# DSP56000 56001 000 01

## An Introduction

MOTOROLA
Semiconductor Products Sector

MTTAS/CN
REV 1.1

# DSP56000/1
# AUDIO CASSETTE COURSE

PREPARED BY
MOTOROLA SEMICONDUCTOR PRODUCTS SECTOR
WORLDWIDE TRAINING & TECHNICAL OPERATIONS
PHOENIX, AZ

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© Copyright Motorola Inc. 1988, All Rights Reserved
PRINTED MARCH, 1989

# WE'D LIKE TO SAY THANKS

## TO OUR EMPLOYEES, CUSTOMERS AND SUPPLIERS

Motorola has won the first annual Malcolm Baldridge National Quality Award. Our recognition comes as the result of a coordinated effort from all our employees and their families, working with their customers and suppliers and all the members of our communities around the world.

Established by the Congress and the U.S. Department of commerce to recognize quality improvement and quality achievement, the Award acknowledges that we, at Motorola, are on track in our quest to achieve virtual perfection in everything we do by 1992. Thanks for your support in making this possible.

Winner 1988

Malcolm Baldrige National Quality Award

Presented by President Ronald Reagan
to Motorola
at The White House November 14, 1988

MOTOROLA

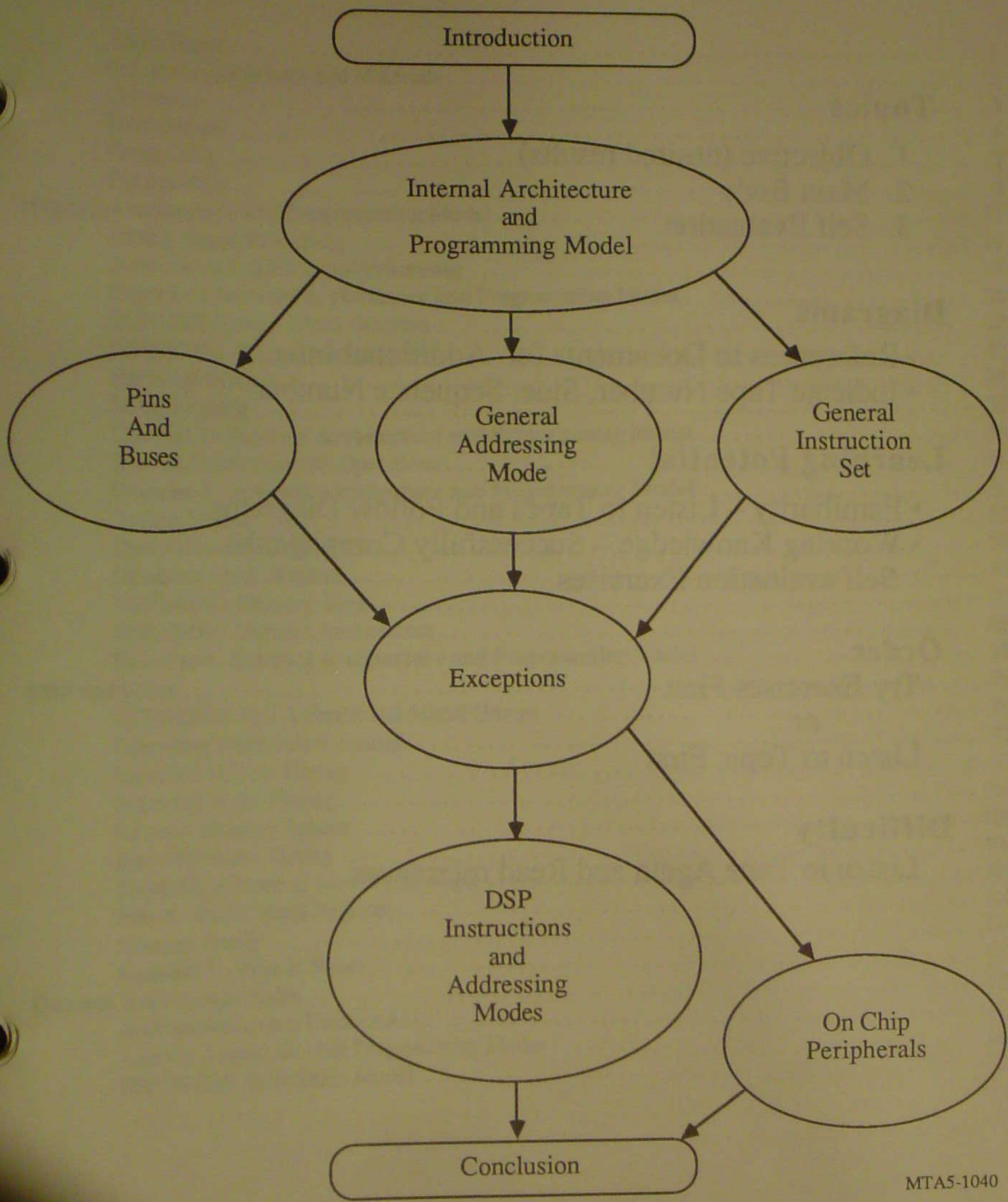
Internal Architecture and Programming Model 1
Pins and Buses 2
General Addressing Modes 3
General Instruction Set 4
Exceptions 5
DSP Instructions and Addressing Modes 6
On Chip Peripherals 7
Glossary 8

# PREREQUISITES

To derive maximum benefit, the student should know:

- ▣ Memory Concepts
- ▣ Binary Numbers
- ▣ Binary Arithmetic
- ▣ Hexadecimal Notation
- ▣ Two's Complement Arithmetic
- ▣ Logic Operations
- ▣ Basic Digital Signal Processing Theory

# COURSE MAP


Introduction
Internal Architecture and Programming Model
Pins And Buses
General Addressing Mode
General Instruction Set
Exceptions
DSP Instructions and Addressing Modes
On Chip Peripherals
Conclusion
MTA5-1040

# TABLE OF CONTENTS


Audio Tapes ..... 1
Reference Documents and Materials ..... 2
Contacts ..... 3
Prerequisites ..... 4
Course Map ..... 5
Organization ..... 6
**Internal Architecture and Programming Model** ..... 11
Analog Signal Processing ..... 12
Overview of Digital Signal Processing ..... 14
**Exercise 1-Internal Architecture and Programming Model** ..... 28
DSP56000 System Block Diagram ..... 30
DSP56000/1 Programming Model ..... 32
Hardware Stack ..... 34
Status Register ..... 36
**Exercise 2 - Internal Architecture and Programming Model** ..... 38
Data ALU and Numeric Operations ..... 40
**Exercise 3 - Internal Architecture and Programming Model** ..... 50
Scaling Modes ..... 52
Data Arithmetic Logic Unit ..... 54
Operating Mode Register ..... 56
DSP56000/1 Memory Maps ..... 58
DSP56000/1 Harvard Architecture ..... 66
**Exercise 4 - Internal Architecture and Programming Model** ..... 78
**Pins and Buses** ..... 79
DSP56000/1 Port A Functional Signal Groups ..... 80
Operating Mode Select Timing ..... 92
Instruction Cycle Timing ..... 94
Read and Write Timing ..... 96
External Memory Spaces ..... 100
Bus Arbitration Timing ..... 104
Single Chip External Memory Example ..... 108
Port A - Bus Control Register ..... 110
Memory Ready ..... 116
**Exercise 5 - Pins & Buses** ..... 117
**General Addressing Modes** ..... 119
Address Arithmetic Units (AAU) ..... 120
Address Arithmetic Unit Programming Modes ..... 122
DSP56000/1 Addressing Model ..... 124

# TABLE OF CONTENTS
## (Cont.)


Basic Move Instruction ..... 126
Address Register Indirect Addressing Modes ..... 128
Special Addressing Modes ..... 142
**Exercise 6 - General Addressing Modes** ..... 152
**General Instruction Set** ..... 153
DSP56000 Instruction Set ..... 156
Parallel Data Moves ..... 158
Move Instructions ..... 178
Load Updated Address ..... 182
**Exercise 7 - General Instructions** ..... 184
Add and Subtract Arithmetic Instructions ..... 186
Shift and Add or Subtract Arithmetic Instructions ..... 188
Arithmetic Shift Instructions ..... 190
Multiply Arithmetic Instructions ..... 192
Compare Arithmetic Instructions ..... 194
Transfer Arithmetic Instructions ..... 196
Divide Iteration ..... 202
Logical Instruction ..... 204
Branching Instructions ..... 210
Bit Manipulation Instructions ..... 212
Bit Field Branching Instructions ..... 214
Miscellaneous Instructions ..... 216
**Exercise 8 - General Instructions** ..... 218
**Exceptions** ..... 219
DSP56000 Processing States ..... 221
Vector Table of Exception Sources ..... 222
Reset ..... 224
Stack Error ..... 226
Wait Exception ..... 228
Stop Exception ..... 230
Trace Mode ..... 232
Fast Interrupt ..... 234
Long Interrupt ..... 236
RTI - Return From Interrupt ..... 238
Software Interrupt ..... 240
Interrupt Priority ..... 242
**Exercise 9 - Exceptions** ..... 247
**DSP Instructions and Addressing Modes** ..... 249
Multiply Accumulate Instruction ..... 250
Modifier Registers ..... 252
Modulo M Addressing ..... 254

# TABLE OF CONTENTS
## (Cont.)


Hardware Do Loops . . . . . 258
Repeat Next Instruction . . . . . 262
**Exercise 10 - DSP Instructions and Addressing Modes** . . . . . 263
Four TAP FIR Filter . . . . . 264
**Exercise 11 - DSP Instructions and Addressing Modes** . . . . . 270
Fast Fourier Transform Butterfly . . . . . 272
Bit Reverse Addressing for N Point FFT . . . . . 276
**Exercise 12 - DSP Instructions and Addressing Modes** . . . . . 278
**On Chip Peripherals** . . . . . 279
On Chip Peripheral Features . . . . . 281
DSP56000 Input/Output Block Diagram . . . . . 282
On Chip Peripheral Memory Map . . . . . 284
Parallel Ports B and C . . . . . 286
**Exercise 13 - On Chip Peripherals** . . . . . 292
Data Transfer from Host to DSP56000 . . . . . 312
Data Transfer from DSP56000 to Host . . . . . 314
Host Exception . . . . . 316
**Exercise 14 - On Chip Peripherals** . . . . . 318
Serial Communications Interface . . . . . 320
SCI Synchronous Mode . . . . . 326
SCI - Transmit and Receive a Character . . . . . 328
Multidrop Mode . . . . . 334
Wake-Up Mode . . . . . 336
SCI-Clock and Timer . . . . . 338
**Exercise 15 - On Chip Peripherals** . . . . . 340
Synchronous Serial Interface . . . . . 342
SSI-Programming Model . . . . . 346
SSI-Clock . . . . . 348
SSI-Frame SYNC Generator . . . . . 350
SSI-Control Pin Functions . . . . . 352
SSI-Exceptions . . . . . 354
SSI-Applications Examples . . . . . 356
**Exercise 16 - On Chip Peripherals** . . . . . 364
**Conclusion** . . . . . 365
Motorola DSP56200 . . . . . 367
DSP56000 Family Development Support Products . . . . . 369
Dr. BuB . . . . . 372
Answers to Exercises . . . . . 373
List of References . . . . . 391
Glossary . . . . . 392

# INTERNAL ARCHITECTURE AND PROGRAMMING MODEL

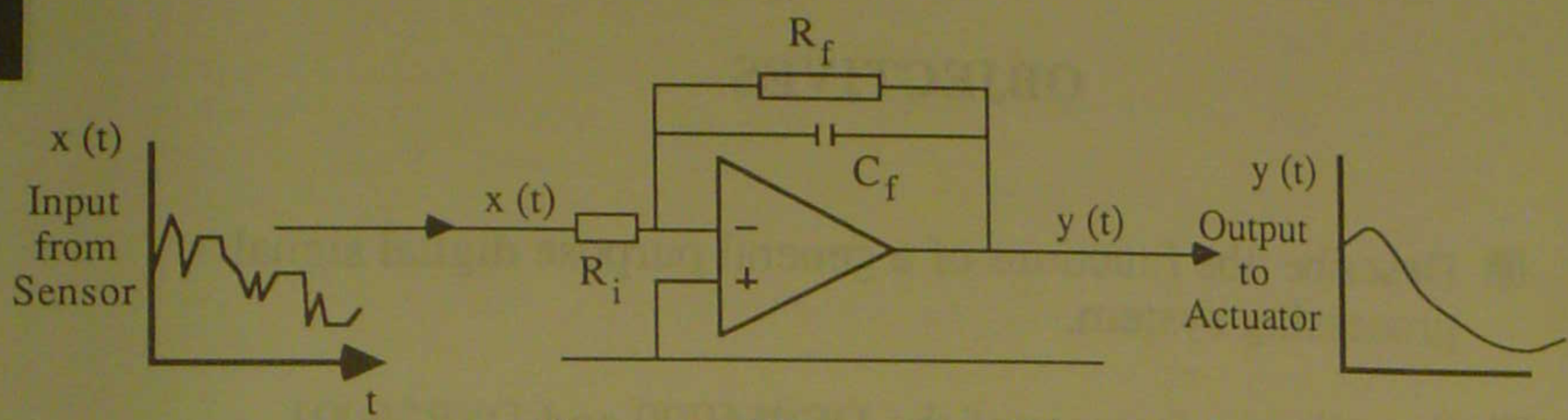
## OBJECTIVES

- Describe the functions of a general purpose digital signal processing system.
- Describe the features of the DSP56000 and DSP56001.
- Describe the numeric representation of the DSP56000/1.
- Describe the operation of the data arithmetic logic unit.
- Describe the Program, X, and Y memory spaces.
- Describe the uses of the Program, X, Y, and general parallel data buses.

# ANALOG SIGNAL PROCESSING

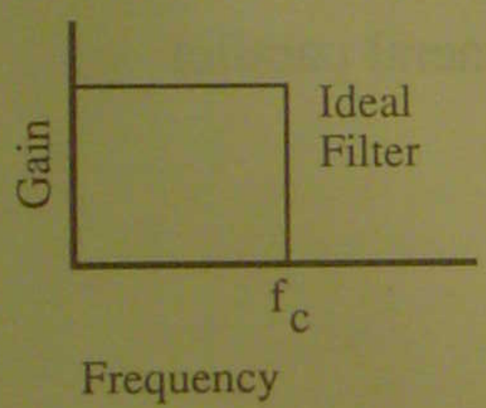
1

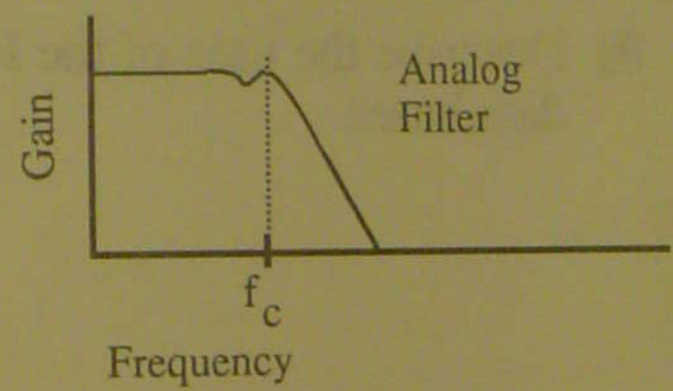
## Analog Filter




$$\frac{y(t)}{x(t)} = -\frac{R_f}{R_i}\left[\frac{1}{1 + j\omega R_f C_f}\right]$$

## Frequency Characteristics



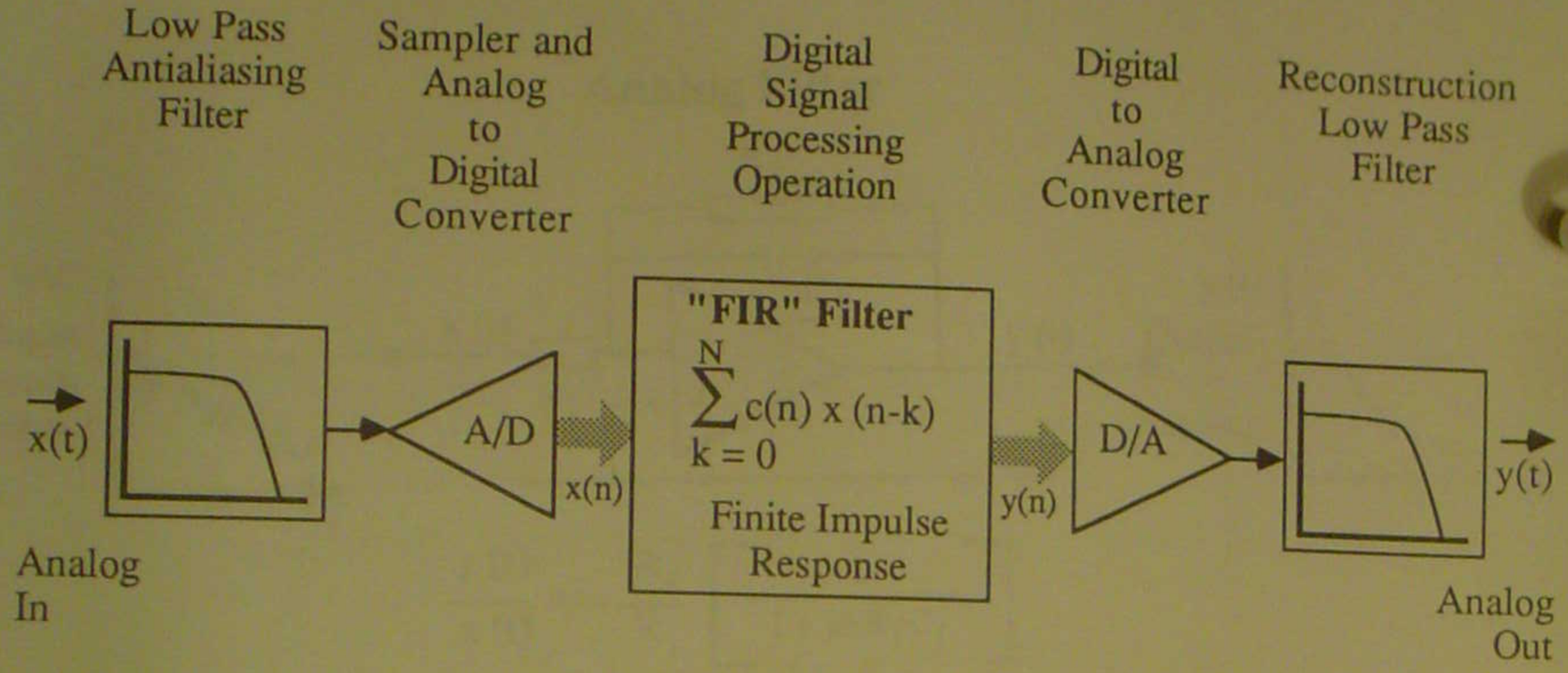

**Drawbacks:**

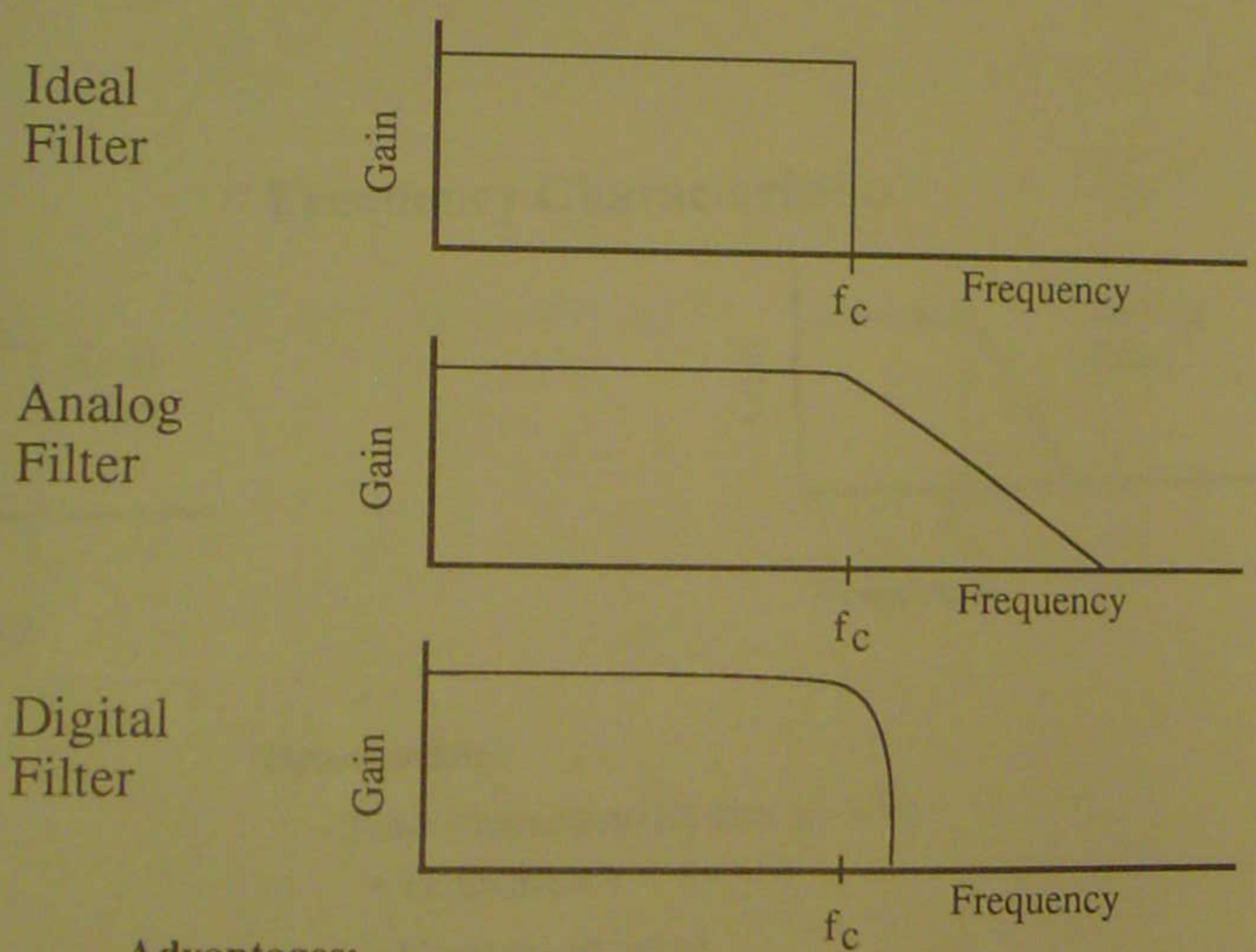
- Filter characteristics change with
  - Temperature
  - Component aging
  - Power supply variation
  - Lot to lot manufacturing
- Low noise immunity
- Requires adjustments
- Little flexibility

MTA5-2100-1

# DIGITAL SIGNAL PROCESSING

## Digital Filter



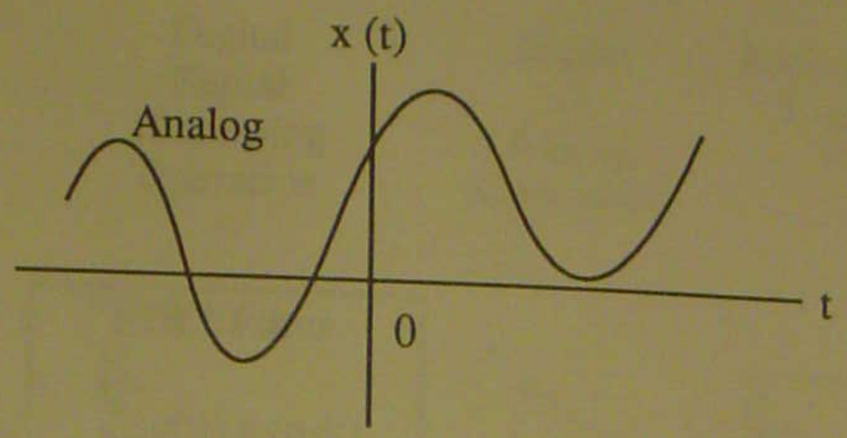

**Advantages:**

- No aging
- No drift
- No adjustments
- High noise immunity and power supply rejection
- Great flexibility
- Self test

MTA5-2110-1

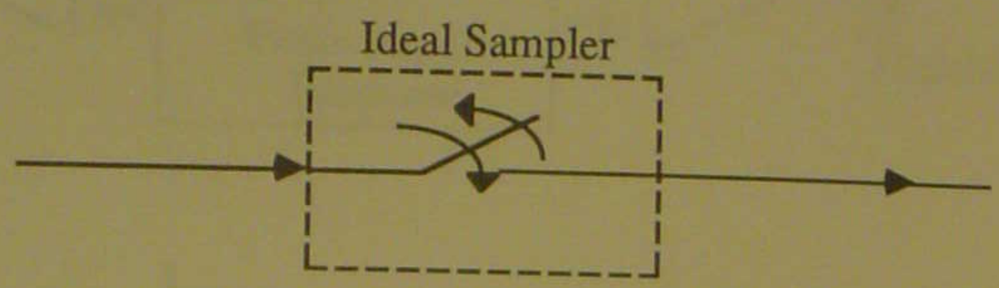
# ANALOG SIGNAL SAMPLING

Continuous Time
and
Amplitude

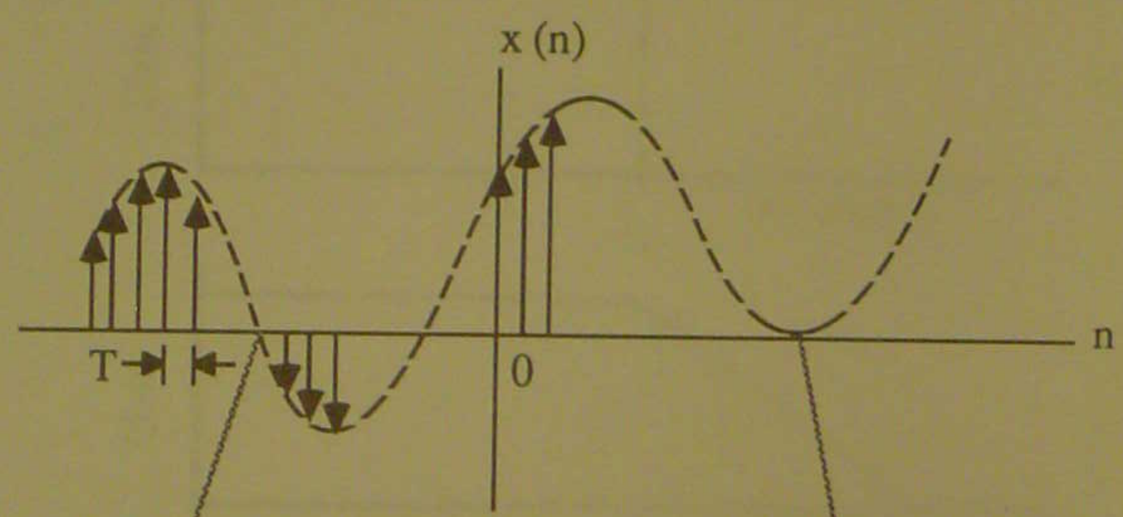


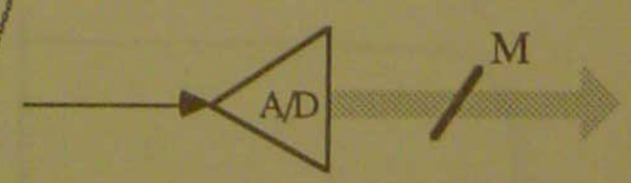

$f_s$ = Sampling Frequency

$T = \frac{1}{f_s}$ = Sampling Period

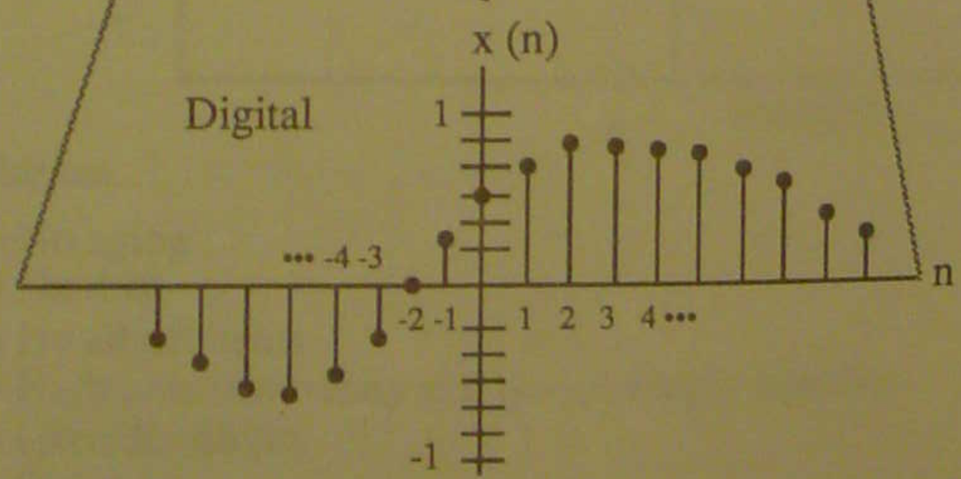



Discrete Time
Continuous
Amplitude



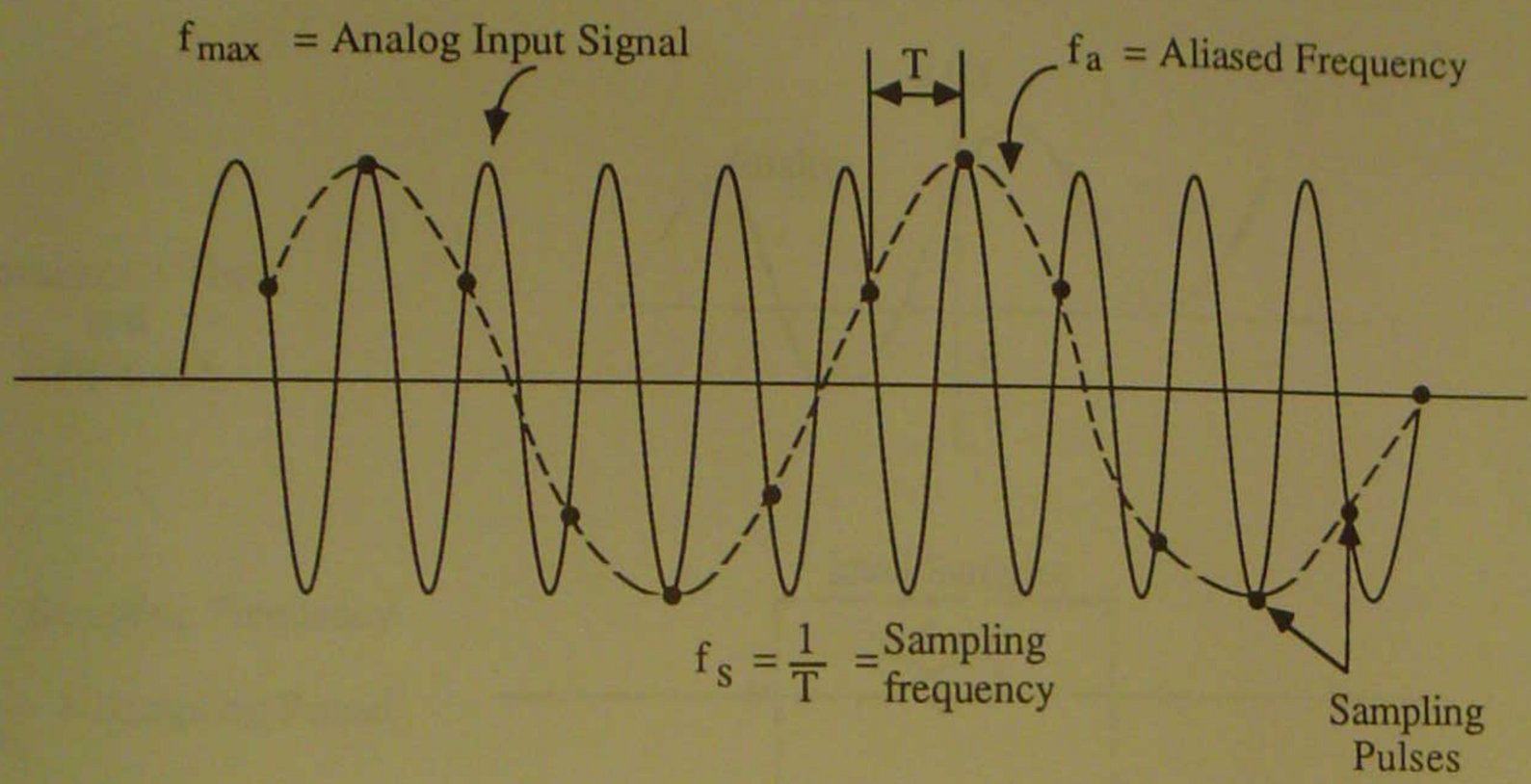




Discrete Time
and
Amplitude

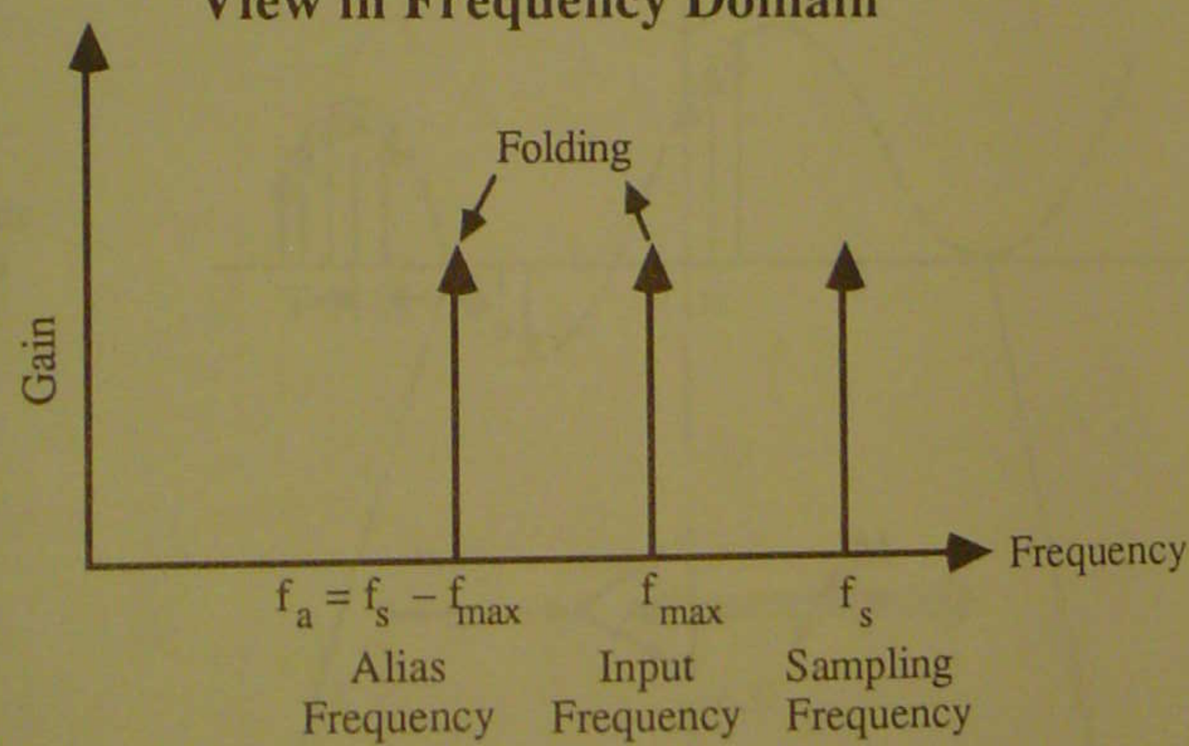



MTA5-2120

# ALIASING



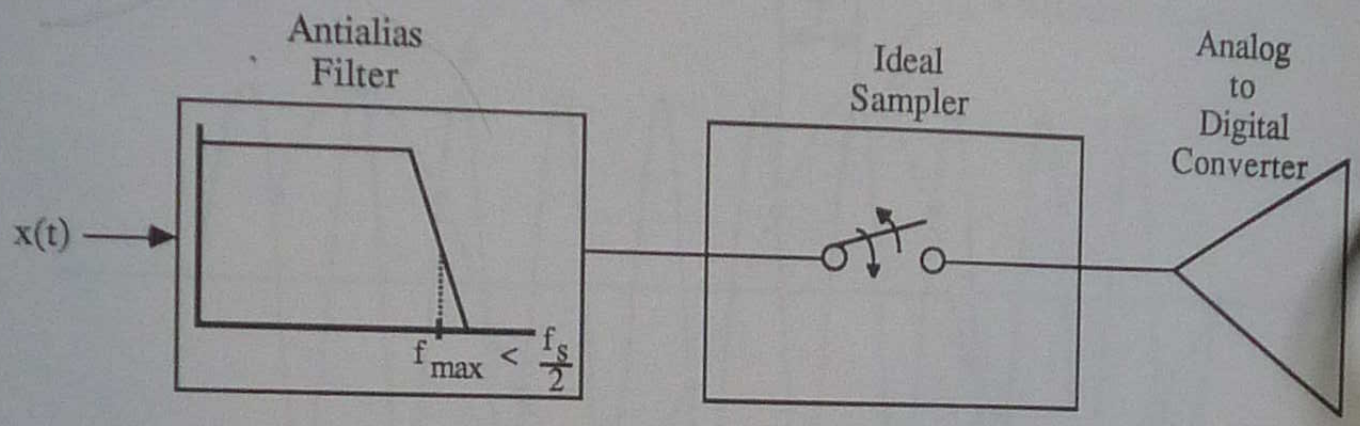

## View in Frequency Domain

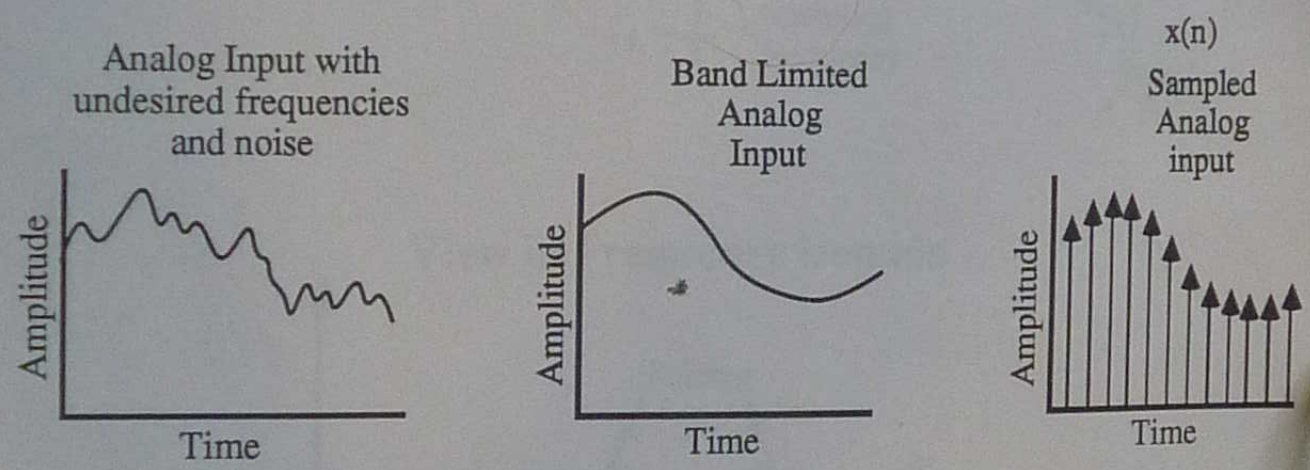


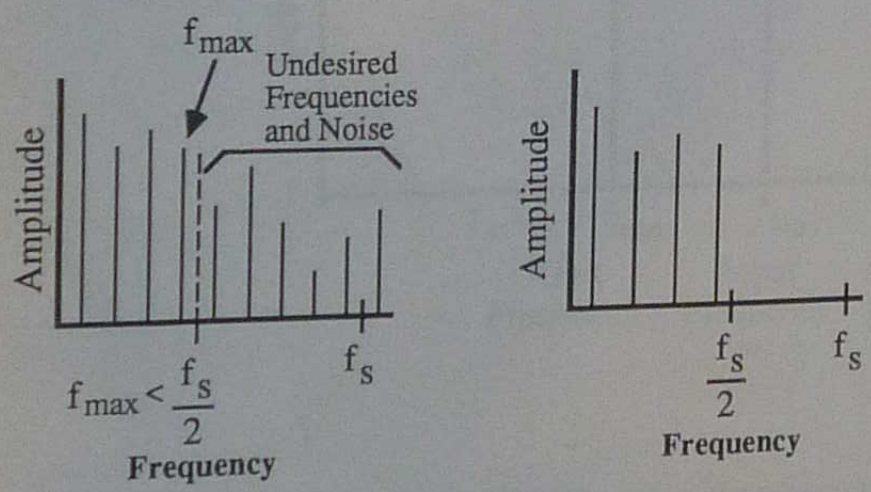

- Aliasing is caused by sampling the input frequency too slowly.
- To prevent aliasing use the Nyquist criterion (Shannon's theorem)
- $f_s \geq 2 \times f_{max}$

MTA5-2130-1

# BANDLIMITING ANTIALIASING FILTER



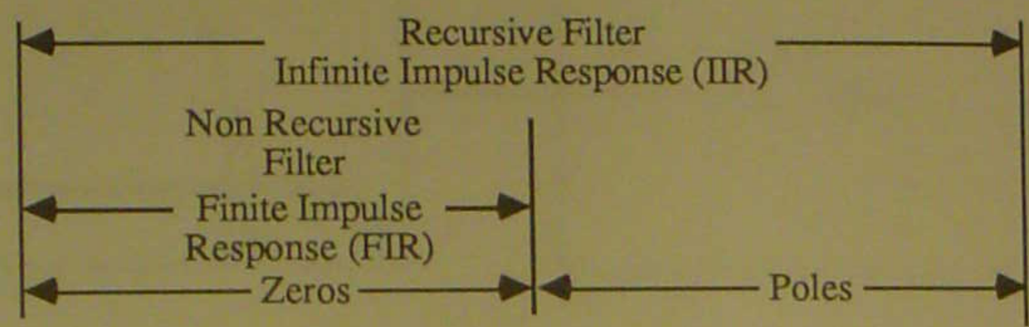

- Solution to aliasing problem:

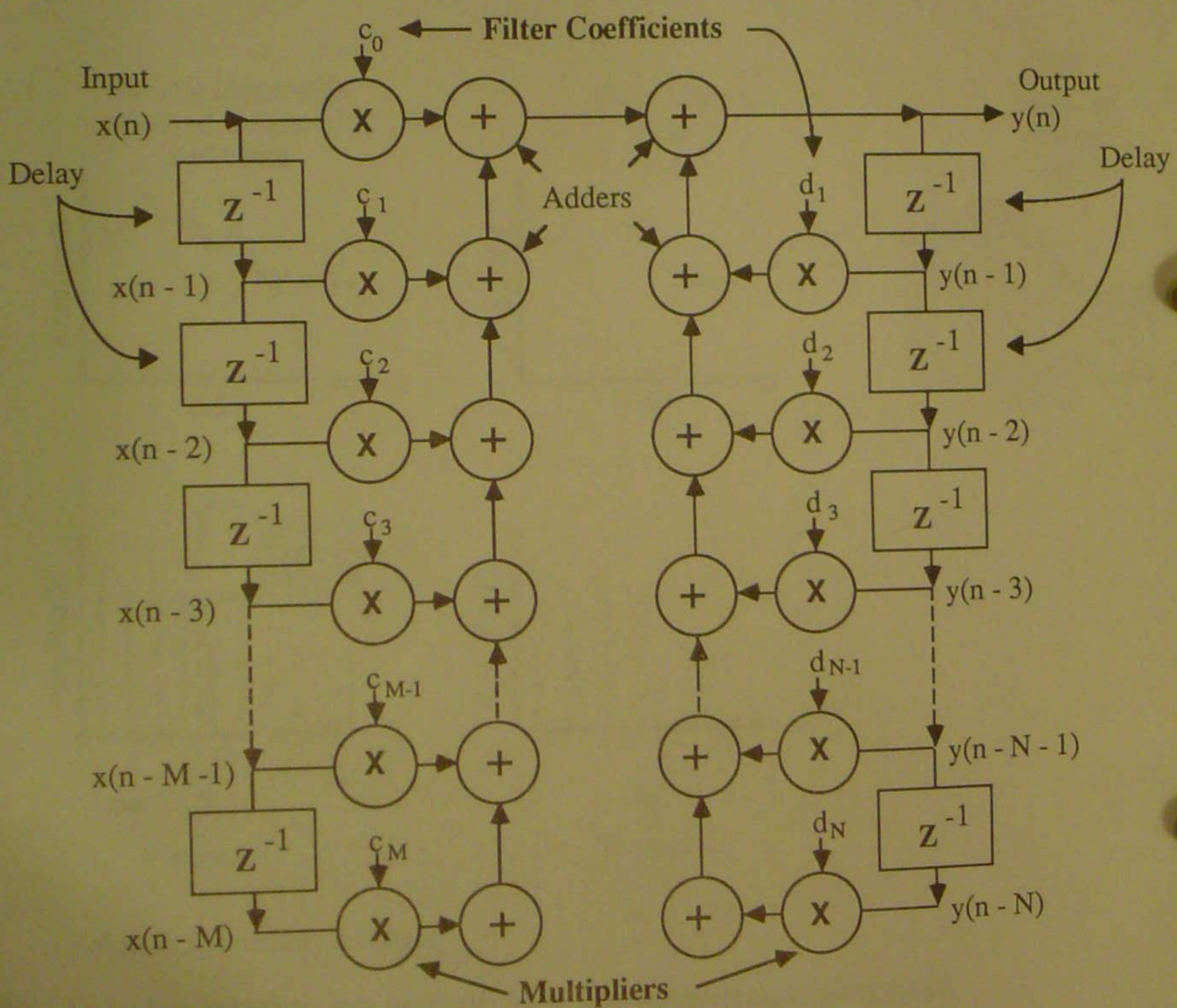
Use a low pass filter to restrict the input frequencies to at most one half the sampling frequency

MTA5-2140

# BASIC DIGITAL SIGNAL PROCESSING EQUATION

$$y(n)=\sum_{k=0}^{M} c(k)\,x(n-k)+\sum_{j=1}^{N} d(j)\,y(n-j)$$

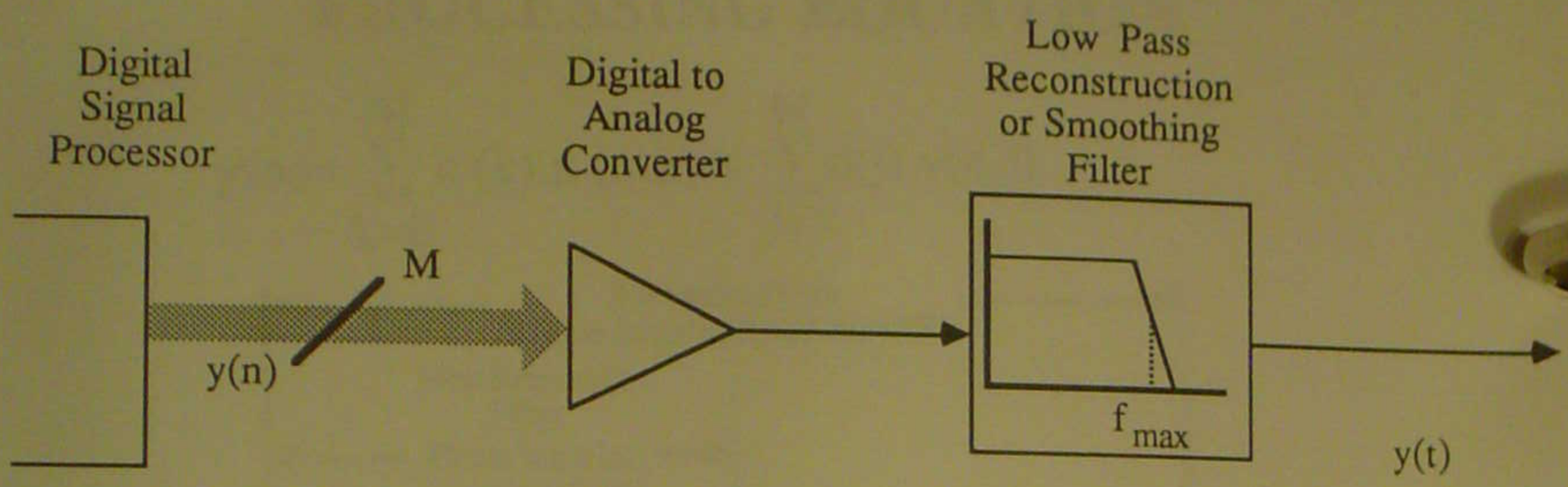


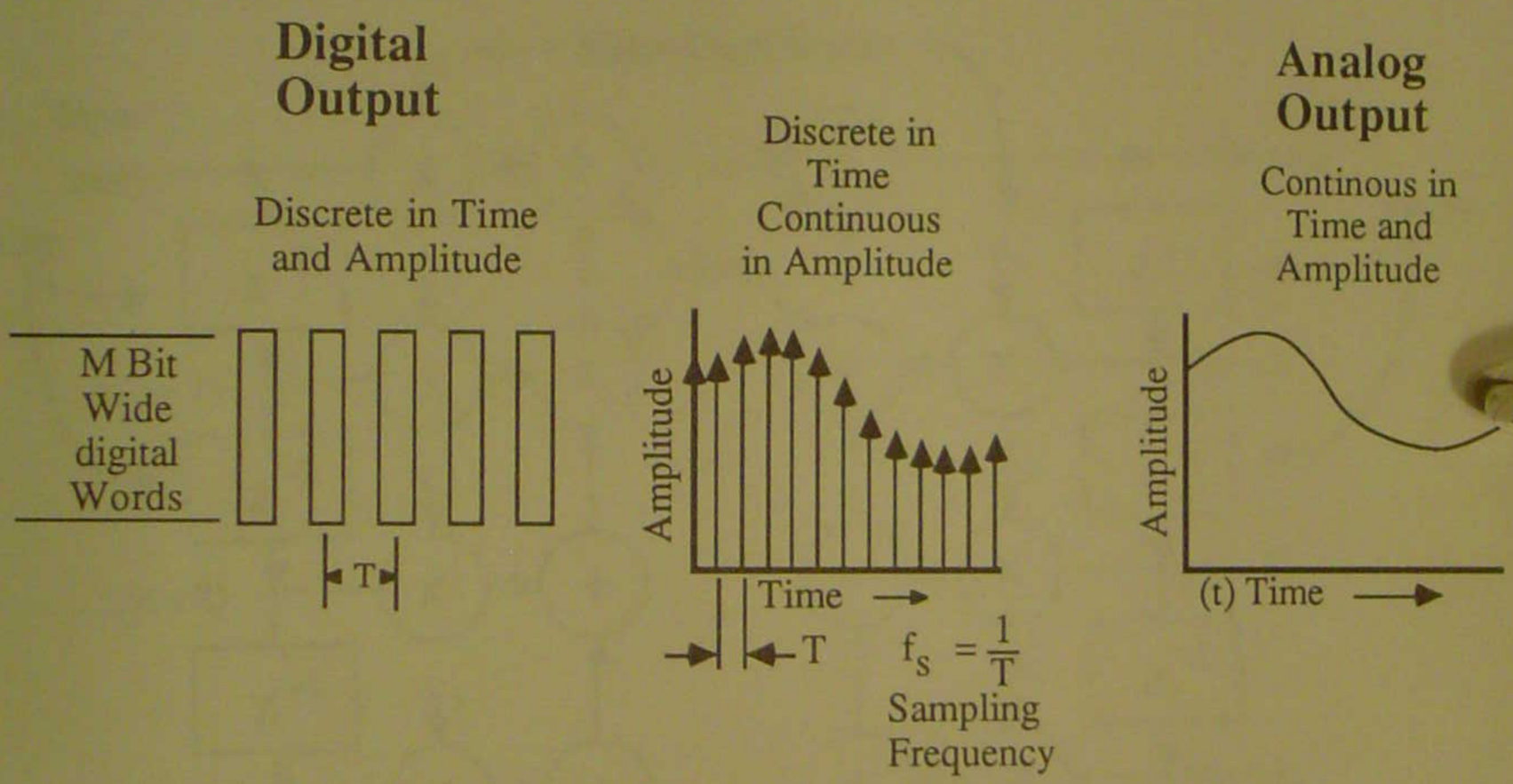




MTA5-2150

- The basic DSP operation is a multiply followed by an accumulate

# ANALOG OUTPUT RECONSTRUCTION

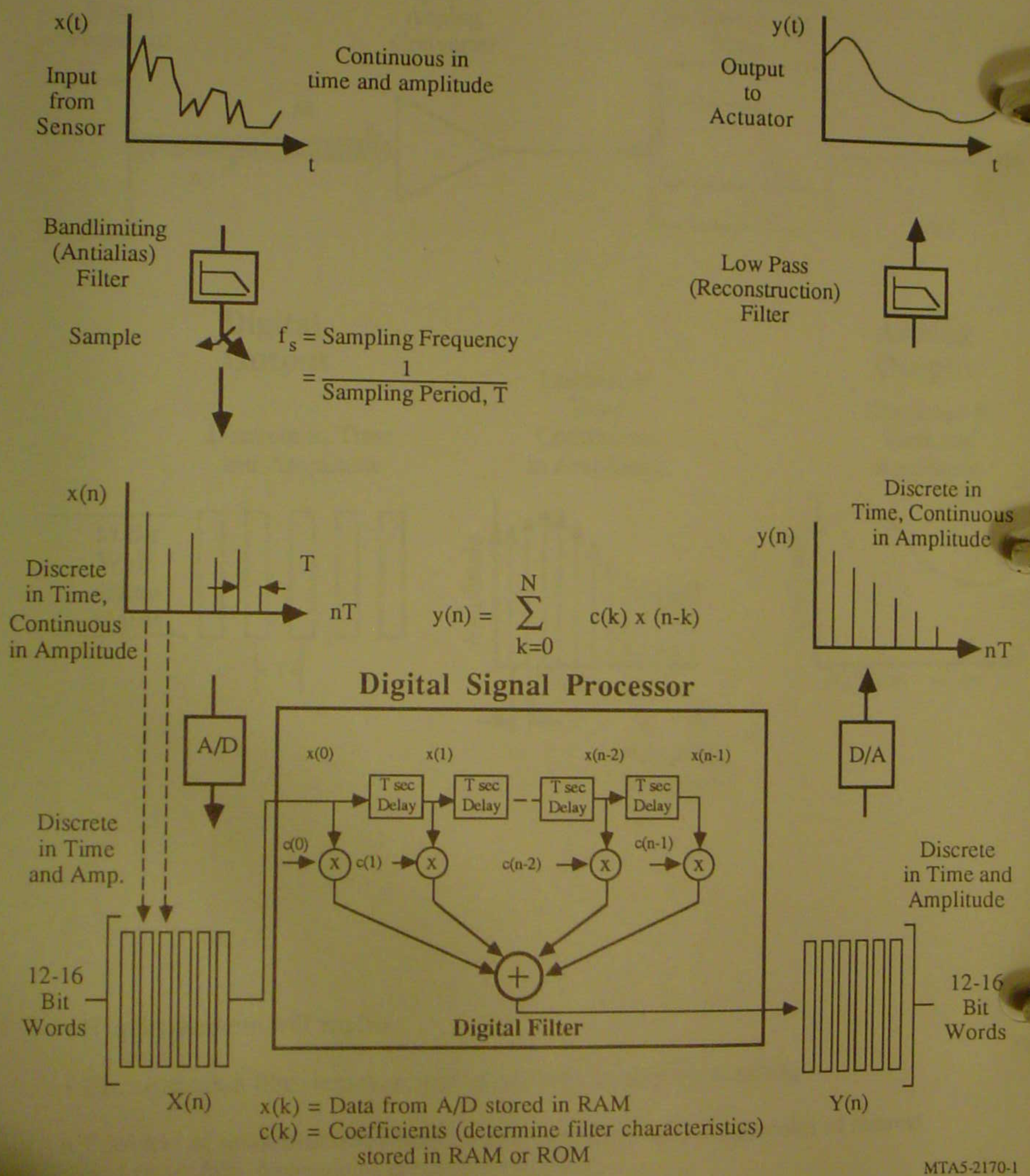



- Nyquist theorem still applies
- Reconstruction filter removes high frequencies created by sampling
- Passband of reconstruction filter is selected to allow only frequencies of interest and reject false frequencies created by sampling

MTA5-2160-1

# DIGITAL SIGNAL PROCESSING SYSTEM
# FINITE IMPULSE RESPONSE FILTER



# EXERCISE 1

## INTERNAL ARCHITECTURE AND PROGRAMMING MODEL

1. Name four drawbacks of Analog Signal Processing.

   A. _____

   B. _____

   C. _____

   D. _____

2. Name four advantages of Digital Signal Processing.

   A. _____

   B. _____

   C. _____

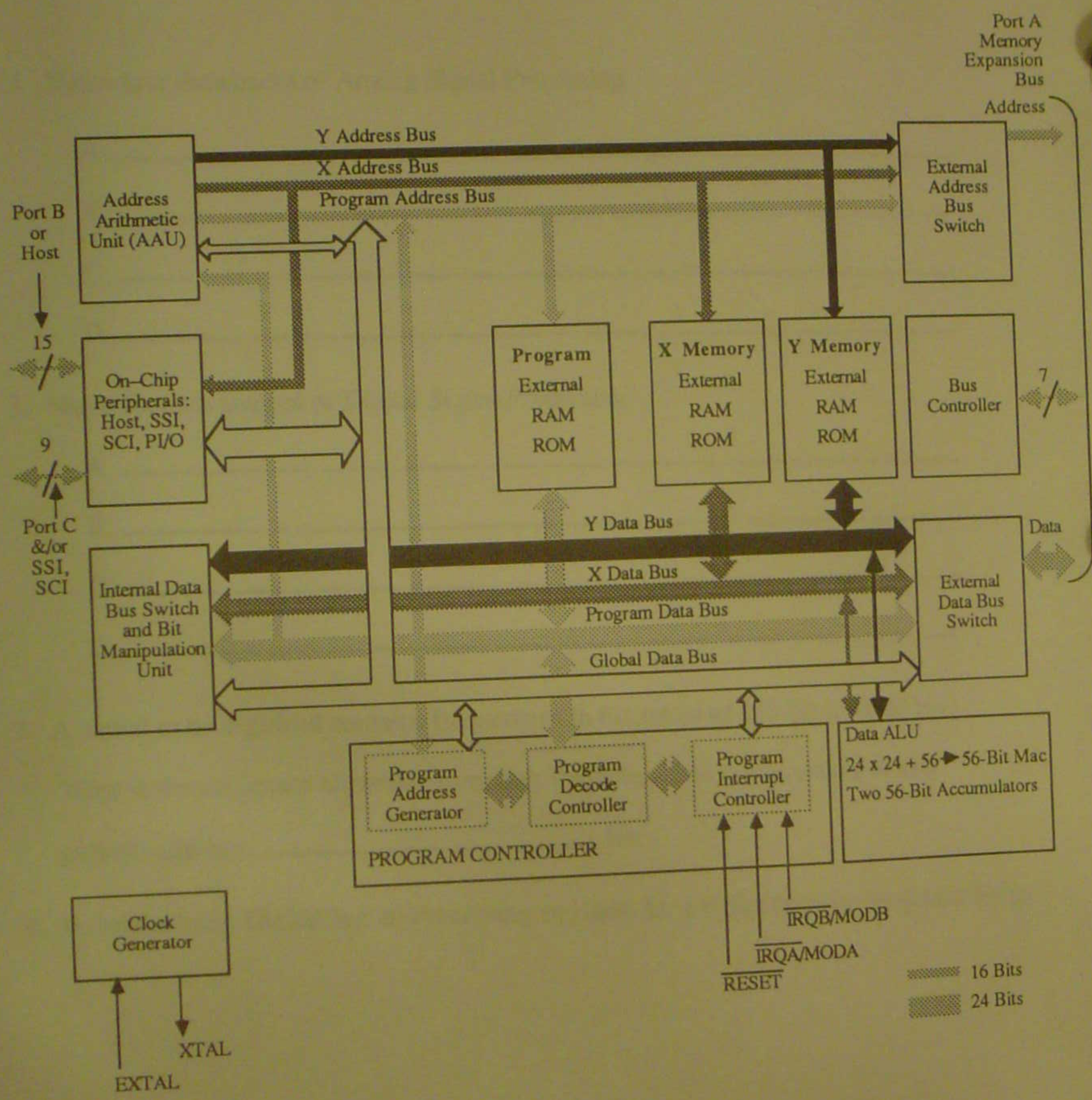
   D. _____

3. A signal to be digitized contains frequencies in the range of 200 Hz to 8000 Hz. What is the minimum allowable sampling frequency that can be used and still prevent aliasing_____ Hz.

4. Write the basic Digital Signal Processing equation for a Finite Impulse Response Filter.

MTA5-2180

# DSP56000/1 BLOCK DIAGRAM

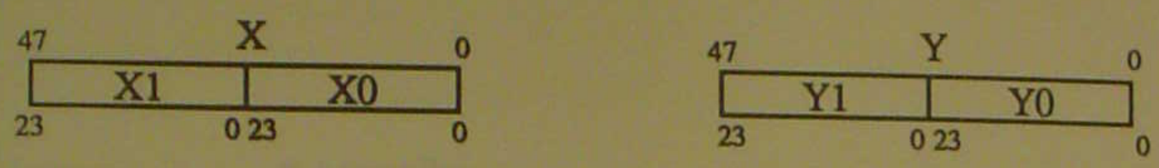



MTA5- 2200-1

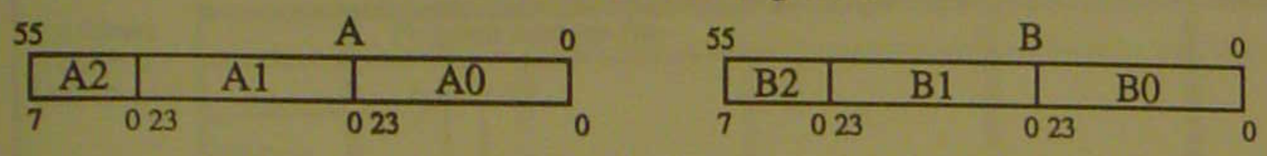
# DSP56000/1 PROGRAMMING MODEL

## Data ALU

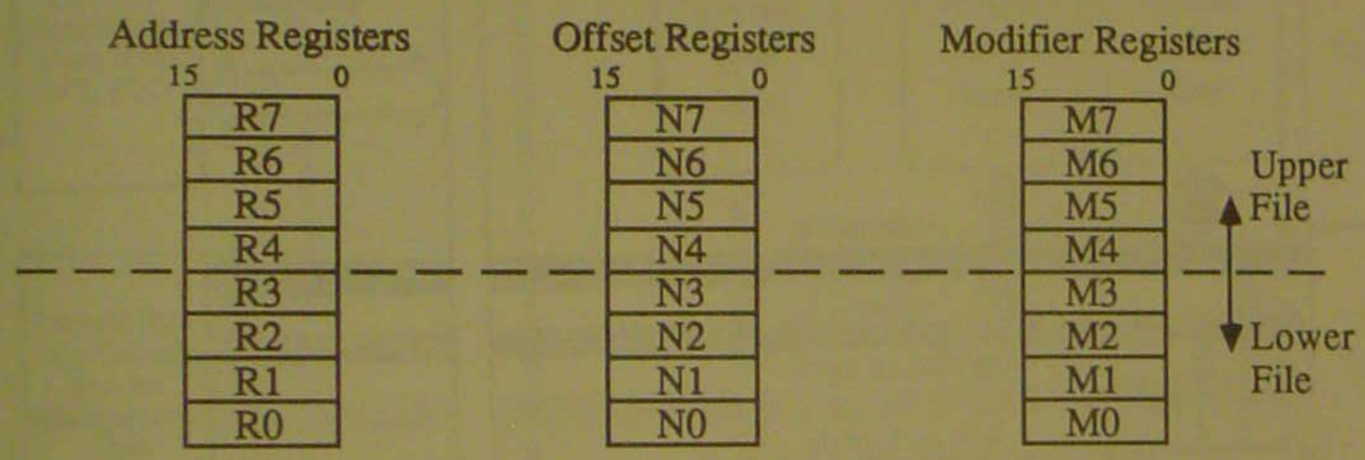
Input Registers



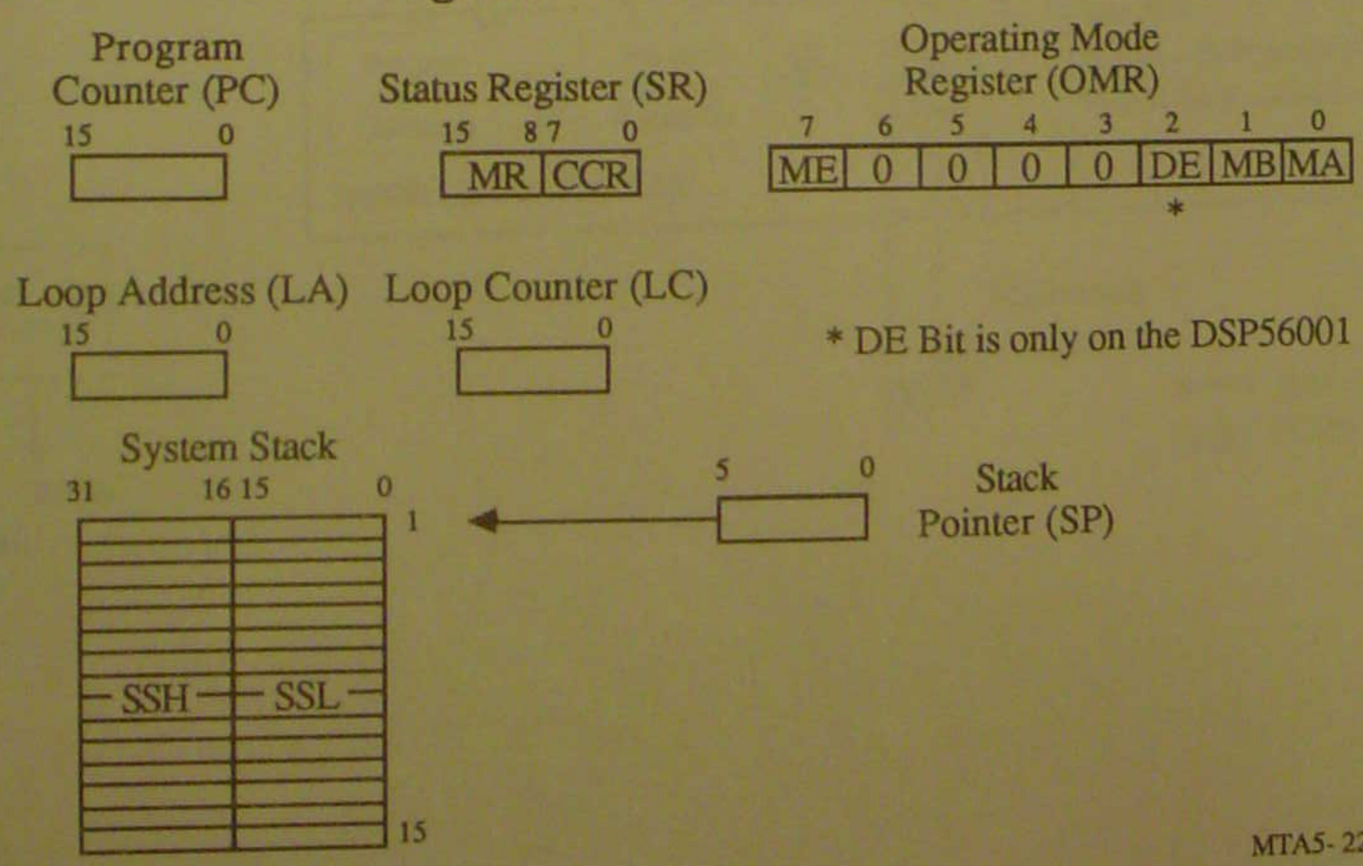

Accumulator Registers




## Address ALU



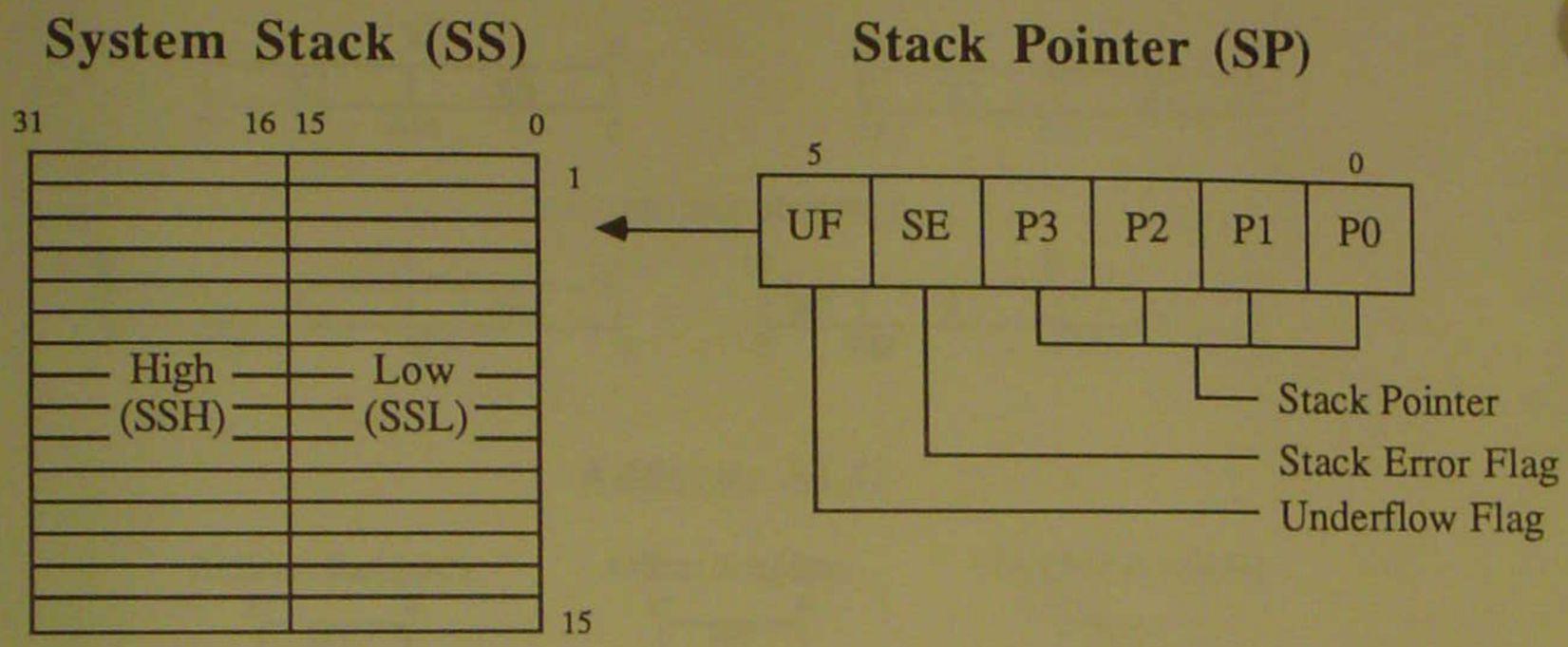

## Program Controller




* DE Bit is only on the DSP56001

MTA5- 2210

# HARDWARE STACK

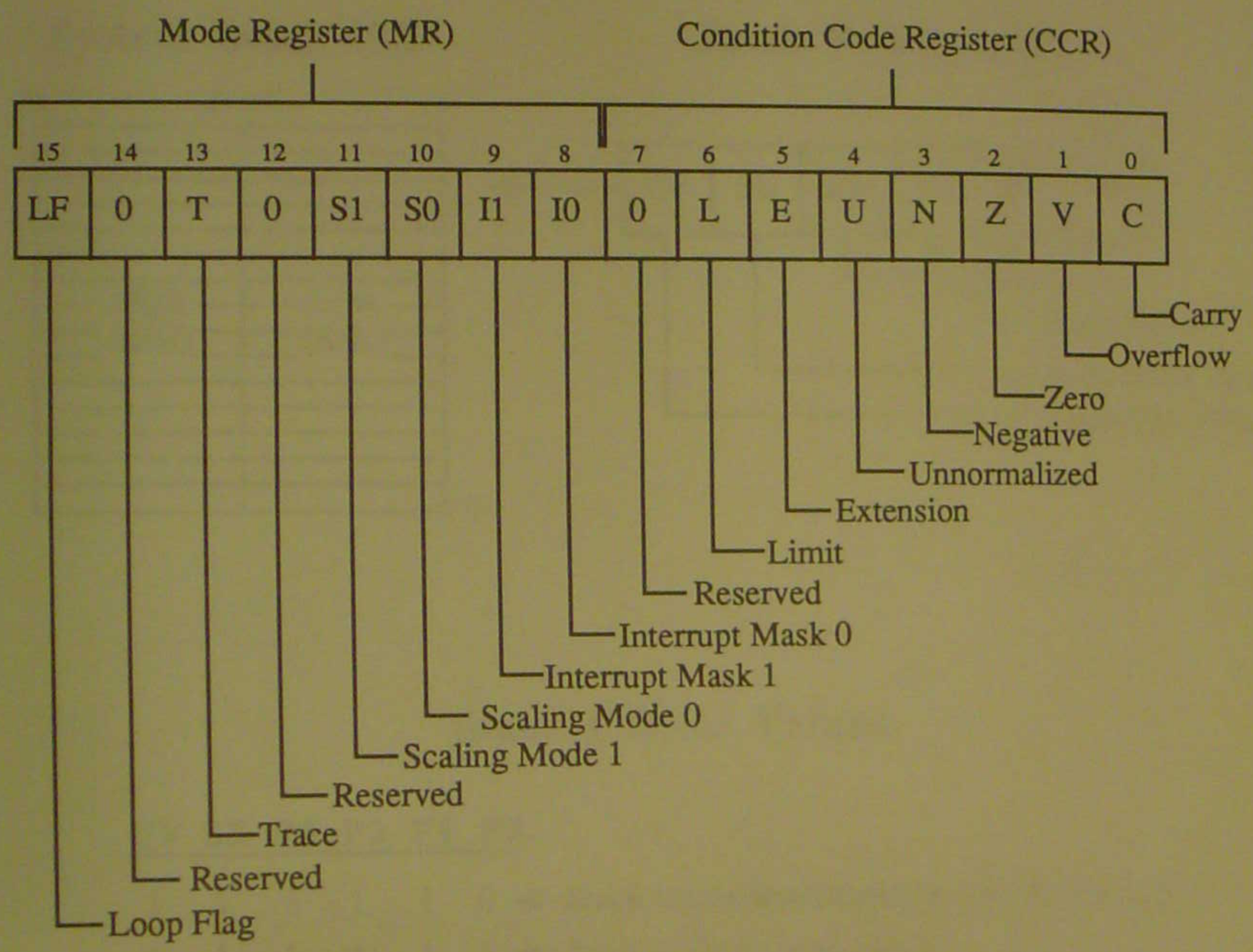



## Stack Pointer Values

| UF | SE | P3 | P2 | P1 | P0 | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | ← Stack Underflow condition after double pull |
| 1 | 1 | 1 | 1 | 1 | 1 | ← Stack Underflow condition |
| 0 | 0 | 0 | 0 | 0 | 0 | ← Stack Empty (reset). Pull causes underflow. |
| 0 | 0 | 0 | 0 | 0 | 1 | ← Stack location 1 |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| 0 | 0 | 1 | 1 | 1 | 0 | ← Stack location 14. |
| 0 | 0 | 1 | 1 | 1 | 1 | ← Stack location 15. Push causes overflow |
| 0 | 1 | 0 | 0 | 0 | 0 | ← Stack overflow condition |
| 0 | 1 | 0 | 0 | 0 | 1 | ← Stack Overflow condition after double push |

MTA5-2220

# STATUS REGISTER




MTA5-2230

# EXERCISE 2

## INTERNAL ARCHITECTURE AND PROGRAMMING MODEL

1. The DSP56000 has three seperate memory spaces called _____ , _____ and _____.

2. The DATA ALU section of the DSP56000 has two____ bit input registers called_____ _____ , and_____ , plus two____ bit accumulator registers called _____ and _____.

3. The Address ALU has three sets of _____bit registers called _____, _____ , and _____ . There are____ registers per set.

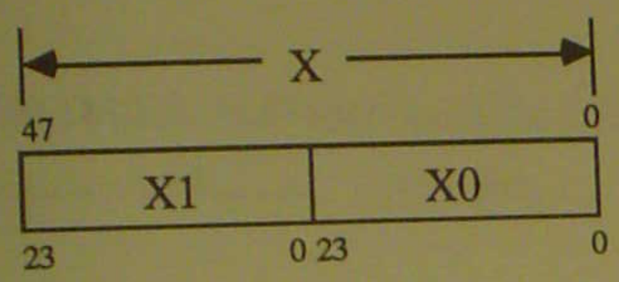
4. The system stack is _____ bits wide and_____ levels deep. Error conditions are indicated in the _____ _____ register with the _____ flag and the _____ _____ flag.

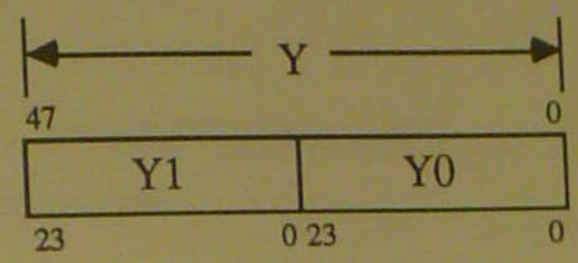
5. The Status Register is ____ bits wide, and it is divided into halves called the _____ _____ , and the _____ _____ _____.

MTA5-2240

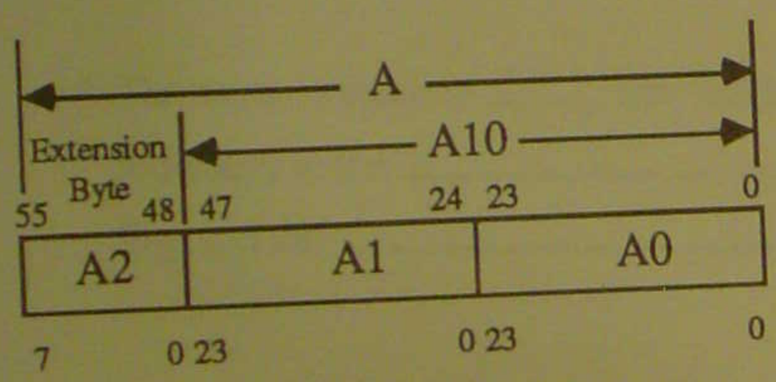
# DATA ALU REGISTERS

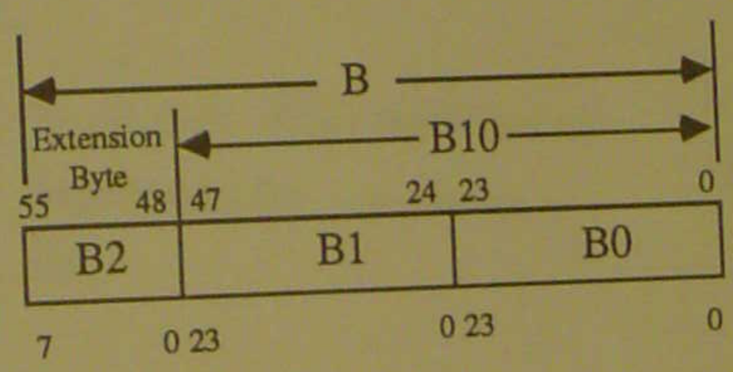
## Data ALU Input Registers



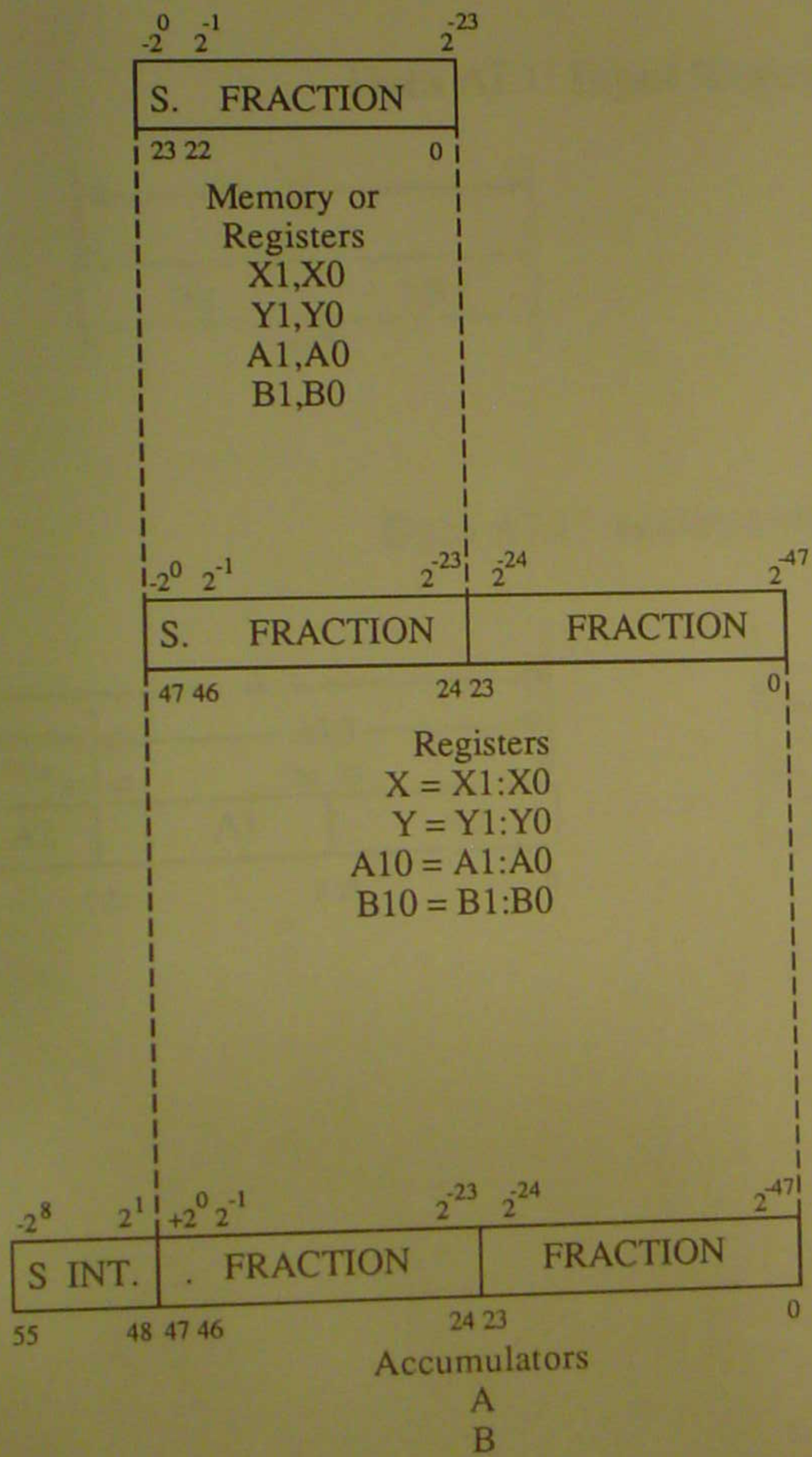

## Data ALU Accumulators




MTA5-2300

# FRACTIONAL DATA REPRESENTATION




**Word Operand**

Largest Value

$1-2^{-23}$ = $7FFFFF

Smallest Value

-1 = $800000

Dynamic Range

$144dB = 20\log_{10}(2^{24})$

**Longword Operand**

Largest Value

$1-2^{-47}$ = $7FFFFFFFFFFF

Smallest Value

-1 = $800000000000

Dynamic Range

$289dB = 20\log_{10}(2^{48})$

**Accumulator Operand**

Largest Value

$256-2^{-47}$ = $7FFFFFFFFFFFFF

Smallest Value

-256 = $80000000000000

Dynamic Range

$336dB = 20\log_{10}(2^{56})$

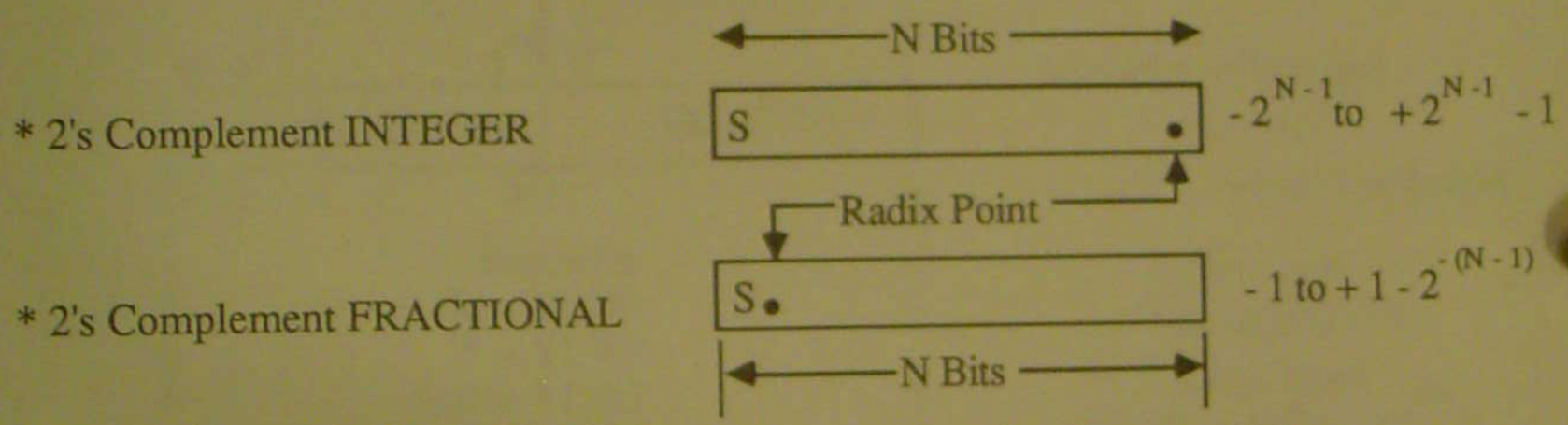
MTA5-2310-2

# FRACTIONAL COMPARISON TO ALTERNATE NUMERIC FORMATS

## Floating Point

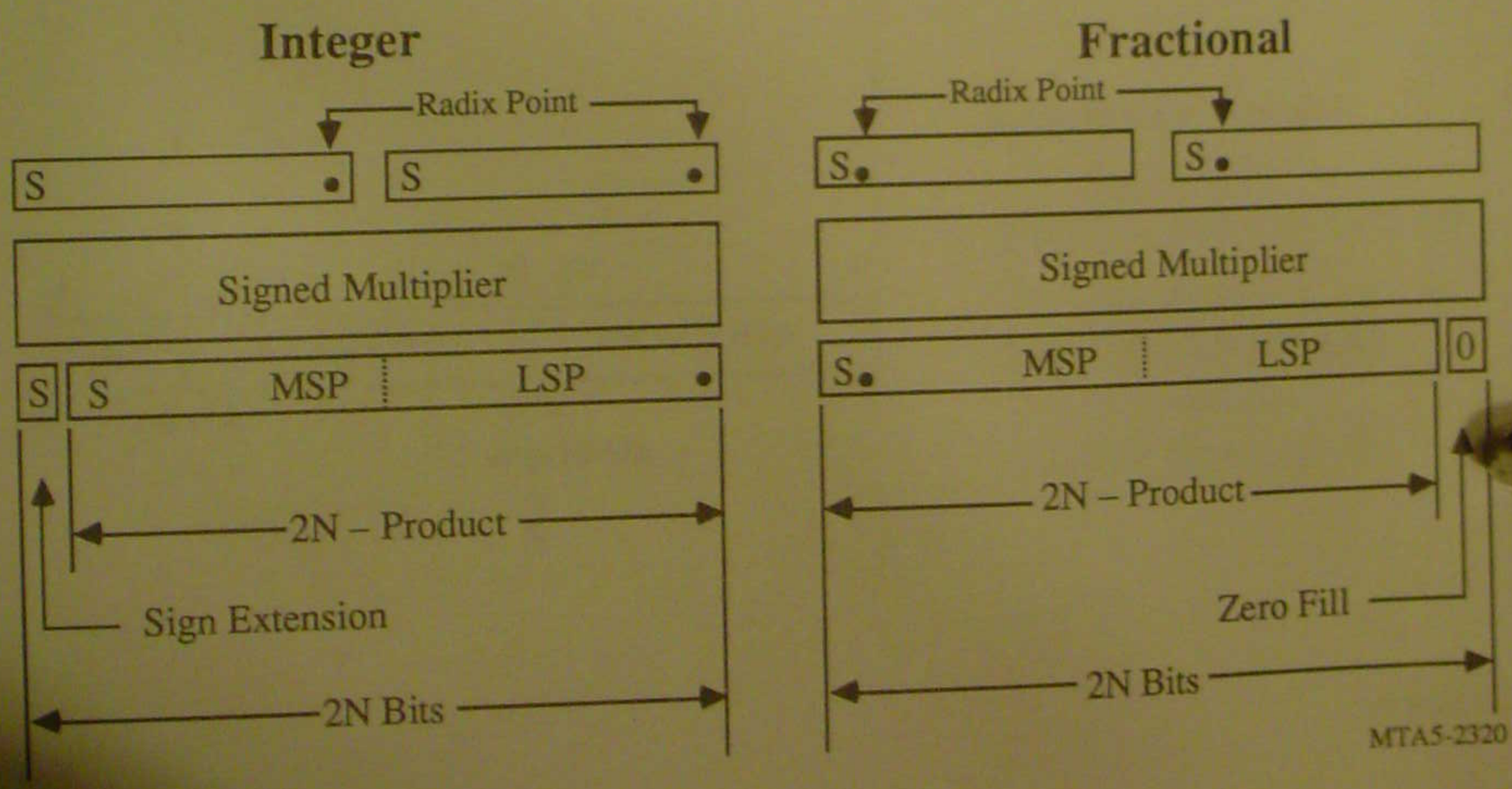
- Greater Dynamic Range
- Less Chance of errors, overflow, underflow, truncation
- Requires large and complex Multiply/Accumulate unit.
- Slow in performance
- Requires more memory
- Expensive, adds to processor cost
- Increases power consumption

## Integer



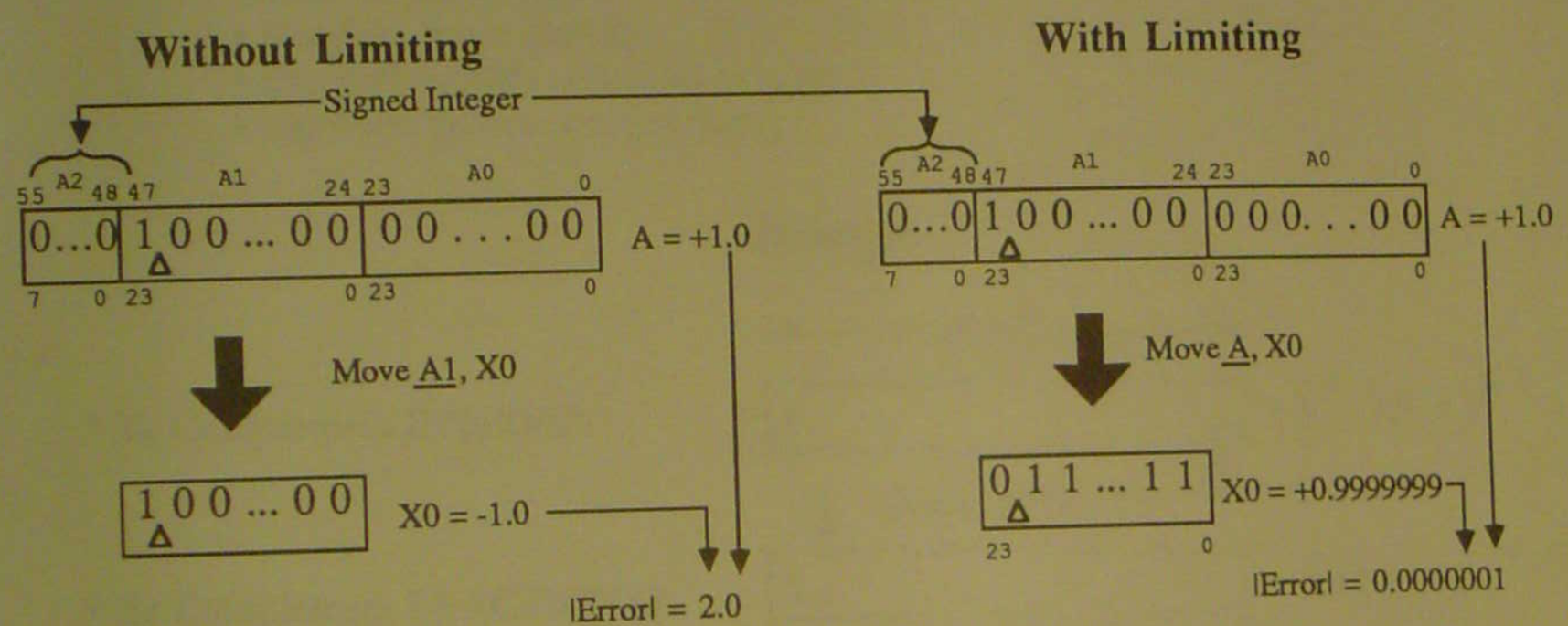

* 2's Complement INTEGER

* 2's Complement FRACTIONAL



# SATURATION ARITHMETIC

- Minimizes overflow errors when reading Accumulator Data.
- Limiting occurs automatically when transferring from Accumulator A or B into 24 or 48 bit destination.
- Latched in Condition Code Register.
- Contents of A or B are not changed.




