Digital Signal Processor

MB8764 Applications Manual

FUJITSU

# Digital Signal Processor

# MB8764 Applications Manual

FUJITSU

Fujitsu Limited

Fujitsu Microelectronics, Inc.

Fujitsu Mikroelektronik GmbH

# Digital Signal Processor

# MB8764

## Applications Manual

Circuit diagrams utilizing Fujitsu products are included as a means of illustrating typical applications; consequently complete information sufficient for construction purposes is not necessarily given. The information in this publication has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for any inaccuracies.

The information conveyed in this publication does not convey any license under the patent rights of Fujitsu Limited.

Fujitsu reserves the right to change specifications without notice.

*Digital Signal Processor MB8764 Applications Manual* is published by the U.S. Technical Publications Department, Fujitsu Microelectronics, 3320 Scott Blvd., Santa Clara, CA 95054–3197, (408) 727–1700

Printed in the U.S.A.

Edition 1.0 (Phase 1)

August 1986

**Fujitsu Limited**
Furukawa Sogo Bldg., 6-1
Marunouchi 2-Chome, Chiyoda-ku
Tokyo 100, Japan
Phone National (03) 216-3211
Phone International 81-3-216-3211
Telex: 2224361 FT TOR J

**Fujitsu Microelectronics, Inc.**
3320 Scott Blvd.
Santa Clara, CA  95054-3197
Phone: (408) 727-1700
Telex: (910) 338-0190

**Fujitsu Mikroelektronik GmbH**
Arabella Center 9, OG/A
Lyoner Straße 44-48 D-6000
Frankfurt 71, F.R. Germany
Phone 069/66 320
Telex 0411 963

# Preface

This document describes the basic hardware/software structures and other applications–related features of the MB8764 16–bit general–purpose Digital Signal Processor (DSP). The applications spectrum of the DSP is extremely broad, ranging from relatively simple telecommunications and signal–processing requirements to very complex signal analysis, multi–order filter designs, and systems that demand the utmost in throughput and computation capabilities.

Architectural features and the general applications spectrum of the DSP are defined in Chapter 1. The remaining chapters describe programming techniques, interface characteristics, application examples, and memory–expansions options. An MB8764 Data Sheet is shown in Appendix B.

# Applications Manual MB8764
# Table of Contents

Section One - Introduction Page

1.1 Architectural Overview ........ 1-1
1.2 Applications Spectrum ........ 1-1

Section Two - Programming Techniques

2.1 Multi-Loop Programming ........ 2-1
2.2 Double Precision Addition And Subtraction ........ 2-4

Section Three - DSP Interfacing

3.1 Memory Interfaces ........ 3-1
3.2 Analog Port Interface ........ 3-4
3.3 Interactive Processing ........ 3-7
3.4 Interfacing Multiple DSPs ........ 3-8

Section Four - Design Calculations and Programming Examples

4.1 Introduction ........ 4-1
4.2 Infinite Impulse Responce (IIR) Filter ........ 4-1
4.3 Finite Impulse Response (FIR) Filter (24 Taps) ........ 4-5
4.4 Echo Canceller ........ 4-8
4.5 Program Example For Fast Fourier Transforms (FFT) ........ 4-10

Section Five - ROM Expansion

5.1 Expansion of Instruction ROM ........ 5-1
5.2 ROM-Bank Switching ........ 5-4

Appendix A - FFT Computation ........ A-1

Appendix B - MB8764 Data Sheet ........ B-1

# Applications Manual MB8764
# List of Illustrations and Programming Examples

**Section One - Introduction** **Page**

Figure 1.1 Block Diagram of MB8764 .............................. 1–2

Figure 1.2 Applications Spectrum of MB8764 Digital Signal Processor .... 1–3

**Section Two - Programming Techniques**

Programming Example #1. Using Nested Loops ........................ 2–1

Programming Example #2. Using A Common Register (CO) for Inner–and–Outer Loop Counter .............. 2–2

Programming Example #3. Triple–Loop Program Using Index Addressing ... 2–3

**Section Three - DSP Interfacing**

Figure 3.1 External ROM Interface .................................. 3–2

Figure 3.2 External RAM Interface ................................. 3–3

Figure 3.3 Analog Port Interface .................................. 3–4

Figure 3.4 I/O Control Signals ..................................... 3–6

Figure 3.5 Inactive Processing With Two DSPs ...................... 3–7

Figure 3.6 Multiple DSP Configuration ............................. 3–9

a. Data Transfer Concepts ............................... 3–9

b. System Block Diagram ................................. 3–9

Figure 3.7 Address/DataFormats .................................... 3–10

Figure 3.8 DMA Timing Cycle ........................................ 3–11

**Section Four - Design Calculations and Programming Examples**

Figure 4.1 Infinite Impulse Filter of the 5th Order ....................... 4–2

Figure 4.2 Test Setup for Measuring Response of IIR Filter ............. 4–4

Figure 4.3 Finite Impulse Response (FIR) Filter with Input/Output Waveforms ................................ 4–5

Figure 4.4 Test Setup for Measuring Response of FIR Filter ............. 4–7

Figure 4.4 Test Setup for Measuring Response of FIR Filter . . . . . . . . . . . . . 4–7

Figure 4.5 Echo Canceller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4–9

a. Concepts of Echo Cancellation . . . . . . . . . . . . . . . . . . . . . . . . . . . 4–9

b. Using Multiple DSPs for Additional Memory . . . . . . . . . . . . . . . . . 4–9

Figure 4.6 Input and Output Waveforms of 64–Point FFT . . . . . . . . . . . . . . . 4–12

Figure 4.7 Frequency and Power Ratios of Waveform A'
(All points not shown) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4–13

**Section Five – ROM Expansion**

Figure 5.1 External ROM Expansion and Address Table . . . . . . . . . . . . . . . . 5–2

Figure 5.2 EROM Timing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5–3

Figure 5.3 IK Word IROM And IK Word EROM . . . . . . . . . . . . . . . . . . . . . . . . 5–4

Figure 5.4 1K Word IROM And 3K Word EROM . . . . . . . . . . . . . . . . . . . . . . . 5–5

**Appendix A – FFT Computations**

Figure A.1 Bit–Reversal Process . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . A–1

Figure A.2 Computations for 8–Point FFT . . . . . . . . . . . . . . . . . . . . . . . . . . . . A–2

**Appendix B – MB8764 Data Sheet**

# Section One – Introduction

**Contents of This Section** **Page**

1.1 Architectural Overview .......................................... 1–1
1.2 Applications Spectrum .......................................... 1–1

## 1.1 Architectural Overview

The MB8764 DSP is the ultimate answer to high–speed digital signal processing. Featuring a 100–nanosecond cycle time and a high–speed pipelined multiplier, the MB8764 provides optimum throughput for computation–intensive applications. In addition, the MB8764 provides the user with an extensive instruction set, expandable memory options, an on–chip DMA channel, and a wide assortment of peripherals and system development tools. In combination, these functions and support devices provide design flexibility, system efficiency, and the lowest possible processing costs. A simplified block diagram of the MB8764 DSP is shown in Figure 1.1.

## 1.2 Applications Spectrum

Basic operations, processing functions, and major applications areas are shown in Figure 1.2.

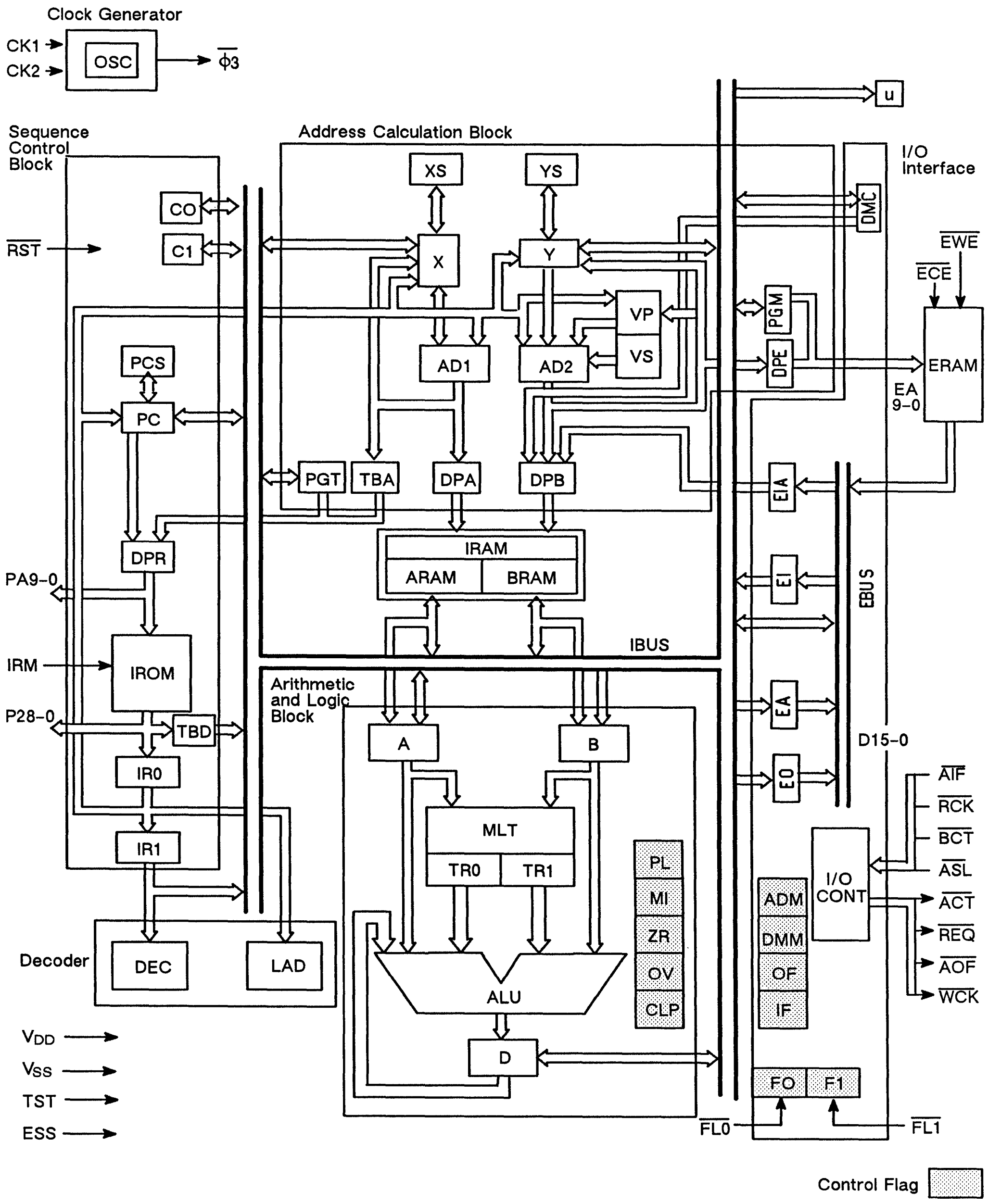
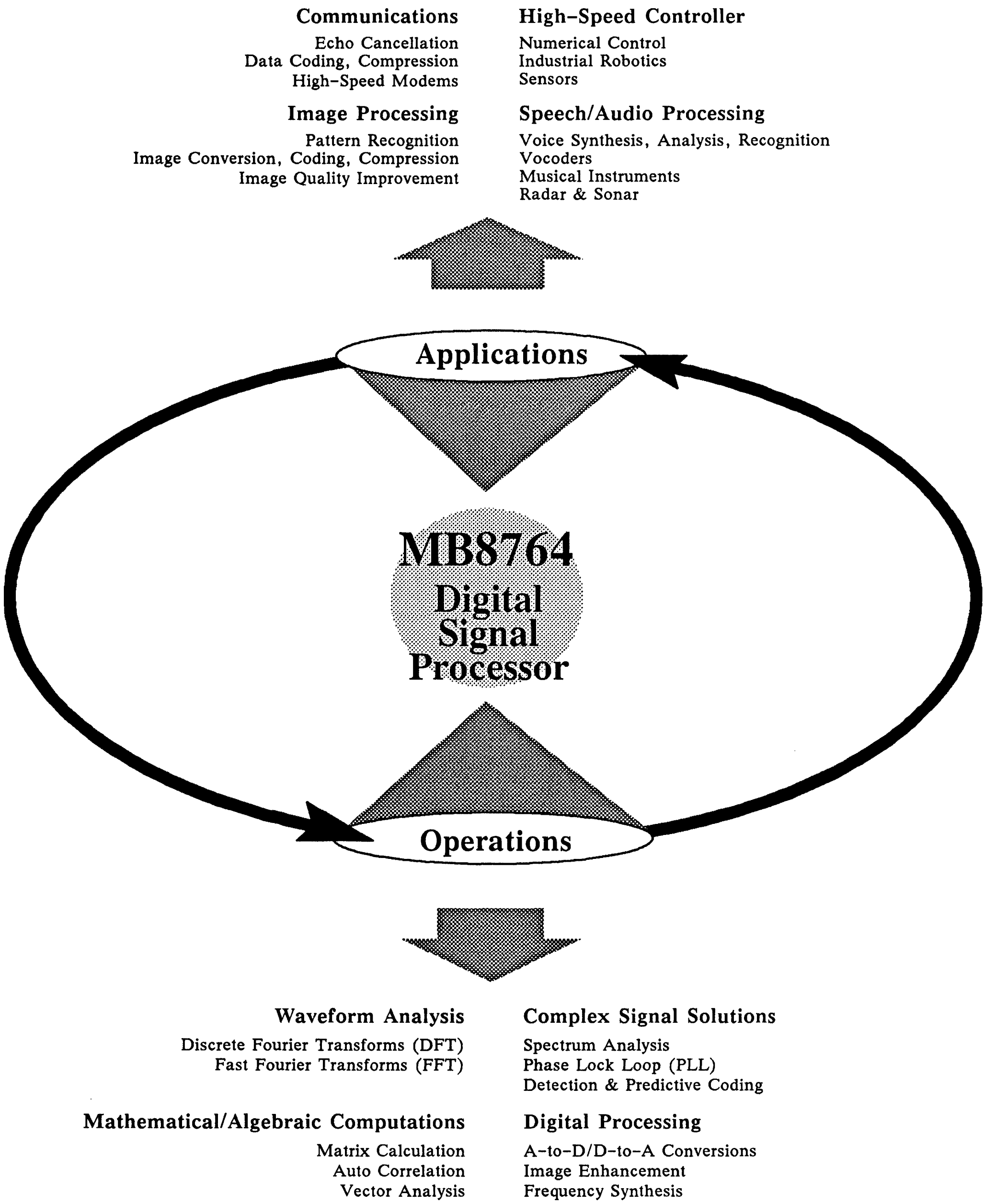


**Figure 1.1 Block Diagram of MB8764**



**Communications**

Echo Cancellation
Data Coding, Compression
High-Speed Modems

**High-Speed Controller**

Numerical Control
Industrial Robotics
Sensors

**Image Processing**

Pattern Recognition
Image Conversion, Coding, Compression
Image Quality Improvement

**Speech/Audio Processing**

Voice Synthesis, Analysis, Recognition
Vocoders
Musical Instruments
Radar & Sonar

**Waveform Analysis**

Discrete Fourier Transforms (DFT)
Fast Fourier Transforms (FFT)

**Complex Signal Solutions**

Spectrum Analysis
Phase Lock Loop (PLL)
Detection & Predictive Coding

**Mathematical/Algebraic Computations**

Matrix Calculation
Auto Correlation
Vector Analysis

**Digital Processing**

A-to-D/D-to-A Conversions
Image Enhancement
Frequency Synthesis

**Figure 1.2 Applications Spectrum of MB8764 Digital Signal Processor**

# Section Two – Programming Techniques

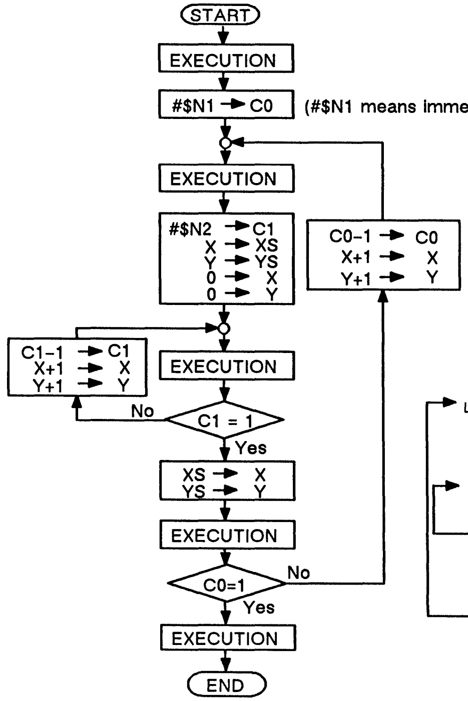
**Contents of This Section** **Page**

2.1 Multi–Loop Programming .......................................... 2–1

2.2 Double Precision Addition And Subtraction ........................ 2–4

## 2.1 Multi–Loop Programming

Programming example #1 shows how nested loops can be utilized. In this dual–loop program, the C0 register is an 8–bit counter and the C1 register is a 4–bit counter.

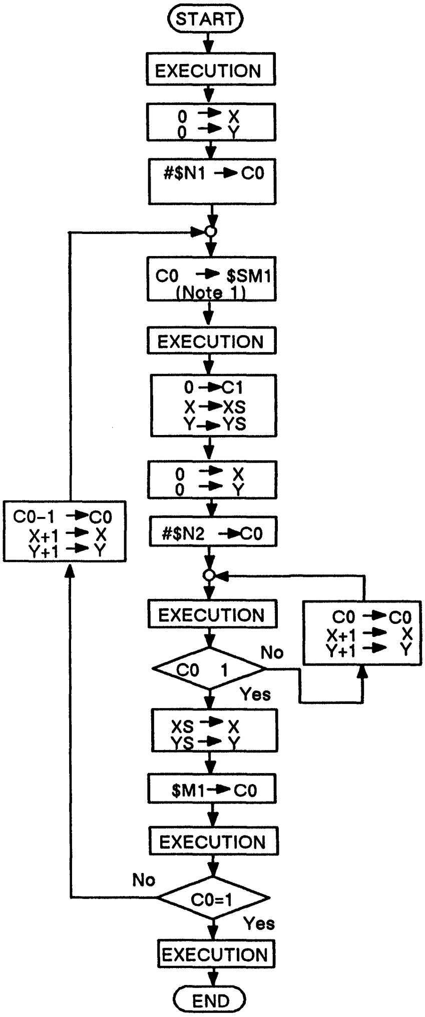


```
            STATEMENT                 PROCESSING

      MOV:NOP #$N1,C0    ....   N1 → C0
→ L1          :
              :
      SXY:NOP #$N2       ....   N2 → C1, X → XS, Y → YS,
→ L2          :                 0 → X, 0 → Y
              :
      JOC:NOP L2,C1      ....   C1–1 → C1, X+1 → X, Y+1 → Y
              :                 When not jumping.....
              :                 XS → X, YS → Y
      JOC:NOP L1, C0     ....   C0–1 → C0, X+1 → X, Y+1 → Y
              :
```

**Programming Example #1. Using Nested Loops**

A dual–loop program that uses the C0 register for both the inner–and–outer loop counter is shown in programming example #2. The number of program statements and the incrementation cycle are the same as those for programming example #1.



