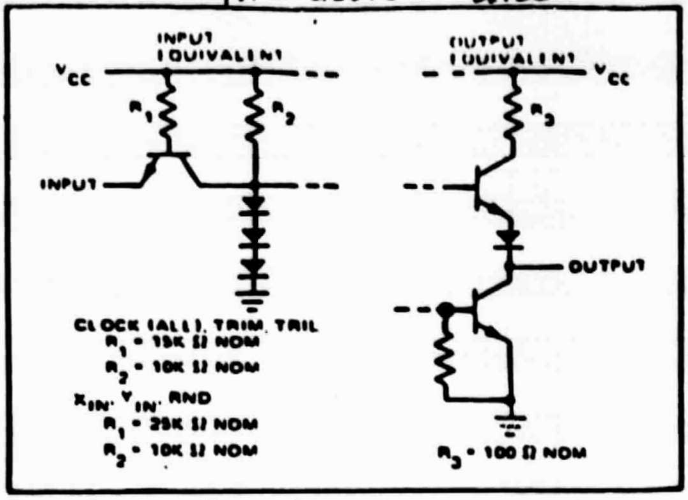


Figure 2. Input/Output Schematics

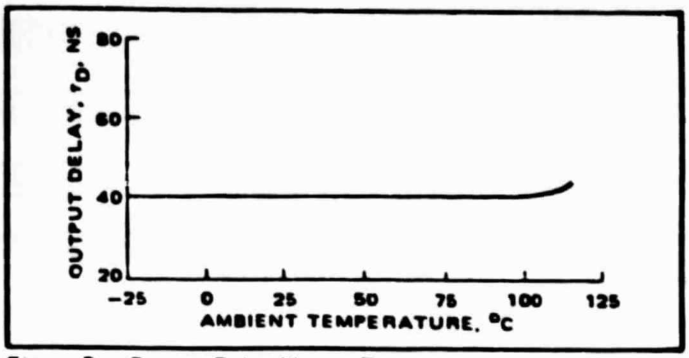


Figure 3. Output Delay Versus Temperature

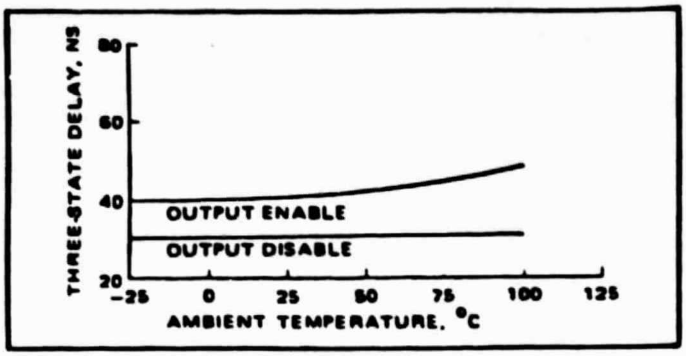


Figure 4. Three State Delay Versus Temperature

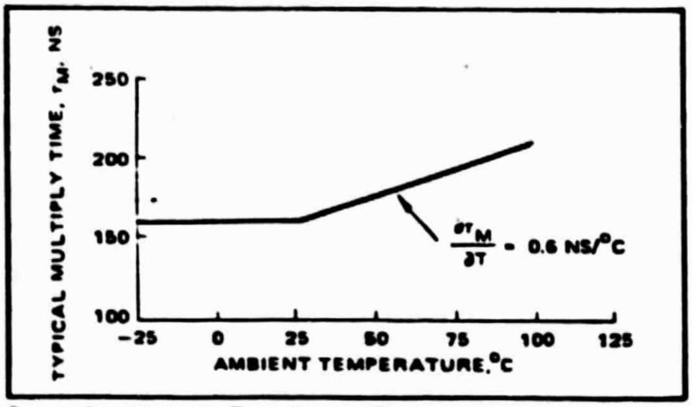


Figure 5 Multiply Time Versus Temperature

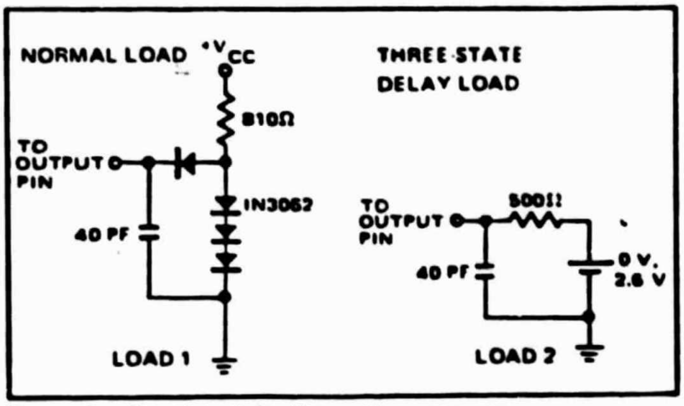


Figure 6 Test Loads for Delay Measurements

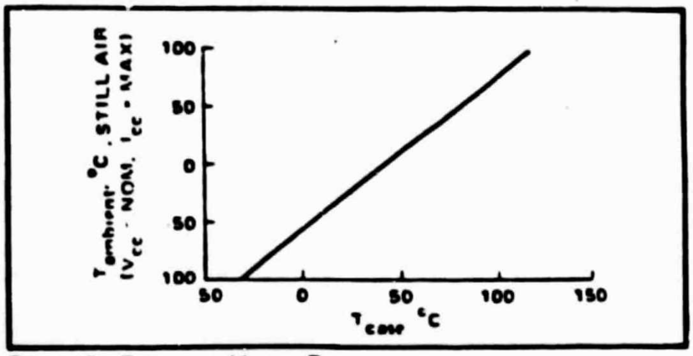


Figure 7. $T_{ambient}$ Versus T_{case}

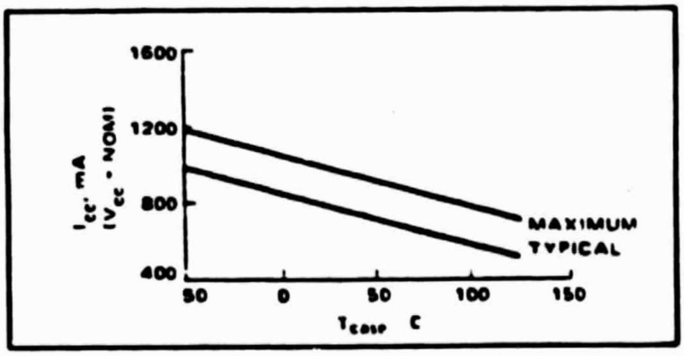


Figure 8. I_{CC} versus T_{case}

absolute maximum ratings over operating temperature range

Supply voltage . . . . −0.5 to 7.0 V
Input voltage . . . . 0 to 5.5 V
Output voltage . . . . 0 to 5.5 V
Operating temperature range: MPY-16AJ ($T_{ambient}$) . . . . 0°C to 70°C
MPY-16AJ (T_{case}) . . . . −55°C to 125°C
Storage temperature range . . . . −65°C to 150°C
Lead temperature (10 seconds) . . . . 300°C
Junction temperature . . . . 175°C

recommended operating conditions

	MPY-16AJ			MPY-16AJ-M			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5.0	5.5	4.5	5.0	5.5	V
Clock pulse width (measured at 1.5 V level)	20			20			ns
Input register setup time, τ_S (see Figure 1)	-5.0			-5.0			ns
Input register hold time, τ_H (see Figure 1)	20			20			ns
Operating ambient temperature (see Note 1)	0		70	−55		125	°C