## Cases Where Limiting Occurs

| Destination Memory Reference | Source Operand | Limited Value (Hex) Acc Sign | XD Bus | YD Bus | Type of access |
|---|---|---|---|---|---|
| X | X:A<br>X:B | +<br>- | 7FFFFF<br>800000 | ----<br>---- | one 24 bit word |
| Y | Y:A<br>Y:B | +<br>- | ----<br>---- | 7FFFFF<br>800000 | one 24 bit word |
| X and Y | X:A Y:A<br>X:A Y:B<br>X:B Y:A<br>X:B Y:B<br>L:AB<br>L:BA | +<br>- | 7FFFFF<br>800000 | 7FFFFF<br>800000 | two 24 bit words |
| L(X:Y) | L:A<br>L:B | +<br>- | 7FFFFF<br>800000 | FFFFFF<br>000000 | one 48 bit long word |

MTA5-2330

# CONVERGENT ROUNDING

The Problem:

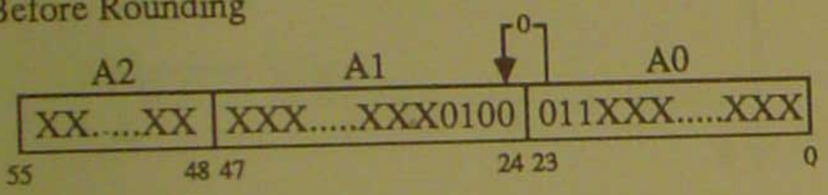
10
9
8 } Round Up to 10
7
6
5 **??? Which Way ???**
4
3 } Round Down to 0
2
1
0

Rounding Five to Either Direction Creates a Bias

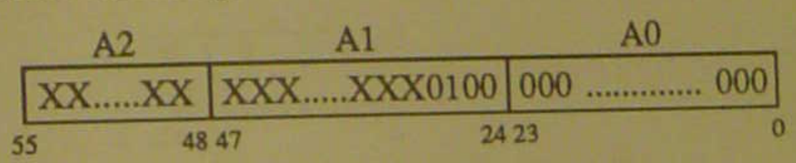
The Solution: Convergent Rounding or Round to Nearest Even.

**Case I: If A0 < $800000 (1/2), then round down (add nothing)**

Before Rounding

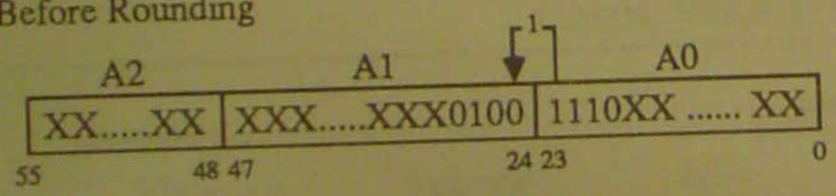



After Rounding

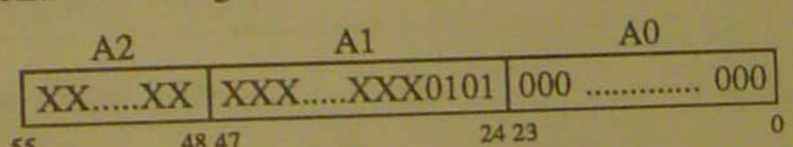



**Case II: If A0 > $800000 (1/2), then round up (add 1 to A1)**

Before Rounding

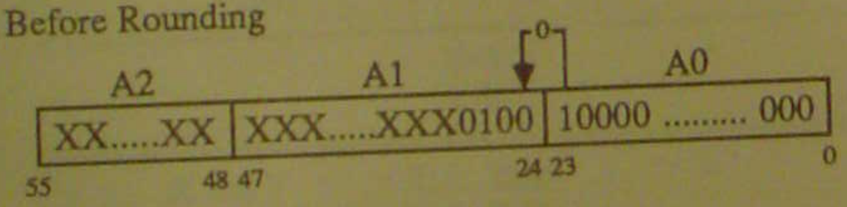



After Rounding

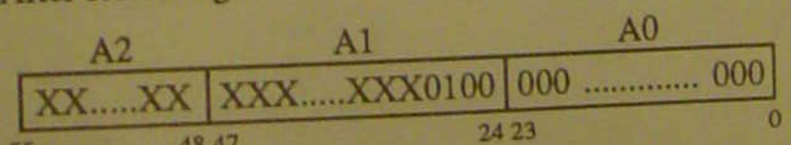



**Case III: If A0 = $800000 (1/2), and the least significant bit of A1 = 0, then round down (add nothing)**

Before Rounding

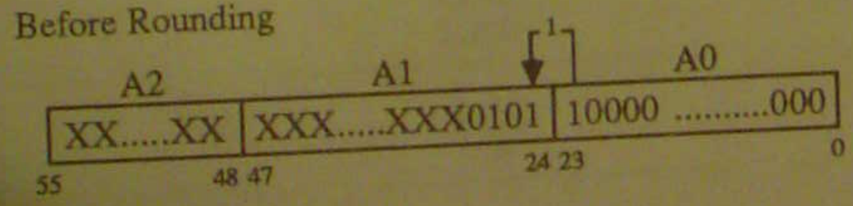



After Rounding

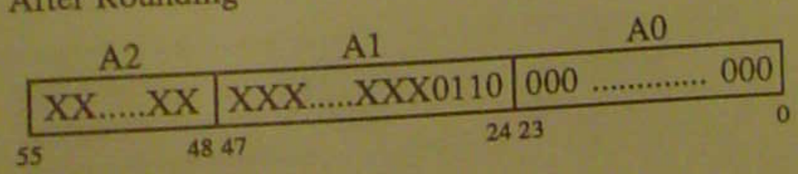



**Case IV: If A0 = $8000000 (1/2), and the least significant bit of A1 = 1, then round up (add one)**

Before Rounding




After Rounding

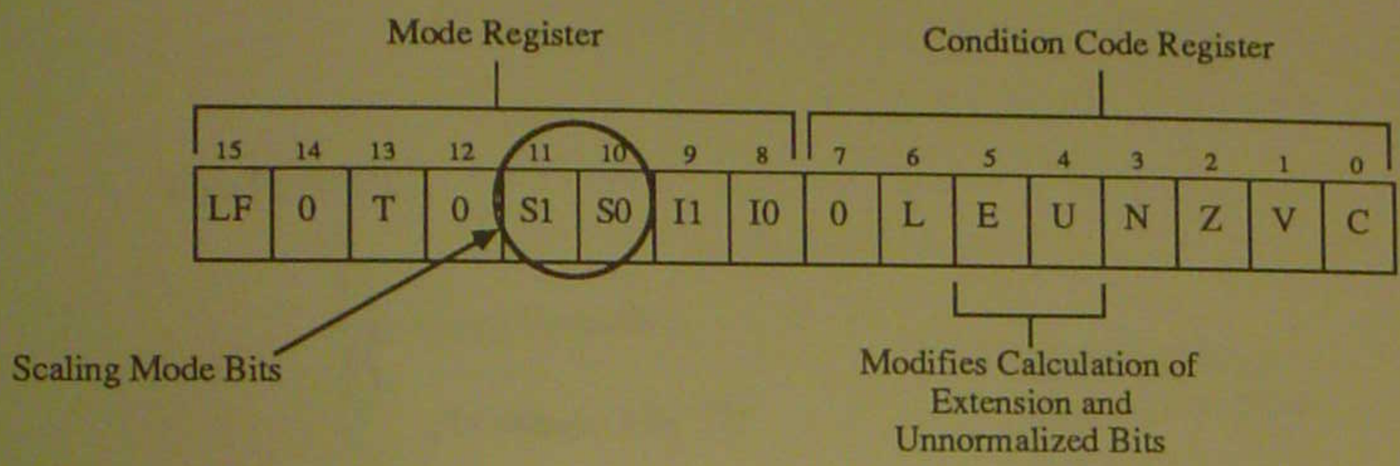



MTA5- 2340-1

- A0 is always cleared.
- Performed During RND, MPYR, MACR.

# SCALING MODES

## Status Register




| S1 | S0 | Rounding bit | Scaling Mode |
|---|---|---|---|
| 0 | 0 | 23 | No Scaling |
| 0 | 1 | 24 | Scale Down (1 bit arithmetic right shift) |
| 1 | 0 | 22 | Scale Up (1 bit arithmetic left shift) |
| 1 | 1 | -- | Reserved for future expansion |

- Allows Dynamic Scaling without changing program code.
- Affects data read from A or B Accumulators
- Scaling bits are cleared during reset and long interrupt service routine.

## Scaling Affects Convergent Rounding

| S1 | S0 | Scaling Mode | Rounding Position | Rounding Constant 55-25 | 24 | 23 | 22 | 21-0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | No Scaling | 23 | 0 ... 0 | 0 | 1 | 0 | 0 ... 0 |
| 0 | 1 | Scale Down | 24 | 0 ... 0 | 1 | 0 | 0 | 0 ... 0 |
| 1 | 0 | Scale Up | 22 | 0 ... 0 | 0 | 0 | 1 | 0 ... 0 |

## Scaling Affects Extension Bit

- Integer Portion of A or B Accumulator
- E=0 When Integer Portion is 11 ... 11 or 00 ... 00, else E=1 for all other patterns

| S1 | S0 | Scaling Mode | Integer Portion |
|---|---|---|---|
| 0 | 0 | No Scaling | Bits 55,54,.....,48,47 |
| 0 | 1 | Scale down | Bits 55,54,.....,49,48 |
| 1 | 0 | Scale up | Bits 55,54,.....,47,46 |

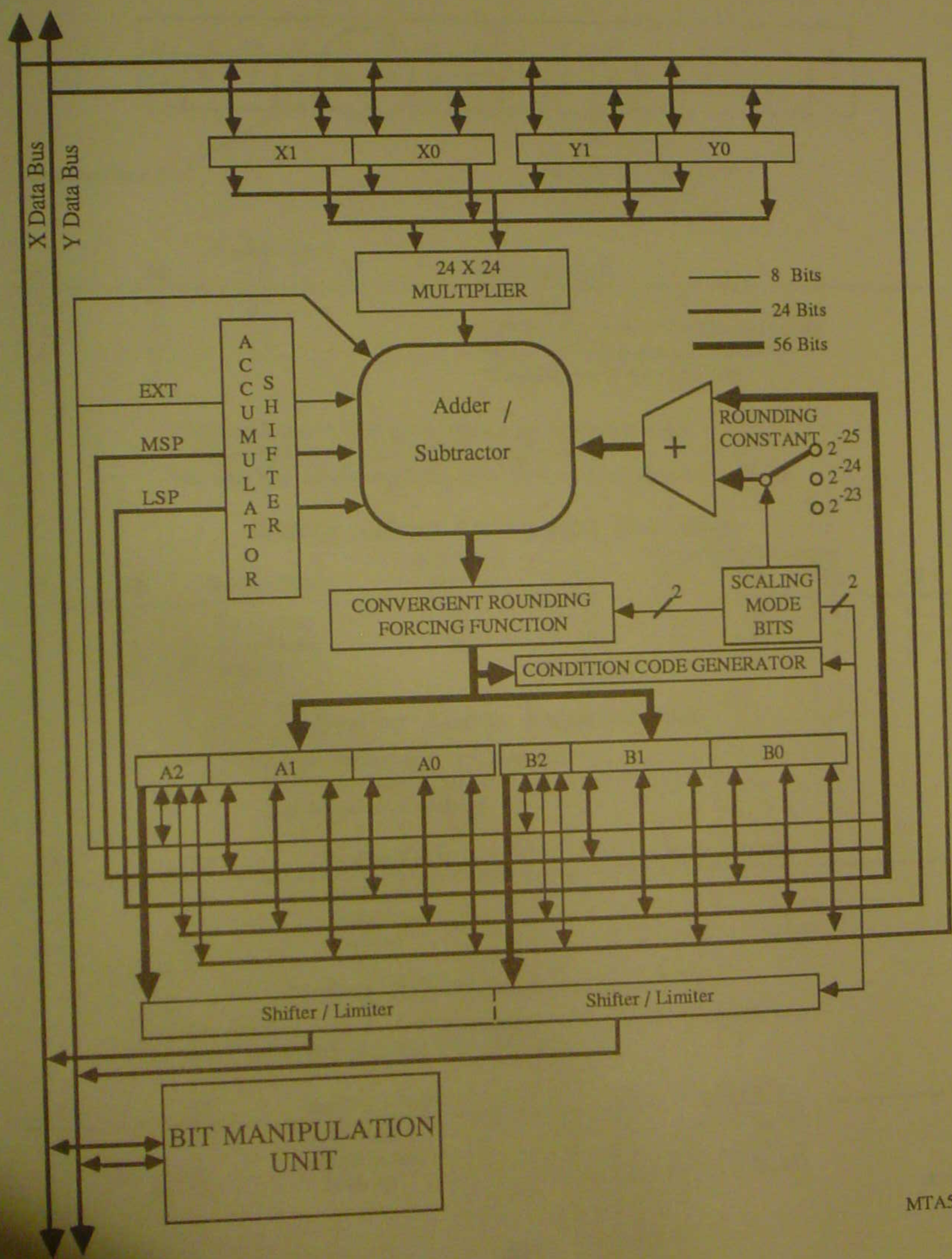
## Scaling Affects Unnormalized Bit

- Unnormalized when the two most significant bits of the most significant product of accumulator A or B are the same.

| S1 | S0 | Scaling Mode | U Bit Computation |
|---|---|---|---|
| 0 | 0 | No Scaling | U = ~ (Bit 47 ⊕ Bit 46) |
| 0 | 1 | Scale down | U = ~ (Bit 48 ⊕ Bit 47) |
| 1 | 0 | Scale up | U = ~ (Bit 46 ⊕ Bit 45) |

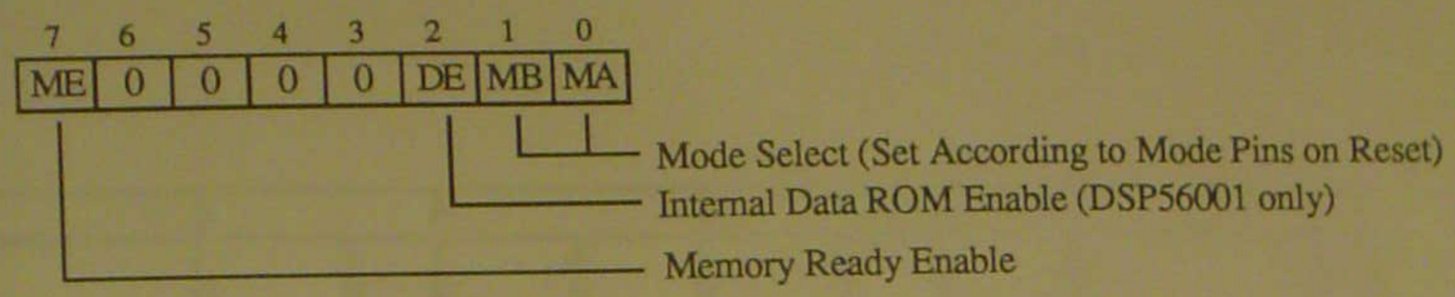
MTA5- 2400

# DATA ARITHMETIC LOGIC UNIT




MTA5-2410

# OPERATING MODE REGISTER




| Operating Mode | Mode Pins | | DSP56000 Program Memory Map | | |
|---|---|---|---|---|---|
| | MB | MA | Internal ROM | External | Reset |
| 0 Non-Expanded | 0 | 0 | $0000 to $0EFF | $0F00 to $FFFF | Internal at $0000 |
| 1 Undefined | 0 | 1 | Undefined | | |
| 2 Normal Expanded | 1 | 0 | $0000 to $0EFF | $0F00 to $FFFF | External at $E000 |
| 3 Development | 1 | 1 | Disabled | $0000 to $FFFF | External at $0000 |

| Operating Mode | Mode Pins | | DSP56001 Program Memory Map | | |
|---|---|---|---|---|---|
| | MB | MA | Internal RAM | External | Reset |
| 0 Non-Expanded | 0 | 0 | $0000 to $01FF | $0200 to $FFFF | Internal at $0000 |
| 1 Bootstrap | 0 | 1 | Special Bootstrap Mode: After Loading Program RAM from External Source. Mode 2 is Automatically Selected and Program Counter = $0000 | | |
| 2 Normal Expanded | 1 | 0 | $0000 to $01FF | $0200 to $FFFF | External at $E000 |
| 3 Development | 1 | 1 | Disabled | $0000 to $FFFF | External at $0000 |

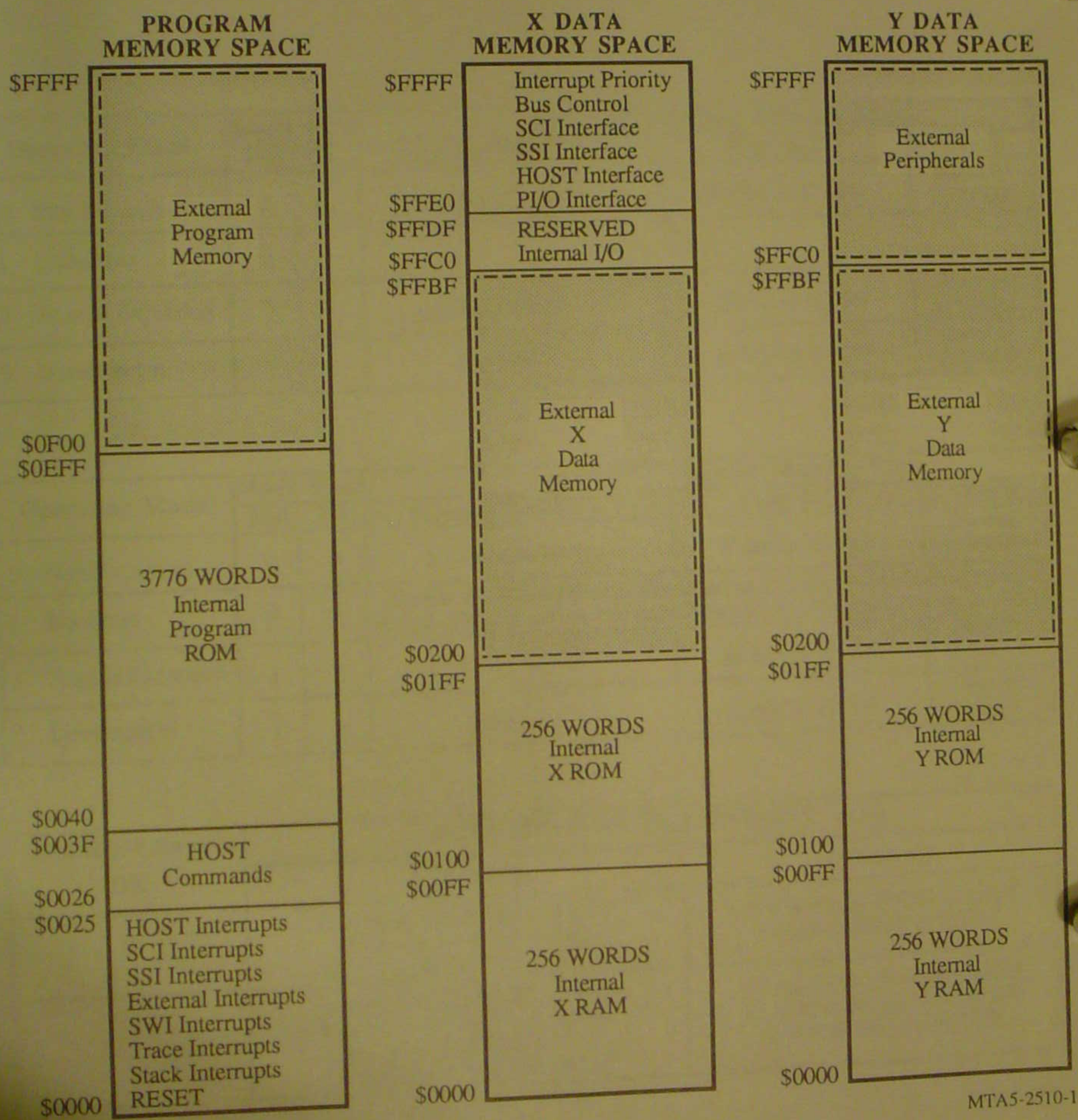
| Data ROM Enable DE | DATA MEMORY MAP (DSP56001 only) | |
|---|---|---|
| | X Memory | Y Memory |
| 0 | Internal RAM: $0000 to $00FF<br>External: $0100 to $FFBF<br>On Chip I/O: $FFC0 to $FFFF | Internal RAM: $0000 to $00FF<br>External: $0100 to $FFFF |
| 1 | Internal RAM: $0000 to $00FF<br>Internal ROM: $0100 to $01FF<br>External: $0200 to $FFBF<br>On Chip I/O: $FFC0 to $FFFF | Internal RAM: $0000 to $00FF<br>Internal ROM: $0100 to $01FF<br>External: $0200 to $FFFF |

MTA5-2500

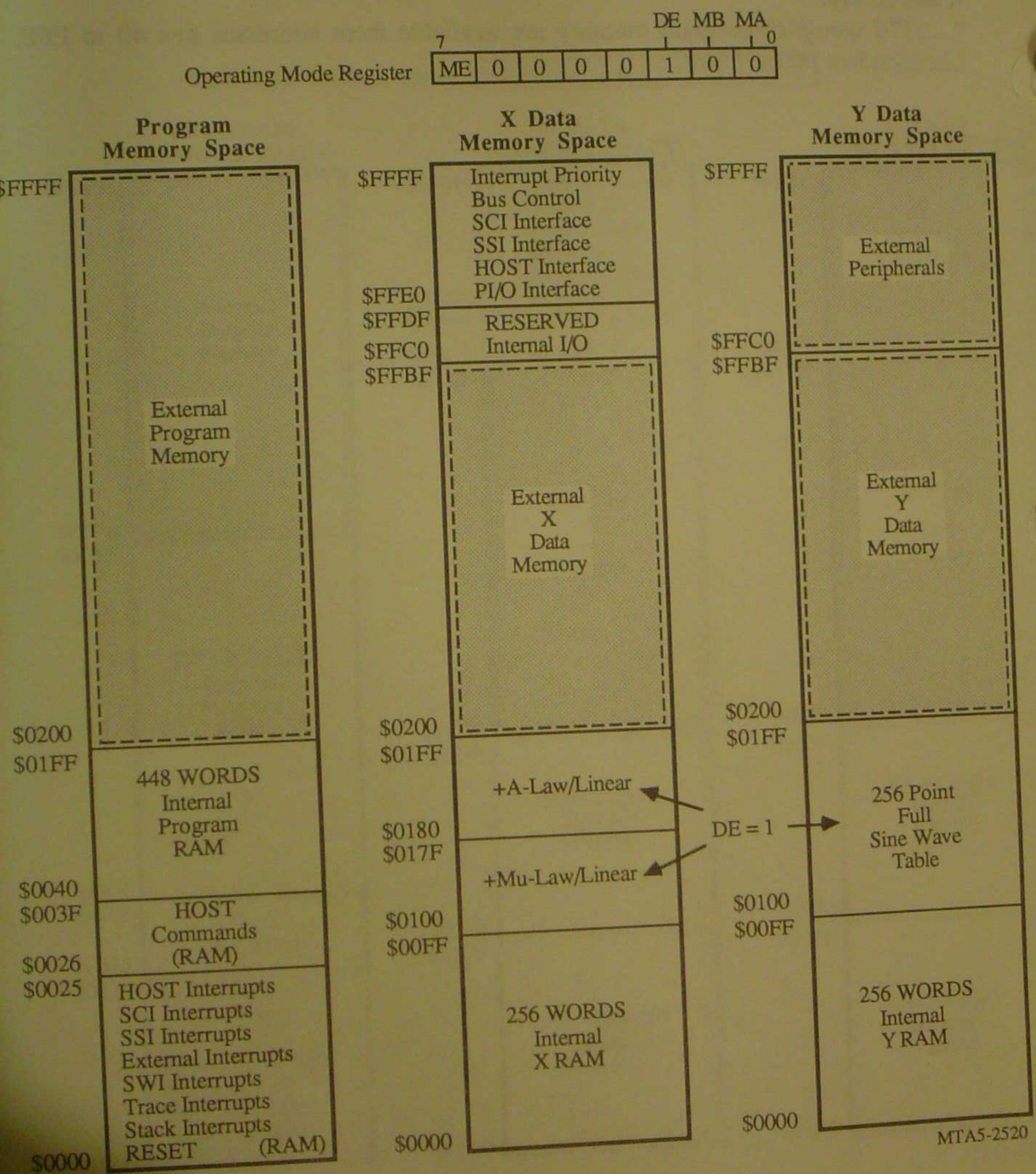
# DSP56000 MEMORY MAP
# MODE 0 - NORMAL EXPANDED AND INTERNAL RESET

**Operating Mode Register**

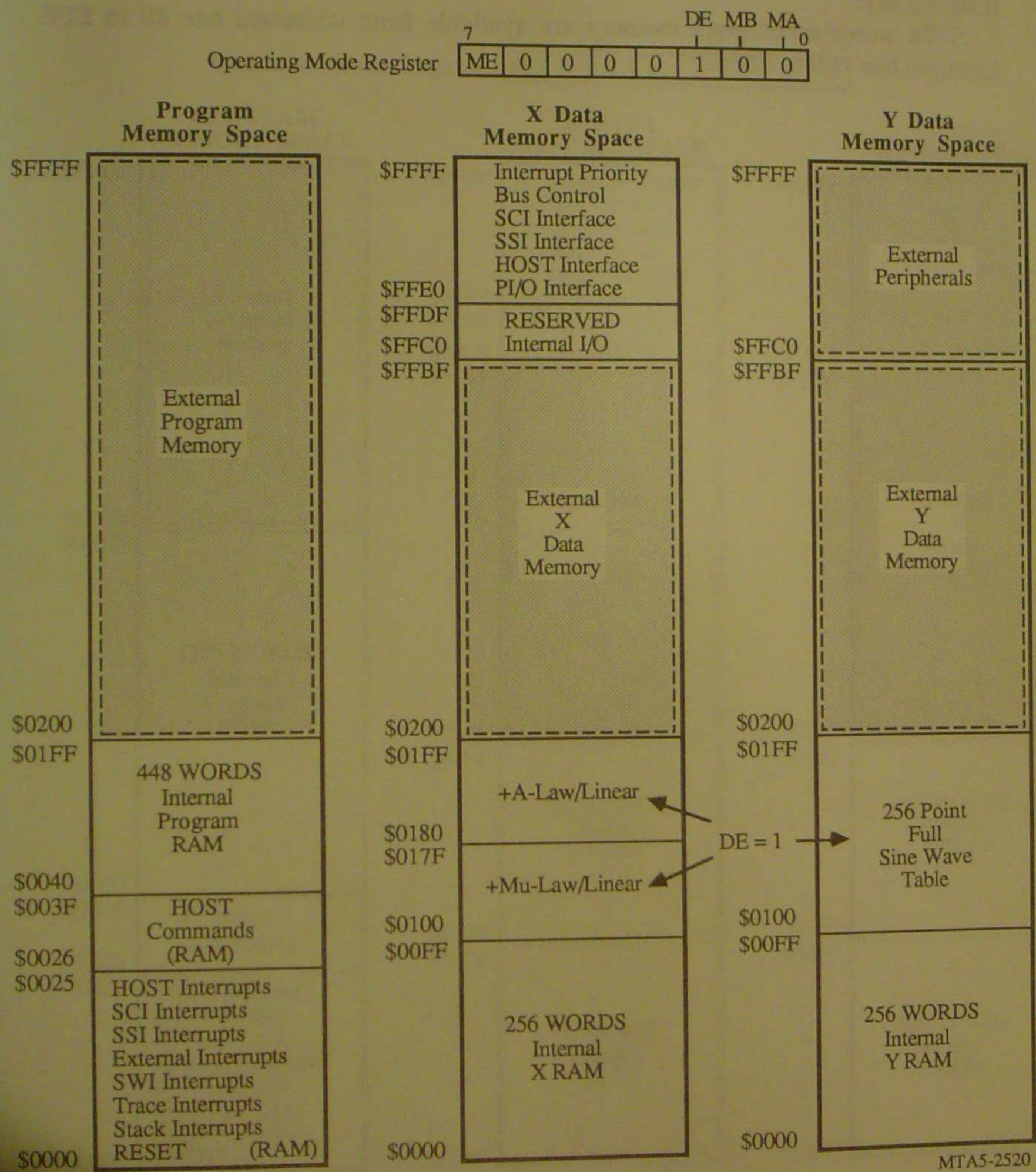
| 7 | | | | | | MB | MA 0 |
|---|---|---|---|---|---|---|---|
| ME | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



# DSP56001 MEMORY MAP MODE 0 - NORMAL EXPANDED AND INTERNAL RESET



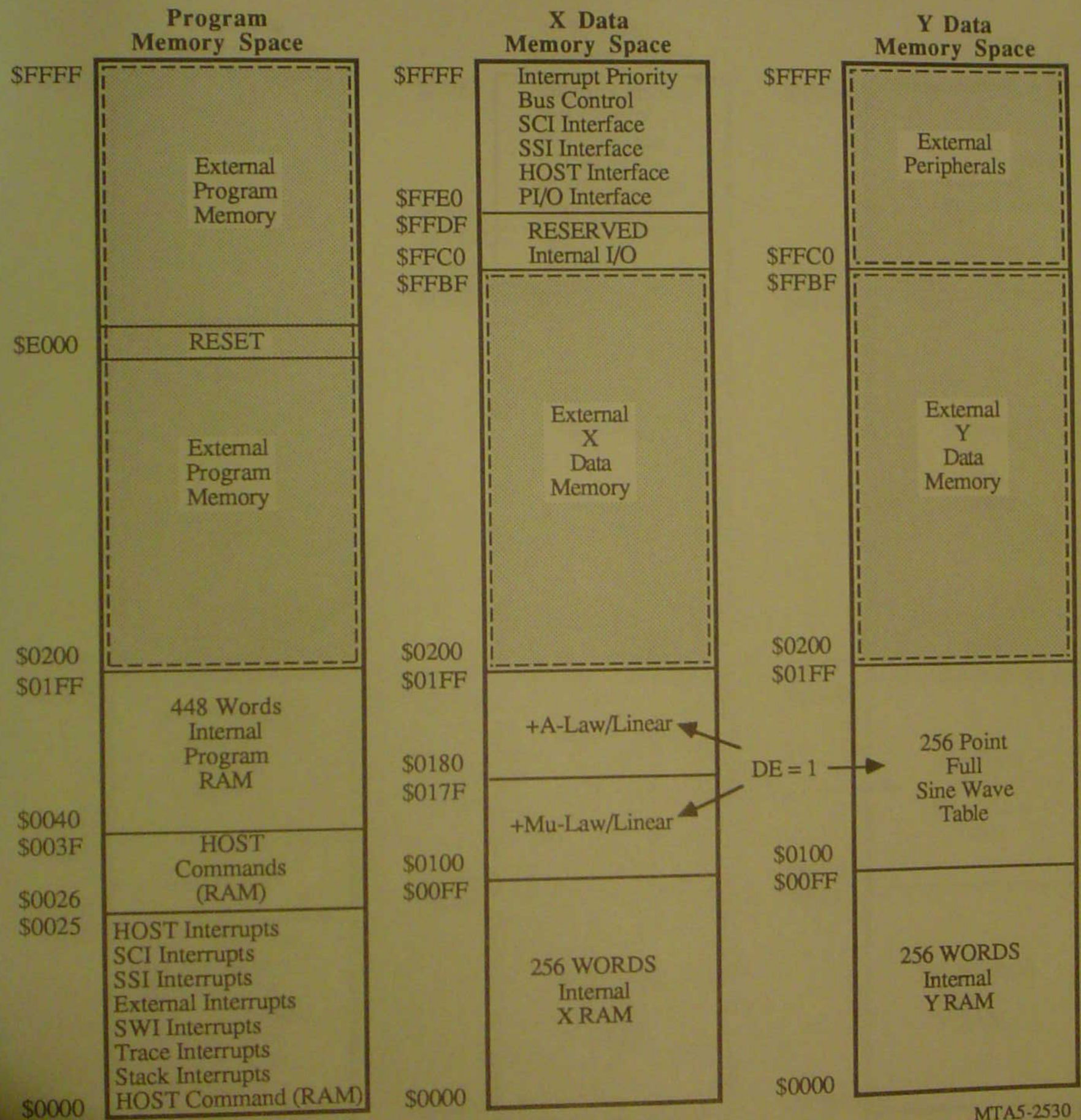
# DSP56001 MEMORY MAP
# MODE 0 - NORMAL EXPANDED AND INTERNAL RESET



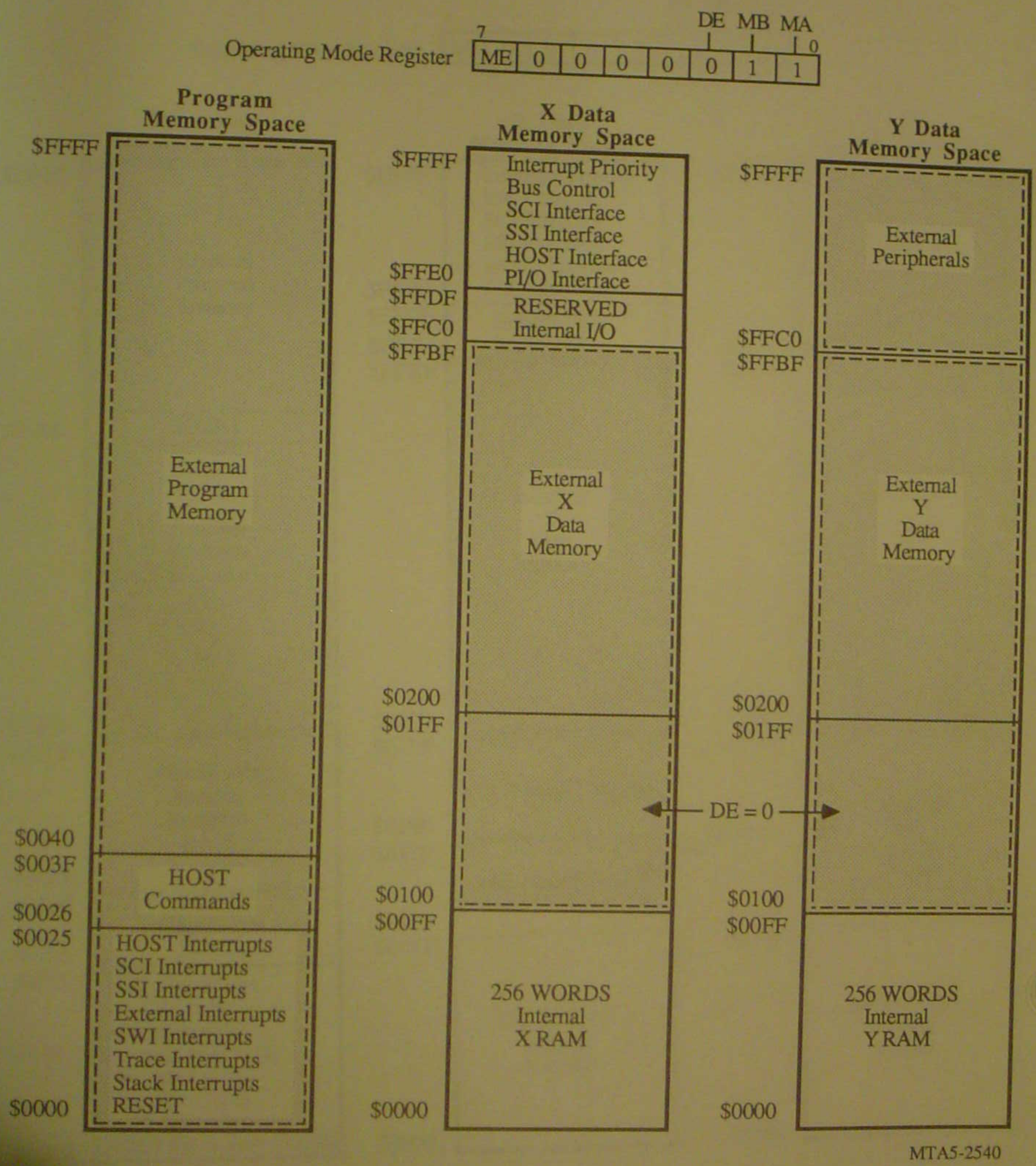
# DSP56001 MEMORY MAP MODE 2 - NORMAL EXPANDED AND EXTERNAL RESET

Operating Mode Register

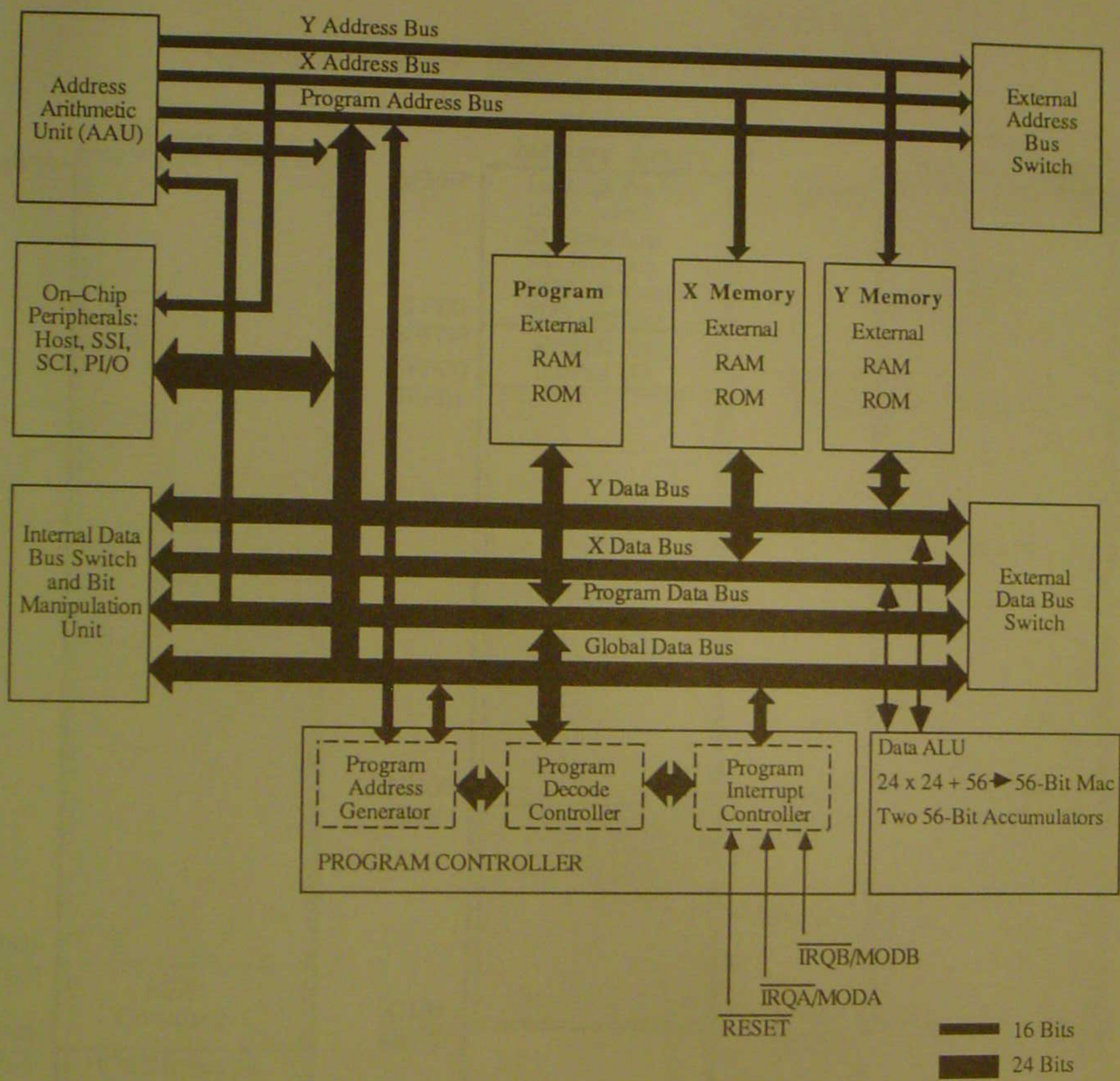
| 7 | | | | | DE | MB | MA 0 |
|---|---|---|---|---|---|---|---|
| ME | 0 | 0 | 0 | 0 | 1 | 1 | 0 |



# DSP56001 MEMORY MAP
# MODE 3 - DEVELOPMENT MODE
# EXTERNAL PROGRAM MEMORY

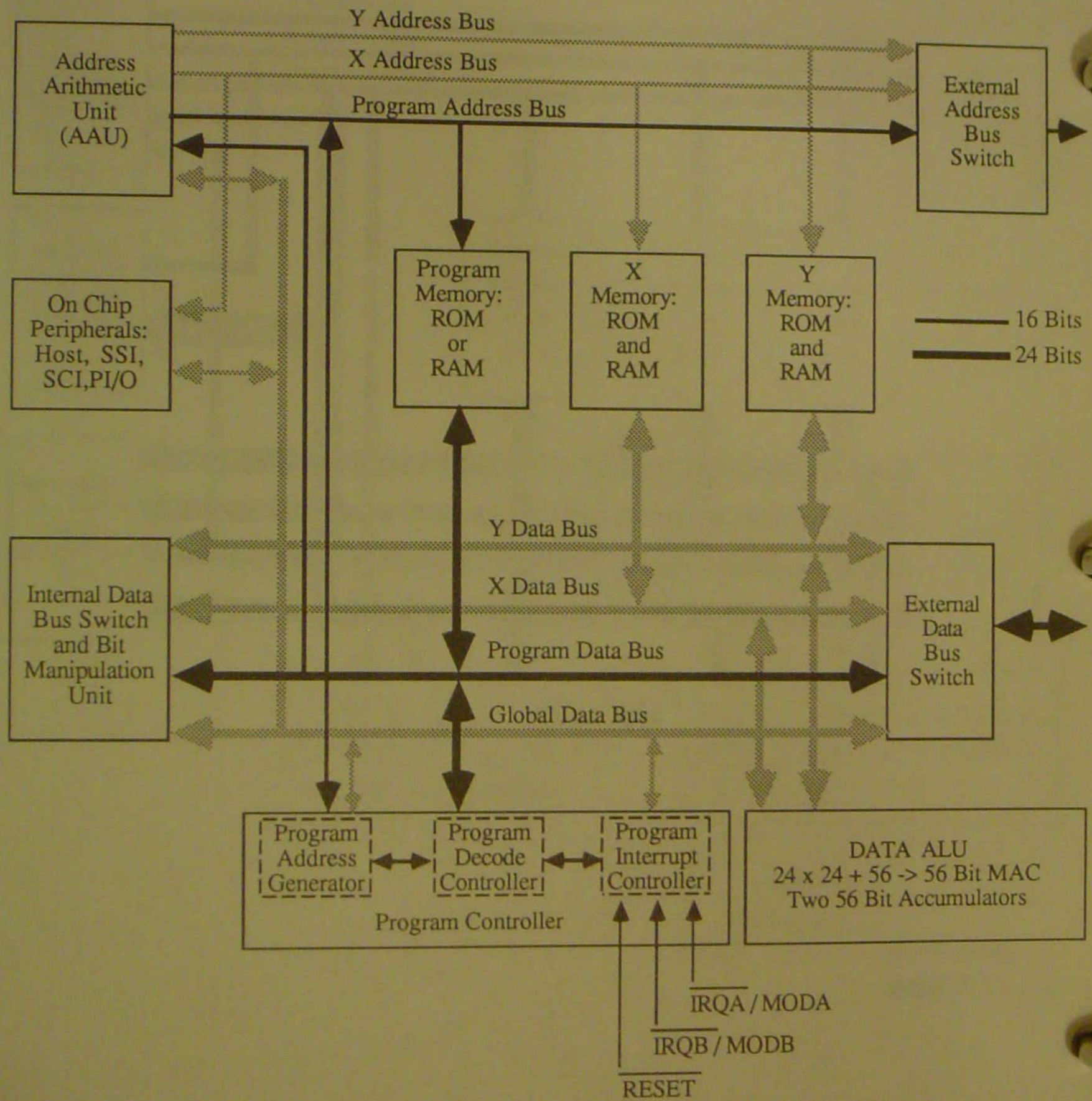


# HARVARD ARCHITECTURE



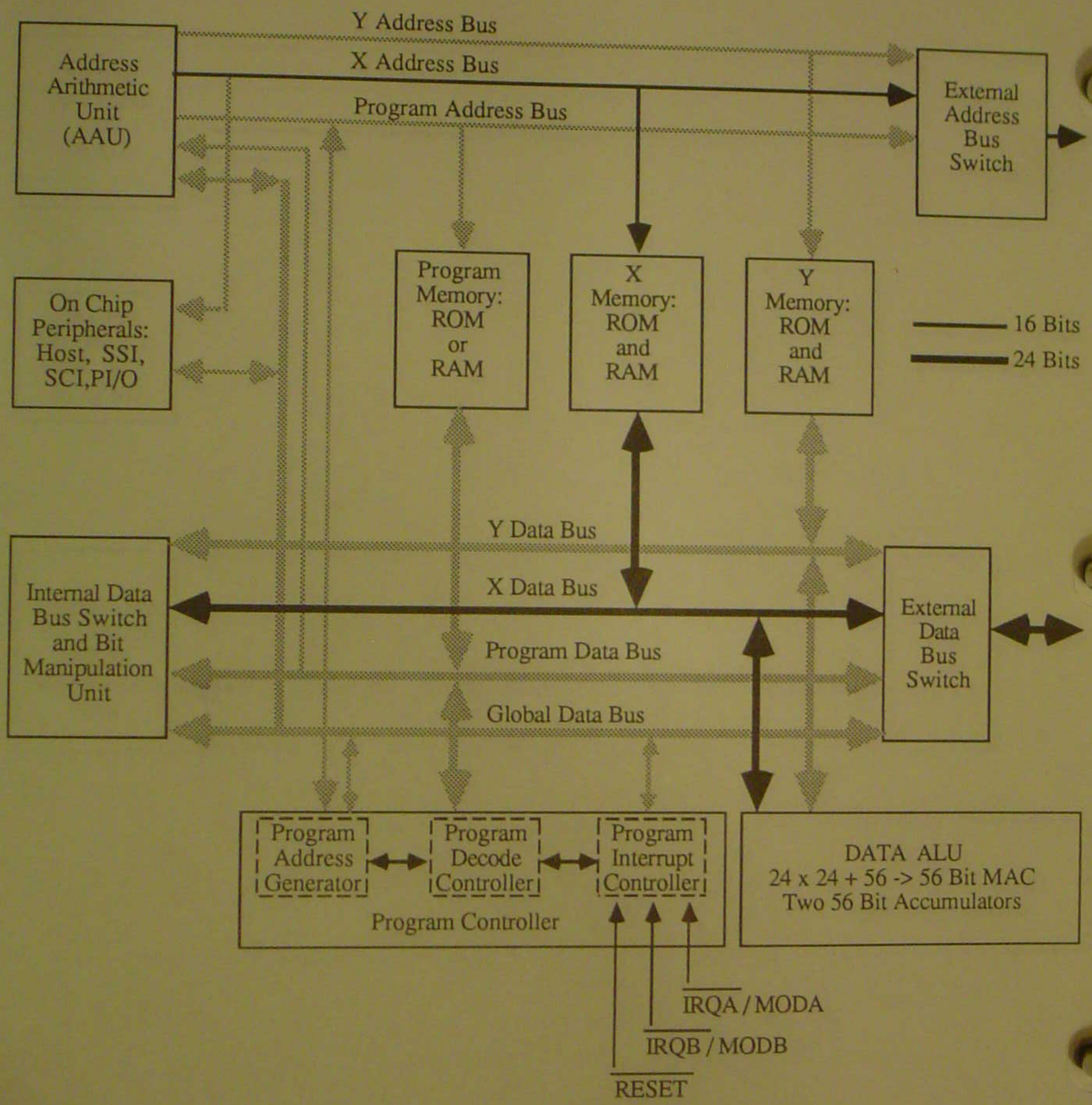

MTA5- 2600-1

# PROGRAM BUS TRANSFERS



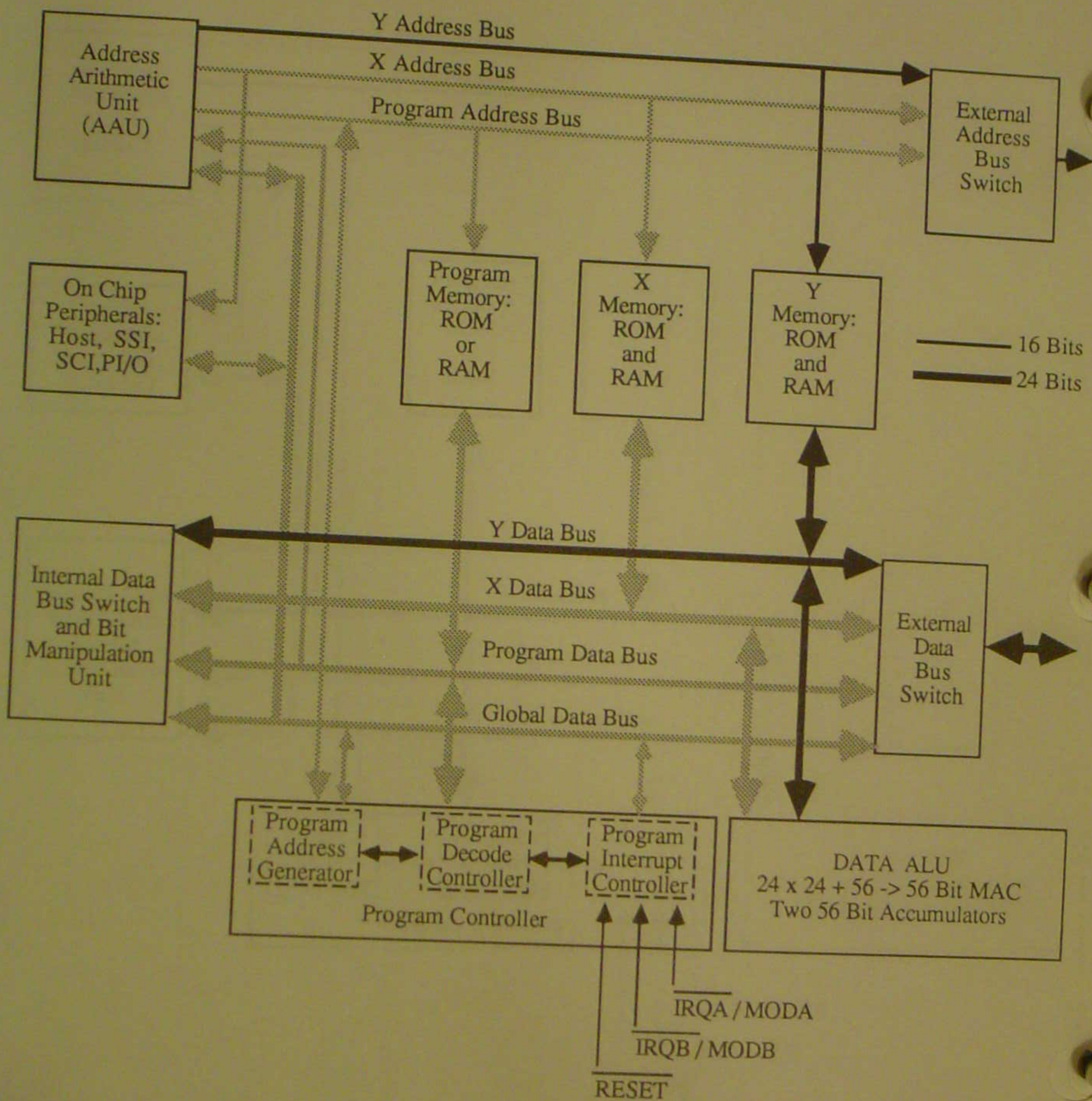

MTA5-2610-1

# X BUS TRANSFERS



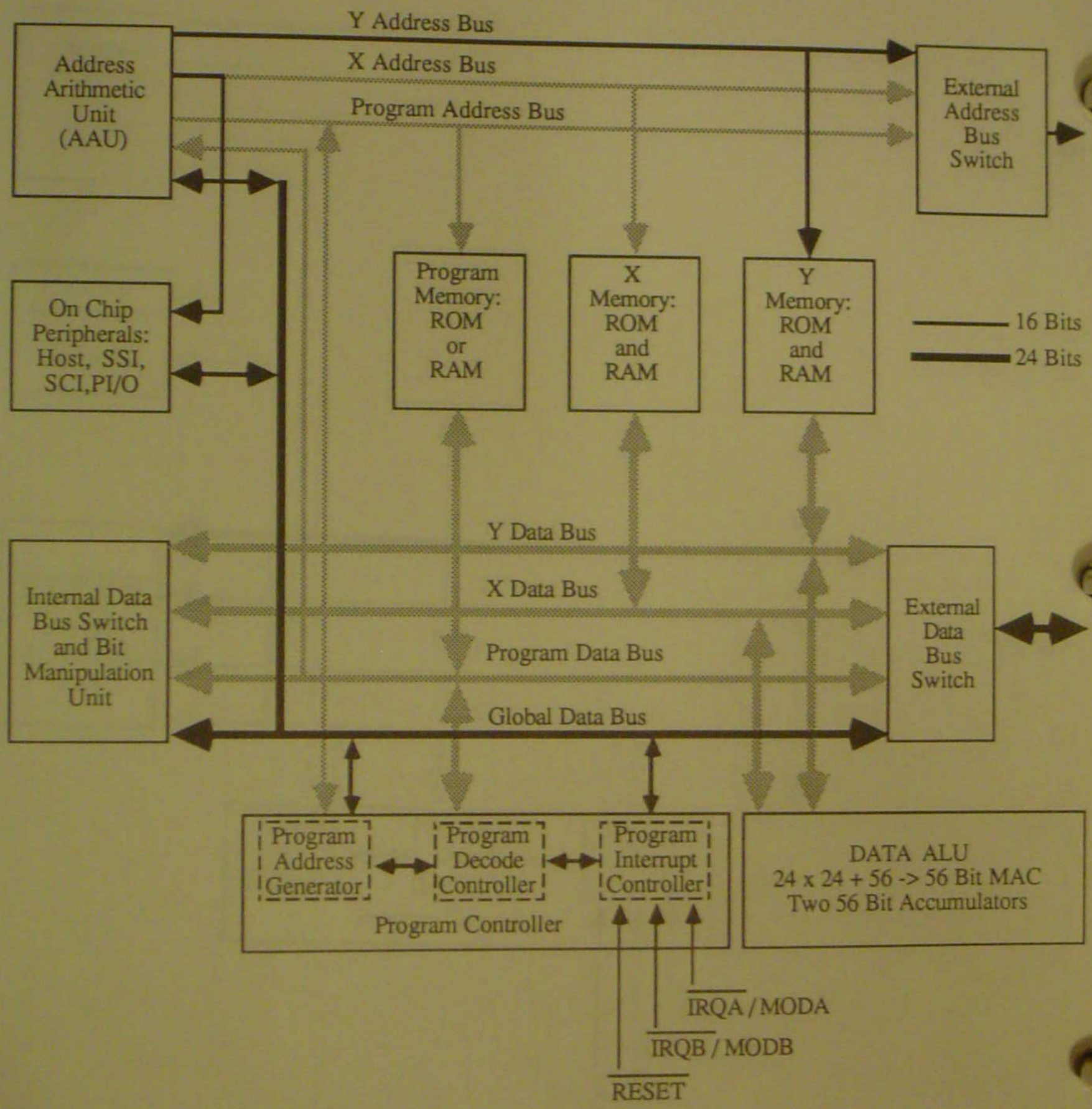

MTA5-2620-1

# Y BUS TRANSFERS




MTA5-2630-1

# GLOBAL BUS TRANSFERS

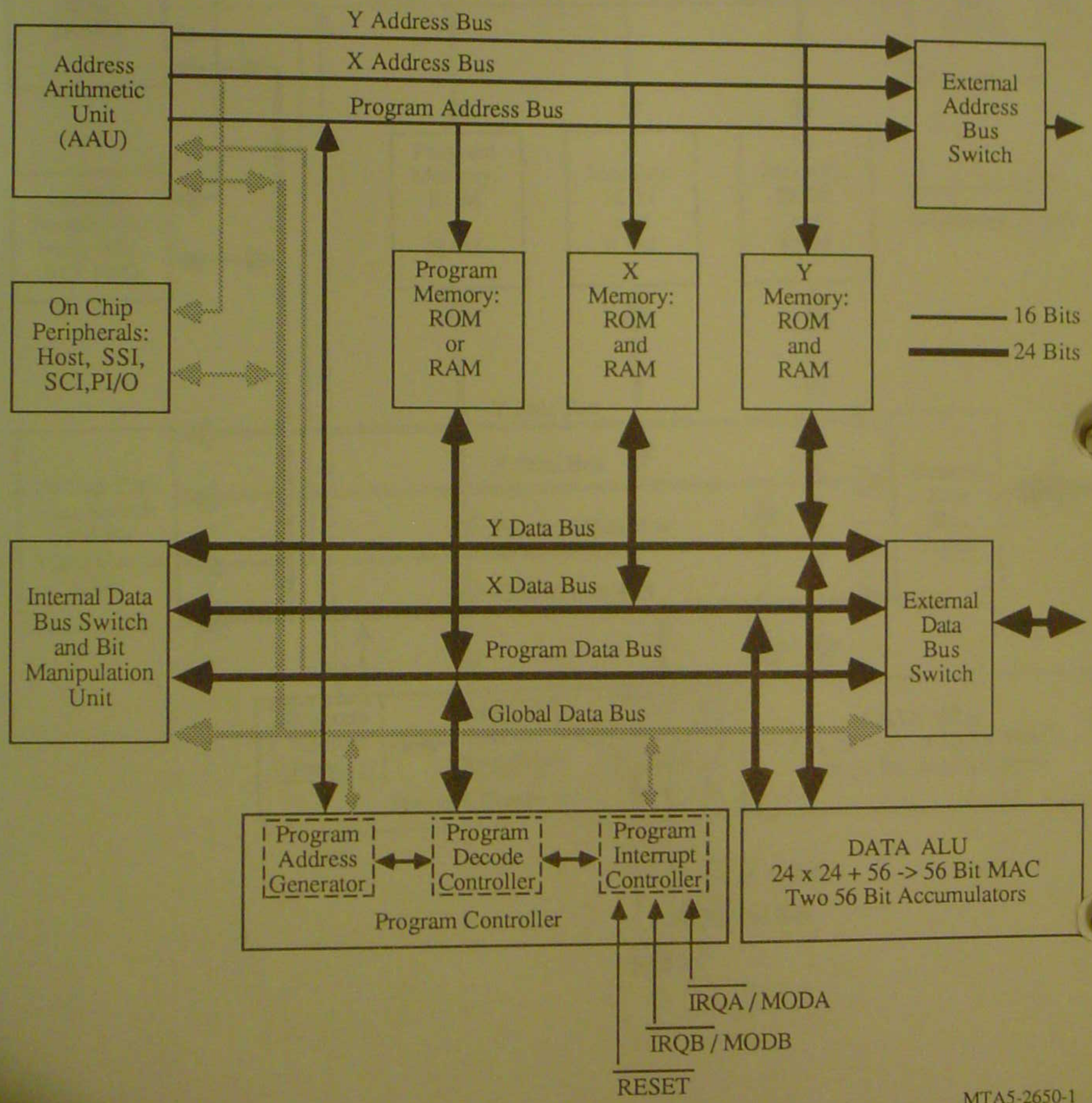



MTA5-2640

# PARALLEL BUS TRANSFERS

**Instruction Pipeline**

| Instruction Cycle (Icyc) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Fetch | F1 | F2 | F3 | F4 | F4e | F5 | F6 | F7 | • | • | • |
| Decode | – | D1 | D2 | D3 | D4 | D4e | D5 | D6 | • | • | • |
| Execute | – | – | E1 | E2 | E3 | E4 | E4e | E5 | • | • | • |




MTA5-2650-1

# EXERCISE 4

## INTERNAL ARCHITECTURE AND PROGRAMMING MODEL

1. The scaling mode bits in the status registers shift transfers right or left from accumulators A or B. These shifts are reflected in the calculation of _____ _____ , plus the _____ bit and the _____ bit in the condition code register.

2. Given that the operating mode register in the DSP56000 is programmed MB = 0, and MA = 0, fill in the addresses in the table below. If the memory resource does not exist for this configuration, then write N/A for that table entry.

| | Program Memory | X Memory | Y Memory |
|---|---|---|---|
| Peripheral stop address | _____ | _____ | _____ |
| Peripheral start address | _____ | _____ | _____ |
| External memory stop address | _____ | _____ | _____ |
| External memory start address | _____ | _____ | _____ |
| Internal ROM stop address | _____ | _____ | _____ |
| Internal ROM start address | _____ | _____ | _____ |
| Internal RAM stop address | _____ | _____ | _____ |
| Internal RAM start address | _____ | _____ | _____ |

3. What is the basic difference between the DSP56000 and DSP56001.

MTA5-2700

# PINS AND BUSES

## OBJECTIVES

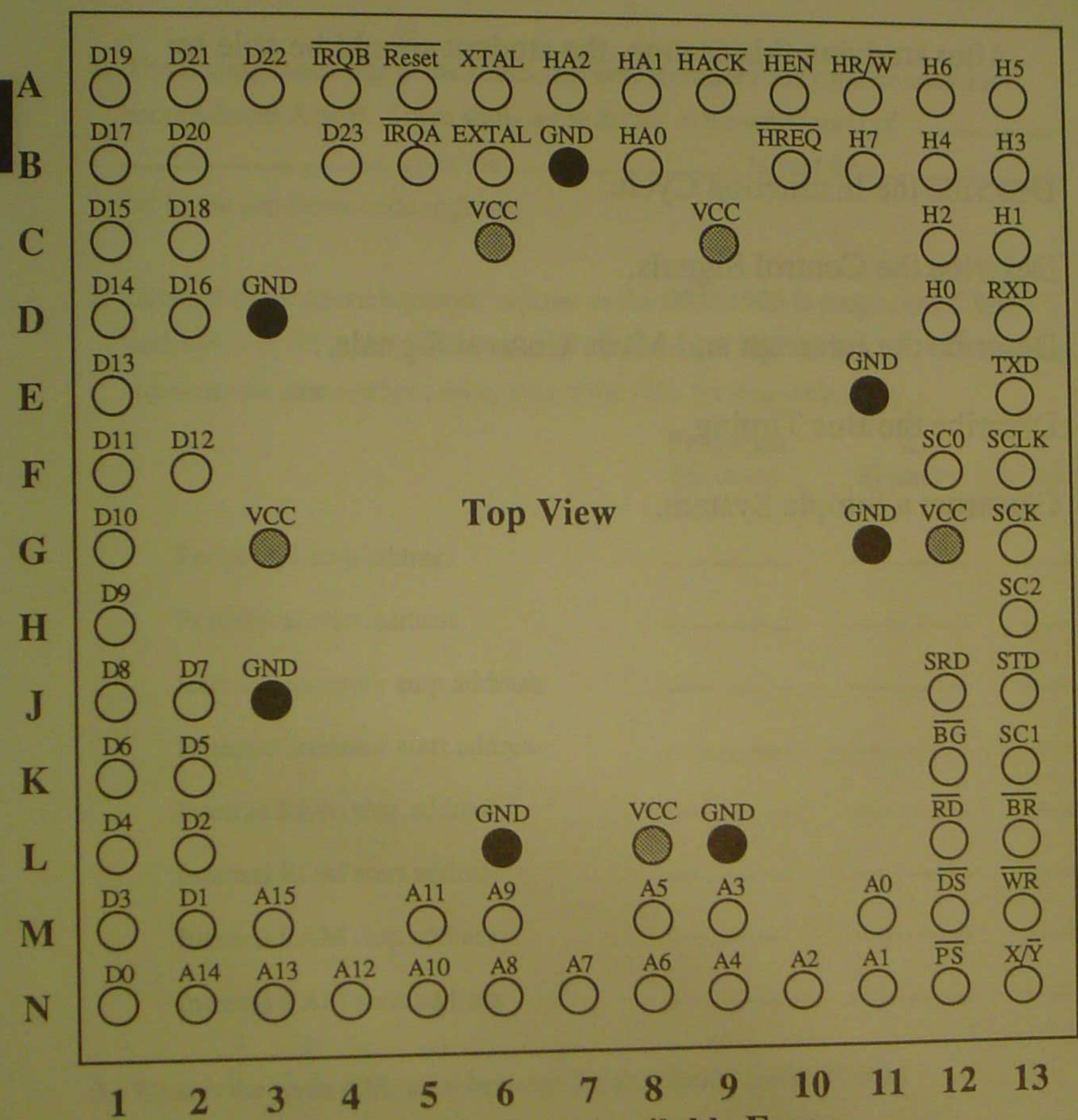
After studying this section, the student should be able to:

2

- Describe the Instruction Cycle.
- Describe the Control Signals.
- Describe the Interrupt and Mode Control Signals.
- Describe the Bus Timing.
- Construct a Simple System.

MTA5-3000

# DSP56000/1 PIN GRID ARRAY PINOUT

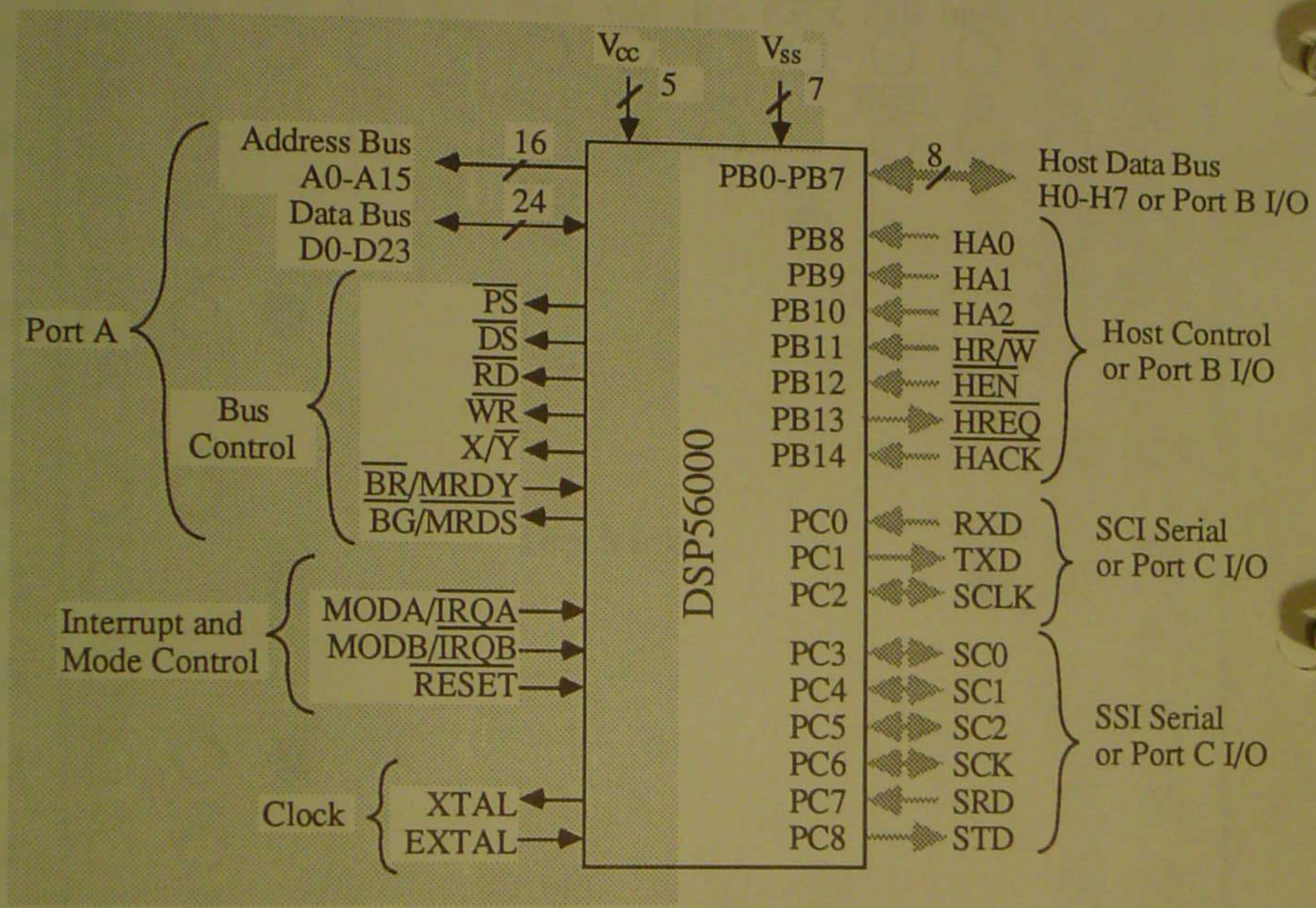



## PGA Sockets Available From

| Manufacturer | Part Number | Telephone | Comment |
|---|---|---|---|
| SAMTEC | MVAS - 120 - ZSTT - 13 | (812) 944-6733 | Includes center cutout |
| | CPAS - 88 - ZSTT - 13BF | | No cutout |
| Advanced Interconnections | 4CS088 - 01TG | (401) 823-5200 | Includes Center cutout |
| AMP | 55269 - 1 | | Zero Insertion Force |

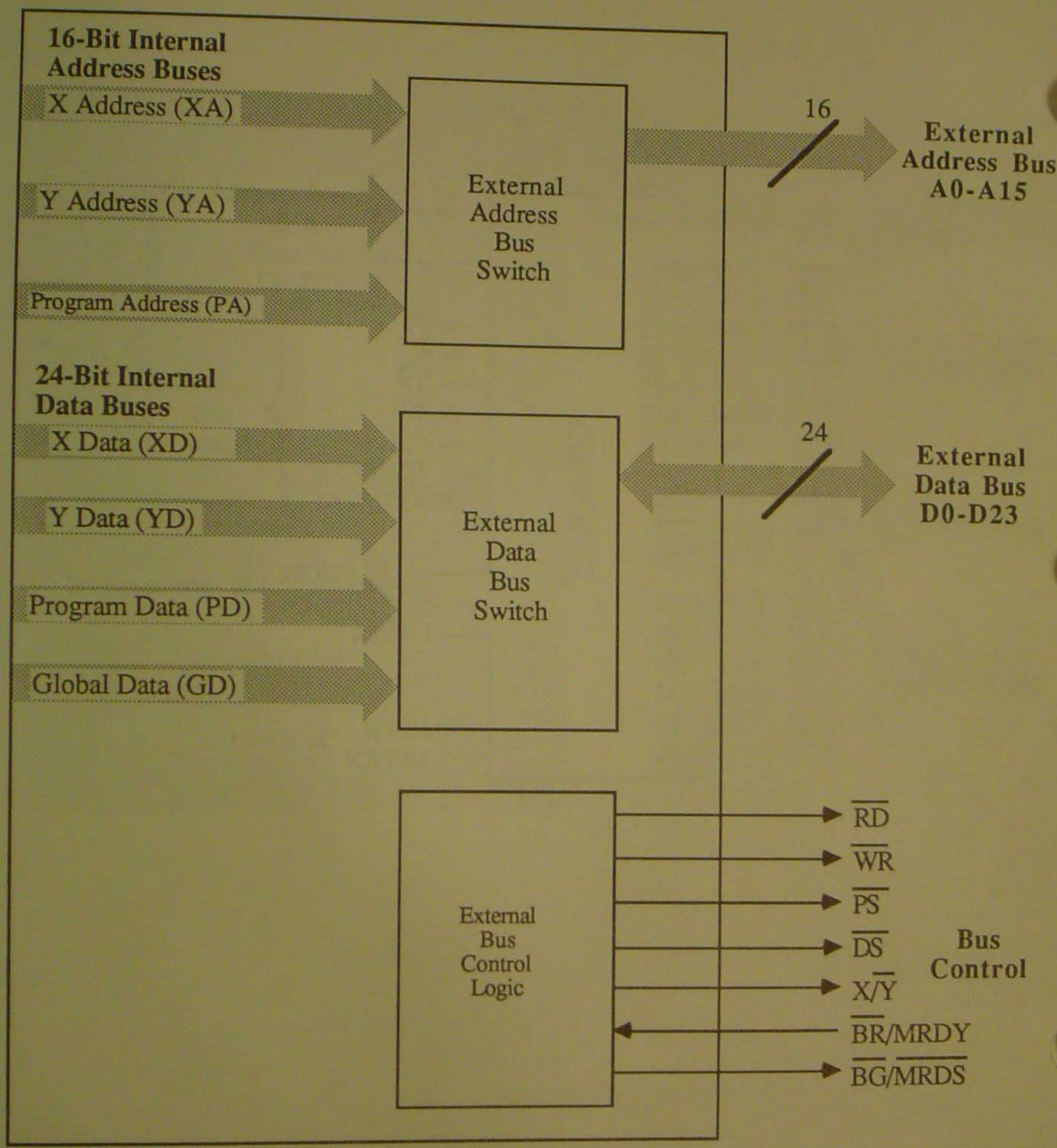
MTA5-3100

# DSP56000/1 FUNCTIONAL SIGNAL GROUPS



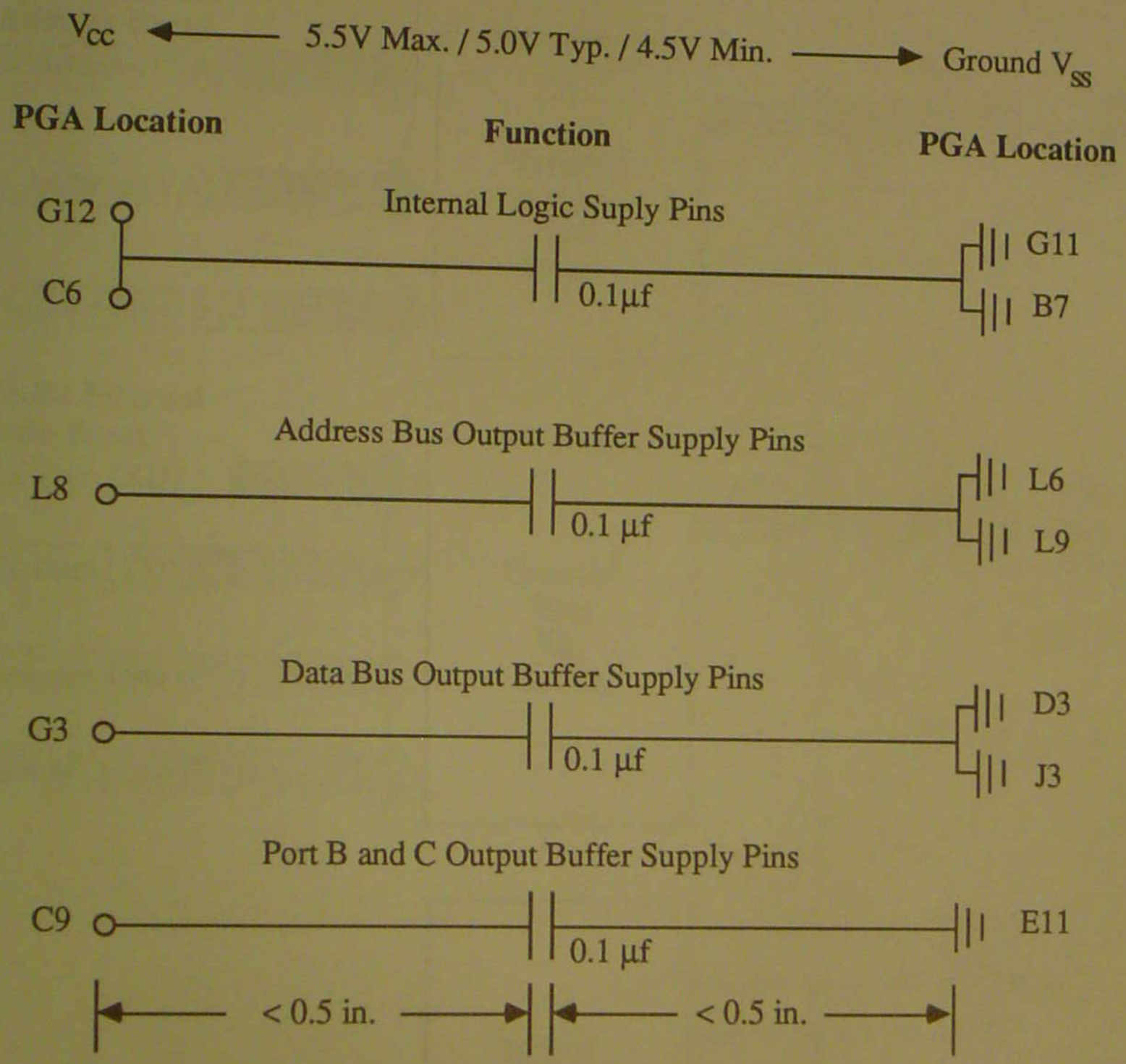

MTA5-3110

# PORTA-EXTERNAL BUS SWITCHES




MTA5-3120

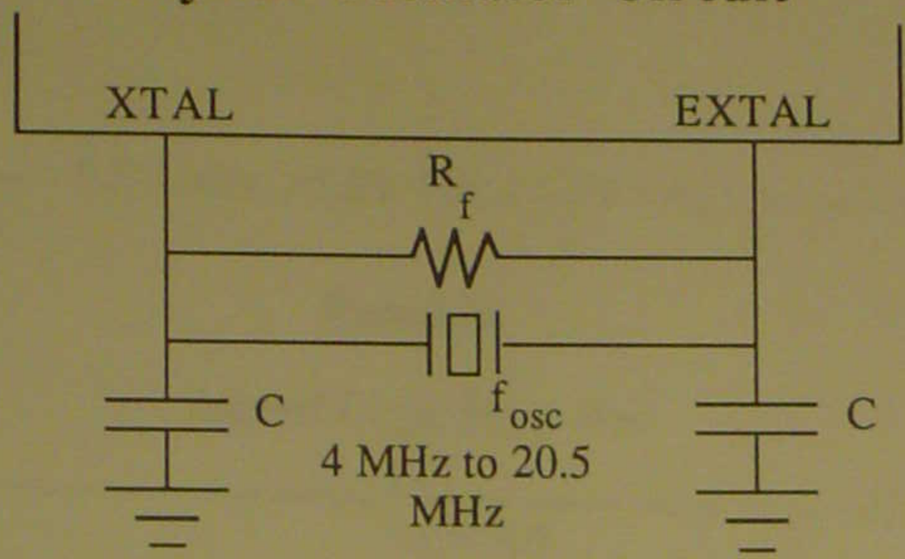
# POWER AND GROUND




MTA5-3130

# DSP56000/1 CLOCK

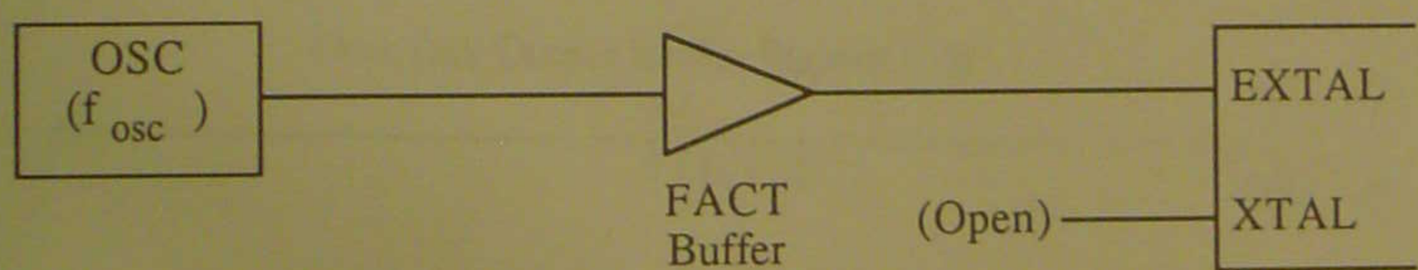
## Crystal Oscillator Circuit



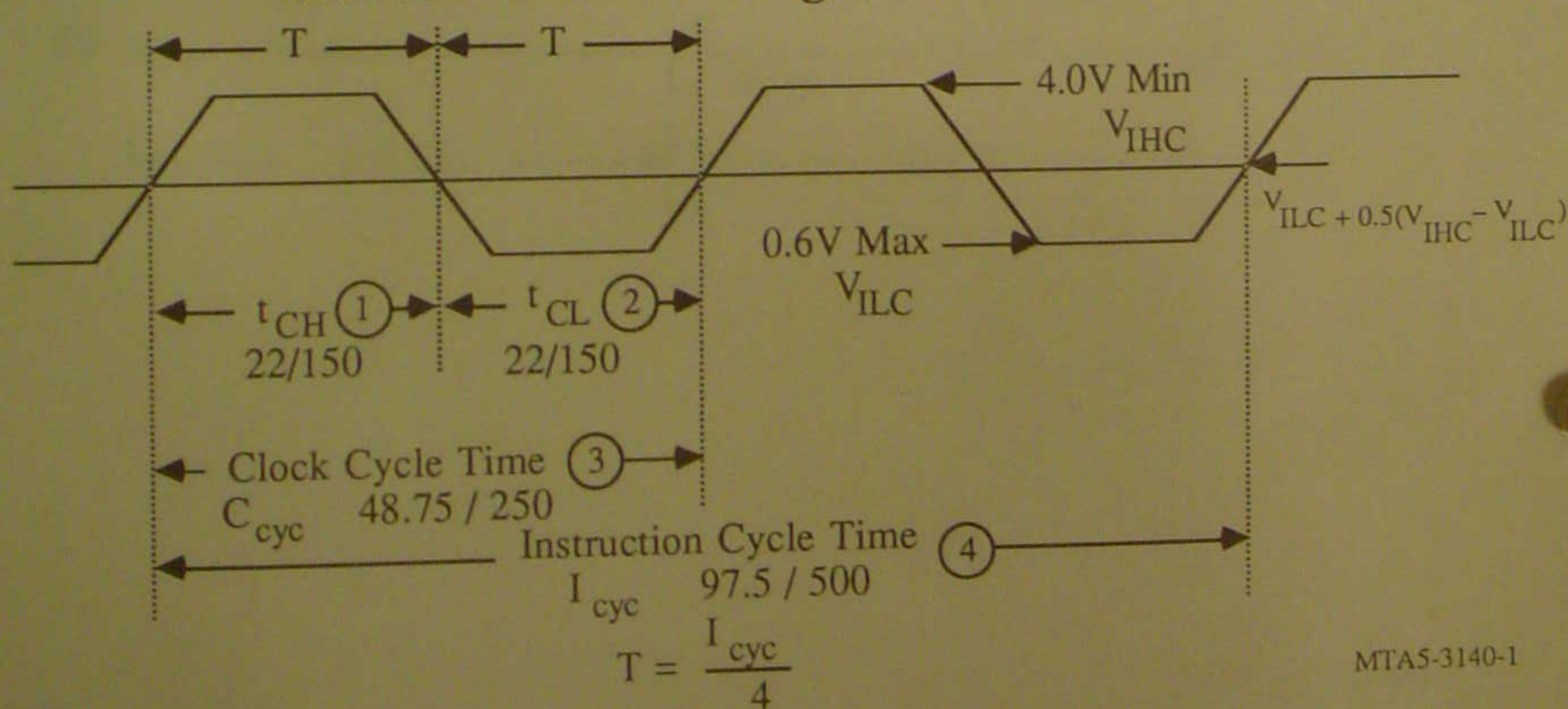

Recommended Component Values

| $f_{osc}$ (MHz) | C(pf) | $R_f$ (Kohms) |
|---|---|---|
| 4 | 82 ± 20% | 680 ± 10% |
| 20 | 47 ± 20% | 680 ± 10% |