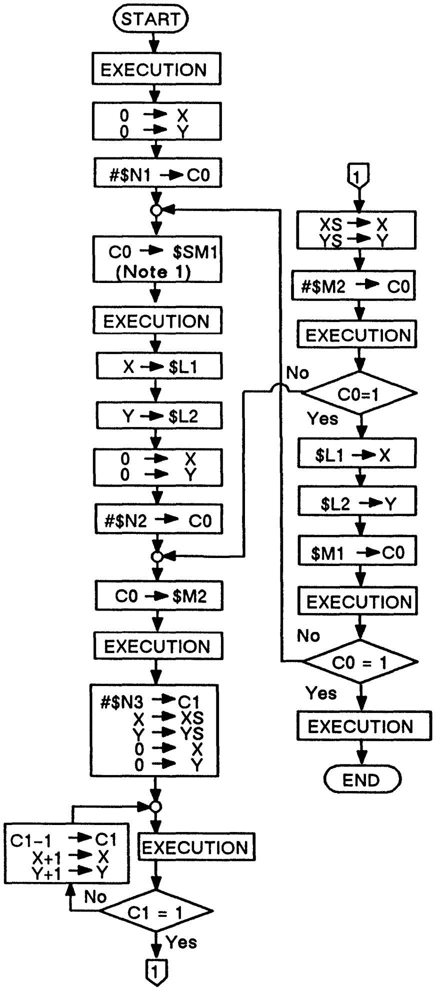
| | STATEMENT | PROCESSING |
|---|---|---|
| | CLR X:Y (Note 3) | 0 → X, 0 → Y |
| | MOV:NOP C0, #M1 | N1 → C0 |
| LOOP1 | MOV:NOP 0, #M1 | C → RAM M1 (Note 2) |
| LOOP2 | ⋮ | |
| | SXY:NOP #$0 (Note 3) | 0 → C1,X → XS,Y → YS<br>0 → X, 0 → Y |
| | MOV:NOP #$N2,C0 | N2 → C0 |
| | ⋮ | |
| | JOC:NOP LOOP2,C0 | C0-1 → C0, X+1 → X, Y=1 → Y |
| | GXY:NOP (Note 3) | XS → X, YS → Y |
| | MOV:NOP $M1,C0 | RAM M1 → C0 |
| | ⋮ | |
| | JOC:NOP LOOP1,C0 | C0-1 → C0, X+1 → X, Y+1 → Y |

**Notes:**

1. $M1 = Address M1 in RAM
2. Address M1 in RAM
3. Required only with Index Addressing

**Programming Example #2. Using A Common Register (CO) for Inner-and-Outer Loop Counter**

Programming example #3 illustrates a triple-loop counter. This example uses index addressing in each loop and, to reduce the number of statements, C1 is used as the jump condition for the innermost loop.



```
            STATEMENT                     PROCESSING

            CLR X:Y  (Note 2)             0 → X, 0 → Y
            MOV:NOP #$N1,C0               N1 → C0
            MOV:NOP C0, $M1               C0 → RAM M1
┌──→ LOOP1         .
│                  .
│           MOV:NOP X,$L1 (Note 2)  X → RAM L1
│           MOV:NOP Y,$L2 (Note 2)  Y → RAM L2
│           CLR X:Y (Note 2)              0 → X, C0 → Y
│           MOV:NOP #$N2,C0               N2 → C0
│           MOV:NOP C0, $M2               C0 → RAM M2
│ ┌──→ LOOP2       .
│ │                .
│ │         SXY:NOP #$N3              N3 → C1, X → XS, Y → YS
│ │ ┌──→ LOOP3     .                  0 → X, 0 → Y
│ │ │              .
│ │ └─────JOC:NOP LOOP3,C1            C0-1 → C1, X+1 → X, Y+1 → 1
│ │              .                    When not jumping.....
│ │              .                     XS → X, YS → Y
│ │         MOV:NOP $M2, C0            RAM M2 → C0
│ │              .
│ │              .
│ └─────────JOC:NOP LOOP2,C0          C0-1 → C0, X+1 → X, Y+1 → Y
│           MOV:NOP $L1,X (Note 2)  RAM L1 → X
│           MOV:NOP $L2,Y (Note 2)  RAM L2 → Y
│           MOV:NOP $M1,C0              RAM M1 → C0
│                  .
│                  .
└───────────JOC:NOP LOOP1,C0          C0-1 → C0, X+1 → X, Y+1 → 1
                   .
```

**Notes:**

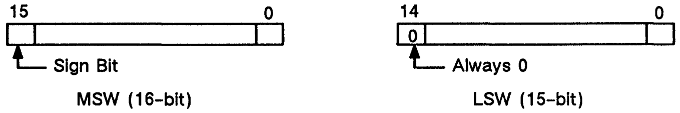
1. $M1 = Address M1 in RAM
2. Required only with Index Addressing

**Programming Example #3. Triple-Loop Program Using Index Addressing**

## 2.2 Double Precision Addition And Subtraction

### 2.2.1 Data Formats

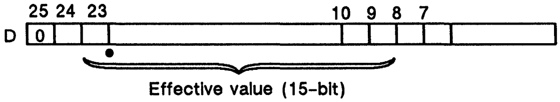
The data word for both the augend and addend is 31–bits long and is in two's complement form. To perform a double–precision add or subtract operation, the data word is separated into two parts--the Least Significant Word (LSW) and the Most Significant Word (MSW). The LSW and MSW format is always the same and is shown in the following diagram



### 2.2.2 Double Precision Addition

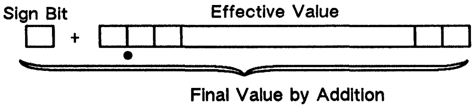
Addition is performed in the following manner:

- Add each LSW value.
- Check if the sum is greater than or equal to two (2).
- Save the result. (The result is stored in RAM location $85).



- Add each MSW value.
- If the sum of the LSWs is greater than or equal to two (2), add one (1) to the sum of the MSWs.
- Save the result plus the sign bit. (The result is stored in RAM location $84)

- The final result of a double precision add is the effective 15-bit value of the LSWs (RAM location $85) and the value of the MSWs plus the sign bit (RAM location $84).



### 2.2.5 Double Precision Subtraction

Subtraction is performed by adding the one's (1) or two's (2) complement of the binary expressions--refer to paragraph 2.2.4. The complementary form of the subtrahend depends on the LSW value. The rule for which complement form to use is as follows:

When all the LSWs (first 15-bits) are zero,

| | |
|---|---|
| LSW 15-bit: | leave all zeroes |
| MSW 16-bit: | transform to two's complement form |

When any of the LSWs (first 15-bits) are not zero,

| | |
|---|---|
| LSW 15-bit: | transform to two's complement form |
| MSW 16-bit: | transform to one's complement form |

In the example that follows, the length of the LSWs and MSWs are reduced to 4-bits.

```
X ... minuend MSW               Y ... subtrahend MSW
X'... minuend LSW               Y'... subtrahend LSW

 A    X ...[0|0 0 0 0]  [1 0 1 0] ... X'
-1
 9    Y ...[0|0 0 0 0]  [0 0 0 1] ... Y'
In this example ... Y' ≠ all 0's
      Therefore ... Y' becomes Ȳ + 1 =   [1 1 1 1]
            and ... Y becomes Ȳ      = [1|1 1 1 1]
Adding, we get:   [0|0 0 0 0]  [1 0 1 0]
                + [1|1 1 1 1]  [1 1 1 1]
              ---------------------------
Result:      [1]  [0|0 0 0 0]  [1 0 0 1]
              ↑___ discard carry
```

---- ADDRESS MAP OF EACH VALUE ----

| | | | |
|---|---|---|---|
| $80 | MSW 1 | $84 | MSW result |
| $81 | LSW 1 | $85 | LSW result |
| $82 | MSW 2 | | |
| $83 | LSW 2 | | |

```
NO. LOC    OP   OPRND1/2   SOURCE STATEMENT
1                                   PRG    DBLPRE
2                          *
3   0000                            ORG    MAIN,$000
4                          *
5                          ***********************************
6                          *    DOUBLE PRECISION SUBTRACT    *
7                          ***********************************
8                          *
9   0000   401283          DBLSUB   MOV:NOP  $83,D      ;GET LSB
10  0001   D08808                   JOC:NOP  ZERO,ZR    ;CHECK FOR ALL ZERO
11  0002   530000                   NOP:NEG             ;IF NOT ZERO TRANSFORM  TO TWO'S COMPLEMENT
12  0003   800A83                   MOV:NOP  D, $83     ;REPLACE OLD LSB
13  0004   401282                   MOV:NOP  $83,D      ;GET MSB
14  0005   5F0000                   NOP:COM             ;TRANSFORM TO ONE'S COMPLEMENT
15  0006   800A82                   MOV:NOP  D, $82     ;REPLACE OLD MSB
16  0007   F0E00C                   JMP:NOP  DBLADD     ;DO ADD
17  0008   401282          ZERO     MOV:NOP  $82,D      ;ELSE
18  0009   530000                   NOP:NEG             ;TRANSFORM MSB TO TWO'S COMPLEMENT
19  000A   800A82                   MOV:NOP  D, $82     ;REPLACE OLD MSB
20  000B   F0E00C                   JMP:NOP  DBLADD     ;DO ADD
21                         *
22                         ***********************************
23                         *    DOUBLE PRECISION ADDITION    *
24                         ***********************************
25                         *
26  000C   404281          DBLADD   MOV:NOP  $81, A     ;GET LSB-1
27  000D   402283                   MOV:NOP  $83,B      ;GET LSB-2
28  000E   520000                   NOP:ADD             ;ADD LSB'S
29  000F   800A85                   MOV:NOP  D, $85     ;SAVE LSB RESULT
30  0010   474280                   MOV:SLA  $80, A     ;GET MSB-1 & SHIFT LSB RESULT TO CHECK FOR CARRY
31  0011   402282                   MOV:NOP  $82, B     ;GET MSB-2
32  0012   520000                   NOP:ADD             ;ADD MSB'S
33  0013   C00001                   LDI:NOP  #$1        ;LOAD A-REGISTER WITH $1
34  0014   DC8416                   JOC:SUM  DONE, OV   ;CHECK FOR LSB OVERFLOW & ADD ONE TO MSB'S
35  0015   5E0000                   NOP:RED             ;IF NO OVERFLOW REDUCE MSB'S BY ONE
36  0016   800A84          DONE     MOV:NOP  D, $84     ;SAVE MSB RESULT
37                         *
38                                  END
```

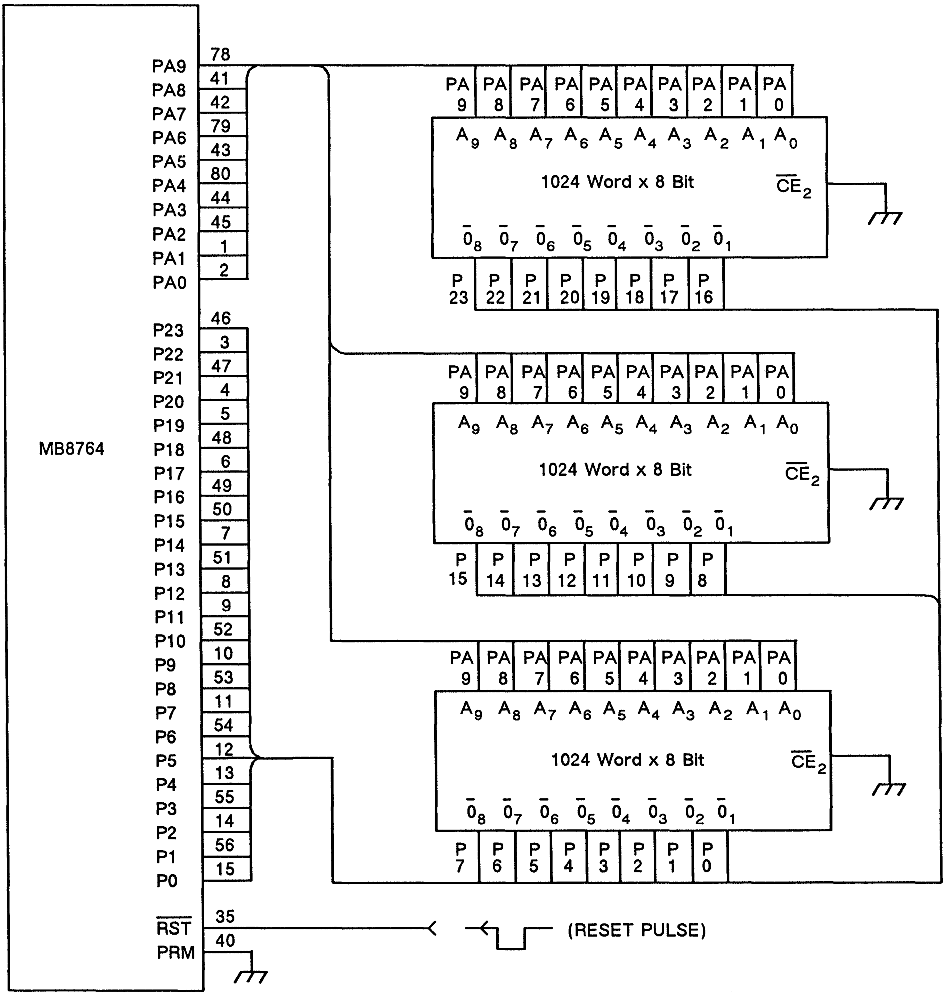
# Section Three – DSP Interfacing

**Contents of This Section** **Page**

3.1 Memory Interfaces . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3–1
3.2 Analog Port Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3–4
3.3 Interactive Processing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3–7
3.4 Interfacing Multiple DSPs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3–8

## 3.1 Memory Interfaces

Figures 3.1 and 3.2, respectively, show how the MB8764 can be interfaced to external ROM and RAM memories. Observe that external logic or timing arbitration circuits are not required.



ROM address (PA9–PAO) is incremented by reset pulse

**Figure 3.1 External ROM Interface**

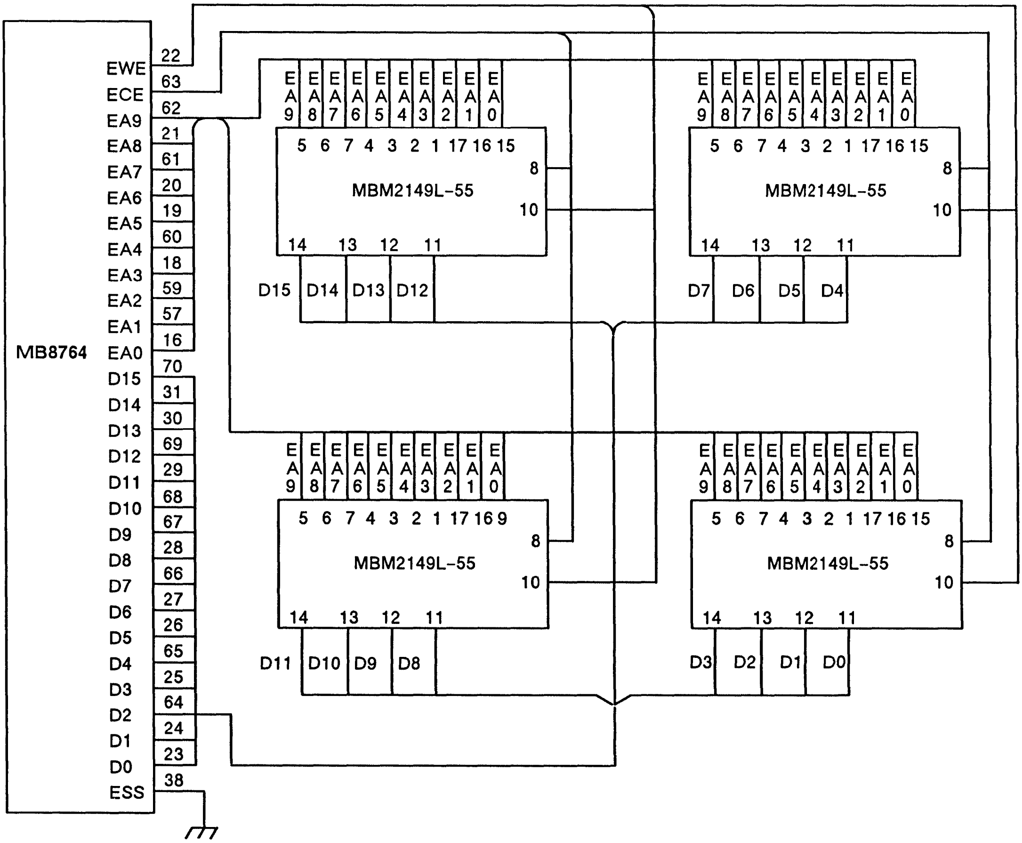


**Figure 3.2 External RAM Interface**

## 3.2 Analog Port Interface

One method of implementing a 16-bit digital-to-analog interface is shown in Figure 3.3. In this type of interface, signal compatibility and I/O timing parameters are critical. The circuits shown address both of these areas and provide the best possible solutions. Timing for the I/O control signals is shown in Figure 3.4.

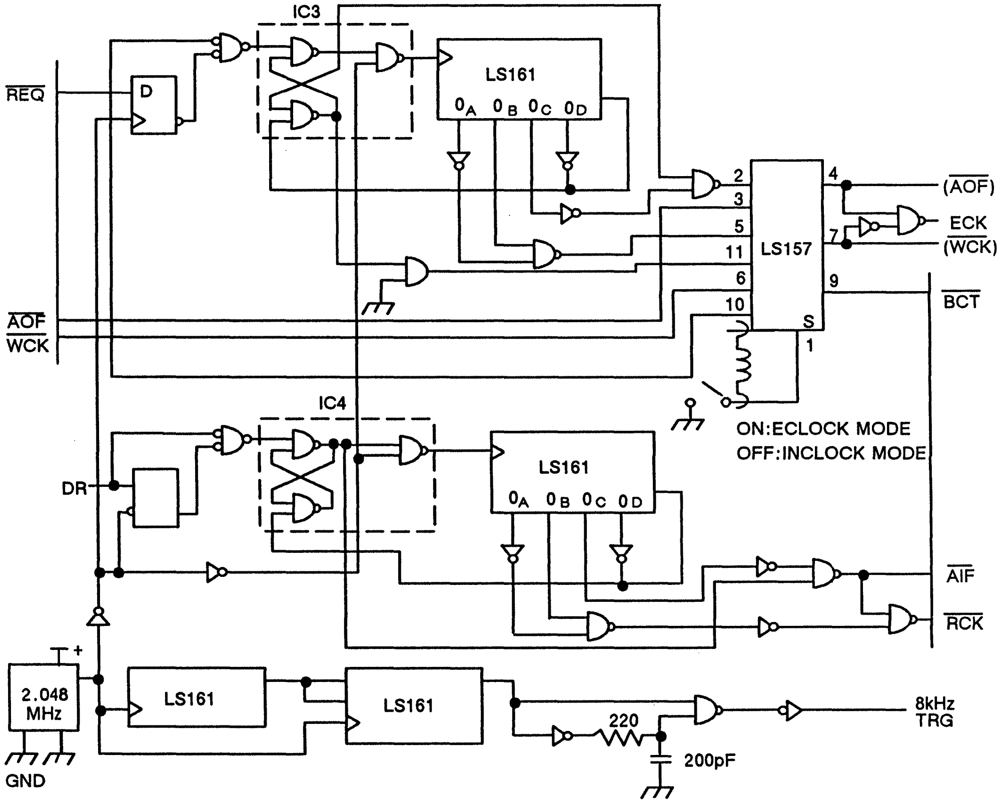


**Figure 3.3 Analog Port Interface**

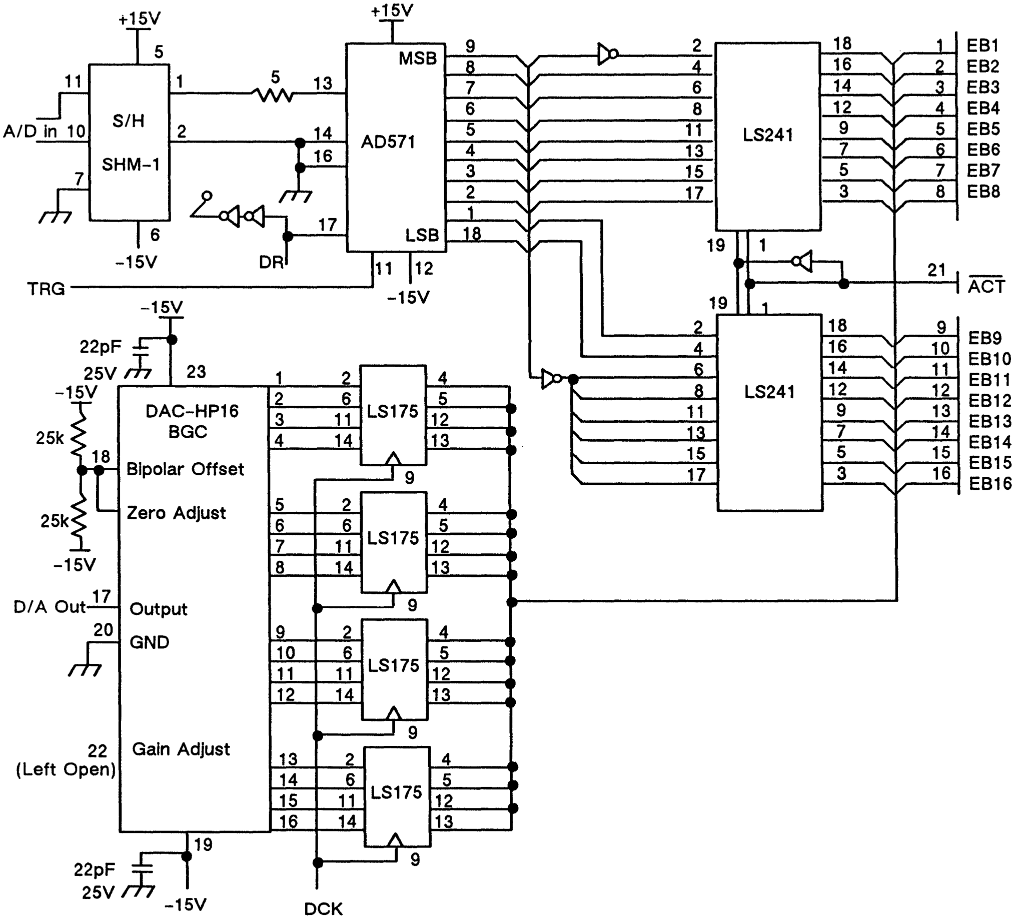
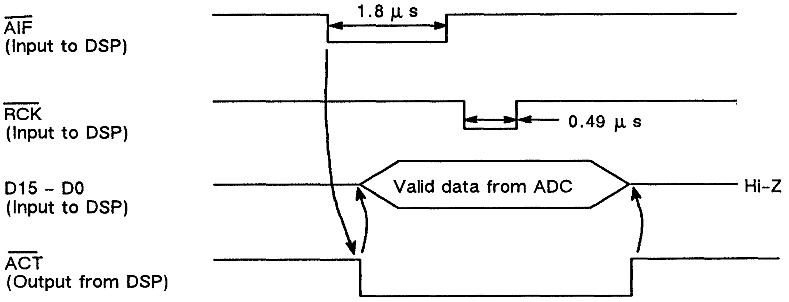


**Figure 3.3 Analog Port Interface (Cont'd)**

## Analog/Digital Converter-to-Digital Signal Processor:



Notes:

1. The above timing sequence is initiated by an 8kHz clock signal derived from the on-board 2.048 MHz master clock.
2. The DSP receives data in the Program (P) mode.