NOTES: 1. MPY-16AJ: $T_{ambient}$, MPY-16AJ-M: T_{case}

electrical characteristics over recommended temperature range

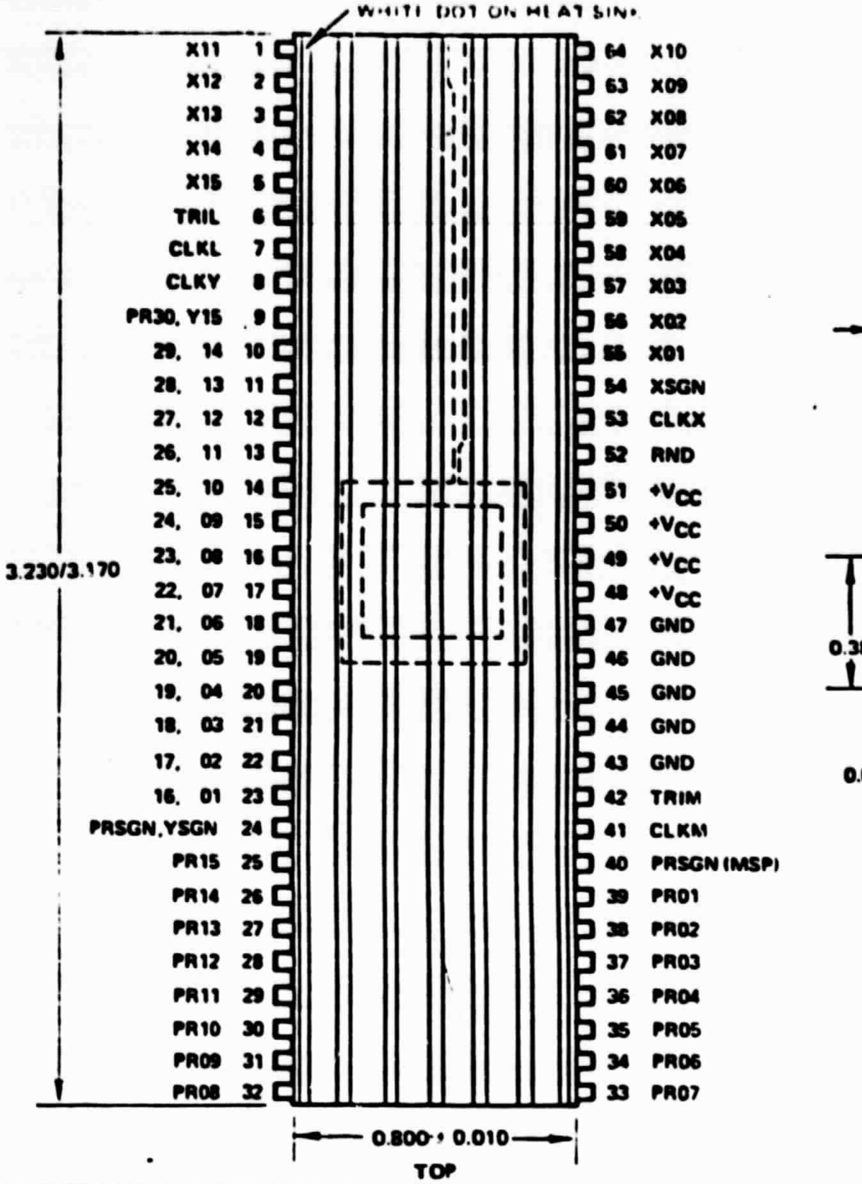
PARAMETER		TEST CONDITIONS		MPY-16AJ			MPY-16AJ-M			UNIT
				MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage			2.0			2.0			V
V_{IL}	Low-level input voltage					0.8			0.8	V
V_{OH}	High-level output voltage	V_{CC} = NOM,	I_{OH} = −0.4 mA	2.4	3.2		2.4	3.2		V
V_{OL}	Low-level output voltage	V_{CC} = MIN,	I_{OL} = 4.0 mA		0.3	0.5		0.3	0.5	V
I_{IH}	High-level input current	V_{CC} = MAX,	V_{IH} = 2.4			75			80	µA
I_{IL}	Low-level input current	V_{CC} = MAX,	V_{IL} = 0.4			−0.75			−1.0	mA
I_{CC}	Supply current	V_{CC} = NOM			800	1000		800	1200	mA

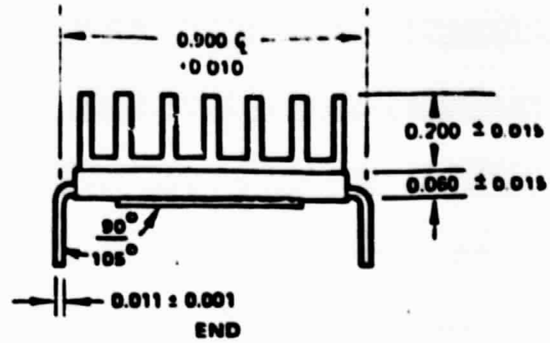
‡At $T_{ambient}$ = 25°C, V_{CC} = NOM.