## External Oscillator Circuit



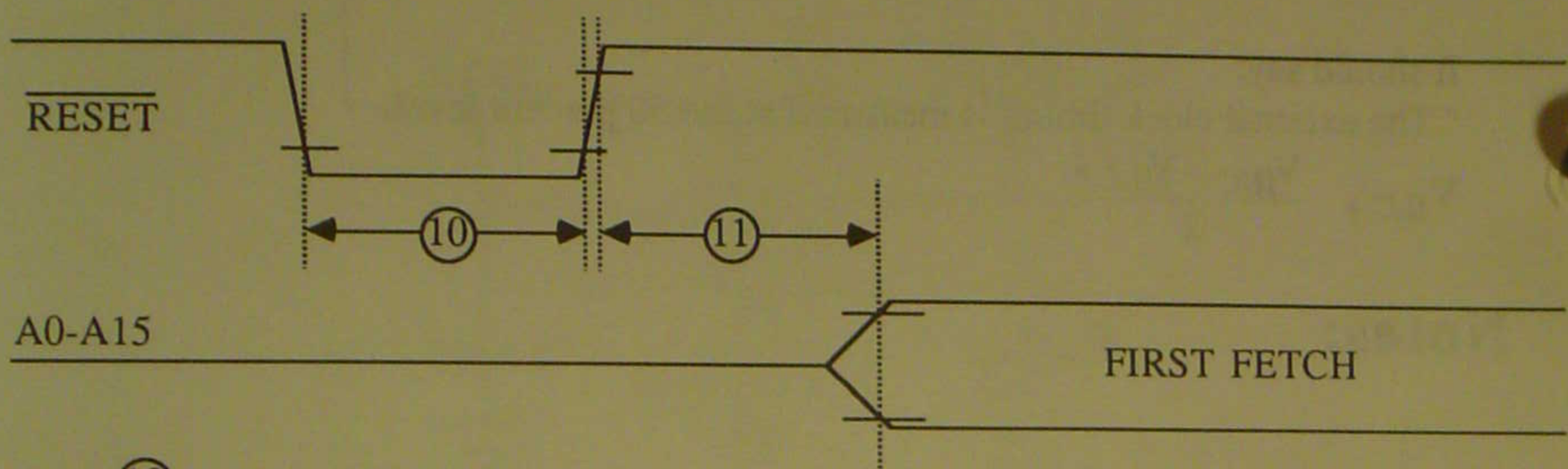

## EXTAL Clock Timing (Min/Max nS)




MTA5-3140-1

# RESET TIMING

## Asynchronous Reset Timing

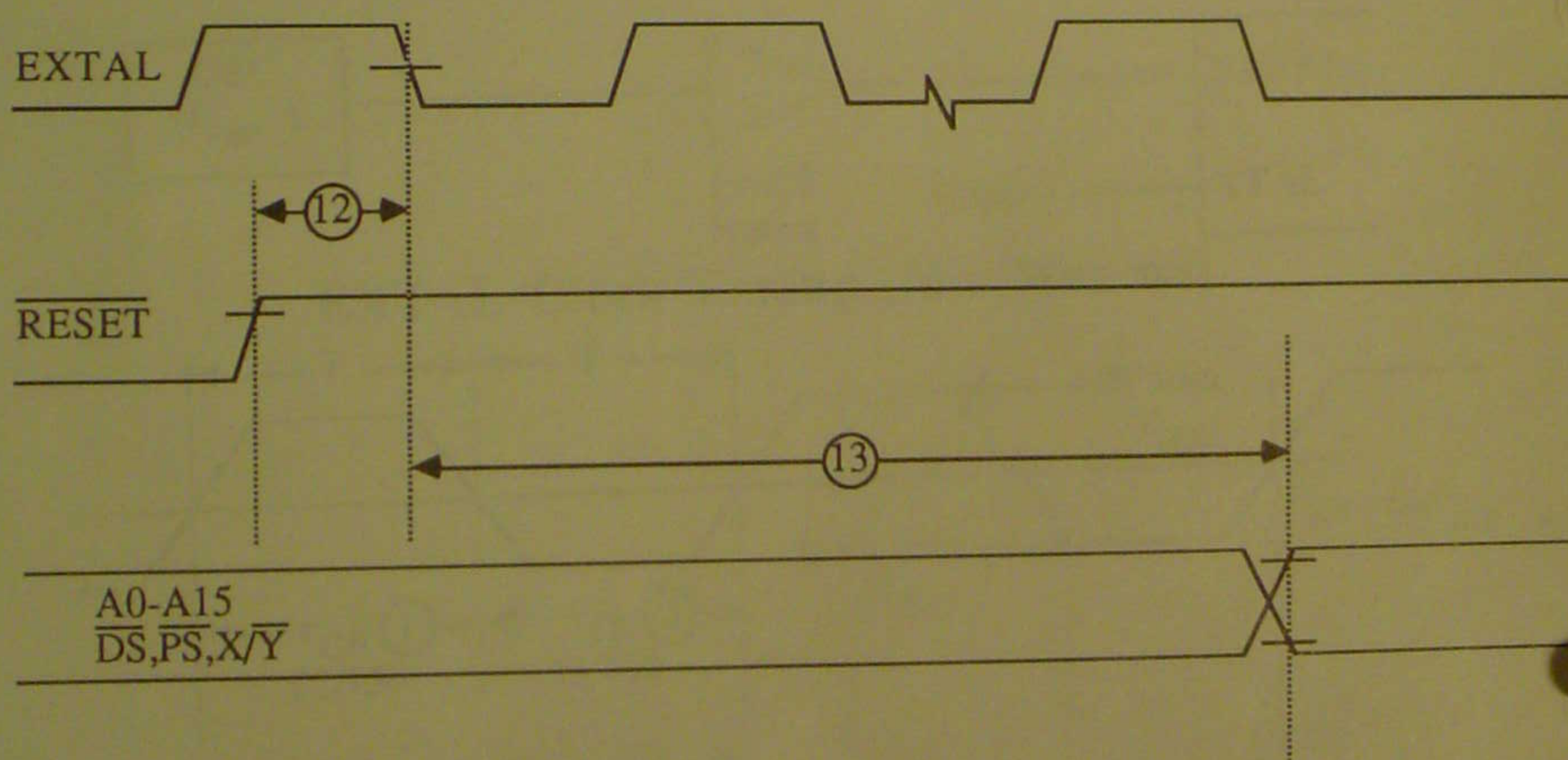



(10) $t_{smin}$ = 150,000T – Clock stabilization delay of on chip oscillator:

A) After power up
B) Exiting stop mode

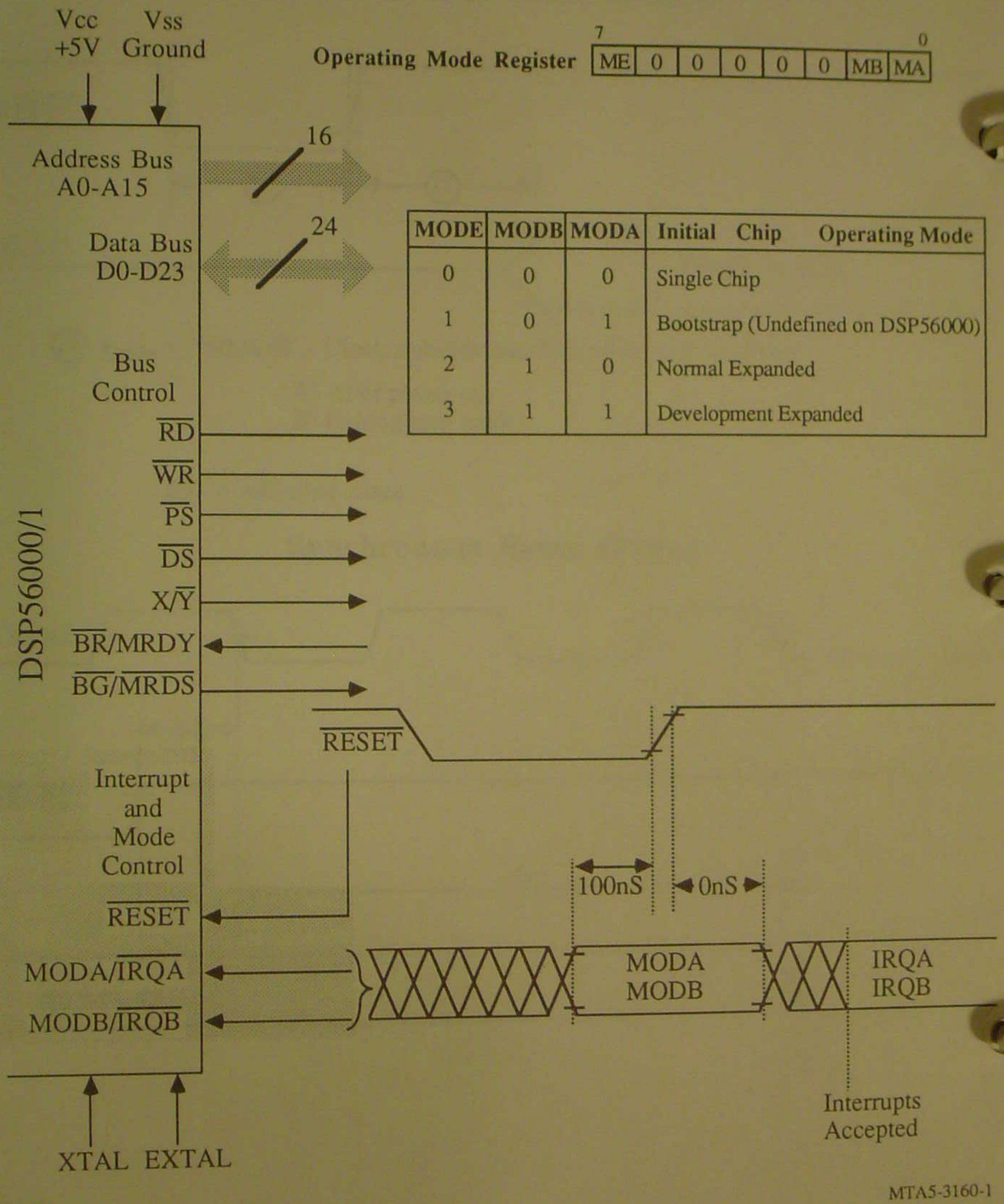
50T – All other cases

## Synchronous Reset Timing



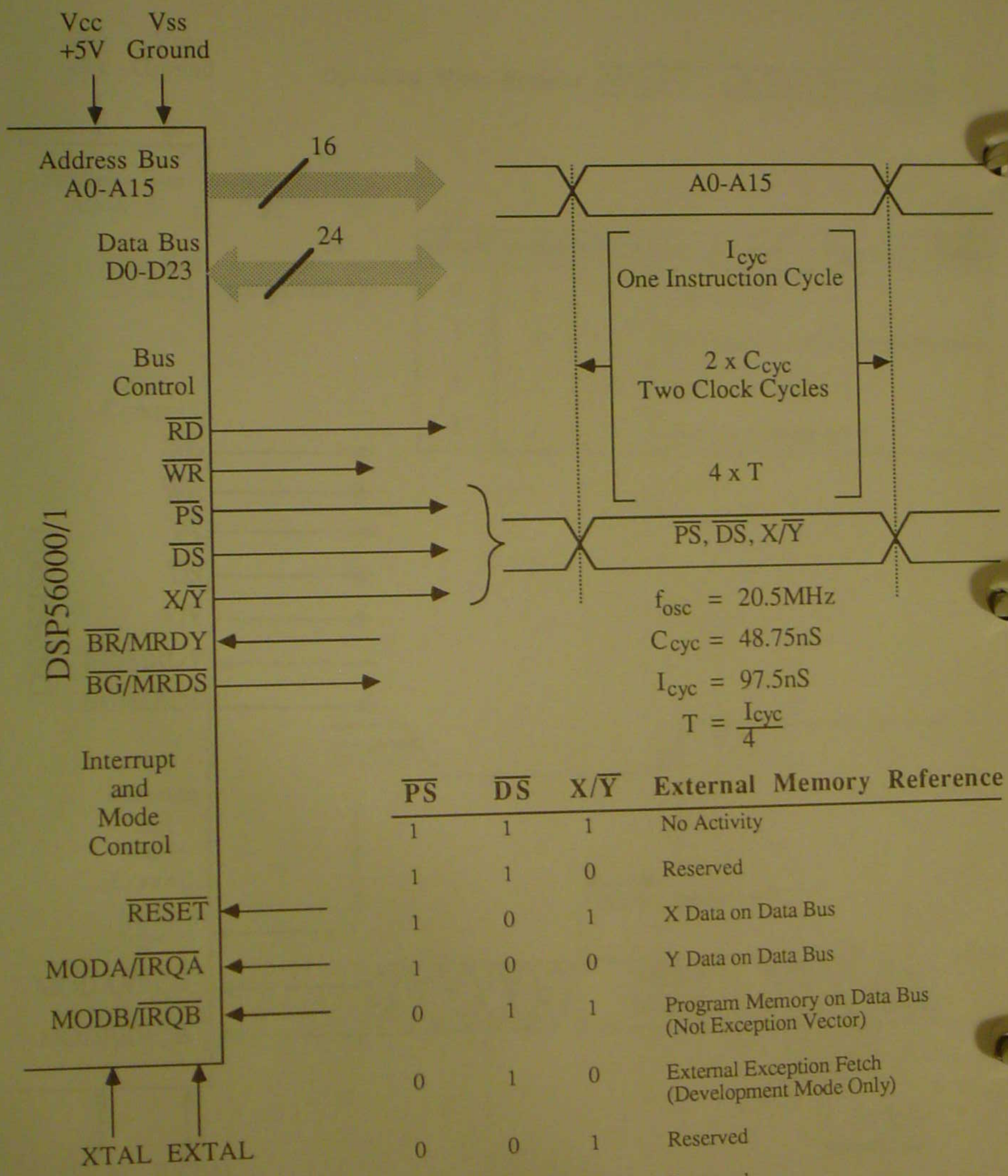

MTA5-3150-1

# OPERATING MODE SELECT TIMING




| MODE | MODB | MODA | Initial Chip Operating Mode |
|---|---|---|---|
| 0 | 0 | 0 | Single Chip |
| 1 | 0 | 1 | Bootstrap (Undefined on DSP56000) |
| 2 | 1 | 0 | Normal Expanded |
| 3 | 1 | 1 | Development Expanded |

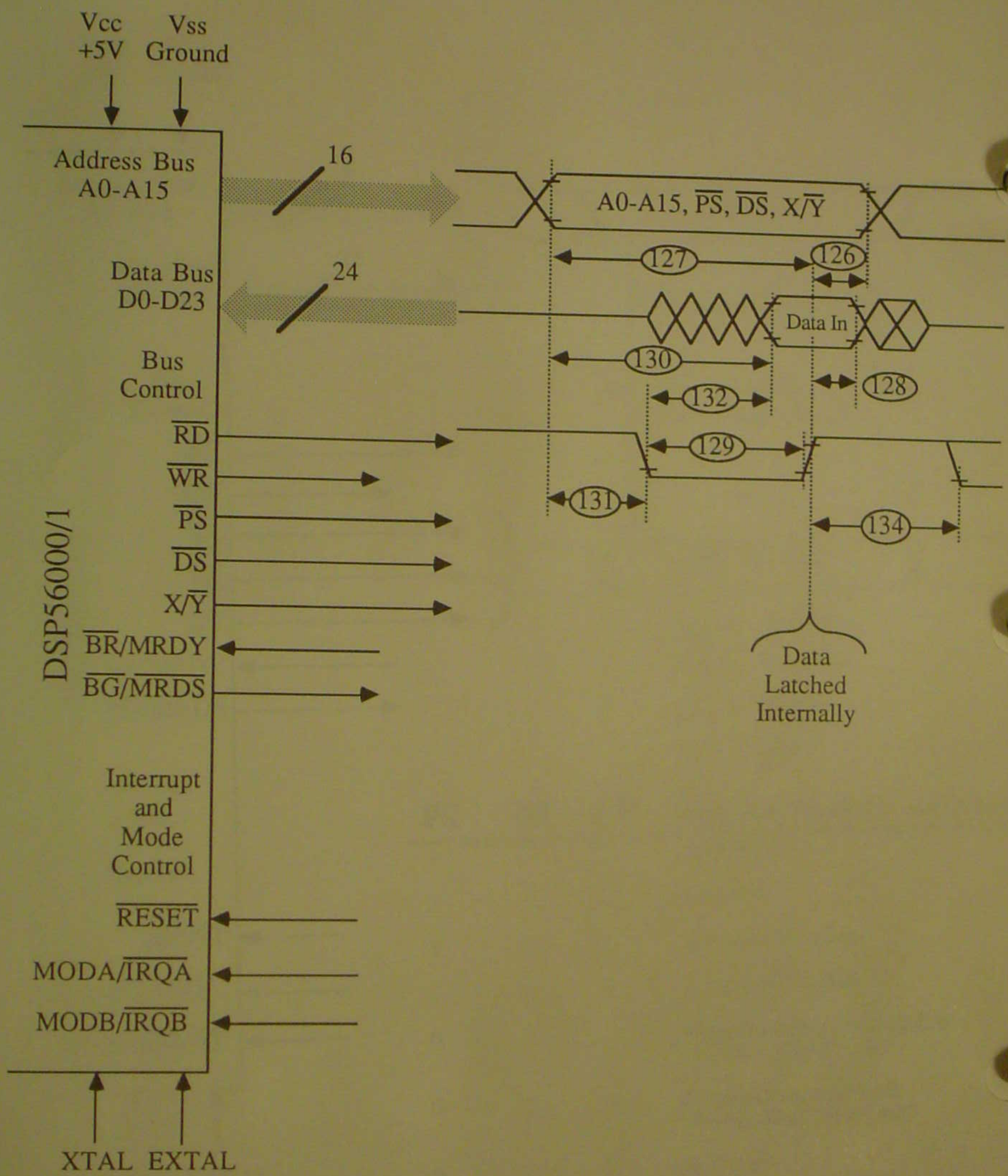
# INSTRUCTION CYCLE TIMING




| $\overline{PS}$ | $\overline{DS}$ | X/$\overline{Y}$ | External Memory Reference |
|---|---|---|---|
| 1 | 1 | 1 | No Activity |
| 1 | 1 | 0 | Reserved |
| 1 | 0 | 1 | X Data on Data Bus |
| 1 | 0 | 0 | Y Data on Data Bus |
| 0 | 1 | 1 | Program Memory on Data Bus (Not Exception Vector) |
| 0 | 1 | 0 | External Exception Fetch (Development Mode Only) |
| 0 | 0 | 1 | Reserved |
| 0 | 0 | 0 | Reserved |

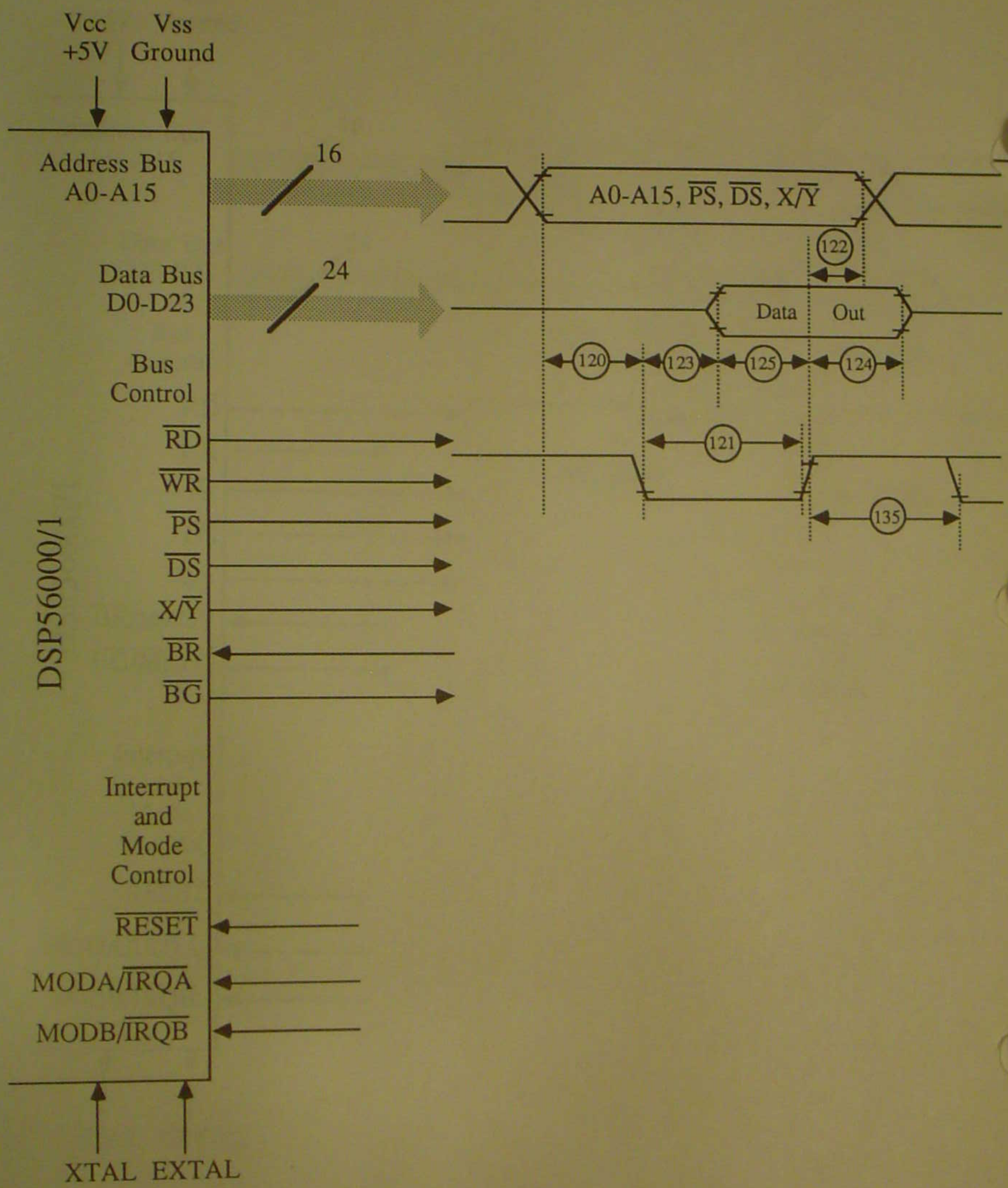
MTA5-3200-2

# READ DATA TIMING



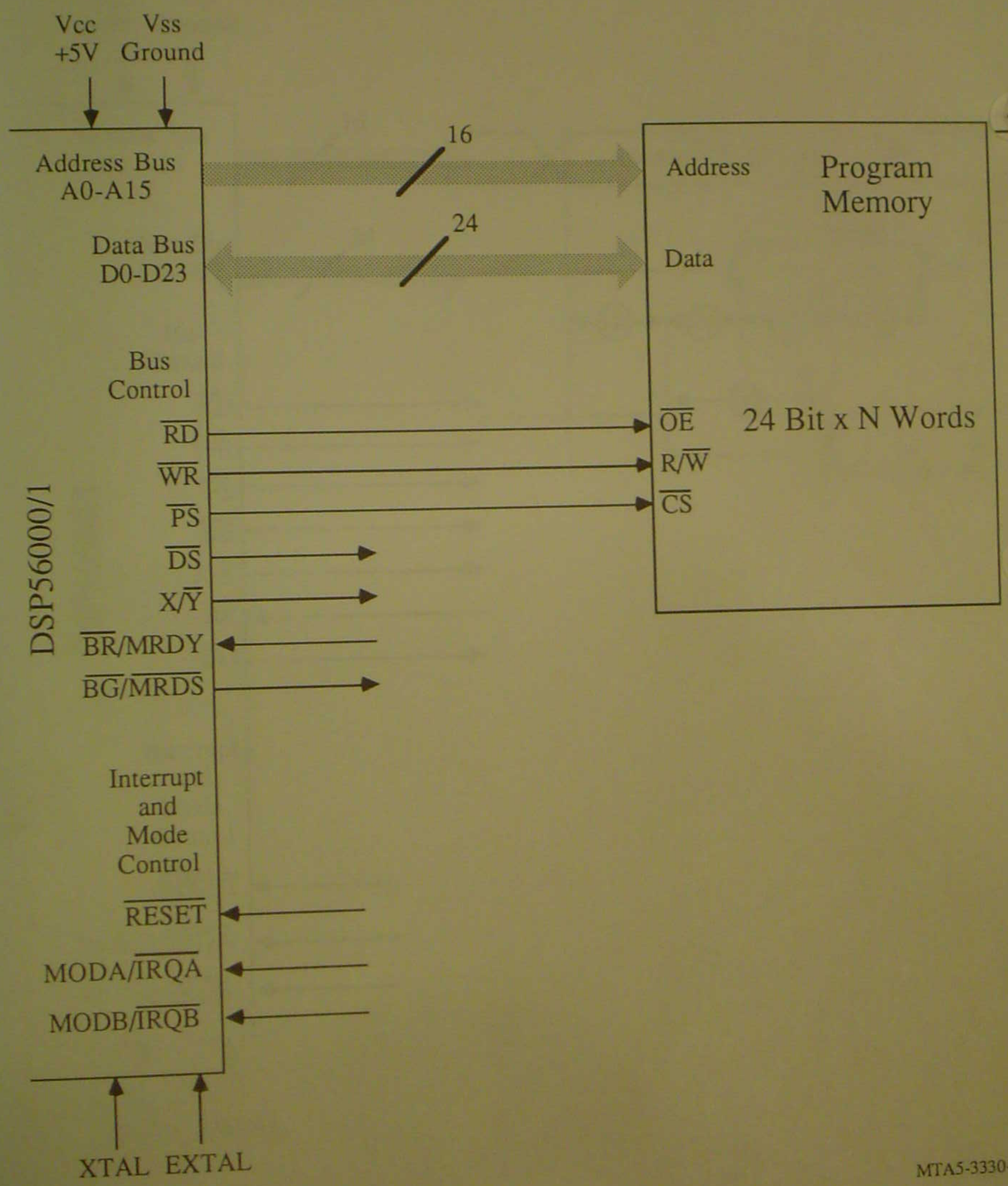

MTA5-3300-1

# WRITE DATA TIMING

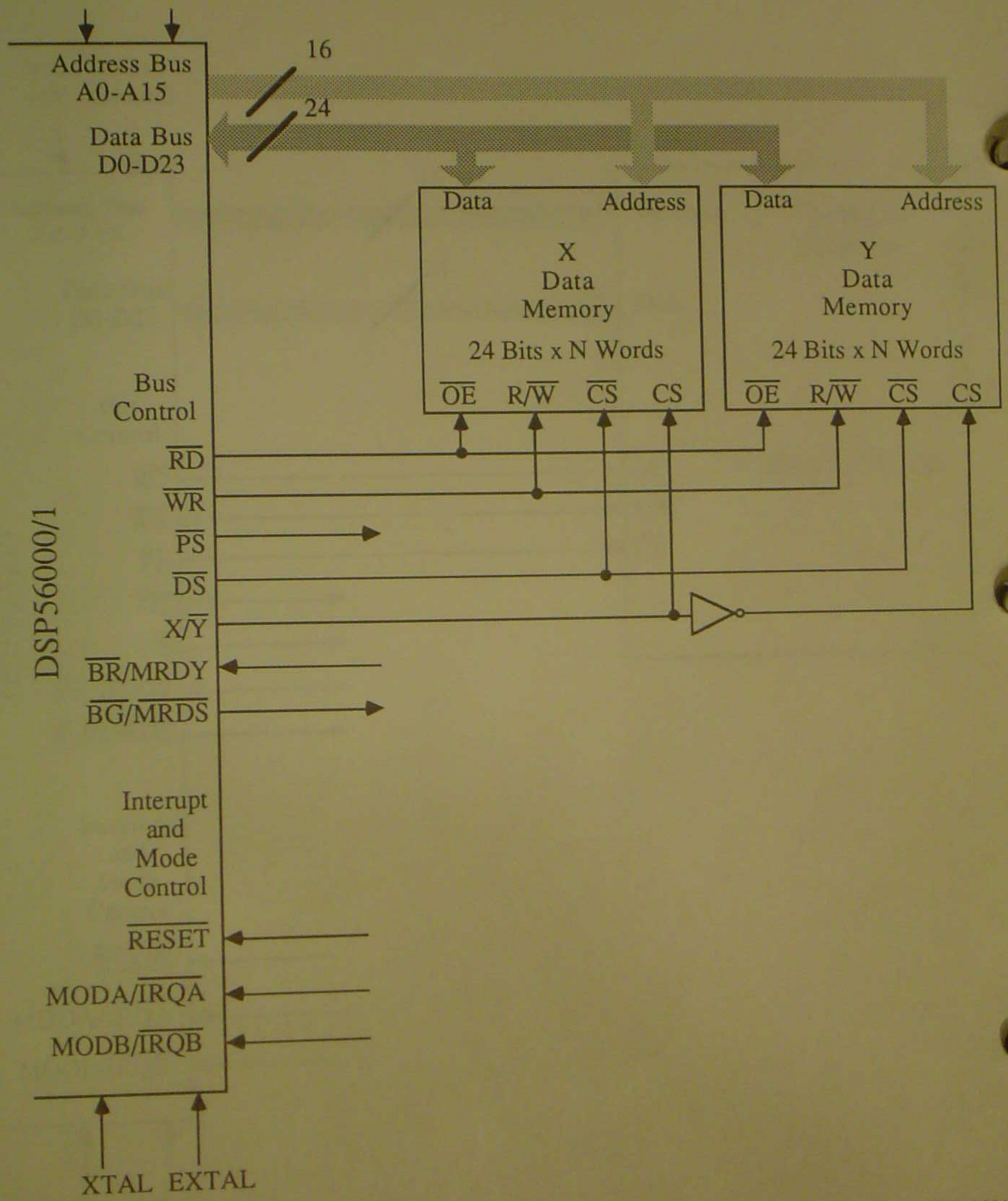



MTA5-3310-1

# EXTERNAL PROGRAM SPACE

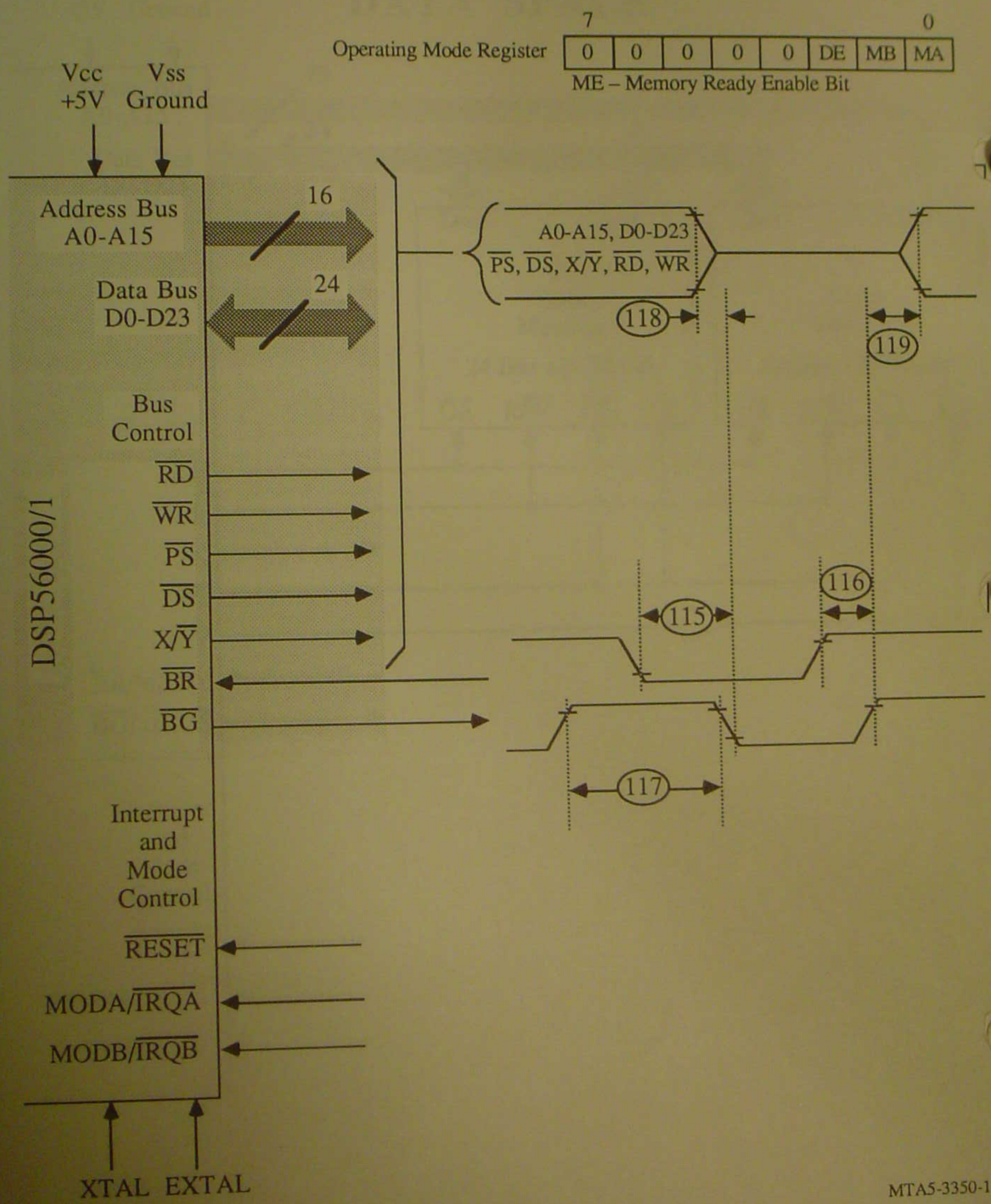


# EXTERNAL X AND Y DATA SPACE

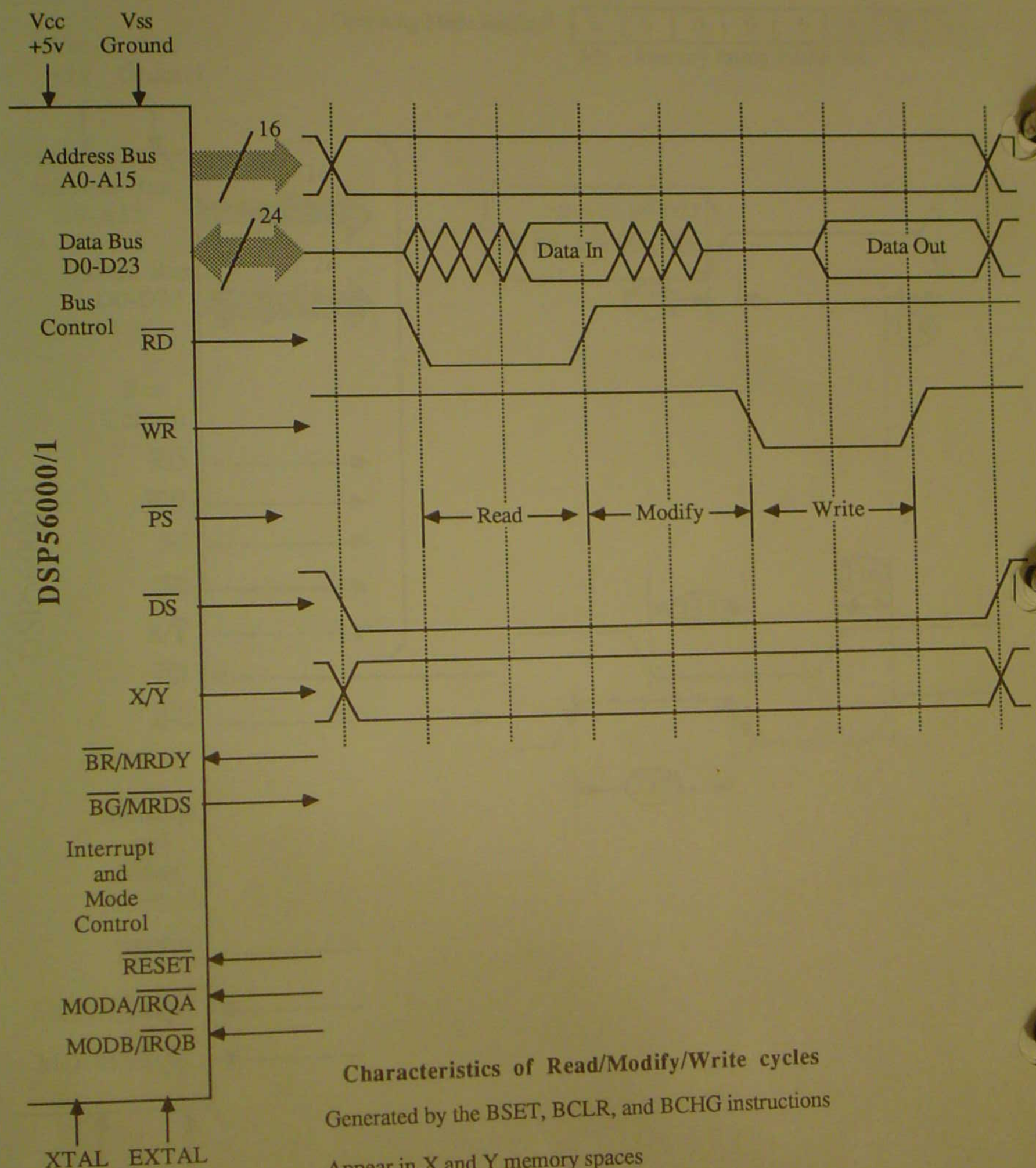



MTA5-3340-1

# BUS ARBITRATION TIMING



# READ / MODIFY / WRITE CYCLE




**Characteristics of Read/Modify/Write cycles**

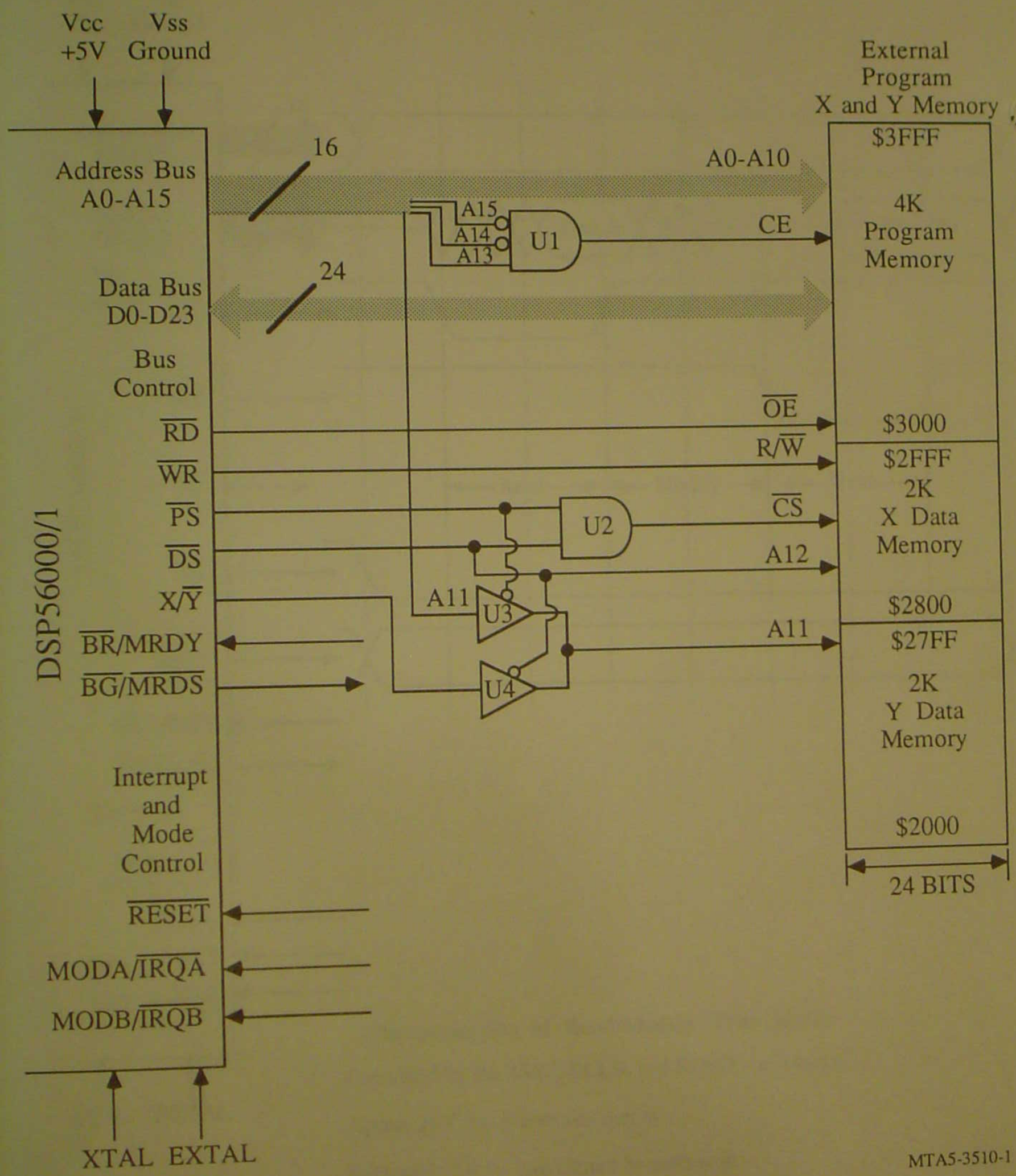
Generated by the BSET, BCLR, and BCHG instructions

Appear in X and Y memory spaces

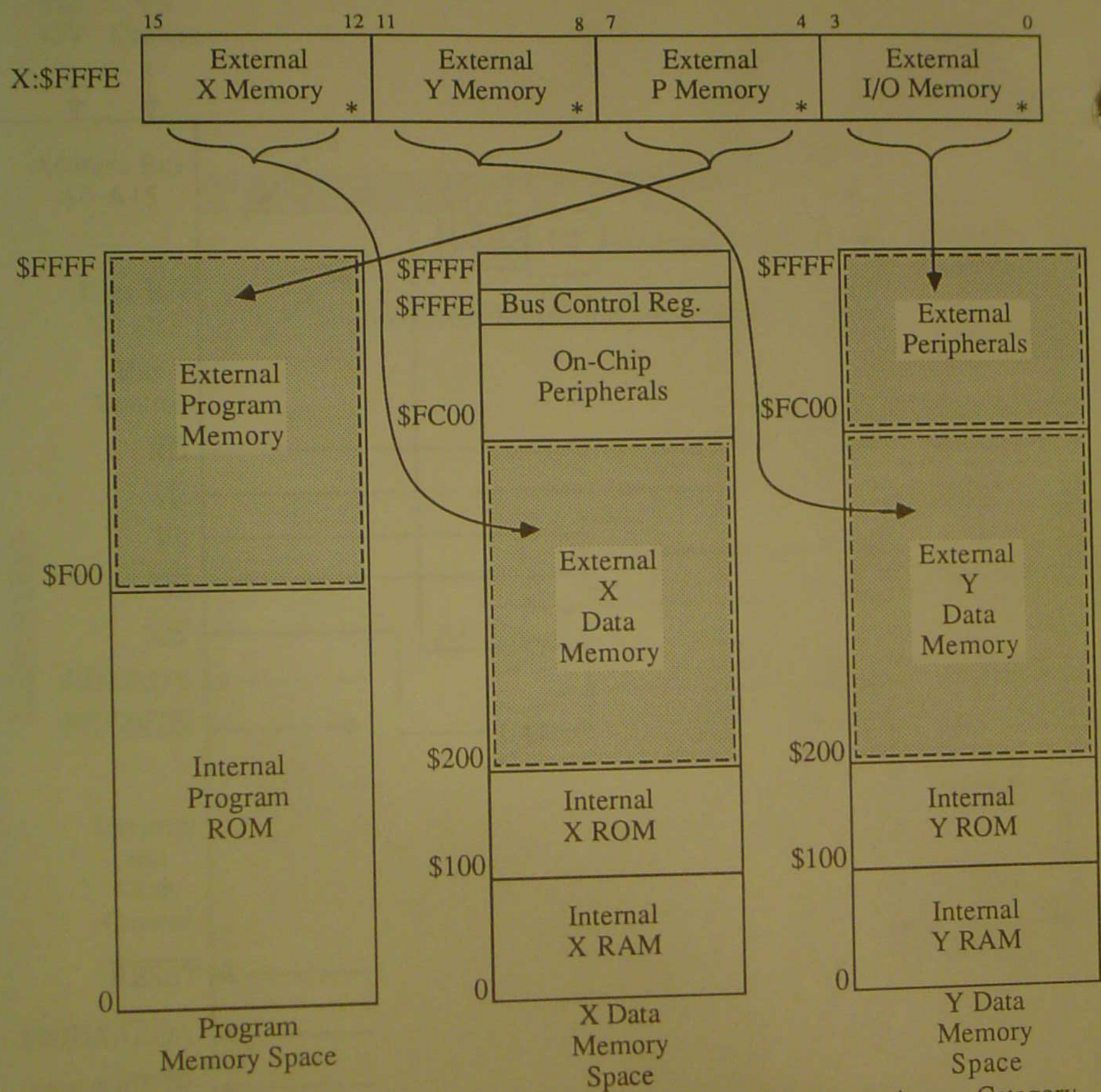
Indivisible Cycle. Bus cannot be arbitrated.

MTA5- 3360

# SINGLE CHIP EXTERNAL MEMORY EXAMPLE



# PORT A
# BUS CONTROL REGISTER (BCR)



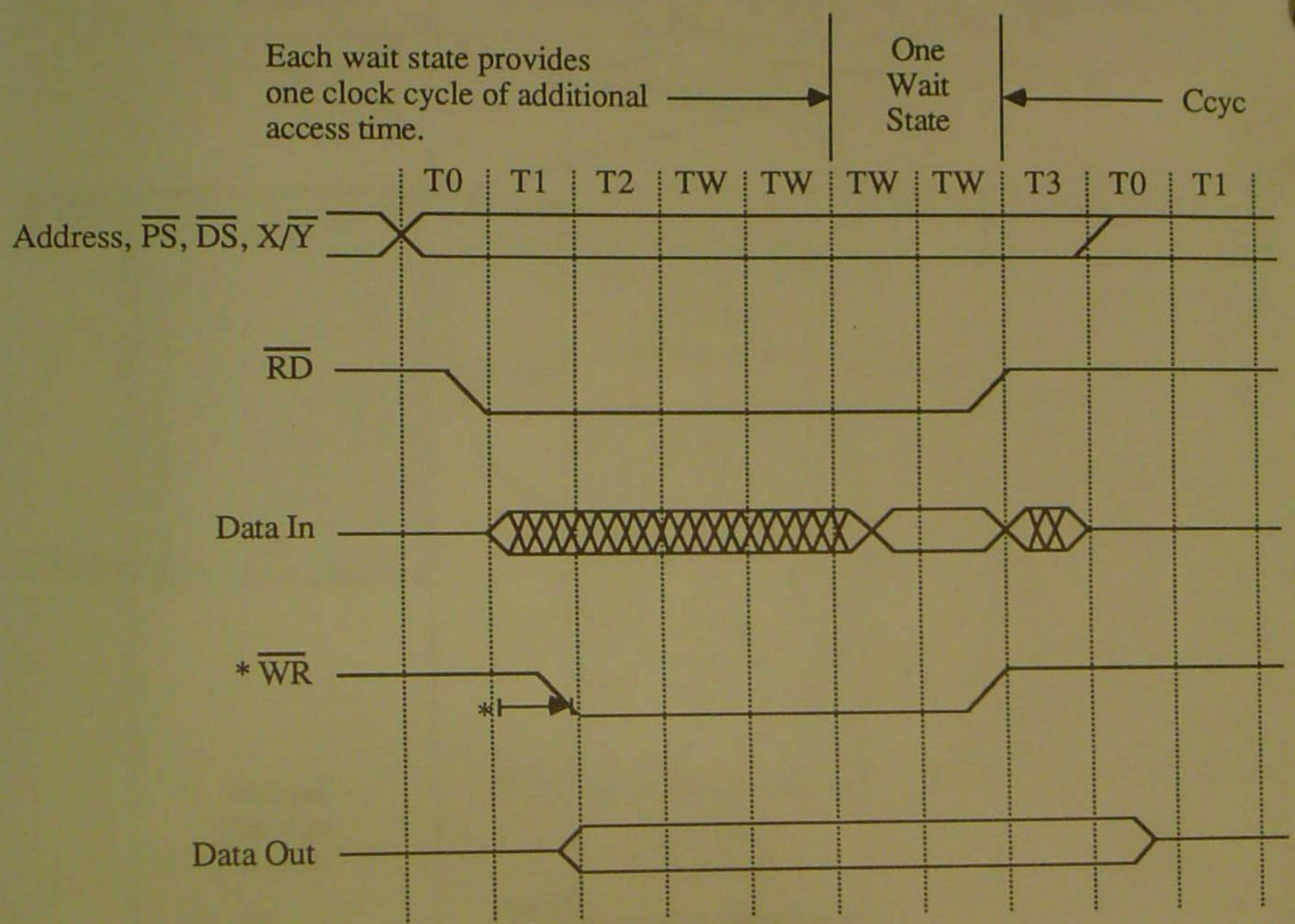

* 0 to 15 WAIT STATES can be inserted into each External Memory Access Category.

BCR is set to $FFFF on RESET.

1 WAIT STATE = Ccyc = Icyc / 2

MTA5-3520-1

# TWO WAIT STATE TIMING (BUS CONTROL REGISTER = 0010)



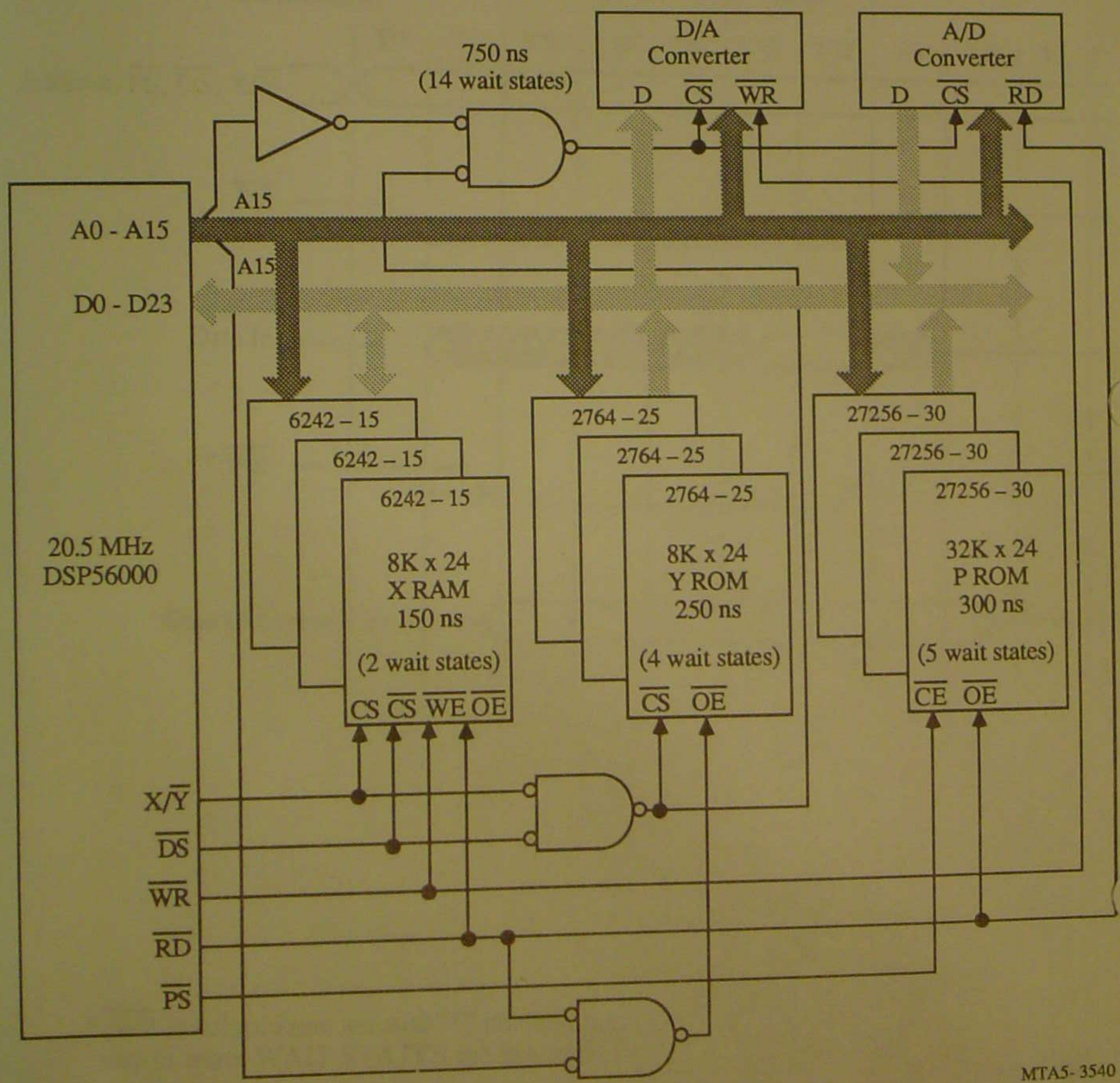

* $\overline{WR}$ is delayed one internal "T" phase when one or more WAIT STATES are inserted.

MTA5-3530

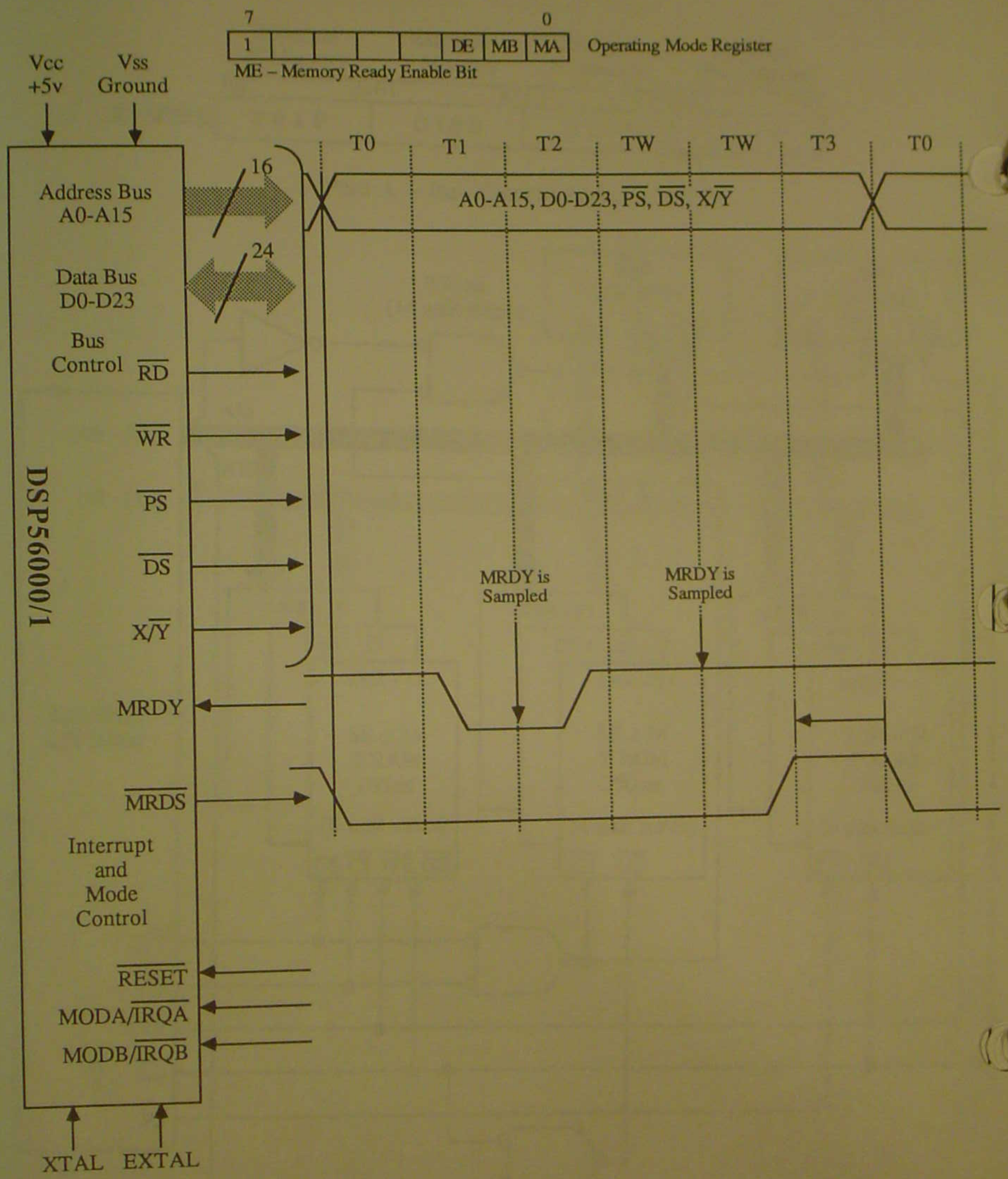
# MIXED SPEED EXPANDED SYSTEM

| | External X Memory (15–12) | External Y Memory (11–8) | External P Memory (7–4) | External I/O Memory (3–0) |
|---|---|---|---|---|
| X: $FFFE | 0 0 1 0 | 0 1 0 0 | 0 1 0 1 | 1 1 1 0 |

**Port A – Bus Control Register (BCR)**



# MEMORY READY

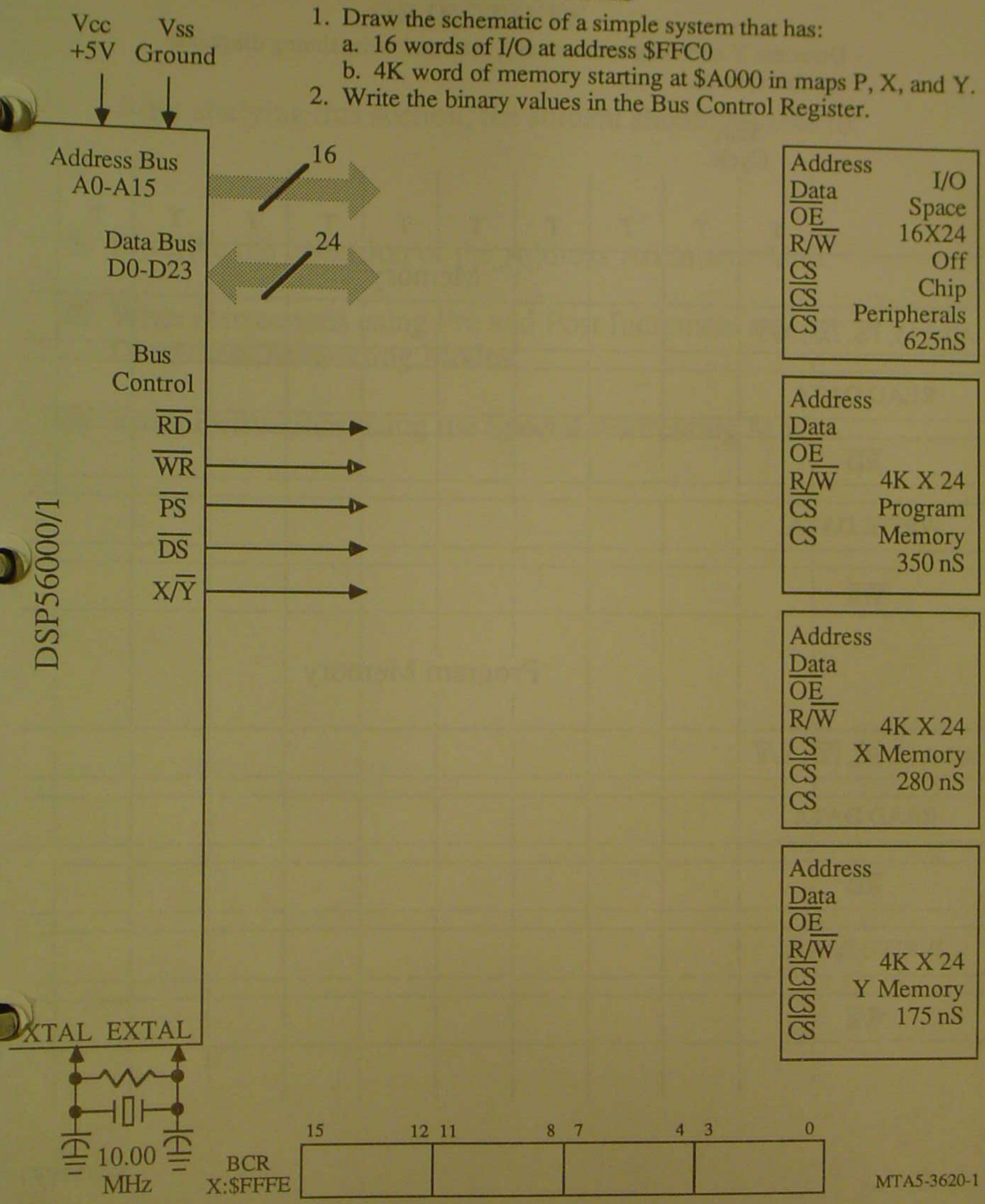



MTA5- 3610

# EXERCISE 5

## PINS AND BUSES

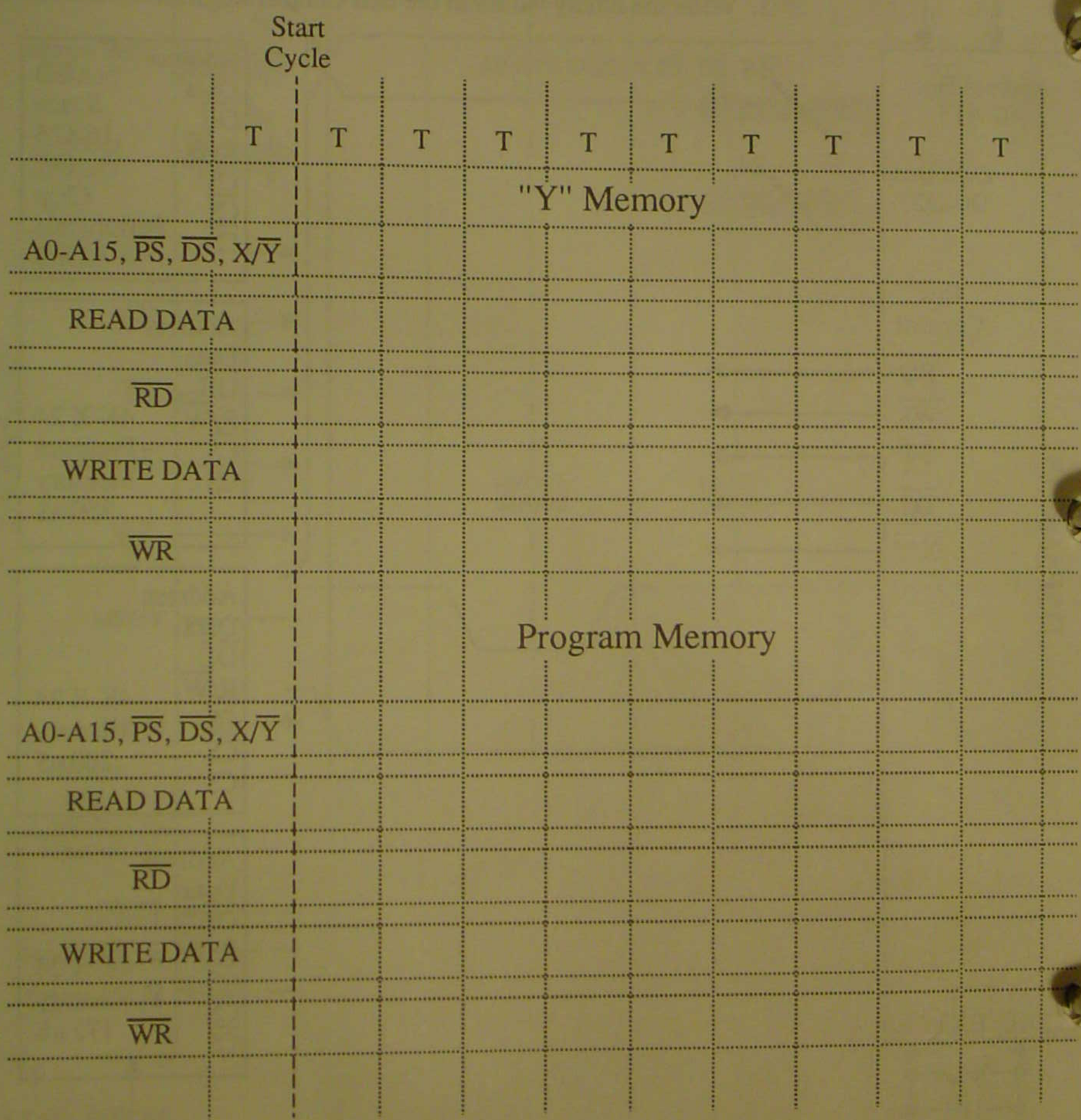
1. Draw the schematic of a simple system that has:
   a. 16 words of I/O at address $FFC0
   b. 4K word of memory starting at $A000 in maps P, X, and Y.
2. Write the binary values in the Bus Control Register.



# EXERCISE 5

## PINS AND BUSES (Cont.)

- Draw the Y and Program memory, read and write timing diagrams for this system.




MTA5-3630-1

# GENERAL ADDRESSING MODES

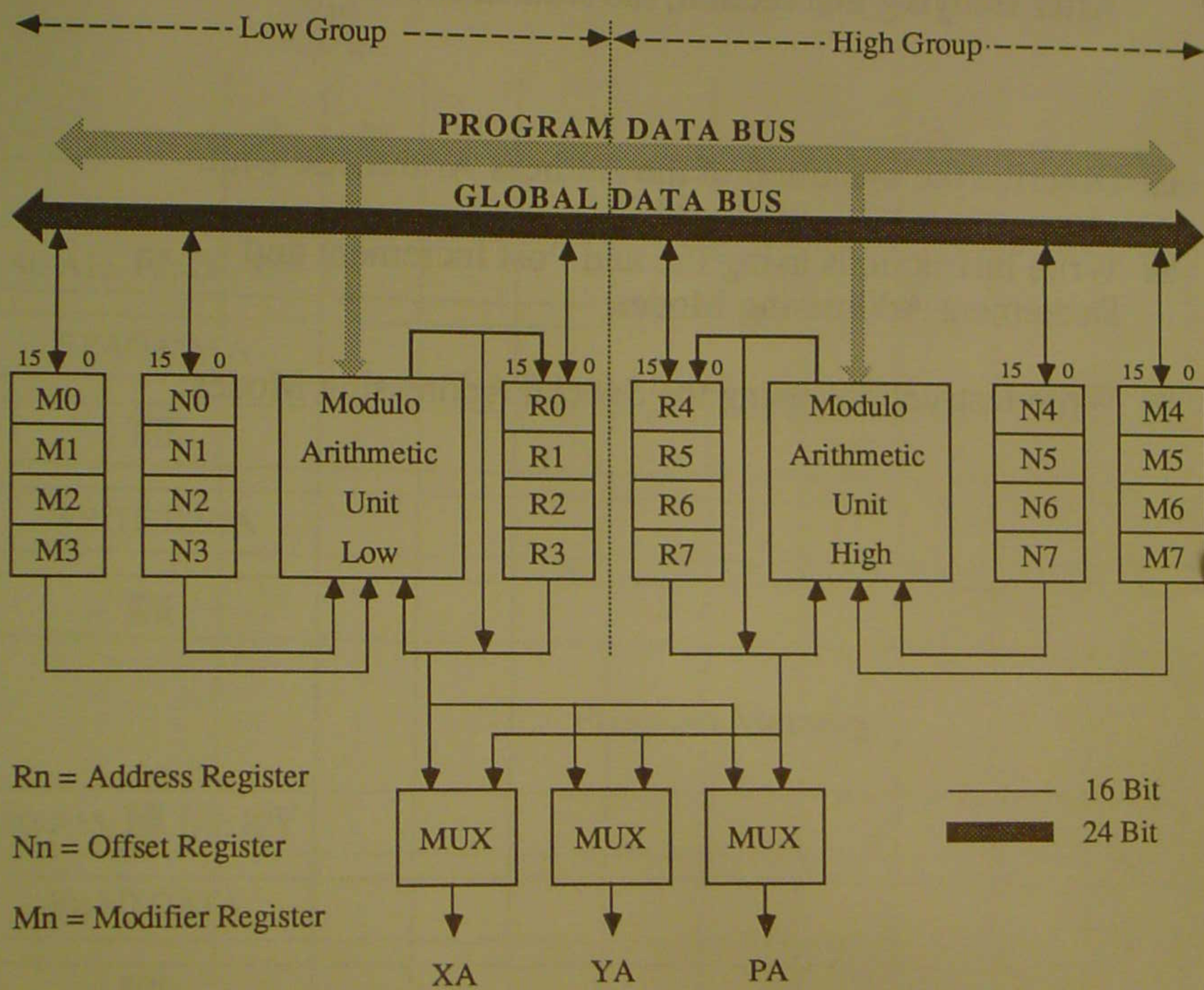
## OBJECTIVES

After studying this section, the student should be able to:

- Describe the operation of the Address Arithmetic Unit.
- Write instructions using Pre and Post Increment and Decrement Addressing Modes.
- Write instructions using the Special Addressing Modes.

# ADDRESS ARITHMETIC UNITS (AAU)

3




Rn = Address Register

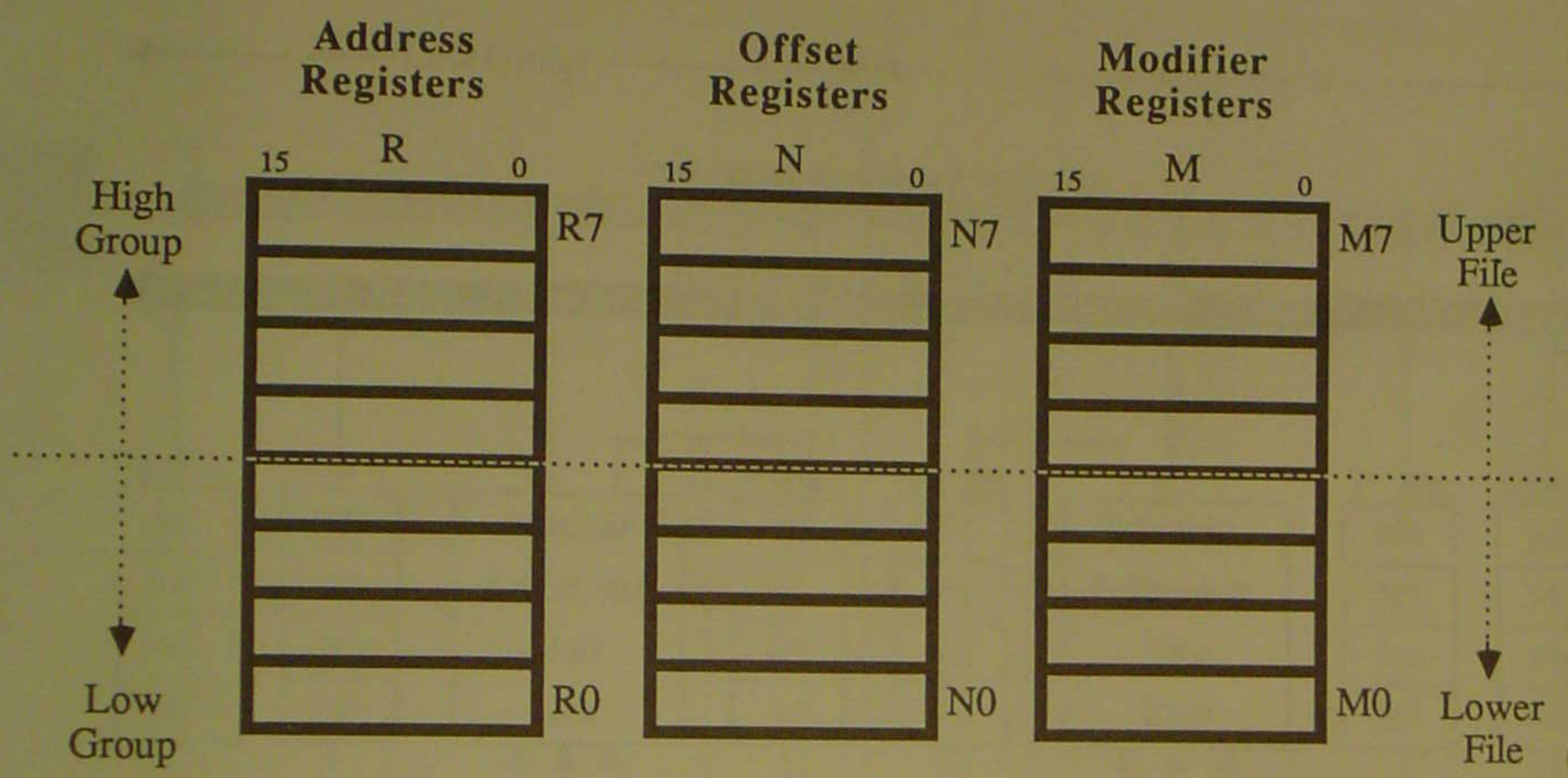
Nn = Offset Register

Mn = Modifier Register

- Generates up to two independent addresses each Instruction Cycle.
- Performs effective Address calculations.

MTA5-4100

# AAU PROGRAMMING MODEL




- Provides storage for 16 bit Addresses, Counters, and Scratch Pad Data
- ALU Registers work in groups of three having the same register number n.
- Address Register Rn, uses Offset Register Nn, and is modified by Register Mn.