## Digital Signal Processor-to-Digital/Analog Converter:

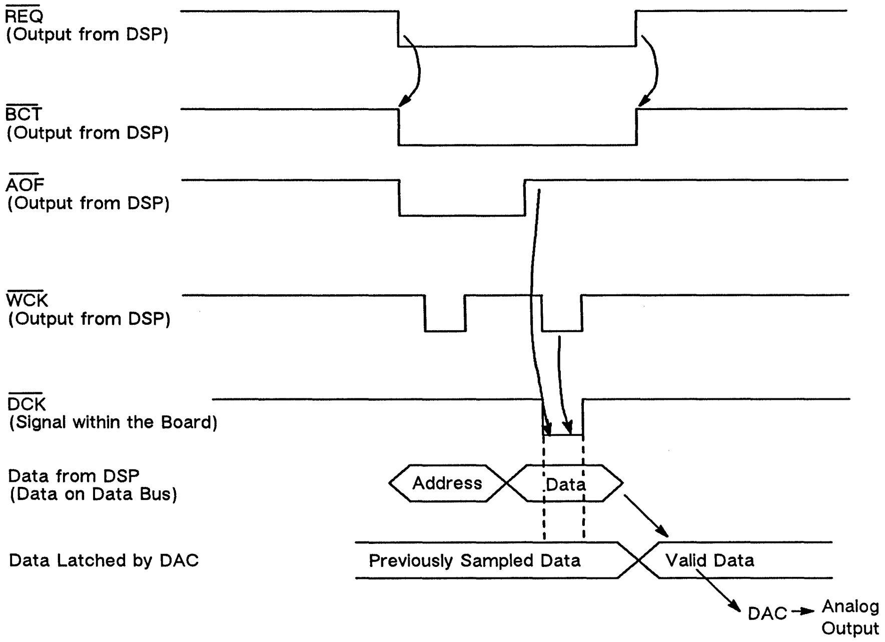


**Figure 3.4 I/O Control Signals**

## 3.3 Interactive Processing

A simple interactive signal processing network is shown in Figure 3.5. The EN and DIR logic provides a bidirectional data link between the two DSPs; however, each DSP is restricted to its associated ERAM (DSP/A ↔ ERAM/A and DSP/B ↔ ERAM/B). A processing system of this type is easy to implement, requires very few parts, and essentially doubles the throughput.

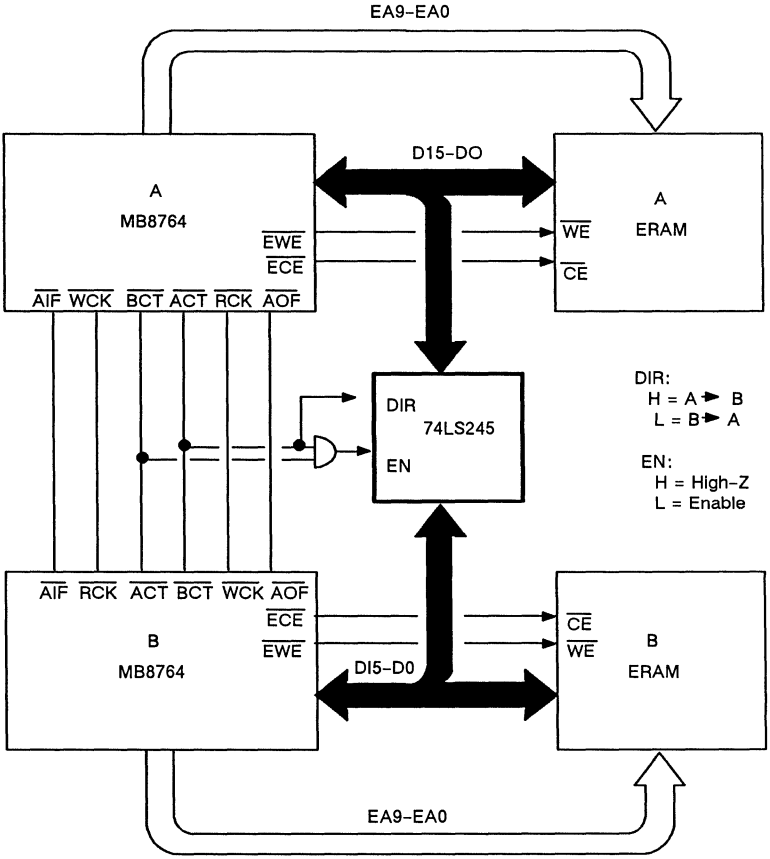


**Figure 3.5 Interactive Processing With Two DSPs**

## 3.4 Interfacing Multiple DSPs

Multiple DSP configurations are particularly useful in applications that require many phases of signal processing or in those requiring hybrid control features. In the example that follows, three DSPs are used to evaluate a transversal filter. Data transfer concepts and a simplified block diagram of the system are shown in Figure 3.6.

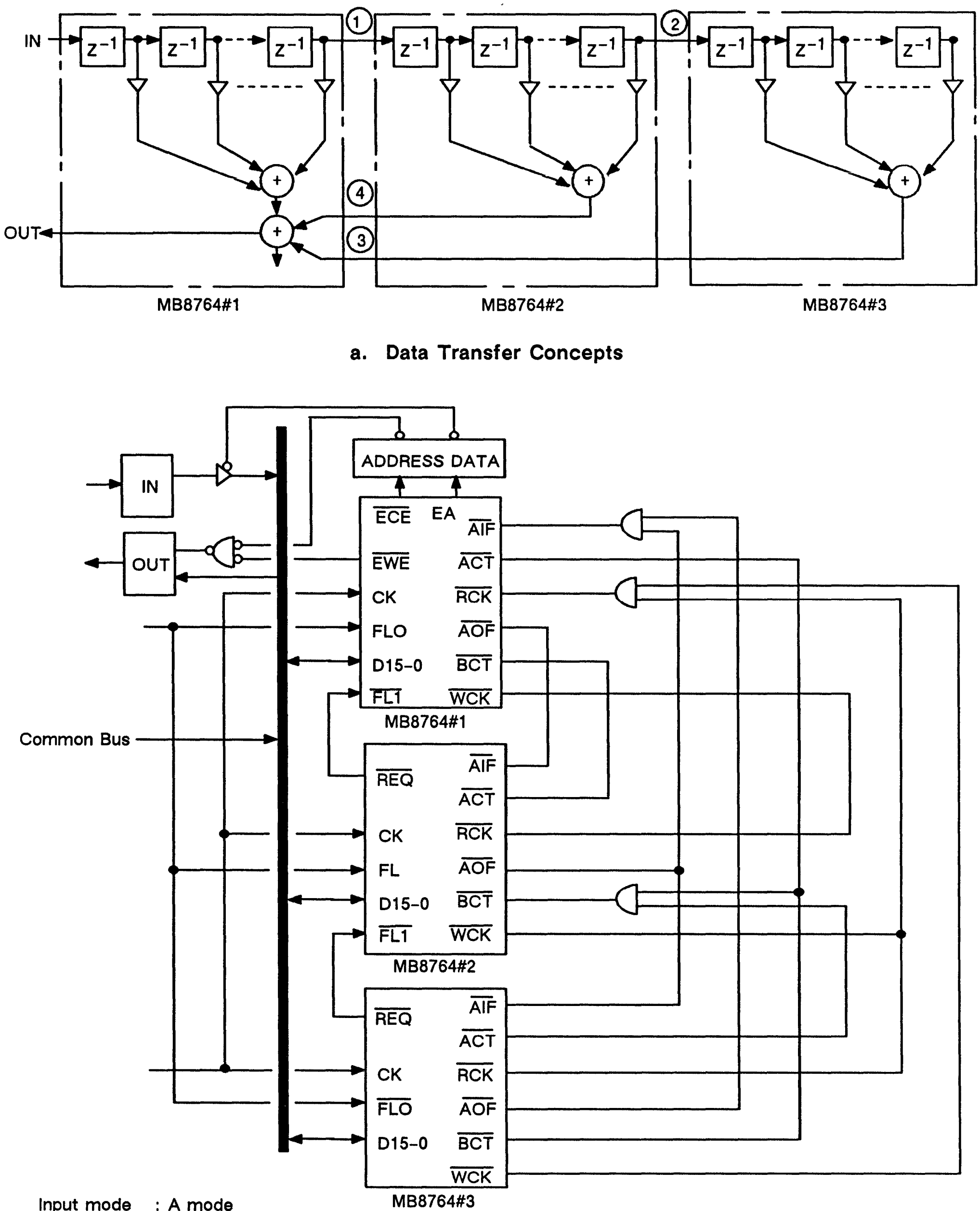
As indicated in Figrue 3.6a, data is transferred between the three DSPs in the following manner:

IN : Input new sample data.
OUT : Output result after execution.
1 : Data transfer from #1 to #2.
2 : Data transfer from #2 to #3.
3 : Transfer filtered result from #3 to #1.
4 : Transfer filtered result from #2 to #1.

In terms of processing dynamics (Figure 3.6b), the following operations occur:

IN, OUT : Accessing ERAM #1
1 : Data transferring from #1 to #2 with address in DMA mode.
2 : Data transferring from #2 to #3 with address in DMA mode.
3 : Data transferring from #3 to #1 with address in DMA mode.
4 : Data transferring from #2 to #1 with address in DMA mode.
In operations 1 and 2 data calculated for the next cycle is transferred.

Address/data formats for each transfer cycle and a DMA timing cycle are shown in Figures 3.7 and 3.8, respectively.



a. Data Transfer Concepts

Input mode : A mode
Output Mode : I mode

b. System Block Diagram

**Figure 3.6 Multiple DSP Configuration**

INPUT/OUTPUT DATA

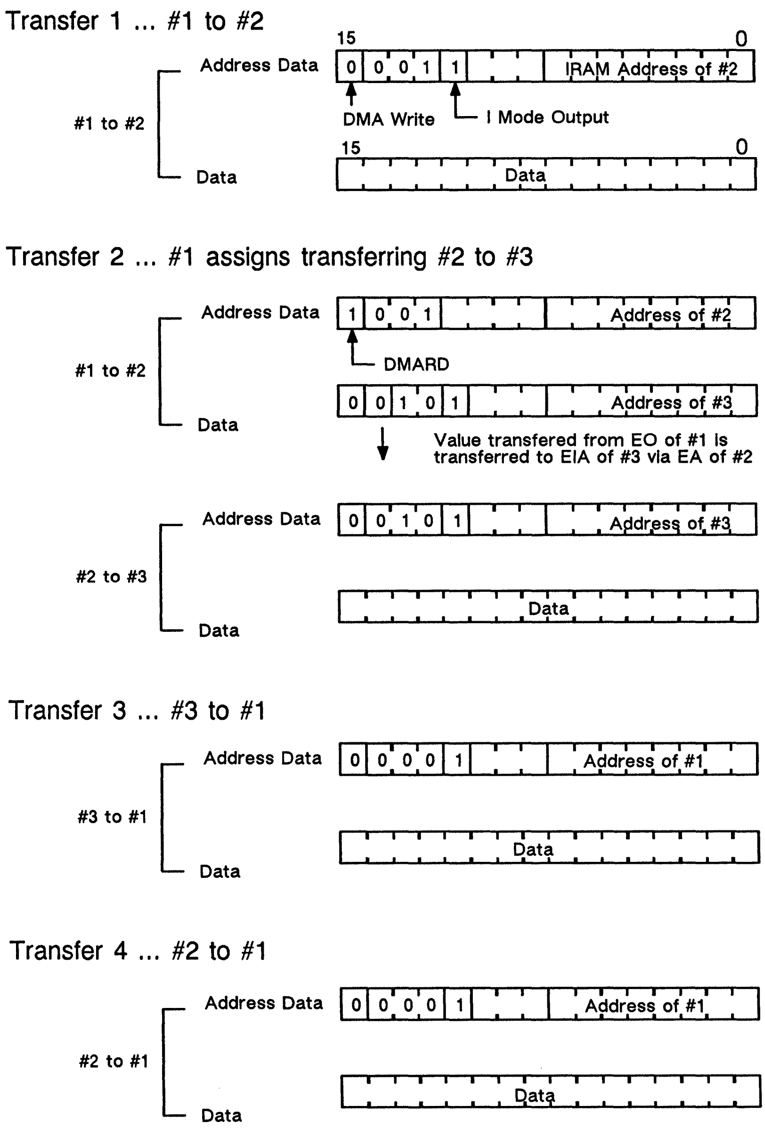


**Figure 3.7 Address/Data Formats**

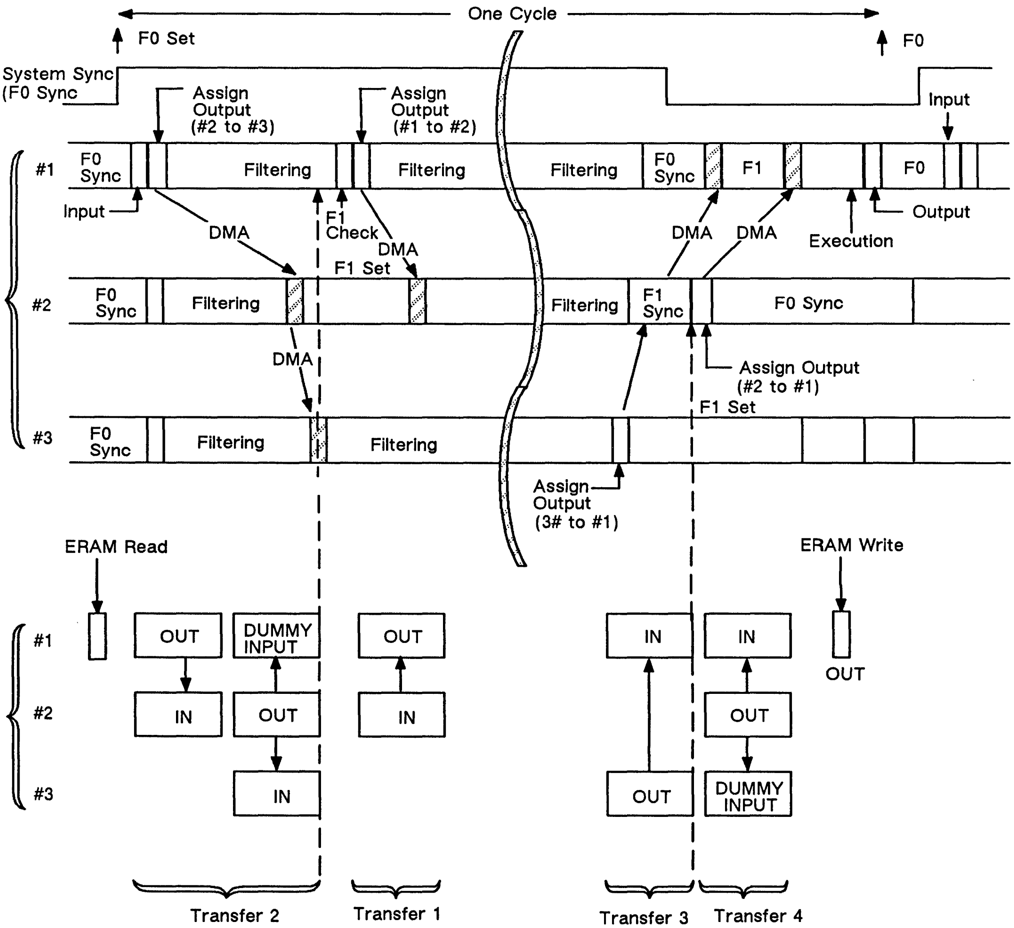


**Figure 3.8 DMA Timing Cycle**

# Section Four – Design Calculations and Programming Examples

**Contents of This Section** **Page**

4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4–1
4.2 Infinite Impulse Response (IIR) Filter . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4–1
4.3 Finite Impulse Response (FIR) Filter (24 Taps) . . . . . . . . . . . . . . . . . . . . 4–5
4.4 Echo Canceller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4–8
4.5 Program Example For Fast Fourier Transforms (FFT) . . . . . . . . . . . . . . . 4–10

## 4.1 Introduction

This chapter shows how the DSP can be used to solve some design problems that would otherwise require extensive logic and considerable development costs in both hardware and software. Using the DSP and its support peripherals, each of the applications that follow can be easily developed and performance–evaluated with a minimum of time and effort.

## 4.2 Infinite Impulse Response (IIR) Filter

The DSP is an ideal choice for calculating component values and frequency responses of an infinite impulse response filter similar to the configuration shown in Figure 4.1. The equation corresponding to each second–order filter section can be expressed as:

$$Yn = \frac{ao + a1Z^{-1} + a2Z^{-2}}{1 + boZ^{-1} + b1Z^{-2}} Xn$$ and, after transformation:

$$Yn = aoXn + n-1(a1X) + n-2(a2X) - n-1(boY) - n-2(b1Y)$$

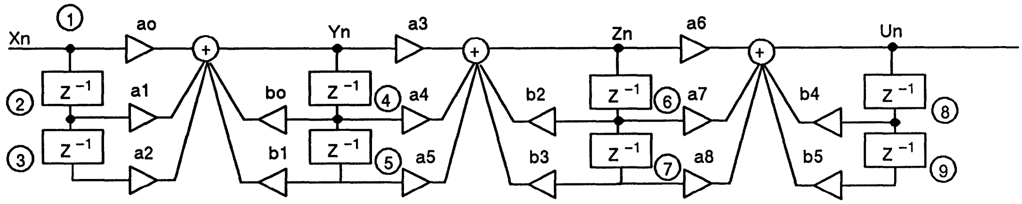


**Figure 4.1 Infinite Impulse Response Filter of the 5th Order**

In Figure 4.1, the memory area corresponding to each signal node (encircled numbers 1 through 9) are located in the numeric area of BRAM; all coefficients are stored in queue in ARAM.

The virtual shift mode is used to sequence data shifting for each signal node. In the programming example and execution cycles that follow, it is assumed that calculations are to proceed indefinitely; thus, initial setting are omitted. As shown in the execution cycles, after the first second-order stage has been calculated, the output value is stored in address 4 and the next stage is calculated. As each stage is calculated, the output of that stage becomes the input for the next. When the last stage is caluclated, the value of each signal node appears in the "after execution" column and the second processing cycle begins.

In the second iteration, the output data of each signal node is indicated with a prime mark (x'). Each filter coordinate (X and Y) is incrementally increased in value and addresses are advanced accordingly. The processing operation can be expanded to achieve the required response characteristics of the filter. By using the virtual shift mode in this program, data shift processing can be omitted.

## IIR Program Example (Second-Order Filter Section)

```
01      SET         VS                       ...  Set virtual shift mode.
02      MOV:NOP     #$200,PGT:PGM            ...  Set PGT and PGM.
    L1  EQU *                                ...  Define as L1 = 03.
03      JOC:NOP     L1,IF                    ...  Wait for new data.
04      MOV:NOP     D,EO                     ...  Transfer previous data to EO.
05      MOV:NOP     #$3,CO                   ...  Set number of loop.
06      MOV         #$800,EA                 ...  Start output (in I mode).
07      MOV:NOP     #$O,X:Y                  ...  Reset X and Y. VP is stored in Y.
08      MOV:NOP     EI, $80(Y)               ...  Set input.
    L2  EQU *                    }
09      LAB:NOP     $0(X),$0(Y)  }
0A      LAB:NOP     $1(X),$1(Y)  }  Filter
0B      LAB:MLT     $2(X),$2(Y)  }  Processing
0C      LAB:MSM     $3(X),$3(Y)  }
0D      LAB:MSM     $4(X),$4(Y)  }
0E      NOP:MSM                  }
0F      MXY :MSM    #$4,#$1                  ...  Change X value.
10      MOV:NOP     D,$81(Y)                 ...  Result is stored in D.
11      JOC:NOP     L2,C0                    ...  Loop control statement.
12      AVP:NOP     #$F                      ...  Subtact VP by 1 for next loop.
13      JMP :NOP    L1
```

## Execution Cycles

| BRAM Address | 1st Cycle | | | | 2nd Cycle |
|---|---|---|---|---|---|
| | Before Execu-tion | After 1st Stage | After 2nd Stage | After Execution | Before Execu-tion |
| 0 | X | X | X | X | X |
| 1 | X | X | X | X | ①' |
| 2 | ① | ① | ① | ②' (=①) | ②' |
| 3 | ② | ② | ② | ③' (=②) | ③' |
| 4 | ③ | ④' | ④' | ④' | ④' |
| 5 | ④ | ④ | ④ | ⑤' (=④) | ⑤' |
| 6 | ⑤ | ⑤ | ⑥' | ⑥' | ⑥' |
| 7 | ⑥ | ⑥ | ⑥ | ⑦' (=⑥) | ⑦' |
| 8 | ⑦ | ⑦ | ⑦ | ⑧' | ⑧' |
| 9 | ⑧ | ⑧ | ⑧ | ⑨' (=⑧) | ⑨' |
| A | ⑨ | ⑨ | ⑨ | X | X |
| B | X | X | X | X | X |
| C | X | X | X | X | X |
| D | X | X | X | X | X |
| E | X | X | X | X | X |
| F | X | X | X | X | X |

## Processing Coefficients

| Coefficient | Equivalent in Figure 4.1 |
|---|---|
| $2D4A ...... | ao |
| $5A93 ...... | a1 |
| $2D4A ...... | a2 |
| $A5EF ...... | bo |
| $DAD3 ...... | b1 |
| $2614 ...... | a3 |
| $4A4A ...... | a4 |
| $2614 ...... | a5 |
| $901F ...... | b2 |
| $C726 ...... | b3 |
| $5000 ...... | a6 |
| $5000 ...... | a7 |
| $0000 ...... | a8 |
| $DE0D ...... | b4 |
| $0000 ...... | b5 |

A test setup designed to measure the response curve of the IIR filter is shown in Figure 4.2. The spectrum analyzer provides a high-resolution display of both gain and frequency.