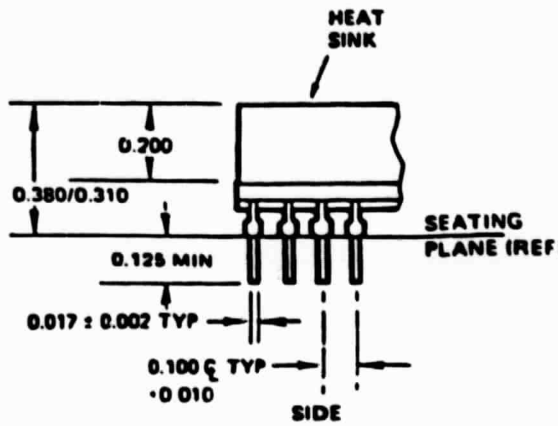
switching characteristics, V_{CC} = 5.0, T_A = 25°C (see Figure 1)

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Multiply time, input register clock To output register clock, τ_m	See Figure 5		160	200	ns
Output delay τ_D	Load 1, see Figures 3, 6		40	50	ns
Three state output delay					
Output enable	Load 2, see Figures 4, 6		40	50	ns
Output disable	Load 2, see Figures 4, 6		30	40	ns

PACKAGE INFORMATION



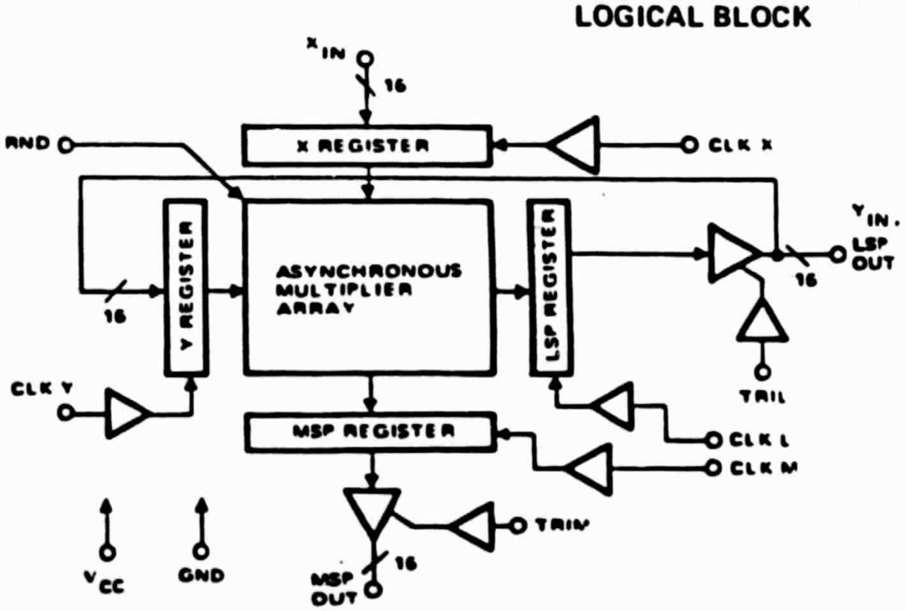




(DIMENSIONS IN INCHES)