MTA5-4110

# DSP56000/1 ADDRESSING MODES

| Addressing Mode | Uses Mn Modifier | Operand Reference S | C | D | A | P | X | Y | L | XY | Assembler Syntax |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Register** | | | | | | | | | | | Register Name |
| Data or Control Register | No | X | X | X | | | | | | | |
| Address Register Rn | No | | | | X | | | | | | Rn |
| Address Offset Register Nn | No | | | | X | | | | | | Nn |
| Address Modifier Register Mn | No | | | | X | | | | | | Mn |
| **Address Register Indirect** | | | | | | | | | | | |
| No Update | Yes | | | | | X | X | X | X | X | (Rn) |
| Postincrement by 1 | Yes | | | | | X | X | X | X | X | (Rn)+ |
| Postdecrement by 1 | Yes | | | | | X | X | X | X | X | (Rn)- |
| Postincrement by Offset Nn | Yes | | | | | X | X | X | X | X | (Rn)+Nn |
| Postdecrement by Offset Nn | Yes | | | | | X | X | X | X | | (Rn)-Nn |
| Indexed by Offset Nn | Yes | | | | | X | X | X | X | | (Rn+Nn) |
| Predecrement by 1 | Yes | | | | | X | X | X | X | | -(Rn) |
| **Special** | | | | | | | | | | | |
| Immediate Data (24 Bits) | No | | | | | X | | | | | #xxxxxx |
| Absolute Address (16 Bits) | No | | | | | X | X | X | X | | xxxx |
| Immediate Short Data (8 Bits) | No | | | | | X | | | | | #xx |
| Short Jump Address (12 Bits) | No | | | | | X | | | | | xxx |
| Absolute Short Address (6 Bits, zero extended) | No | | | | | X | X | X | X | | aa |
| I/O Short Address (6 Bits, ones extended) | No | | | | | | X | X | | | pp |
| Implicit | No | X | X | | | X | | | | | |

for S = System Stack Reference
C = Program Controller Register Reference
D = Data ALU Register Reference
A = Address ALU Register Reference
P = Program Memory Reference
X = X Memory Reference
Y = Y Memory Reference
L = L Memory Reference
XY = XY Memory Reference

MTA5-4120-1

# BASIC MOVE INSTRUCTION

**MOVE Source, Destination**

| Source → | Destination |
|---|---|
| Register | Register |
| Register | Memory |
| Memory | Register |

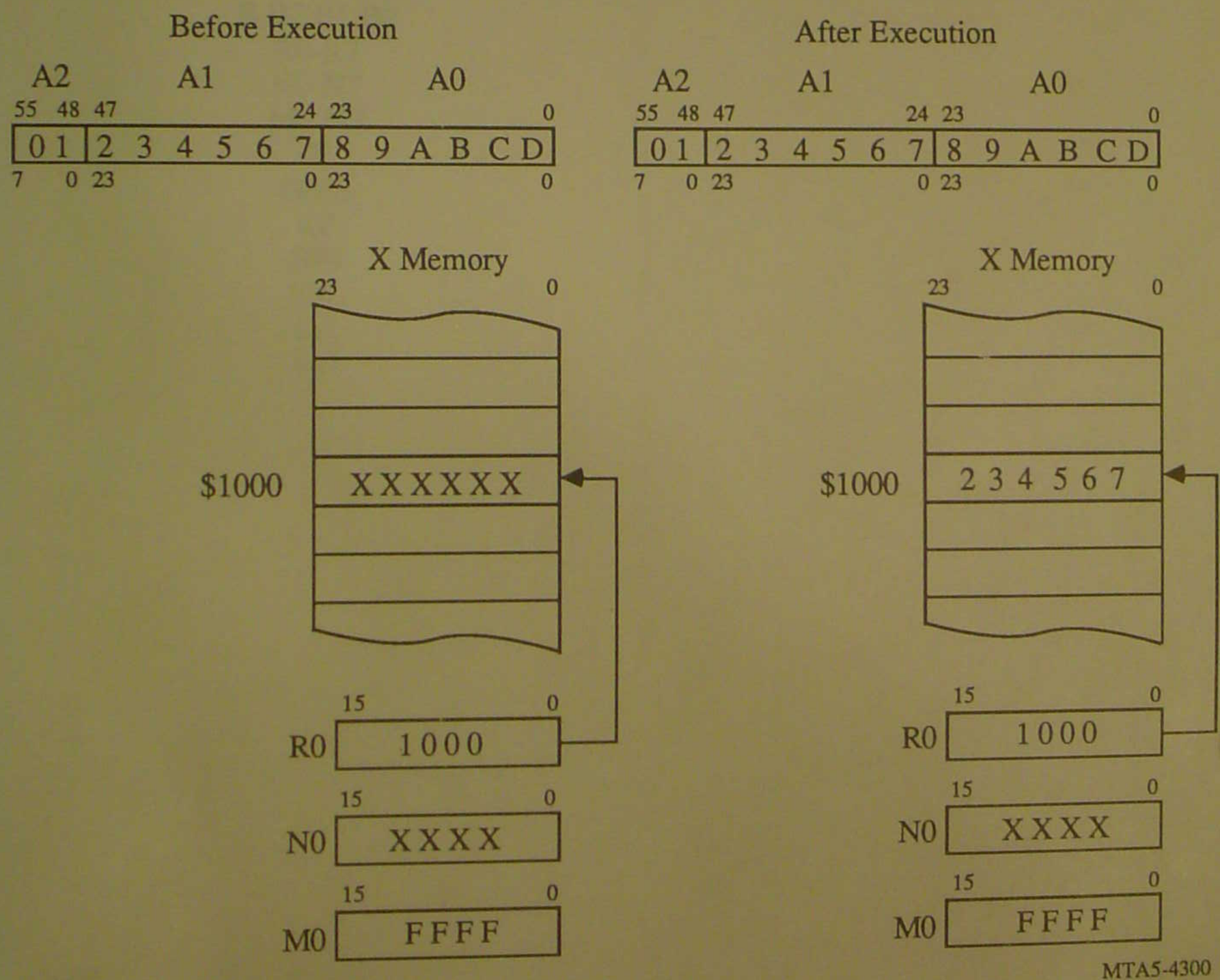
| Registers | Memory Space |
|---|---|
| X1,X0 | Program |
| Y1,Y0 | X |
| A,A2,A1,A0 | Y |
| B,B2,B1,B0 | I/O |
| R0-R7 | |
| N0-N7 | |
| M0-M7 | |
| SR | |
| OMR | |
| SP | |
| SSH | |
| SSL | |
| LA | |
| LC | |

MTA5-4130

# ADDRESS REGISTER INDIRECT NO UPDATE

- Assembler Syntax: (Rn)
- Operands referenced: P, X, Y, XY, L memories.
- Additional instruction execution time (clocks): 0
- Additional effective address words: 0

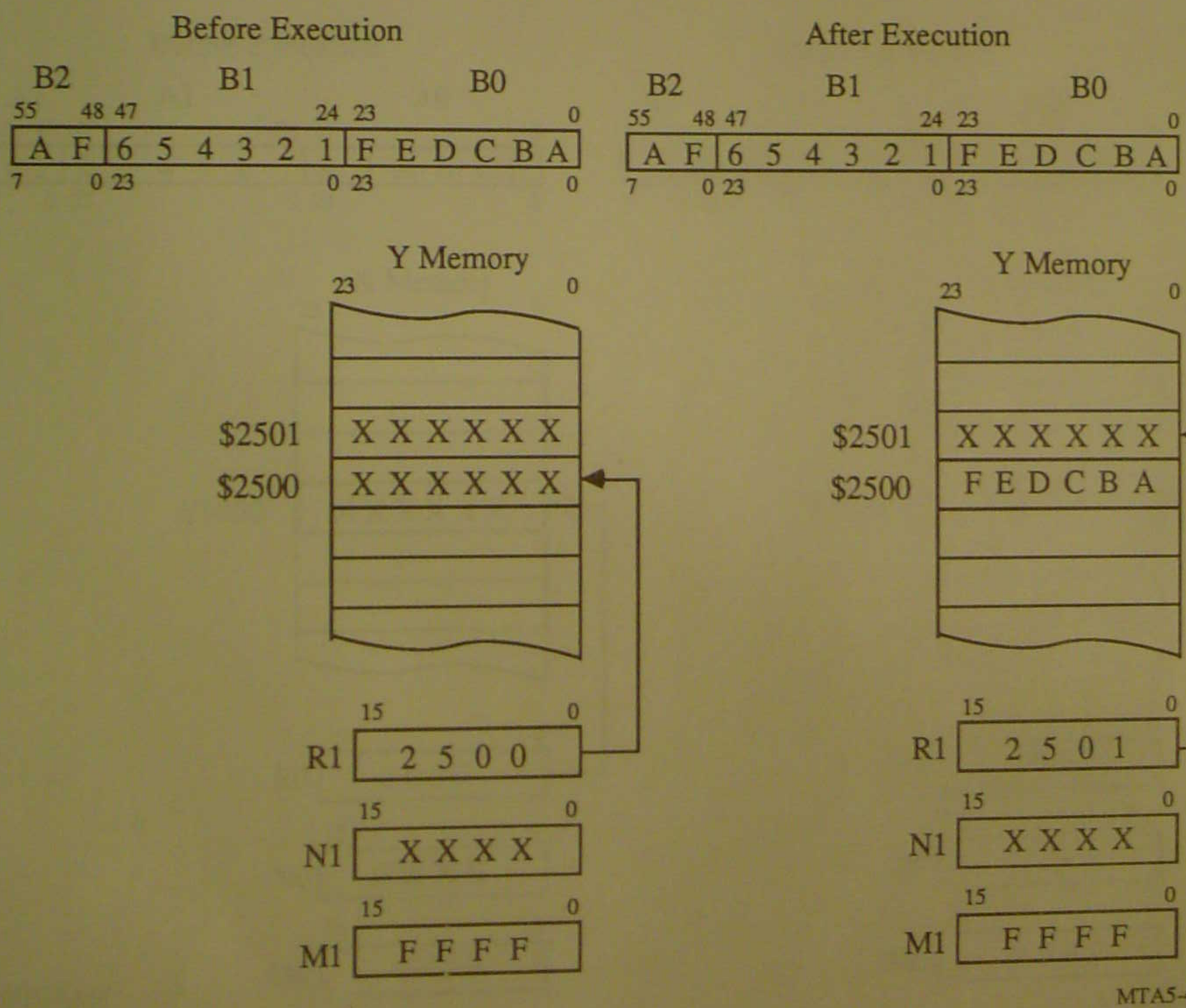
**Example:** MOVE A1, X:(R0)



# ADDRESS REGISTER INDIRECT POST INCREMENT BY ONE

- Assembler Syntax: (Rn) +
- Operands referenced: P, X, Y, XY, L memories.
- Additional instruction execution time (clocks): 0
- Additional effective address words: 0

**Example:** MOVE B0,Y:(R1) +

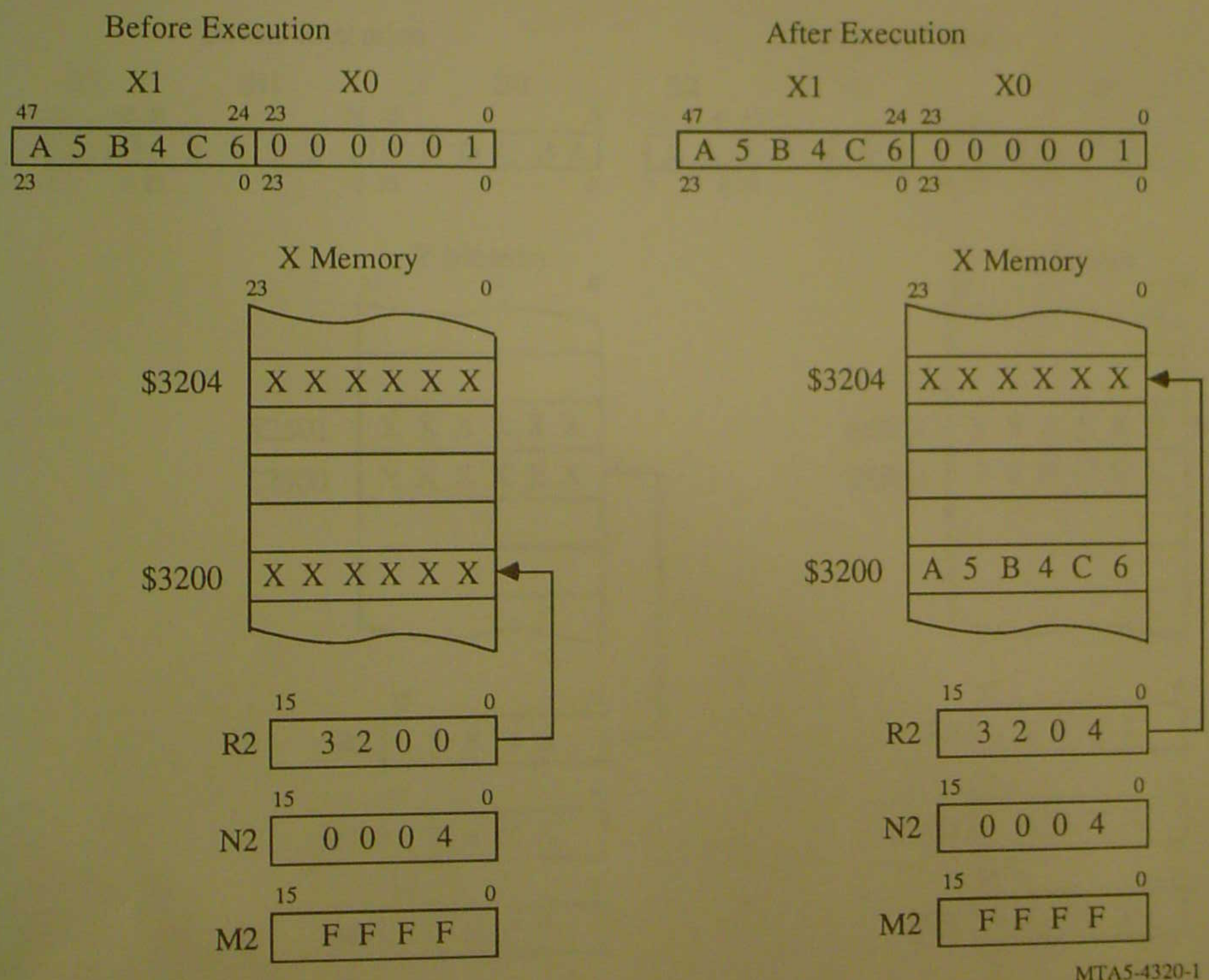



MTA5-4310

# ADDRESS REGISTER INDIRECT POSTINCREMENT BY OFFSET Nn

- Assembler Syntax: (Rn) + Nn
- Operands referenced: P, X, Y, XY, L memories.
- Additional instruction execution time (clocks): 0
- Additional effective address words: 0

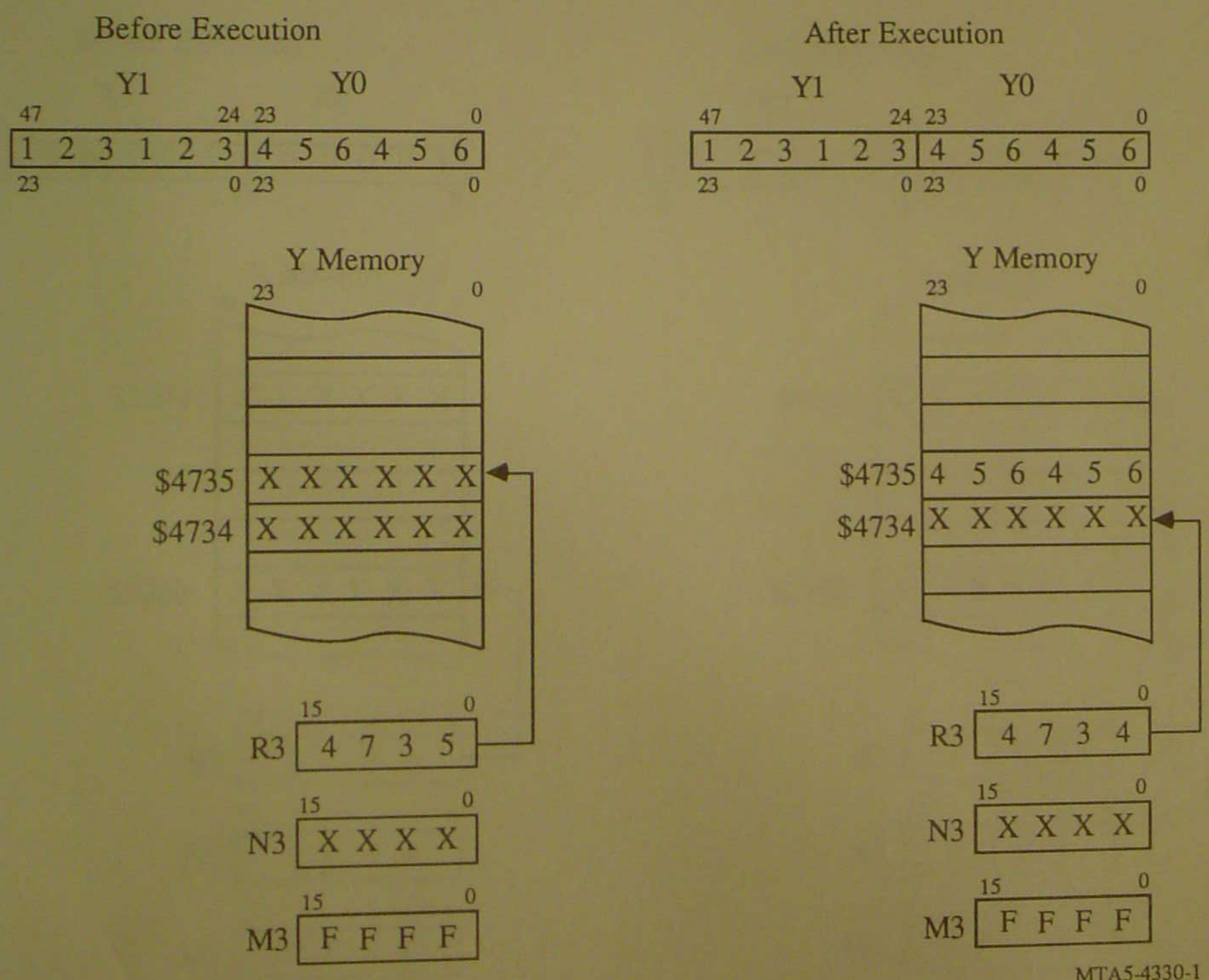
**Example:** MOVE X1, X:(R2) + N2



# ADDRESS REGISTER INDIRECT POSTDECREMENT BY ONE

- Assembler Syntax: (Rn) -
- Operands referenced: P, X, Y, XY, L memories.
- Additional instruction execution time (clocks): 0
- Additional effective address words: 0

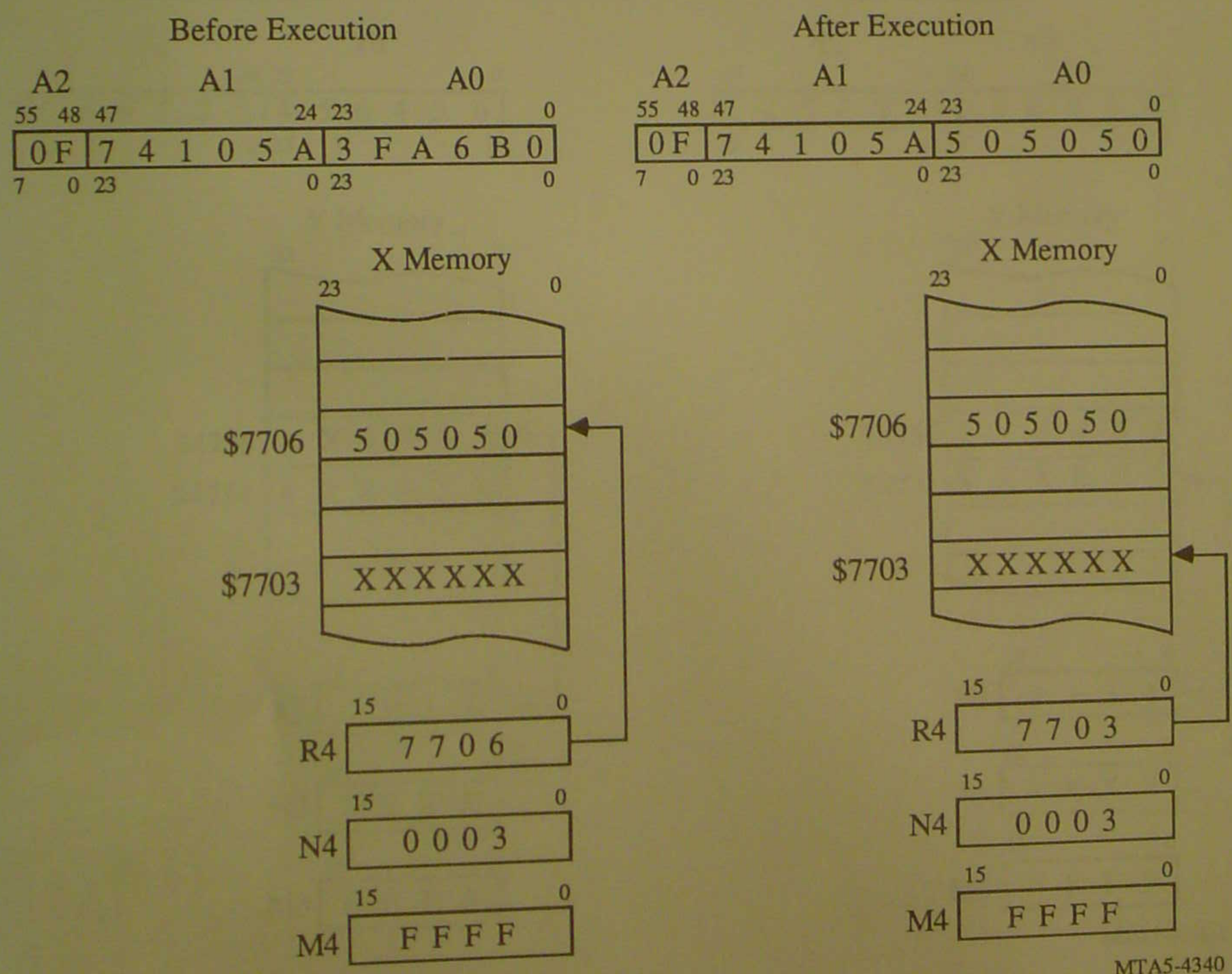
**Example:** MOVE Y0,Y:(R3) -



# ADDRESS REGISTER INDIRECT POSTDECREMENT BY OFFSET Nn

- Assembler Syntax: (Rn) - Nn
- Operands referenced: P, X, Y, L memories.
- Additional instruction execution time (clocks): 0
- Additional effective address words: 0

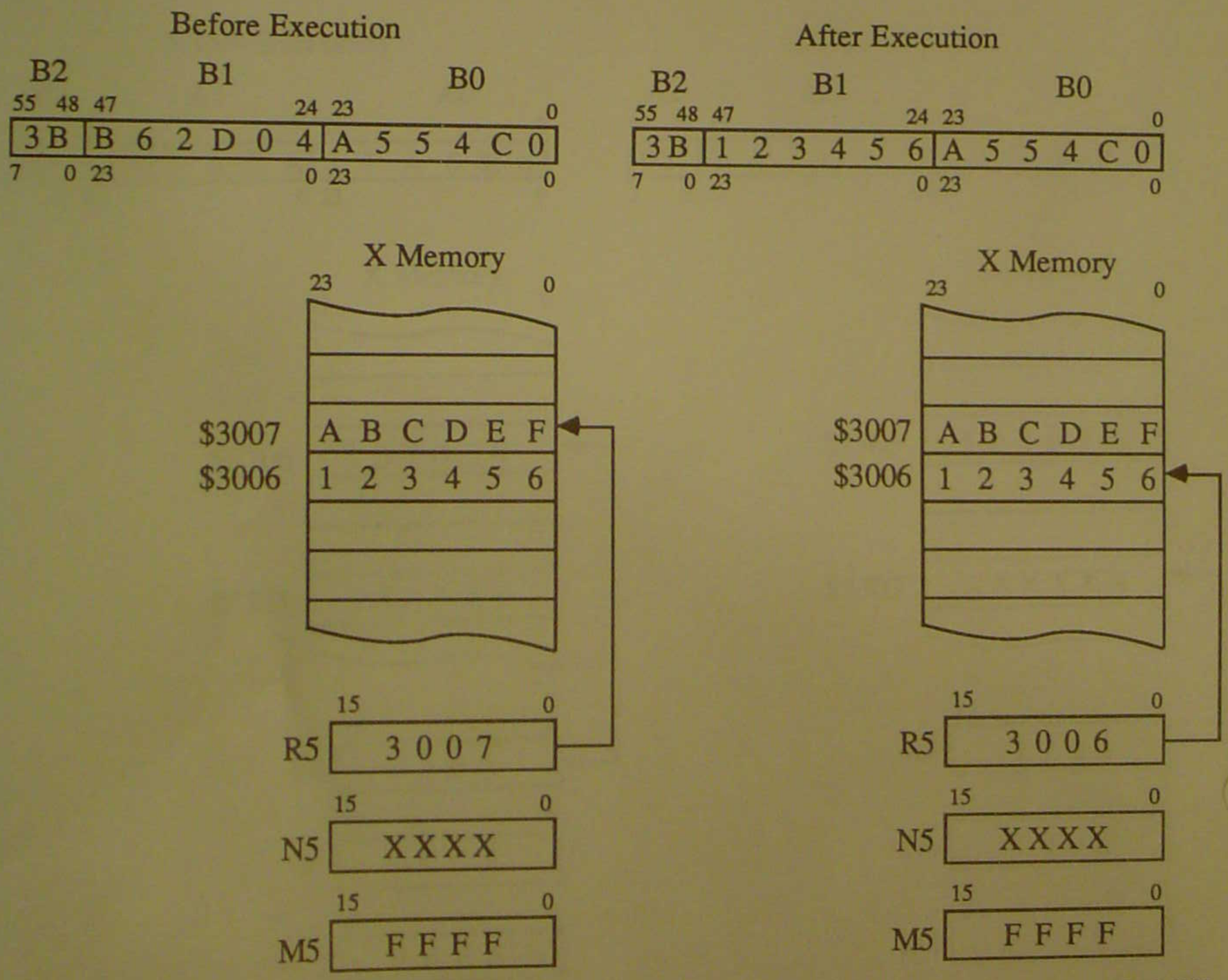
**Example:** MOVE X:(R4) - N4, A0



# ADDRESS REGISTER INDIRECT PREDECREMENT BY ONE

- Assembler Syntax: -(Rn)
- Operands referenced: P, X, Y, L memories.
- Additional instruction execution time (clocks): 2
- Additional effective address words: 0

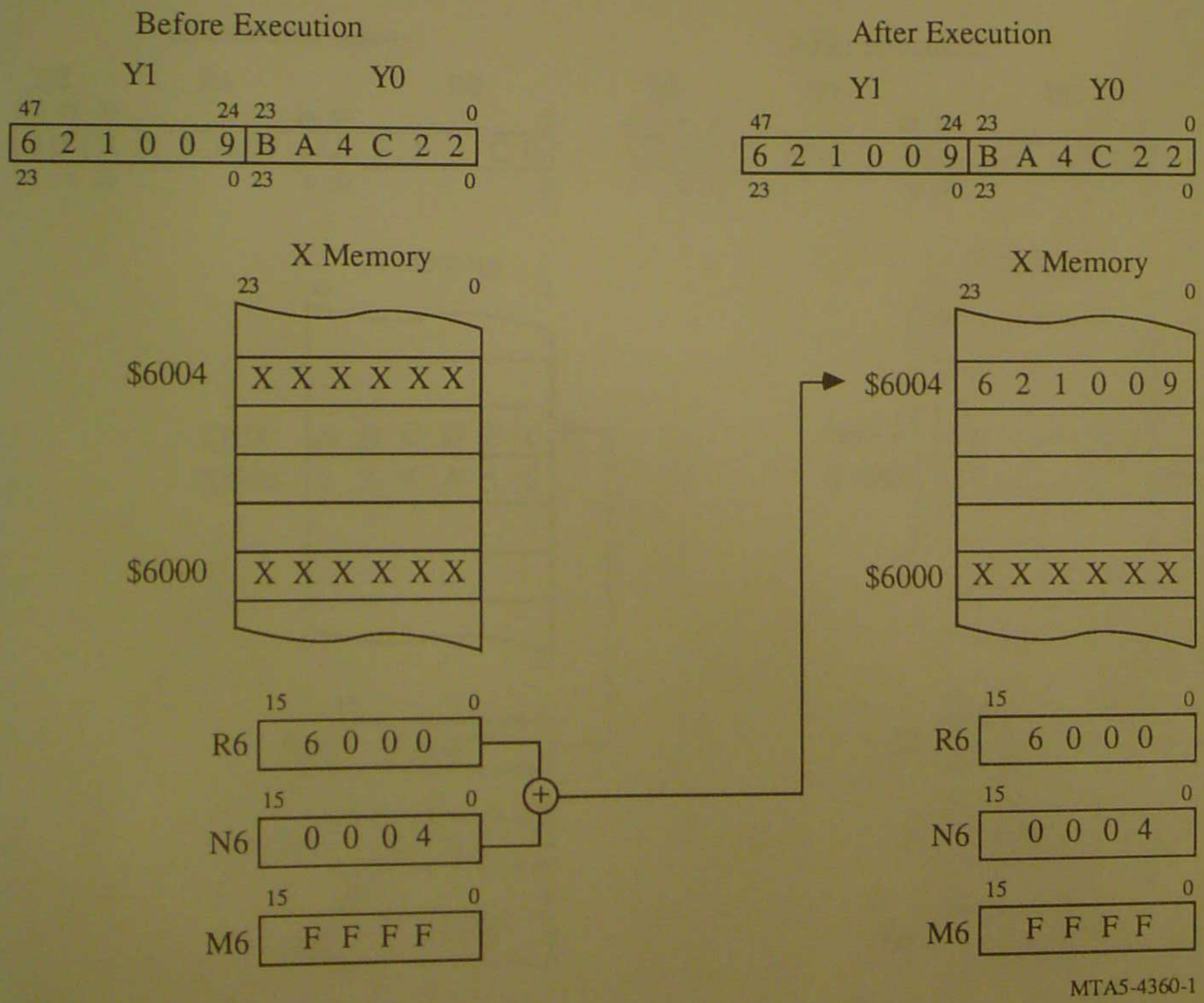
**Example:** MOVE X:- (R5), B1



# ADDRESS REGISTER INDIRECT INDEXED BY OFFSET Nn

- Assembler Syntax: (Rn + Nn)
- Operands referenced: P, X, Y, L memories.
- Additional instruction execution time (clocks): 2
- Additional effective address words: 0

**Example:** MOVE Y1,X:(R6+N6)



# SPECIAL ADDRESSING
# IMMEDIATE

- Assembler Syntax: #xxxxxx
- Operands referenced: P memory
- Additional instruction execution time (clocks): 2
- Additional effective address words: 1

Example A: Immediate into 24 bit register
MOVE #$123456,A0

Before Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| XX | XXXXXXX | XXXXXXX |
| 7 0 | 23 0 | 23 0 |

After Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| XX | XXXXXXX | 123456 |
| 7 0 | 23 0 | 23 0 |

Example B: Immediate into 56 bit register
MOVE #$123456,A

Before Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| XX | XXXXXXX | XXXXXXX |
| 7 0 | 23 0 | 23 0 |

After Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| 00 | 123456 | 000000 |
| 7 0 | 23 0 | 23 0 |

Example C: Negative Immediate into 56 bit register
MOVE #$801234,A

Before Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| XX | XXXXXXX | XXXXXXX |
| 7 0 | 23 0 | 23 0 |

After Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| FF | 801234 | 000000 |
| 7 0 | 23 0 | 23 0 |

MTA5-4400-1

# SPECIAL ADDRESSING
# IMMEDIATE SHORT

- Assembler Syntax: #xx
- Operands referenced: P memory
- Additional instruction execution time (clocks): 0
- Additional effective address words: 0

Example A: Immediate Short into 24 bit register

MOVE #$FF,A1

Destination: A0, A1, A2, B0, B1, B2, R0-R7, N0-N7

Before Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| XX | XXXXXX | XXXXXX |
| 7 0 | 23 0 | 23 0 |

After Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| XX | 0000FF | XXXXXX |
| 7 0 | 23 0 | 23 0 |

Example B: Positive Immediate into 24 bit register

MOVE #$1F,Y1

Destination: X0, X1, Y0, Y1

Before Execution

| Y1 | Y0 |
|---|---|
| 47 24 | 23 0 |
| XXXXXX | XXXXXX |
| 23 0 | 23 0 |

After Execution

| Y1 | Y0 |
|---|---|
| 47 24 | 23 0 |
| 1F0000 | XXXXXX |
| 23 0 | 23 0 |

Example C: Positive Immediate into 56 bit accumulator

MOVE #$1F,A

Destination: A, B

Before Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| XX | XXXXXX | XXXXXX |
| 7 0 | 23 0 | 23 0 |

After Execution

| A2 | A1 | A0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| 00 | 1F0000 | 000000 |
| 7 0 | 23 0 | 23 0 |

Example D: Negative Immediate into 56 bit accumulator

MOVE #$83,B

Destination: A, B

Before Execution

| B2 | B1 | B0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| XX | XXXXXX | XXXXXX |
| 7 0 | 23 0 | 23 0 |

After Execution

| B2 | B1 | B0 |
|---|---|---|
| 55 48 | 47 24 | 23 0 |
| FF | 830000 | 000000 |
| 7 0 | 23 0 | 23 0 |

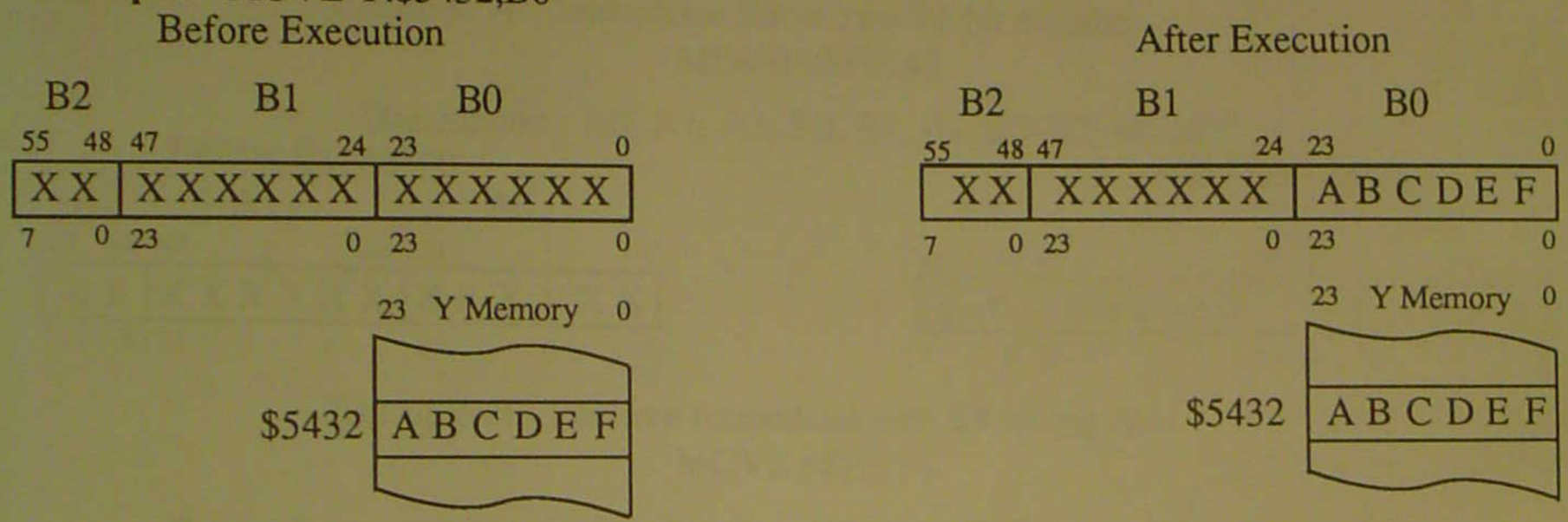
MTA5-4410-1

# SPECIAL ADDRESSING

## Absolute Long – 16 bits

- Assembler Syntax: xxxx
- Operands referenced: P, X, Y, L memories
- Additional instruction execution time (clocks): 2
- Additional effective address words: 1

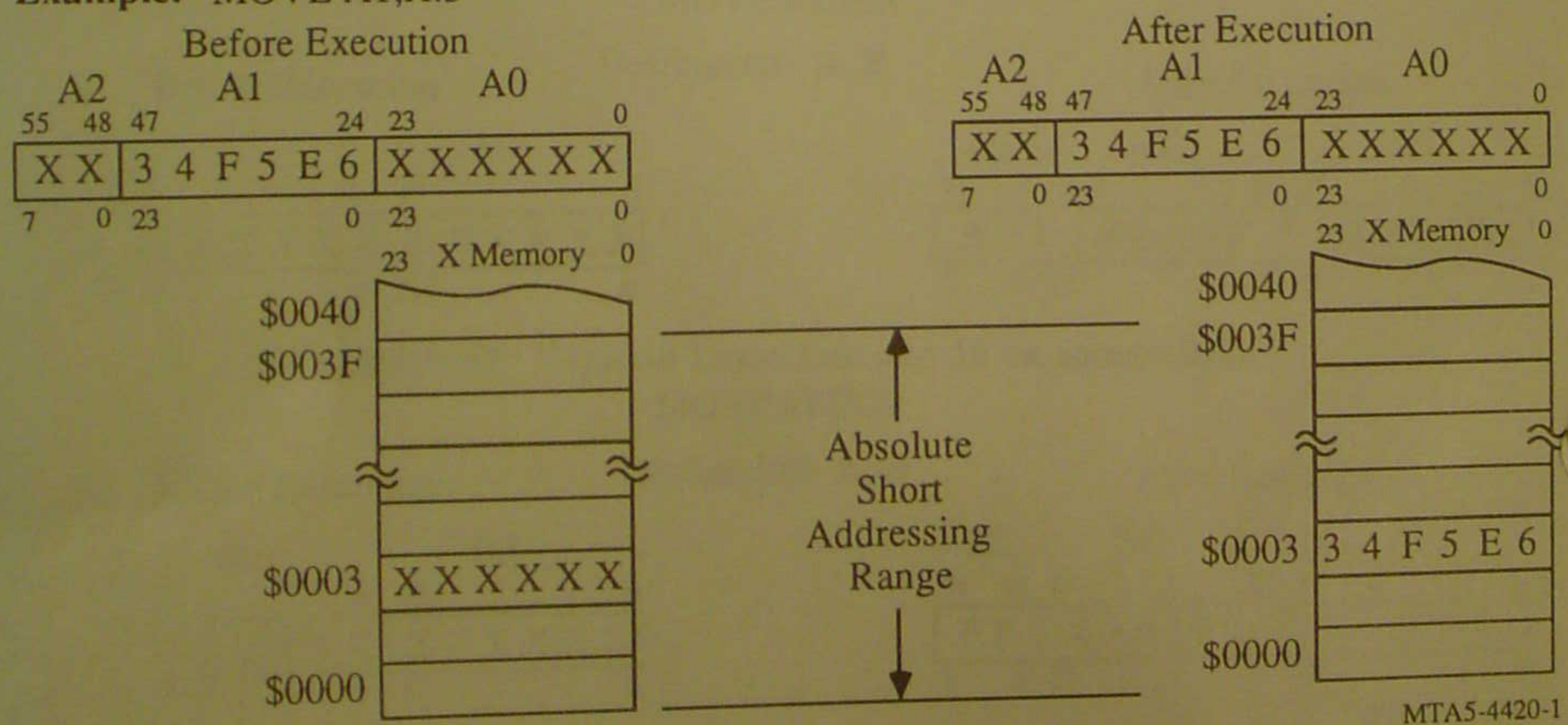
**Example:** MOVE Y:$5432,B0




## Absolute short (6 bits, zero extension)

- Assembler Syntax: aa
- Operands referenced: P, X, Y, L memories
- Additional instruction execution time (clocks): 0
- Additional effective address words: 0

**Example:** MOVE A1,X:3

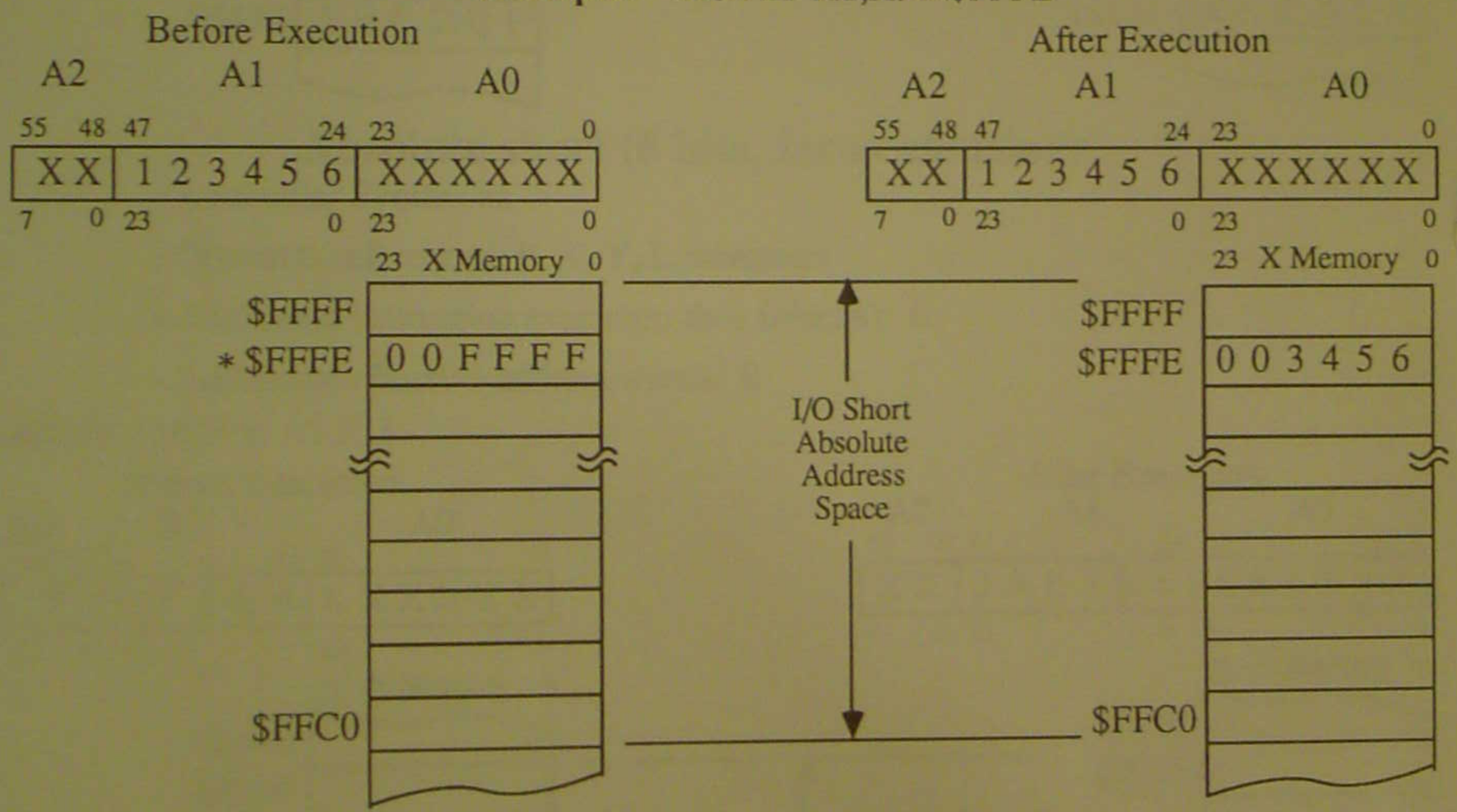


# SPECIAL ADDRESSING

## I/O Short
## (6 bits with ones extension)

- Used with MOVE peripheral data instruction (MOVEP)
- Used with bit manipulation instructions
- Operands referenced: X,Y memories
- Assembler Syntax: pp
- Additional instruction execution times (clocks): 0
- Additional effective address words: 0

**Example:** MOVEP A1,X:<<$FFFE

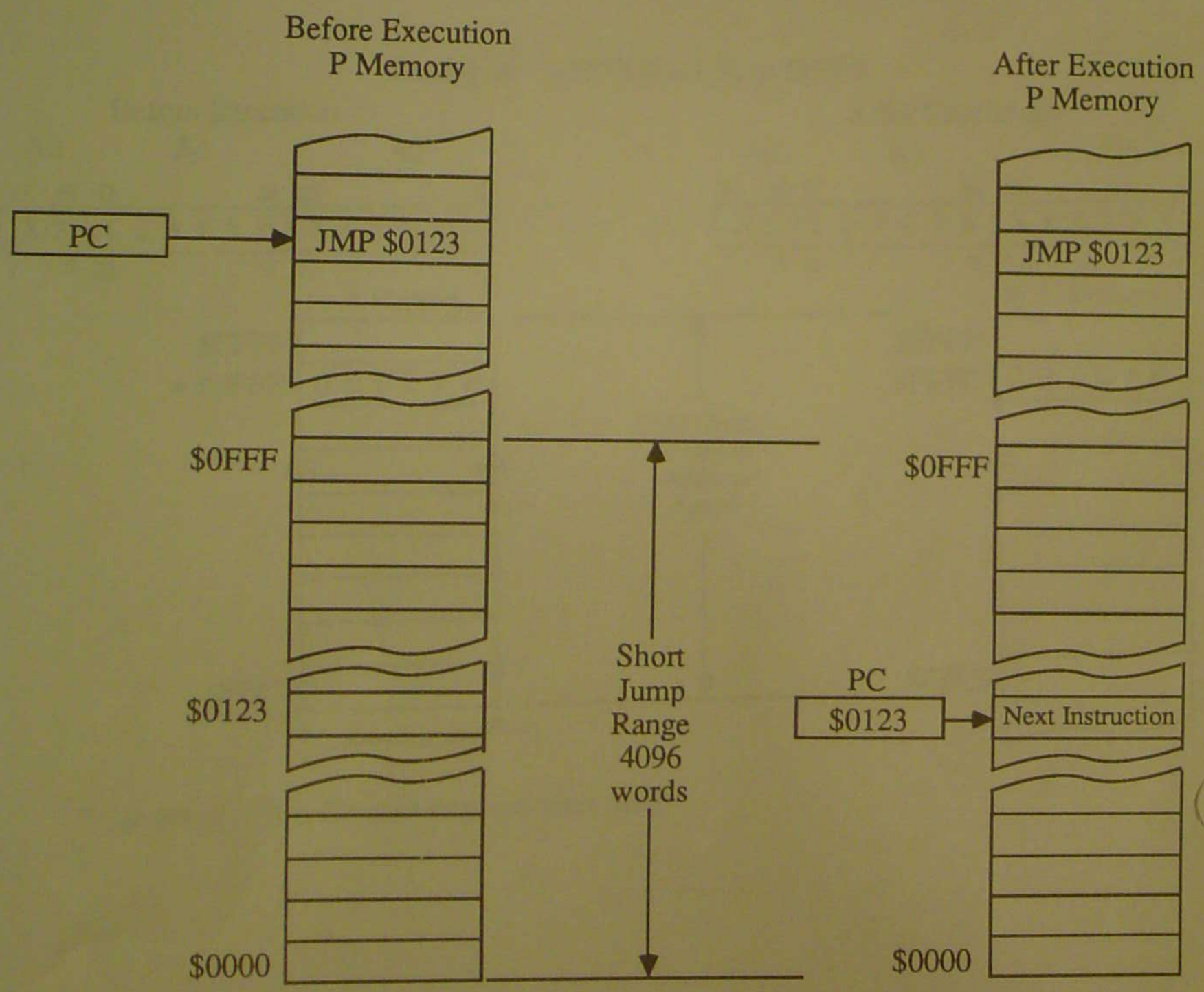



* Contents of Bus Control Register after reset

MTA5-4440-1

# SPECIAL ADDRESSING
# JUMP SHORT

- Absolute jump address
- One word instruction
- 12 bit zero extended
- Conditional jumps and jump to subroutines




MTA5-4450

# EXERCISE 6

## GENERAL ADDRESSING MODES

For this exercise assume the registers are initialized as follows:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R0 | $1004 | N0 | $FFFD | M0 | $FFFF | | Y Memory |
| R1 | $1003 | N1 | $0002 | M1 | $FFFF | $1004 | 0 0 0 0 0 E |
| R2 | $1002 | N2 | $0101 | M2 | $FFFF | $1003 | 6 5 4 3 2 1 |
| R3 | $1001 | N3 | $0003 | M3 | $FFFF | $1002 | 2 F 2 F 2 F |
| R4 | $1000 | N4 | $0003 | M4 | $FFFF | $1001 | D C B A 9 8 |
| R5 | $0FFF | N5 | $0004 | M5 | $FFFF | $1000 | A A A A A A |
| R6 | $0FFE | N6 | $FFFD | M6 | $FFFF | $0FFF | B B B B B B |
| R7 | $0FFD | N7 | $0004 | M7 | $FFFF | $0FFE | D D D 0 0 0 |
| | | | | | | $0FFD | 0 2 3 0 4 5 |

| A2 | A1 | A0 |
|---|---|---|
| $ 0 1 | 2 3 4 5 6 7 | 8 9 A B C D |

| X1 | X0 |
|---|---|
| | |

| B2 | B1 | B0 |
|---|---|---|
| | | |

| Y1 | Y0 |
|---|---|
| F F F F F F | 0 0 0 0 0 0 |

Show the contents of memory and registers after execution of each problem.

A. MOVE Y:$DCBA98, Y0 — Y1 | Y0

B. MOVE Y:(R0), Y1 — R0 | Y1 | Y0

C. MOVE Y:- (R1), B1 — R1 | B2 | B1 | B0

D. MOVE Y:(R2) -, B0 — R2 | B2 | B1 | B0

E. MOVE Y:(R3) +, X1 — R3 | X1 | X0

F. MOVE Y:(R4) + N4, X0 — R4 | X1 | X0

G. MOVE Y:(R6) - N6, A1 — R6 | A2 | A1 | A0

H. MOVE Y:(R7 + N7), A0 — R7 | A2 | A1 | A0

MTA5-4500-1

# GENERAL INSTRUCTION SET

## OBJECTIVES

After studying this section, the student should be able to:

- Understand Parallel Moves.
- Write a MOVE Instruction.
- Write Arithmetic Instructions.
- Write Logical Instructions.
- Use Branching Instructions
- Understand Bit Manipulation Instructions
- Write a Bit Branching Instruction.
- Describe the Instructions NOP, RESET, WAIT, and STOP.
- Write a Simple Program.

# DSP56000 INSTRUCTION SET

- MPU Like
- Makes Pipeline invisible
- 62 instructions
- 6 Groups
  - Move instructions
  - Arithmetic instructions†
  - Logical instructions †
  - Bit manipulation instructions
  - Program control instructions
  - Loop instructions
- 30 instructions allow parallel moves

† Most arithmetic and logical instructions allow parallel data moves

MTA5-5005-1

# PARALLEL DATA MOVE INSTRUCTIONS

- **Syntax:**

| Opcode | Operands | Parallel Move: X Bus Data | Parallel Move: Y Bus Data |
|---|---|---|---|
| | | Source 1 → Destination 1 | Source 2 → Destination 2 |
| MACR | X0, Y0, A | X:(R0) + ,X0 | Y:(R4) + ,Y0 |

**Specifies:**

- Data ALU Operation
- Logical Operation
- Convergent Rounding

**Specifies:**

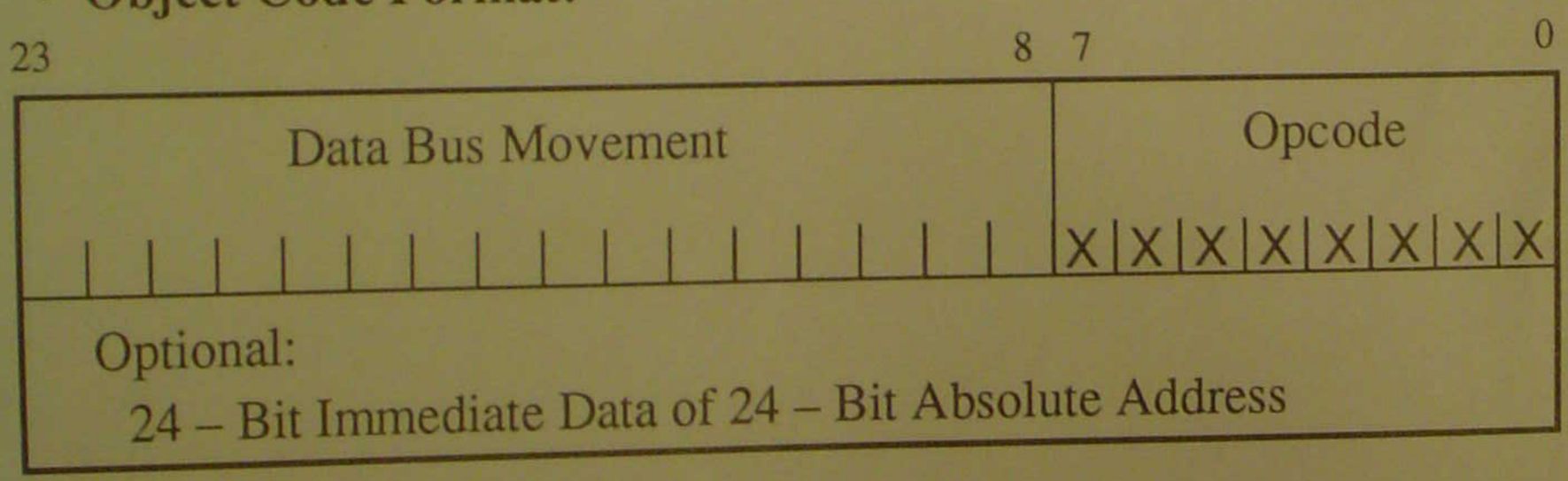
- Up to two optional data transfers
- Two different addressing modes
- Address space qualifiers [X:, Y:, L:, XY:]

**Supports:**

- Scaling
- Limiting
- Sign extension and least significant zero fill
- Duplicate sources
- Duplicate destinations not allowed

- This All Occurs In 97.5 nS!

- **Object Code Format:**




MTA5-5100-2

# IMMEDIATE SHORT DATA MOVE

## Instruction formats

| Assembler Syntax: | | Parallel Move |
|---|---|---|
| Opcode | Operand | Source, Destination |

## Addressing Modes

| Source Operand | | Destination Registers | |
|---|---|---|---|
| Immediate Short | #xx | A2 , A1, A0 | Unsigned Integer Transfer |
| | | B2, B1, B0 | |
| | | R0 – R7 | |
| | | N0 – N7 | |
| | | X1, X0 | Signed Fraction Transfer |
| | | Y1, Y0 | |
| | | A,B | |

**Example 1:**

| Opcode | Operand | PARALLEL MOVE |
|---|---|---|
| ADD | B,A | #1,B1 |

Before Execution

| | | | |
|---|---|---|---|
| A | 80 | 123456 | ABCDEF |
| B | 12 | 345678 | 9ABCDE |

After Execution

| | | | |
|---|---|---|---|
| A | 92 | 468ACF | 468ACD |
| B | 12 | 000001 | 9ABCDE |

**Example 2:**

| Opcode | Operand | PARALLEL MOVE |
|---|---|---|
| ADD | B,A | #$F3,B |

Before Execution

| | | | |
|---|---|---|---|
| A | 01 | 000001 | 000001 |
| B | 01 | 000001 | 000001 |

After Execution

| | | | |
|---|---|---|---|
| A | 02 | 000002 | 000002 |
| B | FF | F30000 | 000000 |

MTA5-5110-1

# REGISTER TO REGISTER DATA MOVE

## Instruction formats

Assembler Syntax:

| Opcode | Operand | Parallel Move |
|---|---|---|
| | | Source, Destination |

## Addressing Modes

**Source or Destination Registers**

X1, X0
Y1, Y0
A, A2 , A1, A0
B, B2, B1, B0
R0 – R7
N0 – N7

**Example 1:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | B,A | A,N7 |

Before Execution

| | | | |
|---|---|---|---|
| A | 00 | 123456 | ABCDEF |
| B | 12 | 345678 | 9ABCDE |
| N7 | 0000 | | |

After Execution

| | | | |
|---|---|---|---|
| A | 12 | 468ACF | 468ACD |
| B | 12 | 345678 | 9ABCDE |
| N7 | 3456 | | |

**Example 2:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | B,A | B,R7 |

Before Execution

| | | | |
|---|---|---|---|
| A | FE | DCBA98 | 765432 |
| B | 12 | 345678 | 9ABCDE |
| R7 | 1234 | | |

After Execution

| | | | |
|---|---|---|---|
| A | 11 | 111111 | 111110 |
| B | 12 | 345678 | 9ABCDE |
| R7 | FFFF | | |

MTA5-5120

# ADDRESS REGISTER UPDATE INSTRUCTION FORMATS

**Assembler Syntax:**

| Opcode | Operand | **Parallel Move** <Effective Address> |
|---|---|---|

## Addressing Modes

| **Effective Address Modes:** | | **Update Operand** |
|---|---|---|
| Postincrement by 1 | (Rn)+ | R0-R7 |
| Postdecrement by 1 | (Rn)- | |
| Postincrement by Offset Nn | (Rn)+Nn | |
| Postdecrement by Offset Nn | (Rn)-Nn | |

**Example 1:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | A,B | (R0)+N0 |

Before Execution

| | | | |
|---|---|---|---|
| A | 12 | 345678 | 9ABCDE |
| B | 12 | 345678 | 9ABCDE |
| R0 | 2000 | | |
| N0 | F000 | | |

After Execution

| | | | |
|---|---|---|---|
| A | 12 | 345678 | 9ABCDE |
| B | 24 | 68ACF1 | 3579BC |
| R0 | 1000 | | |
| N0 | F000 | | |

**Example 2:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | A,B | (R0)-N0 |

Before Execution

| | | | |
|---|---|---|---|
| A | 12 | 345678 | 9ABCDE |
| B | 12 | 345678 | 9ABCDE |
| R0 | 2000 | | |
| N0 | 1000 | | |

After Execution

| | | | |
|---|---|---|---|
| A | 12 | 345678 | 9ABCDE |
| B | 24 | 68ACF1 | 3579BC |
| R0 | 1000 | | |
| N0 | 1000 | | |

MTA5-5130

# X MEMORY DATA MOVE INSTRUCTION FORMATS

| Assembler Syntax: | | Parallel Move |
|---|---|---|
| Opcode | Operand | X:<Effective Address>,Destination |
| Opcode | Operand | Source,X:<Effective Address> |
| Opcode | Operand | #xxxxxx, Destination |

## Addressing Modes

**Effective Address Modes:**

| | |
|---|---|
| Absolute Short Address | aa |
| Absolute Address | xxxx |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1 | (Rn)+ |
| Postdecrement by 1 | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

**Source or Destination for X Memory**

X1,X0
Y1,Y0
A,A2,A1,A0
B,B2,B1,B0
R0-R7
N0-N7

**Example 1:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | Y0,B | A2,X:$1000 |

Before Execution

| | | | |
|---|---|---|---|
| Y | 606060 | 000002 | |
| A | FF | 000006 | 000005 |
| B | 00 | 000009 | 505050 |

X Memory

| | |
|---|---|
| | |
| | |
| $1000 | 000002 |
| | |
| | |

After Execution

| | | | |
|---|---|---|---|
| Y | 606060 | 000002 | |
| A | FF | 000006 | 000005 |
| B | 00 | 00000B | 505050 |

X Memory

| | |
|---|---|
| | |
| | |
| $1000 | FFFFFF |
| | |
| | |

**Example 2:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | Y1,B | #$57005,A0 |

Before Execution

| | | | |
|---|---|---|---|
| Y | 000004 | 000003 | |
| A | 00 | 303030 | 020202 |
| B | 00 | 000002 | 000000 |

After Execution

| | | | |
|---|---|---|---|
| Y | 000004 | 000003 | |
| A | 00 | 303030 | 057005 |
| B | 00 | 000006 | 000000 |

MTA5-5140-2

# X MEMORY and REGISTER DATA MOVE INSTRUCTION FORMATS

| Assembler Syntax: | | Parallel Move | |
|---|---|---|---|
| Opcode | Operand | X:<Effective Address>,Destination 1 | Source 2, Destination 2 |
| Opcode | Operand | Source 1,X:<Effective Address> | Source 2, Destination 2 |
| Opcode | Operand | #xxxxxx, Destination 1 | Source 2, Destination 2 |

## Addressing Modes

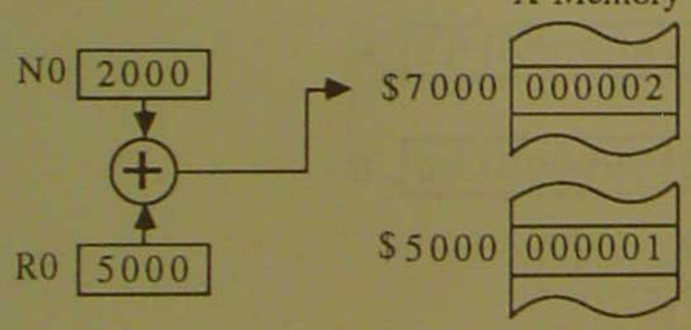
| Effective Address Modes: | | Source 1 or Destination 1 for X memory | Source 2 | Destination 2 |
|---|---|---|---|---|
| Absolute Short | aa | X1,X0 | A,B | Y1,Y0 |
| Absolute Address | xxxx | A,B | | |
| Address Register Indirect with; | | | | |
| No Update | (Rn) | | | |
| Postincrement by 1 | (Rn)+ | | | |
| Postdecrement by 1 | (Rn)- | | | |
| Postincrement by Offset Nn | (Rn)+Nn | | | |
| Postdecrement by Offset Nn | (Rn)-Nn | | | |
| Indexed by Offset Nn | (Rn+Nn) | | | |
| Predecrement by 1 | -(Rn) | | | |

**Example 1:**

| Opcode | Operand | Parallel Move | |
|---|---|---|---|
| ADD | X0,A | A,X:(R0+N0) | A,Y1 |

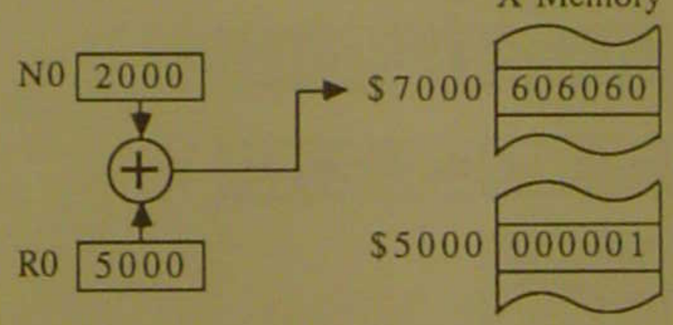
Before Execution

| | | |
|---|---|---|
| X | 020202 | 010101 |
| Y | 5ABD01 | 87CC99 |
| A 00 | 606060 | 505050 |




After Execution

| | | |
|---|---|---|
| X | 020202 | 010101 |
| Y | 606060 | 87CC99 |
| A 00 | 616161 | 505050 |

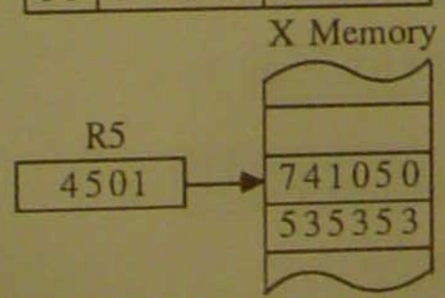



**Example 2:**

| Opcode | Operand | Parallel Move | |
|---|---|---|---|
| ADD | Y1,B | X:-(R5),A | A,Y0 |

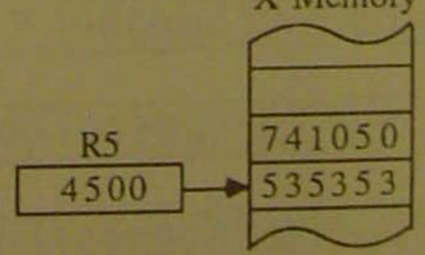
Before Execution

| | | |
|---|---|---|
| Y | 000001 | 000002 |
| A FF | 010101 | 020202 |
| B 00 | 000003 | 000004 |




After Execution

| | | |
|---|---|---|
| Y | 000001 | 800000 |
| A 00 | 535353 | 000000 |
| B 00 | 000004 | 000004 |




MTA5- 5150-2

# Y MEMORY DATA MOVE INSTRUCTION FORMATS

| Assembler Syntax: | | Parallel Move |
|---|---|---|
| Opcode | Operand | Y:<Effective Address>,Destination |
| Opcode | Operand | Source,Y:<Effective Address> |
| Opcode | Operand | #xxxxxx, Destination |

## Addressing Modes

| Effective Address Modes: | |
|---|---|
| Absolute Short Address | aa |
| Absolute Address | xxxx |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1 | (Rn)+ |
| Postdecrement by 1 | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

| Source or Destination for Y Memory |
|---|
| X1,X0 |
| Y1,Y0 |
| A,A2,A1,A0 |
| B,B2,B1,B0 |
| R0-R7 |
| N0-N7 |

**Example 1:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | Y,A | Y:$20,B2 |

Before Execution

| | | | |
|---|---|---|---|
| Y | 000000 | 000001 | |
| A | 00 | FFFFFF | FFFFFF |
| B | 00 | 000023 | 000000 |

Y Memory

| | |
|---|---|
| 20 | 123456 |

After Execution

| | | | |
|---|---|---|---|
| Y | 000000 | 000001 | |
| A | 01 | 000000 | 000000 |
| B | 56 | 000023 | 000000 |

Y Memory

| | |
|---|---|
| 20 | 123456 |

**Example 2:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | Y,B | R7,A |

Before Execution

| | | | |
|---|---|---|---|
| Y | 000000 | 000001 | |
| A | FF | ABCDEF | ABCDEF |
| B | 00 | 7FFFFF | FFFFFF |
| R7 | F234 | | |

After Execution

| | | | |
|---|---|---|---|
| Y | 000000 | 000001 | |
| A | 00 | 00F234 | 000000 |
| B | 00 | 800000 | 000000 |
| R7 | F234 | | |

MTA5-5160-2

# REGISTER and Y MEMORY DATA MOVE INSTRUCTION FORMATS

**Assembler Syntax:**

| Opcode | Operand |
|---|---|
| Opcode | Operand |
| Opcode | Operand |

**Parallel Move**

| | |
|---|---|
| Source 1, Destination 1 | Y: <Effective Address>,Destination 2 |
| Source 1, Destination 1 | Source 2, Y:<Effective Address> |
| Source 1, Destination 1 | #xxxxxx, Destination 2 |

**ADDRESSING MODES**

| Source 1 | Destination 1 |
|---|---|
| A,B | X1, X0 |

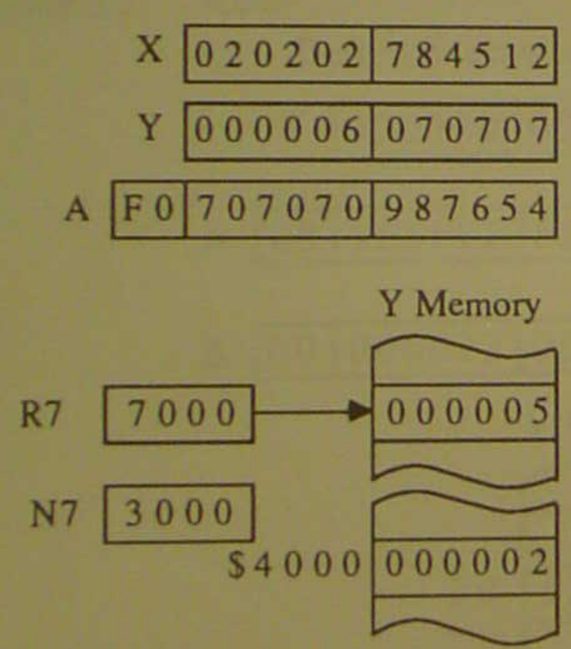
| Y Effective Address MODES: | |
|---|---|
| Absolute Short | aa |
| Absolute Address | xxxx |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1 | (Rn)+ |
| Postdecrement by 1 | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

| Source 2 or Destination 2 for Y memory |
|---|
| Y1, Y0 |
| A, B |

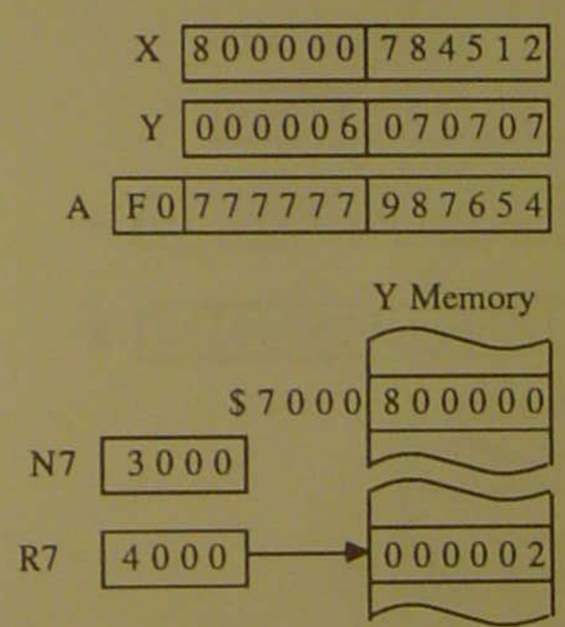
**Example 1:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | Y0,A | A,X1 A,Y:(R7)-N7 |

Before Execution



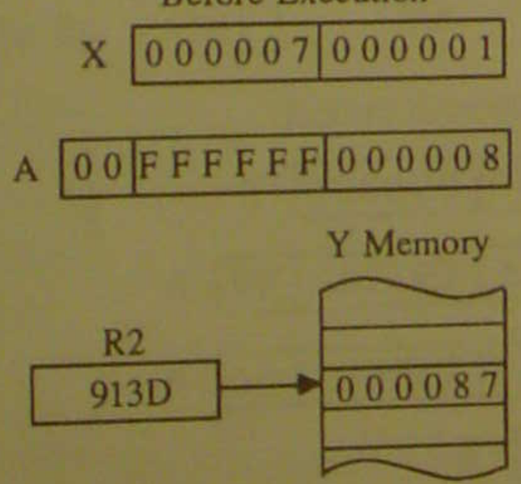

After Execution



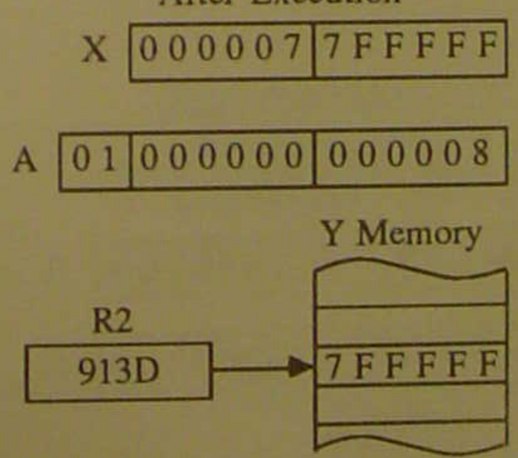

**Example 2:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | X0,A | A,X0 A,Y:(R2) |

Before Execution




After Execution




MTA5-5170-2

# XY MEMORY PARALLEL DATA MOVE INSTRUCTION FORMATS

**Assembler Syntax:**

| | | Parallel Move | |
|---|---|---|---|
| Opcode | Operand | X:<Effective Address>,Destination 1 | Y:<Effective Address>,Destination 2 |
| Opcode | Operand | X:<Effective Address>,Destination 1 | Source 2,Y:<Effective Address> |
| Opcode | Operand | Source 1,X:<Effective Address> | Y:<Effective Address>,Destination 2 |
| Opcode | Operand | Source 1,X:<Effective Address> | Source 2,Y:<Effective Address> |

## Addressing Modes

| Effective Address Modes: | | Source 1 or Destination 1 for X memory | Source 2 or Destination 2 for Y memory |
|---|---|---|---|
| Address Register Indirect with: | | X1,X0,A,B | Y1,Y0,A,B |
| No Update | (Rn) | | |
| Postincrement by 1 | (Rn)+ | | |
| Postdecrement by 1 | (Rn)- | | |
| Postincrement by Offset Nn | (Rn)+Nn | | |

**Example 1:**

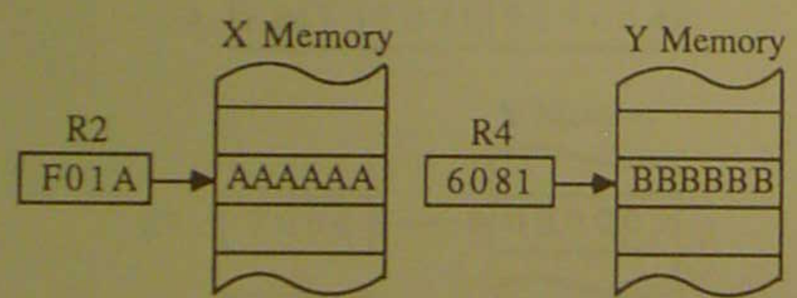
| Opcode | Operand | Parallel Move | |
|---|---|---|---|
| ADD | X0,A | X:(R2)+,X0 | Y:(R4)-,Y1 |

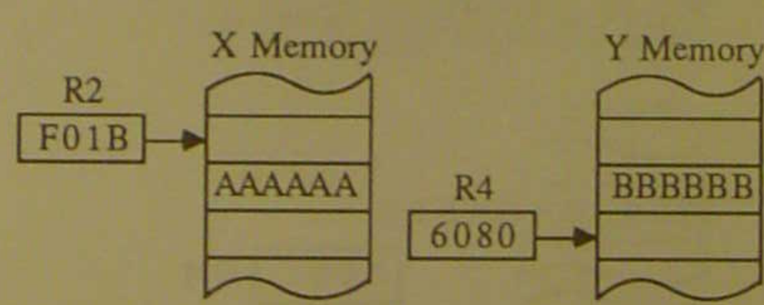
Before Execution

| | | | |
|---|---|---|---|
| X | | E0E0E0 | F0F0F0 |
| Y | | D0D0D0 | C0C0C0 |
| A | 00 | F00000 | F00000 |

After Execution

| | | | |
|---|---|---|---|
| X | | E0E0E0 | AAAAAA |
| Y | | BBBBBB | C0C0C0 |
| A | 00 | E0F0F0 | F00000 |




**Example 2:**

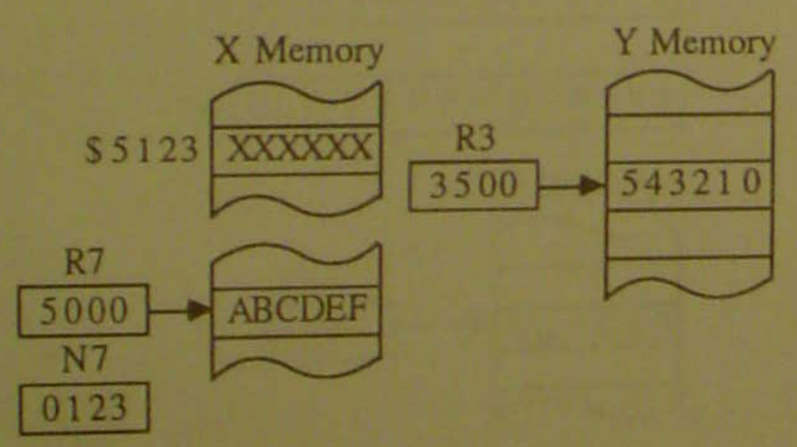
| Opcode | Operand | Parallel Move | |
|---|---|---|---|
| ADD | Y1,B | B,X:(R7)+N7 | Y:(R3),Y1 |

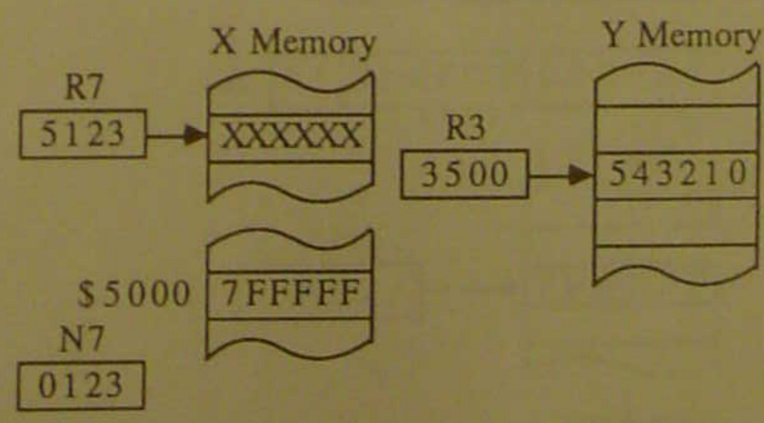
Before Execution

| | | | |
|---|---|---|---|
| Y | | 010101 | 020202 |
| B | 30 | 000000 | 000000 |

After Execution

| | | | |
|---|---|---|---|
| Y | | 543210 | 020202 |
| B | 30 | 010101 | 000000 |




MTA5-5180-2

# LONG (L) MEMORY DATA MOVE INSTRUCTION FORMATS

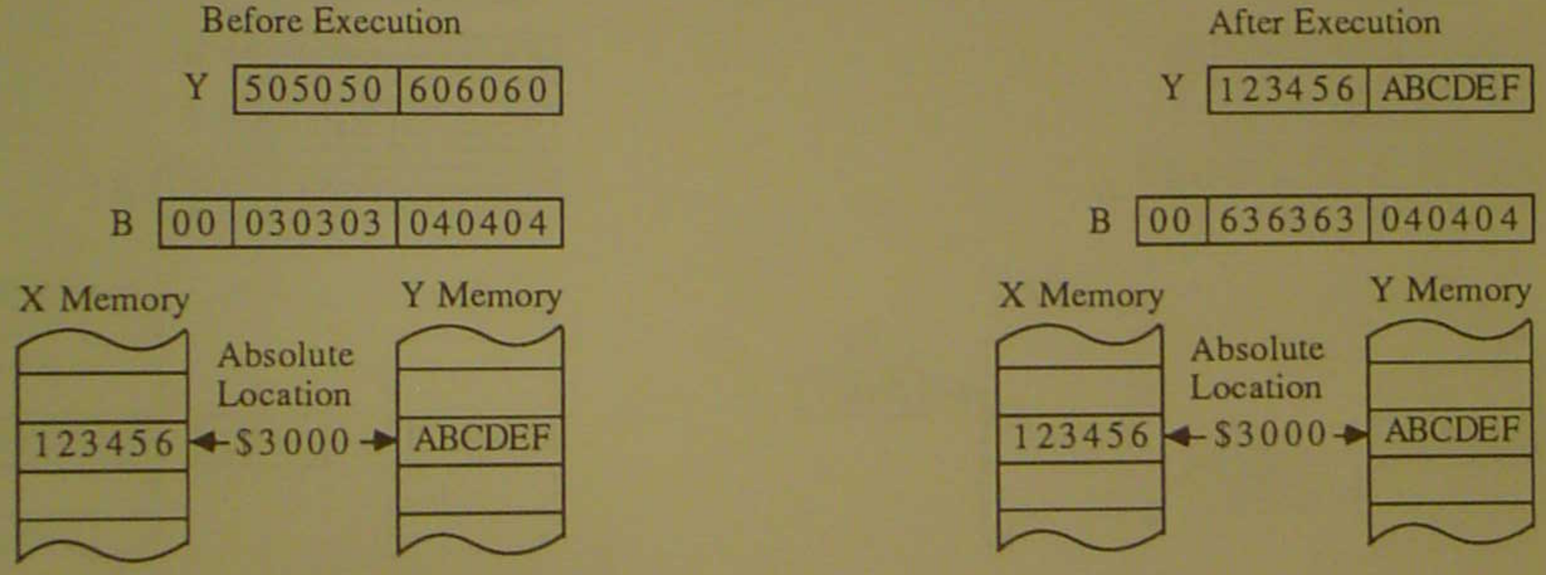
| Assembler Syntax: | | Parallel Move |
|---|---|---|
| Opcode | Operand | L:<Effective Address>,Destination |
| Opcode | Operand | Source,L:<Effective Address> |

## Addressing Modes

| L Effective Address Modes: | | Source or Destination for L Memory |
|---|---|---|
| Absolute Short | aa | X,Y |
| Absolute Address | xxxx | A,A10 |
| Address Register Indirect with; | | B,B10 |
| No Update | (Rn) | AB,BA |
| Postincrement by 1 | (Rn)+ | |
| Postdecrement by 1 | (Rn)- | |
| Postincrement by Offset Nn | (Rn)+Nn | |
| Postdecrement by Offset Nn | (Rn)-Nn | |
| Indexed by Offset Nn | (Rn+Nn) | |
| Predecrement by 1 | -(Rn) | |

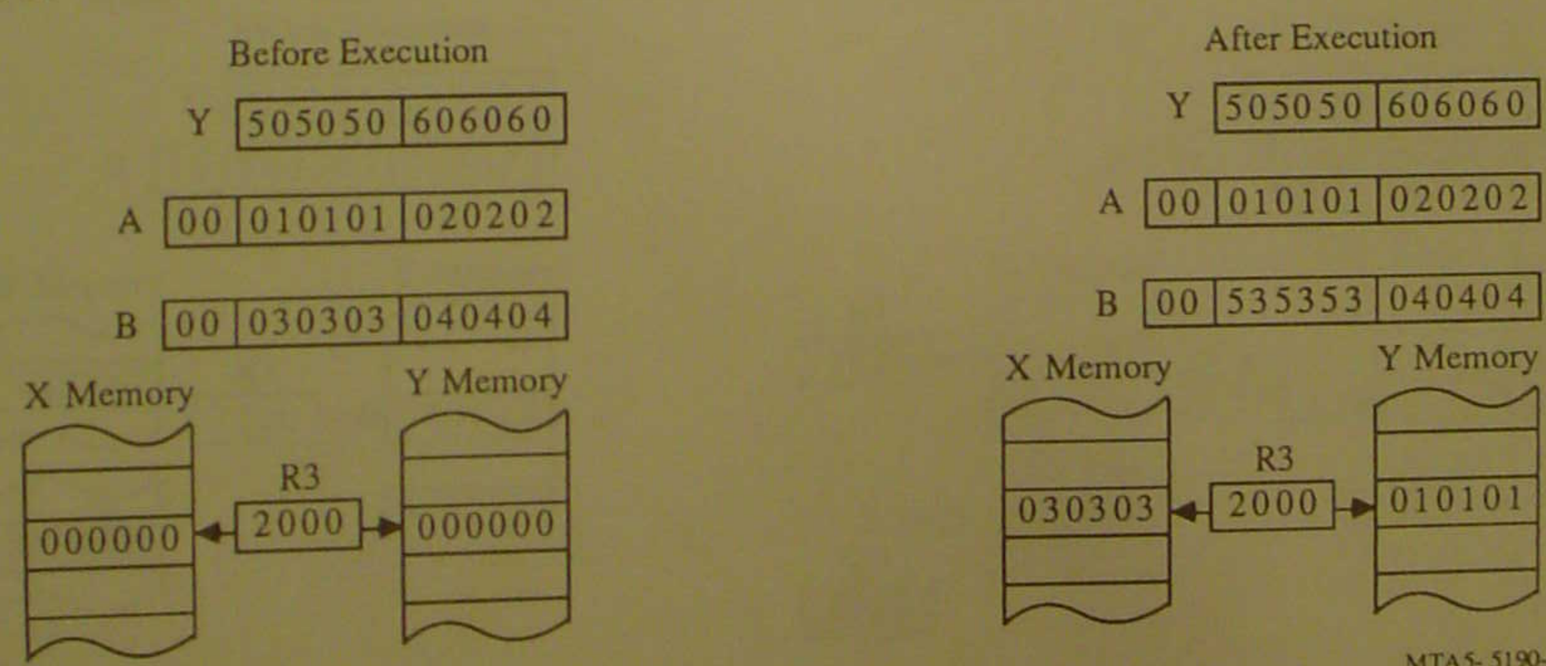
**Example 1:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | Y0,B | L:$3000,Y |




**Example 2:**

| Opcode | Operand | Parallel Move |
|---|---|---|
| ADD | Y1,B | BA,L:(R3) |




MTA5- 5190-2

# MOVE INSTRUCTIONS

| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| MOVE | Move Data Registers<br>S → D | MOVE S,D |

## Operand Combinations

**Source Destination**

All Parallel Move Combinations

**Example:** MOVE B10,L:$A1BD

| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| MOVEC | Move Data Registers<br>S → D | MOVE(C) S,D |

## Operand Combinations

**Source or Destination**

X1,X0
Y1,Y0
A,A2,A1,A0
B,B2,B1,B0
R0 – R7
M0 – M7
SR
OMR
SP,SSH,SSL
LA,LC

| | |
|---|---|
| Immediate Data | #xxxxxx |
| Immediate Short Data | #xx |
| X or Y memory; | |
| Absolute Address | xxxx |
| Absolute Short Address | aa |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1 | (Rn)+ |
| Postdecrement by 1 | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

**Source or Destination Control Registers**

M0-M7
SR
OMR
SP,SSH,SSL
LA,LC

**Example:** MOVE SSL,LC

MTA5-5193-1

# MOVE INSTRUCTIONS (Cont.)

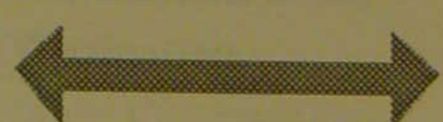
| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| MOVEM | Move Program Memory<br>S → P:<ea><br>P:<ea> → D | MOVE(M) S,P:<ea><br>MOVE(M) P:<ea>,D |

## Operand Combinations

**PROGRAM Memory, Effective Address**

| | |
|---|---|
| Absolute Short Address | aa |
| Absolute Address | xxxx |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1 | (Rn)+ |
| Postdecrement by 1 | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

**Example:** MOVE OMR,P:(R5+N5)



**Register List**

X1,X0
Y1,Y0
A,A2,A1,A0
B,B2,B1,B0
R0 - R7
N0 - N7
M0 - M7
SR
OMR
SP,SSH,SSL
LA,LC

| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| MOVEP | Move Peripheral Data<br>S → X or Y Peripheral<br>X OR Y Peripheral → D | MOVEP S,X:<pp><br>MOVEP S,Y:<pp><br>MOVEP X:<pp>,D<br>MOVEP Y:<pp>,D |

## Operand Combinations

**Source or Destination**

X1,X0
Y1,Y0
A,A2,A1,A0
B,B2,B1,B0
R0 – R7
N0 - N7
M0 – M7
SR
OMR
SP,SSH,SSL
LA,LC

X,Y, or PROGRAM memory;

| | |
|---|---|
| Absolute Address | xxxx |
| Immediate Data | #xxxxxx |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1 | (Rn)+ |
| Postdecrement by 1 | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

**Example:** MOVEP Y:$FFF6, P:$1000

X OR Y
I/O Space

MTA5- 5196-2

# LOAD UPDATED ADDRESS

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| LUA | 16 | Load Updated Address<br>ea –> D | LUA ea, D |

## Operand Combinations

| Effective Addressing Modes | <ea> | Destinations |
|---|---|---|
| Postincrement by 1 | (Rn)+ | R0-R7 |
| Postdecrement by 1 | (Rn)- | N0-N7 |
| Postincrement by Offset Nn | (Rn)+Nn | |
| Postdecrement by Offset Nn | (Rn)-Nn | |

**Example 1:** LUA (R4)-,R3

Before Execution

R3 [ X X X X ]

R4 [ 5 0 0 0 ]

After Execution

R3 [ 4 F F F ]

R4 [ 5 0 0 0 ]

**Example 2:** LUA (R1)-N1,N7

Before Execution

R1 [ 7 A 3 D ]

N1 [ 6 9 D C ]

N7 [ X X X X ]

After Execution

R1 [ 7 A 3 D ]

N1 [ 6 9 D C ]

N7 [ 1 0 6 1 ]

MTA5-5198-1

# EXERCISE 7

## GENERAL INSTRUCTIONS

1. When using an instruction with parallel moves, duplicate _____ are permitted, but duplicate _____ are not permitted.

2. The DSP56000/1 has _____ instructions that are divided into _____ groups.

3. Most instructions in the _____ and _____ groups allow parallel moves.

4. Write one instruction that performs the following operations:
   A. Adds accumulator A to B.
   B. Moves X0 to memory using Register 6 in the Postincrement by 1 mode.
   C. Moves memory to Y1 using Register 3 in the Postincrement by Offset Nn mode.
   Solution:_____

5. Write single instructions to perform the following operations:
   A. Put the processor in Mode 3 with the X and Y ROM enabled._____
   B. Set all external memory and I/O to 5 wait states. _____

6. The following instructions are not permitted. For each instruction, indicate why they are illegal.

ADD B,A X1,B Y1,A _____

ADD A,B X1,X:-(R7) Y:(R0),Y0 _____

ADD A,B X0,X:(R4)+N4 Y0,Y:(R7)+N7 _____

ADD A,B Y:(R3),Y _____

MTA5-5199-1

# ADDITION AND SUBTRACTION ARITHMETIC INSTRUCTIONS

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| ADD | 24,48,56 | Addition<br>S + D → D | ADD S,D [Parallel Move] |
| SUB | 24,48,56 | Subtraction<br>D - S → D | SUB S,D [Parallel Move] |

| Operand Combinations Source | Destination |
|---|---|
| A | B |
| B | A |
| X,X1,X0 | A,B |
| Y,Y1,Y0 | A,B |

**Example:** ADD A,B

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| ADC | 48,56 | Addition<br>S + D + C → D | ADC S,D [Parallel Move] |
| SBC | 48,56 | Subtraction<br>D - S - C → D | SBC S,D [Parallel Move] |

| Operand Combinations Source | Destination |
|---|---|
| X,Y | A,B |

**Example:** SBC Y,B

MTA5-5210

# SHIFT AND ADD OR SUBTRACT ARITHMETIC INSTRUCTIONS

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| ADDR | 56 | Shift Right and Add Accumulators<br>S + D/2 → D | ADDR S,D [Parallel Move] |
| SUBR | 56 | Shift Right and Subtract Accumulators<br>D/2 - S → D | SUBR.S,D [Parallel Move] |
| ADDL | 56 | Shift Left and Add Accumulators<br>S + 2 x D → D | ADDL S,D [Parallel Move] |
| SUBL | 56 | Shift Left and Subtract Accumulators<br>2 x D - S → D | SUBL S,D [Parallel Move] |

**Operand Combinations**

| Source | Destination |
|---|---|
| A | B |
| B | A |

**Example 1:** SUBR B,A #$123456, B

Before Execution

| | | | |
|---|---|---|---|
| A | 22 | 222222 | 222222 |
| B | 11 | 111111 | 111111 |

After Execution

| | | | |
|---|---|---|---|
| A | 00 | 000000 | 000000 |
| B | 00 | 123456 | 000000 |

**Example 2:** ADDL A,B

Before Execution

| | | | |
|---|---|---|---|
| A | 10 | 101010 | 101010 |
| B | 01 | 010101 | 010101 |

After Execution

| | | | |
|---|---|---|---|
| A | 10 | 101010 | 101010 |
| B | 12 | 121212 | 121212 |

MTA5- 5220-1

# ARITHMETIC SHIFT

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| ASR | 56 | Arithmetic Shift Right<br>55 47 23 0 C | ASR D [Parallel Move] |
| ASL | 56 | Arithmetic Shift Left<br>C 55 47 23 0 0 | ASL D [Parallel Move] |

**Destination**

A,B

**Example 1:** ASR A

Before Execution

| | | | | Carry |
|---|---|---|---|---|
| A | 88 | 888888 | 111111 | 0 |

After Execution

| | | | | Carry |
|---|---|---|---|---|
| A | C4 | 444444 | 088888 | 1 |

**Example 2:** ASL B

Before Execution

| Carry | | | | |
|---|---|---|---|---|
| 0 | B | C0 | F11111 | 888888 |

After Execution

| Carry | | | | |
|---|---|---|---|---|
| 1 | B | 81 | E22223 | 111110 |

MTA5-5230

# MULTIPLY ARITHMETIC INSTRUCTIONS

| Mnemonic | Operand Size | Operand Size | Assembler Syntax |
|---|---|---|---|
| MPY | 24, 56 | Signed Multiply<br>± (S1 * S2)→ D | MPY (±) S1, S2, D [Parallel Move] |
| MPYR | 24, 56 | Signed Multiply and Round<br>± (S1 * S2) + r→ D | MPYR (±) S1, S2, D [Parallel Move] |

## Operand Combinations

| Source 1 | Source 2 | Destination |
|---|---|---|
| X0 | X0 | A or B |
| Y0 | Y0 | |
| X1 | X0 | |
| Y1 | Y0 | |
| X0 | Y1 | |
| Y0 | X0 | |
| X1 | Y0 | |
| Y1 | X1 | |

Only these combinations of Source1 and Source2 are valid.
X1 * X1 and Y1 * Y1 are not valid.

**Example:** MPY - X1,Y0,B

Before Execution

| | | | |
|---|---|---|---|
| X | | 000004 | 111111 |
| Y | | 222222 | 084210 |
| B | 00 | 000000 | 000000 |

After Execution

| | | | |
|---|---|---|---|
| X | | 000004 | 111111 |
| Y | | 222222 | 084210 |
| B | FF | FFFFFF | BDEF80 |

MTA5- 5240-1

# COMPARE ARITHMETIC INSTRUCTIONS

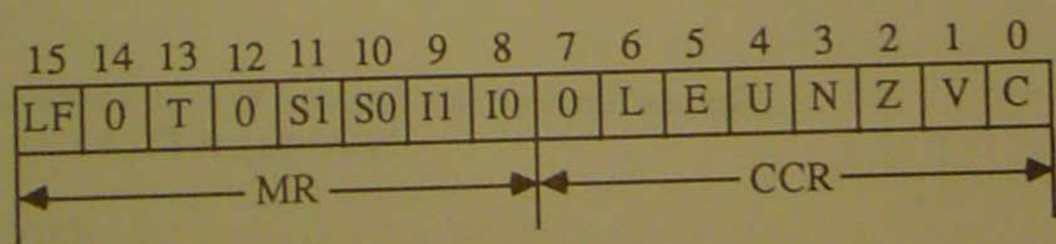
| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| CMP | 24, 56 | Compare<br>D-S | CMP S,D [Parallel Move] |
| CMPM | 24, 56 | Compare Magnitude<br>\|D\| - \|S\| | CMPM S,D [Parallel Move] |
| TST | 56 | Test Accumulator<br>D - 0 | TST D [Parallel Move] |

## Operand Combinations

| Source | Destination |
|---|---|
| A | B |
| B | A |
| X1,X0 | A,B |
| Y1,Y0 | A,B |

The result from the subtract is not stored, the condition code register is updated.

**Condition Codes:**




L - Set if limiting (parallel move) or overflow has occurred in result

E - Set if the signed integer portion of result is in use

U - Set if result is unnormalized

N - Set if bit 55 of result is set

Z - Set if result equals zero

V - Set if overflow has occurred in result

C - Set if a carry (or borrow) occurs from bit 55 of result

Note: The definition of the E and U bits varies according to the scaling mode being used

MTA5-5250-1

# TRANSFER ARITHMETIC INSTRUCTIONS

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| TFR | 24, 56 | Transfer Data ALU Register<br>S→D | TFR S,D [Parallel Move] |
| Tcc | 16, 24, 56 | Conditionally Transfer<br>Data ALU Registers<br>If cc Then S1→D1<br>or<br>If cc Then S1→D1 and S2→D2 | Tcc S1,D1 [S2,D2] |

## Operand Combinations

| Source1 | Destination1 | Source 2 | Destination 2 |
|---|---|---|---|
| A | B | R0-R7 | R0-R7 |
| B | A | | |
| X1,X0 | A,B | | |
| Y1,Y0 | A,B | | |

- Performs transfer without shifting or limiting.
- Uses internal ALU data paths.
- Does not affect Condition Code bits.