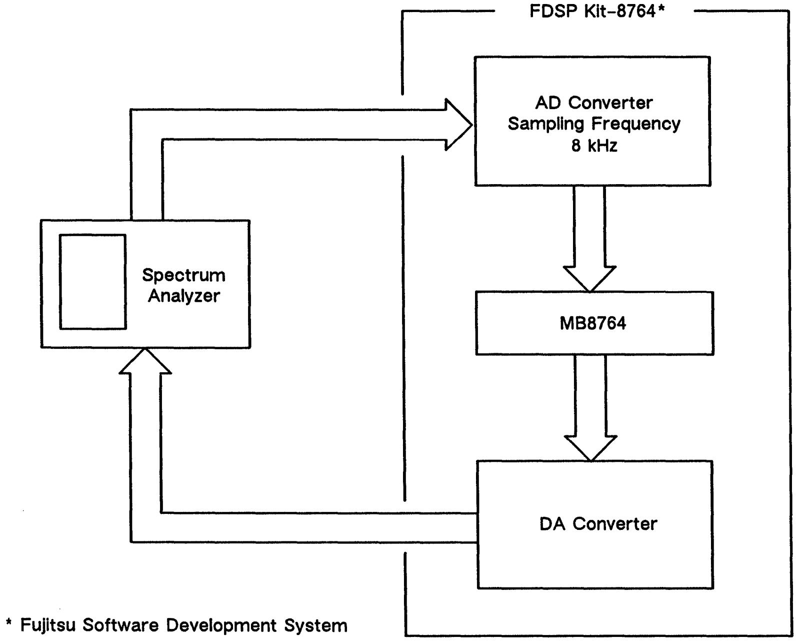
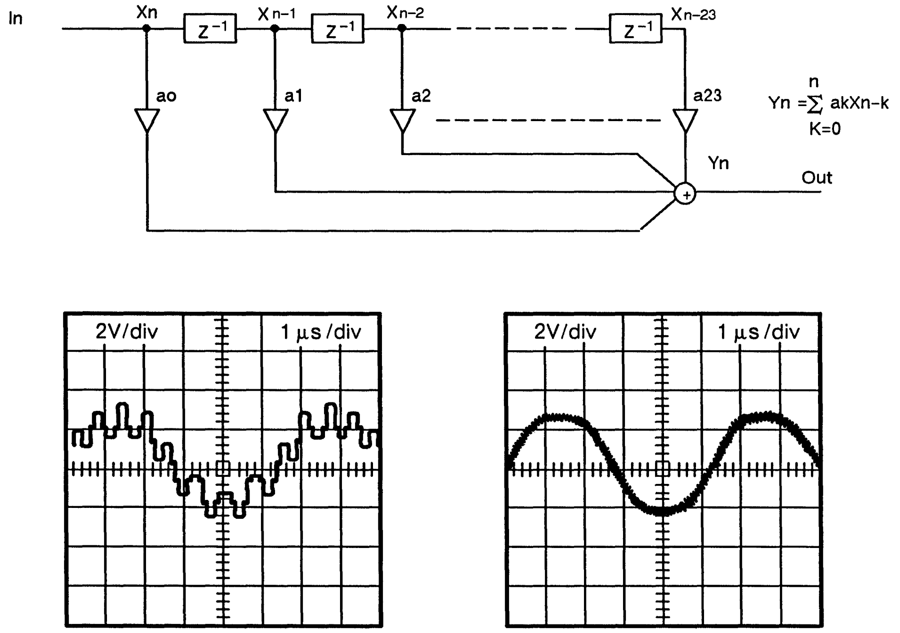


**Figure 4.2 Test Setup for Measuring Response of IIR Filter**

## 4.3 Finite Impulse Response (FIR) Filter (24 Taps)

A FIR filter with typical input and output waveforms is shown in Figure 4.3. As shown in waveform A, all high-frequency components are present at the input of the filter. Each section of the filter is designed to attenuate specific high-order frequencies; thus, the signal at the output summing point (Yn) is nearly a pure sinewave--see waveform B. Filters of this type are frequently used for echo cancellation in communication systems - refer to paragraph 4.4.



**Waveform A: Input**　**Waveform B: Output**

**Figure 4.3 Finite Impulse Response (FIR) Filter with Input/Output Waveforms**

Because of a JMP instruction to address 17, the FIR program example that follows directly outputs the input data. To execute the filter program, the JMP instruction must be changed to a NOP instruction.

## FIR Program Example

```
                  PRG                 TRANS
       **************************************************************
       *                                                            *
       *       24  TAPS TRANSVERSAL FILTER FOR MB8764               *
       *                                                            *
       *       (C) 10, OCTOBER, 1983 BY FUJITSU LIMITED             *
       *                                                            *
       **************************************************************

        ORG              TRF,$000
TRF     EQU*
        CLR              ADM:DMM:Y
IN      JOC              IN,IF
        MOV              #$800,EA
        MOV              EI,$80(Y)
        MOV              EI,$0(Y):A
        MOV              A,EO
        JMP              IN
        LIY              #$37
        LIY              #$F4
        LIY :MLT         #$AD
        LIY :MSM         #$29
        LIY :MSM         #$FEFC
        LIY :MSM         #$FDD2
        LIY :MSM         #$FD90
        LIY :MSM         #$FF11
        LIY :MSM         #$290
        LIY :MSM         #$762
        LIY :MSM         #$C10
        LIY :MSM         #$EF2
        LIY :MSM         #$EF2
        LIY :MSM         #$C10
        LIY :MSM         #$762
        LIY :MSM         #$290
        LIY :MSM         #$FF11
        LIY :MSM         #$FD90
        LIY :MSM         #$FDD2
        LIY :MSM         #$FEFC
        LIY :MSM         #$29
        LIY :MSM         #$AD
        LIY :MSM         #$F4
        LIY :MSM         #$37
        NOP:MSM
        MXY:MSM          #0,#$E7
        MOV              D,EO
        JMP              IN               * RESTART TRANSVERSAL FILTER
        END
```

A test setup designed to measure the response curve of the FIR filter is shown in Figure 4.4

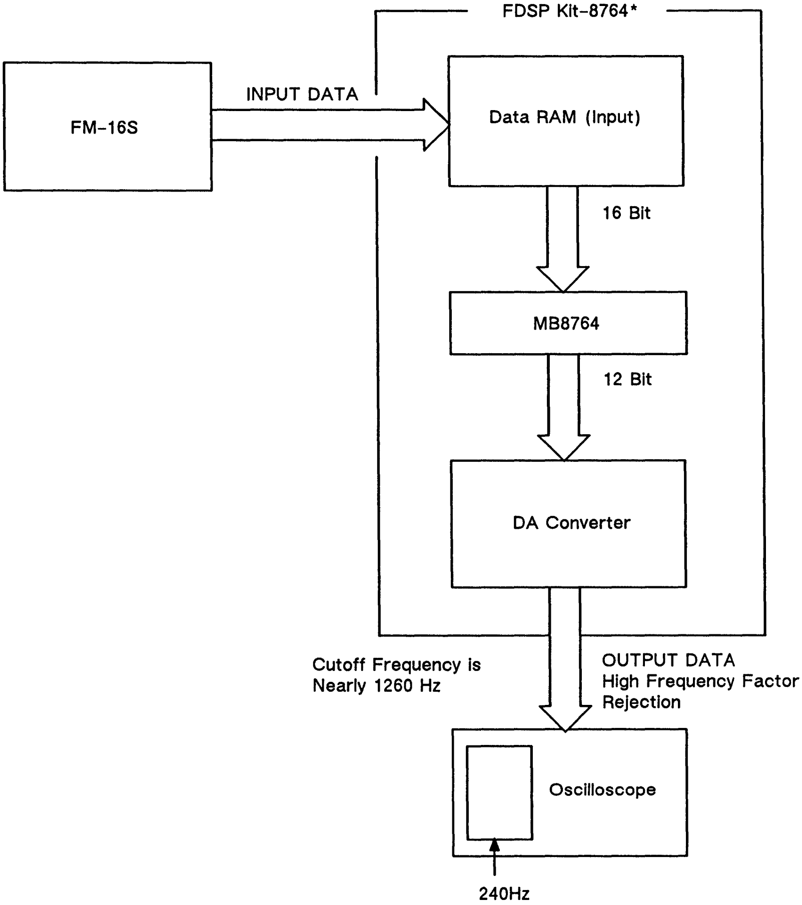


**Figure 4.4 Test Setup for Measuring Response of FIR Filter**

## 4.4 Echo Canceller

In communication systems and other configurations where signal echos are a problem, the MB8764 provides a classic solution. As shown in 4.5a, the actual echo {y(k)} is a spectrum of impulse frequencies caused by line mismatches and is expressed by the equation:

$$y(k) = \sum_{n=o}^{\infty} hnx(k-1) + v(k)$$

The (hk) term in Figure 4.5a represents the impulse response of an echo pass, that is, the A–to–B pass through the hybrid. To cancel the actual echo {y(k)}, an imaginary echo $\{\hat{y}(k)\}$, of opposite phase is generated. These two signals {y(k) and $\hat{y}(k)d\}$ are summed at point B and the result is echo cancellation.

To generate the imaginary echo and the impulse series, the MB8764 computes the following two formulas:

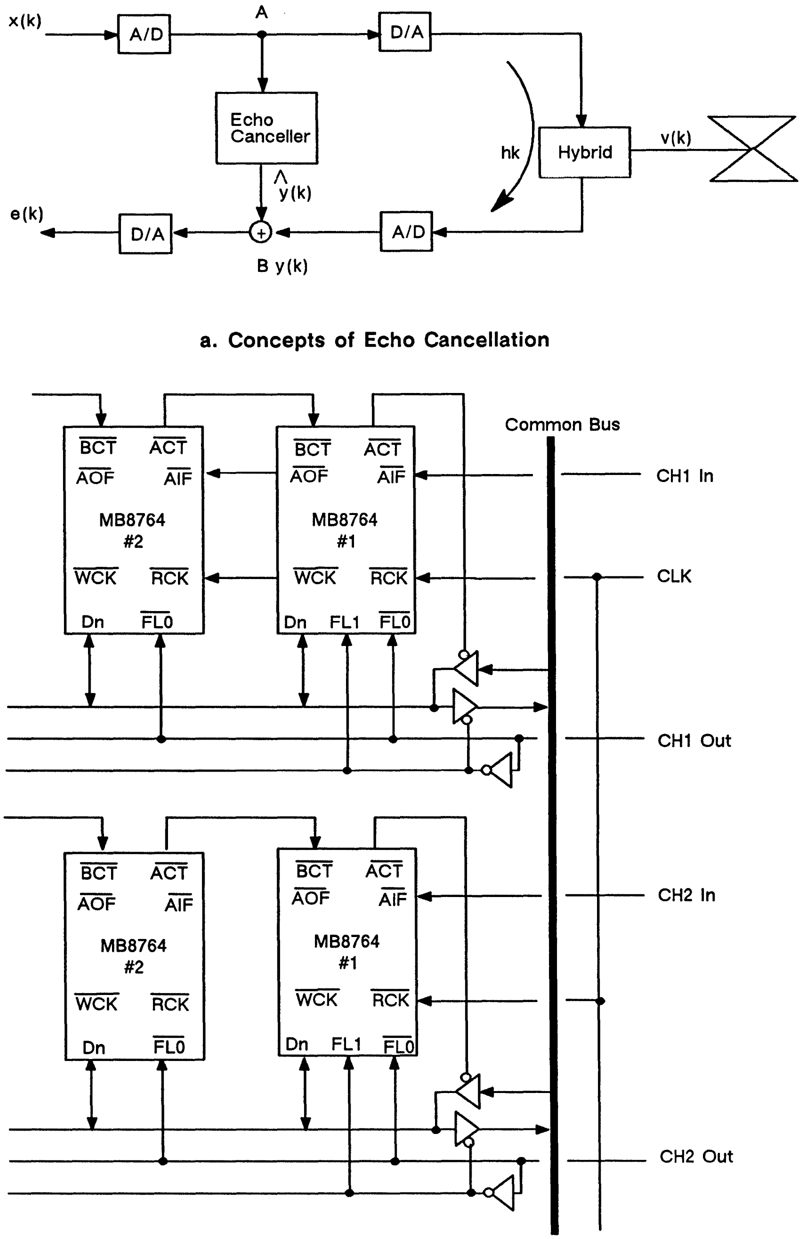
$$\#1 = \hat{y}(k) = \sum_{n=o}^{\infty} hn(k)+x(k-n)$$

$$\#2 = \hat{h}\,(k+1) = \hat{h}\,(k) + Kx(k-n)e(k)$$

Reducing and solving for the instantaneous voltage at the summing point:

$$e(k) = Y(k) - \hat{y}(k)$$

Formula #1 is identical to the FIR transversal filter characteristic described in paragraph 4.3. For the typical echo canceller, several taps of the transversal filter are required to accommodate system delay times. Multiple DSPs are used to satisfy the additional memory requirements; a typical configuration is shown in Figure 4.5b.



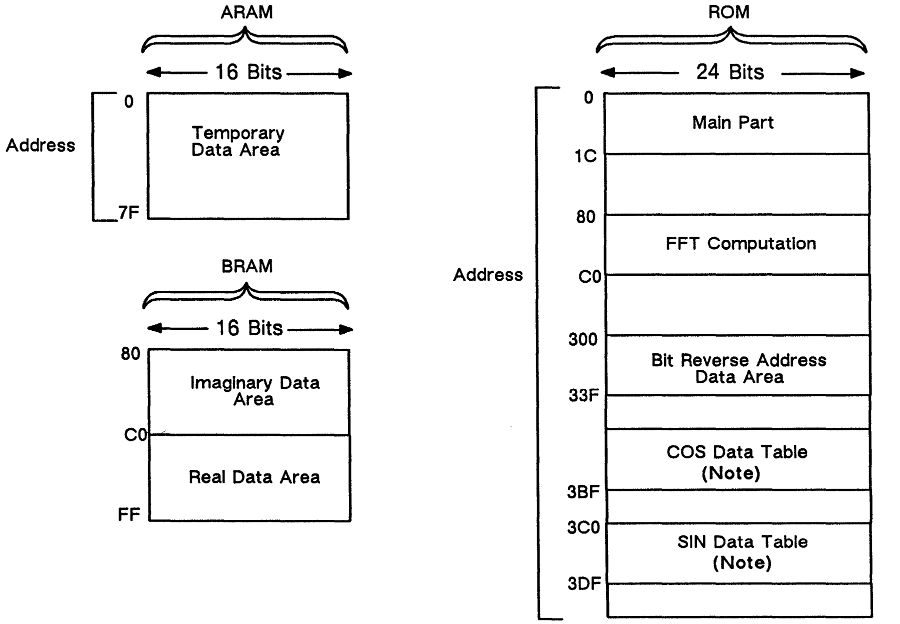
**a. Concepts of Echo Cancellation**

**b. Using Multiple DSPs for Additional Memory**

**Figure 4.5 Echo Canceller**

## 4.5 Program Example For Fast Fourier Transforms (FFT)

The DSP can be used to reduce any complex waveform into discrete component values. In the example that follows, the Fourier–transform program uses a single DSP and a 64–point matrix to analyze the input waveform. Memory location and mapping tables for the Assembly Listing that follows are shown below. A summary of FFT computations is shown in Appendix A.



Note: When ROM is used for sine/cosine tables, lower 16–bits are used for data.

The main part of the ROM table is divided into three basic areas, which are:

Data Input (0001–000A)-- the input data is changed into bit reversed data and stored in the real data area of BRAM; zeroes are stored in the imaginary data area of BRAM.

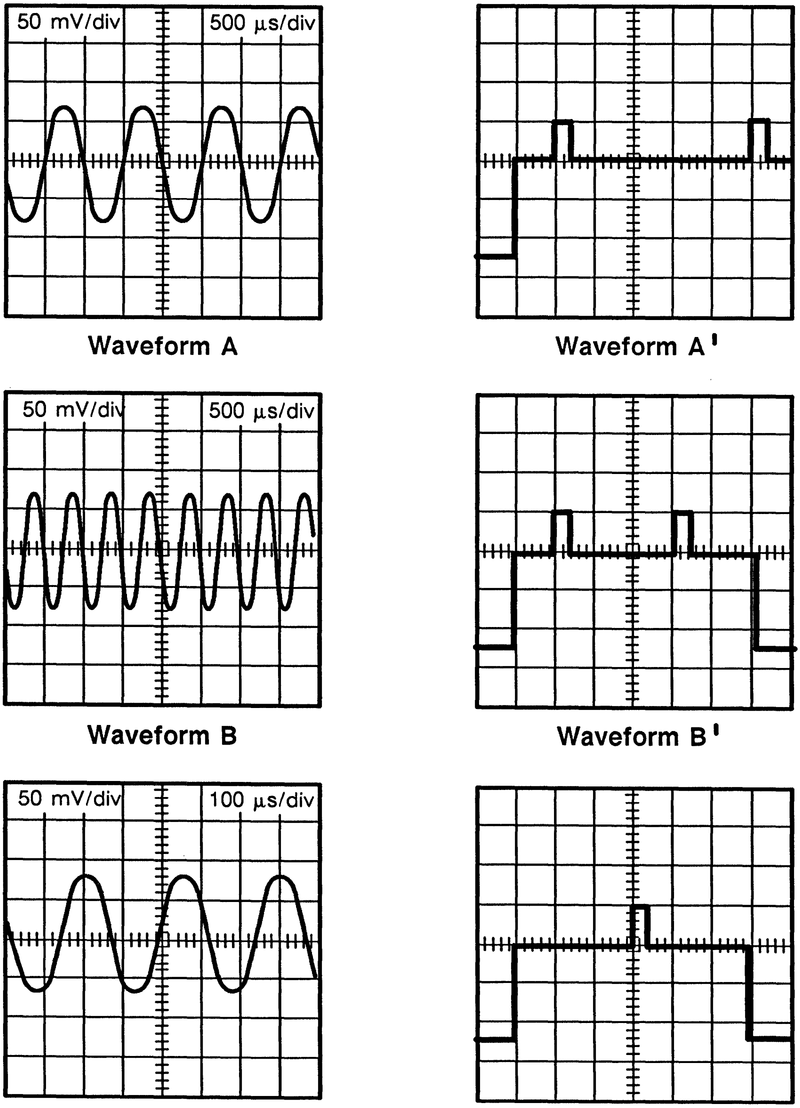
Assignment Jump to FFT Computation (0080–xxxx)-- assigns subroutine jump to FFT computation area.

Data Outppput (000D–001B)-- the output data consists of the square of the imaginary value and the square of the real value; the data is output by the MB8764 DSP in the "I" mode.

The FFT computation area of ROM provides temporary storage for coefficients (SIN and COS data ) used in butterfly computations and also provide sufficient memory space for the actual butterfly calculations.

The bit reverse address data area stores the address data required to properly arrange and correlate the input data. The SIN/COS data tables provide the sine and cosine values for each 1/32nd radian from 0 to $\pi$.

Some examples of a 64–point FFT are shown in Figure 4.6. Waveforms A, B, and C are input signals at different frequencies; waveforms, $A'$ , $B'$, and $C'$ are the corresponding output, signals. In Figure 4.7, the output signal (waveform $A'$) is plotted to show frequency and power–ratio relationships. Point "1" indicates a frequency of 0 Hz and a power ratio of 0. Since the sampling rate of the FDSP KIT–8764 is 8 kHz, the time interval between points "1" and "2" is limited to a resolution of 4 kHz. The DSP divides the time interval into 32 equal parts and shows the power ratio for each discrete point.