NOTE: ALL V_{CC} AND GND PINS MUST BE CONNECTED

LOGICAL BLOCK



CONTROLS

CLK X, X_{IN} REGISTER CLOCK

CLK Y, Y_{IN} REGISTER CLOCK

CLK L, LSP REGISTER CLOCK

CLK M, MSP REGISTER CLOCK

TRIL, LSP THREE STATE CONTROL

TRIM, MSP THREE STATE CONTROL

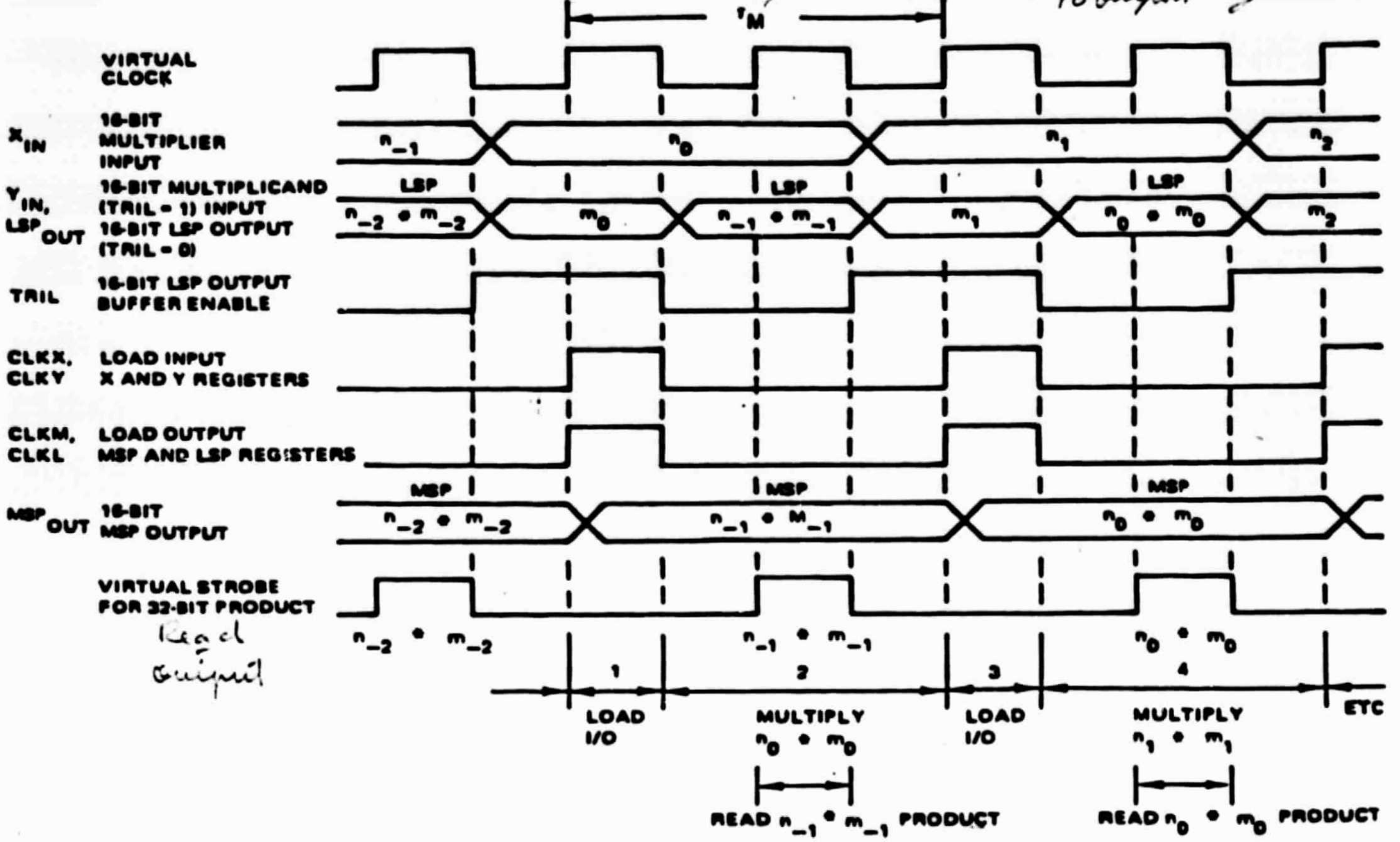
RND, ADDS 2^{-16} TO PRODUCT
(FRACTIONAL 2'S COMPLEMENT FIELD-
SEE PAGE 13 FOR I/O FORMAT)

TYPICAL OPERATING SEQUENCE (3 PORT)

1. LOAD 16 BIT MULTIPLIER (n_0) AND 16 BIT MULTIPLICAND (m_0) INTO X AND Y INPUT REGISTERS, RESPECTIVELY. SIMULTANEOUSLY LOAD OUTPUT REGISTERS WITH THE PRODUCT OF TWO PREVIOUS OPERANDS, n_{-1} AND m_{-1}
2. WAIT FOR COMPLETION OF $n_0 \bullet m_0$ MULTIPLICATION. READ PRODUCT OF PREVIOUS OPERANDS
3. LOAD MSP AND LSP OUTPUT REGISTERS WITH PRODUCT OF n_0 AND m_0. SIMULTANEOUSLY LOAD INPUT REGISTERS WITH n_1 AND m_1.
4. READ PRODUCT OF n_0 AND m_0.