## Tcc Test Conditions

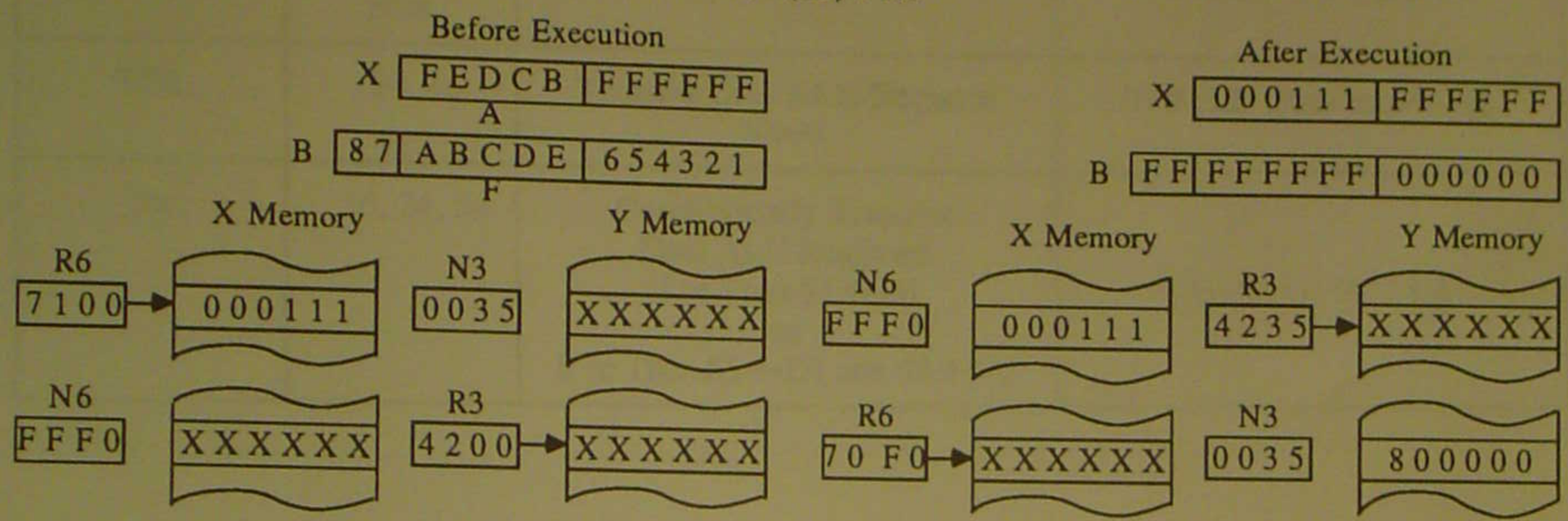
| | | "cc" Mnemonic | Condition |
|---|---|---|---|
| CC (HS) | - | carry clear (higher or same) | C=0 |
| CS (LO) | - | carry set (lower) | C=1 |
| EC | - | extension clear | E=0 |
| EQ | - | equal | Z=1 |
| ES | - | extension set | E=1 |
| GE | - | greater than or equal | $N \oplus V = 0$ |
| GT | - | greater than | $Z+(N \oplus V)=0$ |
| LC | - | limit clear | L=0 |
| LE | - | less than or equal | $Z+(N \oplus V)=1$ |
| LS | - | limit set | L=1 |
| LT | - | less than | $N \oplus V = 1$ |
| MI | - | minus | N=1 |
| NE | - | not equal | $\overline{Z}=0$ |
| NR | - | normalized | $Z+(\overline{U \cdot E})=1$ |
| PL | - | plus | $\overline{N}=0$ |
| NN | - | not normalized | $Z+(\overline{U \cdot E})=0$ |

MTA5- 5260

# TRANSFER EXAMPLES

TFR using parallel move option creates a three move instruction

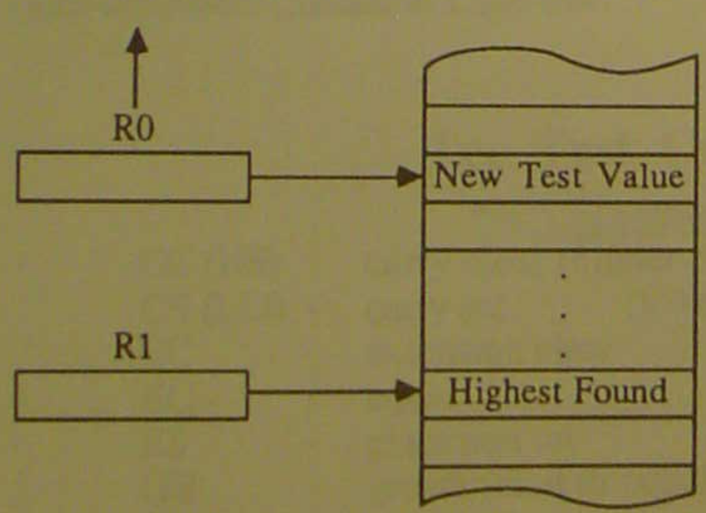
**Example 1:** TFR X0, B  X:(R6) + N6, X1  B,Y:(R3) + N3




Tcc used after CMP or CMPM can perform many useful functions like selectively Transfer

Maximum Value
Minimum Value
Maximum Absolute Value
Minimum Absolute Value

**Example 2:** Array of Data




```
CMP   X0,A
TLT   X0,A        R0,R1
```

MTA5- 5265-2

# ARITHMETIC INSTRUCTIONS

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| NEG | 56 | Negate<br>$0 - D \rightarrow D$ | NEG D [Parallel Move] |
| NORM | 16, 56 | Normalize Accumulator Iteration<br>If $\overline{E} \cdot U \cdot \overline{Z} = 1$,<br>then ASL D and Rn - 1 → Rn<br>Else If E = 1,<br>then ASR D and Rn + 1 → Rn<br>Else<br>NOP | NORM Rn,D<br><br>Parallel Move not allowed |
| RND | 56 | Round Accumulator<br>$D + r \rightarrow D$ | RND D [Parallel Move] |
| ABS | 56 | Absolute Value<br>$\lvert D \rvert \rightarrow D$ | ABS D [Parallel Move] |
| CLR | 56 | Clear Accumulator<br>$0 \rightarrow D$ | CLR D [Parallel Move] |

## Operand Combinations

| Destination | Register Rn |
|---|---|
| A,B | R0-R7 |

**Example 1:** NEG A

Before Execution

A | 00 | 000000 | 000001 |

After Execution

A | FF | FFFFFF | FFFFFF |

**Example 2:**

R0 | 0000 |

NORM R0,A

R0 | FFFF |

NORM R0,A

R0 | FFFE |

NORM R0,A

R0 | FFFE |

Accumulator A is normalized

A | 00 | 100000 | 000000 |

A | 00 | 200000 | 000000 |

A | 00 | 400000 | 000000 |

A | 00 | 400000 | 000000 |

MTA5-5270-1

# DIVIDE ITERATION

| MNEMONIC | OPERAND SIZE | OPERATION | ASSEMBLER SYNTAX |
|---|---|---|---|
| DIV | 24,56 | Divide Iteration<br>If D[55] ⊕ S[23] = 1,<br>then [55 47 ← | ← | 23 ← 0] ←C + S → D<br>Destination Accumulator D<br>else [55 47 ← | ← | 23 ← 0] ←C - S →D<br>Destination Accumulator D | DIV S,D |

**Operand Combinations**

| Source | Destinations |
|---|---|
| X1, X0<br>Y1, Y0 | A,B |

**Example 1:** 1 Quadrant division, 24-bit unsigned quotient, 48-bit unsigned remainder

```
AND #$FE,CCR        ;clear carry bit C (quotient sign bit)
REP #$18            ;form a 24-bit quotient and remainder
DIV X0,A            ;form quotient in A0, remainder in A1
ADD X0,A            ;restore remainder in A1
```

**Example 2:** (4 Quadrant division, 24-bit signed quotient, 48-bit signed remainder)

```
          ABS A      A,B          ;make dividend positive, copy A1 to B1
          EOR X0,B B,X:$0         ;save rem. sign in X:$0, quo. sign in N
          AND #$FE,CCR            ;clear carry bit C (quotient sign bit)
          REP #$18                ;form a 24-bit quotient
          DIV X0,A                ;form quotient in A0, remainder in A1
          TFR A,B                 ;save quotient and remainder in B1,B0
          JPL SAVEQUO             ;go to SAVEQUO if quotient is positive
          NEG B                   ;complement quotient if N bit set
SAVEQUO   TFR X0,B B0,X1          ;save quo. in X1, get signed divisor
          ABS B                   ;get absolute value of signed divisor
          ADD A,B                 ;restore remainder in B1
          JCLR #23,X:$0,DONE      ;go to DONE if remainder is positive
          MOVE #$0,B0             ;clear LS 24 bits of B
          NEG B                   ;complement remainder if negative
DONE      .........
```

**Before Execution**

| | | |
|---|---|---|
| X | 000000 | 123456 |
| A | 00 0E66D7 | F2832C |
| B | 00 000000 | 000000 |

**After Execution**

| | | |
|---|---|---|
| X | 654321 | 123456 |
| A | FF EDCCAA | 654321 |
| B | 00 000100 | 654321 |

MTA5-5280-1

# AND, OR, EOR AND NOT LOGICAL INSTRUCTIONS

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| AND | 24 | Logical AND<br>$S \cdot D[47:24] \rightarrow D[47:24]$ | AND S, D [Parallel Move] |
| OR | 24 | Logical Inclusive OR<br>$S + D[47:24] \rightarrow D[47:24]$ | OR S, D [Parallel Move] |
| EOR | 24 | Logical Exclusive OR<br>$S \oplus D[47:24] \rightarrow D[47:24]$ | EOR S, D [Parallel Move] |
| NOT | 24 | Logical Complement<br>$\overline{D[47:24]} \rightarrow D[47:24]$ | NOT D [Parallel Move] |

## Operand Combinations

| Source | Destination |
|---|---|
| X1, X0 | A, B |
| Y1, Y0 | |

**Example 1:** AND Y0, B

Before Execution

Y | 123456 | F84210

B | FF | 0FFFFF | FFFFFF

After Execution

Y | 123456 | F84210

B | FF | 084210 | FFFFFF

**Example 2:** EOR X1, A

Before Execution

X | F84210 | 123456

A | FF | FFFFFF | FFFFFF

After Execution

X | F84210 | 123456

A | FF | 07BDEF | FFFFFF

MTA5-5310

# IMMEDIATE DATA LOGICAL INSTRUCTIONS

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| ANDI | 8 | AND Immediate with Control Register<br>#xx • D→D | AND(I) #xx, D |
| ORI | 8 | OR Immediate with Control Register<br>#xx + D→D | OR(I) #xx, D |

**Operands**
**Destinations**

MR
CCR
OMR

Does not support parallel moves.

**Example 1:** ANDI #%11111100, OMR

Puts processor in Mode 0
Does not change other bits

**Example 2:** ORI #%00000010, OMR

Sets MB Bit, all others are unchanged

# LOGICAL SHIFT INSTRUCTIONS

| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| LSL | 24 | Logical Shift Accumulator Left<br>C ← [47 ← 24] ← 0 | LSL D [Parallel Move] |
| ROL | 24 | Rotate Accumulator Left<br>C ← [47 ← 24] ← C | ROL D [Parallel Move] |
| LSR | 24 | Logical Shift Accumulator Right<br>0 → [47 → 24] → C | LSR D [Parallel Move] |
| ROR | 24 | Rotate Accumulator Right<br>C → [47 → 24] → C | ROR D [Parallel Move] |

**Operands**
**Destination**
A, B

**Example 1:** LSL A

Before Execution

| | | | |
|---|---|---|---|
| A | 80 | 822222 | 111111 |

Carry [0]

After Execution

| | | | |
|---|---|---|---|
| A | 80 | 044444 | 111111 |

Carry [1]

**Example 2:** ROR B

Before Execution

| | | | |
|---|---|---|---|
| B | F0 | 00000E | F0000F |

Carry [1]

After Execution

| | | | |
|---|---|---|---|
| B | F0 | 800007 | F0000F |

Carry [0]

MTA5- 5330-1

# BRANCHING INSTRUCTIONS

| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| JMP | Jump<br><ea> → PC | JMP <ea> |
| Jcc | Conditional Jump<br>If cc then <ea> → PC<br>Else PC+1 → PC | Jcc <ea> |
| JSR | Jump to Subroutine<br>SP + 1 → SP<br>PC → SSH<br>SR → SSL<br><ea> → PC | JMP <ea> |
| JScc | Conditional Jump to Subroutine<br>If cc then<br>SP + 1 → SP<br>PC → SSH<br>SR → SSL<br><ea> → PC<br>Else<br>PC + 1 → PC | JScc <ea> |
| RTS | Return From Subroutine<br>SSH → PC<br>SP - 1 → SP | RTS |

**Jump Effective Address MODES:**

| | |
|---|---|
| Absolute Address | xxxx |
| Short Jump Address | xxx |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1: | (Rn)+ |
| Postdecrement by 1: | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

**Jump Conditions**

| "cc" Mnemonic | | Condition |
|---|---|---|
| CC (HS) | carry clear (higher or same) | C=0 |
| CS (LO) | carry set (lower) | C=1 |
| EC | extension clear | E=0 |
| EQ | equal | Z=1 |
| ES | extension set | E=1 |
| GE | greater than or equal | $N \oplus V=0$ |
| GT | greater than | $Z+(N \oplus V)=0$ |
| LC | limit clear | L=0 |
| LE | less than or equal | $Z+(N \oplus V)=1$ |
| LS | limit set | L=1 |
| LT | less than | $N \oplus V=1$ |
| MI | minus | N=1 |
| NE | not equal | Z=0 |
| NR | normalized | $Z+(\overline{U} \cdot \overline{E})=1$ |
| PL | plus | N=0 |
| NN | not normalized | $Z+(\overline{U} \cdot \overline{E})=0$ |

MTA5- 5410-1

# BIT MANIPULATION INSTRUCTIONS

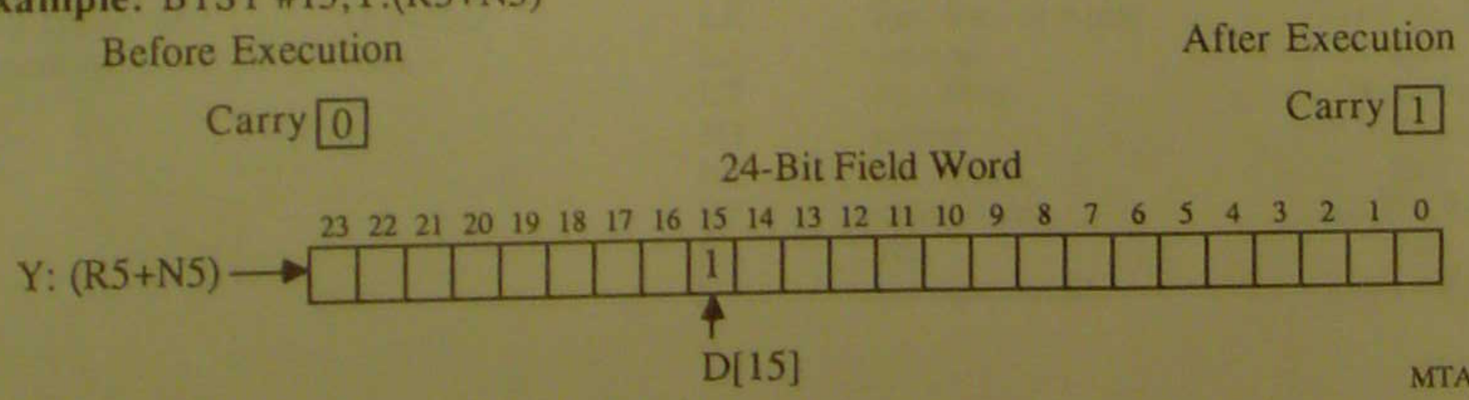
| Mnemonic | Operand Size | Operation | Assembler Syntax |
|---|---|---|---|
| BCLR * | 24 | Bit Test and Clear<br>D[n] → C<br>0 → D[n] | BCLR #n, X:<ea><br>BCLR #n, Y:<ea> |
| BSET * | 24 | Bit Test and Set<br>D[n] → C<br>1 → D[n] | BSET #n, X:<ea><br>BSET #n, Y:<ea> |
| BCHG * | 24 | Bit Test and Change<br>D[n] → C<br>$\overline{D[n]}$ → D[n] | BCHG #n, X:<ea><br>BCHG #n, Y:<ea> |
| BTST | 24 | Bit Test<br>D[n] → C | BTST #n, X:<ea><br>BTST #n, Y:<ea> |

* Generates Indivisible Read-Modify-Write Bus Cycle

D[n] is the $n^{th}$ bit of the 24 bit destination operand field. $0 \le n \le 23$

| X or Y Effective Address MODES | |
|---|---|
| Absolute Address | xxxx |
| Absolute Short Address | aa |
| I/O Short Address | pp |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1: | (Rn)+ |
| Postdecrement by 1: | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

**Example:** BTST #15,Y:(R5+N5)




MTA5-5510-1

# BIT FIELD BRANCHING INSTRUCTIONS

| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| JSET | Jump if Bit Set<br>If S[n] =1<br>Then xxxx → PC<br>Else PC+1 → PC | JSET #n,X:<ea>, xxxx<br>JSET #n,Y:<ea>, xxxx |
| JCLR | Jump if Bit Clear<br>If S[n] =0<br>Then xxxx → PC<br>Else PC+1 → PC | JCLR #n,X:<ea>, xxxx<br>JCLR #n,Y:<ea>, xxxx |
| JSSET | Jump to Subroutine if Bit Set<br>If S[n] =1<br>Then SP+1 → SP<br>PC → SSH<br>SR → SSL<br>xxxx → PC<br>Else PC+1 → PC | JSSET #n,X:<ea>, xxxx<br>JSSET #n,Y:<ea>, xxxx |
| JSCLR | Jump to Subroutine if Bit Clear<br>If S[n] =0<br>Then SP+1 → SP<br>PC → SSH<br>SR → SSL<br>xxxx → PC<br>Else PC+1 → PC | JSCLR #n,X:<ea>, xxxx<br>JSCLR #n,Y:<ea>, xxxx |

**X or Y Effective Address MODES:**

| | |
|---|---|
| Absolute Short Address | aa |
| I/O Short Address | pp |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1: | (Rn)+ |
| Postdecrement by 1: | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

MTA5- 5610-1

# MISCELLANEOUS INSTRUCTIONS

| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| NOP | No Operation<br>PC+1 → PC | NOP |
| RESET | Reset on Chip Peripheral Devices<br>1 → I1<br>1 → I0<br>0 → X:$FFFF<br>PC+1 → PC | RESET |
| WAIT | Disable Clocks to Processor Core and Enter the Wait Processing State | WAIT |
| STOP | Enter the Stop Processing State and Stop the Clock Oscillator | STOP |

# EXERCISE 8

## GENERAL INSTRUCTIONS

1. Accumulator A contains a nonzero number.
   Write a program that will normalize Accumulator A.

2. Write a program that clears the on chip RAM in X and Y memory.

3. Write a program that finds the largest positive value in the Y memory on chip Data ROM.

# EXCEPTIONS

## OBJECTIVES

After studying this section, the student should be able to:

- ▣ Describe the Interrupt Sources.
- ▣ Describe the Interrupt Types.
- ▣ Set the Interrupt Mask.
- ▣ Adjust the Interrupt Priority.

# DSP56000 PROCESSING STATES

- Normal
  - Instruction Execution
- Reset
  - Assertion of $\overline{\text{RESET}}$ Pin.
- Wait
- Stop
- Exception
  - External Interrupts
    - IRQA
    - IRQB
  - Internal
    - Software Interrupt
    - on Chip Peripheral
    - Error Condition
    - Tracing

# VECTOR TABLE OF EXCEPTION SOURCES

| Exception Starting Address | Program Memory Space<br>Exception Source | |
|---|---|---|
| $0000 | Hardware $\overline{\text{RESET}}$ | Two Words Per Vector |
| $0002 | Stack Error | |
| $0004 | Trace | |
| $0006 | SWI (Software Interrupt) | |
| $0008 | $\overline{\text{IRQA}}$ External Hardware Interrupt | External Interrupts |
| $000A | $\overline{\text{IRQB}}$ External Hardware Interrupt | |
| $000C | SSI Receive Data | Synchronous Serial Interface — Internal Interrupts |
| $000E | SSI Receive Data with Exception Status | |
| $0010 | SSI Transmit Data | |
| $0012 | SSI Transmit Data with Exception Status | |
| $0014 | SCI Receive Data | Synchronous Communications Interface — Internal Interrupts |
| $0016 | SCI Receive Data with Exception Status | |
| $0018 | SCI Transmit Data | |
| $001A | SCI Idle Line | |
| $001C | SCI Timer | |
| $001E | Reserved for Hardware Development | |
| $0020 | Host Receive Data | Host Interface — Internal Interrupts |
| $0022 | Host Transmit Data | |
| $0024 | Host Command (Default) | |
| $0026 | Available for Host Command | |
| $0028 | Available for Host Command | |
| $002A | Available for Host Command | |
| $002C | Available for Host Command | |
| $002E | Available for Host Command | |
| $0030 | Available for Host Command | |
| $0032 | Available for Host Command | |
| $0034 | Available for Host Command | |
| $0036 | Available for Host Command | |
| $0038 | Available for Host Command | |
| $003A | Available for Host Command | |
| $003C | Available for Host Command | |
| $003E | Available for Host Command | |

MTA5-6110-1

# RESET

**Assertion of $\overline{\text{RESET}}$**

RESET → Port B & C, SCI, SSI, HOST

1. Reset on chip peripherals.

2. Set Modifier Registers to $FFFF.

| | |
|---|---|
| M0 | $FFFF |
| M1 | $FFFF |
| M2 | $FFFF |
| M3 | $FFFF |
| M4 | $FFFF |
| M5 | $FFFF |
| M6 | $FFFF |
| M7 | $FFFF |

3. Clear Interrupt Priority Register.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X:$FFFF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | SCI | | SSI | | HOST | | RESERVED | | | | IRQB | | | IRQA | | |

4. Set Bus Control Register to $FFFF.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X:$FFFE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | X Memory | | | | Y Memory | | | | P Memory | | | | Peripherals | | | |

5. Clear the Stack Pointer.

| 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| UF | SE | P3 | P2 | P1 | P0 |

6. Initialize Status Register.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | E | U | N | Z | V | C |
| LF | | T | | S1 | S0 | I1 | I0 | | L | | | | | | |

7. Clear the Data ROM Enable Bit and the Memory Ready Enable list in the OMR Register.

8. Stay in RESET until negated.

**Negation of $\overline{\text{RESET}}$**

MODB MODA

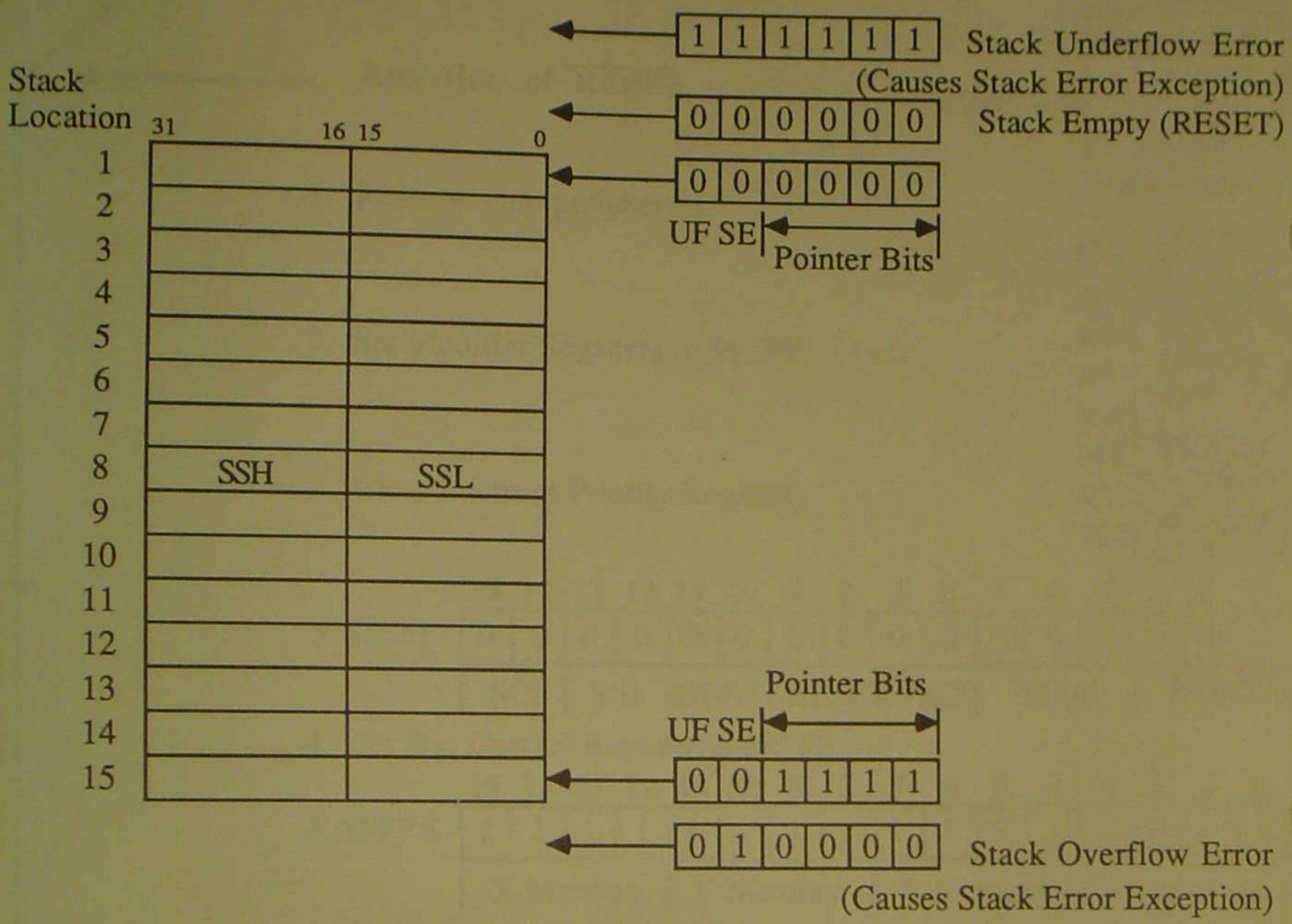
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | X | X |
| ME | | | | | DE | | |

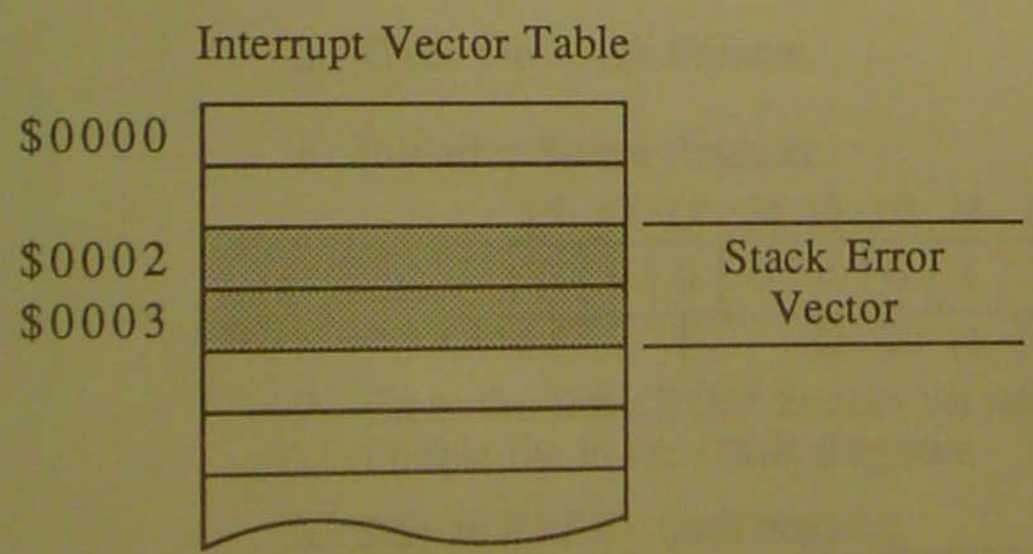
9. Load Operating Mode Register from Mode Pins.

10. Start normal execution:
    If Mode 2 P:$E000
    Else P: $0000
    **JMP START**

MTA5-6120-1

# STACK ERROR




## Characteristics of Stack Errors

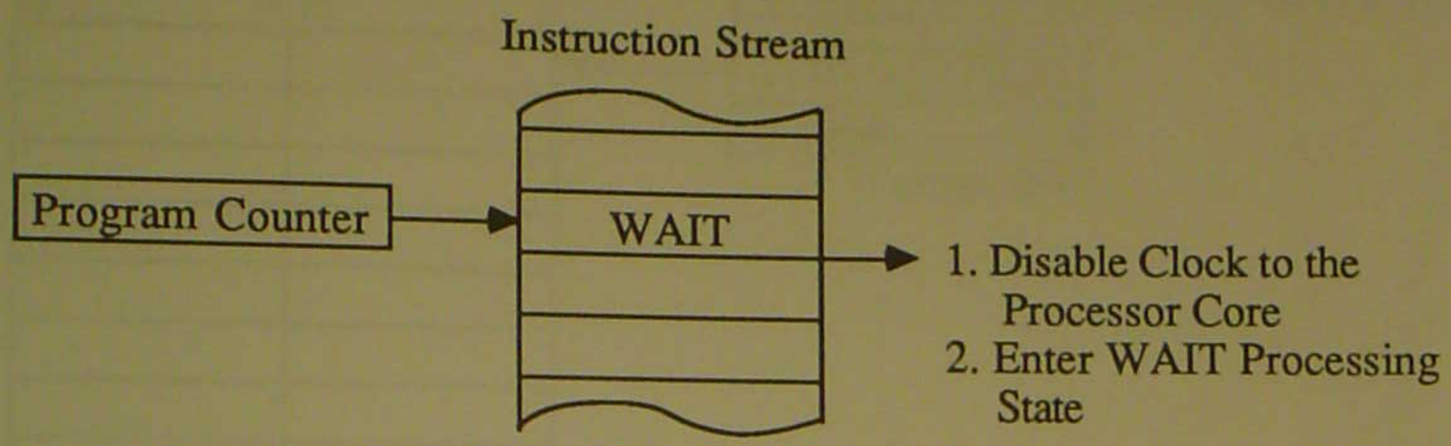
Occurs on Stack Overflow or Underflow.

Stack Error Bit (SE) changing from 0 to 1 creates a non maskable level 3 Stack Error Exception.

MTA5-6130

# WAIT EXCEPTION

## Entering Wait State



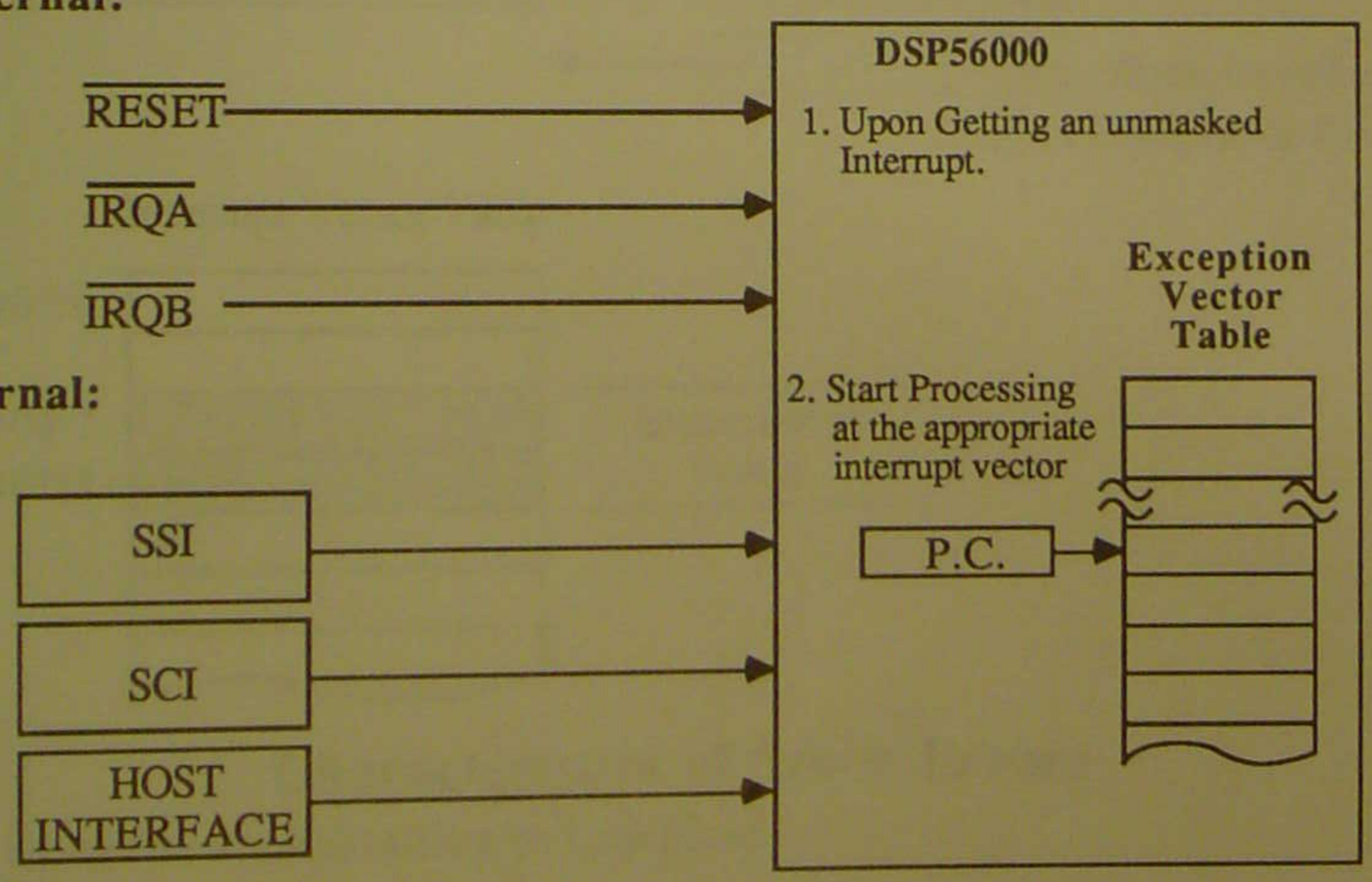

## Exiting Wait State

* RESET

* Any Unmasked Internal or External Hardware Interrupt.

**External:**

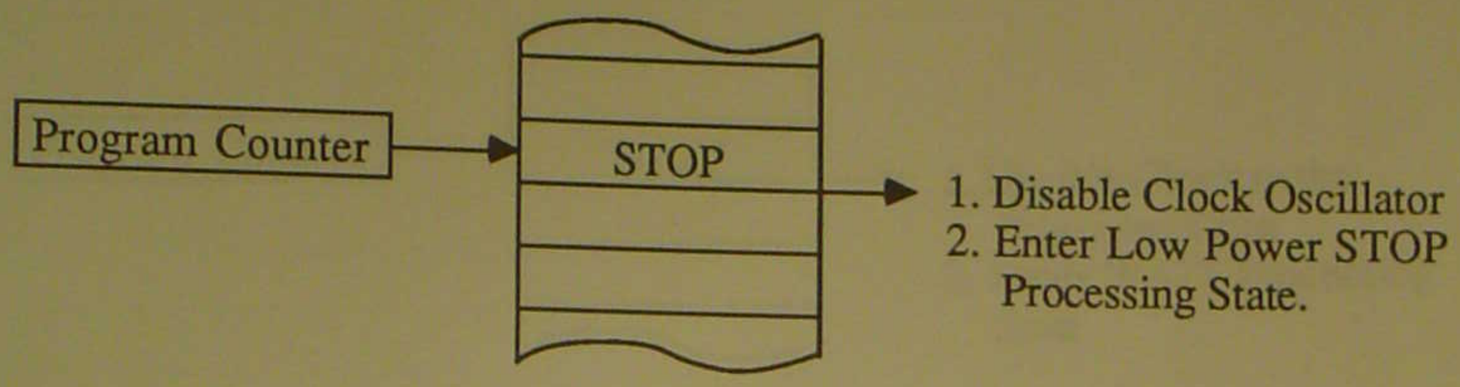
**Internal:**



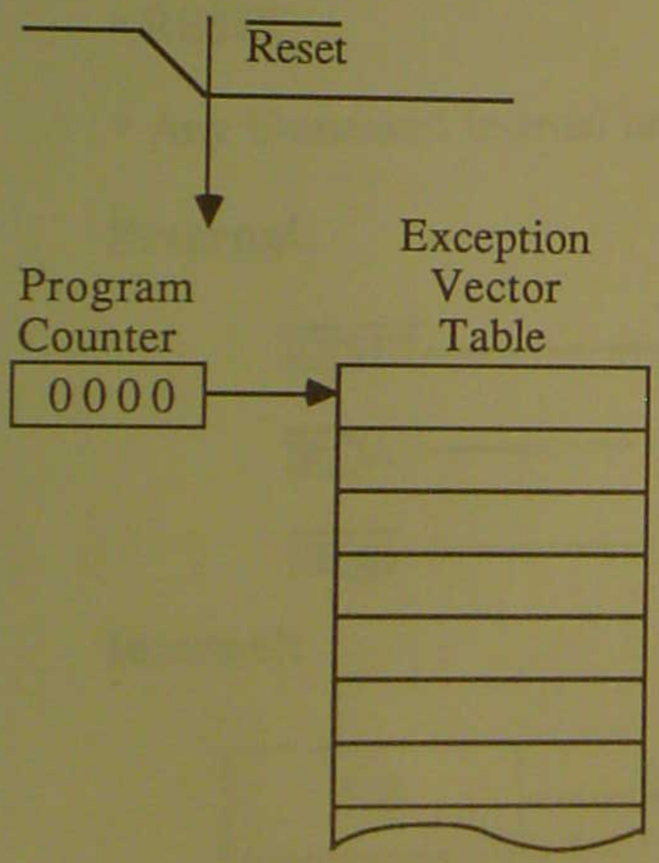

MTA5-6140

# STOP EXCEPTION

## Entering Stop State

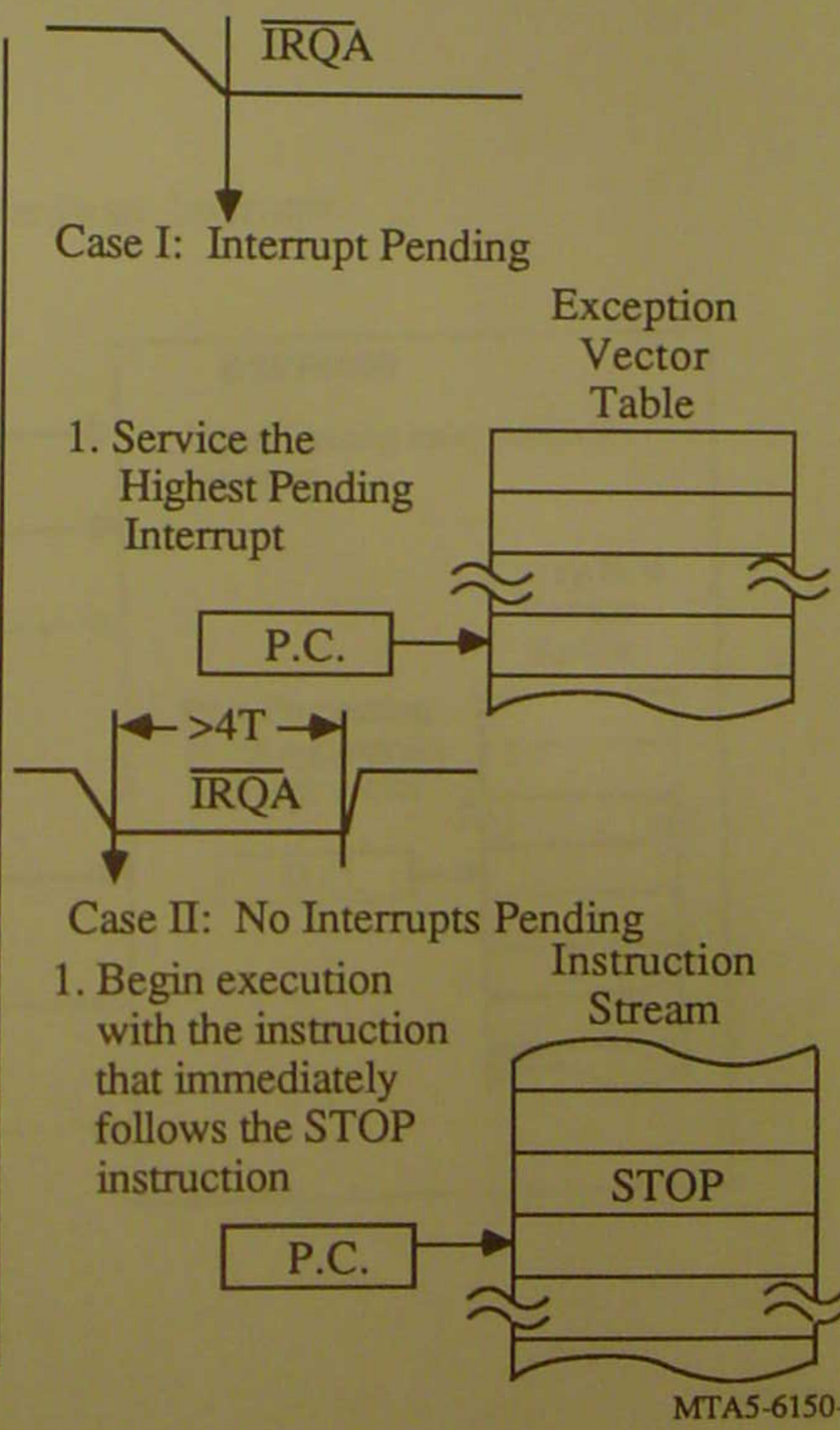



## Exiting From Stop State




1. Enable Clock and Peripherals.

2. Start RESET Exception Processing.




MTA5-6150-1

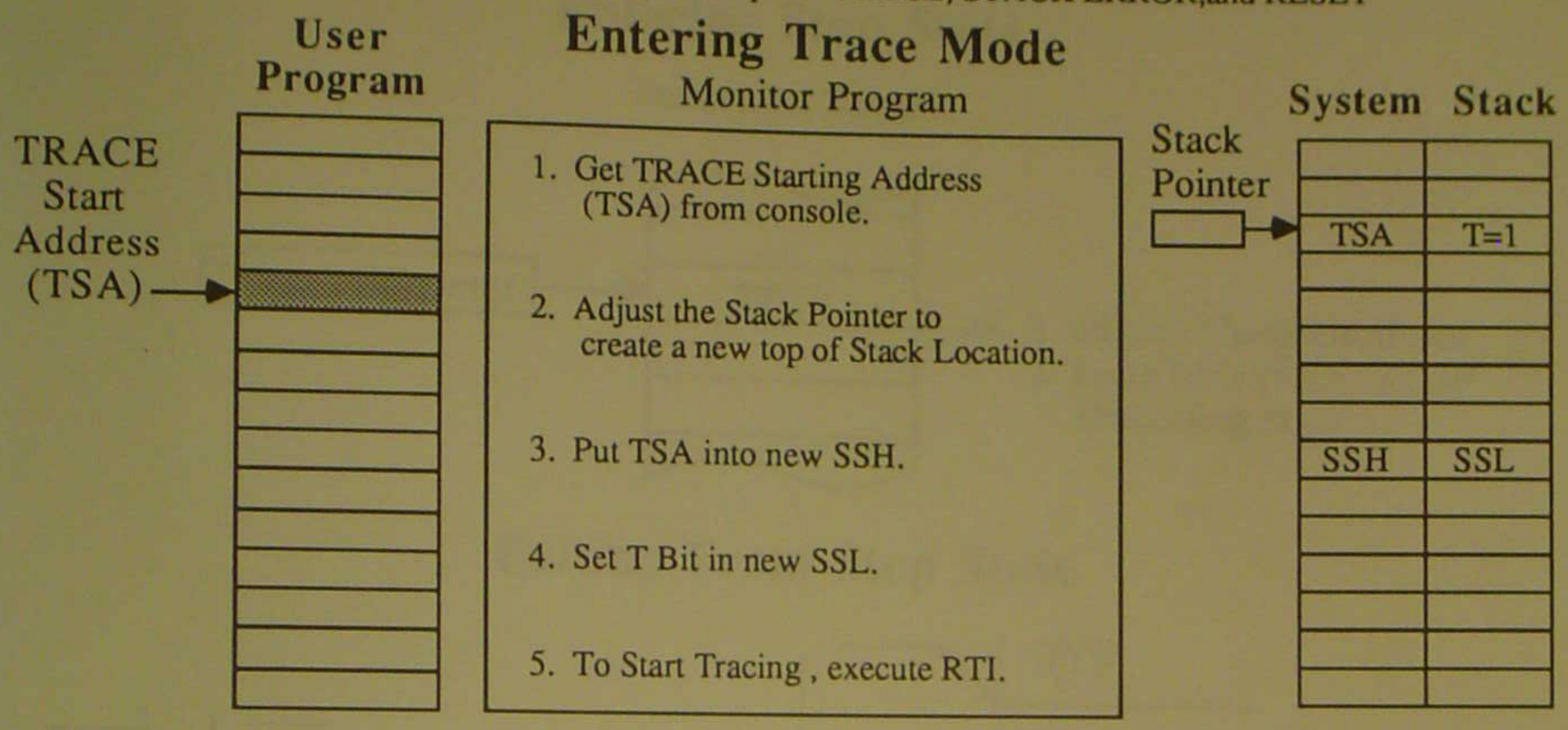
# TRACE MODE

Useful for Debugging

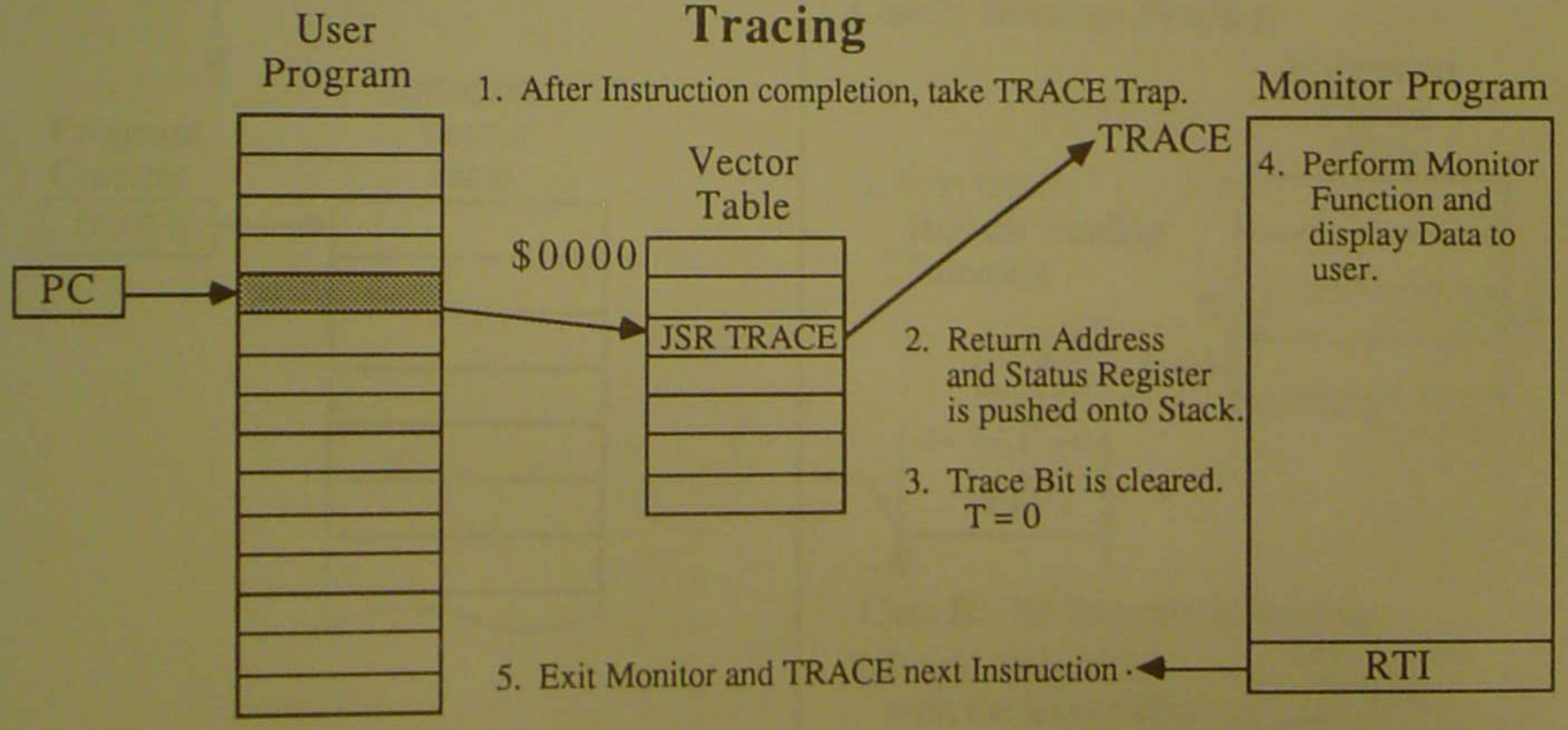
Traps out to Trace Exception Vector after execution of each instrcution

Masks out all external and peripheral Interrupts except for TRACE, STACK ERROR,and RESET

## Entering Trace Mode




## Tracing



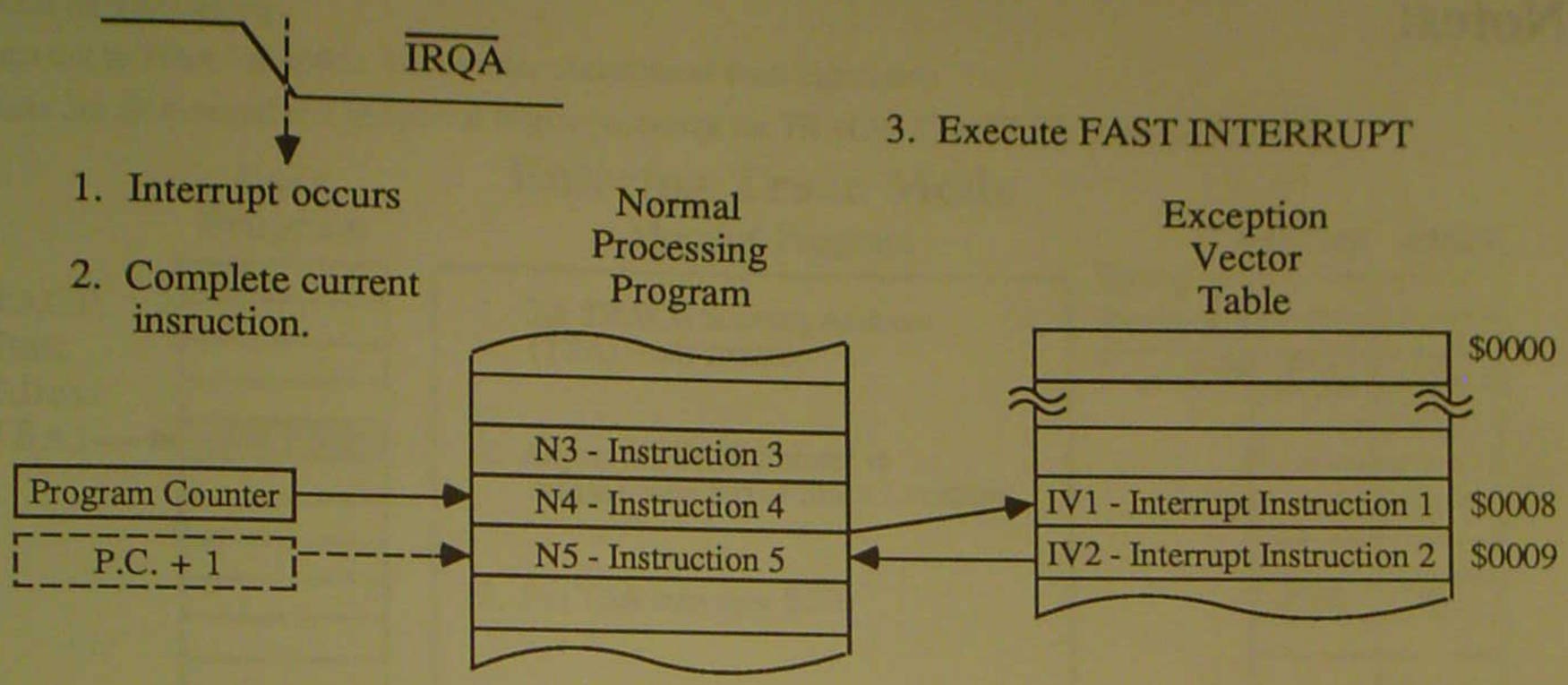

## Characteristics of TRACE Exceptions

TRACE is a NON Maskable level 3 exception .

Fast Interrupt Routines cannot be traced.

MTA5-6160-1

# FAST INTERRUPT




## Characteristics of Fast Interrupts:

Fast interrupts consist of a single two word instruction, or two, one word instructions

The Program Counter is never updated

Normal Instruction fetching resumes using the Program Counter following the completion of the Fast Interrupt Routine

It is NOT interruptable, even by an interrupt of a higher priority

Status is NOT preserved. Caution should be taken when using instructions which modify the Status Register.

A Fast interrupt MUST NOT use a "Change of Flow" instruction

A jump to subroutine instruction within the fast interrupt routine consitutes a Long Interrupt Routine.

## Characteristics of Fast Interrupts

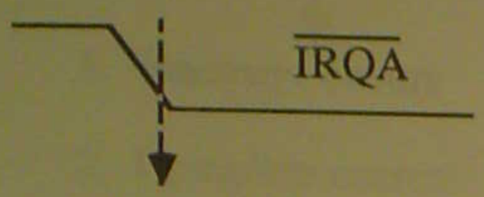
Minimum Interrupt Cycle Time

| INSTRUCTION CYCLE | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| FETCH | N3 | N4 | IV1 | IV2 | N5 | N6 | N7 | N8 | N9 | N10 |
| DECODE | N2 | N3 | N4 | IV1 | IV2 | N5 | N6 | N7 | N8 | N9 |
| EXECUTE | N1 | N2 | N3 | N4 | IV1 | IV2 | N5 | N6 | N7 | N8 |

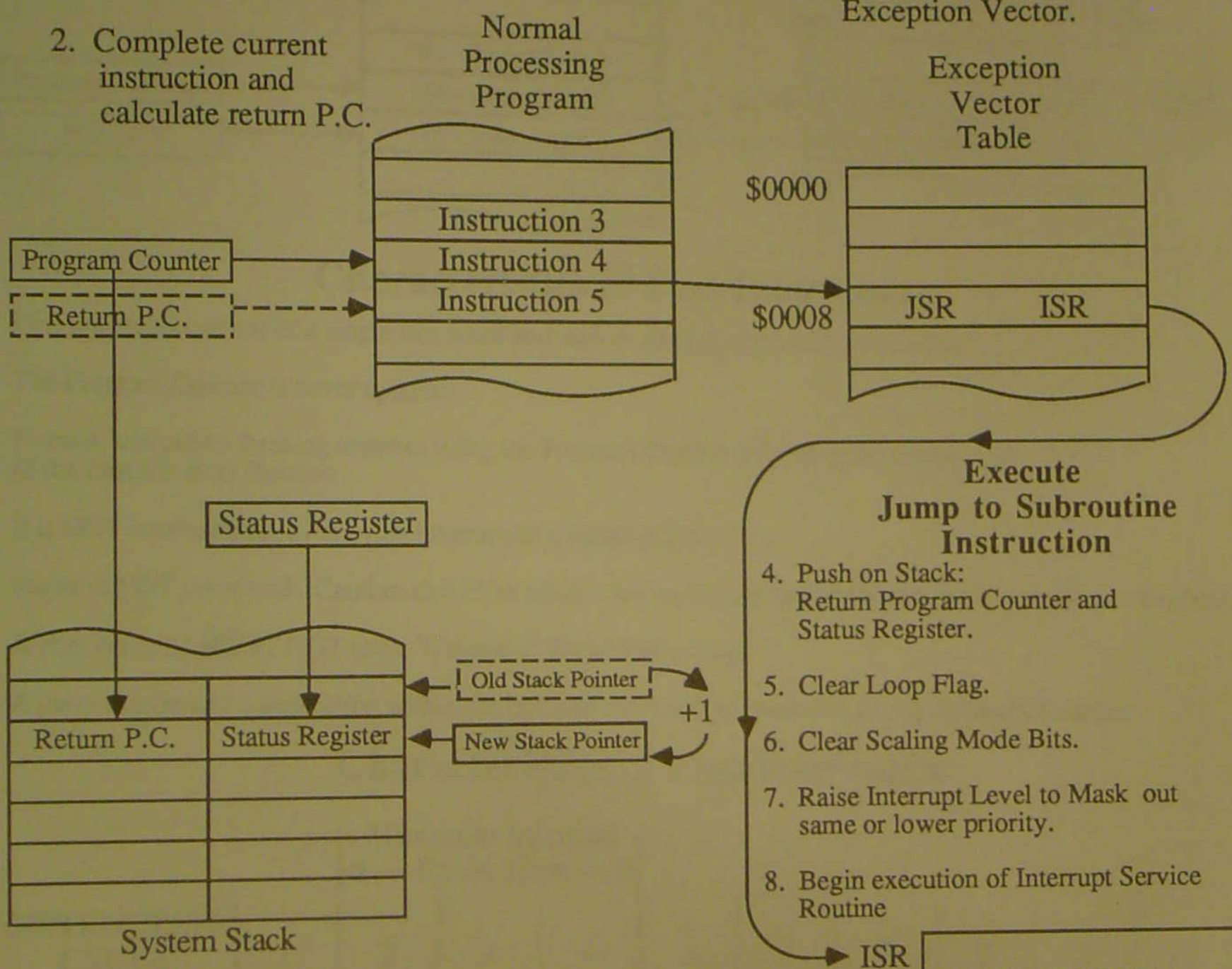
Nn - Normal Instruction Word
IVn - Interrupt Vector Word (NO Change of Flow)

MTA5-6200

# LONG INTERRUPT




1. Interrupt occurs
2. Complete current instruction and calculate return P.C.
3. Start Program Execution at Exception Vector.

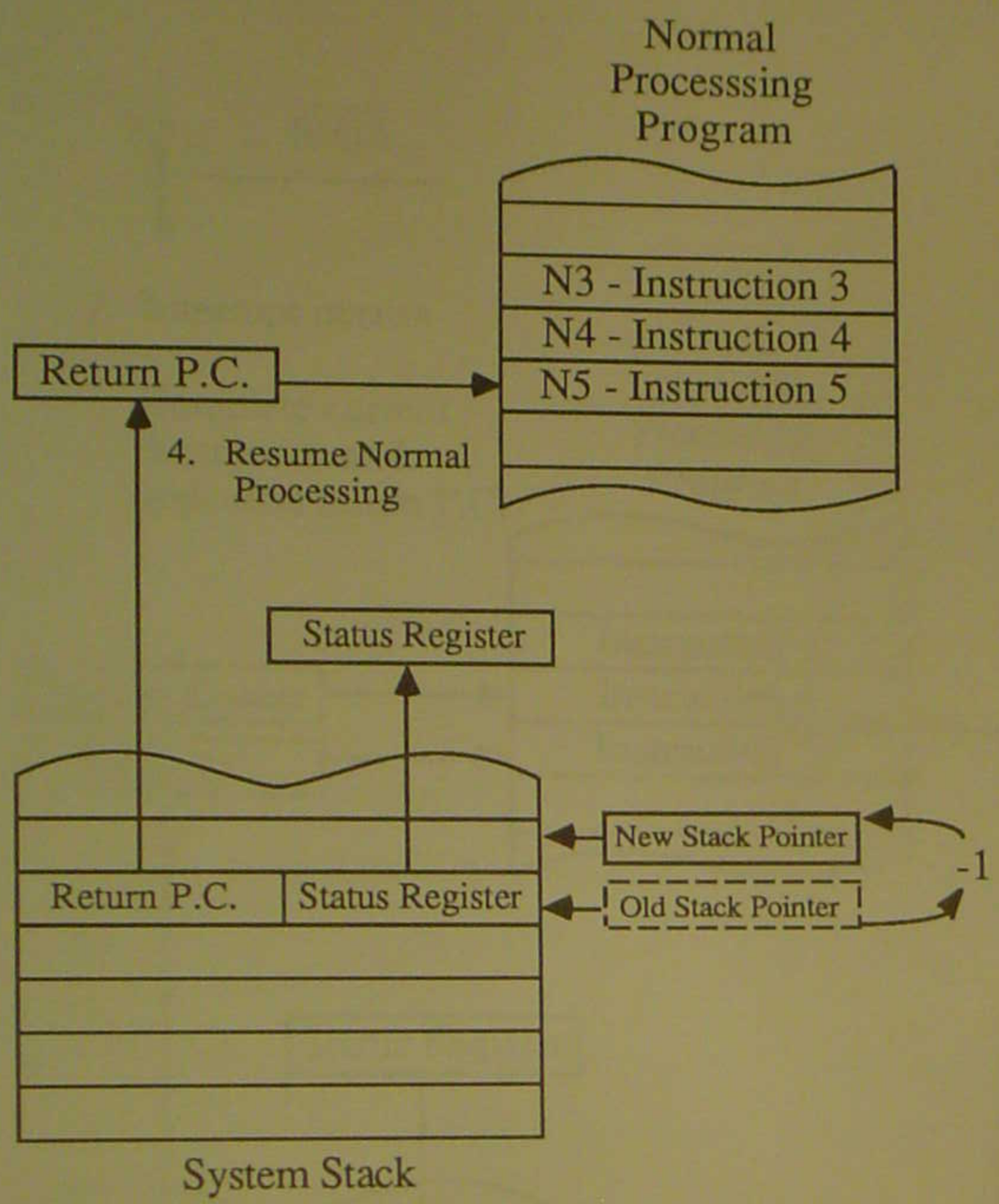


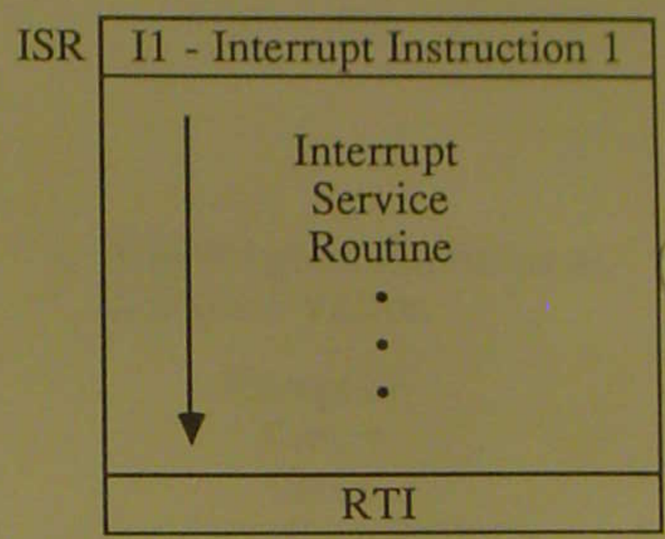

## Characteristics of Long Interrupts

- Entered when a "Change of Program Flow" instruction is used in the Exception Vector Table. i.e. JSR.
- Program Counter is updated to the Return Address.
- Interrupt Service Routine is terminated with a RTI.

MTA5-6210

# RTI – RETURN FROM INTERRUPT




**Execute RTI Instruction**

1. Restore Program Counter
2. Restore Status Register
   - Loop Flag
   - Trace Bit
   - Scaling Mode Bits
   - Interrupt Level
   - Condition Code
3. Decrement Stack Pointer

## Long Interrupt Instruction Timing

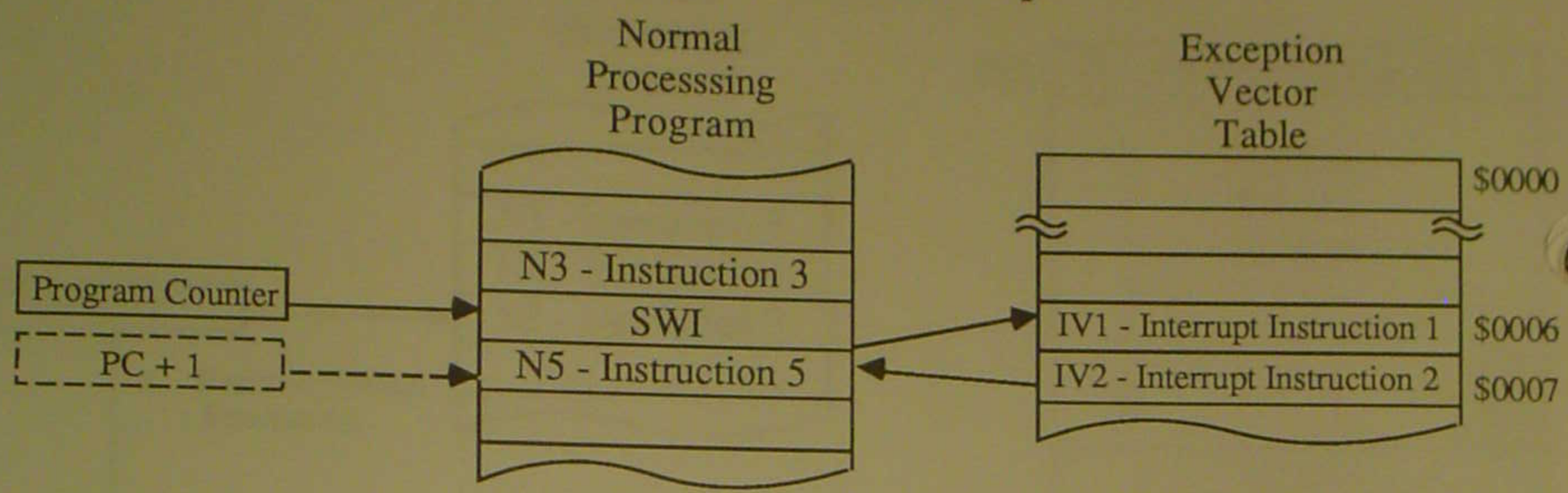
| INSTRUCTION CYCLE | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| FETCH | N3 | N4 | IV1 | IV2 | I1 | I2 | I3 | N5 | N6 | N7 |
| DECODE | N2 | N3 | N4 | JSR | ISR | I1 | RTI | X | N5 | N6 |
| EXECUTE | N1 | N2 | N3 | N4 | JSR | ISR | I1 | RTI | X | N5 |

Nn - Normal Instruction Word
IVn - Interrupt Vector Word (Change of Flow)
JSR - Jump to Interrupt Service Routine Instruction
I1 - Interrupt Service Routine Instruction(s)
RTI - Return from Interrupt Instruction
X - RTI Execution Time

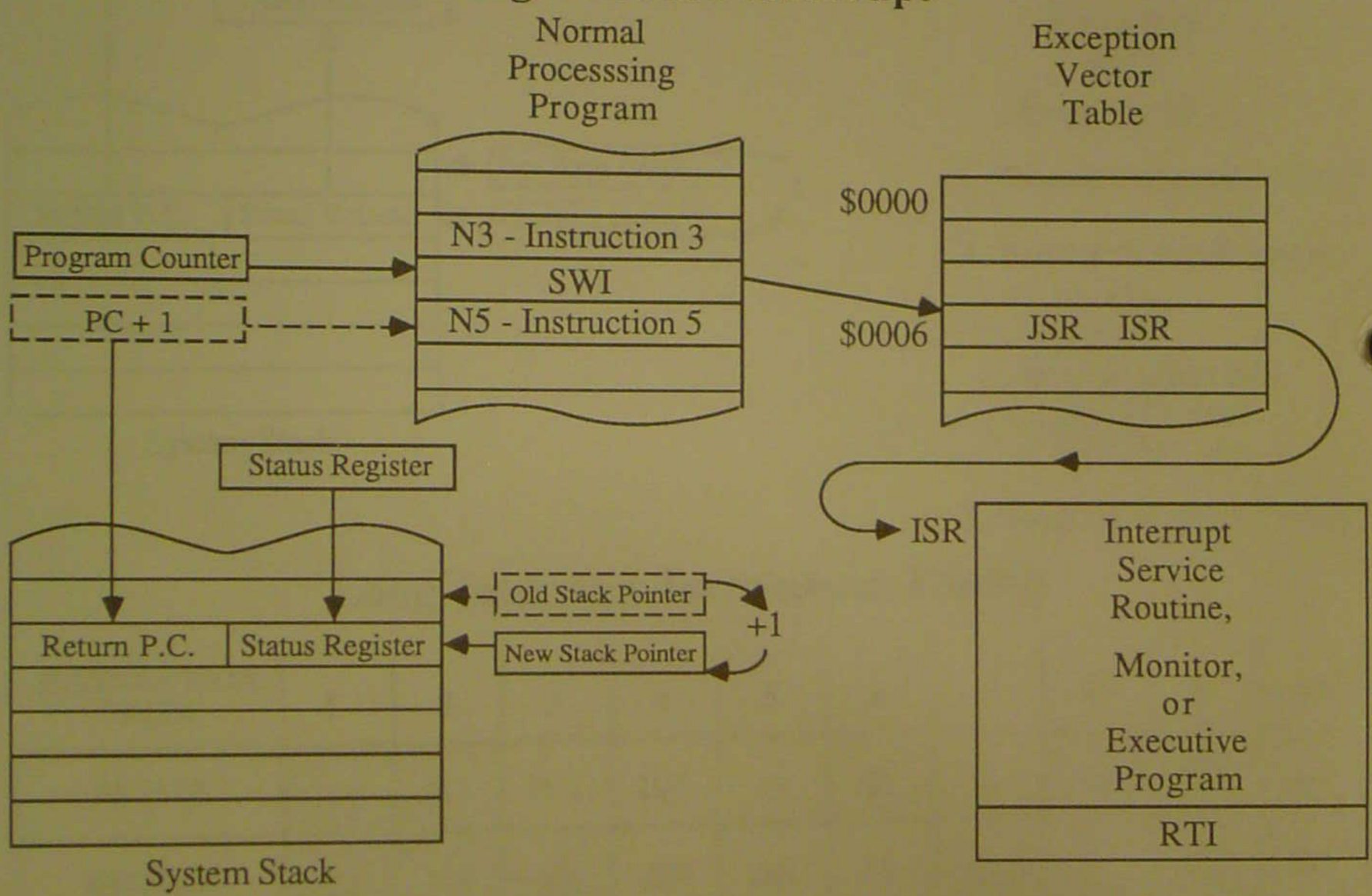
MTA5-6220

# SOFTWARE INTERRUPT

## Fast Software Interrupt




## Long Software Interrupt




## Characteristics of Software Interrupts

SWI is Non Maskable and is serviced immediately after execution of the SWI instruction.

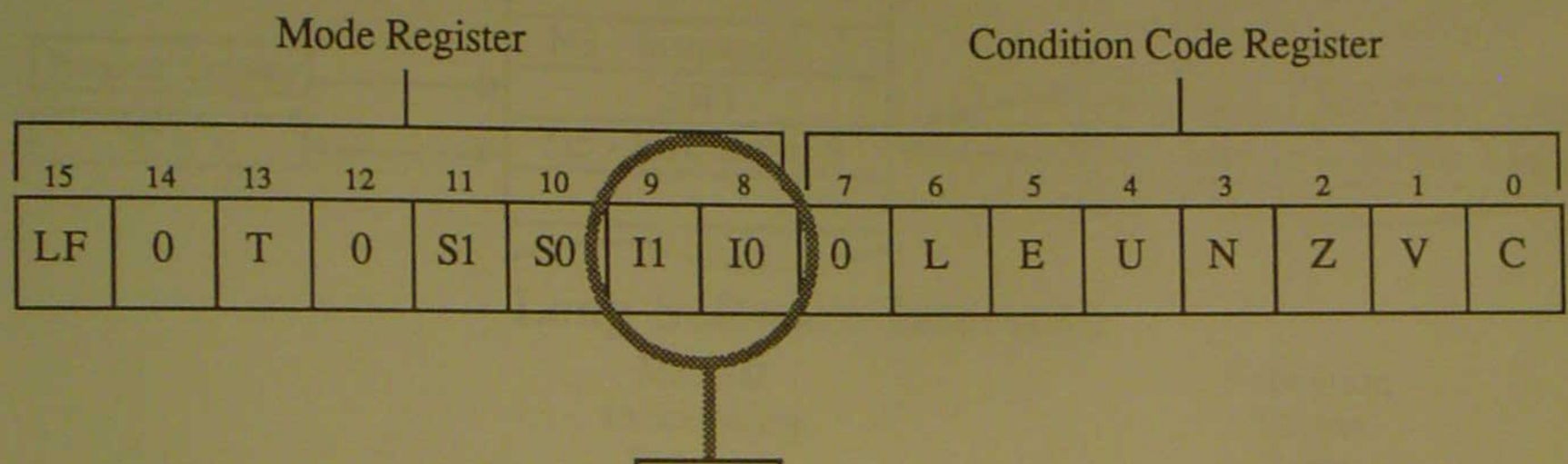
SWI sets the Interrupt Priority to level 3.

Useful for System Calls to Executive Routines.

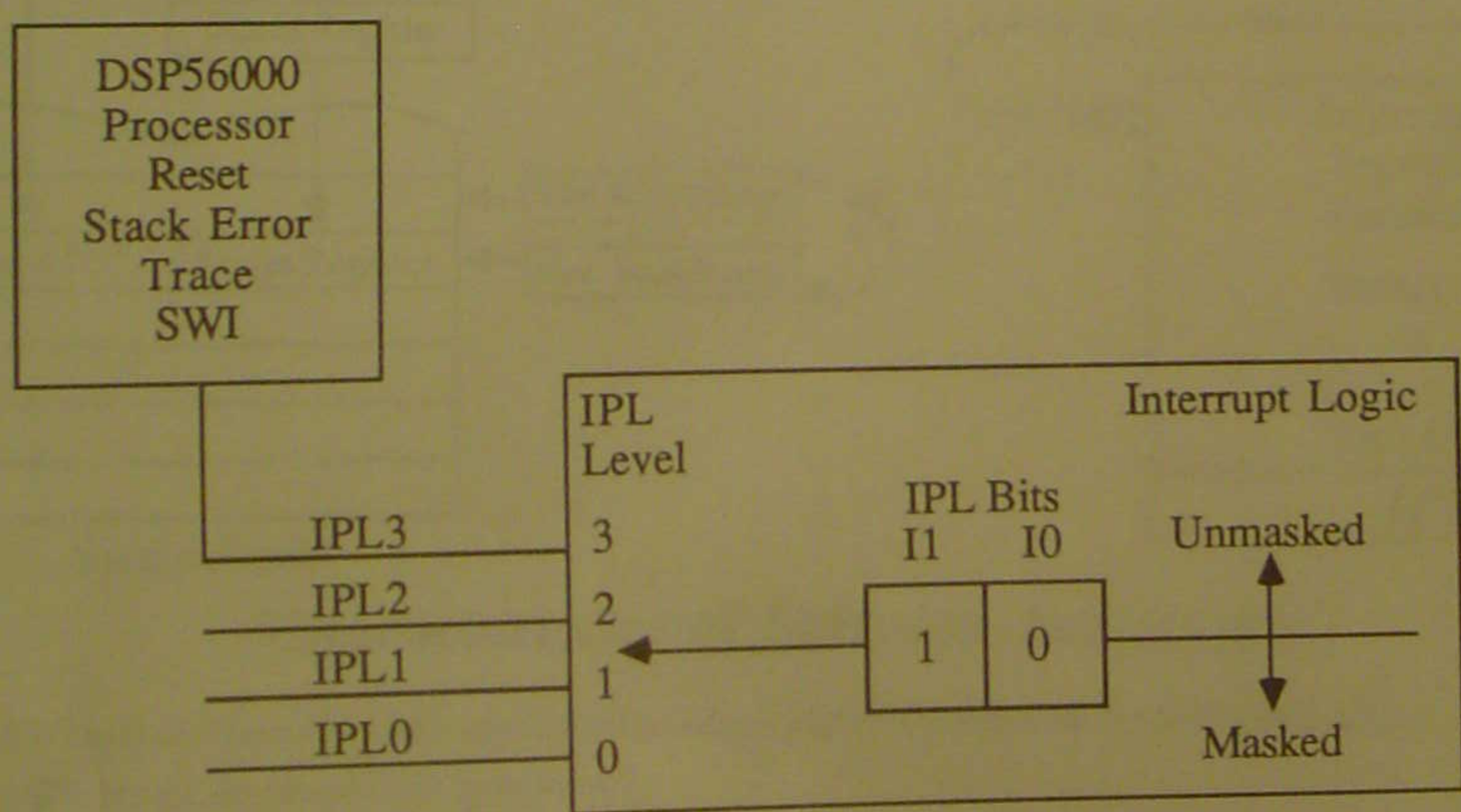
MTA5-6230

# INTERRUPT PRIORITY LEVEL MASK BITS

## Status Register




| Priority | I1 | I0 | Exceptions Permitted | Exceptions Masked |
|---|---|---|---|---|
| Low | 0 | 0 | IPL 0, 1, 2, 3 | None |
| ↓ | 0 | 1 | IPL 1, 2, 3 | IPL 0 |
| | 1 | 0 | IPL 2, 3 | IPL 0, 1 |
| High | 1 | 1 | IPL 3 | IPL 0, 1, 2 |




MTA5-6300

# INTERRUPT PRIORITY REGISTER

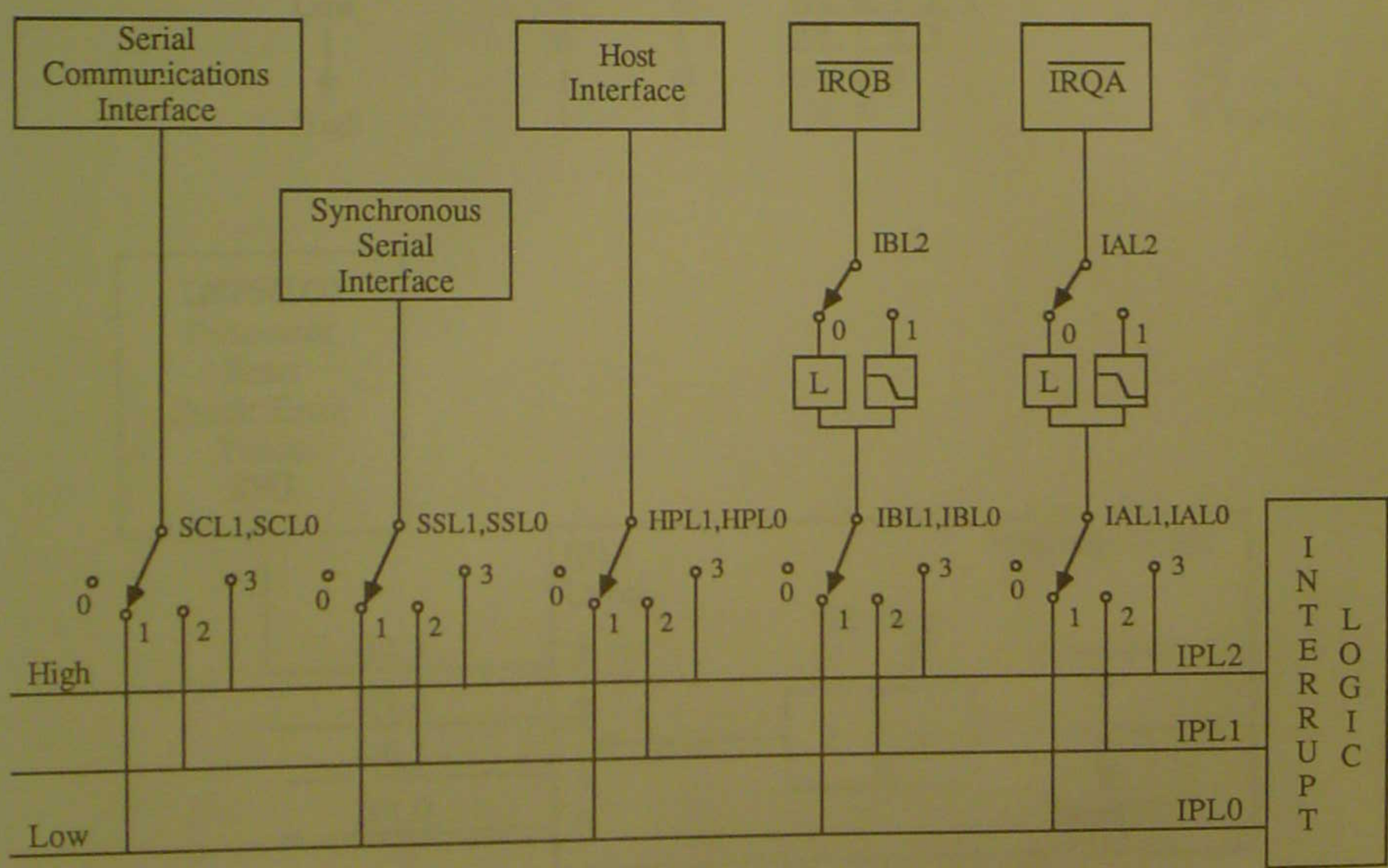
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X:$FFFF | SCL1 | SCL0 | SSL1 | SSL0 | HPL1 | HPL0 | 0 | 0 | 0 | 0 | IBL2 | IBL1 | IBL0 | IAL2 | IAL1 | IAL0 |
| | SCI IPL | | SSI IPL | | Host IPL | | RESERVED | | | | IRQB Mode | | | IRQA Mode | | |

## Interrupt Priority Level Bits

| Priority | xxL1 | xxL0 | Enabled | Interrupt Priority Level (IPL) |
|---|---|---|---|---|
| Low | 0 | 0 | No | Masked |
| ↓ | 0 | 1 | Yes | 0 |
| | 1 | 0 | Yes | 1 |
| High | 1 | 1 | Yes | 2 |

## External Interrupt Trigger Mode Bits

| IAL2 or IBL2 | Trigger Mode |
|---|---|
| 0 | Level |
| 1 | Negative Edge |




MTA5-6400

# EXCEPTION PRIORITIES WITHIN AN IPL

| Priority | Exception | Enabled by | Interrupt Priority Levels |
|---|---|---|---|
| highest | Hardware $\overline{\text{RESET}}$ | Non Maskable | 3 |
| ↑ | Stack Error | Non Maskable | 3 |
| | Trace | Non Maskable | 3 |
| | $\overline{\text{SWI}}$ | Non Maskable | 3 |
| | $\overline{\text{IRQA}}$ (External Interrupt) | IRQA Mode Bits | 0, 1, 2 |
| | $\overline{\text{IRQB}}$ (External Interrupt) | IRQB Mode Bits | 0, 1, 2 |
| | Host Command Interrupt | HCIE | 0, 1, 2 |
| | Host Receive Data Interrupt | HRIE | |
| | Host Transmit Data Interrupt | HTIE | |
| | SSI Rx Data with Exception Interrupt | RIE | 0, 1, 2 |
| | SSI Rx Data Interrupt | RIE | |
| | SSI Tx Data with Exception Interrupt | TIE | |
| | SSI Tx Data Interrupt | TIE | |
| | SCI Rx Data with Exception Interrupt | RIE | 0, 1, 2 |
| | SCI Rx Data Interrupt | RIE | |
| | SCI Tx Data Interrupt | TIE | |
| | SCI Idle Line Interrupt | ILIE | |
| lowest | SCI Timer Interrupt | TMIE | |

MTA5-6410

# EXERCISE 9

## EXCEPTIONS

### For this exercise the following information is given;

The Status Register and Interrupt Priority Registers are set as shown:

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LF | 0 | T | 0 | S1 | S0 | 1 | 0 | 0 | L | E | U | N | Z | V | C |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X:$FFFF | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

There are Interrupts pending from all On Chip Peripherals. Pins IRQA and IRQB are Grounded.