**Waveform A** **Waveform A'**

**Waveform B** **Waveform B'**

**Waveform C** **Waveform C'**

**Figure 4.6 Input and Output Waveforms of 64–Point FFT**

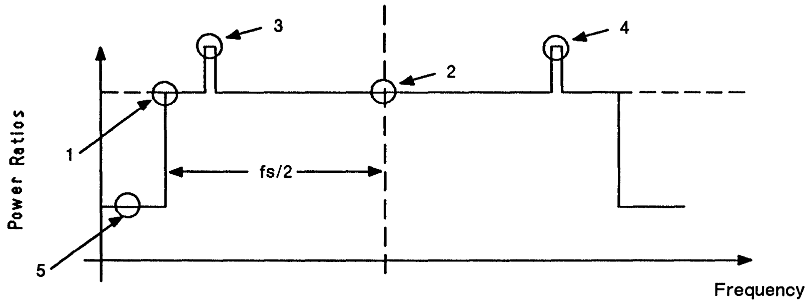


**Figure 4.7 Frequency and Power Ratios of Waveform A' (All points not shown)**

```
MB8764     FDSP3   ASM64  (V01/L01,831130)                     ASSEMBLY LIST          DATE 0.0.0
             INPUT
             C:FFT64P
                         TEXT          FFT64P
NO.         LOC         OP    OPRIND1  OPRIND2        SOURCE STATEMENT

 1                                         *    PRG                  FFT64P
 2                                         *           % INPT OUTPUT %
 3                                         *
 4
 5    0000                                      ORG                  IO,$000
 6    0000      70CF00                          MOV:NOP              #$300,PGT:PGM:X:Y
 7    0001      702040                          MOV:NOP              #$40,C0
 8    0002      D09002                          JOC:NOP              $2,IF
 9    0003      010080                          LTB:NOP              $0(X),$0
10    0004      A14000                          MOV:NOP              A,Y
11    0005      B20BC0                          MOV:NOP              EI,$C0(Y)
12    0006      D04002                          JOC:NOP              $2,C0
13    0007      E04800                          CLR                  Y:A
14    0008      702040                          MOV:NOP              #$40,C0
15    0009      810B80                          MOV:NOP              A,$80(Y)
16    000A      D04009                          JOC:NOP              $9,C0
17    000B      F00080                          JSR:NOP              $80
18    000C      E00100                          CLR                  IF
19    000D      70C000                          MOV:NOP              #$0,Y:X
20    000E      702040                          MOV:NOP              #$40,C0
21    000F      406380                          MOV:NOP              $80(Y),B:A
22    0010      4063C0                          MOV:NOP              $C0(Y),B:A
23    0011      540000                          NOP:MLT
24    0012      580000                          NOP:MSM
25    0013      D09013                          JOC:NOP              $13,IF
26    0014      808000                          MOV:NOP              D,EO
27    0015      F30800                          MOV                  #$800,EA
28    0016      E00100                          CLR                  IF
29    0017      D0400F                          JOC:NOP              $F,C0
30    0018      C08000                          LDI:NOP              #$8000
31    0019      818000                          MOV:NOP              A,EO
32    001A      D0A01A                          JOC:NOP              $1A,OF
33    001B      F30800                          MOV                  #$800,EA
34    001C      F0E000                          JMP:NOP              $0
35                                         *
36                                         *          % FFT64P %
37                                         *
38    0080                                      ORG                  FFT,$080
39    0080      700780                          MOV:NOP              #$380,PGT
40    0081      700800                          MOV:NOP              #$0,PGM
41    0082      E00600                          CLR                  DMM:ADM
42    0083      C00020                          LDI:NOP              #$20
43    0084      810A41                          MOV:NOP              A,$41
44    0085      C00002                          LDI:NOP              #$2
45    0086      810A42                          MOV:NOP              A,$42
46    0087      C00001                          LDI:NOP              #$1
47    0088      810A40                          MOV:NOP              A,$40
48    0089      702006                          MOV:NOP              #$6,C0
49    008A      C00000            GEN01         LDI:NOP              #$0
50    008B      810A02                          MOV:NOP              A,$2
51    008C      910A06                          MOV:NOP              C0,$6
52    008D      602241                          MOV:NOP              $41,C0
53    008E      C00000            GEN02         LDI:NOP              #$0
54    008F      810A01                          MOV:NOP              A,$1
55    0090      910A07                          MOV:NOP              C0.$7
56    0091      704000                          MOV:NOP              #$0,Y
57    0092      602240                          MOV:NOP              $40,C0
58    0093      904A08            GEN03         MOV:NOP              Y,A:$8
59    0094      402202                          MOV:NOP              $2,B
60    0095      422240                          MOV:ADD              $40,B
61    0096      804A00                          MOV:NOP              D,$0:A
62    0097      628201                          MOV:ADD              $1,X
63    0098      800A03                          MOV:NOP              D,$3
```

(Continued from page 4-14)

```
MB8764    FDSP3  ASM64  (VO1/LO1,831130)                 ASSEMBLY LIST          DATE 0.0.0
           INPUT
           C:FFT64P
                   TEXT        FFT64P
NO.      LOC      OP    OPRIND1   OPRIND2    SOURCE STATEMENT

 64    0099    604203                         MOV:NOP         $3,Y
 65    009A    0141C0                         LTB:NOP         $20(x),$40(Y)
 66    009B    018180                         LTB:NOP         $40(X),$0(Y)
 67    009C    054180                         LTB:MLT         $20(X),$0(Y)
 68    009D    0B81C0                         LTB:MRD         $40(X),$40(Y)
 69    009E    840A05                         MOV:MLT         D,$5
 70    009F    684200                         MOV:MSM         $0,Y
 71    00A0    800A04                         MOV:NOP         D,$4
 72    00A1    404205                         MOV:NOP         $5,A
 73    00A2    4023C0                         MOV:NOP         $C0(Y),B
 74    00A3    664203                         MOV:SUB         $3,Y
 75    00A4    800BC0                         MOV:NOP         D,$C0(Y)
 76    00A5    624200                         MOV:ADD         $0.Y
 77    00A6    800BC0                         MOV:NOP         D,$C0(Y)
 78    00A7    800BC0                         MOV:NOP         D,$C0(Y)
 79    00A8    404204                         MOV:NOP         $4,A
 80    00A9    402380                         MOV:NOP         $80(Y),B
 81    00AA    664203                         MOV:SUB         $3,Y
 82    00AB    800B80                         MOV:NOP         D,$80(Y)
 83    00AC    624200                         MOV:ADD         $0,Y
 84    00AD    800B80                         MOV:NOP         D,$80(Y)
 85    00AE    404201                         MOV:NOP         $1,A
 86    00AF    402241                         MOV:NOP         $41,B
 87    00B0    624208                         MOV:ADD         $8,Y
 88    00B1    800A01                         MOV:NOP         D,$1
 89    00B2    D04093                         JOC:NOP         GENO3,C0
 90    00B3    404202                         MOV:NOP         $2,A
 91    00B4    402242                         MOV:NOP         $42,B
 92    00B5    622207                         MOV:ADD         $7,C0
 93    00B6    800A02                         MOV:NOP         D,$2
 94    00B7    D0408E                         JOC:NOP         GEN02,C0
 95    00B8    401241                         MOV:NOP         $41,D
 96    00B9    652206                         MOV:SRA         $6,C0
 97    00BA    800A41                         MOV:NOP         D,$41
 98    00BB    401242                         MOV:NOP         $42,D
 99    00BC    800A40                         MOV:NOP         D,$40
100    00BD    570000                         NOP:SLA
101    00BE    800A42                         MOV:NOP         D,$42
102    00BF    D0408A                         JOC:NOP         GENO1,C0
103    00C0    F02000                         RTS
104                                         *
105                                         *
106                                         *        % TABLE 1%
107    0300                                   ORG        TAB1,$300
108    0300    FF0000                         DCW        $0000:$0020:$0010:$0030:$0008:$0028:$0018:$0038
               FF0020
               FF0010
               FF0030
               FF0008
               FF0028
               FF0018
               FF0038
109    0308    FF0004                         DCW        $0004:$0024:$0014:$0034:$000C:$002C:$001C:$003C
               FF0024
               FF0014
               FF0034
               FF000C
               FF002C
               FF001C
               FF003C
110    0310    FF0002                         DCW        $0002:$0022:$0012:$0032:$000A:$002A:$001A:$003A
               FF0022
               FF0012
               FF0032
               FF000A
               FF002A
               FF001A
               FF003A
 MB8764   FDSP3  ASM64  (VO1/LO1,831130)                  ASSEMBLY LIST          DATE 0.0.0
           INPUT
           C:FFT64P
                   TEXT       FFT64P
 NO.      LOC     OP    OPRIND1   OPRIND2   SOURCE STATEMENT

 111   0318      FF0006                     DCW        $0006:$0026:$0016:$0036:$000E:$002E:$001E:$003E
                 FF0026
                 FF0016
                 FF0036
                 FF000E
                 FF002E
                 FF001E
                 FF003E
 112   0320      FF0001                     DCW        $0001:$0021:$0011:$0031:$0009:$0029:$0019:$0039
                 FF0021
                 FF0011
                 FF0031
                 FF0009
                 FF0029
                 FF0019
                 FF0039                     DCW        $0005:$0025:$0015:$0035:$000D:$002D:$001D:$003D
 113   0328      FF0005
                 FF0025
                 FF0015
                 FF0035
                 FF000D
                 FF002D
                 FF001D
                 FF003D
 114   0330      FF0003                      DCW       $0003:$0023:$0013:$0033:$000B:$002B:$001B:$003B
                 FF0023
                 FF0013
                 FF0033
                 FF000B
                 FF002B
                 FF001B
                 FF003B
 115   0338      FF0007                      DCW       $0007:$0027:$0017:$0037:$000F:$002F:$001F:$003F
                 FF0027
                 FF0017
                 FF0037
                 FF000F
                 FF002F
                 FF001F
                 FF003F
 116                                  *
 117                                  *                    % TABLE 2%
 118                                  *
 119   03A0                                  ORG       TAB2,$3A0
 120   03A0      FF4000                      DCW       $4000:$3FB1:$3EC5:$3D3F:$3B21:$3871:$3537:$3179
                 FF3FB1
                 FF3EC5
                 FF3D3F
                 FF3B21
                 FF3871
                 FF3537
                 FF3179
 121   03A8      FF2D41                      DCW       $2D41:$289A:$238E:$1E2B:$187E:$1294:$0C7C:$0646
                 FF289A
                 FF238E
                 FF1E2B
                 FF187E
                 FF1294
                 FF0C7C
                 FF0646
 122   03B0      FF0000                      DCW       $0000:$F9BA:$F384:$ED6C:$E782:$E1D5:$DC72:$D766
                 FFF9BA
                 FFF384
                 FFED6C
                 FFE782
                 FFE1D5
                 FFDC72
                 FFD766
```

```
MB8764    FDSP3  ASM64  (VO1/LO1,831180)                    ASSEMBLY LIST          DATE 0.0.0
           INPUT
          C:FFT64P
                  TEXT        FFT64P
NO.       LOC       OP    OPRIND1   OPRIND2   SOURCE STATEMENT


123    03B8     FFD2BF                  DCW        $D2BF:$CE87:$CAC9:$C78F:$C4DF:$C2C1:$C13B:$C04F
                FFCE87
                FFCAC9
                FFC78F
                FFC4DF
                FFC2C1
                FFC13B
                FFC04F
124                                   *
125                                   *              % TABLE 3%
126                                   *
127    03C0                             ORG        TAB3,$3C0
128    03C0     FF0000                  DCW        $0000:$0646:$0C7C:$1294:$187E:$1E2B:$238E:$289A
                FF0646
                FF0C7C
                FF1294
                FF187E
                FF1E2B
                FF238E
                FF289A
129    03C8     FF2D41                  DCW        $2D41:$3179:$3537:$3871:$3B21:$3D3F:$3EC5:$3FB1
                FF3179
                FF3537
                FF3871
                FF3B21
                FF3D3F
                FF3EC5
                FF3FB1
130    03D0     FF4000                  DCW        $4000:$3FB1:$3EC5:$3D3F:$3B21:$3871:$3537:$3179
                FF3FB1
                FF3BC5
                FF3D3F
                FF3B21
                FF3871
                FF3537
                FF3179
131    03D8     FF2D41                  DCW        $2D41:$289A:$238E:$1E2B:$187E:$1294:$0C7C:$0646
                FF289A
                FF238E
                FF1E2B
                FF187E
                FF1294
                FF0C7C
                FF0646

132                                   *
133                                   *
134                                   *  END
135
```

```
MB8764   FDSP3   ASM64  (V01/L01,831130)                    ASSEMBLY LIST        DATE 0.0.0

 ID     SYMBOL       START           END         VALUE      TYPE

 1       IO           0000           001C                  SECT ABS
 2       FFT          0080           00C0                  SECT ABS
 3       TAB1         0300           033F                  SECT ABS
 4       TAB2         03A0           03BF                  SECT ABS
 5       TAB3         03C0           03DF                  SECT ABS

MB8764   FDSP3   ASM64  (V01/L01,831130)                    ASSEMBLY LIST        DATE 0.0.0

 SYMBOL    S      VALUE    TYPE     DEF.   REFERENCES (STATEMENT NUMBER)

 GEN01     2      008A      A       49      102
 GEN02     2      008E      A       53       94
 GEN03     2      0093      A       58       89

 END OF ASSEMBLY (NO ERROR)
```

# Section Five - ROM Expansion

**Contents of This Section** **Page**

5.1 Expansion of Instruction ROM ..................................... 5–1
5.2 ROM–Bank Switching ............................................. 5–4

## 5.1 Expansion of Instruction ROM

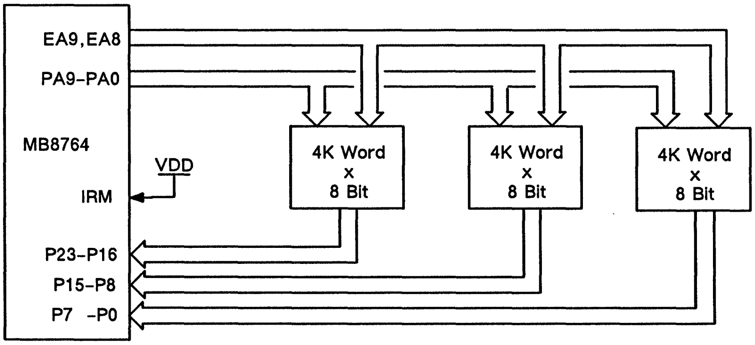
By using the PGM register in the MB8764, the expansion ROM area can be increased from 1K to 4K words; however, the expansion RAM is reduced from 1K to 256 words. The PGM register is controlled solely by the user's program; all program addresses must be precisely organized to avoid errors. A typical expansion circuit and a properly organized address table are shown in Figure 5.1.

Operations for expanding the external ROM area to 4K words are as follows:

- Set IRM to 1.
- Reset PA9–PA0 and PGM to 0.
- Set PGM value in the last statement of each PGM area as shown in Figure 5.1.
- When an unconditional jump to another PGM area is required, two transfer instructions and a jump instruction are used--refer to address/instruction listing that follows.

```
              ADDRESS              INSTRUCTION
                                        :
            ⎧ n-1                       :
            ⎪ n                MOV:NOP   #$a,PGM
 Except a  ⎨
            ⎪ n+1              JMP:NOP   $m
            ⎩ n+2                       :
                                        :
        a  ⎧ m                 NOP      :
           ⎩ m+1
```

At the destination address of the jump, a NOP instruction must be included. If the jump is to the same PGM area, there are no restrictions. With a conditional jump, it is not possible to assign the same PGM area as the destination.



| PGM | PA9-PA0 | External ROM |
|---|---|---|
| 0 | 000 | |
| | 3FE | MOV:NOP #$1,PGM |
| 0 | 3FF | JMP:NOP $0 |
| 1 | 000 | |
| | 3FE | MOV:NOP #$2,PGM |
| 1 | 3FF | JMP:NOP $0 |
| 2 | 000 | |
| | 3FE | MOV:NOP #$3,PGM |
| 2 | 3FF | JMP:NOP $0 |
| 3 | 000 | |
| | 3FE | MOV:NOP #$0,PGM |
| 3 | 3FF | JMP:NOP $0 |

**Figure 5.1 External ROM Expansion and Address Table**

A timing diagram for external ROM expansion is shown in Figure 5.2.

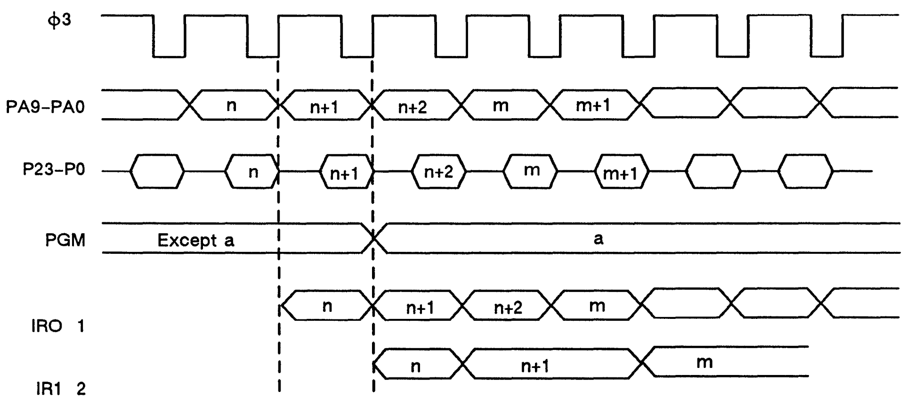


**Figure 5.2 EROM Timing**

## 5.2 ROM-Bank Switching

The instruction area of ROM can be expanded to 4K words by simply connecting the internal/external ROM-switching pin (IRM) to EA8 or EA9. When both EA8 and EA9 are set to 0, external ROM (EROM) is selected. Figures 5.3 and 5.4 show how internal ROM (IROM) and EROM can be used together to serve a variety of applications.

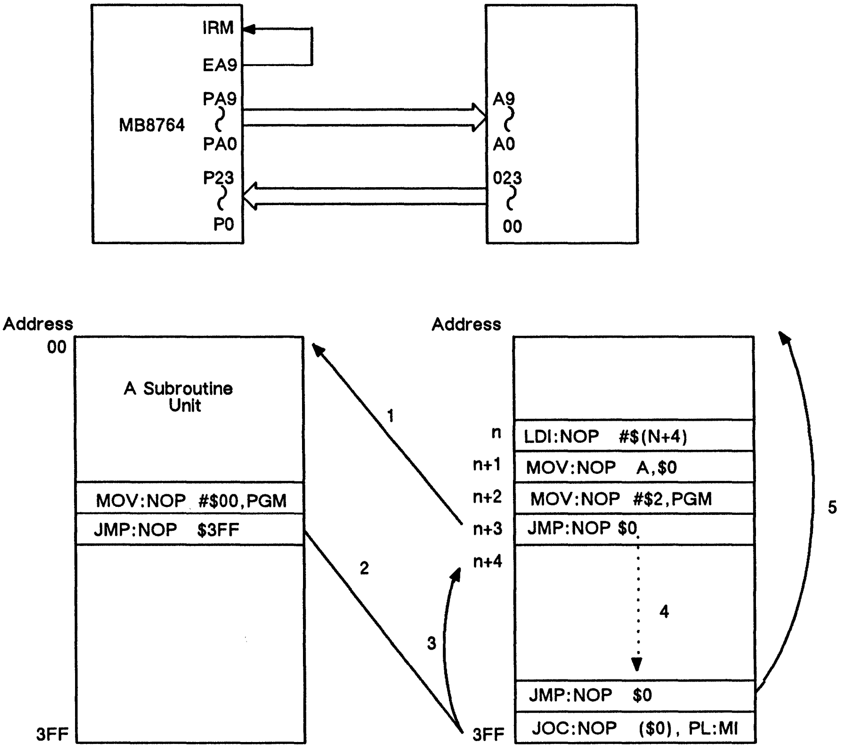


**Figure 5.3 IK Word IROM And IK Word EROM**

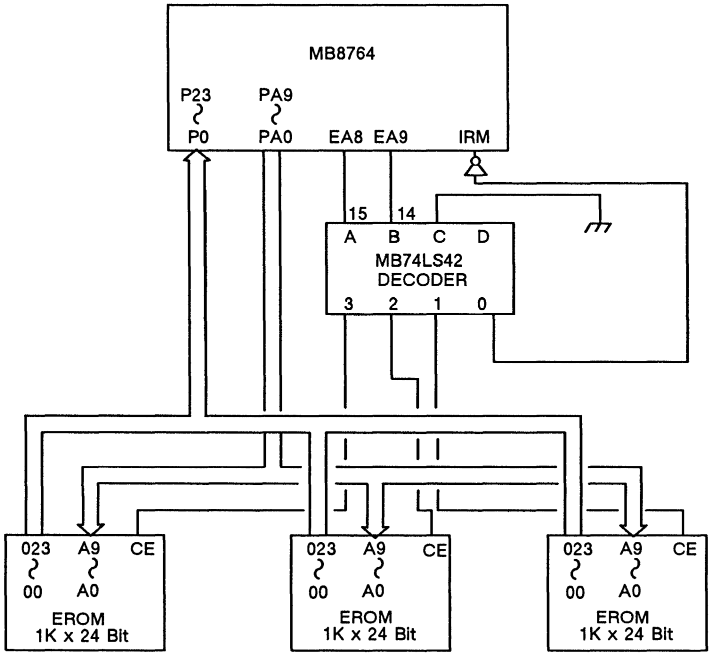


**Figure 5.4 1K Word IROM And 3K Word EROM**

# Appendix A
# FFT Computations

Computations for Fast Fourier Transforms (FFT) use two digital processes to plot frequencies and power ratios of a complex waveform. In order of execution, these processes are Bit Reverse Processing and Butterfly Computations. The bit reversal process is shown in Figure A.1.

| Original | | Bit Reversed | |
|---|---|---|---|
| Digit | Binary | Binary | Digit |
| 0 | 000 | 000 | 0 |
| 1 | 001 | 001 | 4 |
| 2 | 010 | 010 | 2 |
| 3 | 011 | 110 | 6 |
| 4 | 100 | 001 | 1 |
| 5 | 101 | 101 | 5 |
| 6 | 110 | 011 | 3 |
| 7 | 111 | 111 | 7 |

**Figure A.1 Bit-Reversal Process**

In the example shown in Section Four, the bit-reversed data is prepared in ROM and stored in queue in RAM. Butterfly computations are then performed. Figure A.2 shows the computations for an 8-point FFT; the example in Section Four has been expanded to a 64-point plot.

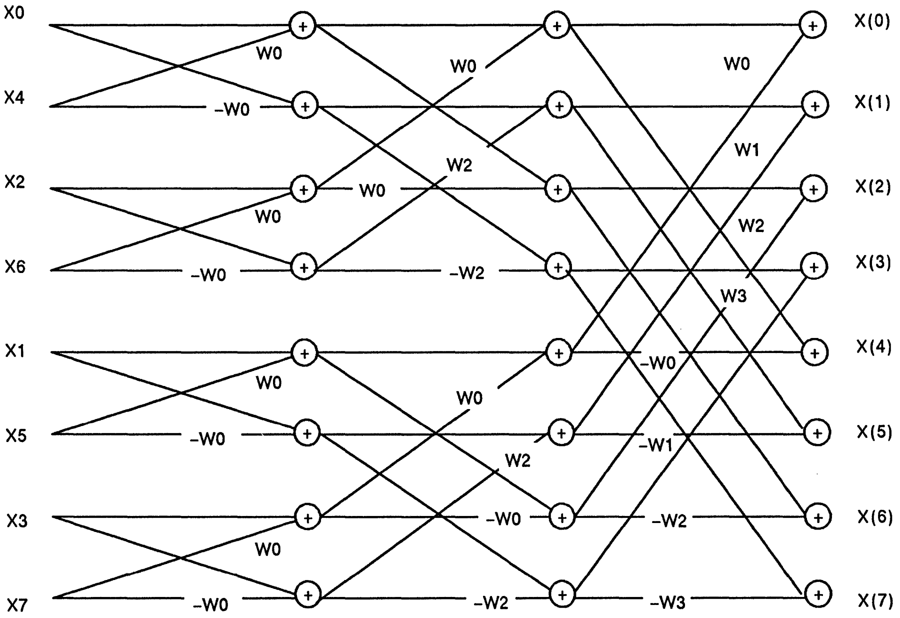


**Figure A.2 Computations for 8-Point FFT**

# Appendix B
# MB8764 Data Sheet

FUJITSU

# GENERAL PURPOSE DIGITAL SIGNAL PROCESSOR

# MB 8764

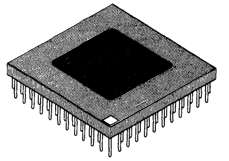
December 1985
Edition 2.0

## GENERAL PURPOSE DIGITAL SIGNAL PROCESSOR

The Fujitsu MB 8764 is a general purpose silicon-gate CMOS digital signal processor (DSP) integrated circuit. The MB 8764 features a high-speed pipelined multiplier, supports concurrent operations with compound instructions and multiple data paths, offers flexible and expandable memory options and has an on-chip DMA channel.