NOTE: TRIM = RND = 0 DURING ABOVE SEQUENCE.

multiplying time . input register clock
To output register clock



INPUT/OUTPUT FORMAT FOR FRACTIONAL 2'S COMPLEMENT FIELD

X_{IN}

X SGN	X_1	X_2	X_3	...	X_{13}	X_{14}	X_{15}
	2^{-1}	2^{-2}	2^{-3}		2^{-13}	2^{-14}	2^{-15}

Y_{IN}

Y SGN	Y_1	Y_2	Y_3	...	Y_{13}	Y_{14}	Y_{15}
	2^{-1}	2^{-2}	2^{-3}		2^{-13}	2^{-14}	2^{-15}
	2^{-16}	2^{-17}	2^{-18}		2^{-28}	2^{-29}	2^{-30}

PR SGN	PR_1	PR_2	PR_3	...	PR_{13}	PR_{14}	PR_{15}	PR SGN	PR_{16}	PR_{17}	PR_{18}	...	PR_{28}	PR_{29}	PR_{30}

MSP — LSP

0 → positive
1 → neg

$$X = -1 \bullet X_{SIGN} + \sum_{n=1}^{15} X_n 2^{-n} \qquad PR = -1 \bullet PR_{SIGN} + \sum_{n=1}^{30} PR_n 2^{-n}$$

THE RESULTING VALUES FOR X AND PR GIVEN IN THE ABOVE EVALUATIONS (Y IS EXPRESSED IN THE SAME MANNER AS X) ARE IN FRACTIONAL 2's COMPLEMENT FORMAT. THE VALUE FOR THE SIGN VARIABLE IS 0 FOR POSITIVE OR ZERO NUMBERS AND 1 FOR NEGATIVE NUMBERS.

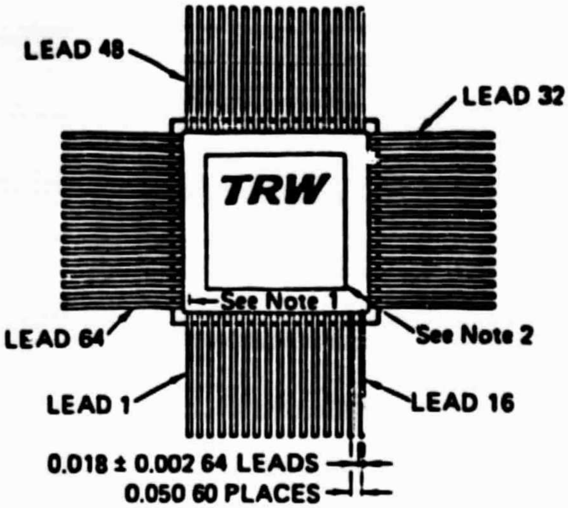
AN OVERFLOW OCCURS IN THE ATTEMPTED MULTIPLICATION OF THE 2'S COMPLEMENT NUMBER 10000 ... (-1 BASE 10) WITH ITSELF, YIELDING A RESULT OF THE SAME NUMBER, I.E.

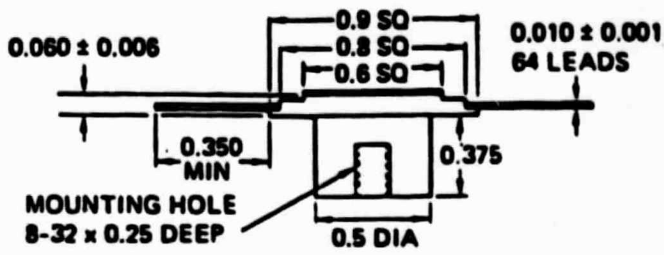
$$(-1)_{10} \bullet (-1)_{10} = (-1)_{10}$$

THE PRODUCT SIGN BIT IS AVAILABLE REDUNDANTLY AS THE MSB OF BOTH THE MSP AND LSP WORDS

64 LEAD FLAT PACKAGE INFORMATION
(NOT AVAILABLE FOR MPY-8)

FLAT PACK OPERATIONAL TEMPERATURE RANGE: MPY–12A, MPY–16A 0°C TO 70°C AMBIENT WITH θ_{C-A} OF 8°C/W PROVIDED BY THE USER





(DIMENSIONS IN INCHES)

MPY-12A

Pin Out			
1	GND	33	N.C.
2	GND	34	X11
3	GND	35	10
4	SGN MSP	36	09
5	PR01	37	08
6	02	38	07
7	03	39	06
8	04	40	05
9	05	41	04
10	06	42	03
11	07	43	02
12	08	44	01
13	09	45	XSGN
14	10	46	CLKX
15	11	47	CLKY
16	CLK MSP	48	RND
17	CLK LSP	49	Y11
18	TRIM	50	10
19	TRIL	51	09
20	SGN LSP	52	08
21	PR12	53	07
22	13	54	06
23	14	55	•V_{CC}
24	15	56	•V_{CC}
25	16	57	•V_{CC}
26	17	58	Y05
27	18	59	04
28	19	60	03
29	20	61	02
30	21	62	01
31	22	63	YSGN
32	N.C	64	GND