### Answer the following questions:

1. The Interrupt Priority is at level _____ .

2. List the Priority Level for each Interrupt.

| SCI | SSI | HOST | IRQB | IRQA |
|---|---|---|---|---|
| ____ | ____ | ____ | ____ | ____ |

3. Will the Interrupt Exception be taken?
   Answer YES or NO for each Pending Interrupt.

| SCI | SSI | HOST | IRQB | IRQA |
|---|---|---|---|---|
| ____ | ____ | ____ | ____ | ____ |

4. The DSP56000 has five processing states. They are; _____ , _____ _____ , _____ , and _____.

5. There are _____ individual Interrupt Exception Sources.

6. The four Unmaskable Level Three Exceptions are; _____ , _____ _____ , and _____ _____ .

7. Exit from the STOP State is by the assertion of either _____ , or _____

8. While Tracing the only allowable Exceptions are _____ , _____ , or _____ .

MTA5-6500

# DSP INSTRUCTIONS AND ADDRESSING MODES

## OBJECTIVES

After studying this section, the student should be able to:

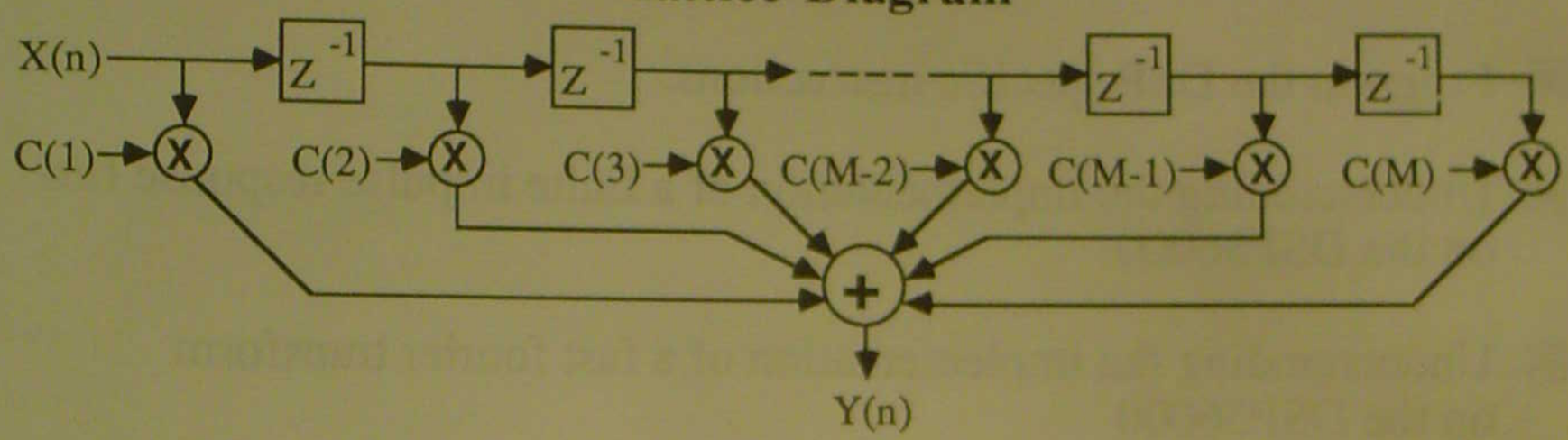
- Program the DSP specific instructions.
- Understanding the implementation of a finite impulse response filter on the DSP56000
- Understanding the implementation of a fast fourier transform on the DSP56000

# MULTIPLY ACCUMULATE INSTRUCTION

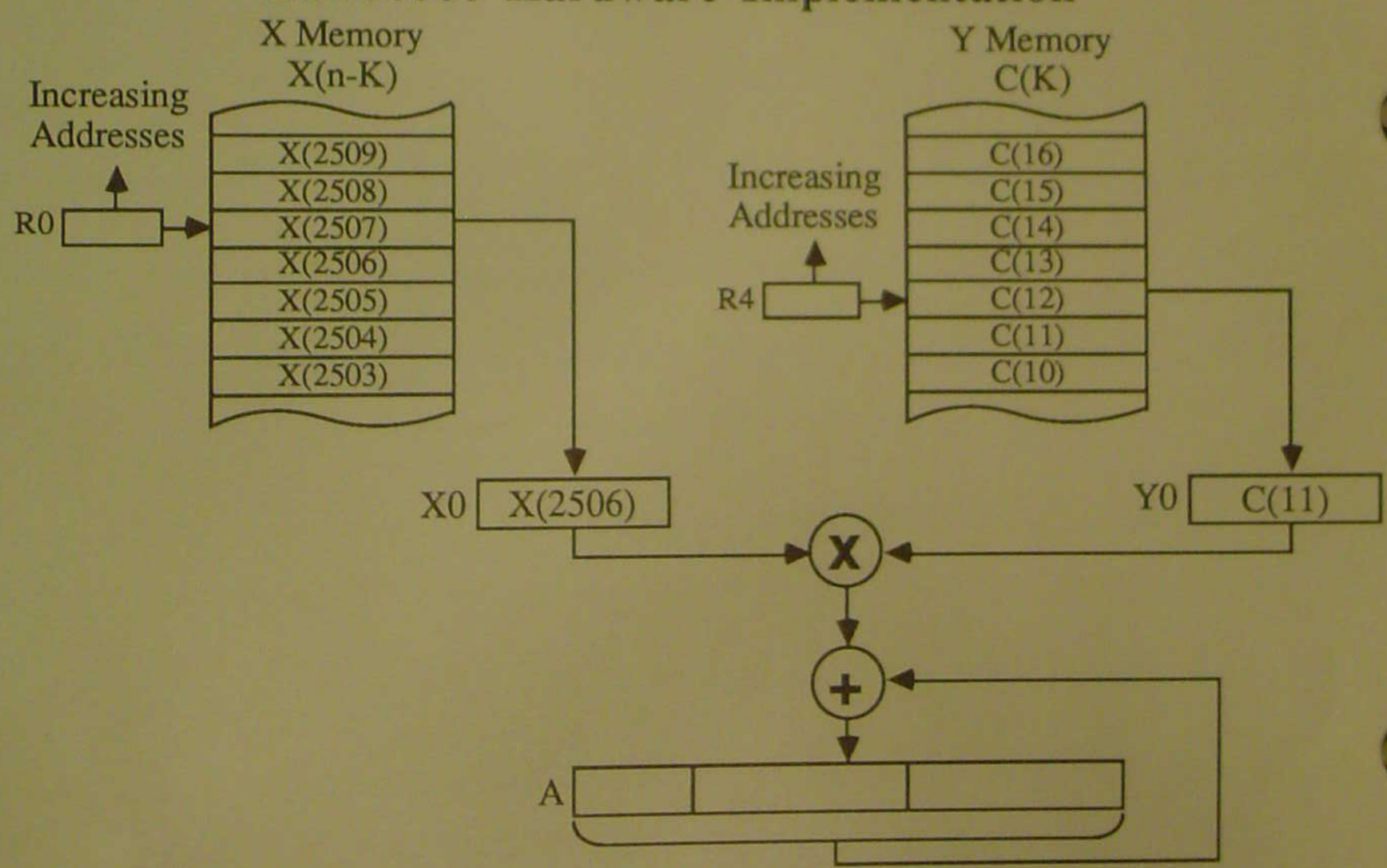
## Direct Form Finite Impulse Response Filter

$$Y(n) = \sum_{K=0}^{M} C(K)\, X(n-K)$$

## Lattice Diagram




## DSP56000 Hardware Implementation

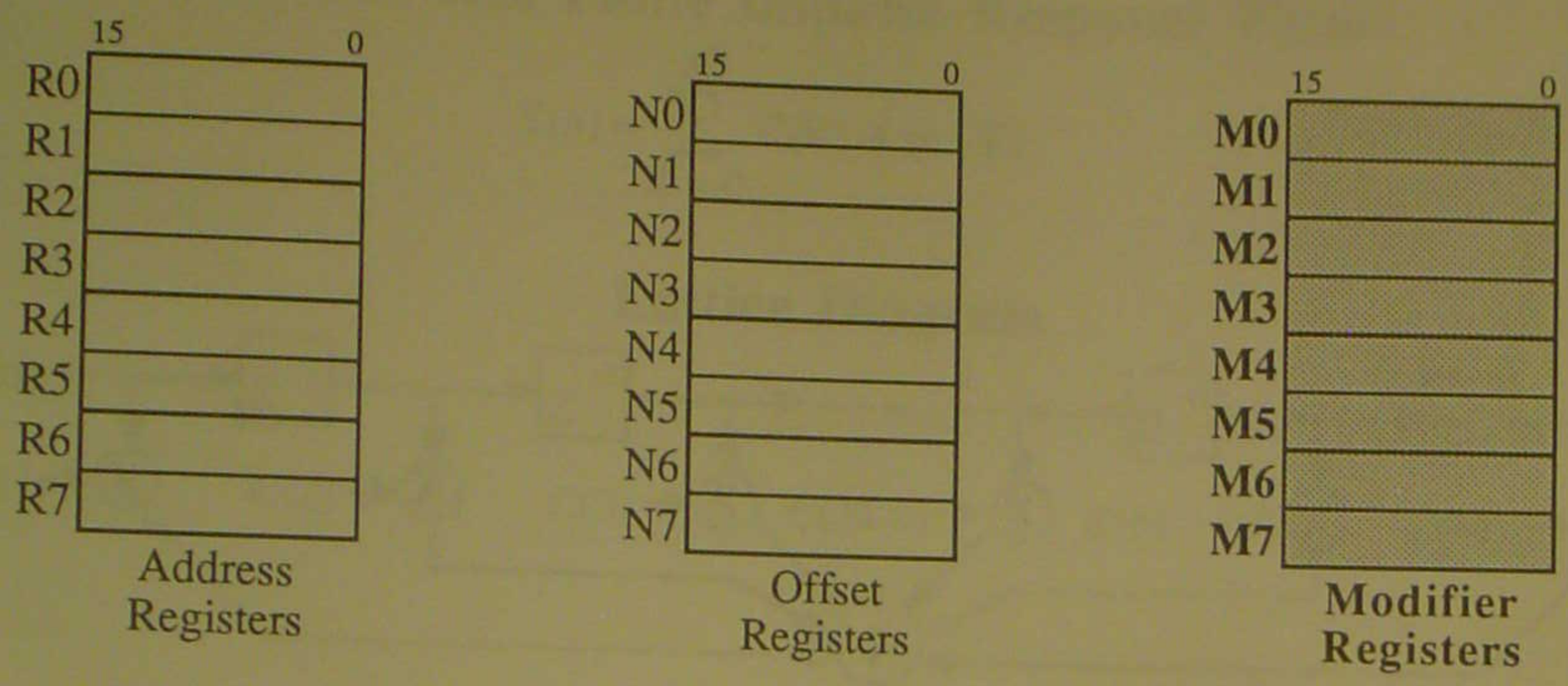



DSP56000 Software Implementation

| | | | |
|---|---|---|---|
| **MAC** | **X0,Y0,A** | **X:(R0)+,X0** | **Y:(R4)+,Y0** |

MTA5-8110

# MODIFIER REGISTERS




| MODIFIER Mn VALUE | ADDRESSING MODE ARITHMETIC |
|---|---|
| 0 | Reverse Carry (Bit Reversed) |
| 1<br>2<br>⋮<br>⋮<br>⋮<br>32766<br>32767 | Modulo 2<br>Modulo 3<br>⋮<br>Modulo (Mn + 1)<br>⋮<br>Modulo 32767<br>Modulo 32768 |
| ⋮<br>65535 | Reserved<br>Linear (Modulo 65536) |

**USED BY ALL FORMS OF ADDRESS REGISTER INDIRECT**

| | |
|---|---|
| No Update | Rn |
| Postincrement by 1 | (Rn)+ |
| Postdecrement by 1 | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

MTA5-8120

# MODULO M ADDRESSING

1. Modifier Register Mn = M-1.

2. Lower Bound = $\overset{15}{XX}\ldots XX00\ldots\overset{0}{00}$ where $2^J \geq M$

   $\leftarrow J \rightarrow$

3. Upper Bound = $\overset{15}{XX}\ldots XX00\ldots\overset{0}{00} + M - 1$

   $\leftarrow J \rightarrow$

4. Lower Bound ≤ Address Register Rn ≤ Upper Bound

5. Offset Register Nn = Increment ≤ M

**Example:** MOVE X0,X:(R0)+N0

Starting Address

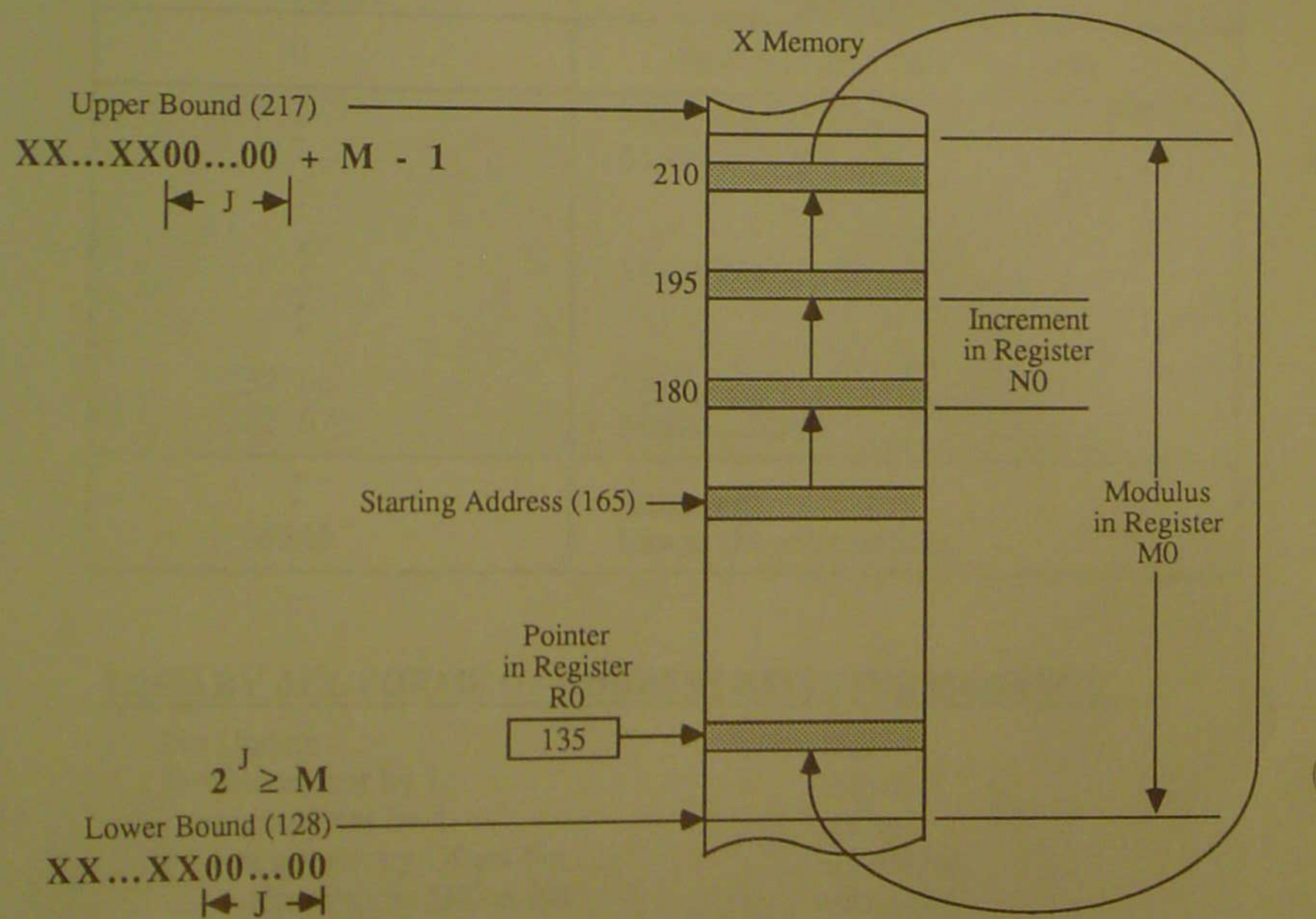
R0 [ 165 ]

Increment

N0 [ 15 ]

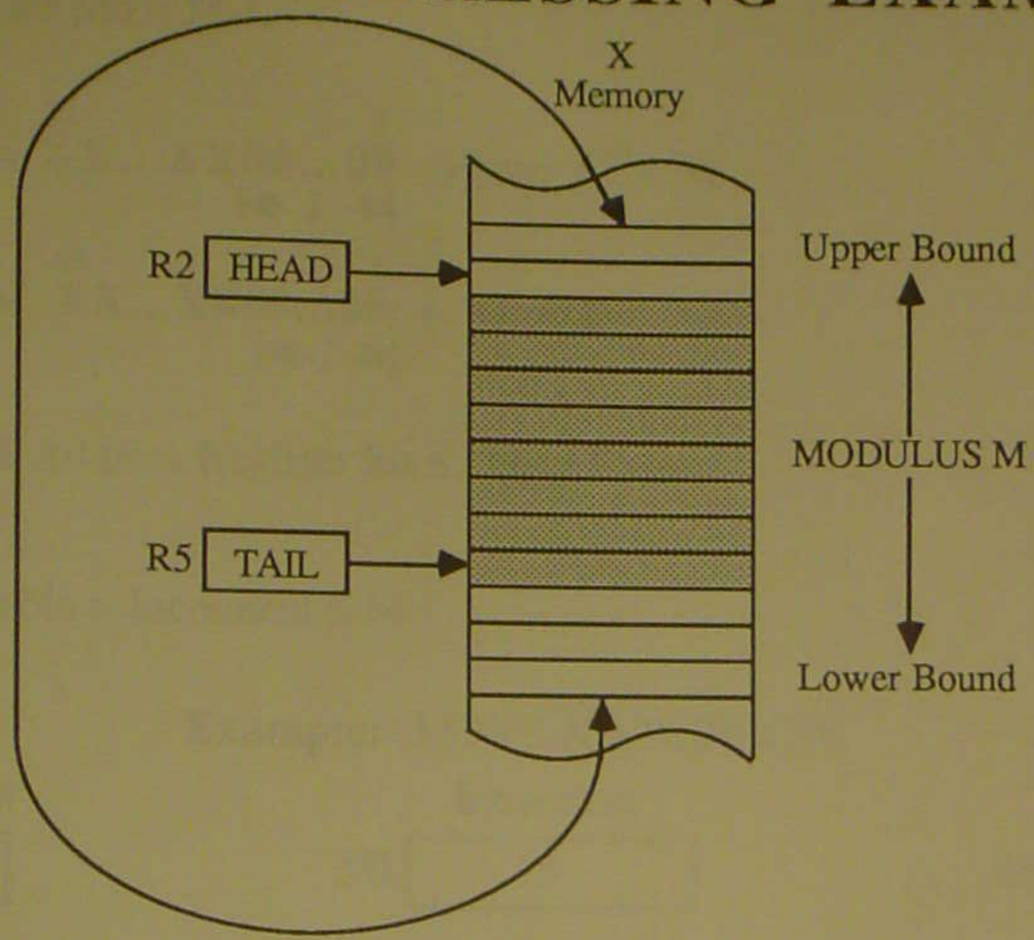
Modulo M = 90

M0 [ 89 ]




MTA5-8130-1

# MODULO M ADDRESSING EXAMPLE




R2 | HEAD | N2 | XXXX | M2 | M-1

**Add To Head:** **MOVE X0,X:(R2)+**
**Remove From Head:** **MOVE X:-(R2),X0**

R5 | TAIL | N5 | XXXX | M5 | M-1

**Add To Tail:** **MOVE X1,X:-(R5)**
**Remove From Tail:** **MOVE X:(R5)+,X1**

## Applications

F.I.F.O.'s
Circular Queues
Delay Lines
Shift Registers
Sample Buffers
Finite Sum of Products

MTA5-8140

# HARDWARE DO LOOPS

| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| DO | **Start of Loop**<br>SP+1→SP; LA→SSH; LC→SSL; #xxx→LC<br>SP+1→SP; PC→SSH; SR→SSL; EXPR-1→LA<br>1→LF<br>**End of Loop**<br>If LC≠1 then LC-1→LC; SSH→PC<br>Else SSL(LF)→SR; SP-1→SP<br>SSH→LA; SSL→LC; SP-1→SP | **Static**<br>DO #xxx, Expr<br>**Dynamic**<br>DO X: <ea>,Expr<br>DO Y: <ea>, Expr<br>DO S, Expr |

**#xxx Static Loop Count**

Immediate Short Data #xxx

**X or Y Dynamic Loop Count Effective Address MODES:**

| | |
|---|---|
| Absolute Short Address | aa |
| Address Register Indirect with; | |
| No Update | (Rn) |
| Postincrement by 1 | (Rn)+ |
| Postdecrement by 1 | (Rn)- |
| Postincrement by Offset Nn | (Rn)+Nn |
| Postdecrement by Offset Nn | (Rn)-Nn |
| Indexed by Offset Nn | (Rn+Nn) |
| Predecrement by 1 | -(Rn) |

**S Dynamic Loop Count Operands:**

X1, X0
Y1, Y0
A,A2,A1,A0
B,B2,B1,B0
SR
OMR
SP,SSL
LA,LC
R0-R7
N0-N7
M0-M7

**DO LOOP EXAMPLE**

```
        DO      #35,END1                ;BEGIN LOOP 1
                  :
        DO      X:(R1)+, END2           ;BEGIN LOOP 2
                  :
        DO      Y:$1EA,END3             ;BEGIN LOOP 3
                  :
        DO      B1,END4                 ;BEGIN LOOP 4
                  :
        MAC     X0,Y0,A                 ;LAST INSTRUCTION IN LOOP 4
END4              :                     ;FIRST INSTRUCTION AFTER LOOP 4
        ADD     B,A   X:(R1)+,X0        ;LAST INSTRUCTION IN LOOP 3
END3              :                     ;FIRST INSTRUCTION AFTER LOOP 3
        MOVE    X:(R7)+N7,A             ;LAST INSTRUCTION IN LOOP 2
END2              :                     ;FIRST INSTRUCTION AFTER LOOP 2
        MOVE    B,Y:-(R0)               ;LAST INSTRUCTION IN LOOP 1
END1                                    ;FIRST INSTRUCTION AFTER LOOP1
```

MTA5-8150

# END CURRENT DO LOOP

| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| ENDDO | SSL(LF) → SR; SP-1 → SP<br>SSH → LA; SSL → LC, SP-1 → SP | ENDDO |

**END DO EXAMPLE**

```
        :
DO              Y0,NEXT         ;exec. loop ending at NEXT (Y0) times
        :
        MOVEC   LC,A            ;get current value of loop counter (LC)
        CMP     Y1,A            ;compare loop counter with value in Y1
        JNE     ONWARD          ;go to ONWARD if LC not equal to Y1
        ENDDO                   ;LC equal to Y1, restore all DO registers
        JMP     NEXT            ;go to NEXT
ONWARD      :                   ;LC not equal to Y1, continue DO loop
            :                   ;(last instruction in DO loop)
NEXT    MOVE    #$123456,X1     ;(first instruction AFTER DO loop)
```

MTA5-8160

# REPEAT NEXT INSTRUCTION

| Mnemonic | Operation | Assembler Syntax |
|---|---|---|
| REP | LC → TEMP; #xxx → LC<br>Repeat Next Instruction Until LC = 1<br>TEMP → LC | REP #xxx<br>REP X:<ea><br>REP Y:<ea><br>REP S |

| #xxx Static Loop Count | X or Y Dynamic Loop Count Effective Address MODES: | | S Dynamic Loop Count Operands: |
|---|---|---|---|
| Immediate Short Data #xxx | Absolute Short Address | aa | X1,X0 |
| | Address Register Indirect with; | | Y1,Y0 |
| | No Update | (Rn) | A,A2,A1,A0 |
| | Postincrement by 1: | (Rn)+ | B,B2,B1,B0 |
| | Postdecrement by 1: | (Rn)- | SR |
| | Postincrement by Offset Nn | (Rn)+Nn | OMR |
| | Postdecrement by Offset Nn | (Rn)-Nn | SP,SSH,SSL |
| | Indexed by Offset Nn | (Rn+Nn) | LA,LC |
| | Predecrement by 1 | -(Rn) | R0-R7 |
| | | | N0-N7 |
| | | | M0-M7 |

## REP EXAMPLE

```
      :
REP   X0                                  ;repeat (X0) times
MAC   X1,Y1,A   X:(R1)+,X1    Y:(R4)+,Y1  ;X1 * Y1 + A ---> A, update X1,Y1
      :
```

## Characteristics of REP Instruction

- Only single word instructions are repeatable
- REP is not interruptable

MTA5-8170

# EXERCISE 10

## DSP INSTRUCTIONS AND ADDRESSING MODES

1. The REPEAT instruction can be used only on _____ word instructions and is not _____ .

2. The ENDDO instruction is used to _____ a DO Loop.

3. Hardware DO Loops can be nested to a maximum of _____ levels without storing the stack

4. The Maximum number of DO Loop iterations is _____ .

5. Modulo N addressing is useful for creating _____ , _____ , _____ _____ and _____ _____ .

6. Given the initial conditions of the Address Arithmetic Unit Registers:

| | Address Registers R | Offset Registers N | Modifier Registers M |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 3 | 2 | 5 |
| 2 | 5 | 5 | 7 |
| 3 | 4 | 8 | 10 |
| 4 | 7 | 7 | 12 |
| 5 | 8 | 11 | 15 |
| 6 | 8 | 11 | 18 |
| 7 | 12 | 12 | 21 |

Fill in the new values of the Address registers after the execution of this code sequence. (NOTE: All numbers are in Base Ten and the problem should be worked in Base Ten.)

```
MOVE X:(R0)+N0,X0   Y:(R4)+N4,Y0
MOVE X:(R1)+N1,X0   Y:(R5)+N5,Y0
MOVE X:(R2)+N2,X0   Y:(R6)+N6,Y0
MOVE X:(R3)+N3,X0   Y:(R7)+N7,Y0
```

| | Address Registers R |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |

MTA5-8180

# FOUR TAP FIR FILTER EXAMPLE

$$y(n) = \sum_{k=0}^{3} c(k)\ x(n-k)$$

$$y(n) = c(0)x(n) + c(1)x(n-1) + c(2)x(n-1) + c(3)x(n-3)$$

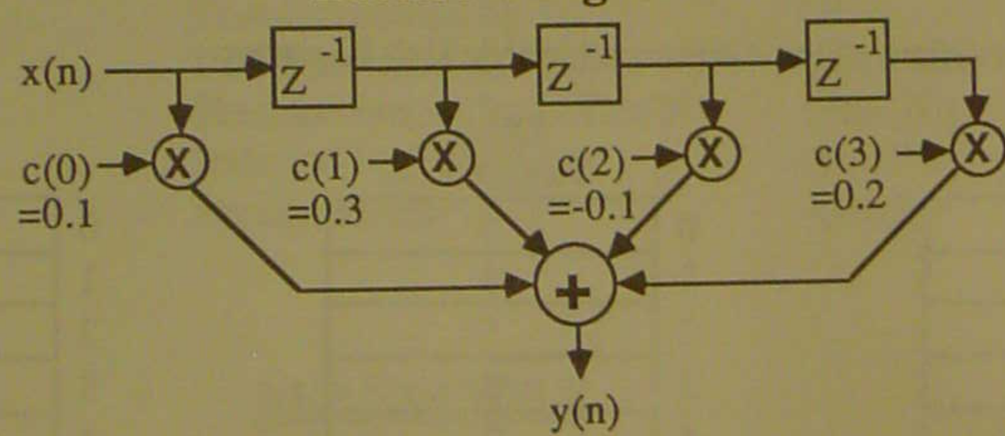
with z transform:

$$\frac{Y(z)}{X(z)} = c(0) + c(1)z^{-1} + c(2)z^{-2} + c(3)z^{-3}$$

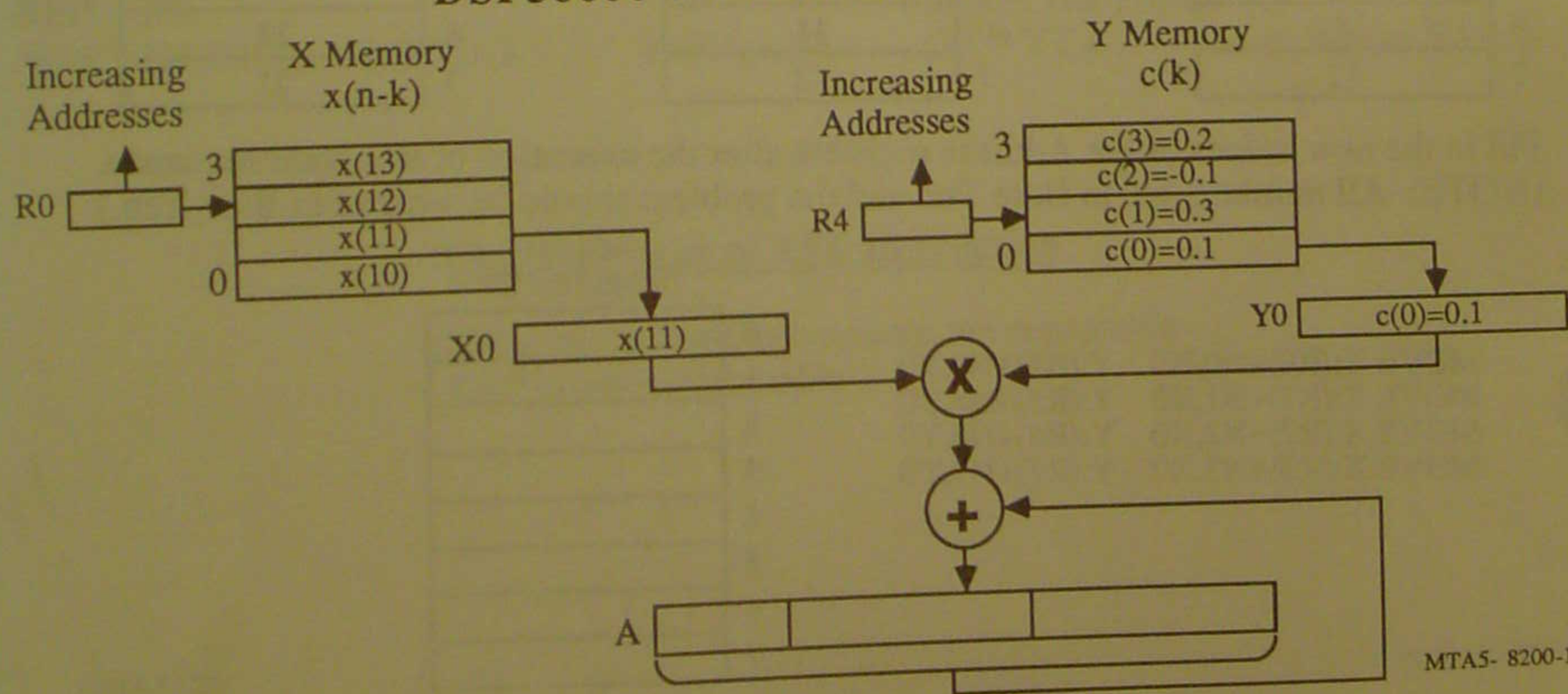
where:

| | | |
|---|---|---|
| x(n) | = | input sample at time nT |
| y(n) | = | output sample at time nT |
| T | = | sampling period |
| c(n) | = | filter impulse response at time nT |
| c(0) | = | 0.1 |
| c(1) | = | 0.3 |
| c(2) | = | –0.1 |
| c(3) | = | 0.2 |

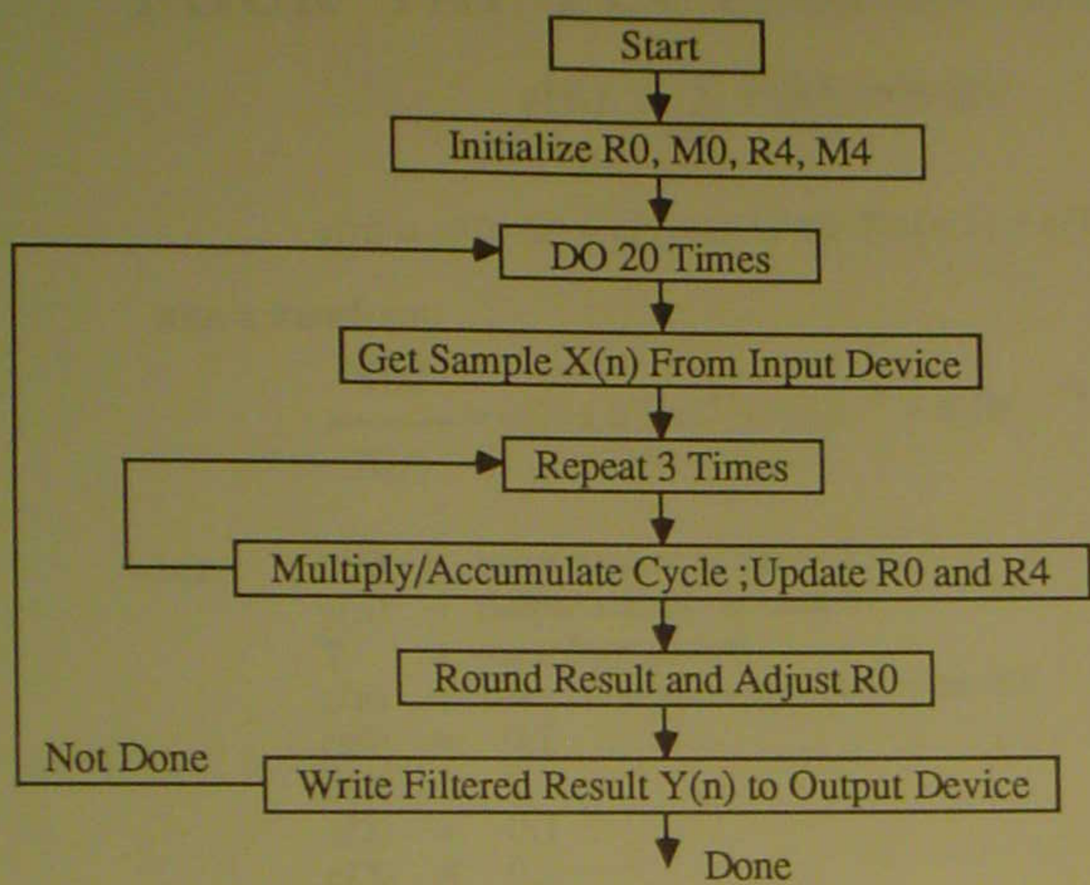
## Lattice Diagram




## DSP56000 IMPLEMENTATION



# FOUR TAP FIR FILTER PROGRAM

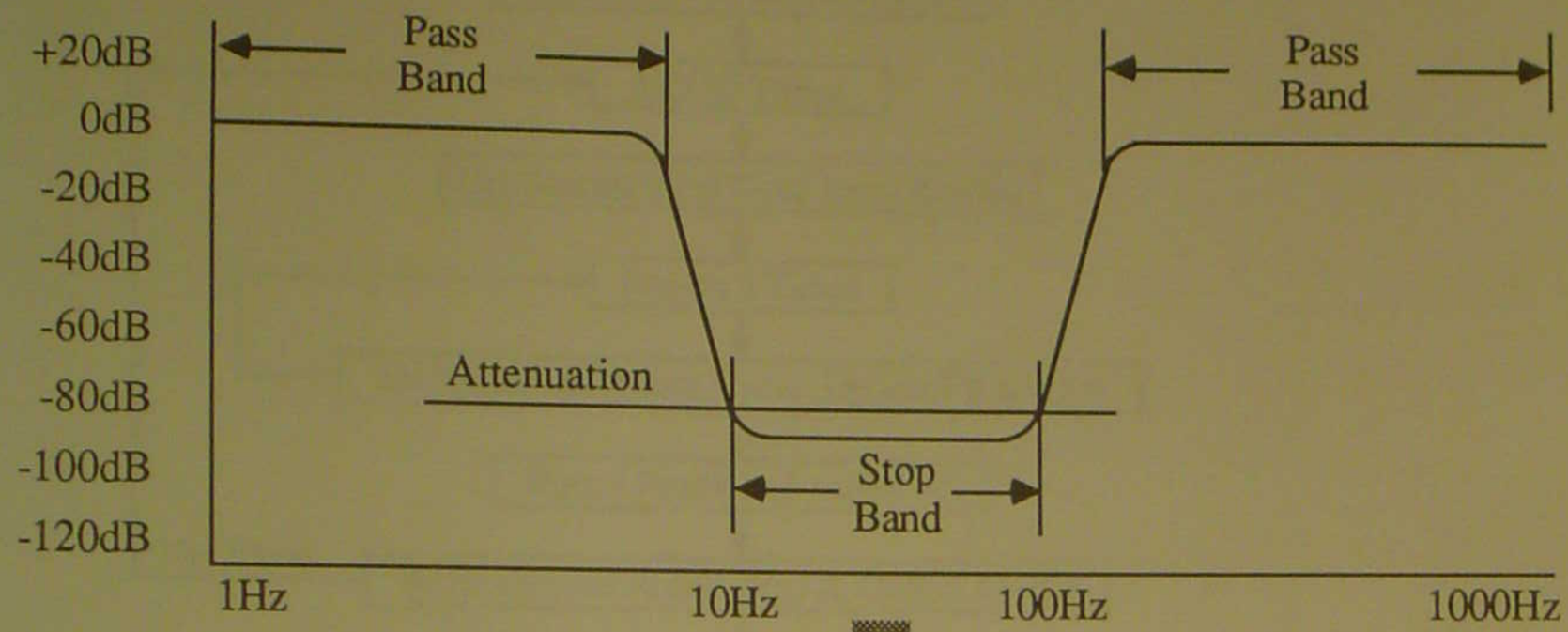



```
;           Four Tap FIR program

datin       equ     $ffff                   ;location in Y memory of input device
datout      equ     $fffe                   ;location in Y memory of output device
npts        equ     20                      ;number of points to process
ntaps       equ     4                       ;number of taps in filter

            org     x:0                     ;starting address in X memory
states      dsm     ntaps                   ;define storage for filter states

            org     y:0                     ;starting address in Y memory
coef        dc      .1, .3, -.1, .2         ;filter coefficients

            org     p:$100                  ;starting address in program memory
start                                       ;initialize parameters
            move    #states,r0              ;point to filter states with R0
            move    #ntaps-1,m0             ;setup modulo addressing (ntaps)
            move    #coef,r4                ;point to filter coefficients
            move    #ntaps-1,m4             ;setup modulo addressing (ntaps)

            do      #npts,_endp             ;do one pass for each sample point

            movep   y:datin,x0              ;get sample from input device

;           Beginning of four tap FIR filter loop

            clr     a x0,x:(r0)+    y:(r4)+,y0          ;save first state
            rep     #ntaps-1                ;repeat once for each coefficient
            mac     x0,y0,a     x:(r0)+,x0      y:(r4)+,y0      ;multiply/accumulate
            macr    x0,y0,a  (r0)-                              ;round result

            movep   a,y:datout              ;output sample

_endp
            end
```

MTA5-8210

# SELECTION OF FILTER COEFFICIENTS

## Determine Filter Characteristics




**Key Parameters Into Filter Design Package**

| | | |
|---|---|---|
| Atlanta Signal Processors<br>Program: D.F.P.F.<br>770 Spring St.<br>Atlanta, GA 30308<br>(404) 892-7265 | I.E.E.E. Inc.<br>345 E. 47th St.<br>New York City, N.Y. 10017<br>(212) 705-7900 | Signal Technology<br>Program: I.L.S.<br>5951 Encina Rd.<br>Goleta, CA 93117<br>(805) 683-3771 |
| Signix Corp.<br>Program: Dispro<br>19 Pelham Island Rd.<br>Wayland, MA 01778<br>(617) 358-5955 | Whitman Engineering Inc.<br>Program: Madcap<br>520 S. Maitland Ave.<br>Maitland, FL 32751<br>(305) 628-4516 | Momentum Data Systems<br>1666 Newport Blvd. #115<br>Costa Mesa, Ca 92627<br>(714) 548-3257 |

**Filter Coefficients Are Generated**

```
C( 1) = 0.3271374 = C(49)
C( 2) = 0.2731500 = C(48)
C( 3) = 0.2008393 = C(47)
C( 4) = 0.1154480 = C(46)
C( 5) = -0.2329397 = C(45)
C( 6) = -0.6874674 = C(44)
C( 7) = -0.1538262 = C(43)
C( 8) = -0.2257197 = C(42)
        "
        "
        "
```

MTA5-8220-1

# EXERCISE 11

## DSP INSTRUCTIONS AND ADDRESSING MODES

1. Given the four tap finite impulse response filter and its four coefficient c(0) through c(3);

$$y(n) = \sum_{K=0}^{3} c(k)\ x(n-k)$$

$$\begin{aligned} c(0) &= 0.5 \\ c(1) &= -0.3 \\ c(2) &= -0.6 \\ c(3) &= 0.4 \end{aligned}$$

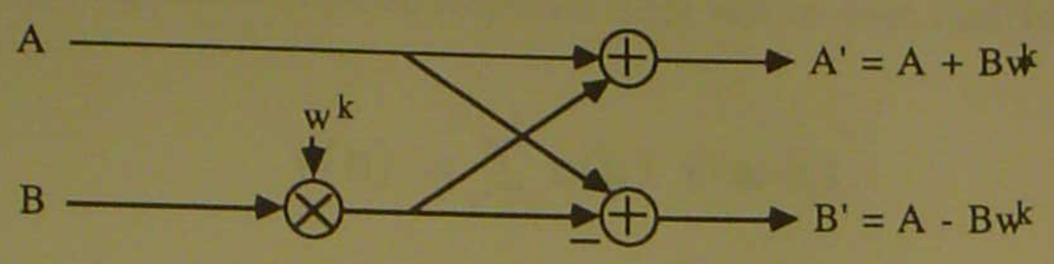
For the input sequence x(n) on the left, find the output sequence y(n) for the requested values.

| | |
|---|---|
| x(0) = 0.1 | |
| x(1) = 0.2 | |
| x(2) = 0.3 | |
| x(3) = 0.4 | y(3) = _____ |
| x(4) = 0.5 | y(4) = _____ |
| x(5) = 0.4 | y(5) = _____ |
| x(6) = 0.3 | y(6) = _____ |
| x(7) = 0.2 | y(7) = _____ |

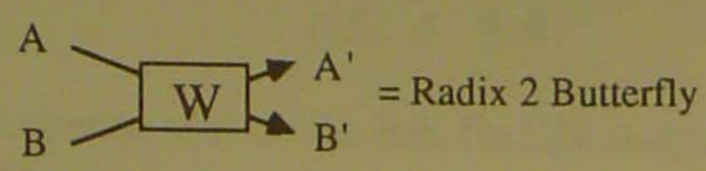
MTA5-8230-1

# FAST FOURIER TRANSFORM BUTTERFLY

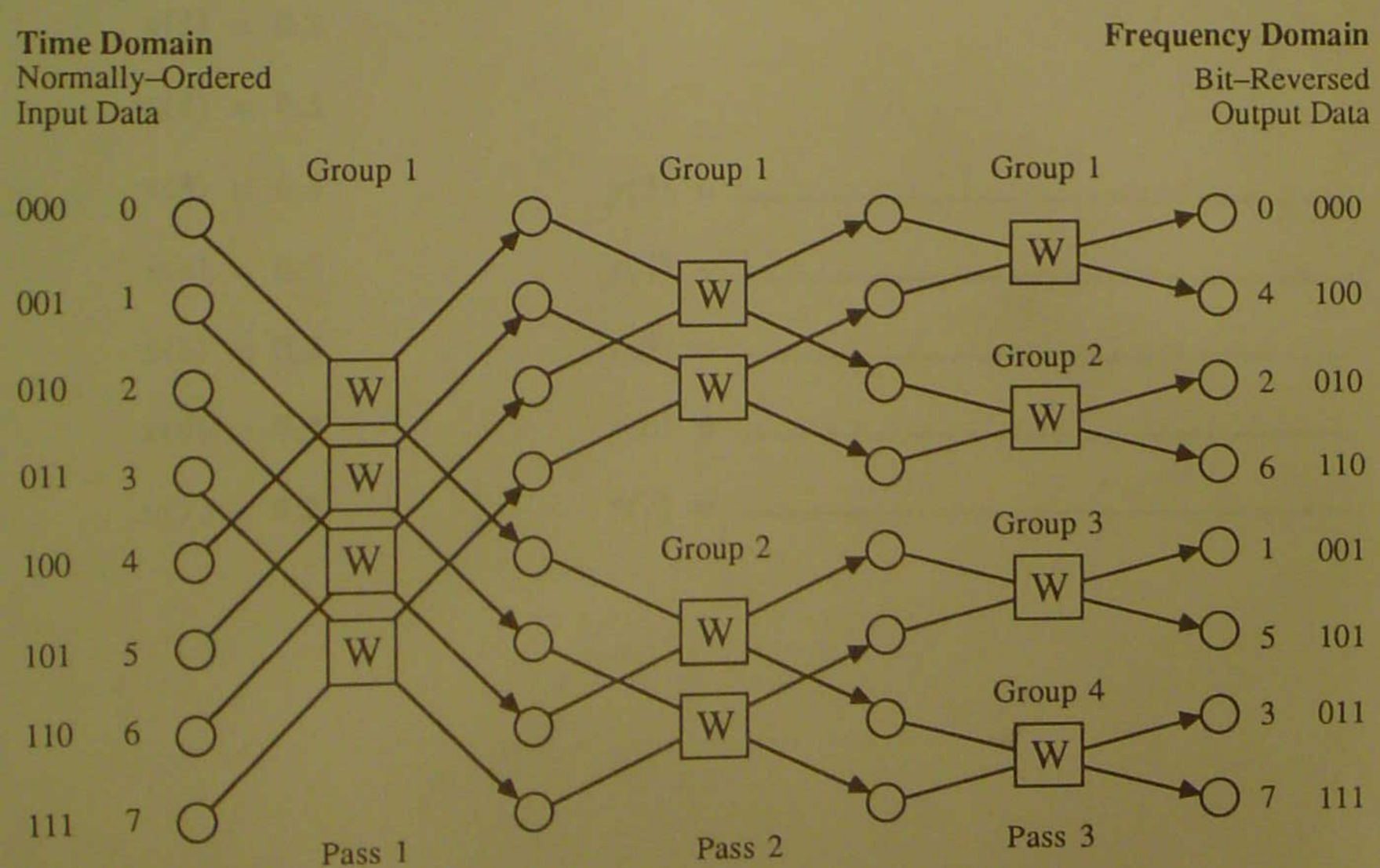
## Radix 2, decimation in time, complex FFT butterfly



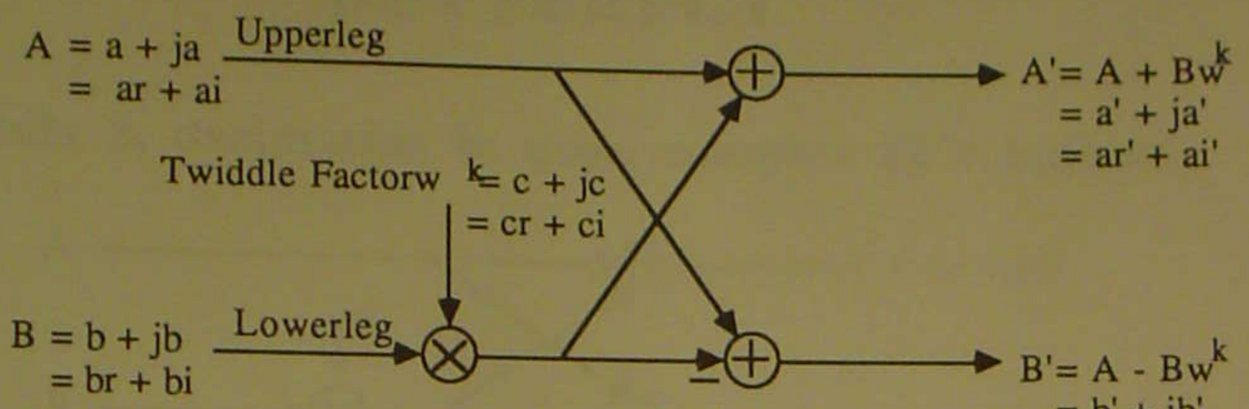

## Schematic Symbol




## 8 Point FFT Transform

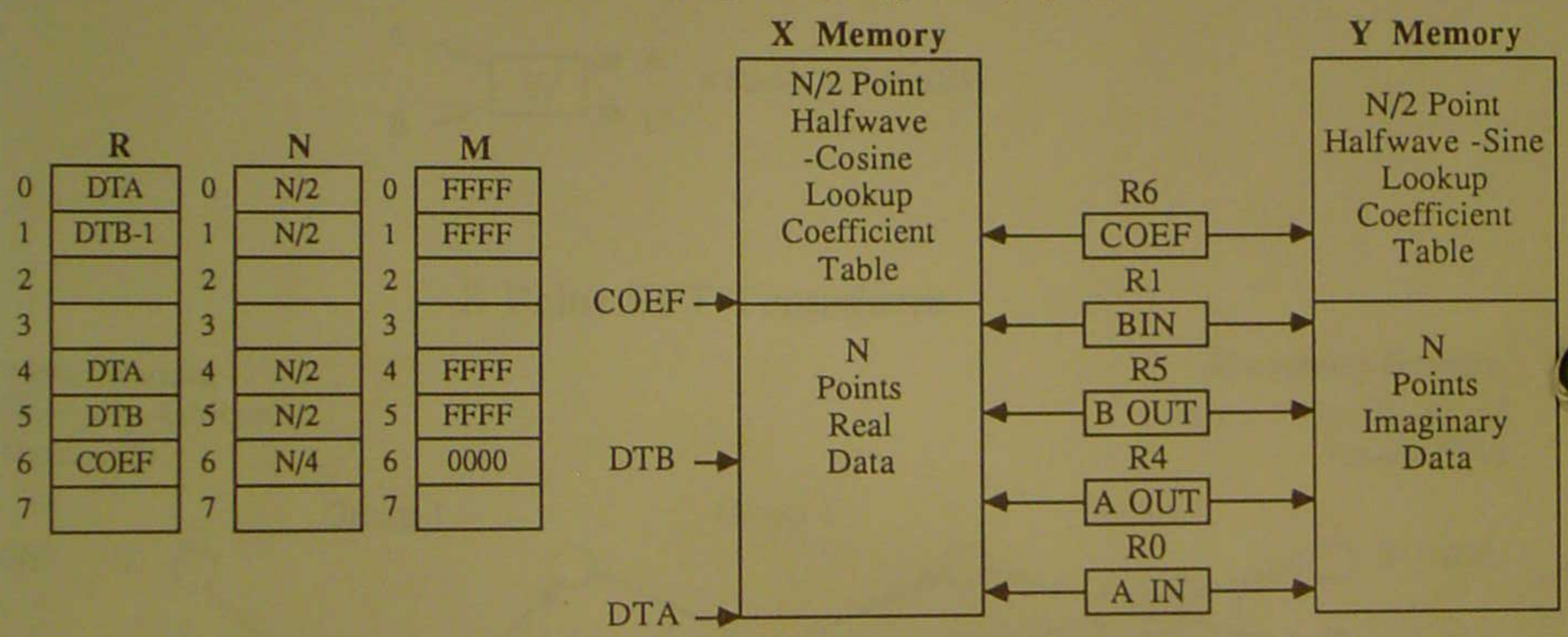


# BUTTERFLY ALGORITHM




**FFT Butterfly Equation**

$$
\begin{aligned}
ar' &= ar + cr * br - ci * bi \\
ai' &= ai + ci * br + cr * bi \\
br' &= ar - cr * br + ci * bi = 2*ar - ar' \\
bi' &= ai - ci * br - cr * bi = 2*ai - ai'
\end{aligned}
$$




**Butterfly Program for NPoint F.F.T.**

```
        do      n2,_end_grp
        move    x:(r1), x1      y:(r6),y0       ;Lookup -Sine & -Cosine
        move    x:(r5), a       y:(r0),b
        move    x:(r6)+n6,x0                    ;Update C pointer
        do      n0,_end_bfy             ;Radix 2 DIT Butterfly Kernel
        mac     x1,y0,b         y:(r1)+,y1
        macr    -x0,y1,b        a,x:(r5)+               y:(r0),a
        subl    b,a             x:(r0),b                b,y:(r4)
        mac     x1,x0,b         x:(r0)+,a       a,y:(r5)
        macr    -y1,y0,b        x:(r1),x1
        subl    b,a             b,x:(r4)+       y:(r0),b
_end_bfy
        move    a,x:(r5)+n5             y:(r1)+n1,y1
        move    x:(r0)+n0,x1            y:(r4)+n4,y1
_end_grp
```

MTA5- 8310-1

# BIT REVERSED ADDRESSING FOR N POINT FFT

1. Modifier Register Mn = $0000
2. Lower Bound = $\overset{15}{XX} \ldots XX00 \ldots \overset{0}{00}$ where $2^K \geq N$ (Minimize K)
   $\leftarrow K \rightarrow$
3. Upper Bound = $\overset{15}{XX} \ldots XX00 \ldots \overset{0}{00} + 2^K - 1$
   $\leftarrow K \rightarrow$
4. Lower Bound ≤ Address Register Rn ≤ Upper Bound
5. Offset Register Nn = $2^{K-1}$

**1024 Point FFT Bit Reversed Example**

$N = 1024 = 2^K$

$K = 10$

Offset = $2^{K-1}$ = 512

Lower Bound = 3 X 1024
= 3072

Upper Bound = 3072 + 1024 -1
= 4095

| | | K |
|---|---|---|
| 3088 = | 0 0 0 0 1 1 | 0 0 0 0 0 1 0 0 0 0 |
| | Reverse K LSB's | |
| 3104 = | 0 0 0 0 1 1 | 0 0 0 0 1 0 0 0 0 0 |
| +1 | No Carry Beyond K LSB'S | Increment K LSB's |
| 3105 = | 0 0 0 0 1 1 | 0 0 0 0 1 0 0 0 0 1 |
| | Reverse K LSB's | |
| 3601 = | 0 0 0 0 1 1 | 1 0 0 0 0 1 0 0 0 0 |

**Starting Address**

R0 [ 3072 ] ( 0)

MOVE X:(R0)+N0,X0

R0 [ 3584 ] (512)

MOVE X:(R0)+N0,X0

R0 [ 3328 ] (256)

MOVE X:(R0)+N0,X0

R0 [ 3840 ] (768)

**Increment**

N0 [ 512 ]

**Bit Reverse**

M0 [ 0000 ]

X Memory

Upper Boundary (4095)
XX . . . XX00 . . . 00
$\leftarrow K \rightarrow$
$2^K \geq N$

N=1024
Word
FFT
Buffer

Lower Boundary (3072)
XX . . . XX00 . . . 00
$\leftarrow K \rightarrow$

MTA5-8320-1

# EXERCISE 12

## DSP INSTRUCTIONS AND ADDRESSING MODES

1. Given the initial conditions of Address Arithmetic Unit Registers R0, N0, and M0;

R0 [ 0000 ]  N0 [ $80 ]  M0 [ 0000 ]

Show the new values of R0 for the code sequence.:

```
REP #5
MOVE X:(R0)+N0,X0
```

R0 [ ]

R0 [ ]

R0 [ ]

R0 [ ]

R0 [ ]

MTA5-8330

# ON CHIP PERIPHERALS

## OBJECTIVES

After studying this section, the student should be able to:

- Describe the Input/Output section of the DSP56000.
- Program a Parallel Transfer.
- Describe a Host Transfer.
- Program a Serial Transfer Using the Serial Communications Interface.
- Describe the Synchronous Serial Interface.

# ON CHIP PERIPHERAL FEATURES

- 24 Separate Input/Output Pins

- Software Programmable as General Purpose Parallel I/O or Alternate Peripheral Function.
  - HOST Interface
  - Serial Communications Interface (SCI)
  - Synchronous Serial Interface (SSI)

- All Peripherals are Memory Mapped so Standard MOVE Instructions can be used.

- Special MOVEP Instruction for full I/O service capability.

- I/O Short Addressing provides shorter, faster instructions.

- All Control Ports and Data Registers are Read/Write so Standard Bit Manipulation Instructions can be used.

- Bit Addressing Instruction provide useful I/O Service Routines: BCHG, BCLR, BSET, BTST, JCLR, JSCLR, JSET, JSSET

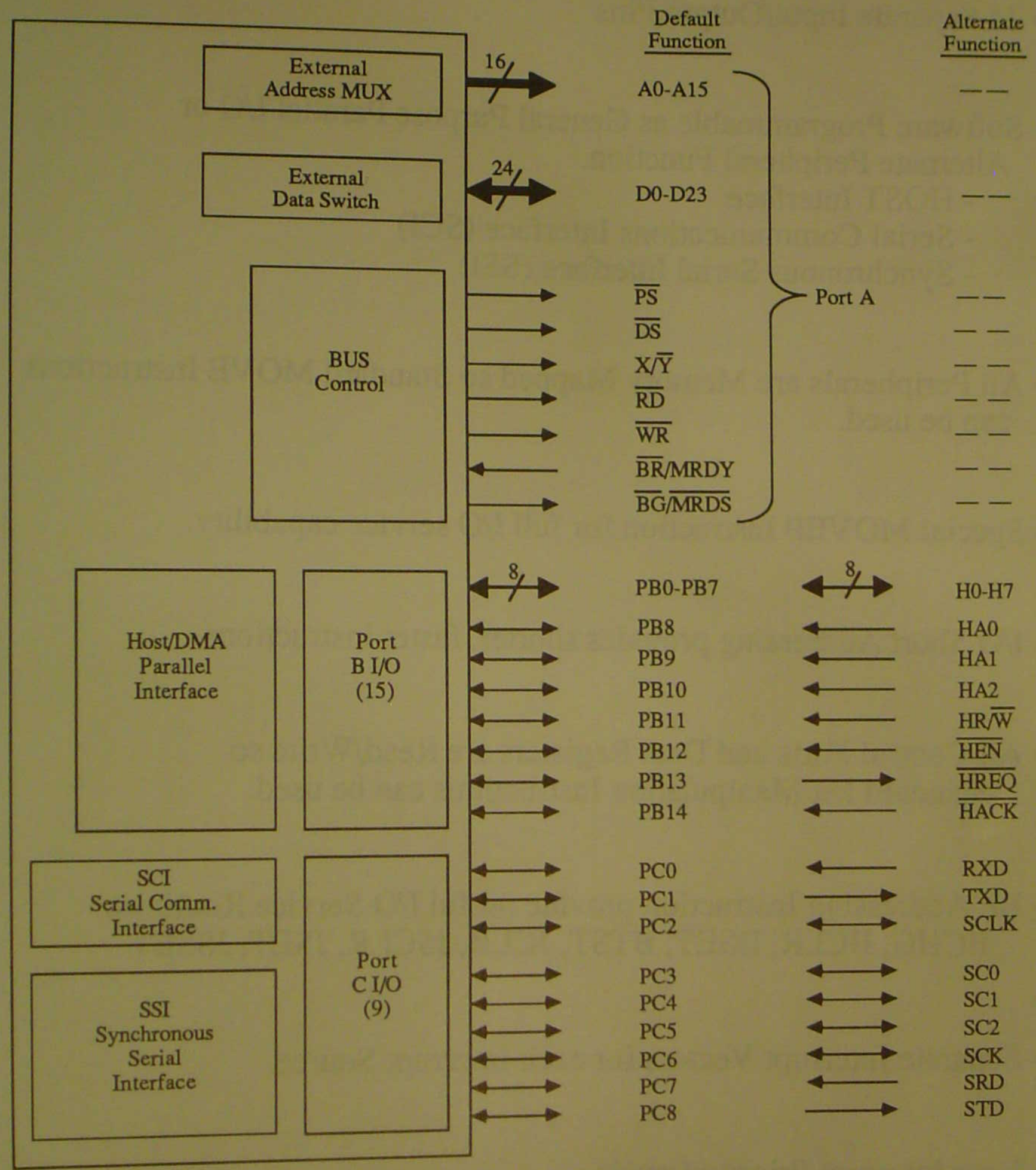
- Separate Interrupt Vectors for each Interrupt Source.

- Four Interrupt Priority Levels.

- Either Fast or Long Interrupts may be used.

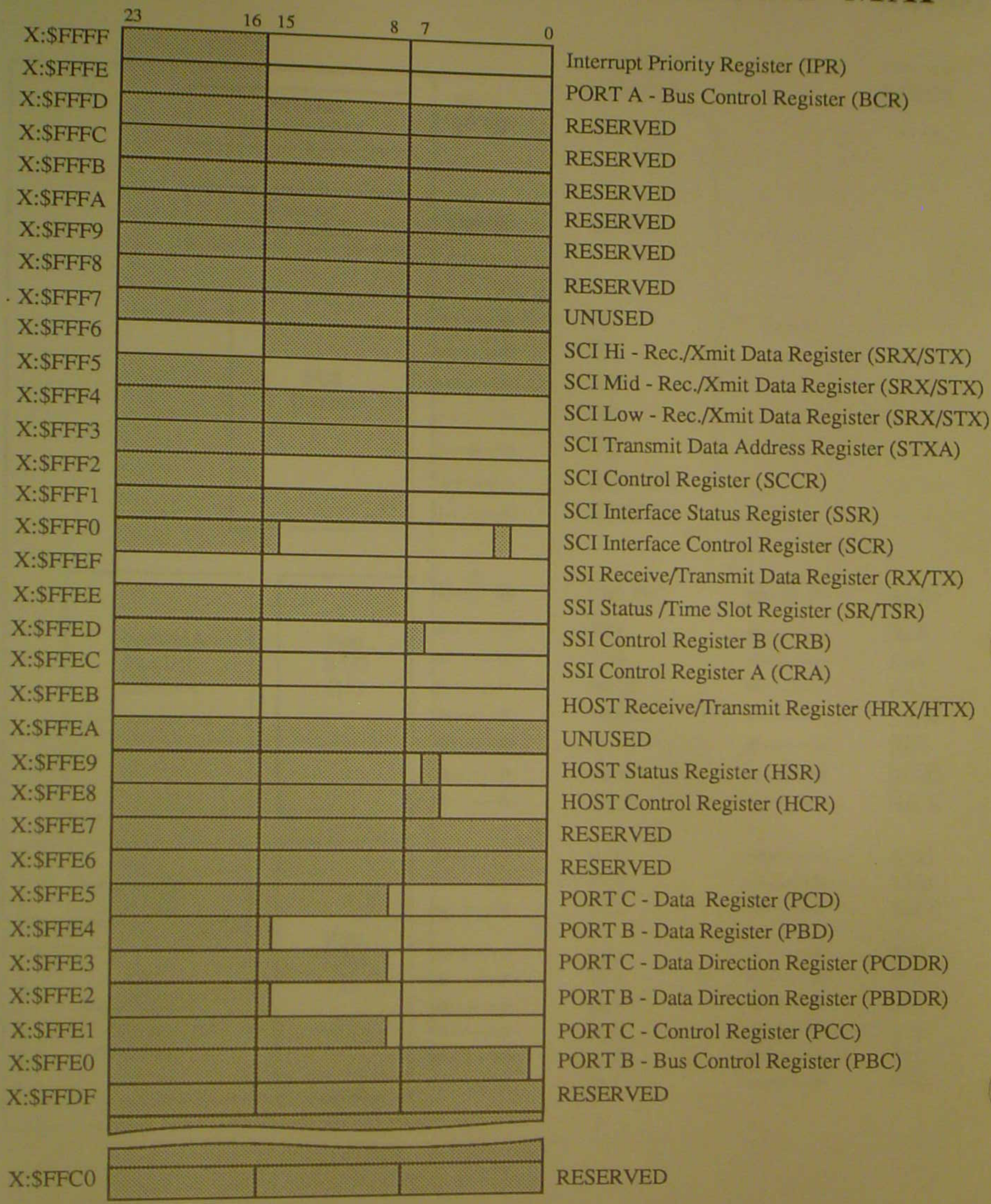
MTA5-7005

# DSP56000 INPUT/OUTPUT BLOCK DIAGRAM




MTA5-7008

# ON CHIP PERIPHERAL MEMORY MAP

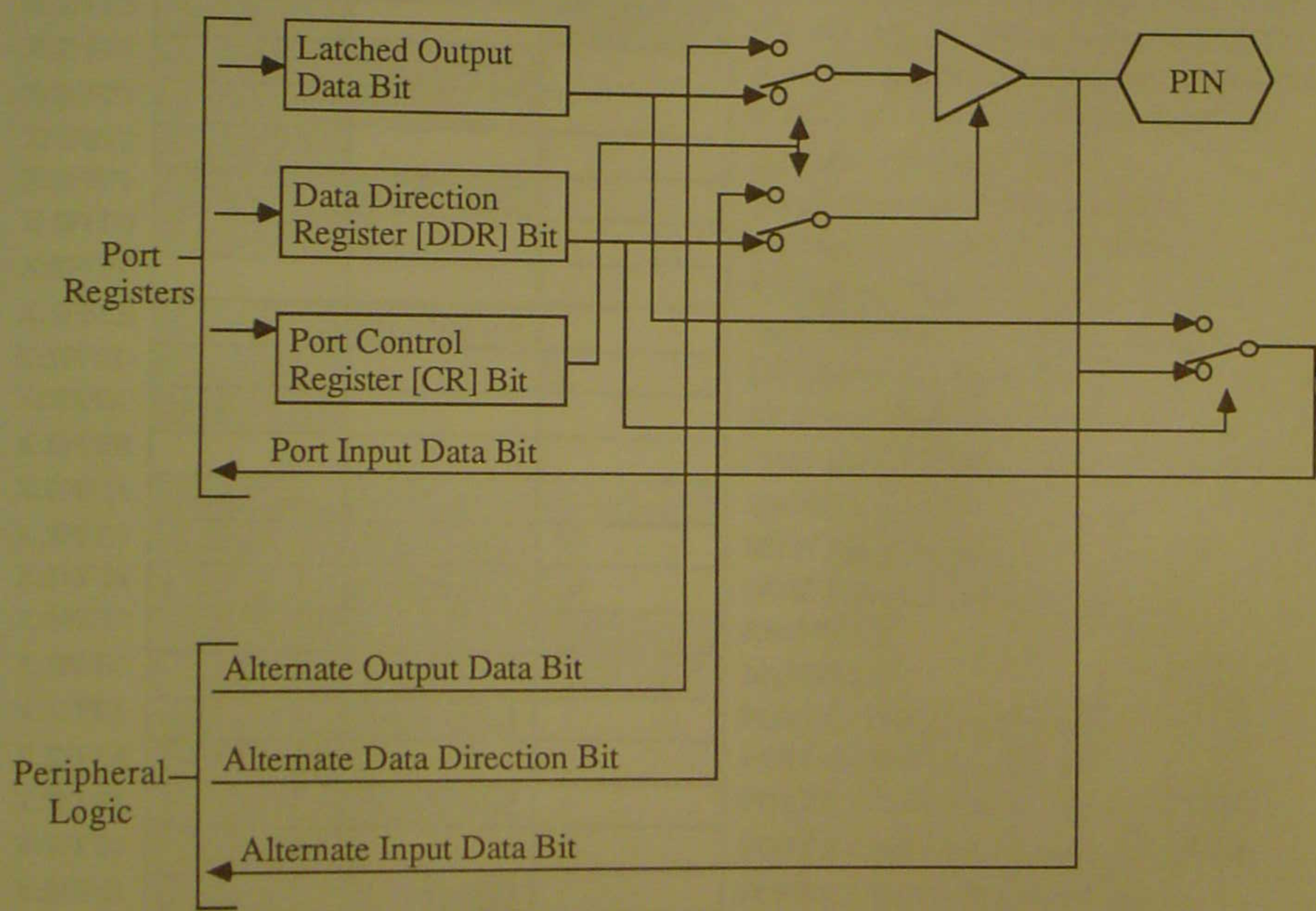



LEGEND = Read as Zeros, Write as Don't Care.

MTA5-7010

# PORTS B AND C CONTROL AND DIRECTION BITS

| Port Control Register Bit | Data Direction Register Bit | Pin Function |
|---|---|---|
| 0 | 0 | Port Input Pin |
| 0 | 1 | Port Output Pin |
| 1 | X | Alternate Function |




MTA5-7100

# PARALLEL PORT B

## X:$FFE0 PORT B Control Register (PBC)

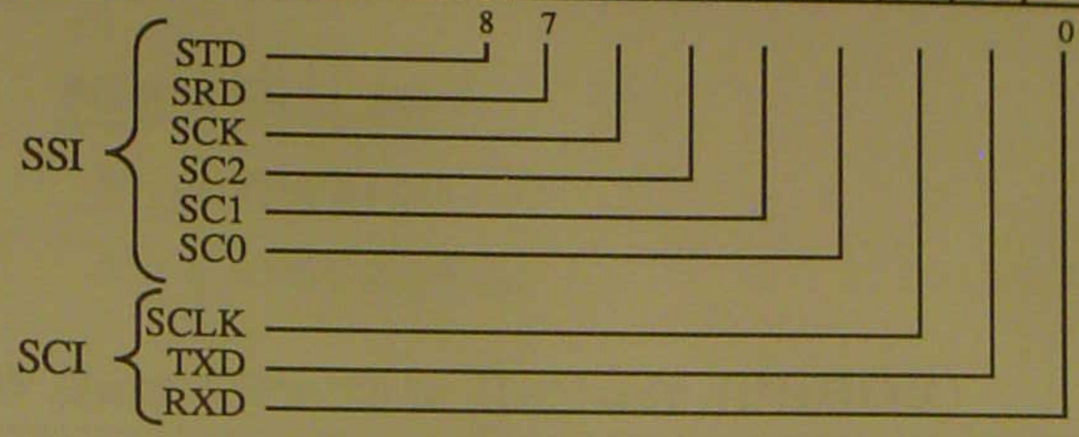
| 23 | | | | | | | | | | | | | | | | | | | | | | 16 15 | | | | | | | | | | | | | | | | | | |

# PARALLEL PORT C

## X:$FFE1 PORT C Control Register (PCC)

| 23 | | | | | | | 16 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CC8 | CC7 | CC6 | CC5 | CC4 | CC3 | CC2 | CC1 | CC0 |

| CCx | FUNCTION |
|---|---|
| 0 | Parallel I/O |
| 1 | Serial Interface |



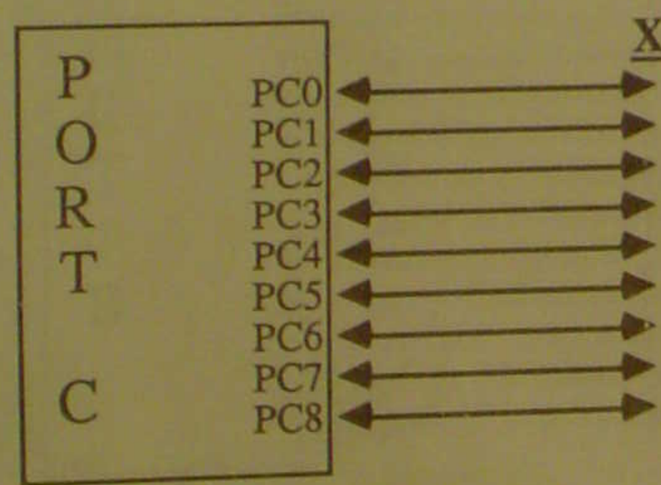

## X:$FFE3 PORT C Data Direction Register (PCDDR)

| 23 | | | | | | | 16 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CD8 | CD7 | CD6 | CD5 | CD4 | CD3 | CD2 | CD1 | CD0 |

| CDx | DATA DIRECTION |
|---|---|
| 0 | Input |
| 1 | Output |

## X:$FFE5 PORT C Data Register (PCD)

| 23 | | | | | | | 16 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PC8 | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |

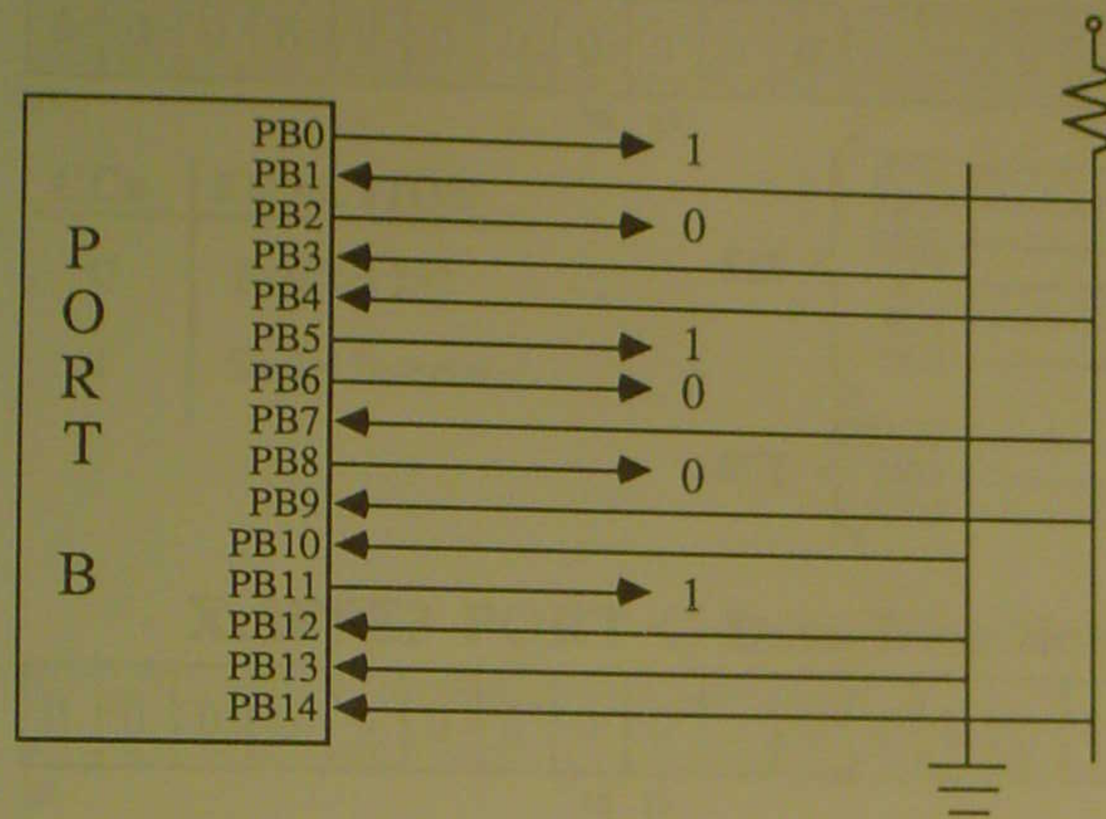



| Enabled by bits in X:$FFE1 | Direction Selected by X:$FFE3 | Input/Output Performed by X:$FFE5 |
|---|---|---|
| CC0 | CD0 | PC0 |
| CC1 | CD1 | PC1 |
| CC2 | CD2 | PC2 |
| CC3 | CD3 | PC3 |
| CC4 | CD4 | PC4 |
| CC5 | CD5 | PC5 |
| CC6 | CD6 | PC6 |
| CC7 | CD7 | PC7 |
| CC8 | CD8 | PC8 |

MTA5-7120

# EXERCISE 13

## ON CHIP PERIPHERALS




1. For this circuit , write the values in the PORT B Control Register, Data Direction Register and Data Register.

### X:$FFE0 PORT B Control Register (PBC)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | | | | | | | 16 | 15 | | | | | | | 8 | 7 | | | | | | | BC 0 |

### X:$FFE2 PORT B Data Direction Register (PBDDR)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | | | | | | | 16 | 15 | BD 14 | BD 13 | BD 12 | BD 11 | BD 10 | BD 9 | BD 8 | BD 7 | BD 6 | BD 5 | BD 4 | BD 3 | BD 2 | BD 1 | BD 0 |

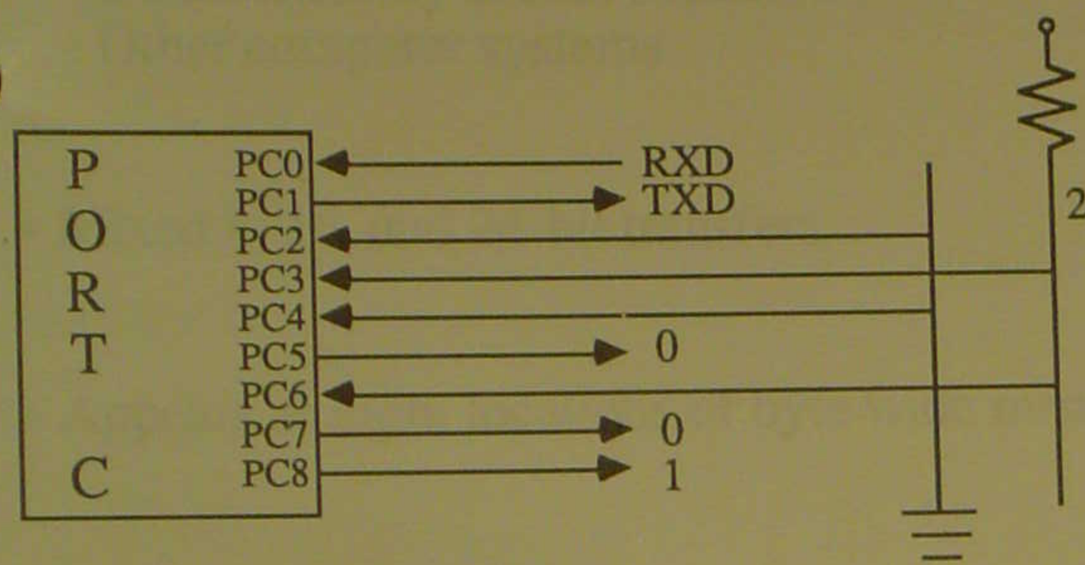
### X:$FFE4 PORT B Data Register (PBD)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | | | | | | | 16 | 15 | PB 14 | PB 13 | PB 12 | PB 11 | PB 10 | PB 9 | PB 8 | PB 7 | PB 6 | PB 5 | PB 4 | PB 3 | PB 2 | PB 1 | PB 0 |

MTA5-7190

# EXERCISE 13

## ON CHIP PERIPHERALS (Cont.)




2. For this circuit , write the values in the PORT C Control Register, Data Direction Register and Data Register. If desired use "X" for Don't Care.

### X:$FFE1 PORT C Control Register (PCC)

| 23 | | | | | | | | | | | | | | | 16 | 15 | | | | | | | | | CC 8 | CC 7 | CC 6 | CC 5 | CC 4 | CC 3 | CC 2 | CC 1 | CC 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | |

### X:$FFE3 PORT C Data Direction Register (PCDDR)

| 23 | | | | | | | | | | | | | | | 16 | 15 | | | | | | | | | CD 8 | CD 7 | CD 6 | CD 5 | CD 4 | CD 3 | CD 2 | CD 1 | CD 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | |

### X:$FFE5 PORT C Data Register (PCD)

| 23 | | | | | | | | | | | | | | | 16 | 15 | | | | | | | | | PC 8 | PC 7 | PC 6 | PC 5 | PC 4 | PC 3 | PC 2 | PC 1 | PC 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | |

MTA5-7195

# HOST INTERFACE FEATURES

- Allows the DSP56000 to be a memory mapped peripheral to
  - All popular microprocessors
  - Direct memory access controllers
  - Other computer systems

- Mixed 8, 16, and 24-bit transfers.

- Appears as eight locations of byte wide memory

- Fifteen pin interface

- 10.25 Megabytes per second transfer rate (Max.)
  - 8 Megabyte per second burst data transfer rate
  - 1.67 Megabyte per second interrupt driven data transfer rate

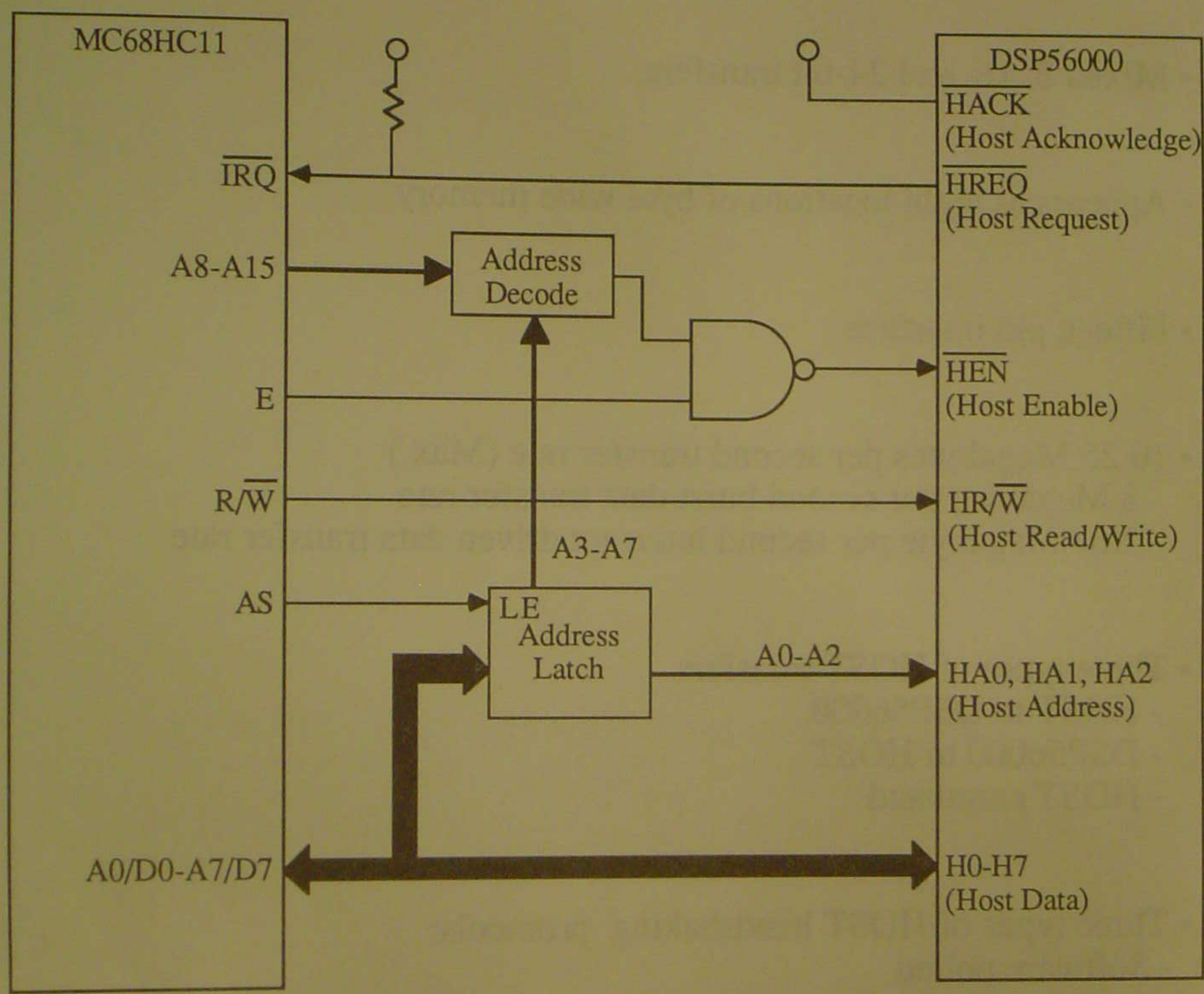
- Three types of HOST transfers
  - HOST to DSP56000
  - DSP56000 to HOST
  - HOST command

- Three types of HOST handshaking protocols:
  - Software polled
  - Interrupt driven (Fast or Long)
  - Direct memory access

- HOST interrupt has M68000 family compatible interrupt acknowledge vector.

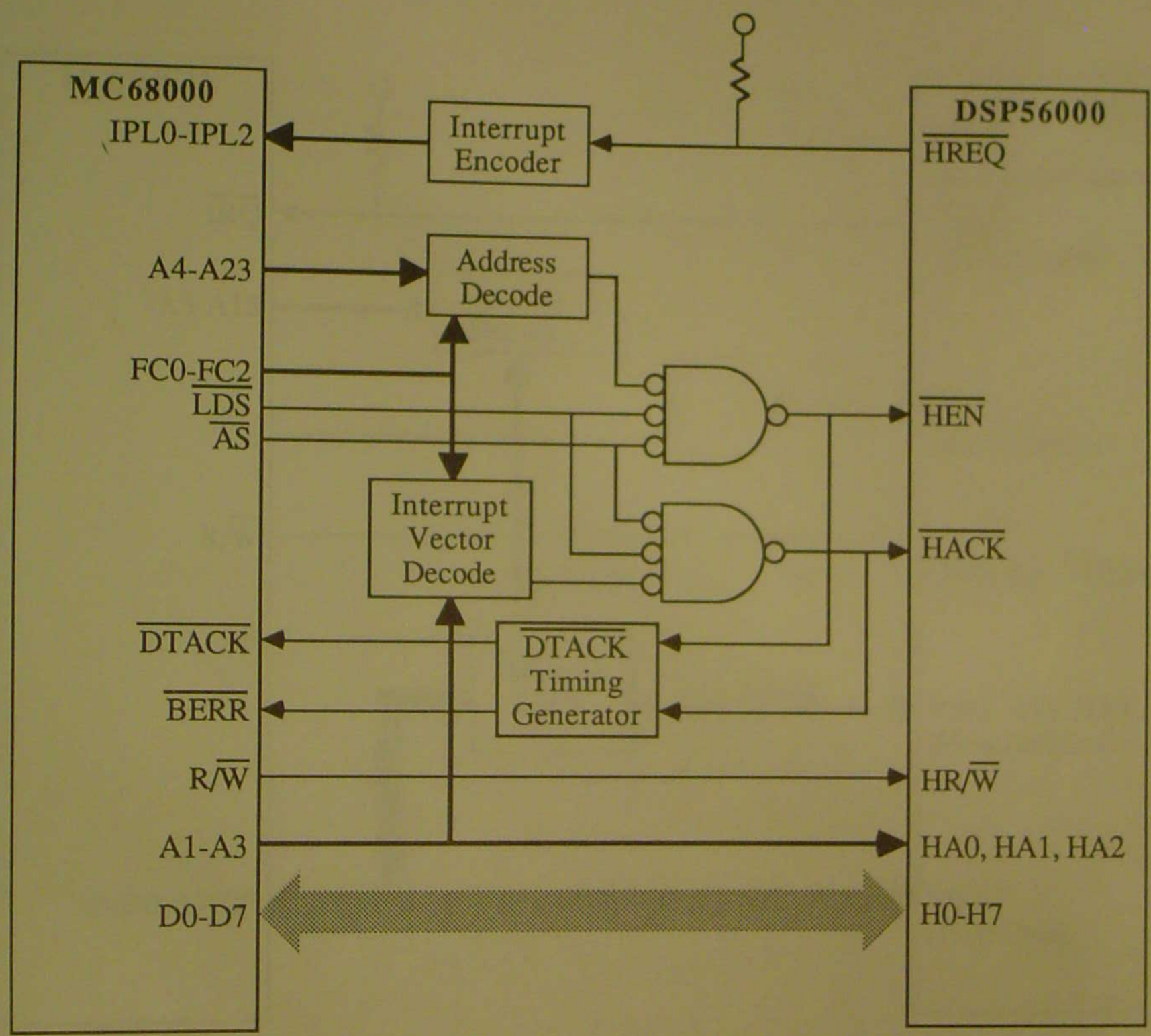
MTA5-7200-1

# MC68HC11 TO DSP56000 HOST INTERFACE



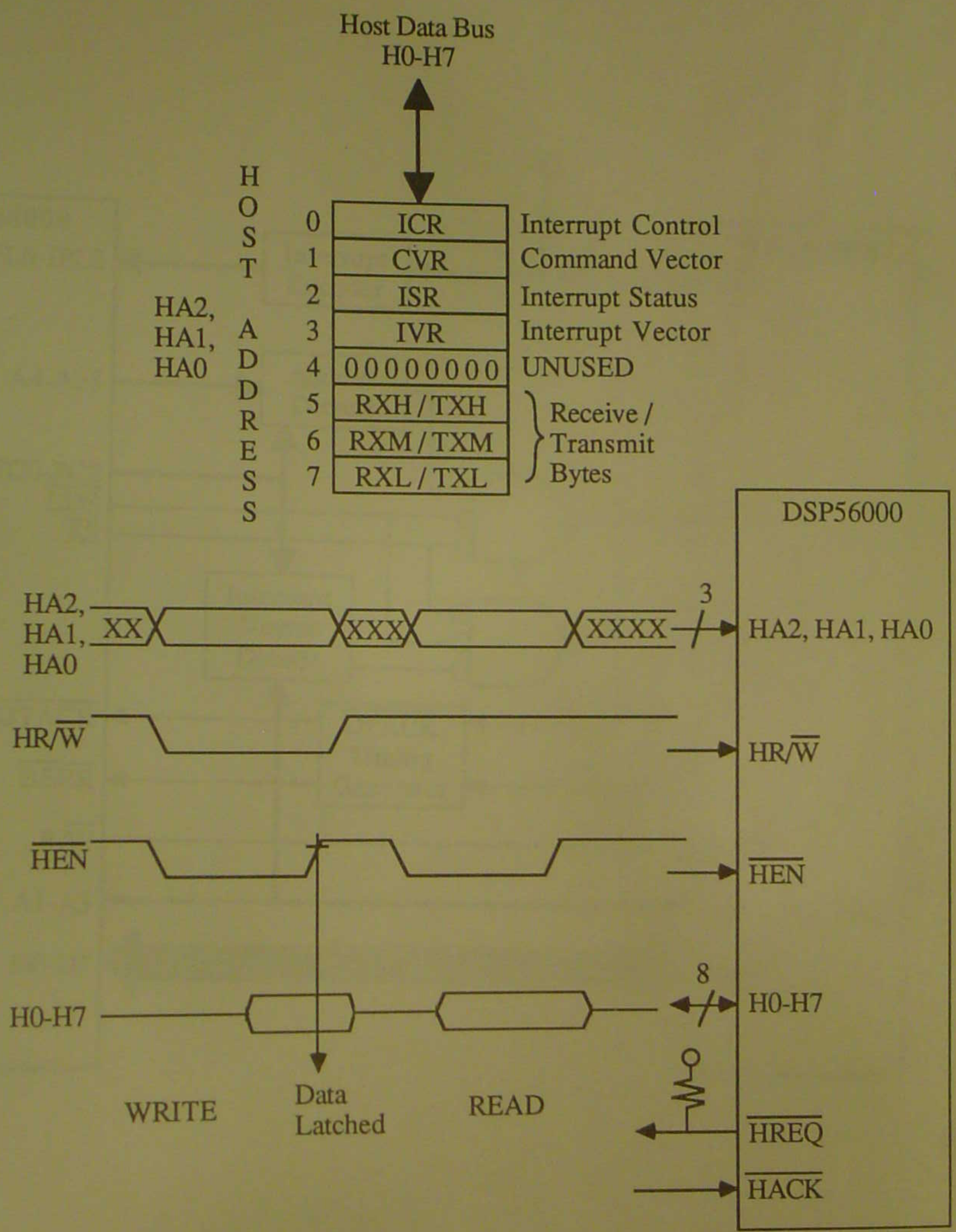

MTA5-7210

# MC68000 TO DSP56000 HOST INTERFACE




MTA5-7220

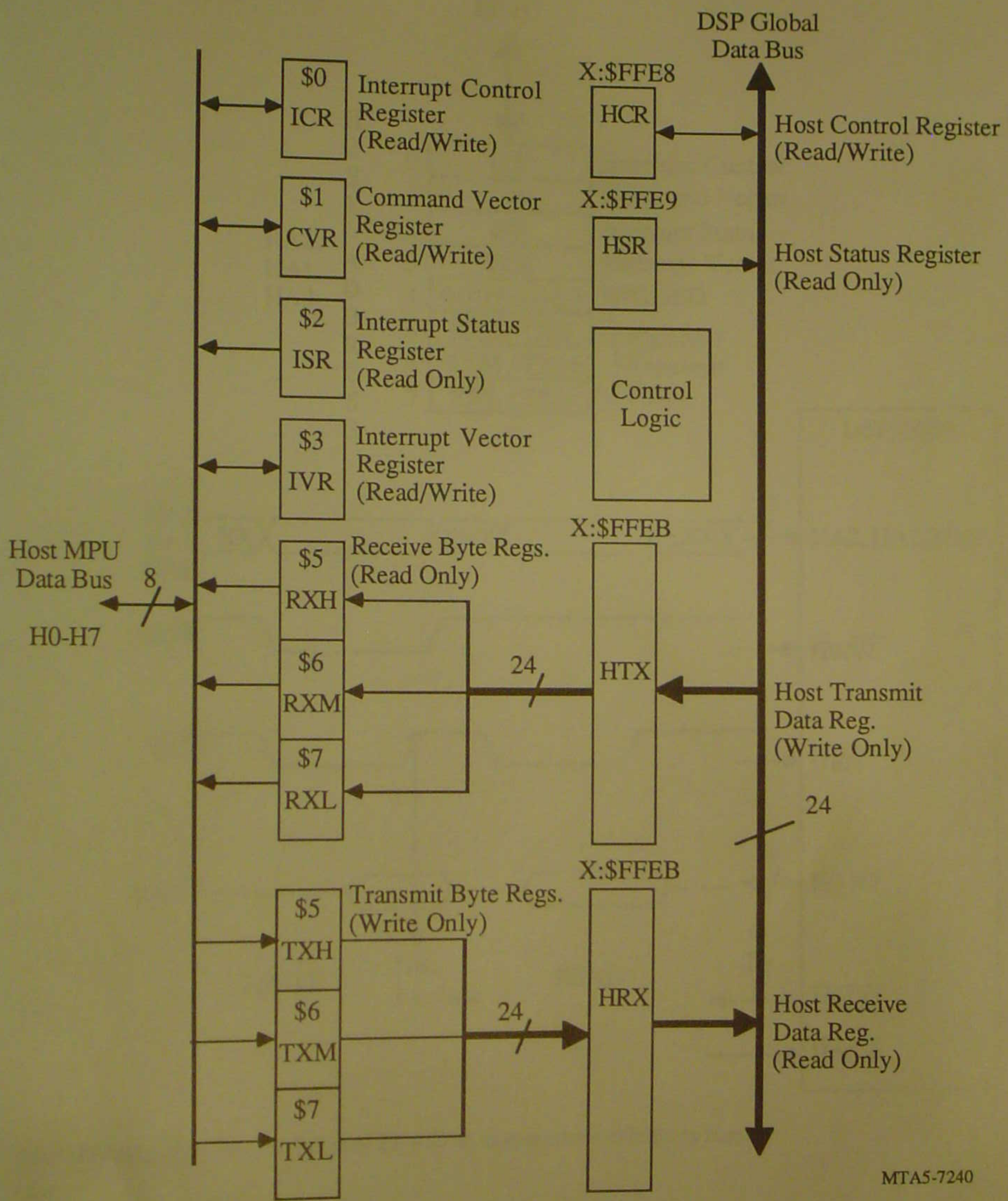
# HOST TRANSFER TIMING




Note: Minimum $\overline{\text{HEN}}$ assertion time of 2T + 60 nS to guarantee stable status flags.