With its high-speed operation, the MB 8764 gives high throughput in various applications, such as telecommunications, signal processing and image processing.

Being packaged in the 88-pin pin grid array, the MB 8764 allows a complex system to be built with the external program ROM and data RAM accessed through dedicated address and data buses.

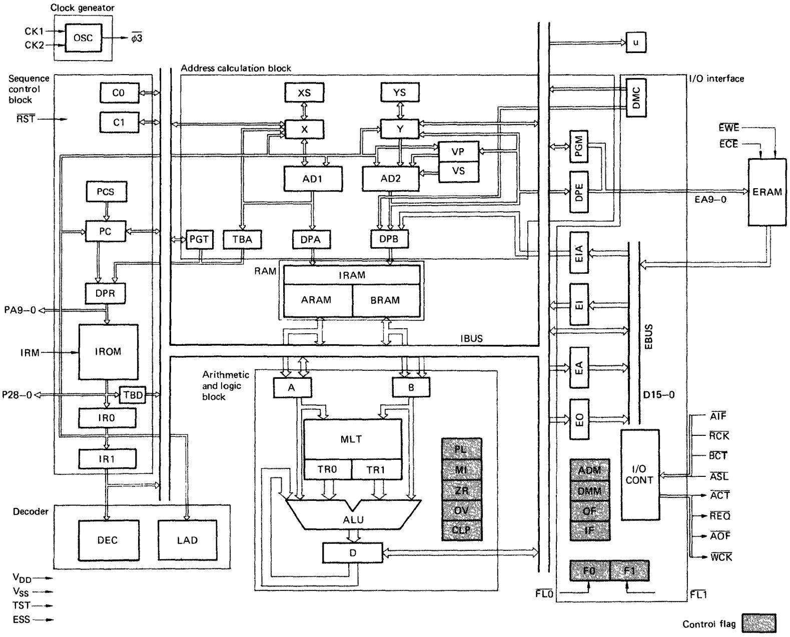


CERAMIC PACKAGE
RIT-88C-A01

- General purpose high-speed digital signal processing
- High speed operation
  - 100ns cycle time
- Parallel pipelined multiply function
  - 16 bits x 16 bits → 26 bits
- Divide function
  - 26 bits ÷ 16 bits → 16 bits
- Program ROM
  - 1024 words x 24 bits
  - Internal (mask-programmed) and external ROM selectable
- Part of the program ROM can be used for constant data storage
- Two built-in 128 x 16 bits RAMs
- Expansion RAM function
  - Expandable up to 1024 words x 16 bits
  - Two access speed rates can be selected
- Numerous I/O functions
  - 16-bit parallel interface
  - Three input modes and two output modes including DMA
- Powerful instruction set using compound instructions
  - One level of subroutine nesting (multi-level nesting can be programmed)
  - Two levels of loop nesting (multi-level nesting can be programmed)
  - Compound instructions (for example, an arithmetic/logic instruction combined with a move instruction) enable concurrent processing
  - 15 arithmetic/logic instructions
- Addressing
  - Direct addressing
  - Indexed addressing
  - Immediate addressing
  - Virtual shift addressing
- Silicon-gate CMOS process
- Single 5 volt power supply, TTL I/O interface (except pins for clock signals)
- 88-pin space-saving pin grid array package
- Support tool, including cross-assembly software and evaluation board

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit.

## Fig. 1 – BLOCK DIAGRAM



# FUNCTION OF BLOCK

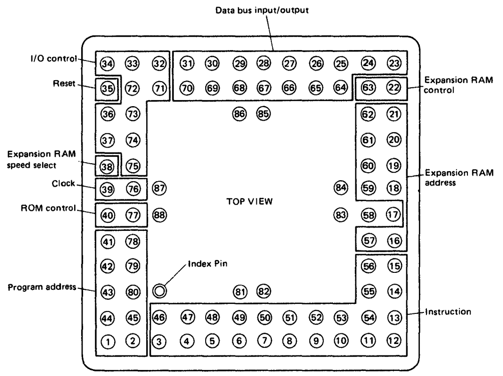
| Block | Function |
|---|---|
| Clock generator | This block generates a cycle clock ($\overline{\phi3}$) used for internal operations. The clock pulses can be generated by supplying a clock signal from an external circuit through external pins CK1 and CK2, or by a crystal resonator and capacitors connected to CK1 and CK2. The master clock (MCLK) obtained by either of the above two methods has the same frequency as that of the CK1/CK2 clock and generates a cycle clock $\overline{\phi3}$ having the frequency of the machine cycle (which is half the MCLK frequency). All internal operations are timed by the cycle clock $\overline{\phi3}$. |
| Sequence control block | This block controls the DSP instruction execution sequence. The program counter (PC) is reset to address 0 by the $\overline{\text{RST}}$ pulse, and is incremented by 1 at each leading edge of $\overline{\phi3}$ after $\overline{\text{RST}}$ is turned off. The PC output is connected to the address input of the internal microinstruction ROM (IROM) via the ROM pointer (DPR), and the ROM data is read out sequentially according to the PC value.<br>The DPR value is also output through PA9 to PA0 to the outside to permit access to an external ROM (EROM). Data from the EROM is input to the MB 8764 through P23 to P0. At any given time, either the IROM or EROM can be used, and the choice is controlled by the IRM input. The IROM is a mask ROM with a capacity of 1,024 words x 24 bits. The ROM that has the same organization can be used for the EROM. The IROM and EROM are functionally identical. The ROM output data is transferred to the instruction register IR0 at the beginning of a cycle (that is, at the leading edge of $\overline{\phi3}$), moved to the instruction register IR1 at the beginning of the following cycle, then decoded and executed.<br>To perform a branch instruction,address can be loaded into PC through IR0 and the IBUS, and the PC value can be saved in RAM or in another register through the IBUS. PCS is single PC stack used for subroutine execution. Two loop counters, C0 and C1, are provided to facilitate the handling of loops.<br>This block also has a cycle counter (CYC) that controls execution of multi-cycle instructions. This counter automatically stops incrementing PC during execution of a multi-cycle instruction. |

# FUNCTION OF BLOCK (Cont'd)

| Block | Function |
|---|---|
| Decoder | Instruction codes fetched from the instruction ROM and transferred to instruction registers IR0 and IR1 at the beginning of each cycle are moved to the look-ahead decoder (LAD) and decoder (DEC), respectively, then interpreted and executed. Execution of an instruction (the execution cycle) usually takes place while the instruction is stored in IR1. The DEC output controls the enable lines of the registers required for execution.<br>Before an instruction is executed, LAD controls calculation of the effective address in RAM, interprets operations to be performed in the arithmetic and logic block, and decodes the number of cycles required for the instruction. The number of cycles required for an instruction is the number of machine cycles during which the instruction is stored in IR1. |
| Address calculation block | This block calculates the effective (execution) address in RAM (IRAM/ERAM) or ROM (table ROM).<br>The address calculation block consists of index registers X and Y, stacks XS and YS for index registers X and Y, a 7-bit adder (AD1), an 8-bit adder (AD2), the virtual shift pointer (VP), and the virtual shift mode register (VS).<br>An effective address is calculated in the LAD cycle, and the result is used as the execution address in the following execution cycle. An address in the table ROM is first calculated in AD1, then used to read table data through the table address register (TBA) and ROM pointer (DPR).<br>To access IRAM by an instruction having one address, the effective address is first calculated in AD2, then the result is used to access IRAM through the RAM pointer (DPB). To access IRAM by an instruction having two addresses, the effective address in ARAM is calculated in AD1, the effective address in BRAM is calculated in AD2, and the results are used to access ARAM and BRAM through DPA and DPB.<br>An address in ERAM is calculated by AD2 and the result is used to access ERAM through the ERAM pointer (DPE).<br>Note that the table ROM is accessed by adding the value of page register PGT as the MSB element of the address, and the ROM data (16-bit) is output to IBUS through TBD. ERAM is accessed by adding the value of page register PGM as the MSB element of the address. |
| RAM | This device has two 128-word x 16-bit RAM areas called ARAM and BRAM. ARAM and BRAM can be used as two independent RAMs, or as a single RAM (IRAM) having a continuous address space. If the internal RAM is not sufficient, an external RAM (ERAM) can be connected to the chip. The ERAM can be used as an extension of BRAM or IRAM, but its address space is independent of BRAM or IRAM. |
| Arithmetic and logic block | Arithmetic and logic instructions are executed in this block. Execution of an instruction is timed by the machine cycle. This block consists of input registers A and B, an accumulator D that receives the operation result, a multiplier MLT, and an arithmetic and logic unit ALU.<br>Multiplication is performed by à two-stage parallel multiplier in which MLT and ALU functions are pipelined.<br>MLT multiplies the values of A and B unconditionally at each instruction and stores the intermediate results in the temporary registers TR0 and TR1. The final result of multiplication is obtained by having the ALU add the values of TR0 and TR1 according to a subsequent multiply instruction. Since the multiplier has a two-stage pipeline structure, it takes two cycles to obtain the multiplication result in D after data have been loaded into A and B.<br>Operations other than multiplication are performed by the ALU alone, and the result is stored directly in D.<br>The arithmetic and logic block also includes operation flags (PL, MI, ZR, and OV) that can be used to indicate conditions for conditional branch instructions. Register D has a longer bit length than the internal bus (IBUS), so a control register CLP is provided to output clipped data when the D value overflows the IBUS. |
| I/O interface | The I/O interface is used to exchange data between the DSP chip and an external circuit. It consists of I/O registers, an I/O controller and flags. The I/O controller controls data transfer to/from the external circuit independently of the execution of instructions.<br>Data can be input from an external circuit through EI with or without address information through EIA. There are three input modes: the P, D, and A modes. These modes are distinguished by values set by instructions in the mode registers ADM and DMM. When data is set in EI, the input flag IF is set. In the P mode, the EI value is transferred to another register or to RAM by the program. In the D or A mode, the EI value is transferred to IRAM by cycle stealing. In the D mode, DMC is selected as the IRAM address, while in the A mode, EIA is selected. IF is reset when the EI contents are transferred to another location.<br>Data is output to an external circuit through EA and EO. There are two output modes, and they are distinguished by the instruction data placed in EA.<br>OF is set when data is placed in EA, and is reset when data output to the external circuit is completed.<br>The data exchange between the DSP and an external circuit as explained above is performed through I/O control pins for synchronization with the external circuit.<br>The I/O interface also includes the F0 and F1 flags. These are set by external input signals and used for program control or synchronization. |

# PIN ASSIGNMENT

Fig. 2 – PIN ASSIGNMENT



| No. | Name | No. | Name | No. | Name | No. | Name | No. | Name | No. | Name |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | PA1 | 16 | EA0 | 31 | $D_{14}$ | 46 | P23 | 61 | EA7 | 76 | CK2 |
| 2 | PA0 | 17 | – | 32 | $\overline{\mathrm{REQ}}$ | 47 | P21 | 62 | EA9 | 77 | TST |
| 3 | P22 | 18 | EA3 | 33 | $\overline{\mathrm{BCT}}$ | 48 | P18 | 63 | $\overline{\mathrm{ECE}}$ | 78 | PA9 |
| 4 | P20 | 19 | EA5 | 34 | $\overline{\mathrm{RCK}}$ | 49 | P16 | 64 | $D_2$ | 79 | PA6 |
| 5 | P19 | 20 | EA6 | 35 | $\overline{\mathrm{RST}}$ | 50 | P15 | 65 | $D_4$ | 80 | PA4 |
| 6 | P17 | 21 | EA8 | 36 | $\overline{\mathrm{FLO}}$ | 51 | P13 | 66 | $D_7$ | 81 | GND |
| 7 | P14 | 22 | $\overline{\mathrm{EWE}}$ | 37 | $\overline{\mathrm{WCK}}$ | 52 | P10 | 67 | $D_9$ | 82 | $V_{CC}$ |
| 8 | P12 | 23 | $D_0$ | 38 | ESS | 53 | P8 | 68 | $D_{10}$ | 83 | $V_{CC}$ |
| 9 | P11 | 24 | $D_1$ | 39 | CK1 | 54 | P6 | 69 | $D_{12}$ | 84 | GND |
| 10 | P9 | 25 | $D_3$ | 40 | IRM | 55 | P3 | 70 | $D_{15}$ | 85 | GND |
| 11 | P7 | 26 | $D_5$ | 41 | PA8 | 56 | P1 | 71 | $\overline{\mathrm{ACT}}$ | 86 | $V_{CC}$ |
| 12 | P5 | 27 | $D_6$ | 42 | PA7 | 57 | EA1 | 72 | $\overline{\mathrm{AIF}}$ | 87 | $V_{CC}$ |
| 13 | P4 | 28 | $D_8$ | 43 | PA5 | 58 | – | 73 | $\overline{\mathrm{FL1}}$ | 88 | GND |
| 14 | P2 | 29 | $D_{11}$ | 44 | PA3 | 59 | EA2 | 74 | $\overline{\mathrm{AOF}}$ | | |
| 15 | P0 | 30 | $D_{13}$ | 45 | PA2 | 60 | EA4 | 75 | $\overline{\mathrm{ASL}}$ | | |

| No. | Name | I/O | Function |
|---|---|---|---|
| 78 | PA9 | Output | Program address MSB |
| 41 | PA8 | Output | Program address BIT8 |
| 42 | PA7 | Output | Program address BIT7 |
| 79 | PA6 | Output | Program address BIT6 |
| 43 | PA5 | Output | Program address BIT5 |
| 80 | PA4 | Output | Program address BIT4 |
| 44 | PA3 | Output | Program address BIT3 |
| 45 | PA2 | Output | Program address BIT2 |
| 1 | PA1 | Output | Program address BIT1 |
| 2 | PA0 | Output | Program address LSB |
| 46 | P23 | I/O | Instruction MSB |
| 3 | P22 | I/O | Instruction BIT22 |
| 47 | P21 | I/O | Instruction BIT21 |
| 4 | P20 | I/O | Instruction BIT20 |
| 5 | P19 | I/O | Instruction BIT19 |
| 48 | P18 | I/O | Instruction BIT18 |
| 6 | P17 | I/O | Instruction BIT17 |
| 49 | P16 | I/O | Instruction BIT16 |
| 50 | P15 | I/O | Instruction BIT15 |
| 7 | P14 | I/O | Instruction BIT14 |
| 51 | P13 | I/O | Instruction BIT13 |
| 8 | P12 | I/O | Instruction BIT12 |
| 9 | P11 | I/O | Instruction BIT11 |
| 52 | P10 | I/O | Instruction BIT10 |
| 10 | P9 | I/O | Instruction BIT9 |
| 53 | P8 | I/O | Instruction BIT8 |
| 11 | P7 | I/O | Instruction BIT7 |
| 54 | P6 | I/O | Instruction BIT6 |
| 12 | P5 | I/O | Instruction BIT5 |
| 13 | P4 | I/O | Instruction BIT4 |

| No. | Name | I/O | Function |
|---|---|---|---|
| 55 | P3 | I/O | Instruction BIT3 |
| 14 | P2 | I/O | Instruction BIT2 |
| 56 | P1 | I/O | Instruction BIT1 |
| 15 | P0 | I/O | Instruction LSB |
| 39 | CK1 | Input | Master clock input pin 1 |
| 76 | CK2 | Input | Master clock input pin 2 |
| 35 | $\overline{\mathrm{RST}}$ | Input | Initialization |
| 40 | IRM | Input | Internal/external ROM switching |
| 77 | TST | Input | Internal ROM test mode |
| 62 | EA9 | Output | Expansion RAM address MSB |
| 21 | EA8 | Output | Expansion RAM address BIT8 |
| 61 | EA7 | Output | Expansion RAM address BIT7 |
| 20 | EA6 | Output | Expansion RAM address BIT6 |
| 19 | EA5 | Output | Expansion RAM address BIT5 |
| 60 | EA4 | Output | Expansion RAM address BIT4 |
| 18 | EA3 | Output | Expansion RAM address BIT3 |
| 59 | EA2 | Output | Expansion RAM address BIT2 |
| 57 | EA1 | Output | Expansion RAM address BIT1 |
| 16 | EAO | Output | Expansion RAM address LSB |
| 70 | D15 | I/O | Data bus I/O MSB |
| 31 | D14 | I/O | Data bus I/O BIT14 |

| No. | Name | I/O | Function |
|---|---|---|---|
| 30 | D13 | I/O | Data bus I/O BIT13 |
| 69 | D12 | I/O | Data bus I/O BIT12 |
| 29 | D11 | I/O | Data bus I/O BIT11 |
| 68 | D10 | I/O | Data bus I/O BIT10 |
| 67 | D9 | I/O | Data bus I/O BIT9 |
| 28 | D8 | I/O | Data bus I/O BIT8 |
| 66 | D7 | I/O | Data bus I/O BIT7 |
| 27 | D6 | I/O | Data bus I/O BIT6 |
| 26 | D5 | I/O | Data bus I/O BIT5 |
| 65 | D4 | I/O | Data bus I/O BIT4 |
| 25 | D3 | I/O | Data bus I/O BIT3 |
| 64 | D2 | I/O | Data bus I/O BIT2 |
| 24 | D1 | I/O | Data bus I/O BIT1 |
| 23 | D0 | I/O | Data bus I/O LSB |
| 34 | $\overline{\mathrm{RCK}}$ | Input | Data read clock |
| 33 | $\overline{\mathrm{BCT}}$ | Input | Data bus output enable |
| 72 | $\overline{\mathrm{AIF}}$ | Input | Data input request |
| 36 | $\overline{\mathrm{FLO}}$ | Input | Flag input |
| 73 | $\overline{\mathrm{FL1}}$ | Input | Flag input |
| 75 | $\overline{\mathrm{ASL}}$ | Input | Data output type specification in E mode |
| 37 | $\overline{\mathrm{WCK}}$ | Output | Data write clock |
| 74 | $\overline{\mathrm{AOF}}$ | Output | Output data type specification in I mode |
| 71 | $\overline{\mathrm{ACT}}$ | Output | Input enable |
| 32 | $\overline{\mathrm{REQ}}$ | Output | Data bus request |
| 22 | $\overline{\mathrm{EWE}}$ | Output | ERAM write clock |
| 63 | $\overline{\mathrm{ECE}}$ | Output | ERAM chip enable |
| 38 | ESS | Input | ERAM speed select |

# ABSOLUTE MAXIMUM RATINGS*1

| Parameter | Symbol | Rating | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| Power supply voltage | $V_{CC}$ | -0.3*2 | 7.0 | V |
| Input voltage | $V_I$ | -0.3*2 | $V_{CC}$ + 0.3*2 | V |
| Output voltage | $V_O$ | -0.3*2 | $V_{CC}$ + 0.3*2 | V |
| Operating temperature | $T_{OP}$ | 0 | 85 | °C |
| Storage temperature | $T_{STG}$ | -55 | 150 | °C |

**Note:** *1 Permanent device damage may occur if the absolute maximum ratings are exceeded. Functional operation should be restricted to the conditions as detailed in the operational sections of this data sheet. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

*2 This value applies in a steady condition. It may be 0.5 V in a transient condition (for 20 to 30 ns).

# RECOMMENDED OPERATING CONDITIONS

| Parameter | Symbol | Condition | | | Unit |
|---|---|---|---|---|---|
| | | Min | Typ | Max | |
| Power supply voltage | $V_{CC}$ | 4.5 | 5.0 | 5.5 | V |
| Operating temperature | $T_{OP}$ | 0 | | 85 | °C |

# DC CHARACTERISTICS

**(Recommended Operating Conditions unless otherwise specified.)**

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Input high voltage | $V_{IH}$ | Other than CK1, CK2 | 2.0 | | $V_{CC}$ +0.3 | V |
| | $V_{IHCK}$ | CK1, CK2 | 4.0 | | $V_{CC}$ +0.3 | V |
| Input low voltage | $V_{IL}$ | Other than CK1, CK2 | -0.3 | | 0.8 | V |
| | $V_{ILCK}$ | CK1, CK2 | -0.3 | | 0.6 | V |
| Output high voltage | $V_{OH}$ | $I_{OH}$ = -0.4 mA | 2.7 | | $V_{CC}$ | V |
| Output low voltage | $V_{OL}$ | $I_{OL}$ = 2 mA | | | 0.4 | V |
| Input leakage current | $I_{LI}$ | $V_I$ = 0 to 5.5 V | -25 | | 25 | μA |
| Input leakage current (Three-state pin input) | $I_{LZ}$ | $V_I$ = 0 to 5.5 V | -40 | | 40 | μA |
| Static power supply current | $I_{CCS}$ | | | 1 | | mA |
| Power supply current | $I_{CC}$ | $f_{OP}$ = 8 MHz | | 60 | | mA |

# CAPACITANCE

**($V_{CC}$ = $V_I$ = 0 V, $f_M$ = 8 MHz)**

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Input pin | $C_{IN}$ | | | 5 | pF |
| Output pin | $C_{OUT}$ | | | 5 | pF |
| I/O pin | $C_{I/O}$ | | | 8 | pF |

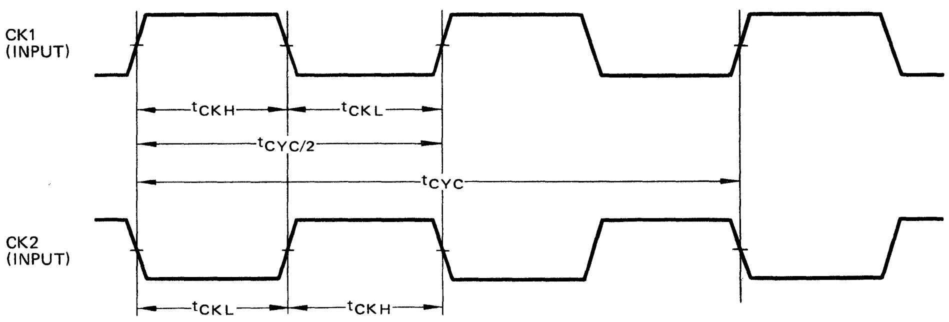
# AC CHARACTERISTICS

## EXTERNAL CLOCK TIMING

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Cycle time *1 | $t_{CYC}$ | 100 | | | ns |
| High voltage pulse width | $t_{CKH}$ | 20 | | | ns |
| Low voltage pulse width | $t_{CKL}$ | 20 | | | ns |

**Note:** *1 Value when ERAM (extended RAM) is not used. When ERAM is used, follow the specifications for the ERAM interface AC characteristics. This note also applies to the following AC characteristics.