MPY-16A

Pin Out			
1	PRSGN	33	CLKY
2	PR01	34	CLKL
3	02	35	•TRIL
4	03	36	X15
5	04	37	X14
6	05	38	13
7	06	39	12
8	07	40	11
9	08	41	10
10	09	42	09
11	10	43	08
12	11	44	07
13	12	45	06
14	13	46	05
15	14	47	04
16	15	48	03
17	PRSGN, YSGN	49	02
18	PR16, Y01	50	01
19	17, 02	51	XSGN
20	18, 03	52	CLKX
21	19, 04	53	•RND
22	20, 05	54	•V_{CC}
23	21, 06	55	GND
24	22, 07	56	GND
25	23, 08	57	•V_{CC}
26	24, 09	58	•V_{CC}
27	25, 10	59	GND
28	26, 11	60	GND
29	27, 12	61	GND
30	28, 13	62	•V_{CC}
31	29, 14	63	TRIM
32	PR30, Y15	64	CLKM

Notes:

1. Top View Pin Index Number
2. Unit Identification
3. All V_{CC} and GND must be externally connected

NOTES ON TWO'S COMPLEMENT ARITHMETIC

For an N bit binary number, the Two's Complement range extends from -2^{N-1} through $2^{N-1}-1$ for integers and -1 through $1-2^{-(N-1)}$ for fractions. The complete range for the example of N = 4 is given in Figure 1.

4-Bit Two's Complement Number				Base 10 Number	
Sign Bit (MSB)			LSB	Integer	Fraction
0	1	1	1	+7	+7/8
0	1	1	0	+6	+6/8
0	1	0	1	+5	+5/8
0	1	0	0	+4	+4/8
0	0	1	1	+3	+3/8
0	0	1	0	+2	+2/8
0	0	0	1	+1	+1/8
0	0	0	0	0	0
1	1	1	1	-1	-1/8
1	1	1	0	-2	-2/8
1	1	0	1	-3	-3/8
1	1	0	0	-4	-4/8
1	0	1	1	-5	-5/8
1	0	1	0	-6	-6/8
1	0	0	1	-7	-7/8
1	0	0	0	-8	-8/8

Figure 1. 4-Bit Two's Complement Range

Two's complement notation is especially useful to many computer systems. It offers the advantage of having only a single representation for the number zero, as opposed to two for sign-magnitude and one's complement systems. Additionally, it precludes the use of "subtractors" — positive or negative numbers may be added to one another without any regard for the sign of the numbers: the result will always be correct in two's complement notation.

Although positive number representation is the same for both sign-magnitude and two's complement, negative numbers are determined by the following equation:

$$(\text{Two's Complement}) = 2^N - |X|$$

where $|X|$ is the magnitude of the desired negative number and N is the total number of bits used in the two's complement field, including the sign bit, for integers (N = 1 for fractions).

Although the preceding might seem to be a cumbersome function one must perform before every negation, it turns out that it is easily implemented with hardware. The equivalent of the above equation in hardware is simply the inversion of all bits of $|X|$, including the sign bit, plus the addition of binary '1' to the LSB. The same procedure reapplied to the two's complement number yields $|X|$. Inversion is quite straightforward and the addition can generally be performed with the typically unused Carry-In input in the LSB adder.

FRACTIONAL/INTEGER MULTIPLICATION

The MPY-Series multipliers may be used in either a fractional or integer mode — the difference is conceptual. For example, using a 4-bit case, the multiplier does not know (or care) whether it is performing the multiplication 6 x (-2) = -12 or (6/8) x (-2/8) = -12/64: the input and output binary fields will be the same. Fractional multiplication (using fields as defined in the previous specifications) offers the advantage of more convenient single precision usage. The MSB is that which is closest to the binary point in fractional representation (the LSB for integer representation). The fractional notation is additionally the most convenient when implementing a floating point multiplication system.

8-1-213-535-1831

TRW LSI PRODUCTS
P.O. BOX 1125, REDONDO BEACH, CALIFORNIA 90278
(213)535-1831