MTA5-7230-1

# HOST INTERFACE BLOCK DIAGRAM



# HOST VIEW OF DSP56000

| | 7 | Modes | | Flags | | | | 0 |
|---|---|---|---|---|---|---|---|---|
| $0 | INIT | HM1 | HM0 | HF1 | HF0 | 0 | TREQ | RREQ |
| | Cleared on RESET | | | | | | | |

**Interrupt Control Register (ICR)**
**(Read/Write )**

| | 7 | | | 4 3 | 0 |
|---|---|---|---|---|---|
| $1 | HC | 0 | 0 | Host Vector | |
| | | | | $12 on RESET | |

**Command Vector Register (CVR)**
**(Read/Write )**

| | 7 | | | Flags | | | | 0 |
|---|---|---|---|---|---|---|---|---|
| $2 | HREQ | DMA | 0 | HF3 | HF2 | TRDY | TXDE | RXDF |

**Interrupt Status Register (ISR)**
**(Read Only)**

| | 7 | 4 3 | 0 |
|---|---|---|---|
| $3 | Interrupt Vector Number | | |
| | $0F on RESET | | |

**Interrupt Vector Register (IVR)**
**(Read/Write)**

**Receive Byte Registers (RXH:RXM:RXL)**
**(Read Only)**

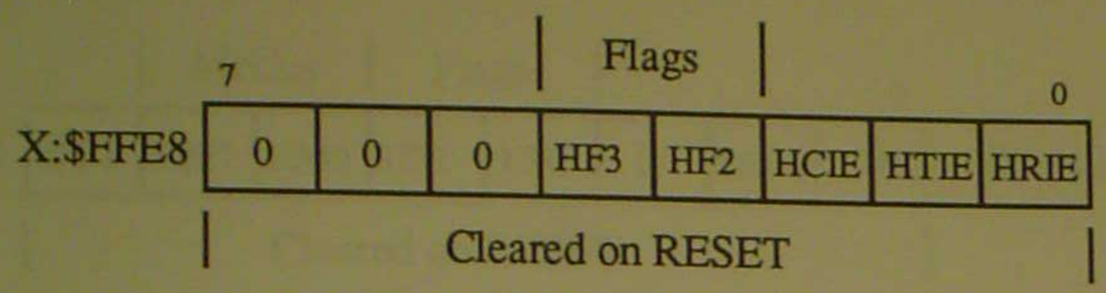
| 23 $5 16 | 15 $6 8 | 7 $7 0 |
|---|---|---|
| RXH<br>Receive High Byte | RXM<br>Receive Middle Byte | RXL<br>Receive Low Byte |
| 7 0 | 7 0 | 7 0 |

**Transmit Byte Registers (TXH:TXM:TXL)**
**(Write Only)**

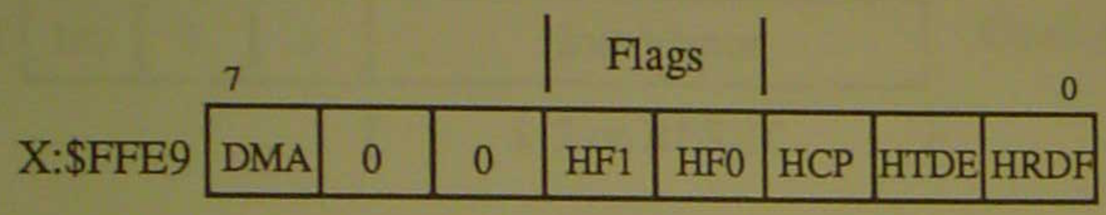
| 23 $5 16 | 15 $6 8 | 7 $7 0 |
|---|---|---|
| TXH<br>Transmit High Byte | TXM<br>Transmit Middle Byte | TXL<br>Transmit Low Byte |
| 7 0 | 7 0 | 7 0 |

MTA5-7250

# DSP56000 VIEW OF HOST INTERFACE

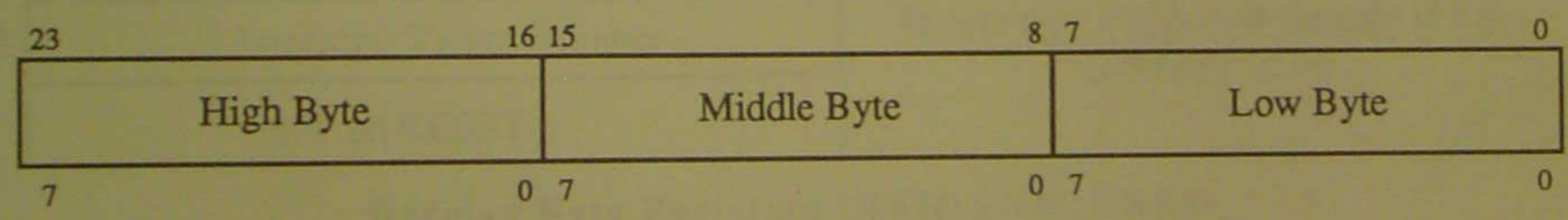



**Host Control Register (HCR)**
**(Read/Write)**

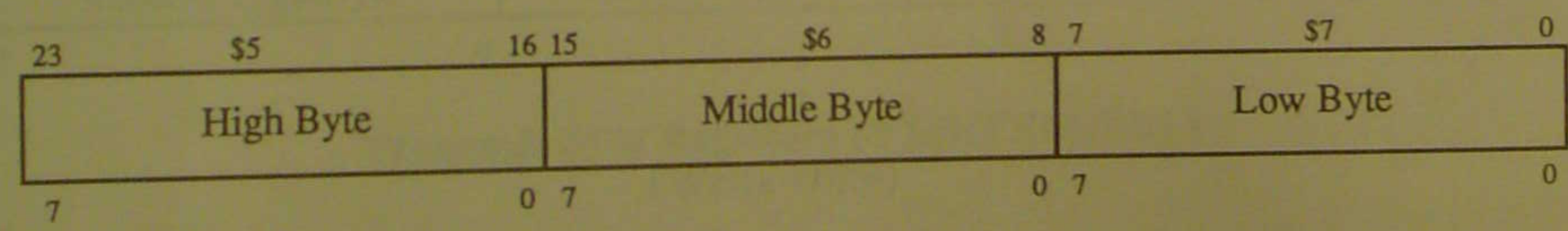



**Host Status Register (HSR)**
**(Read Only)**

**Host Receive Data Register (HRX)**
**X:$FFEB (Read Only)**




**Host Transmit Data Register (HTX)**
**X:$FFEB (Write Only)**




MTA5-7260

# HOST COMMAND

## View From Host

1. Write CVR with desired HV
2. Set HC = 1.

$1 Command Vector Register (CVR)

| 7 | | | 0 |
|---|---|---|---|
| 1 | 0 | 0 | Host Vector (HV) |

HC
Host Command

$12

Exception Vector Address = HV X 2

## View From DSP56000

3. HCP is set until Exception Processing starts.

X:$FFE9 Host Status Register (HSR)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| DMA | 0 | 0 | HF1 | HF0 | 1 | HTDE | HRDF |

HCP
Host Command
Pending

4. Host Command is Masked until HCIE = 1

X:$FFE8 Host Control Register (HCR)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | HF3 | HF2 | 1 | HTIE | HRIE |

HCIE
Host Command
Interrupt Enable

### Exception Vector Table

| | |
|---|---|
| P:$0000 | |
| | |
| | |
| | |
| | |
| | |
| P:$0024 | Host Command Default Vector |
| | Available for Host Command |
| | |
| | |
| | Available for Host Command |
| P:$003E | Available for Host Command |

Fast Interrupt
or
Long Interrupt

## Characteristics of Host Commands

Steps 1 and 2 may be combined into one step. HC and HV can be written simultaneously.

HV may be Programmed to any Exception Vector.

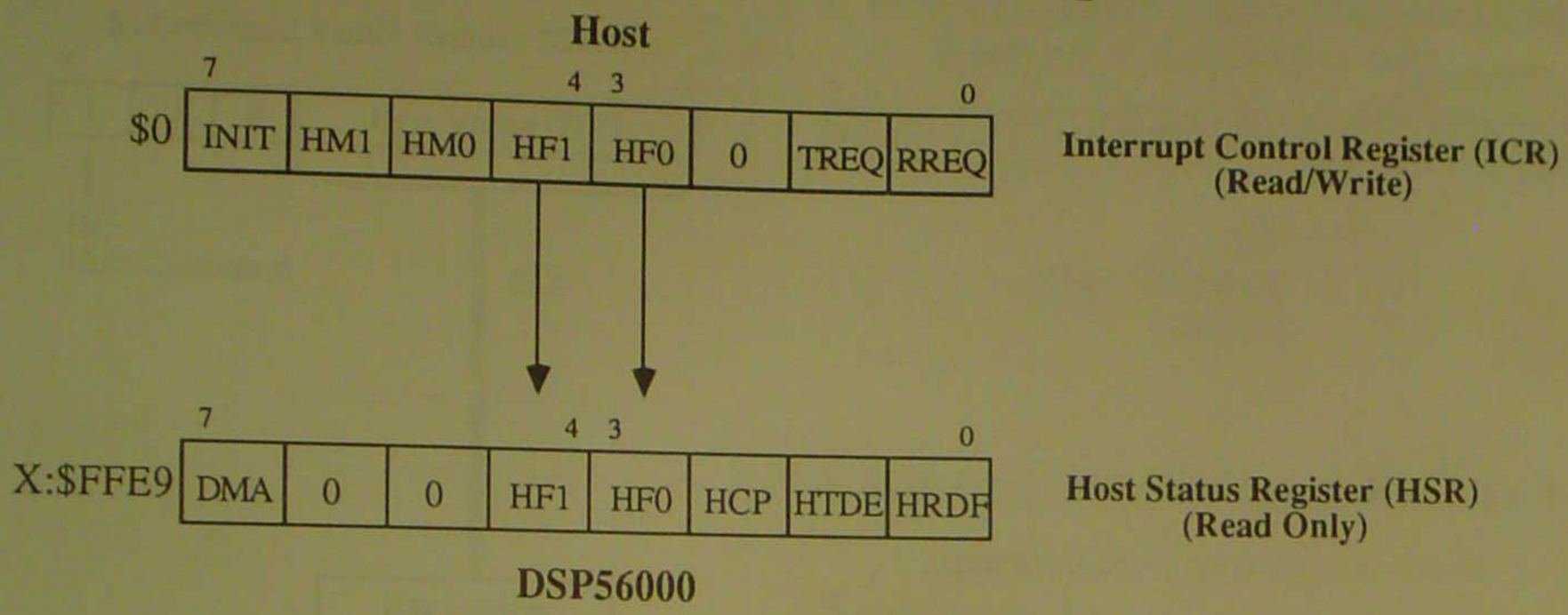
Upon RESET, HV = $12. Exception Vector $0024.

It is recommended that HV = 0 (RESET) not be used.

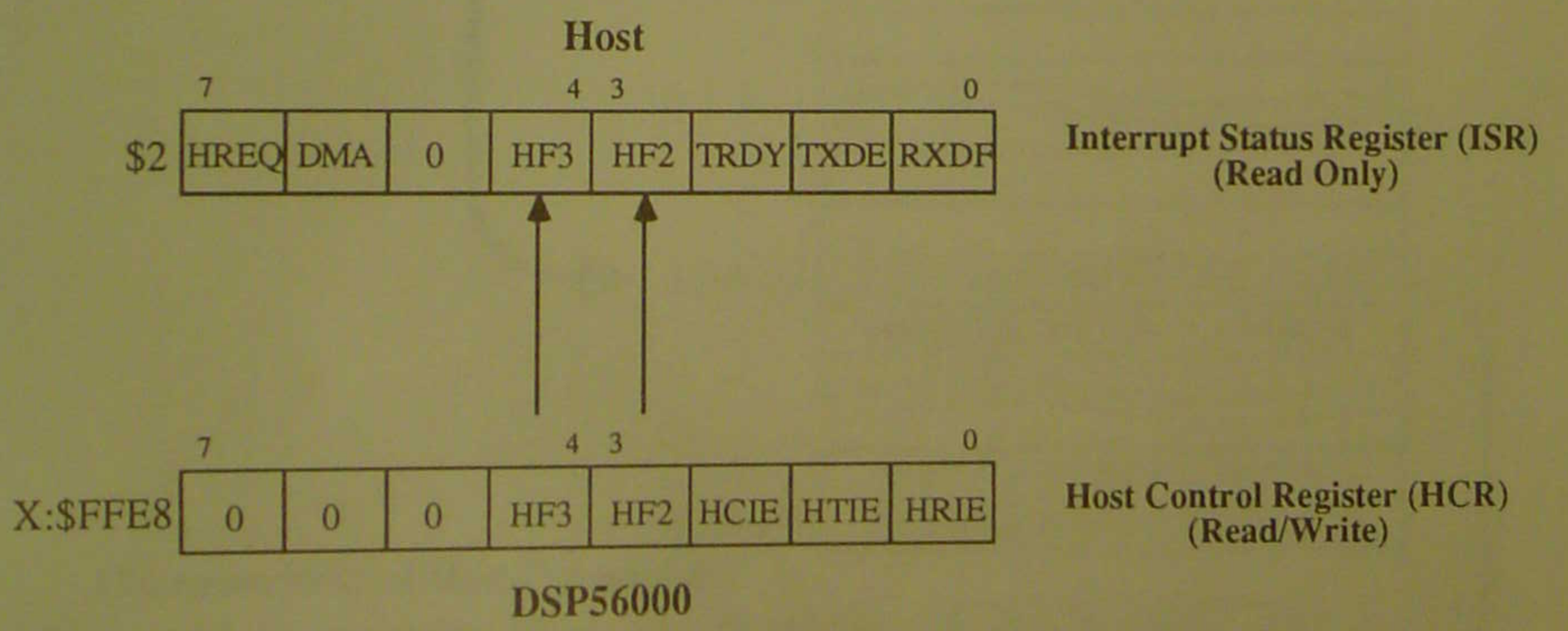
MTA5-7280-1

# HOST FLAGS

## Host to DSP56000 Status Flags




## DSP56000 to Host Status Flags




MTA5-7285-1

# DATA TRANSFER FROM HOST TO DSP56000

## View from Host

1. When TXDE = 1, TBR is empty

$2 Interrupt Status Register (ISR)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| HREQ | DMA | 0 | HF3 | HF2 | TRDY | 1 | RXDF |

TXDE — Transmit Data Register Empty

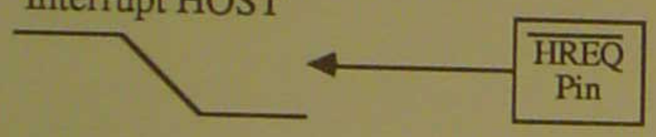
2. HOST may poll TXDE

$0 Interrupt Control Register (ICR)

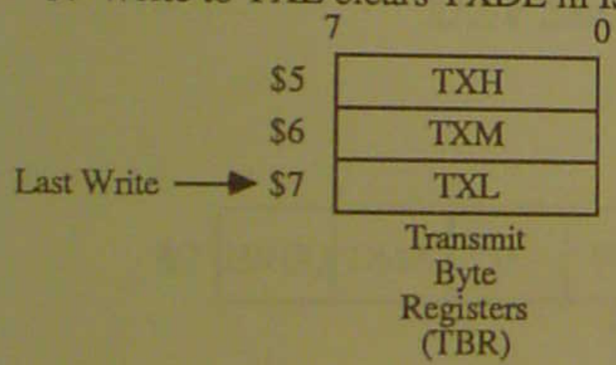
| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| INIT | HM1 | HM0 | HF1 | HF0 | 0 | 1 | RREQ |

TREQ — Transmit Request Enable

3. If TREQ = 1 then $\overline{\text{HREQ}}$ pin is asserted to Interrupt HOST




4. HOST writes data to transmit Byte Registers
5. Write to TXL clears TXDE in ISR




## View from DSP56000

6. If DSP56000 has old data in HRX, then HRDF = 1

7. When DSP56000 reads HRX, then HRDF = 0

X:$FFE9 Host Status Register (HSR)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| DMA | 0 | 0 | HF1 | HF0 | HCP | HTDE | 0 |

HRDF — Host Receive Data Full

8. When TXDE = 0 and HRDF = 0, then transfer occurs

X:$FFEB Host Receive Data Register (HRX)

| 23 | | 0 |
|---|---|---|
| High Byte | Middle Byte | Low Byte |

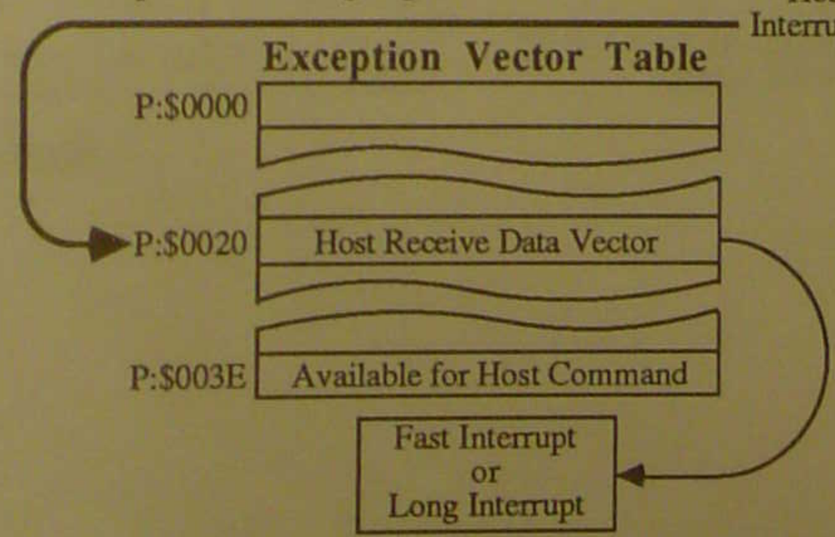
9. The transfer sets HRDF for the DSP56000 to poll

X:$FFE8 Host Control Register (HCR)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | HF3 | HF2 | HCIE | HTIE | 1 |

HRIE — Host Receive Interrupt Enable

10. If HRDF = 1, and Interrupts are enables then Exception Processing begins

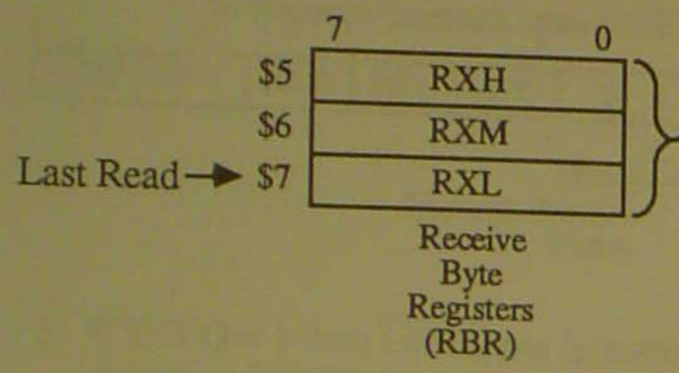



MTA5-7286-1

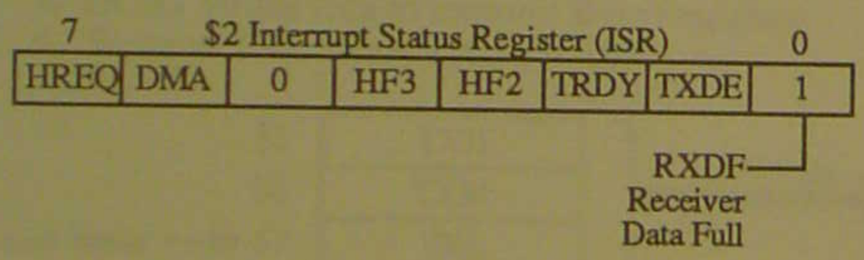
# DATA TRANSFER FROM DSP56000 TO HOST

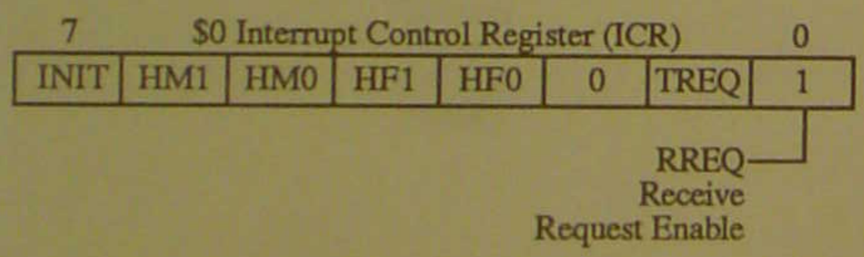
## View From Host

5. Read of RXL by HOST Clears RXDF in ISR
6. When RXDF = 0 and HTDE = 0 then TRANSFER Occurs

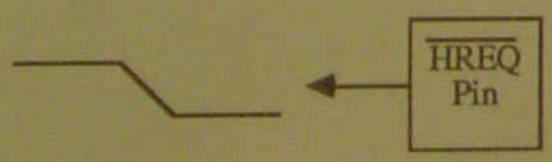



7. The TRANSFER Sets RXDF for the HOST to Poll



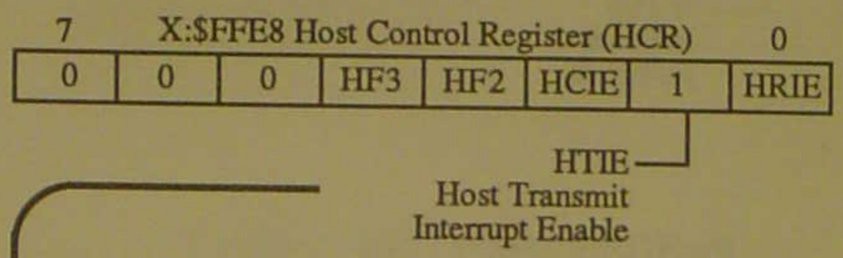

8. If RREQ = 1 then $\overline{\text{HREQ}}$ Pin is Asserted to Interrupt HOST



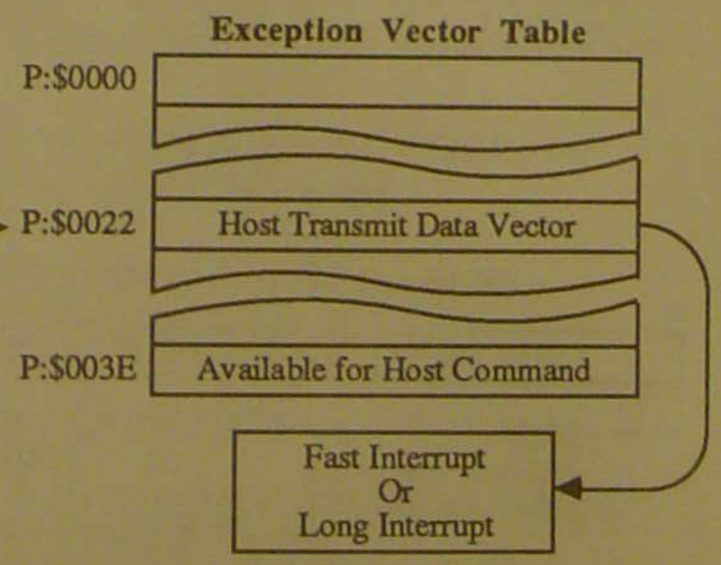

## View From DSP56000

1. When HTDE = 1, then HTX is Empty

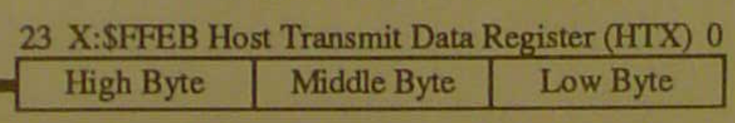
2. DSP56000 may Poll HTDE




3. If HTIE = 1, and Interrupts are Enabled, then Exception Processing Begins

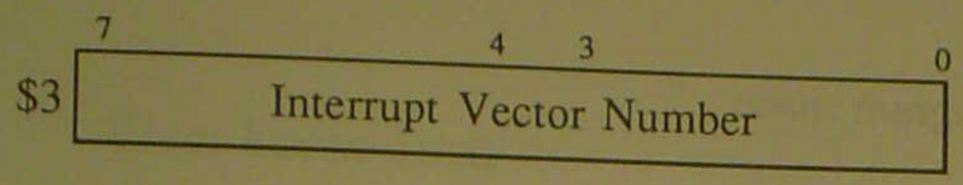



4. DSP56000 Writes Data to HTX. This clears HTDE in HSR

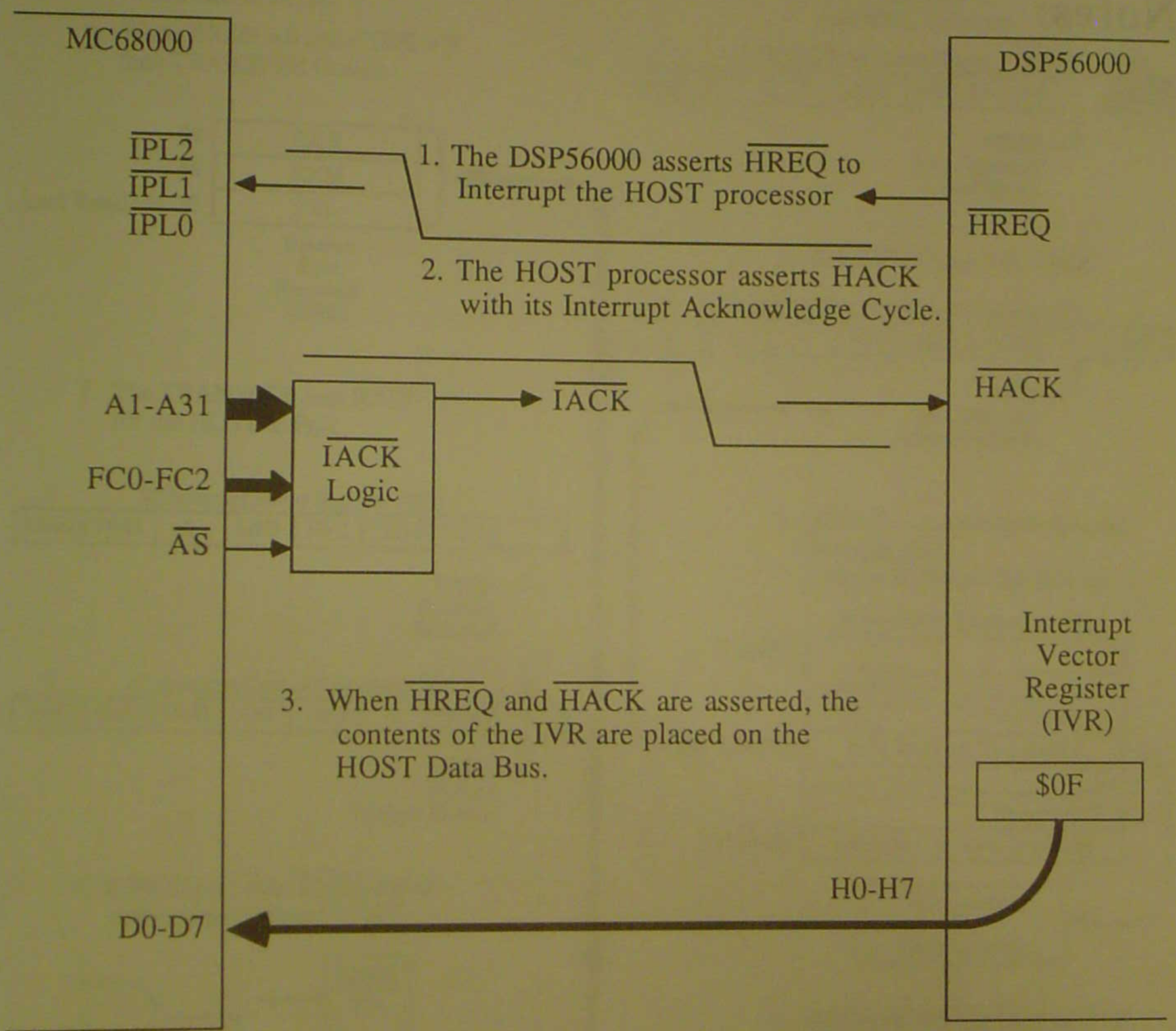



MTA5- 7288-1

# HOST EXCEPTION




**Interrupt Vector Register (IVR) (Read/Write)**




## Characteristics of the HOST Exception

Interrupt Vector Register may be programmed to any value from 0 to 255.

On RESET it is set to $0F. This corresponds to the Uninitialized Exception Vector in the M68000 Family.

To use the HOST Exception, the DMA Mode must be disabled.

MTA5-7290-1

# EXERCISE 14

## ON CHIP PERIPHERALS

1. The Host interface permits _____ , _____ , and _____ bit transfers.

2. The three types of Host transfers are:

   A. _____ to _____

   B. _____ to _____

   C. _____ _____

3. The three types of Host handshaking protocols are:

   A. _____ _____

   B. _____ _____

   C. _____ _____ _____

4. The Host interface has _____ user defined Host status flags.

5. The Host Vector in the Command Vector Register is multiplied by _____ to form the Exception Vector Address. It is initialized to _____ on RESET.

6. The Host interface can interrupt the Host processor by using the _____ pin.

7. The Host interface can provide an interrupt vector on the host data bus when both _____ and _____ are asserted. This feature is not available in the _____ mode. The Host Exception Vector is initialized to _____ on RESET.

MTA5-7295-1

# SERIAL COMMUNICATIONS INTERFACE FEATURES

- Three Pin Interface;
  - TXD - Transmit Data
  - RXD - Receive Data
  - SCLK - Serial Clock

- 320 KBit per second NRZ Asynchronous Communications Interface

- MULTIDROP Mode for Multi-Processor Systems

- Two WAKE-UP Modes
  - IDLE LINE
  - ADDRESS BIT

- 2.5MBit per second Synchronous Serial Mode

- On Chip or External Baud Rate Generation

- Periodic Interrupt Timer

- Separate Interrupt Vectors for each Interrupt Source

- Four Interrupt Priority Levels

- Fast or Long Interrupts

MTA5-7300

# SERIAL COMMUNICATIONS INTERFACE REGISTER SET

**X:$FFF0 - SCI Interface Control Register (SCR) (Read/Write)**

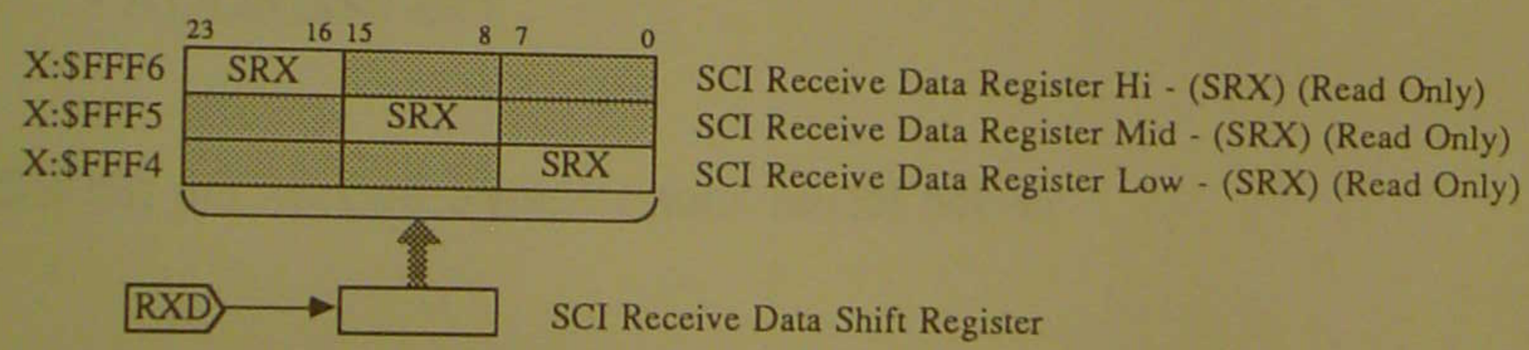
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | TMIE | TIE | RIE | ILIE | TE | RE | WOMS | RWU | WAKE | SBK | 0 | WDS2 | WDS1 | WDS0 |

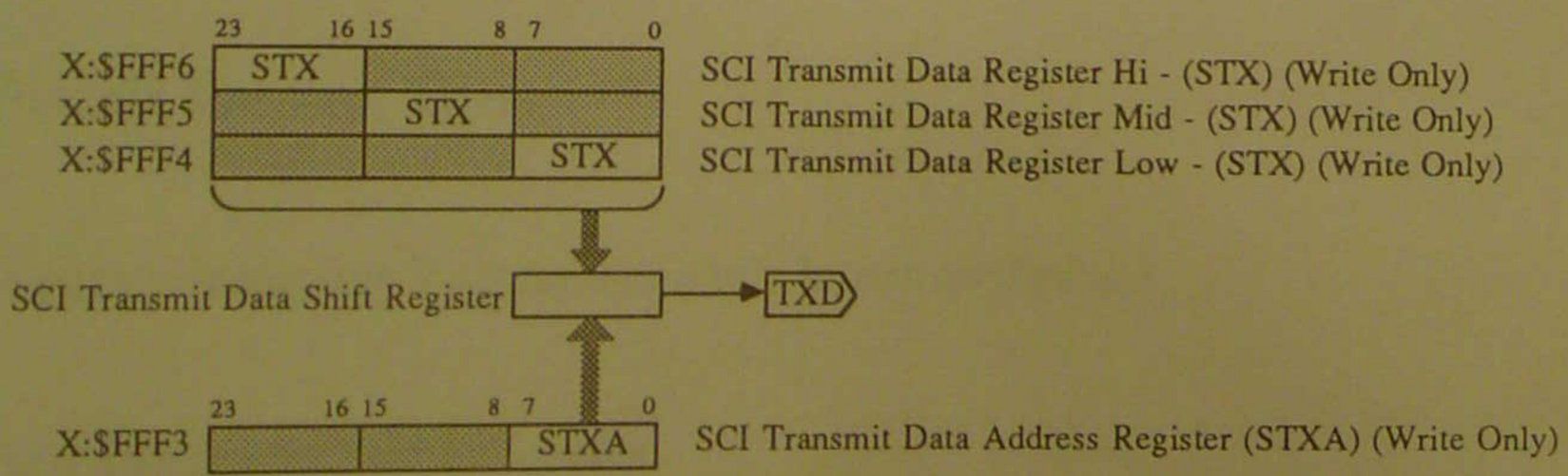
**X:$FFF1 - SCI Interface Status Register (SSR) (Read Only)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | R8 | FE | PE | OR | IDLE | RDRF | TDRE | TNRE |

**X:$FFF2 - SCI Clock Control Register (SCCR) (Read/Write)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TCM | RCM | SCP | COD | CD11 | CD10 | CD9 | CD8 | CD7 | CD6 | CD5 | CD4 | CD3 | CD2 | CD1 | CD0 |




MTA5- 7310-1

# SERIAL FORMATS

## Mode 0

| | 2 | 1 | 0 |
|---|---|---|---|
| X:$FFF0 | 0 | 0 | 0 |
| | WDS2 | WDS1 | WDS0 |

8 Bit Synchronous Data (Shift Register Mode)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |

← One Byte from Shift Register →

## Mode 2

| | 2 | 1 | 0 |
|---|---|---|---|
| X:$FFF0 | 0 | 1 | 0 |
| | WDS2 | WDS1 | WDS0 |

10 Bit Asynchronous (1 Start Bit - 8 Data Bits - 1 Stop Bit)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Start Bit | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7,or Data Type | Stop Bit |

## Mode 4

| | 2 | 1 | 0 |
|---|---|---|---|
| X:$FFF0 | 1 | 0 | 0 |
| | WDS2 | WDS1 | WDS0 |

11 Bit Asynchronous
(1 Start Bit - 8 Data Bits - Even Parity Bit - 1 Stop Bit)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Start Bit | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7,or Data Type | Even Parity | Stop Bit |

## Mode 5

| | 2 | 1 | 0 |
|---|---|---|---|
| X:$FFF0 | 1 | 0 | 1 |
| | WDS2 | WDS1 | WDS0 |

11 Bit Asynchronous
(1 Start Bit - 8 Data Bits - Odd Parity Bit - 1 Stop Bit)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Start Bit | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7,or Data Type | Odd Parity | Stop Bit |

## Mode 6

| | 2 | 1 | 0 |
|---|---|---|---|
| X:$FFF0 | 1 | 1 | 0 |
| | WDS2 | WDS1 | WDS0 |

11 Bit Multidrop
(1 Start Bit - 8 Data Bits - Data Type Bit - 1 Stop Bit)

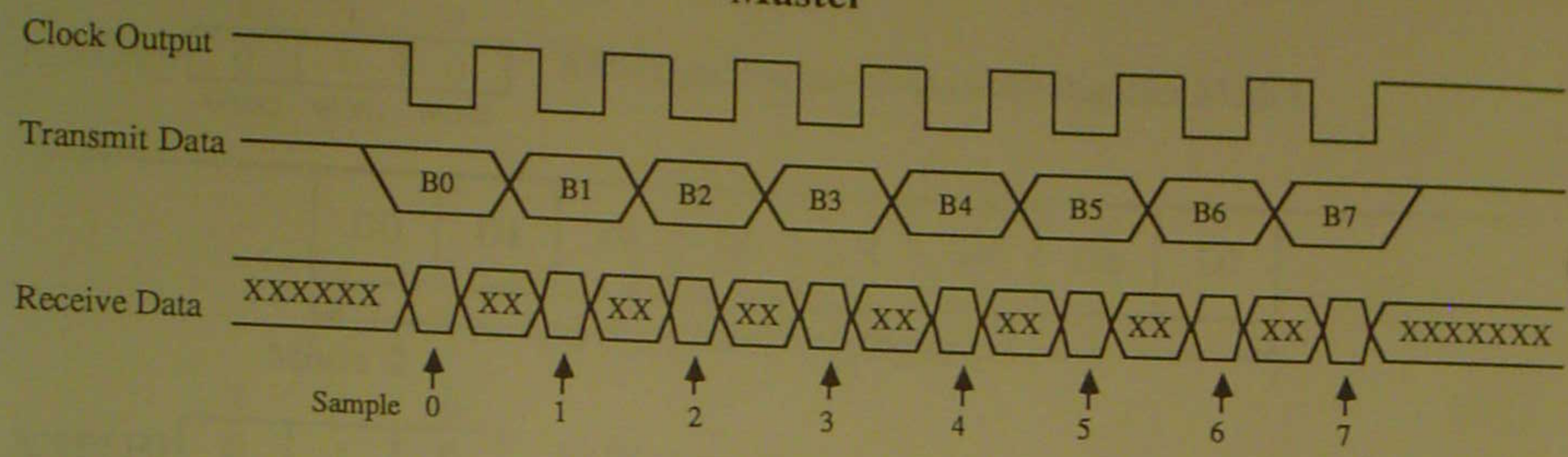
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Start Bit | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | Data Type | Stop Bit |

$$\text{Data Type} = \begin{cases} 1 = \text{Address Byte} \\ 0 = \text{Data Byte} \end{cases}$$

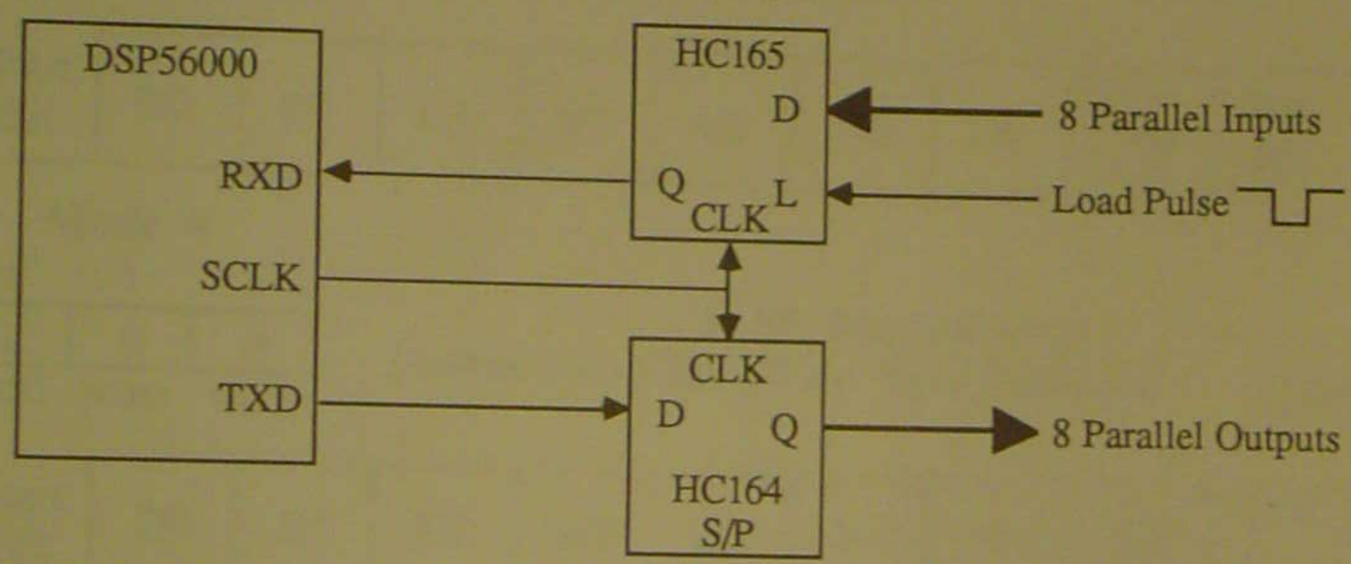
MTA5-7320

# SCI SYNCHRONOUS MODE

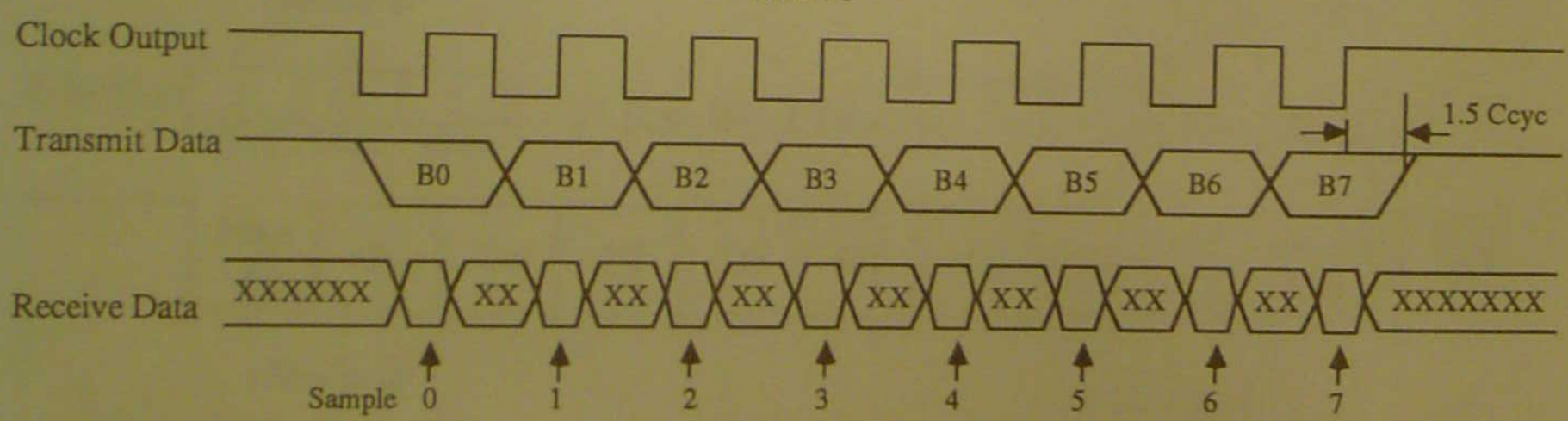
## Master



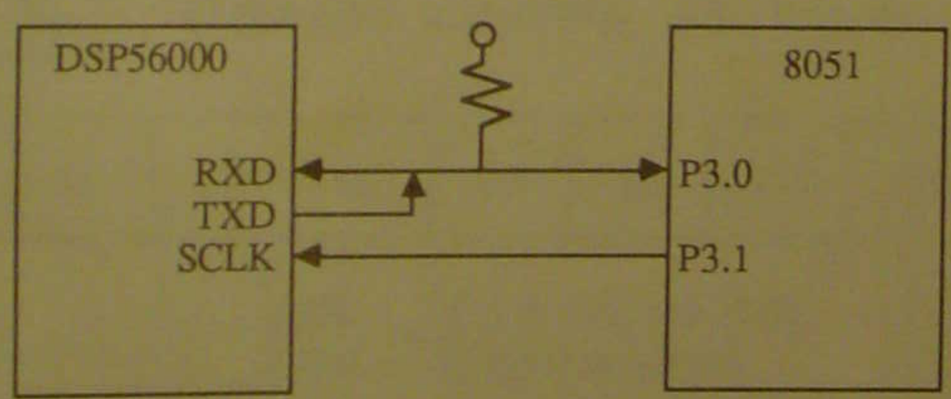

## Example




## Slave




**Example**




MTA5-7330

# SCI-TRANSMIT A CHARACTER

## Initialization

1. Hardware or Software RESET
2. Program SCR with Desired Mode and Features. Turn on Transmitter (TE = 1) Optionally Enable Transmitter Interrupts. (TIE = 1).

X:$FFF0-SCI Interface Control Register (SCR)(Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | TMIE | 1 | RIE | ILIE | 1 | RE | WOMS | RWU | WAKE | SBK | 0 | WDS2 | WDS1 | WDS0 |

TIE — Transmit Interrupt Enable

TE — Transmitter Enable

3. Set the TXD Bit in PCC to Enable the SCI Transmitter System.

X:$FFE1 Port C Control Register (PCC)

| 23 | | | | | | | 16 | 15 | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CC 8 | CC 7 | CC 6 | CC 5 | CC 4 | CC 3 | CC 2 | 1 | CC 0 |

SCI

| CCx | FUNCTION |
|---|---|
| 0 | PARALLEL I/O |
| 1 | SERIAL INTERFACE |

TXD — Transmit Data Pin Enable

4. Depending on the Setting of WDS2 to WDS0 in SCR, the Transmitter will Broadcast a Preamble of 10 or 11 "Ones".

## Transmission

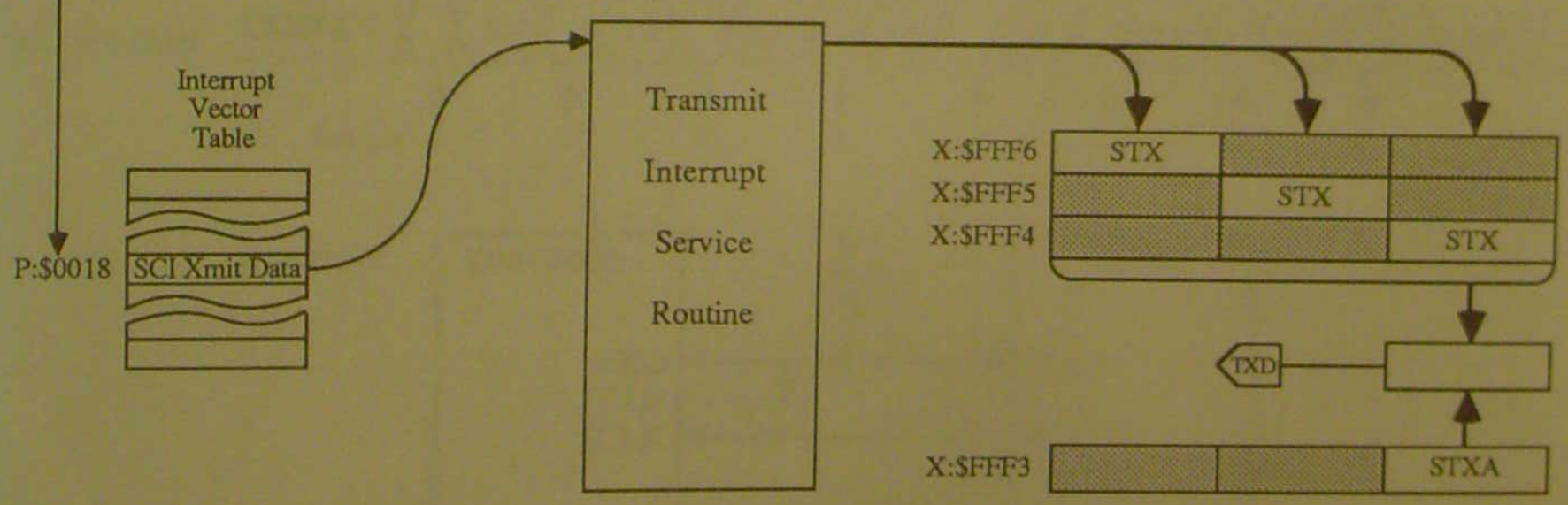
X:$FFF1 - SCI Interface Status Register (SSR)(Read Only)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | R8 | FE | PE | OR | IDLE | RDRF | 1 | 1 |

TDRE — Transmit Data Register Empty

TRNE — Transmitter Empty

5. When STX is empty, then TDRE = 1
6. When STX is empty and the Transmit Data Shift Register is empty then TRNE = 1
7. If TIE = 1 in SCR and TDRE = 1 in SSR, then an interrupt is generated.
8. Transmitter interrupt service routine stores next character into STX.
9. Writing STX(A) Clears TDRE in SSR.




MTA5- 7340

# SCI - RECEIVE A CHARACTER

## Initialization

1. Hardware or Software RESET

2. Program SCR with desired mode and features. Turn on Receiver (RE = 1). Optionally Enable Receiver Interrupts. (RIE = 1).

**X:$FFF0 - SCI Interface Control Register (SCR) (Read/Write)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | TMIE | TIE | 1 | ILIE | TE | 1 | WOMS | RWU | WAKE | SBK | 0 | WDS2 | WDS1 | WDS0 |
| | | | | RIE | | | RE | | | | | | | | |

3. Set the RXD bit in PCC to enable the SCI Receiver System.

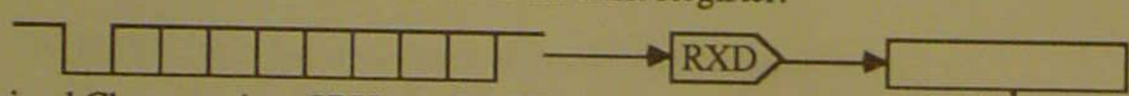
**X:$FFE1 PORT C Control Register (PCC)**

| 23 | | | | | | | 16 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CC 8 | CC 7 | CC 6 | CC 5 | CC 4 | CC 3 | CC 2 | CC 1 | 1 |
| | | | | | | | | | | | | | | | | | | | | | SCI | | RXD |

| CCx | Function |
|---|---|
| 0 | Parallel I/O |
| 1 | Serial Interface |

## Reception

4. The receiver is idle until a character is received in the Data Shift Register.

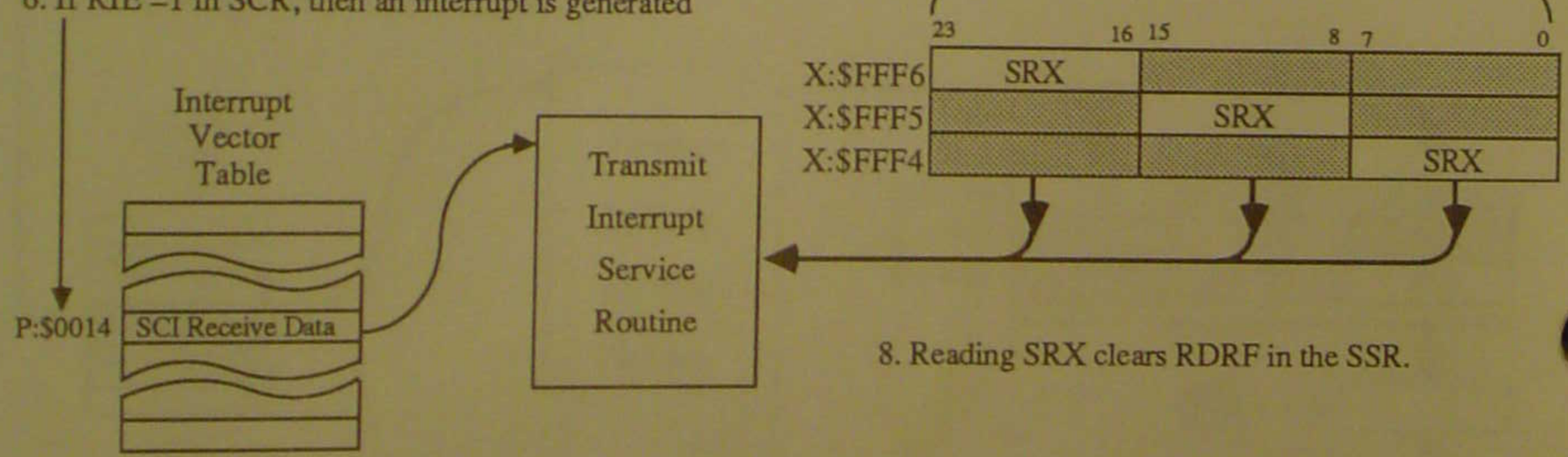



5. Transferring the received Character into SRX sets RDRF in the SSR.

X:$FFF1 - SCI Interface Status Register (SSR) (Read Only)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| R8 | FE | PE | OR | IDLE | 1 | TDRE | TNRE |
| | | | | | RDRF | | |

6. If RIE =1 in SCR, then an interrupt is generated




8. Reading SRX clears RDRF in the SSR.

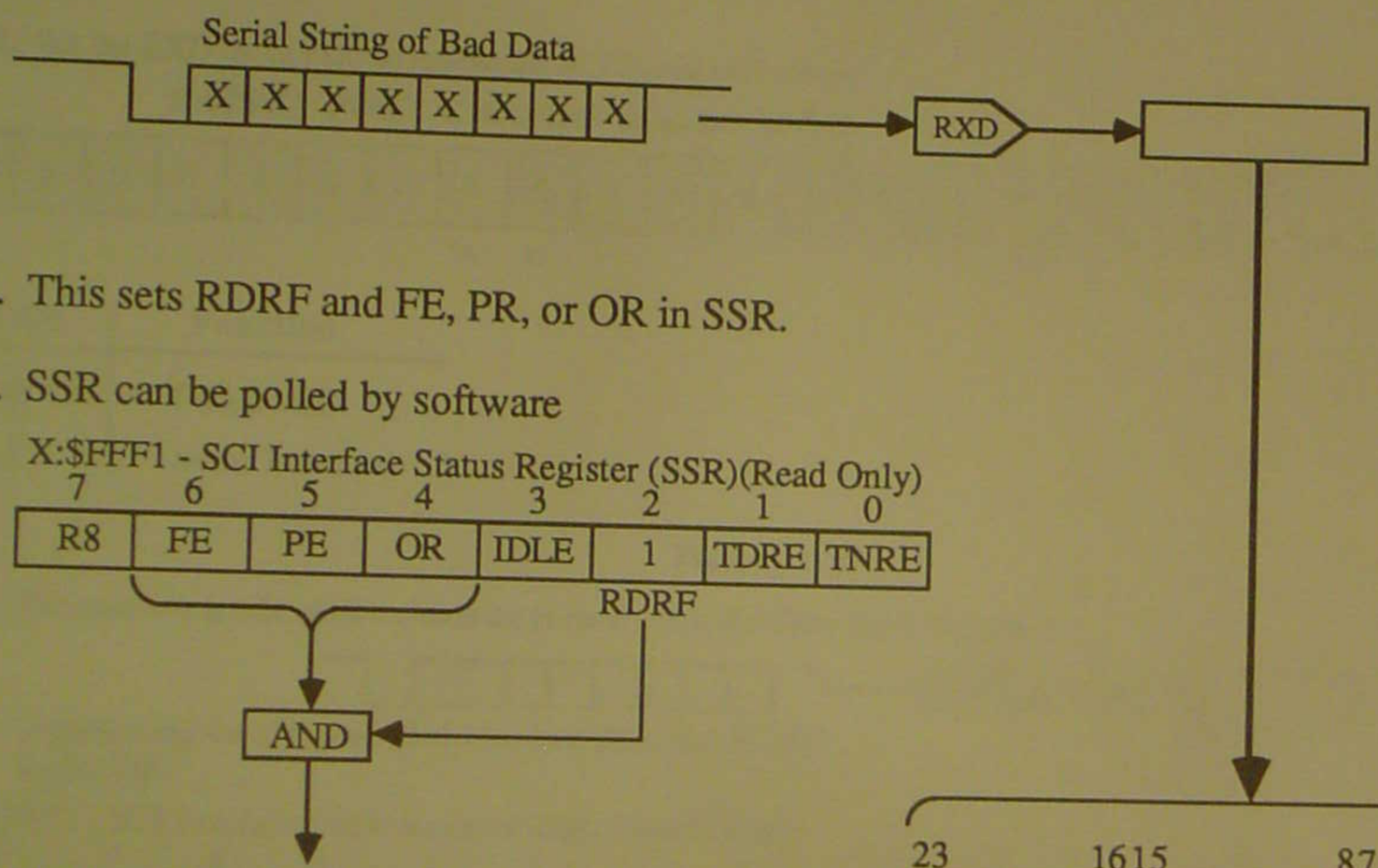
7. The Receive Interrupt Service Routine reads the received character.

MTA5- 7350

# SCI - RECEIVE A CHARACTER WITH EXCEPTION

1. A character is received with at least one of the following errors:

   - Framing Error (FE = Bit 6 in SSR)
   - Parity Error (PE = Bit 5 in SSR)
   - Overrun Error (OR = Bit 4 in SSR)



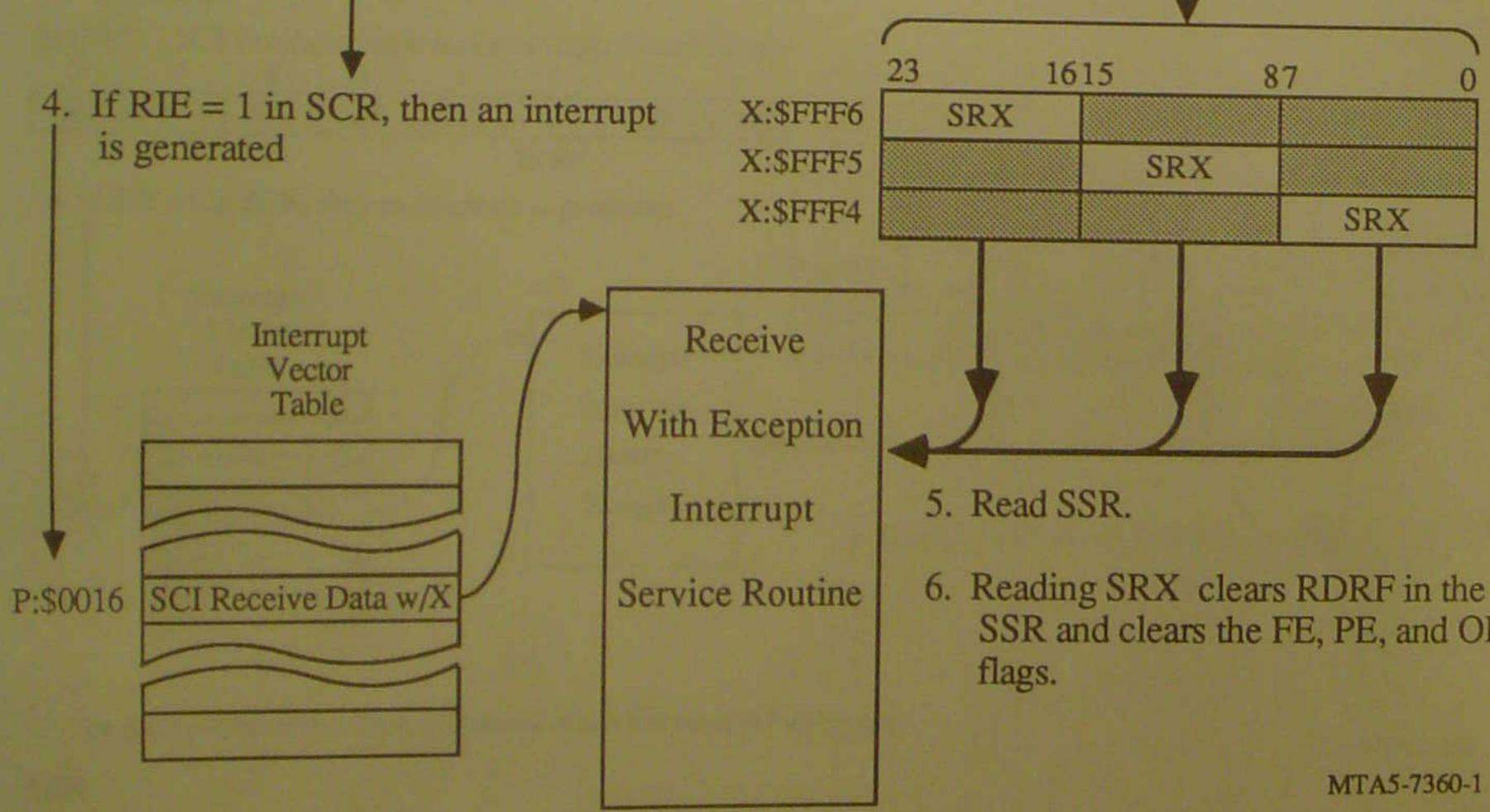

2. This sets RDRF and FE, PR, or OR in SSR.

3. SSR can be polled by software

X:$FFF1 - SCI Interface Status Register (SSR)(Read Only)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| R8 | FE | PE | OR | IDLE | 1 | TDRE | TNRE |

4. If RIE = 1 in SCR, then an interrupt is generated




5. Read SSR.

6. Reading SRX clears RDRF in the SSR and clears the FE, PE, and OR flags.

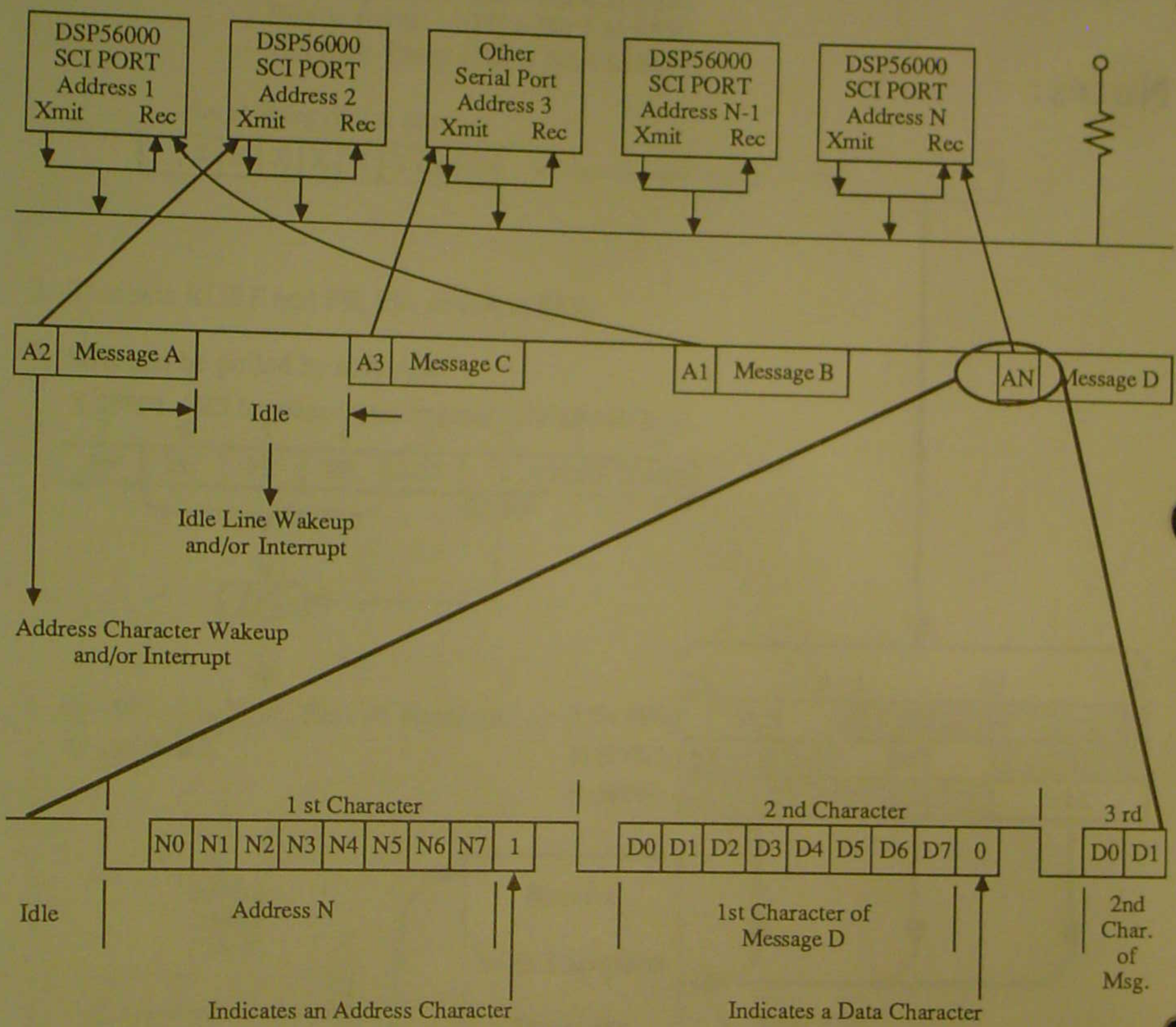
MTA5-7360-1

# MULTIDROP MODE

**X:$FFF0 - SCI Interface Control Register (SCR) (Read/Write)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | TMIE | TIE | RIE | ILIE | TE | RE | 1 | RWU | WAKE | SBK | 0 | 1 | 1 | 0 |
| | | | | | | | | | | | | | WDS2 | WDS1 | WDS0 |

Wire OR Mode Select - WOMS




MTA5-7370

# WAKE-UP MODES

**X:$FFF0 - SCI Interface Control Register (SCR) (Read/Write)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | TMIE | TIE | 1 | 1 | TE | RE | WOMS | 1 | WAKE | SBK | 0 | WDS2 | WDS1 | WDS0 |

RIE — Receive Interrupt Enable

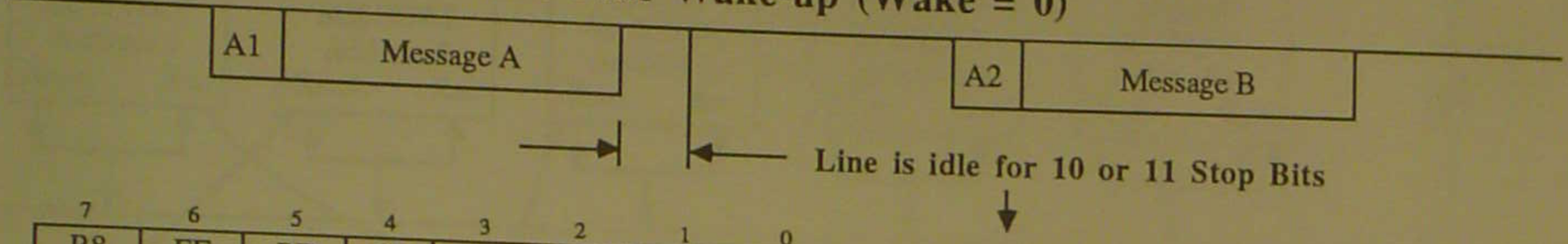
ILIE — Idle Line Interrupt Enable

RWU — Receiver Wake-up Enable

SBK — Send Break

WAKE: 1 = Address Wake-up; 0 = Idle Line Wake-up

## Idle Line Wake-up (Wake = 0)

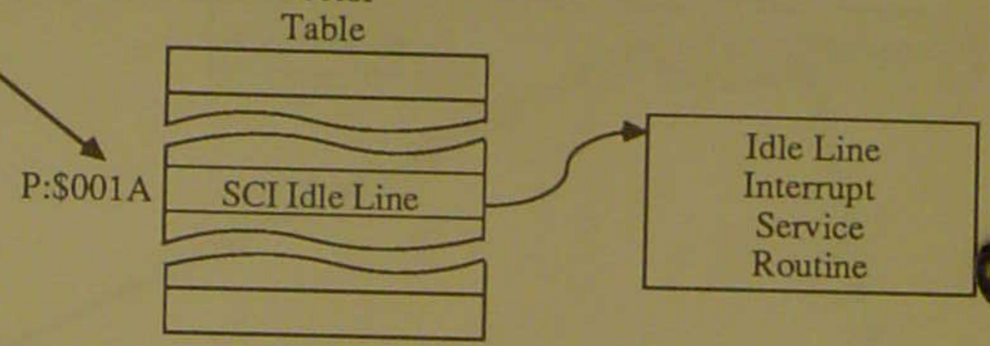



| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| R8 | FE | PE | OR | 1 | RDRF | TDRE | TNRE |

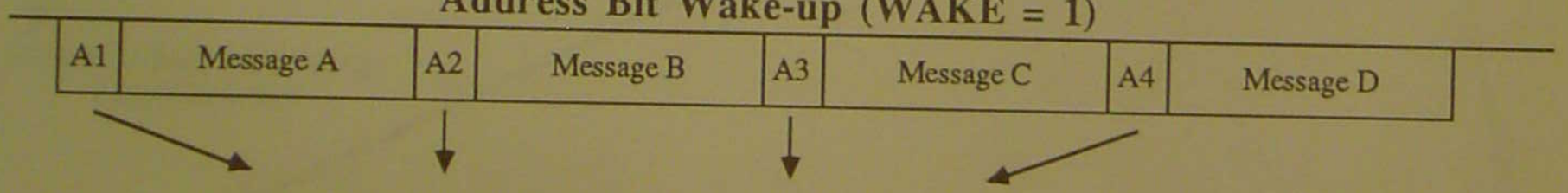
IDLE

1. When Line is idle, Idle = 1 in SSR and RWU is is Cleared. The Receiver "Wakes Up."

2. If ILIE = 1 in SCR, then an SCI Idle Line Interrupt is pending




## Address Bit Wake-up (WAKE = 1)



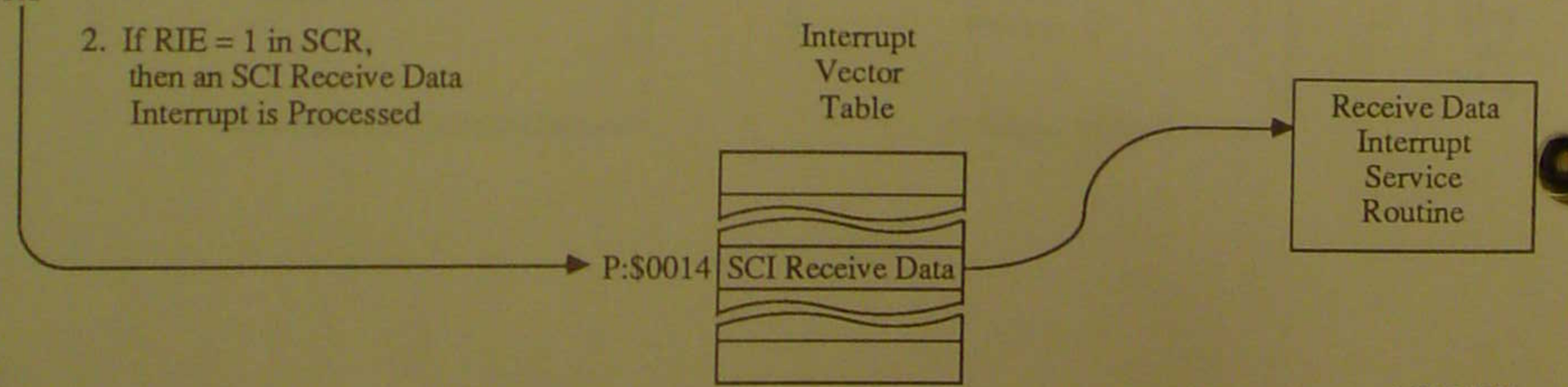

1. When Address Character is received, then R8 = 1 in SSR and RWU is cleared. The Receiver "Wakes Up".

X:$FFF1 - SCI Interface Status Register (SSR) (Read Only)

| 1 | FE | PE | OR | 1 | RDRF | TDRE | TDRE |
|---|---|---|---|---|---|---|---|

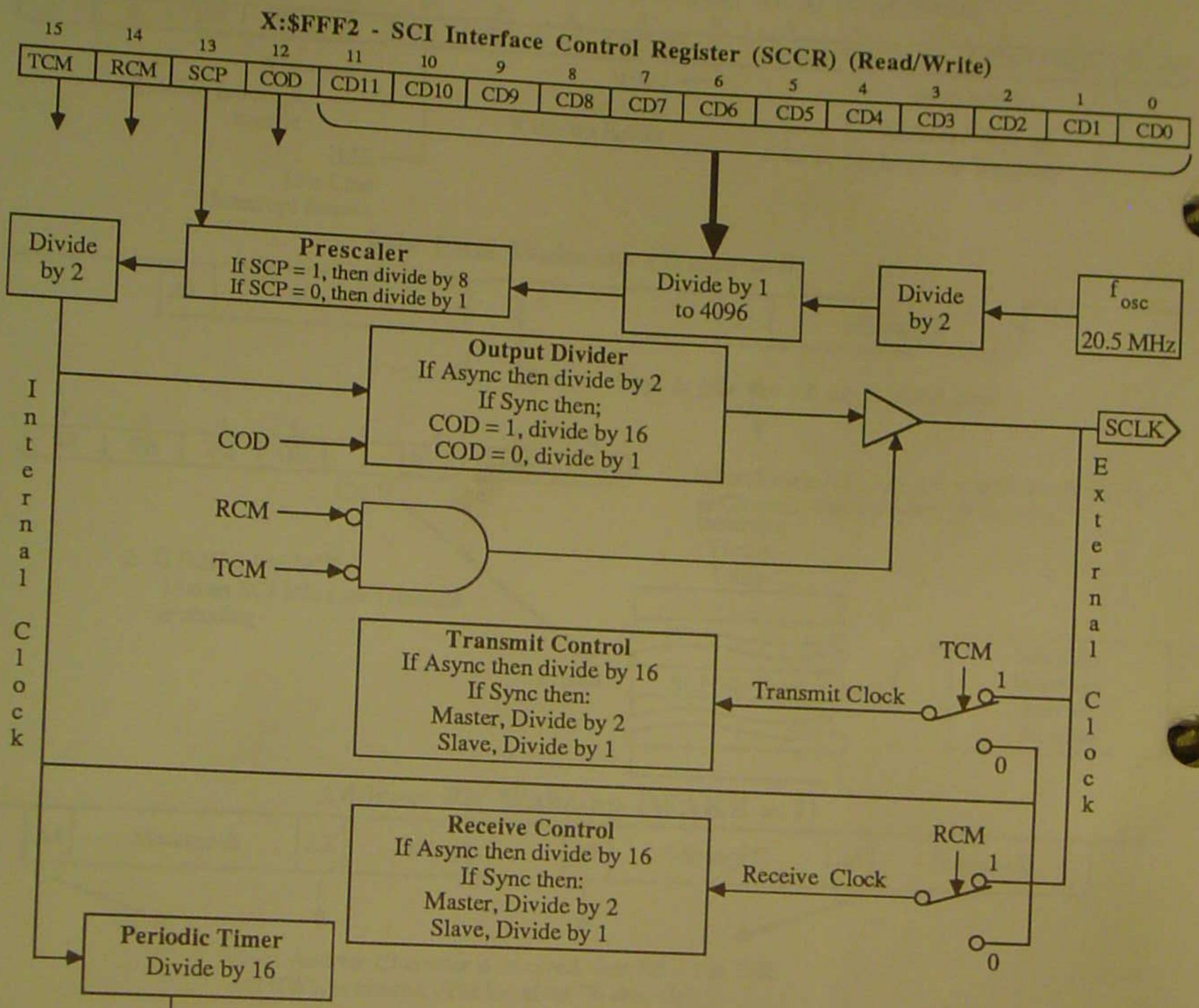
R8

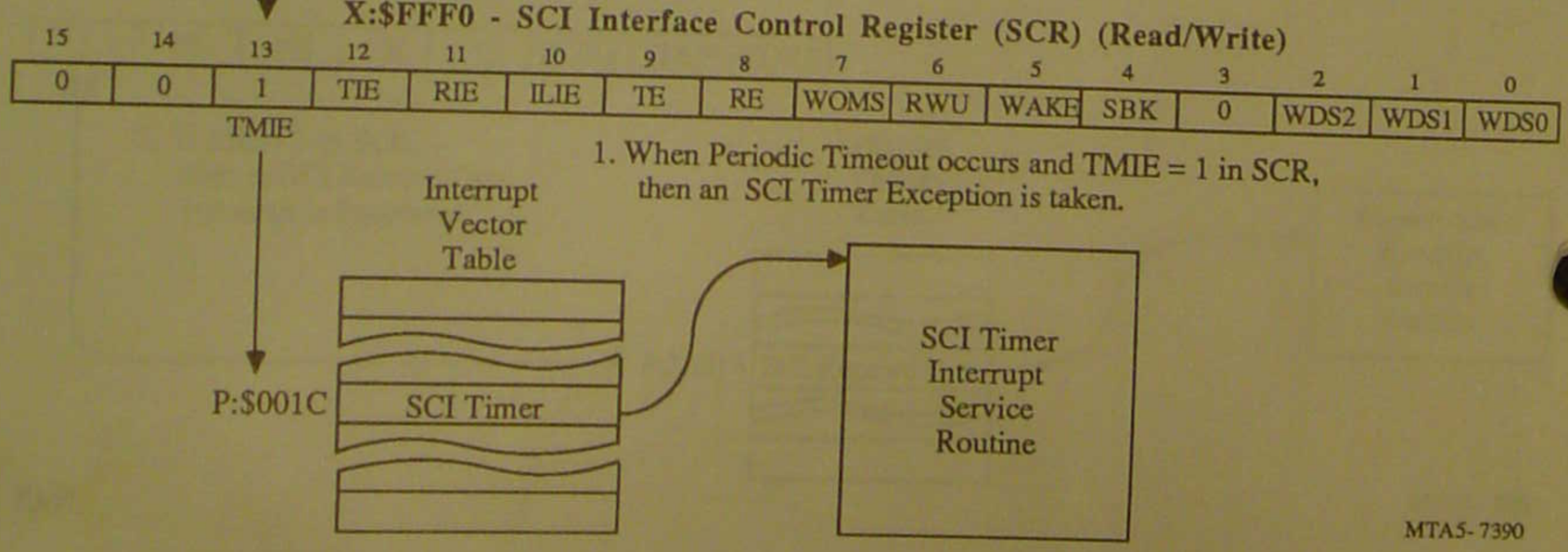
2. If RIE = 1 in SCR, then an SCI Receive Data Interrupt is Processed




MTA5- 7380

# SCI - CLOCK AND TIMER




MTA5-7390

# EXERCISE 15

## ON CHIP PERIPHERALS

1. The Serial Communications Interface supports _____ serial formats.

2. The Serial Communications Interface has _____ exception vectors.

3. To transmit data as a data byte, the character is stored in the _____ register.

4. To transmit data as an Address byte, the character is stored in the _____ register.

5. The three types of receive errors detected by the SCI are _____ _____, _____ _____, and _____ _____.

6. The two types of wake up modes are _____ _____, and _____ _____.

7. For a multidrop SCI configuration, all the transmit and receive lines may be connected to a bus, providing that the _____ _____ _____ _____ bit is set in the Interface Control Register.

MTA5-7395

# SYNCHRONOUS SERIAL INTERFACE FEATURES

- 5 Megabit per Second Serial Interface

- 6 Pin Interface

- Interface to a wide varety of Standard Serial Devices
  - CODECS
  - Time Division Multiplexed Networks
  - Serial Peripherals
  - DSP56000 to DSP56000 Networks
  - Motorola SPI Peripherals
  - Economical Shift Registers

- Clock - Continuous, Gated, Internal, External

- Programmable Word Length - 8, 12, 16, or 24 bits

- On Chip Serial Clock and Frame Sync Generator

- Double Buffered, User Programmable, Output and Input, Control and Status Bits.