**Fig. 3 – EXTERNAL CLOCK TIMING DIAGRAM**



## INTERNAL OSCILLATOR (Crystal oscillator connected)

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Cycle time | $t_{CYC}$ | 100 | | | ns |
| Crystal frequency | $f_{CYC}$ | | 16 | 20 | MHz |

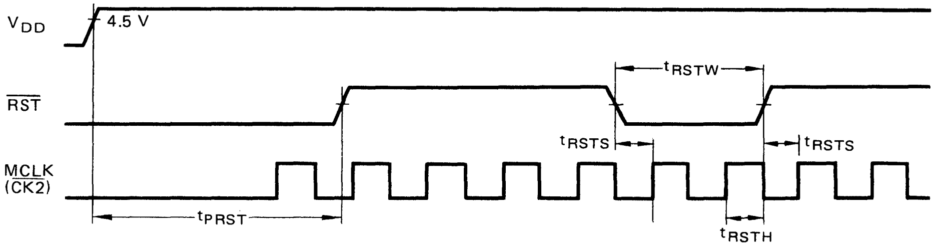
## RESET INPUT TIMING

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Power-on reset *1 | $t_{PRST}$ | | 1 | | ms |
| MCLK setup *2 | $t_{RSTS}$ | 20 | | | ns |
| MCLK hold *2 | $t_{RSTH}$ | 15 | | | ns |
| Reset input pulse width | $t_{RSTW}$ | $t_{CYC}$ +35 | | | ns |

**Note:** *1 The time specification for power-on reset applies to the internal oscillation mode.
In the external clock mode, the reset pulse must be entered so that the leading edge of MCLK ($\overline{CK2}$) can be produced while $\overline{RST}$ = 0.

*2 In the external clock mode, MCLK is considered to be $\overline{CK2}$ (the inversion of the clock input from CK2). This note also applies to the following AC characteristics.

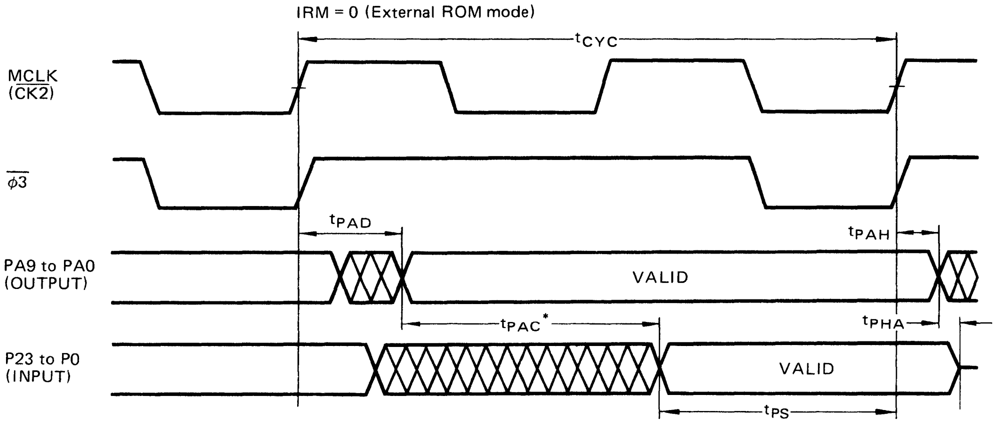
### Fig. 4 – RESET INPUT TIMING DIAGRAM



## EXTERNAL ROM INTERFACE TIMING

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Address output delay (from MCLK pulse) | $t_{PAD}$ | $C_L$ = 50pF | | 60 | 75 | ns |
| Address output hold (from MCLK pulse) | $t_{PAH}$ | $C_L$ = 50pF | 20 | | | ns |
| Data hold time (to address) | $t_{PHA}$ | $C_L$ = 50pF | 0 | | | ns |
| Data setup (before MCLK pulse) | $t_{PS}$ | $C_L$ = 50pF | 10 | 10 | | ns |

### Fig. 5 – EXTERNAL ROM INTERFACE TIMING DIAGRAM



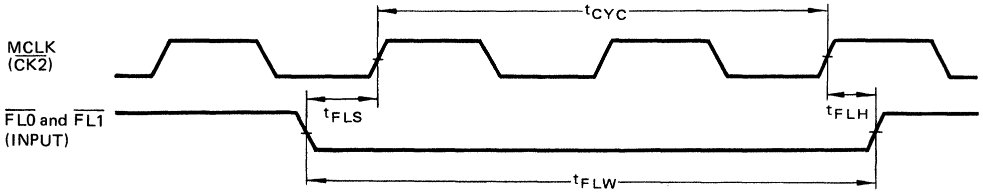
**Note:** *$t_{PAC}$ is the address access time provided by the specification of the external ROM.

### FLAG ($\overline{FL0}$ and $\overline{FL1}$) INPUT TIMING

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Setup time | $t_{FLS}$ | 15 | | | ns |
| Hold time | $t_{FLH}$ | 30 | | | ns |
| Pulse width*1 | $t_{FLW}$ | $t_{CYC}$ + 45 | | | ns |

**Note:** *1 $t_{FLW}$ (Min) = 2 x $t_{CYC}$ + 45 when ERAM is used with ESS = 1.

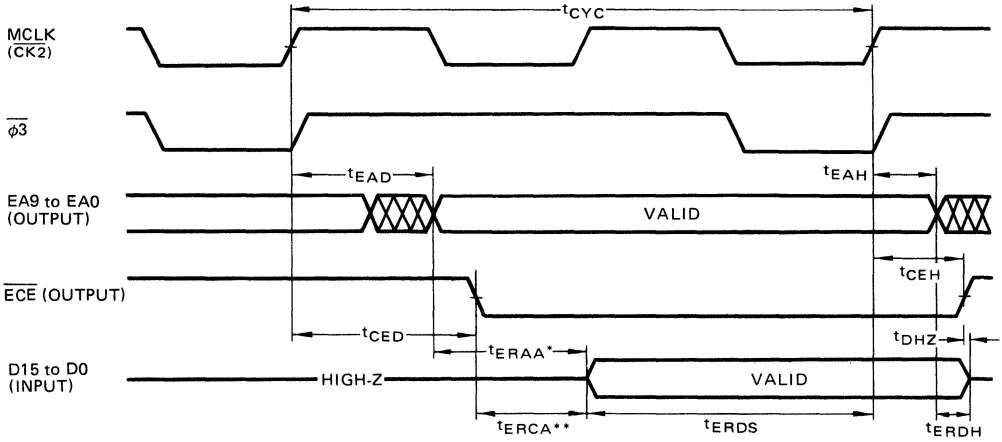
**Fig. 6 – FLAG INPUT TIMING DIAGRAM**



### EXPANSION RAM INTERFACE TIMING (ESS = 0, Read Cycle)

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Address output delay | $t_{EAD}$ | $C_L$ = 50pF | | 50 | 60 | ns |
| Address output hold | $t_{EAH}$ | $C_L$ = 50pF | 10 | 13 | | ns |
| Chip enable output delay | $t_{CED}$ | $C_L$ = 50pF | | 57 | 70 | ns |
| Chip enable output hold | $t_{CEH}$ | $C_L$ = 50pF | 17 | 19 | | ns |
| Output disable | $t_{DHZ}$ | $C_L$ = 50pF | 0 | | | ns |
| Data input setup time | $t_{ERDS}$ | $C_L$ = 50pF | 30 | 25 | | ns |
| Data input hold time | $t_{ERDH}$ | $C_L$ = 50pF | 0 | | | ns |

**Fig. 7 – EXPANSION RAM INTERFACE TIMING DIAGRAM (ESS = 0, Read Cycle)**

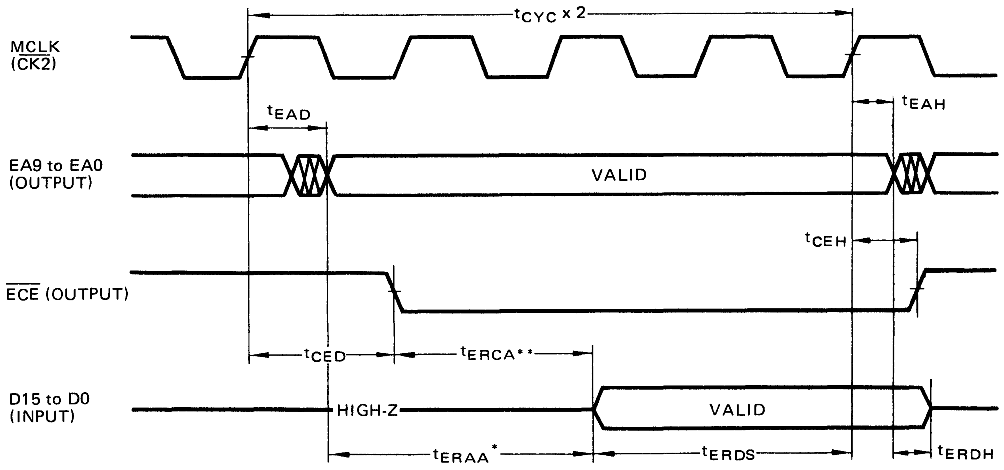


**Note:** *$t_{ERAA}$ is the address access time provided by the specification of the expansion RAM.
**$t_{ERCA}$ is the chip select access time provided by the specification of the expansion RAM.

### EXPANSION RAM INTERFACE TIMING (ESS = 1, Read Cycle)

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Address output delay | $t_{EAD}$ | $C_L$ = 50pF |  | 50 | 60 | ns |
| Address output hold | $t_{EAH}$ | $C_L$ = 50pF | 10 | 13 |  | ns |
| Chip enable output delay | $t_{CED}$ | $C_L$ = 50pF |  | 57 | 70 | ns |
| Chip enable output hold | $t_{CEH}$ | $C_L$ = 50pF | 17 | 19 |  | ns |
| Data input setup time | $t_{ERDS}$ | $C_L$ = 50pF | 30 | 25 |  | ns |
| Data input hold time | $t_{ERDH}$ | $C_L$ = 50pF | 0 |  |  | ns |

**Fig. 8 — EXPANSION RAM INTERFACE TIMING DIAGRAM (ESS = 1, Read Cycle)**

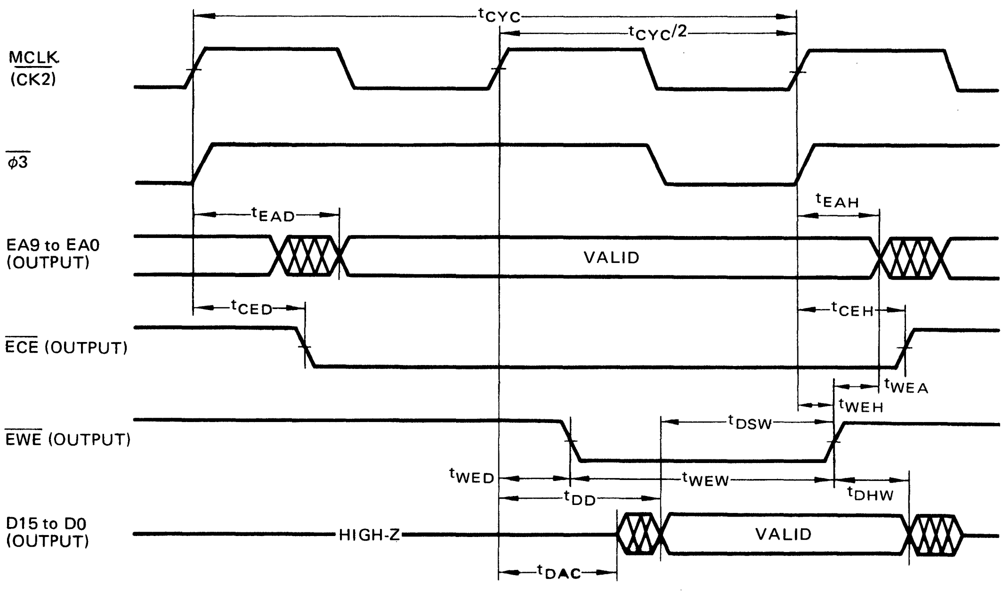


**Note:** *$t_{ERAA}$ is the address access time provided by the specification of the expansion RAM.
**$t_{ERCA}$ is the chip select access time provided by the specification of the expansion RAM.

### EXPANSION RAM INTERFACE TIMING (ESS = 0, Write Cycle)

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Address output delay | $t_{EAD}$ | $C_L$ = 50pF | | 50 | 60 | ns |
| Address output hold | $t_{EAH}$ | $C_L$ = 50pF | 10 | 13 | | ns |
| Address hold (after $\overline{EWE}$) | $t_{WEA}$ | $C_L$ = 50pF | 5 | | | ns |
| Chip enable output delay | $t_{CED}$ | $C_L$ = 50pF | | 57 | 70 | ns |
| Chip enable output hold | $t_{CEH}$ | $C_L$ = 50pF | 17 | 19 | | ns |
| Write enable output delay | $t_{WED}$ | $C_L$ = 50pF | | 40 | 50 | ns |
| Write enable output hold | $t_{WEH}$ | $C_L$ = 50pF | 5 | | 35 | ns |
| Write enable pulse width | $t_{WEW}$ | $C_L$ = 50pF | $\frac{t_{CYC}}{2} - 30$ | | | ns |
| Data output delay | $t_{DD}$ | $C_L$ = 50pF + 1TTL | | 52 | 70 | ns |
| Data setup (before $\overline{EWE}$) | $t_{DSW}$ | $C_L$ = 50pF + 1TTL | $\frac{t_{CYC}}{2} - 50$ | | | ns |
| Data hold (after $\overline{EWE}$) | $t_{DHW}$ | $C_L$ = 50pF + 1TTL | 5 | | | ns |
| Data output active delay | $t_{DAC}$ | $C_L$ = 50pF + 1TTL | | 52 | 70 | ns |

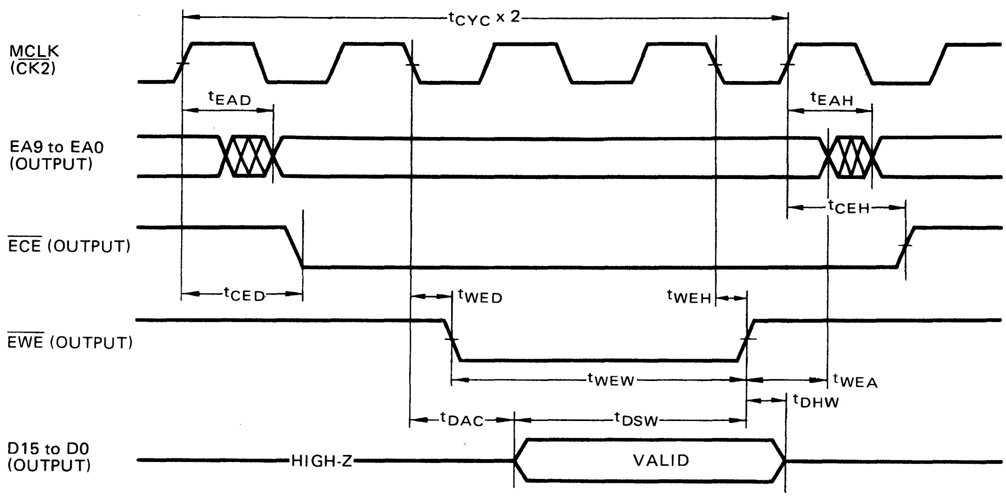
Fig. 9 – EXPANSION RAM INTERFACE TIMING DIAGRAM (ESS = 0, Write Cycle)



## EXPANSION RAM INTERFACE TIMING (ESS = 1, Write Cycle)

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Address output delay | $t_{EAD}$ | $C_L$ = 50pF | | 50 | 60 | ns |
| Address output hold | $t_{EAH}$ | $C_L$ = 50pF | 10 | 13 | | ns |
| Address output hold (after $\overline{EWE}$) | $t_{WEA}$ | $C_L$ = 50pF | | 25 | | ns |
| Chip enable output delay | $t_{CED}$ | $C_L$ = 50pF | | 57 | 70 | ns |
| Chip enable output hold | $t_{CEH}$ | $C_L$ = 50pF | 17 | 19 | | ns |
| Write enable output delay | $t_{WED}$ | $C_L$ = 50pF | | | 50 | ns |
| Write enable output hold | $t_{WEH}$ | $C_L$ = 50pF | 10 | | 35 | ns |
| Write enable pulse width | $t_{WEW}$ | $C_L$ = 50pF | $t_{CYC} - 40$ | | | ns |
| Data output active delay | $t_{DAC}$ | $C_L$ = 50pF + 1TTL | | 57 | 75 | ns |
| Data setup (during $\overline{EWE}$) | $t_{DSW}$ | $C_L$ = 50pF + 1TTL | $t_{CYC} - 65$ | | | ns |
| Data hold (after $\overline{EWE}$) | $t_{DHW}$ | $C_L$ = 50pF + 1TTL | 5 | | | ns |

**Fig. 10 – EXPANSION RAM INTERFACE TIMING DIAGRAM (ESS = 1, Write Cycle)**

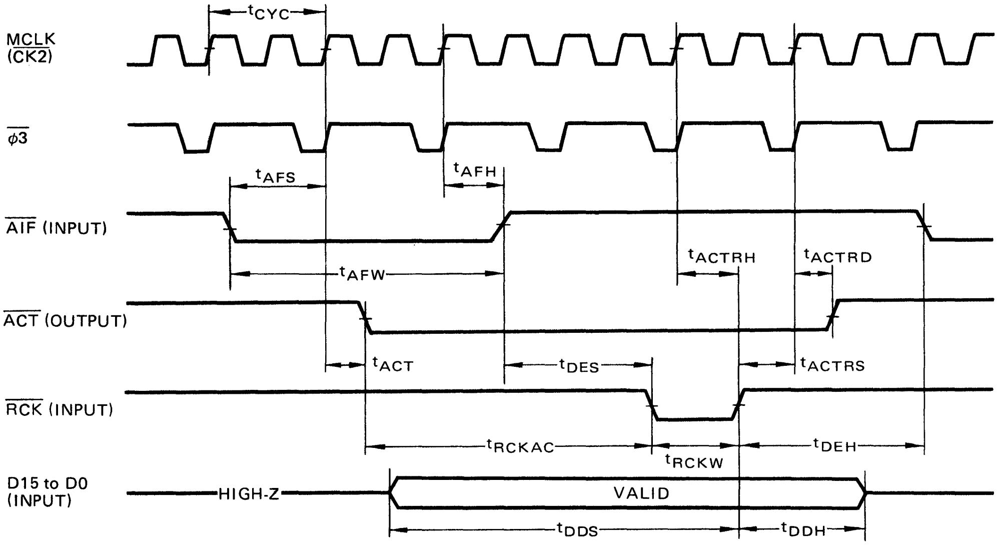


## P MODE AND D MODE INPUT TIMING

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{AIF}}$ setup | $t_{AFS}$ | | 30 | | | ns |
| $\overline{\text{AIF}}$ hold | $t_{AFH}$ | | 20 | | | ns |
| $\overline{\text{AIF}}$ pulse width *1 | $t_{AFW}$ | | $t_{CYC}$ +50 | | | ns |
| $\overline{\text{ACT}}$ fall delay | $t_{ACT}$ | $C_L$ = 50pF + 1TTL | | | 70 | ns |
| $\overline{\text{ACT}}$ reset delay | $t_{ACTRD}$ | $C_L$ = 50pF + 1TTL | | | 70 | ns |
| $\overline{\text{RCK}}$ input enable | $t_{RCKAC}$ | | 0 | | | ns |
| $\overline{\text{RCK}}$ pulse width | $t_{RCKW}$ | | 40 | | | ns |
| $\overline{\text{RCK}}$ enable setup | $t_{DES}$ | | 35 | | | ns |
| $\overline{\text{RCK}}$ enable hold | $t_{DEH}$ | | 25 | | | ns |
| Data setup | $t_{DDS}$ | | 25 | | | ns |
| Data hold | $t_{DDH}$ | | 25 | | | ns |
| $\overline{\text{ACT}}$ reset setup | $t_{ACTRS}$ | | 60 | | | ns |
| $\overline{\text{ACT}}$ reset hold | $t_{ACTRH}$ | | 10 | | | ns |

**Note:** *1 $t_{AFW}$ (Min) =2 x $t_{CYC}$ + 50 when ERAM is used with ESS = 1.

**Fig. 11 – P MODE AND D MODE INPUT TIMING DIAGRAM**

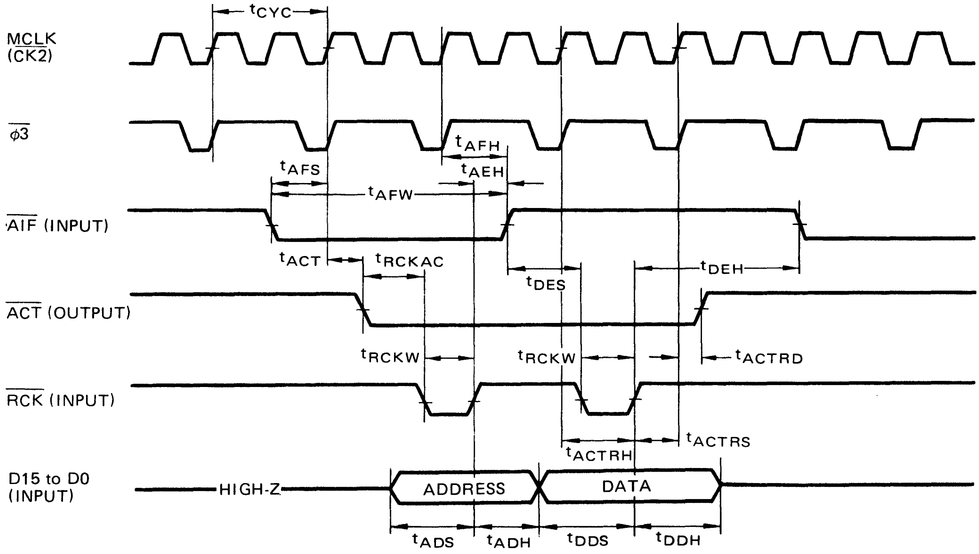


## A MODE INPUT TIMING

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{AIF}}$ setup | $t_{AFS}$ | | 30 | | | ns |
| $\overline{\text{AIF}}$ hold | $t_{AFH}$ | | 20 | | | ns |
| $\overline{\text{AIF}}$ pulse width *1 | $t_{AFW}$ | | $t_{CYC}$ + 50 | | | ns |
| $\overline{\text{ACT}}$ fall delay | $t_{ACT}$ | $C_L$ = 50pF + 1TTL | | | 70 | ns |
| $\overline{\text{ACT}}$ reset delay | $t_{ACTRD}$ | $C_L$ = 50pF + 1TTL | | | 70 | ns |
| $\overline{\text{RCK}}$ input enable | $t_{RCKAC}$ | | 0 | | | ns |
| $\overline{\text{RCK}}$ pulse width | $t_{RCKW}$ | | 40 | | | ns |
| $\overline{\text{RCK}}$ enable hold | $t_{AEH}$ | | 25 | | | ns |
| $\overline{\text{RCK}}$ enable setup | $t_{DES}$ | | 35 | | | ns |
| $\overline{\text{RCK}}$ enable hold | $t_{DEH}$ | | 25 | | | ns |
| Address setup | $t_{ADS}$ | | 25 | | | ns |
| Address hold | $t_{ADH}$ | | 25 | | | ns |
| Data setup | $t_{DDS}$ | | 25 | | | ns |
| Data hold | $t_{DDH}$ | | 25 | | | ns |
| $\overline{\text{ACT}}$ reset setup | $t_{ACTRS}$ | | 60 | | | ns |
| $\overline{\text{ACT}}$ reset hold | $t_{ACTRH}$ | | 10 | | | ns |

**Note:** *1 $t_{AFW}$ (Min) = 2 x $t_{CYC}$ + 50 when ERAM is used with ESS = 1.

**Fig. 12 – A MODE INPUT TIMING DIAGRAM**

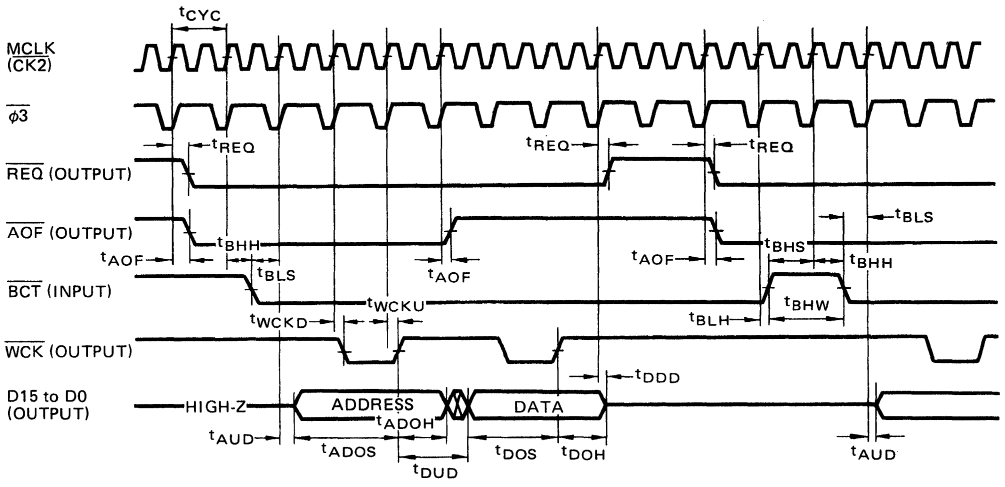


### I MODE OUTPUT TIMING

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{REQ}}$ output delay | $t_{REQ}$ | $C_L$ = 50pF + 1TTL | | | 75 | ns |
| $\overline{\text{AOF}}$ output delay | $t_{AOF}$ | $C_L$ = 50pF + 1TTL | | | 65 | ns |
| $\overline{\text{BCT}}$ level 0 setup | $t_{BLS}$ | | 40 | | | ns |
| $\overline{\text{BCT}}$ level 0 hold | $t_{BLH}$ | | 15 | | | ns |
| $\overline{\text{BCT}}$ level 1 setup | $t_{BHS}$ | | 40 | | | ns |
| $\overline{\text{BCT}}$ level 1 hold | $t_{BHH}$ | | 15 | | | ns |
| $\overline{\text{BCT}}$ level 1 pulse width *1 | $t_{BHW}$ | | $t_{CYC}$ + 55 | | | ns |
| $\overline{\text{WCK}}$ fall delay | $t_{WCKD}$ | $C_L$ = 50pF + 1TTL | | | 65 | ns |
| $\overline{\text{WCK}}$ rise delay | $t_{WCKU}$ | $C_L$ = 50pF + 1TTL | | | 65 | ns |
| Address output delay | $t_{AUD}$ | $C_L$ = 50pF + 1TTL | | | 85 | ns |
| Data output delay | $t_{DUD}$ | $C_L$ = 50pF + 1TTL | | | 80 | ns |
| Data output disable | $t_{DDD}$ | $C_L$ = 50pF + 1TTL | | | 70 | ns |
| Address setup | $t_{ADOS}$ | $C_L$ = 50pF + 1TTL | 170 | | | ns |
| Address hold | $t_{ADOH}$ | $C_L$ = 50pF + 1TTL | 65 | | | ns |
| Data setup | $t_{DOS}$ | $C_L$ = 50pF + 1TTL | 170 | | | ns |
| Data hold | $t_{DOH}$ | $C_L$ = 50pF + 1TTL | 65 | | | ns |

**Note:** *1 $t_{BHW}$ (Min) = 2 x $t_{CYC}$ + 55 when ERAM is used with ESS = 1.

**Fig. 13 – I MODE OUTPUT TIMING DIAGRAM**

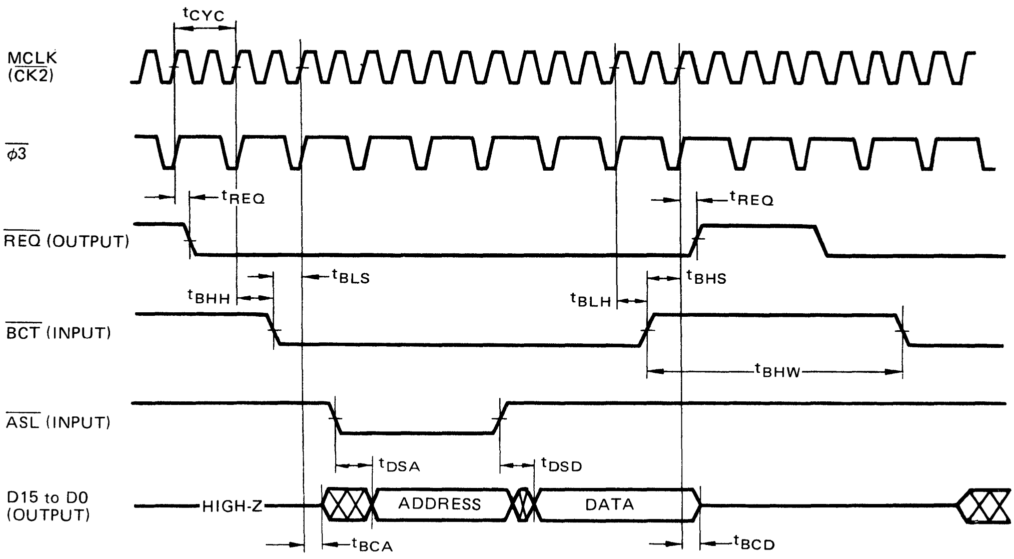


## E MODE OUTPUT TIMING

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{REQ}}$ output delay | $t_{REQ}$ | $C_L$ = 50pF + 1TTL | | | 75 | ns |
| $\overline{\text{BCT}}$ level 0 setup | $t_{BLS}$ | | 40 | | | ns |
| $\overline{\text{BCT}}$ level 0 hold | $t_{BLH}$ | | 15 | | | ns |
| $\overline{\text{BCT}}$ level 1 setup | $t_{BHS}$ | | 40 | | | ns |
| $\overline{\text{BCT}}$ level 1 hold | $t_{BHH}$ | | 15 | | | ns |
| $\overline{\text{BCT}}$ level 1 pulse width *1 | $t_{BHW}$ | | $t_{CYC}$ +55 | | | ns |
| Output active delay | $t_{BCA}$ | $C_L$ = 50pF + 1TTL | | | 85 | ns |
| Address output from fall of $\overline{\text{ASL}}$ | $t_{DSA}$ | $C_L$ = 50pF + 1TTL | | | 85 | ns |
| Data output from rise of $\overline{\text{ASL}}$ | $t_{DSD}$ | $C_L$ = 50pF + 1TTL | | | 85 | ns |
| Output inactive | $t_{BCD}$ | $C_L$ = 50pF + 1TTL | | | 70 | ns |

**Note:** *1 $t_{BHW}$ (Min) = 2 x $t_{CYC}$ + 55 when ERAM is used with ESS = 1.

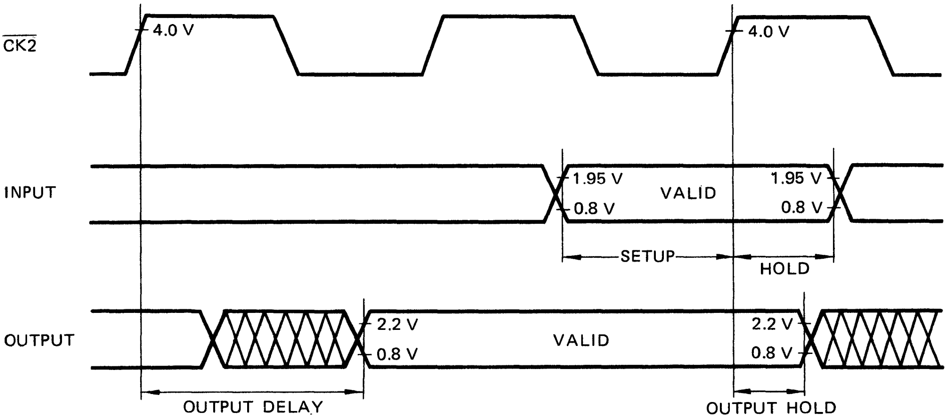
### Fig. 14 – E MODE OUTPUT TIMING DIAGRAM



## AC CHARACTERISTICS MEASUREMENT CONDITIONS

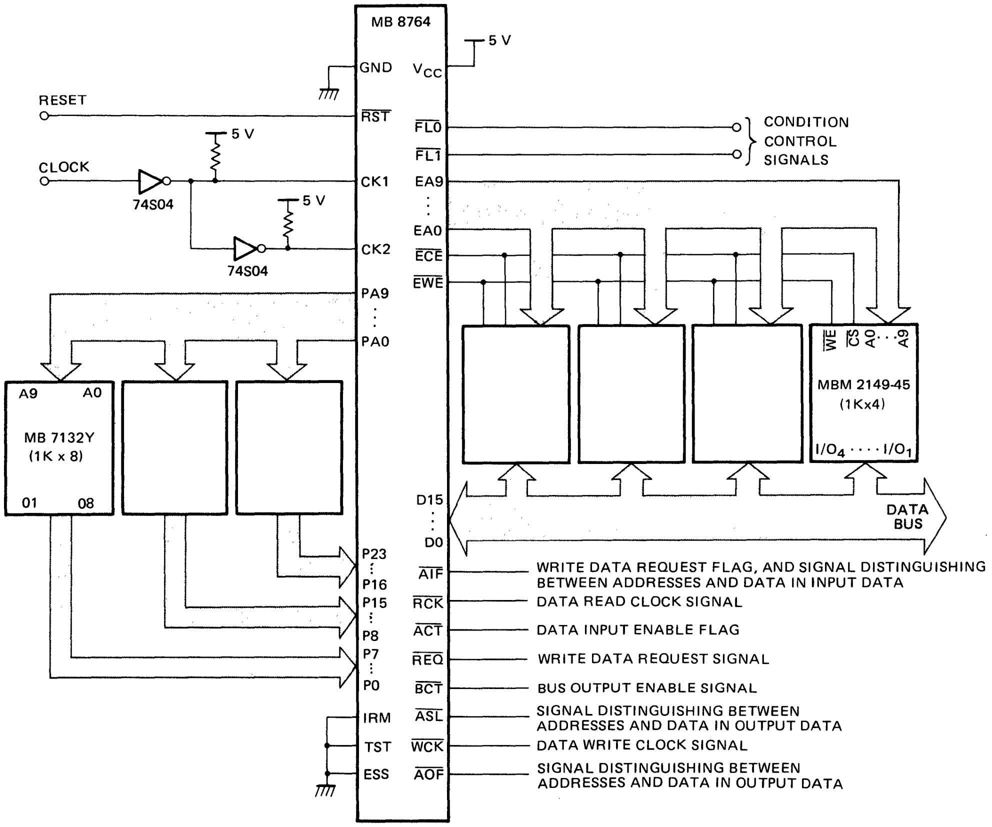
| Parameter | Symbol | Condition |
|---|---|---|
| Power supply voltage | $V_{DD}$ | 5 V ± 10% |
| Ambient temperature | $T_A$ | 0 to 85°C |

Fig. 15 – AC CHARACTERISTICS MEASUREMENT WAVEFORMS



# APPLICATION INFORMATION

Fig. 16 – EXAMPLE OF MB 8764 APPLICATION CIRCUIT
(with 1K-word External ROM and 1K-word External RAM)



# INSTRUCTION SET

## ARITHMETIC AND LOGIC INSTRUCTIONS

| Mnemonic | Processing performed | Mnemonic | Processing performed |
|---|---|---|---|
| NOP | No operation | ABS | \|D\| → D |
| ADD | A + B → D | NEG | −D → D |
| MLT | A x B → D | SRA | Shift D right arithmetic → D |
| SUB | B − A → D | SLA | Shift D left arithmetic → D |
| MSM | D + A x B → D | AND | D ∩ A → D |
| MRD | D − A x B → D | ORA | D ∪ A → D |
| SUM | D + A → D | DIV | D ÷ A → D |
| RED | D − A → D | COM | $\overline{D}$ → D |

## TRANSFER INSTRUCTIONS

| Mnemonic | Processing performed |
|---|---|
| LTB: (Arithmetic/logic instruction) $a, $b | ROMT → A, BRAM/ERAM → B |
| LAB: (Arithmetic/logic instruction) $a, $b | ARAM → A, BRAM/ERAM → B |
| MAB: (Arithmetic/logic instruction) $a, $b | ARAM → BRAM/ERAM |
| MBA: (Arithmetic/logic instruction) $a, $b | BRAM/ERAM → ARAM |
| MOV: (Arithmetic/logic instruction) $a, Reg [:Reg . . ] | IRAM/ERAM → Register |
| MOV: (Arithmetic/logic instruction) #$d, Reg [:Reg . . ] | Immediate data (d) → Register |
| MOV: (Arithmetic/logic instruction) Reg, Reg [:Reg . . ] | Register → Register |
| LDI: (Arithmetic/logic instruction) #$d | d → A |
| LIB: (Arithmetic/logic instruction) #$d | d → A, BRAM → B |

## JUMP INSTRUCTIONS

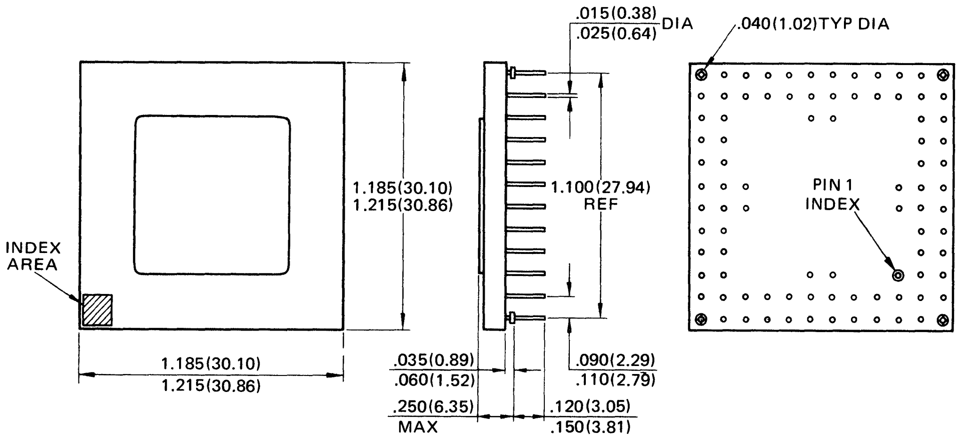
| Mnemonic | Processing performed |
|---|---|
| JMP: (Arithmetic/logic instruction) #$d | Unconditional jump (d → PC) |
| JOC: (Arithmetic/logic instruction) #$d, flag | Conditional jump (d→PC) |
| JOC: (Arithmetic/logic instruction) $a, flag | Conditional jump (IRAM/ERAM → PC) |
| JSR: (Arithmetic/logic instruction) #$d | Jump to subroutine (PC → PCS, d → PC) |
| RTS: (Arithmetic/logic instruction) | Return from subroutine |

## MISCELLANEOUS INSTRUCTIONS

| Mnemonic | Processing performed |
|---|---|
| CLR: [Reg [:Reg . . . ] . . . ] | Clear register |
| SET: [Reg [:Reg . . . ] . . . ] | Set register |
| MXY: (Arithmetic/logic instruction) #$\$d_1$, #$Sd_2$ | X + $d_1$ → X, Y + $d_2$ → Y |
| LIY: (Arithmetic/logic instruction) #$d | d → A,BRAM → B, Y + 1 → Y |
| AVP: (Arithmetic/logic instruction) #$d | VP + d → VP |
| LVP: (Arithmetic/logic instruction) #$d | d → VP |
| ADY: (Arithmetic/logic instruction) | Y+YS → Y |
| GXY: (Arithmetic/logic instruction) | XS → X, YS → Y |
| SXY: (Arithmetic/logic instruction) #$d | d → C1, X → XS, Y → YS, O → X, O → Y |
| NOP: (Arithmetic/logic instruction) | No operation |

# PACKAGE DIMENSIONS

88-LEAD CERAMIC (METAL SEAL) REPEATED QUAD IN-LINE PACKAGE
(CASE No.: RIT-88C-A01)



Dimensions in
inches (millimeters)

Circuit diagrams utilizing Fujitsu products are included as a means of illustrating typical semiconductor applications; consequently, complete information sufficient for construction purposes is not necessarily given. The information has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, such information does not convey to the purchaser of the semiconductor devices described herein any license under the patent rights of Fujitsu Limited or others. Fujitsu Limited reserves the right to change device specifications.

**FUJITSU LIMITED**

*Communications and Electronics*

6-1, Marunouchi 1-chome, Chiyoda-ku, Tokyo 100, Japan
Phone: National (03) 216-3211 International (Int'l Prefix) 81-3-216-3211 Telex: J22833 Cable: "FUJITSULIMITED TOKYO"

For further information please contact:

**FUJITSU LIMITED**

**Semiconductors Marketing:** Furukawa Sogo Bldg., 6-1, Marunouchi 2-chome, Chiyoda-ku, Tokyo 100, Japan
Phone: National (03) 216-3211 International (Int'l Prefix) 81-3-216-3211 Telex: 2224361 FTTOR J

**FUJITSU MICROELECTRONICS, INC.:**
3320 Scott Blvd., Santa Clara, CA95054-3197, U.S.A.
Phone: 408-727-1700 Telex: 910-338-0190 FUJITSU SNTA

**FUJITSU MIKROELEKTRONIK GmbH.:**
Arabella Center 9. OG./A
Lyoner Straße 44-48 D-6000 Frankfurt 71, F.R. Germany
Phone: 069/66 320 Telex: 0411 963

OV0002–868F1 Printed in USA