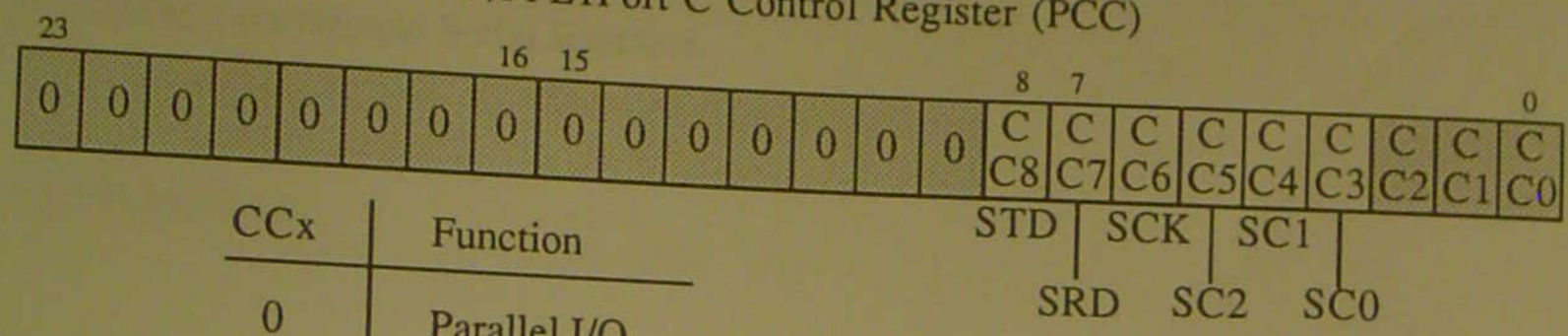
- Operating Modes - Normal, Network, on-Demand

- Four Interrupt Vectors
  - Receive
  - Receive with Exception
  - Transmit
  - Transmit with Exception

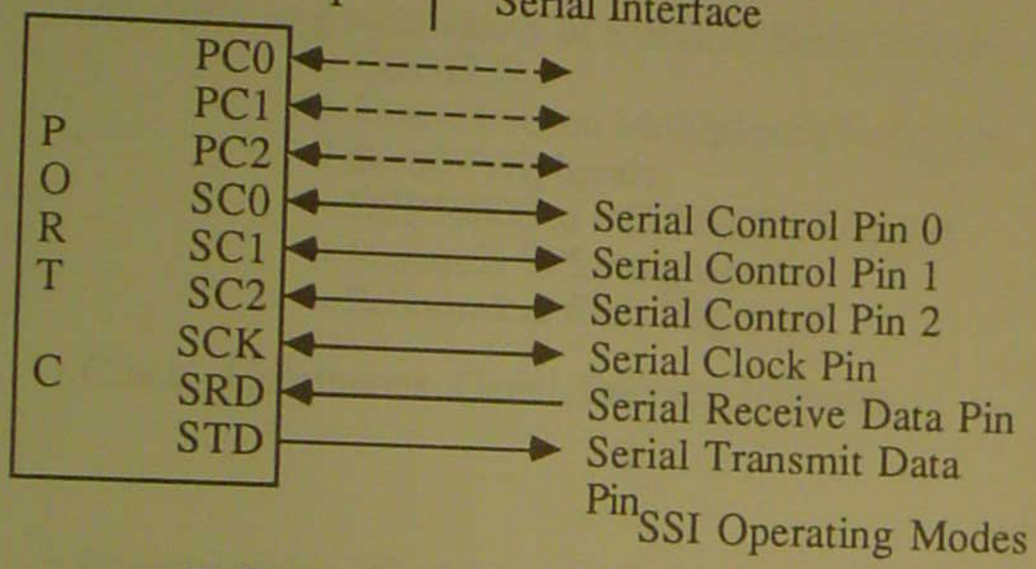
MTA5-7400-1

# SSI FUNCTIONAL DIAGRAM

X:$FFE1Port C Control Register (PCC)




| CCx | Function |
|---|---|
| 0 | Parallel I/O |
| 1 | Serial Interface |




## INITIALIZATION

1. Hardware or Software RESET
2. Program SSI Control Registers
3. Configure SSI PCC Bits as Serial Interface

## SSI Operating Modes

| Tx,Rx Sections | Serial Clock | Protocol | Typical Applications |
|---|---|---|---|
| Asynchronous | Continuous | Normal | single asynchronous codec stream-mode channel interface |
| Asynchronous | Gated | Normal | DSP-to DSP serial peripherals (A/D,D/A) |
| Synchronous | Continuous | Normal | multiple synchronous codecs |
| Synchronous | Gated | Normal | SPI-type devices DSP-to-MCU |
| Synchronous | Continuous | Network | TDM codec network TDM DSP networks |
| Asynchronous | Continuous | Network | TDM networks |
| Synchronous | Gated | On Demand | DSP to SPI peripherals |
| Asynchronous | Gated | On Demand | Parallel to serial and serial to parallel conversion |

MTA5-7410-1

# SSI - PROGRAMMING MODEL

X:$FFEC-SSI Control Register A (CRA)(Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSR | WL1 | WL0 | DC4 | DC3 | DC2 | DC1 | DC0 | PM7 | PM6 | PM5 | PM4 | PM3 | PM2 | PM1 | PM0 |

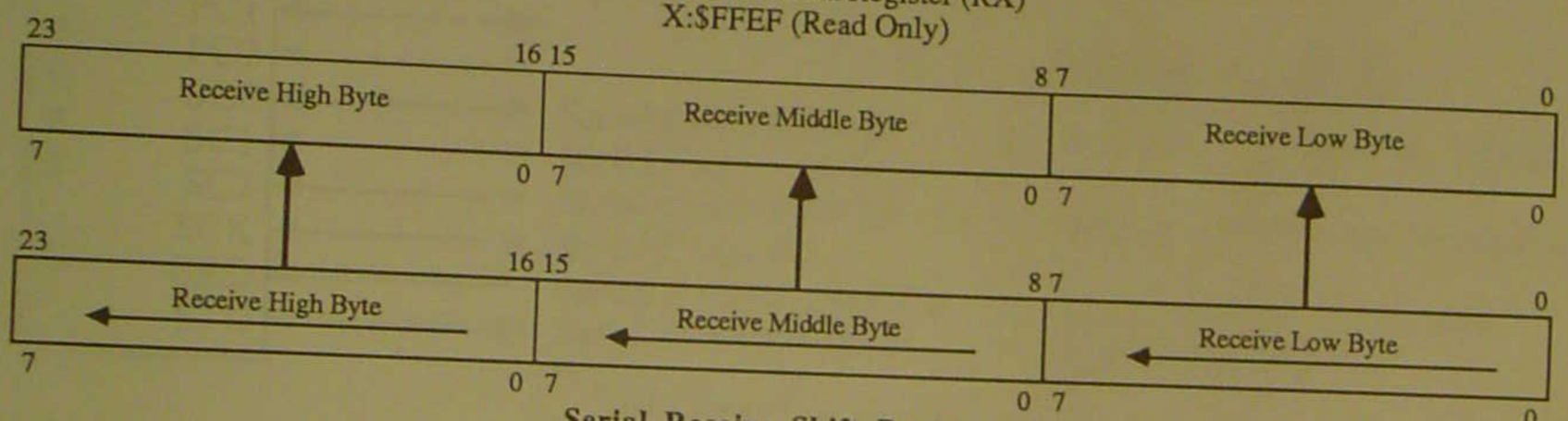
X:$FFED-SSI Control Register B (CRB)(Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | MOD | GCK | SYN | FSL | 0 | 0 | SCKD | SCD2 | SCD1 | SCD0 | OF1 | OF0 |

X:$FFEE - SSI Status Register (SR)(Read)
X:$FFEE - SSI Time Slot Register (TSR)(Write)

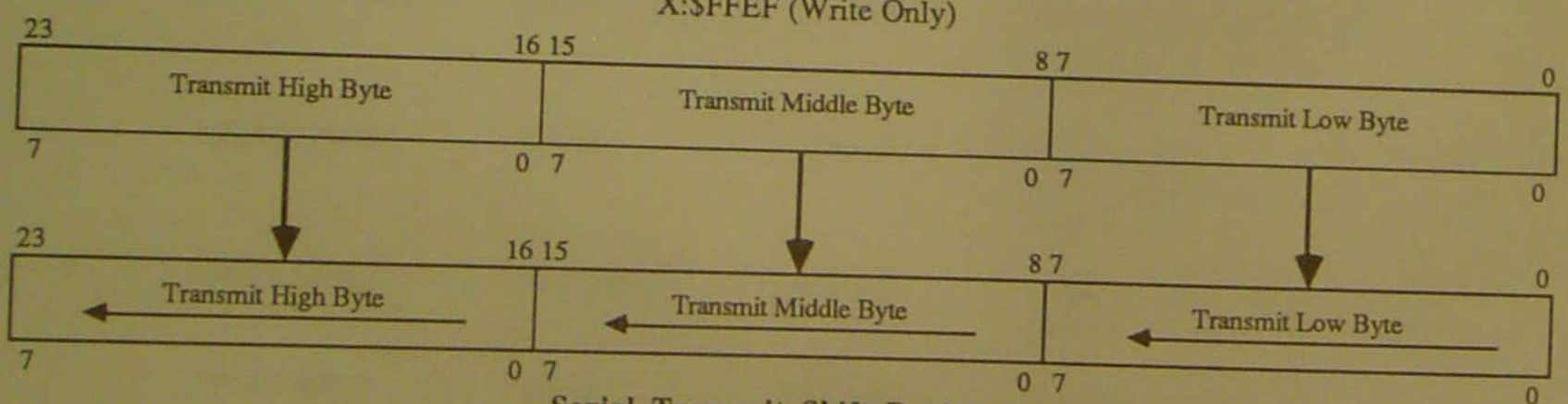
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RDF | TDE | ROE | TUE | RFS | TFS | IF1 | IF0 |
| * | * | * | * | * | * | * | * |

Serial Receive Data Register (RX)
X:$FFEF (Read Only)

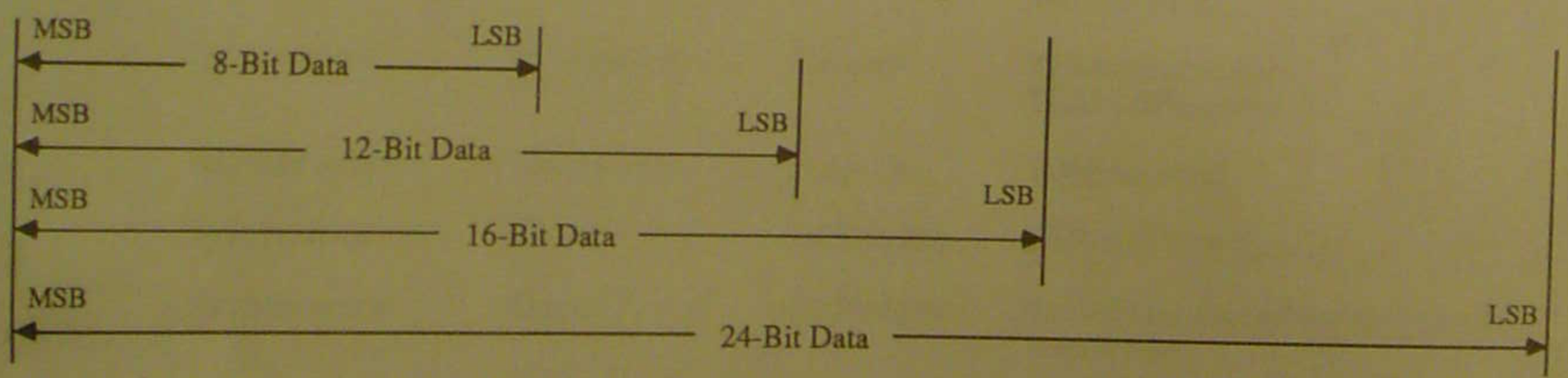



**Serial Receive Shift Register**

Serial Transmit Data Register (TX)
X:$FFEF (Write Only)

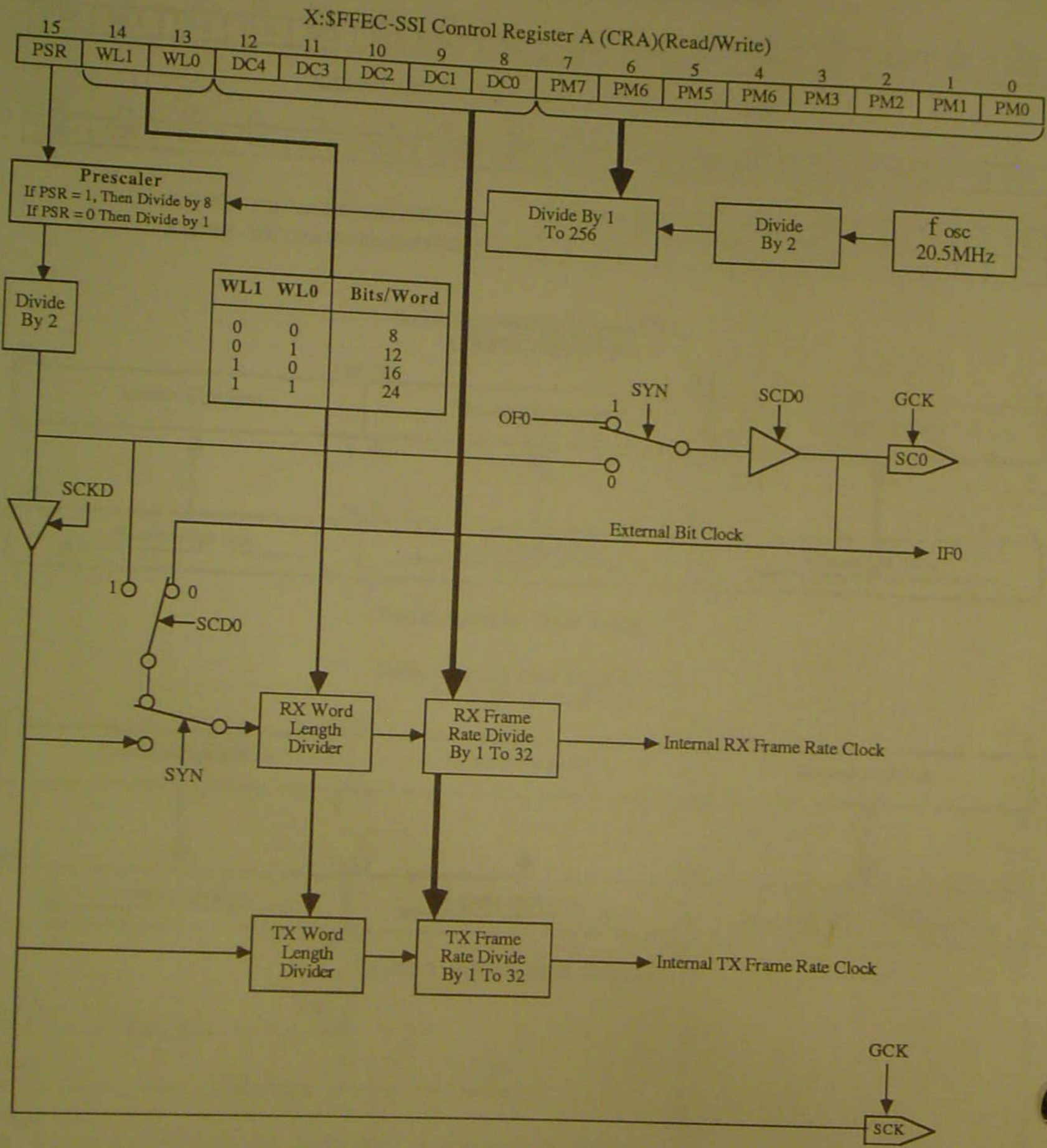



**Serial Transmit Shift Register**




MTA5- 7420

# SSI - CLOCK

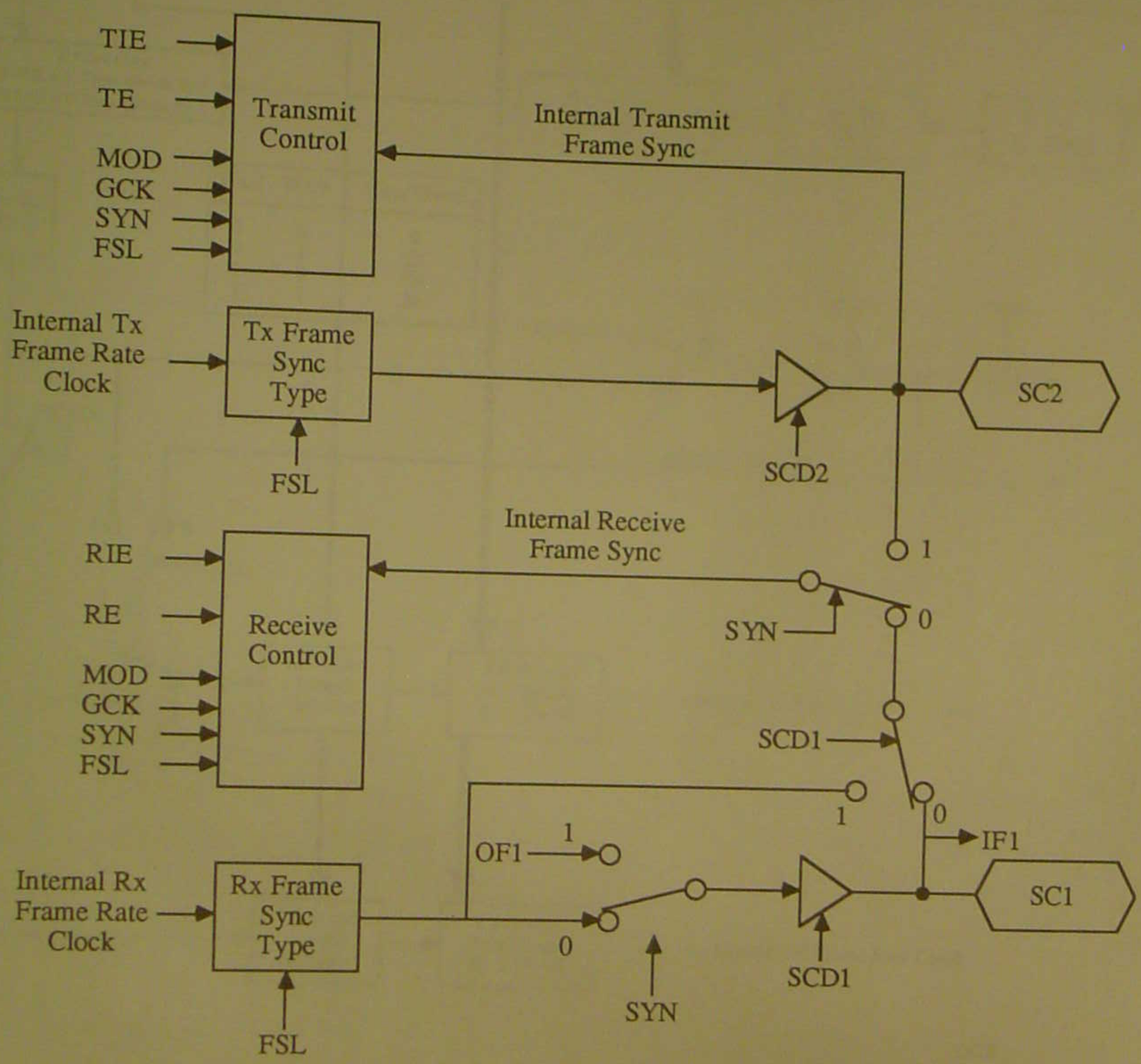



MTA5- 7430

# SSI-FRAME SYNC GENERATOR

X:$FFED-SSI Control Register B (CRB) (Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | MOD | GCK | SYN | FSL | 0 | 0 | SCKD | SCD2 | SCD1 | SCD0 | OF1 | OF0 |




MTA5-7440

# SSI - CONTROL PIN FUNCTIONS

| SSI Pin Name | Asynchronous (SYN = 0) | | | | Synchronous (SYN = 1) | | | |
|---|---|---|---|---|---|---|---|---|
| | Continuous Clock (GCK = 0) | | Gated Clock (GCK = 1) | | Continuous Clock (GCK = 0) | | Gated Clock (GCK = 1) | |
| | Input ( = 0) | Output ( = 1) | Input ( = 0) | Output ( = 1) | Input ( = 0) | Output ( = 1) | Input ( = 0) | Output ( = 1) |
| **SC0** (SCD0) | RX Clock External | RX Clock Internal | RX Clock External | RX Clock Internal | Input Flag IF0 | Output Flag OF0 | Input Flag IF0 | Output Flag OF0 |
| **SC1** (SCD1) | RX Frame Sync External | RX Frame Sync Internal | NOT USED | RX Frame Sync Internal | Input Flag IF1 | Output Flag OF1 | Input Flag IF1 | Output Flag OF1 |
| **SC2** (SCD2) | TX Frame Sync External | TX Frame Sync Internal | NOT USED | TX Frame Sync Internal | TX and RX Frame Sync External | TX and RX Frame Sync Internal | NOT USED | TX and RX Frame Sync Internal |
| **SCK** (SCKD) | TX Clock External | TX Clock Internal | TX Clock External | TX Clock Internal | TX and RX Clock External | TX and RX Clock Internal | TX and RX Clock External | TX and RX Clock Internal |

(CONTROL BIT NAME)

X:$FFED - SSI Control Register B (CRB)(Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | MOD | GCK | SYN | FSL | 0 | 0 | SCKD | SCD2 | SCD1 | SCD0 | OF1 | OF0 |

Gated Clock Control

Synchronous/ Asynchronous Control

Clock Source Direction

Serial Control Direction

Output Flags

1 = OUTPUT
0 = INPUT

X:$FFEE - SSI Status Register (SR)(Read)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RDF | TDE | ROE | TUE | RFS | TFS | IF1 | IF0 |

Input Flags

MTA5-7445

# SSI - EXCEPTIONS

X:$FFED - SSI Control Register (CRB) (Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | MOD | GCK | SYN | FSL |

SSI Exception Mask

| Exception Starting Address | Exception Vector Table |
|---|---|
| $0000 | |
| $000C | SSI Receive Data |
| $000E | SSI Receive Data with Exception Status |
| $0010 | SSI Transmit Data |
| $0012 | SSI Transmit Data with Exception Status |

X:$FFEE - SSI Status Register (SR) (Read)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RDF | TDE | ROE | TUE | RFS | TFS | IF1 | IF0 |

SSI Status Bits

**Receive Interrupt Service Routine**

1. Interrupt is generated when: RIE = 1 and RDF = 1 and ROE = 0.
2. Pending Interrupt is cleared by reading RX.

**Receive with Exception Status Interrupt Service Routine**

1. Interrupt is generated when: RIE = 1 and RDF = 1 and ROE =1.
2. ROE is cleared by reading SR, followed by:
3. Reading RX to clear Pending Interrupt.

**Transmit Interrupt Service Routine**

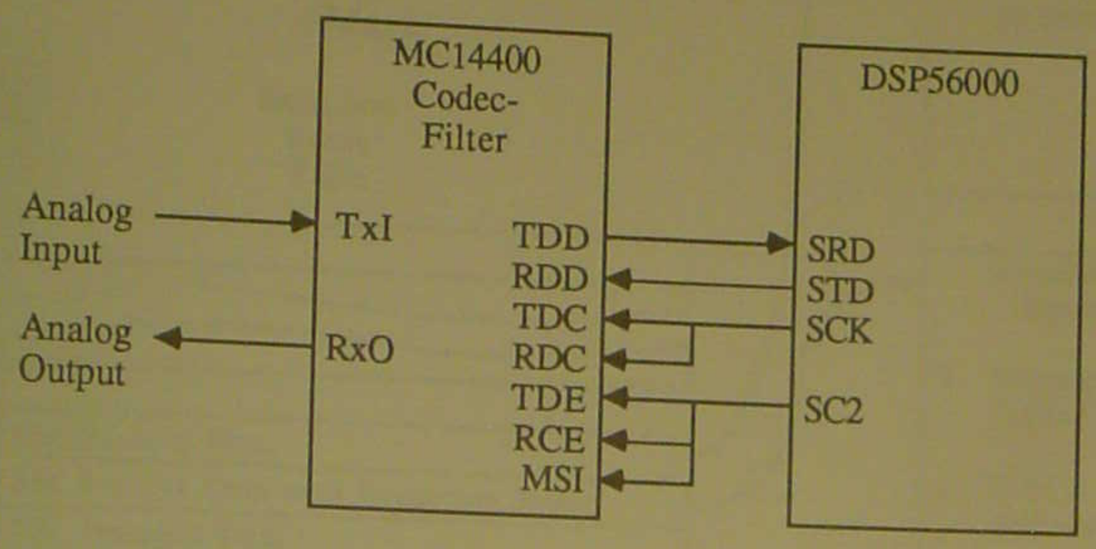
1. Interrupt is generated when: TIE = 1 and TDE = 1 and TUE = 0.
2. Pending Interrupt is cleared by writing to TX or TSR.

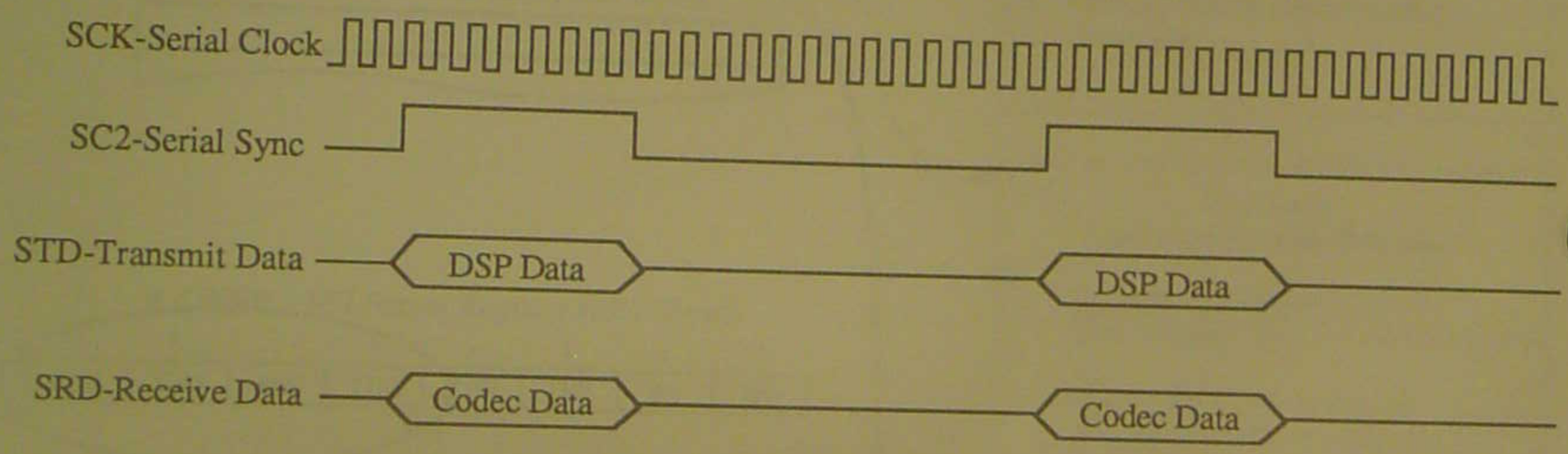
**Transmit with Exception Status Interrupt Service Routine**

1. Interrupt is generated when: TIE = 1 and TDE = 1 and TUE = 1.
2. TUE is cleared by reading SR, followed by:
3. Writing to TX or TSR to clear Pending Interrupt.

MTA5- 7450-1

# SSI CODEC EXAMPLE




X:$FFED-SSI Control Register B (CRB) (Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | SCD1 | SCD0 | OF1 | OF0 |

MOD
SSI Mode Select
0=NORMAL

GCK
Gated Clock Control
0=CONTINUOUS CLOCK

SYN
Sync/Async Control
1=SYNCHRONOUS

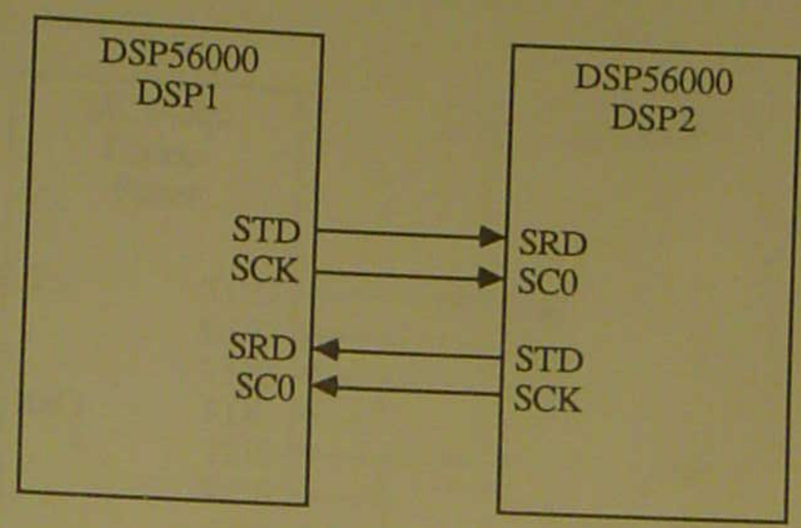
FSL
Frame Sync Length
0=WORD CLOCK

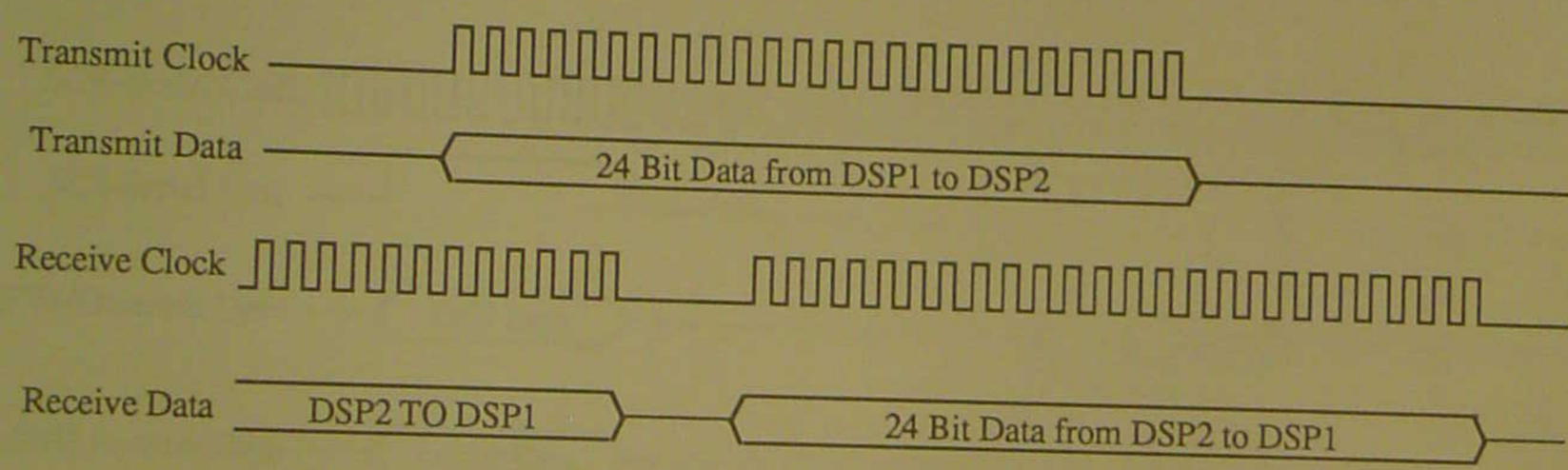
SCKD
Clock Source Direction
1=OUTPUT

SCD2
Serial Control 2 Direction
1=OUTPUT

MTA5-7460

# SSI-DSP56000 TO DSP56000 INTERFACE EXAMPLE




X:$FFEC-SSI Control Register A (CRA) (Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSR | 1 | 1 | 0 | 0 | 0 | 0 | 0 | PM7 | PM6 | PM5 | PM6 | PM3 | PM2 | PM1 | PM0 |

WL1 WL0 DC4 DC3 DC2 DC1 DC0 – On Demand

24 Bit Word Length

X:$FFED-SSI Control Register B (CRB) (Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | 0 | 1 | 0 | FSL | 0 | 0 | 1 | SCD2 | SCD1 | 0 | OF1 | OF0 |

MOD
SSI Mode Select
1=NETWORK

GCK
Gated Clock Control
1=GATED CLOCK

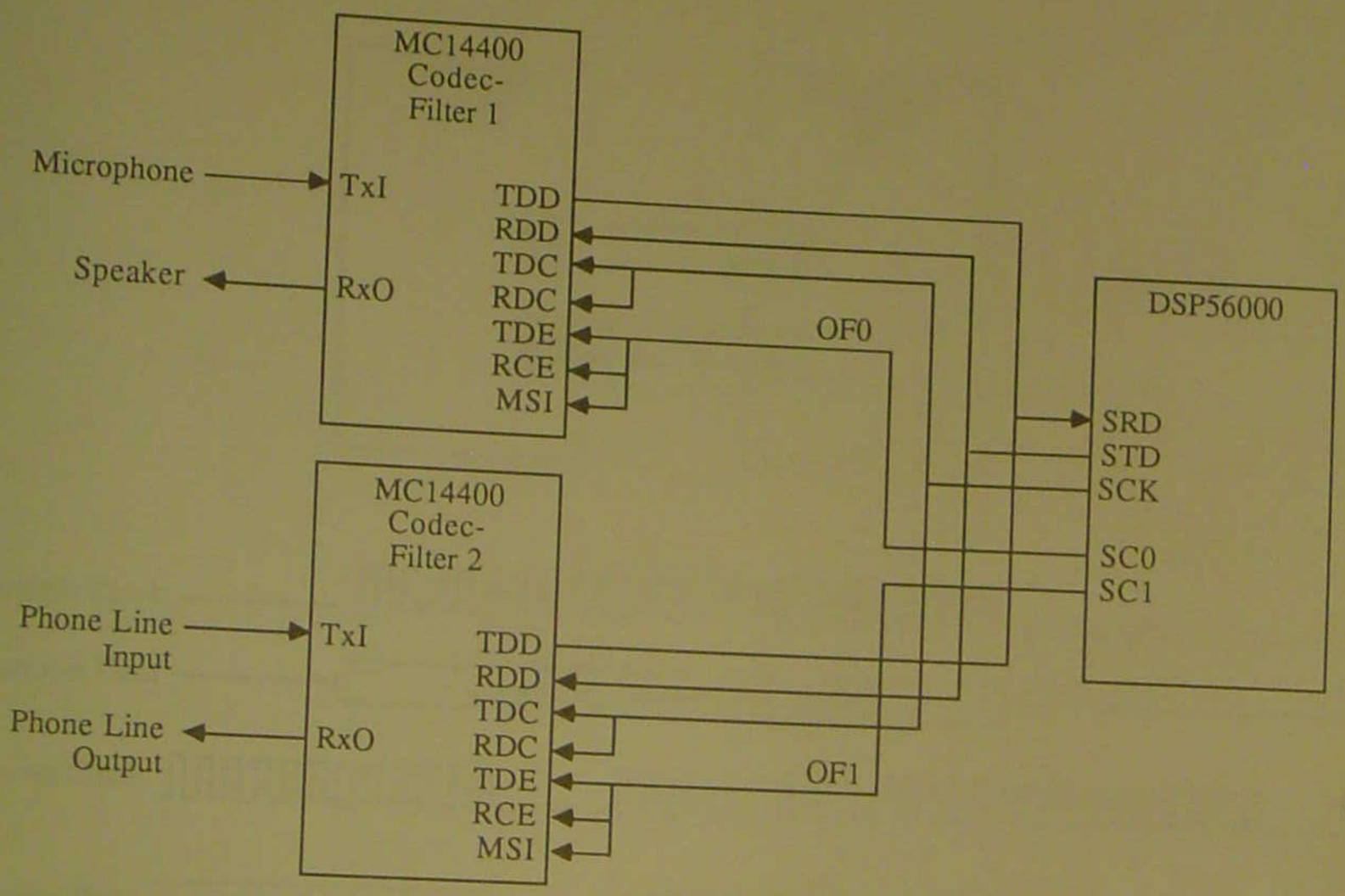
SYN
Sync/Async Control
0=ASYNCHRONOUS

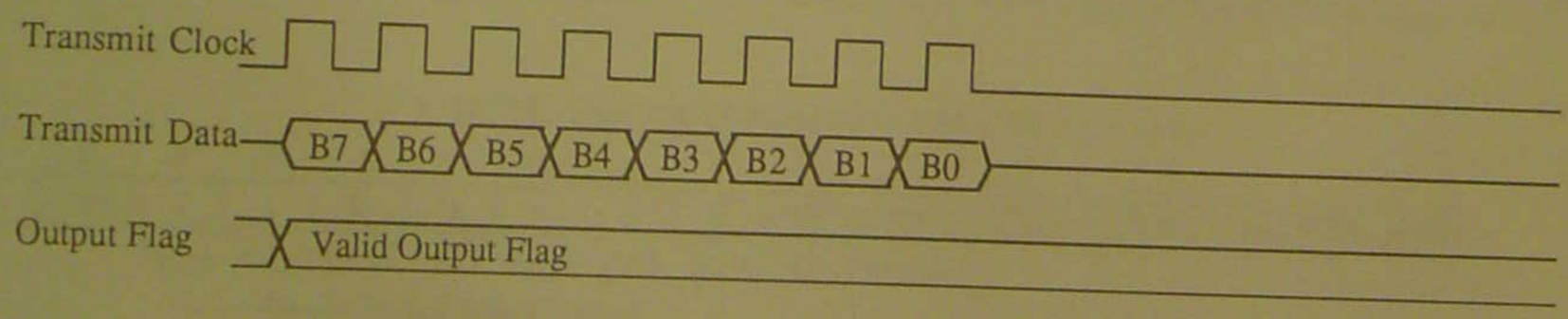
SCKD
Clock Source Direction
1=OUTPUT

SCD0
Serial Control 2 Direction
0=INPUT

MTA5-7470

# SSI SPEAKER PHONE EXAMPLE




| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | 0 | 1 | 1 | FSL | 0 | 0 | 1 | SCD2 | 1 | 1 | OF1 | OF0 |

MOD — SSI Mode Select 0=NORMAL

GCK — Gated Clock Control 1=GATED CLOCK

SYN — Sync/Async Control 1=SYNCHRONOUS

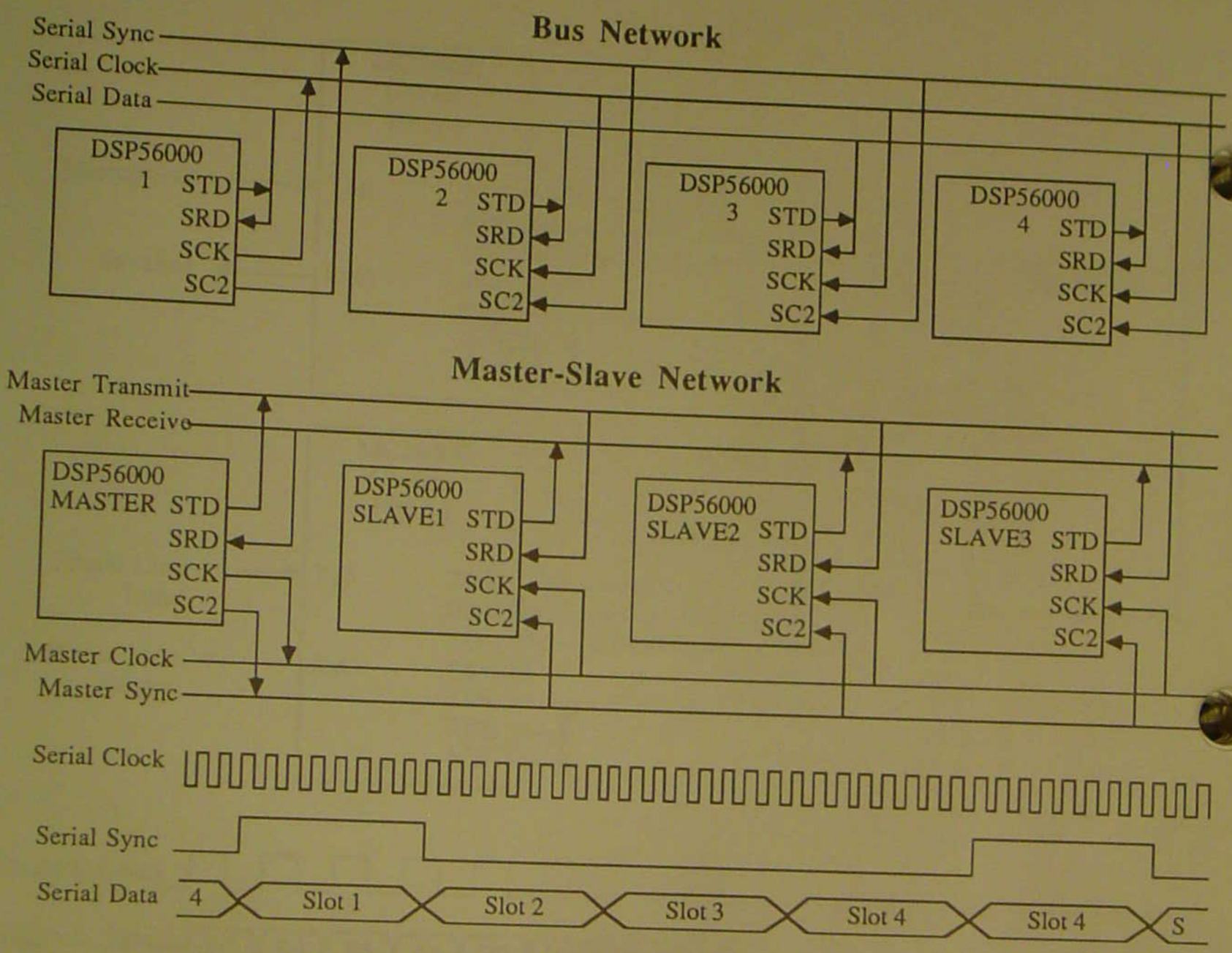
SCKD — Clock Source Direction 1=OUTPUT

SCD1 and SCD0 — Serial Control 1 and 0 Direction 1=OUTPUT

OF1: 0 = 1 = FILTER 1; 1 = 0 = FILTER 2

MTA5-7480-1

# SSI TIME DIVISION MULTIPLIED NETWORKING EXAMPLE




X: $FFED - SSI Control Register B (CRB) (Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | 1 | 0 | 1 | 0 | 0 | 0 | SCKD | SCD2 | SCD1 | SCD0 | OF1 | OF0 |

MOD — SSI Mode Select 1=NETWORK

GCK — Gated Clock Control 0=CONTINUOUS CLOCK

SYN — Sync/Async Control 1=SYNCHRONOUS

FSL — Frame Sync Length 0=WORD WIDTH

SCKD — Clock Source Direction 1=OUTPUT (MASTER) 0=INPUT (SLAVE)

SCD2 — Serial Control 2 Direction 1=OUTPUT (MASTER) 0=INPUT (SLAVE)

MTA5-7490-2

# EXERCISE 16

## ON CHIP PERIPHERALS

1. The synchronous serial interface uses _____ pins of port C.

2. The S.S.I. has a programmable word length and can transmit _____, _____, and _____ bit data.

3. The synchronous serial interface has _____ exception vectors.

4. The data in the S.S.I. transmit and receive registers are _____ justified and are broadcast with the _____ significant bit first.

5. Program control register B to support the following configuration:
   Asynchronous mode
   Gated clock
   SC0 pin is an external receive clock input
   SC1 pin is an internal receive frame sync
   SC2 pin is an internal transmit frame sync
   SCK pin is an internal transmit clock output
   Normal Mode
   Word Wide Frame Sync Length

X:$FFED-SSI Control Register B (CRB)(Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | | | | | 0 | 0 | | | | | OF1 | OF0 |
| | | | | MOD | GCK | SYN | FSL | | | SCKD | SCD2 | SCD1 | SCD0 | | |

6. Program control register B to support the following configuration:
   Synchronous mode
   Continuous clock
   SC0 pin is Output Flag Zero
   SC1 pin is Output Flag One
   SC2 pin is an internal transmit frame sync
   SCK pin is an external transmit and receive clock input
   Network mode
   Bit wide frame sync length
   Output Flag Zero is set
   Output Flag One is cleared

X:$FFED-SSI Control Register B (CRB)(Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | | | | | 0 | 0 | | | | | | |
| | | | | MOD | GCK | SYN | FSL | | | SCKD | SCD2 | SCD1 | SCD0 | OF1 | OF0 |

MTA5-7495-1

# RESOURCE COMPARISON

| Resource | Motorola DSP56000/1 | T.I. 32010 | T.I. 32020 | T.I. 320C25 | NEC 7720 | NEC 77230 | AT&T WE DSP32 | Analog Devices ADSP- 2100 |
|---|---|---|---|---|---|---|---|---|
| Instructions Rate (nSec) | 97.5 | 200 | 200 | 100 | 250 | 150 | 250 | 125 |
| Data Size (Bits) | 24 | 16 | 16 | 16 | 16 | 32 Flt Pt (24,8) | 32 Flt. Pt. (24,8) | 16 |
| Program Size (Bits) | 24 | 16 | 16 | 16 | 23 | 32 | 32 | 24 |
| Data RAM Size | 2x256 | 144 | 544 | 544 | 128 | 2x512 | 2x512 | NONE |
| Data ROM Size | 2x256 | NONE | NONE | NONE | 510 | 1024 | 512 Data & Prog. | NONE |
| Program Memory Size | 3840 ROM or 512 RAM | 1536 ROM | 256 OF 544 RAM can be Program | 4096 ROM | 512 ROM | 2048 ROM | Not Harvard | 16 RAM Cache |
| Program Stack Depth | 2x15 | 4 | 4 | 8 | 4 | 8 | NONE | 16 |
| Multiplier Size | 24x24 | 16x16 | 16x16 | 16x16 | 16x16 | 24x24 Flt. Pt. | 24x24 | 16x16 |
| Multiplier Input Registers | 4 | 1 | 1 | 1 | 2 | 2 | NONE | 5 |
| ALU Size (Bits) | 56 | 32 | 32 | 32 | 16 | 47 | 40 | 16 |
| Accumulator Registers | 2x56 | 1x32 | 1x32 | 1X32 | 2x16 | 8x55 | 4x40 | 1x40 |
| Address Registers | 24 | 2 | 5 | 8 | 2 | 5 | 21 | 24 |
| Memory Expansion | 192K | 4K | 128K | 128K | NONE | 4K Prog. Master 4K Data Master 8K Data Slave | 16K | 48K |
| Interrupt Sources | 18 | 1 | 7 | 7 | 1 | 2 | NONE | 4 |
| On-Chip Peripherals | 2 Serial 1 Parallel | NONE | 1 Serial | 1 Serial | 1 Serial 1 Parallel | 2 Serial 1 Parllel | 1 Serial 1 Parallel | NONE |
| Peripheral Registers | 32 | NONE | 6 | 6 | 5 | 5 | 9 | NONE |
| I/O Flags | 24 | 1 | 1 | 1 | 2 | 4 | NONE | NONE |
| Pins | 88 | 40 | 68 | 68 | 28 | 68 | 40/100 | 100 |
| Technology | 1.5u CMOS | 2.7u NMOS | 2.4u NMOS | 1.8u CMOS | 3u NMOS | 1.75u CMOS | 1.5u NMOS | 1.5u CMOS |
| Power Dissipation (Max) | 0.5W Typical at 20.5 MHz | 1.4W | 1.8W | 1 | 1.75W | 1.5W | 2.7W | 0.6W |

MTA5- 9000-1

# BENCHMARKS

| Benchmark | | Motorola DSP56000/1 | T.I. 32010 | T.I. 32020 | T.I. 320C25 | NEC 7720 | NEC 77230 | AT&T WE DSP32 | Analog Devices ADSP-2100 |
|---|---|---|---|---|---|---|---|---|---|
| Real FIR Filter with Data Shift | (L) | 0.1 uSec/Tap | 0.8uSec/Tap(6) | (14) | (15) | (11) | 0.6 uSec/Tap(7) | 0.5 uSec/Tap(3) | 0.125 uSec/Tap |
| | (S) | | 0.4uSec/Tap(6) | 0.2 uSec/Tap | 0.1 uSec/Tap(1) | 0.75 uSec/Tap | 0.15 uSec/Tap | 0.25 uSec/Tap | (7) |
| IIR Biquad Filter | (L) | 0.4 uSec | (13) | (14) | (15) | (11) | (7) | 1.25 uSec | 0.875 uSec |
| | (S) | | 2.0 uSec | 2.0 uSec | 1.0 uSec | 2.25 uSec | 0.9 uSec | (3) | (7) |
| Complex FFT | | | | | | | | | |
| 32 Point | (L) | 68.1 uSec | | | | | | | |
| | (S) | | | | | (11) | (7) | | |
| 256 Point | (L) | 713 uSec | | | | 700 uSec | 150 uSec | | |
| | (S) | | (4) | (5) | 3500 uSec(15) | | | | |
| 512 Point | (L) | 2039 uSec | 4100 uSec | 2440 uSec | 1500 uSec | | | 4300 uSec | |
| | (S) | | | | | | | (3) | |
| 1024 Point | (L) | 3391 uSec | | | | (2), (7) | (7) | | |
| | (S) | | (4) | (5) | | (2), (7) | 4700 uSec (8) | 19200 uSec | 7200 uSec |
| | | | 31000 uSec | 14180 uSec | 7100 uSec(2) | 77000 uSec | 12300 uSec | (3) | (7) |
| Matrix Multiplication | (L) | 1.7 uSec | (6) | (6) | (15) | | | | |
| [1x3] [3x3] | (L) | | 5.4 uSec | 5.4 uSec | 2.7 uSec | | | | |
| [10x10] [10x10] | (S) | 172 uSec | | | | | | | 220 uSec |
| LMS Adaptive | (L) | 0.3 uSec/Tap | | (9), (12) | 0.6 uSec/Tap | | | | |
| Real FIR Filter | (S) | | | 1.2 uSec/Tap | 0.4 uSec/Tap | | | | |
| Division | (L) | 2.7 uSec | | | | (7) | (7) | | |
| | (S) | | | | | 26.0 uSec | 4.8 uSec | | |
| Find Maximum Value | (L) | 0.2 usec | | | | | | 1.25 uSec | |
| and Index in Block | (S) | Per Point | | | | | | 1.0 uSec | |
| | | | | | | | | Per Point(10) | |
| Two Dimentional | (L) | 1.0 uSec | | | | | | | 1.25 uSec |
| Convolution (3x3 Mask) | | Per Output | | | | | | | Per Output (7) |
| Complex FIR Filter with Data Shift | | 0.4 uSec/Tap | | | | | | | 0.5 uSec/Tap (7) |

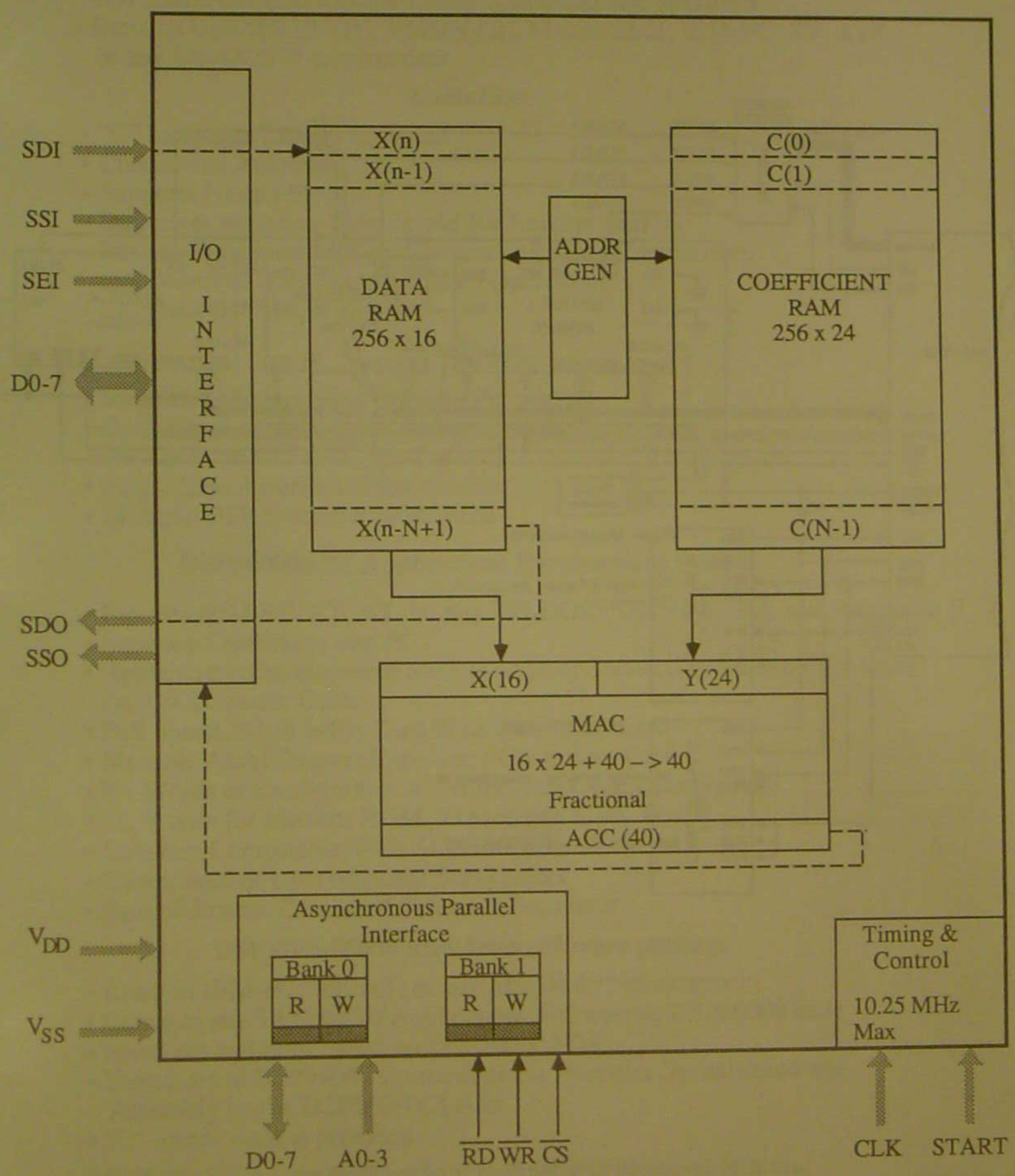
**Notes:**

(L) Looped Code
(S) Straight Line Coded
(1) Internal Data Only
(2) Not Possible Due to Lack of Mem. Expansion

**References:**

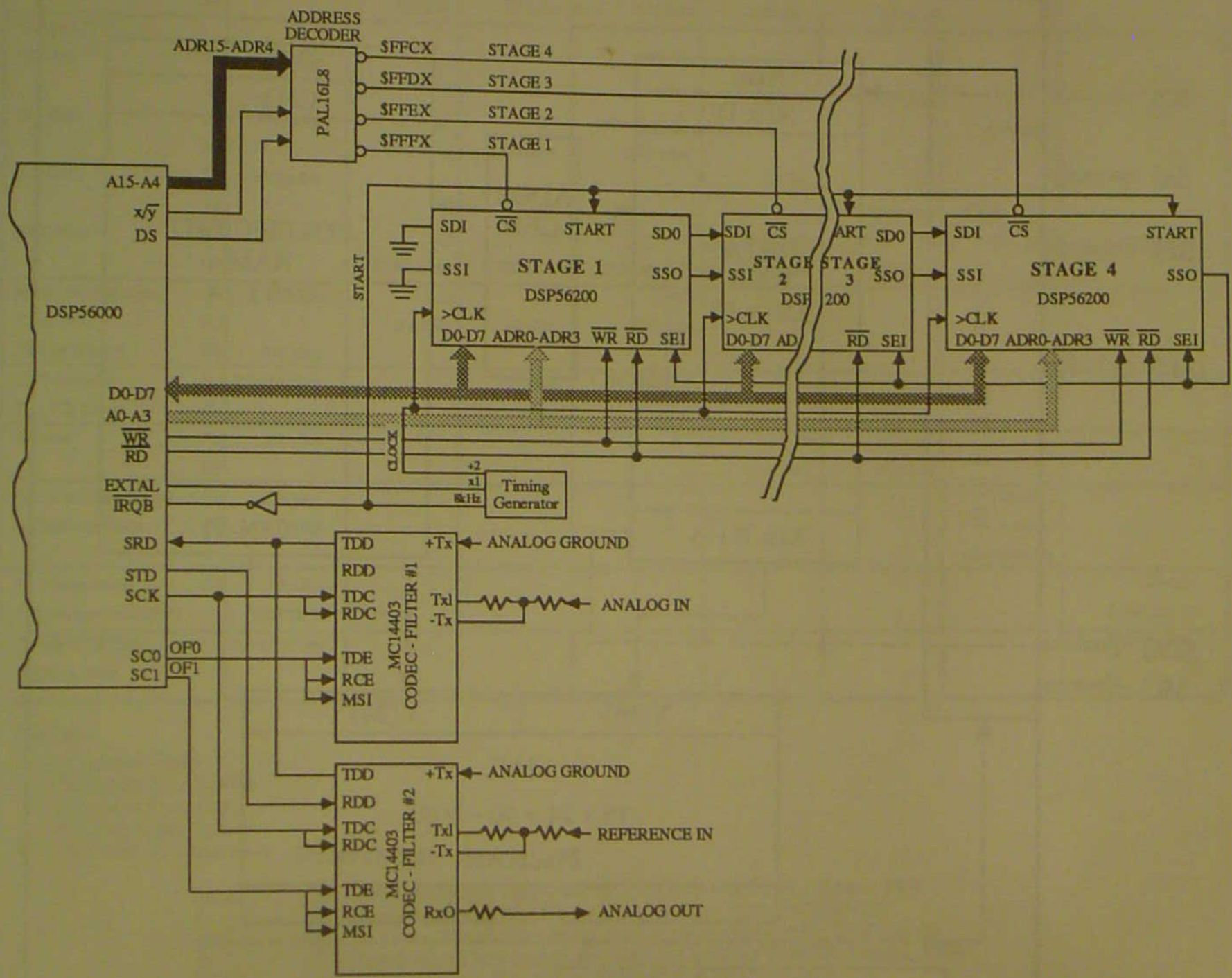

(3) 1986 Mini/Micro Northeast (ELECTRO) Conference Sesson 18/4
(4) Details on Signal Processing, Issue 5, June 1985, TI SPRN005
(5) Details on Signal Processing, Issue 6, November 1985, TISPRN006
(6) Digital Signal Processing Applications with the TMS320 Family, 1986
(7) Electric Design, Feb. 20, 1986 Pg. 133 & 160
(8) Electronic Design, June 12, 1986, Pg. 142
(9) Electronic Magazine Article Feb, 3, 1986, Pg. 23
(10) ICASSP86 Proceedings, Session 44.3
(11) NEC uPD7720 Signal Processing Interface (SPI) User's Manual
(12) TI FYI Newsletter, Vol.3, #2, Feb. 1986
(13) TMS320(10) Product Description, TISPRV001
(14) TMS32020 Product Description, TISPRV006
(15) TMS320C25 Product Description, TI SPRV006


MTA5-9010


SDI
SSI
SEI
D0-7
SDO
SSO
$V_{DD}$
$V_{SS}$
I/O INTERFACE
X(n)
X(n-1)
DATA RAM 256 x 16
X(n-N+1)
ADDR GEN
C(0)
C(1)
COEFFICIENT RAM 256 x 24
C(N-1)
X(16)
Y(24)
MAC
16 x 24 + 40 –> 40
Fractional
ACC (40)
Asynchronous Parallel Interface
Bank 0
R
W
Bank 1
R
W
Timing & Control
10.25 MHz Max
D0-7
A0-3
RD WR CS
CLK
START

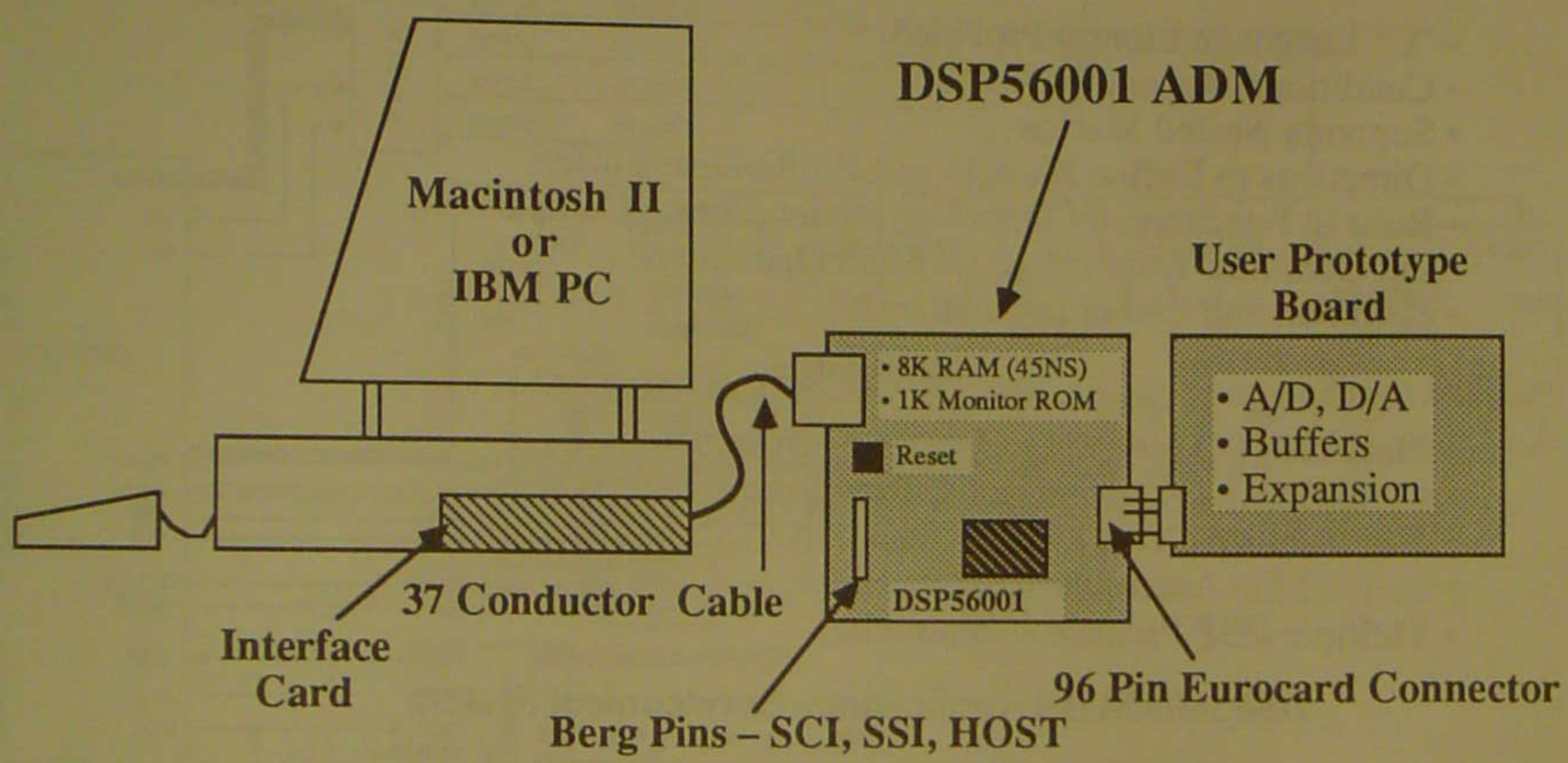
MTA5-9020

# DSP56000/DSP56200 ADAPTIVE FILTER SYSTEM




MTA5-9030

# DSP56000ADS APPLICATIONS DEVELOPMENT SYSTEM FOR DSP56000 FAMILY PRODUCTS

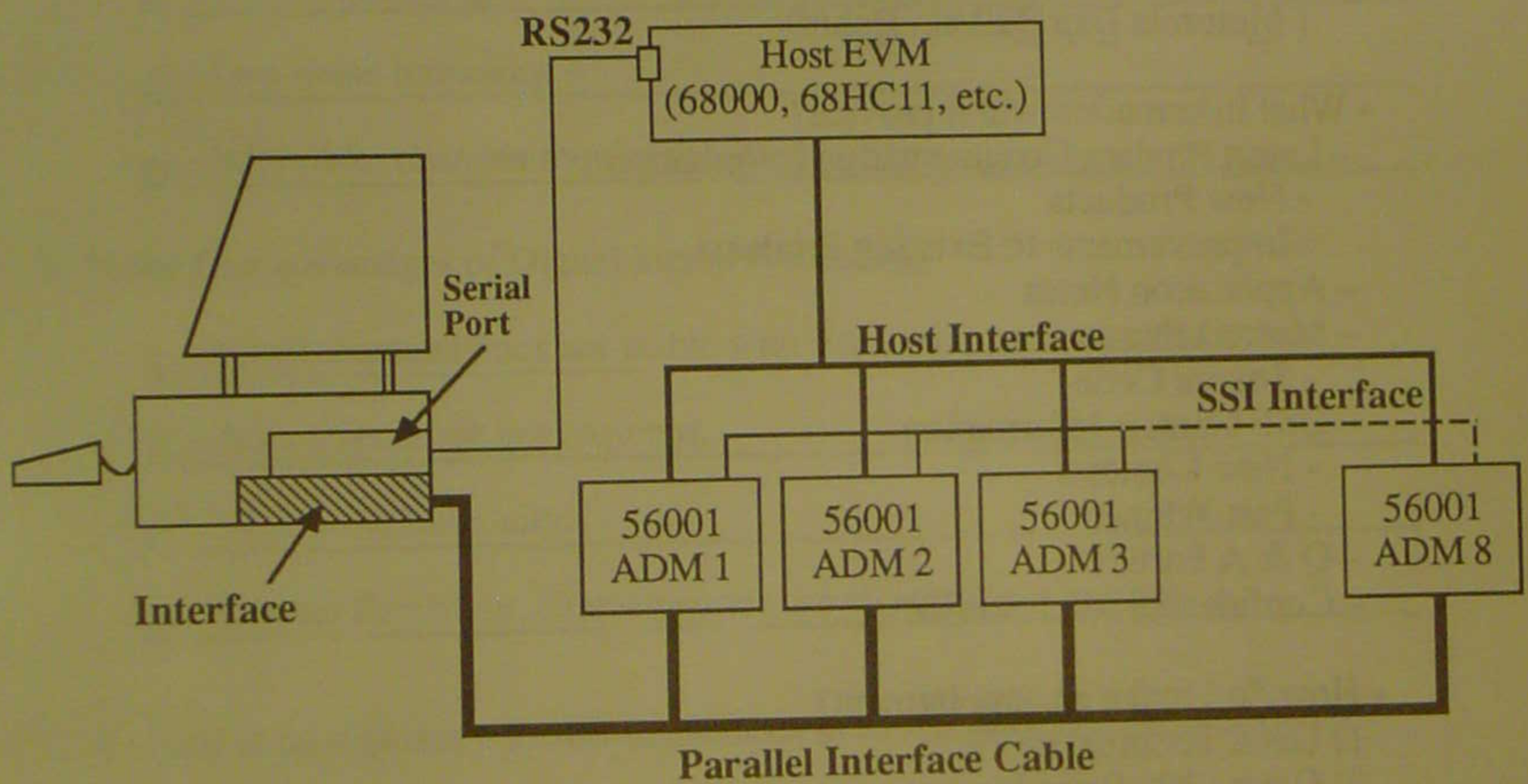



## Commands:

| Memory/Register Modification | File I/O | Execution Control | Miscellaneous | |
|---|---|---|---|---|
| Assemble | Input | Break | Evaluate | Radix |
| Change | Output | Go | Device | System |
| Copy | Load | Force | Quit | Path |
| Display | Log | Step | Help | PC |
| Disassemble | Save | Trace | | |

MTA5-9050

# MULTIPLE ADM HOOKUP




MTA5-9060

# ANSWERS TO EXERCISE 1

## INTERNAL ARCHITECTURE AND PROGRAMMING MODEL

1. Name four drawbacks of Analog Signal Processing.

   A. Filter characteristics are not stable, and vary with time, temp, and manufacturing.

   B. They require adjustment

   C. Low noise immunity

   D. Inflexible, changes require rewiring

2. Name four advantages of Digital Signal Processing.

   A. Filter characteristics are stable with time, temp, and manufacturing.

   B. Adjustments are not required.

   C. High noise immunity.

   D. Greater flexibility, characteristics are changed by software.

3. A signal to be digitized contains frequencies in the range of 200 Hz to 8000 Hz.

   What is the minimum allowable sampling frequency that can be used and still prevent aliasing. 16,000 Hz.

4. Write the basic Digital Signal Processing equation for a Finite Impulse Response Filter.

$$y(n) = \sum_{k=0}^{M} c(k)x(n-k)$$

MTA5-9200

# ANSWERS TO EXERCISE 2

## INTERNAL ARCHITECTURE AND PROGRAMMING MODEL

1. The DSP56000 has three seperate memory spaces called Program, X and Y.

2. The DATA ALU section of the DSP56000 has two 48 bit input registers called X, and Y, plus two 56 bit accumulator registers called A and B.

3. The Address ALU has three sets of 16 bit registers called Address, Offset, and Modifier. There are 8 registers per set.

4. The system stack is 32 bits wide and 15 levels deep. Error conditions are indicated in the Stack Pointer register with the Underflow flag and the Stack Error flag.

5. The Status Register is 16 bits wide, and it is divided into halves called the Mode Register, and the Condition Code Register.

MTA5-9205-2

# ANSWERS TO EXERCISE 3

## INTERNAL ARCHITECTURE AND PROGRAMMING MODEL

1. A word operand is 24 bits. A longword operand is 48 bits, and an accumulator operand is 56 bits.

2. Input register X can be subdivided into registers X1, and X0.

3. Input register Y can be subdivided into registers Y1, and Y0.

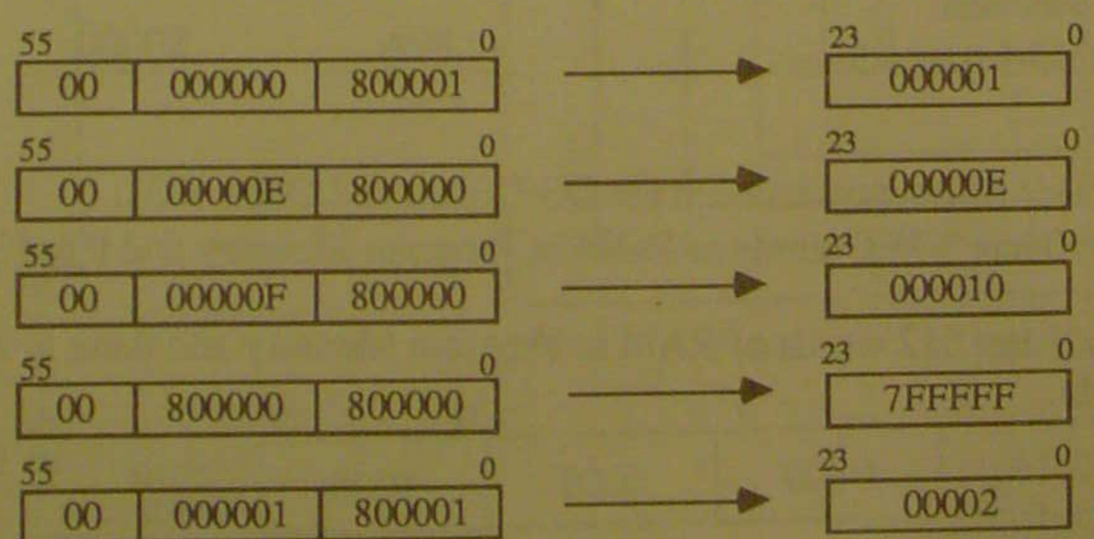
4. Accumulator A can be subdivided into registers A2, A1, A0, and A10.

5. Accumulator B can be subdivided into registers B2, B1, B0, and B10.

6. Accumulator A10 can be subdivided into registers A1, and A0.

7. Accumulator B10 can be subdivided into registers B1, and B0.

8. The values in the 56-bit accumulators on the left are convergently rounded and moved to the 24-bit destinations on the right. Write the result value in the destination, and apply the proper limiting rules.




MTA5-9210-2

# ANSWERS TO EXERCISE 4

## INTERNAL ARCHITECTURE AND PROGRAMMING MODEL

1. The scaling mode bits in the status registers shift transfers right or left from accumulators A or B. These shifts are reflected in the calculation of Convergent Rounding, plus the Extension bit and the Unnormalized bit in the condition code register.

2. Given that the operating mode register in the DSP56000 is programmed MB = 0, and MA = 0, fill in the addresses in the table below. If the memory resource does not exist for this configuration, then write N/A for that table entry.

| | Program Memory | X Memory | Y Memory |
|---|---|---|---|
| Peripheral stop address | N/A | $FFFF | $FFFF |
| Peripheral start address | N/A | $FFC0 | $FFC0 |
| External memory stop address | $FFFF | $FFBF | $FFBF |
| External memory start address | $0F00 | $0200 | $0200 |
| Internal ROM stop address | $0EFF | $01FF | $01FF |
| Internal ROM start address | $0000 | $0100 | $0100 |
| Internal RAM stop address | N/A | $00FF | $00FF |
| Internal RAM start address | N/A | $0000 | $0000 |

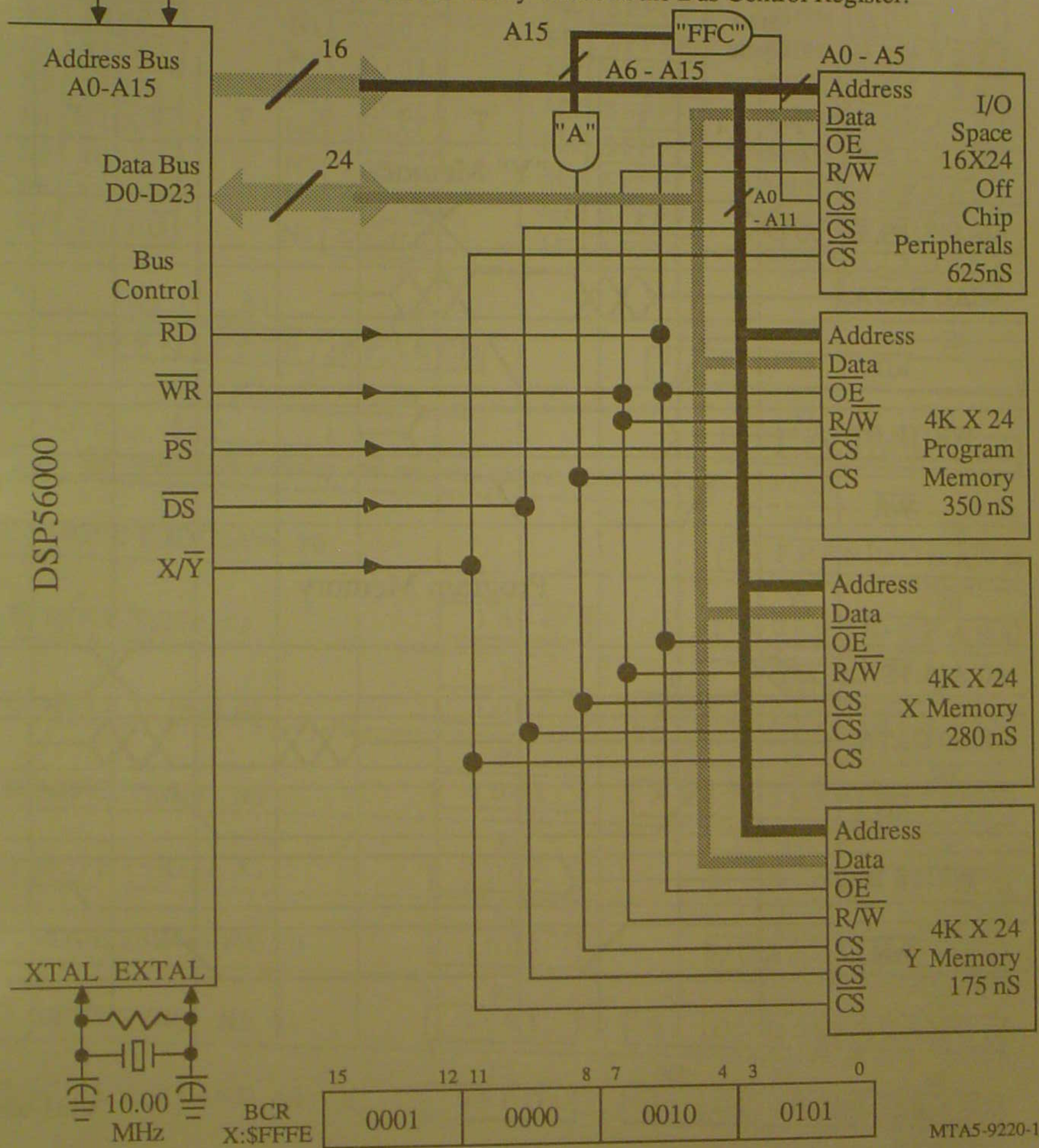
3. What is the basic difference between the DSP56000 and DSP56001.
The DSP56000 has 3.75K words of ROM in Program Memory and there is no DE bit in the OMR.
The DSP56001 has 512 words of RAM in Program Memory and there is a DE bit in the OMR.

MTA5-9215-1

# ANSWER TO EXERCISE 5

## PINS AND BUSES

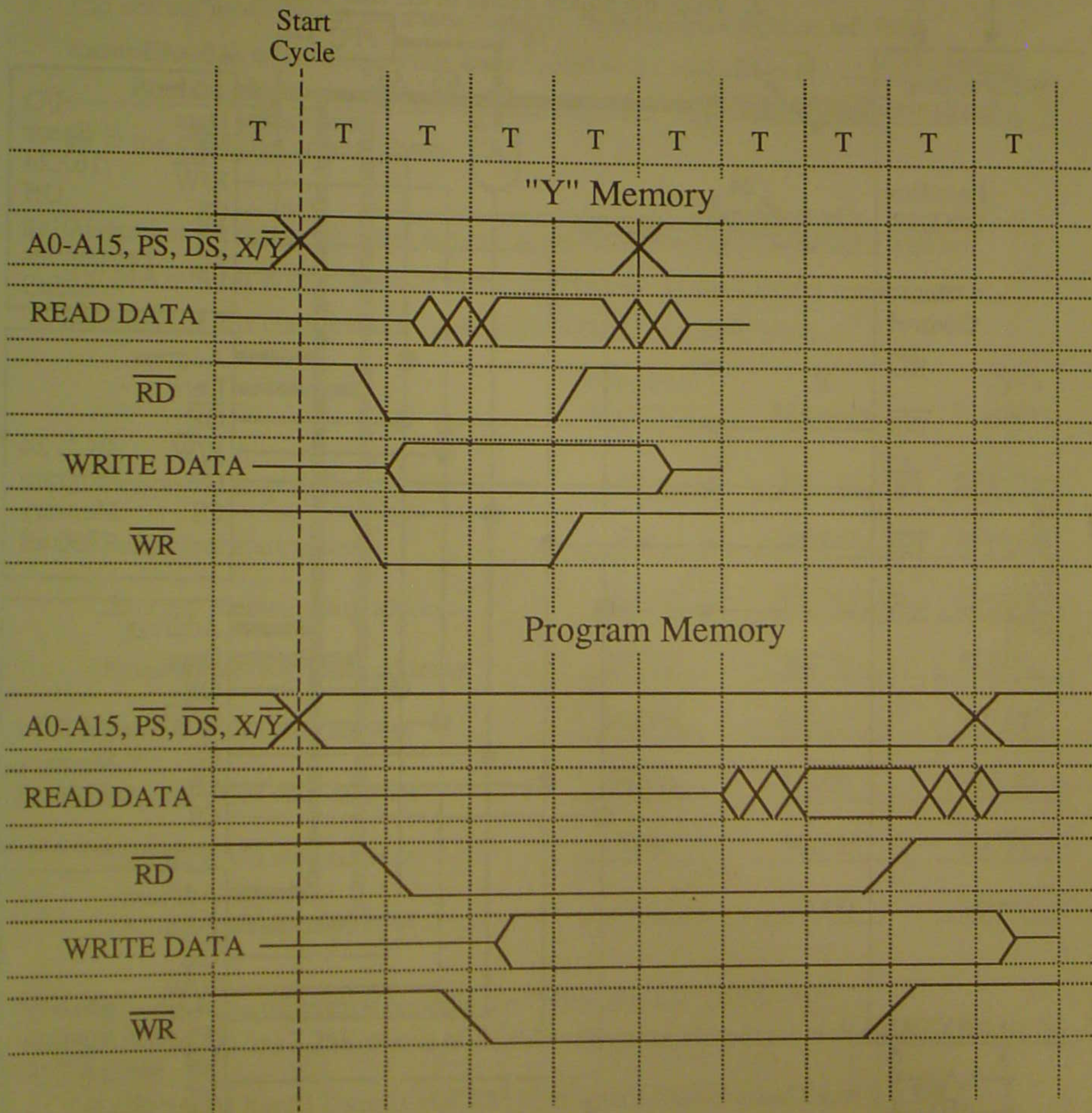
1. Draw the schematic of a simple system that has:
   a. 16 words of I/O at address $FFC0
   b. 4K word of memory starting at $A000 in maps P, X, and Y.
2. Write the binary values in the Bus Control Register.



# ANSWERS TO EXERCISE 5

## PINS AND BUSES (Cont.)

- Draw the read and write timing diagrams for the system in exercise 1 for the Y Memory and Program Memory




MTA5-9225

# ANSWERS TO EXERCISE 6

## GENERAL ADDRESSING MODES

For this exercise assume the registers are initialized as follows:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R0 | $1004 | N0 | $FFFD | M0 | $FFFF | | Y Memory |
| R1 | $1003 | N1 | $0002 | M1 | $FFFF | $1004 | 000000E |
| R2 | $1002 | N2 | $0101 | M2 | $FFFF | $1003 | 654321 |
| R3 | $1001 | N3 | $0003 | M3 | $FFFF | $1002 | 2F2F2F |
| R4 | $1000 | N4 | $0003 | M4 | $FFFF | $1001 | DCBA98 |
| R5 | $0FFF | N5 | $0004 | M5 | $FFFF | $1000 | AAAAAA |
| R6 | $0FFE | N6 | $FFFD | M6 | $FFFF | $0FFF | BBBBBB |
| R7 | $0FFD | N7 | $0004 | M7 | $FFFF | $0FFE | DDD000 |
| | | | | | | $0FFD | 023045 |

| A2 | A1 | A0 |
|---|---|---|
| $01 | 234567 | 89ABCD |

| B2 | B1 | B0 |
|---|---|---|
| | | |

| X1 | X0 |
|---|---|
| | |

| Y1 | Y0 |
|---|---|
| FFFFFF | 000000 |

Show the contents of memory and registers after execution of each problem.

A. MOVE Y:$DCBA98, Y0

| Y1 | Y0 |
|---|---|
| FFFFFF | DCBA98 |

B. MOVE Y:(R0), Y1

| R0 | Y1 | Y0 |
|---|---|---|
| 1004 | 000000E | DCBA98 |

C. MOVE Y:- (R1), B1

| R1 | B2 | B1 | B0 |
|---|---|---|---|
| 1002 | XX | 2F2F2F | XXXXXX |

D. MOVE Y:(R2) -, B0

| R2 | B2 | B1 | B0 |
|---|---|---|---|
| 1001 | XX | 2F2F2F | 2F2F2F |

E. MOVE Y:(R3) +, X1

| R3 | X1 | X0 |
|---|---|---|
| 1002 | DCBA98 | XXXXXX |

F. MOVE Y:(R4) + N4, X0

| R4 | X1 | X0 |
|---|---|---|
| 1003 | DCBA98 | AAAAAA |

G. MOVE Y:(R6) - N6, A1

| R6 | A2 | A1 | A0 |
|---|---|---|---|
| 1001 | 01 | DDD000 | 89ABCD |

H. MOVE Y:(R7 + N7), A0

| R7 | A2 | A1 | A0 |
|---|---|---|---|
| 0FFD | 01 | DDD000 | DCBA98 |

MTA5-9230-2

# ANSWERS TO EXERCISE 7

## GENERAL INSTRUCTIONS

1. When using an instruction with parallel moves, duplicate ___Sources___ are permitted, but duplicate ___Destinations___ are not permitted.

2. The DSP56000/1 has ___62___ instructions that are divided into ___6___ groups.

3. Most instructions in the ___Arithmetic___ and ___Logical___ groups allow parallel moves.

4. Write one instruction that performs the following operations:
   A. Adds accumulator A to B.
   B. Moves X0 to memory using Register 6 in the Postincrement by 1 mode.
   C. Moves memory to Y1 using register 3 in the Postincrement by Offset Nn mode.
   Solution: ___ADD A,B X0,X:(R6)+ Y:(R3)+N3,Y1___

5. Write single instructions to perform the following operations:
   A. Put the processor in Mode 3 with the X and Y ROM enabled. ___MOVE #7,OMR___
   B. Set all external memory and I/O to 5 wait states. ___MOVEP #$5555,X:$FFFE___

6. The following instructions are not permitted. For each instruction, indicate why they are illegal.

| | | | |
|---|---|---|---|
| ADD B,A | X1,B | Y1,A | Duplicate Destination to "A" |
| ADD A,B | X1,X:-(R7) | Y:(R0),Y0 | Predecrement addressing not allowed for XY parallel data move |
| ADD A,B | X0,X:(R4)+N4 | Y0,Y:(R7)+N7 | Registers 4 & 7 are from the same AAU group |
| ADD A,B | Y:(R3),Y | | "Y" is not allowed as a destination for this form of parallel move |

MTA5-9235

# ANSWERS TO EXERCISE 8

## GENERAL INSTRUCTIONS

1. Accumulator A contains a nonzero number.
   Write a program that will normalize Accumulator A.

```
LOOP    NORM    R0,A
        JNN     LOOP
```

2 Write a program that clears the on chip RAM in X and Y memory.

```
        MOVE    #$100,R0        POINTER TO LONG MEMORY
        CLR     B               USED TO CLEAR LONG MEMORY
LOOP    MOVE    B,L:-(R0)       CLEAR ONE LOCATION AND DEC.
        MOVE    R0,A            IS POINTER ZERO?
        TST     A               TEST POINTER
        JNE     LOOP            LOOP IF NOT DONE
```

3 Write a program that finds the largest positive value in the Y memory on chip Data ROM.

```
        MOVE    #$100,B                 END OF Y MEMORY
        MOVE    #$200,R0                START OF Y MEMORY
        MOVE    R0,R1                   POINTER TO LARGEST VALUE
        MOVE    #-1,A                   START COMPARISON TO -1
LOOP    MOVE    Y:-(R0),Y0              GET VALUE FROM Y MEMORY
        CMP     Y0,A                    COMPARE TO LAST VALUE
        TLT     Y0,A        R0,R1       IF LARGER, UPDATE A AND R1
        MOVE    R0,X0                   TEST FOR END OF LOOP
        CMP     X0,B                    IF NOT DONE THEN
        JNE     LOOP                    TEST NEXT LOCATION
```

Note: These program examples can be written more efficiently by using the looping constructs of REP and DO that are discussed in the DSP instructions and addressing modes section.

MTA5-9240

# ANSWERS TO EXERCISE 9

## EXCEPTIONS

**For this exercise the following information is given;**

The Status Register and Interrupt Priority Registers are set as shown:

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LF | 0 | T | 0 | S1 | S0 | 1 | 0 | 0 | L | E | U | N | Z | V | C |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| X:$FFFF | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

There are Interrupts pending from all On Chip Peripherals. Pins IRQA and IRQB are Grounded.

## Answer the following questions:

1. The Interrupt Priority is at level 2 .

2. List the Priority Level for each Interrupt.

| SCI | SSI | HOST | IRQB | IRQA |
|---|---|---|---|---|
| MASKED | 0 | 1 | 1 | 2 |

3. Will the Interrupt Exception be taken? Answer YES or NO for each Pending Interrupt.

| SCI | SSI | HOST | IRQB | IRQA |
|---|---|---|---|---|
| NO | NO | NO | NO | NO |

Note: IRQA is grounded and will never see an edge

4. The DSP56000 has five processing states. They are; Reset , Normal , Stop , Wait , and Exception .

5. There are 32 individual Interrupt Exception Sources.

6. The four Unmaskable Level Three Exceptions are; Reset , Trace , SWI , and Stack Error .

7. Exit from the STOP State is by the Assertion of either Reset , or IRQA .

8. While Tracing the only allowable Exceptions are Reset , Trace , or Stack Error .

MTA5-9245

# ANSWERS TO EXERCISE 10

## DSP INSTRUCTIONS AND ADDRESSING MODES

1. The REPEAT instruction can be used only on ___1___ word instructions and is not ___Interruptable___ .

2. The ENDDO instruction is used to ___Abort___ a DO Loop.

3. Hardware DO Loops can be nested to a maximum of ___7___ levels without storing the stack.

4. The Maximum number of DO Loop iterations is ___65,536___ .

5. Modulo N addressing is useful for creating ___F.I.F.O.'s___ , ___Queues___ , ___Shift___ ___Registers___ and ___Delay___ ___Lines___ .

Given the initial conditions of the Address Arithmetic Unit Registers:

| | Address Registers R | Offset Registers N | Modifier Registers M |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 3 | 2 | 5 |
| 2 | 5 | 5 | 7 |
| 3 | 4 | 8 | 10 |
| 4 | 7 | 7 | 12 |
| 5 | 8 | 11 | 15 |
| 6 | 8 | 11 | 18 |
| 7 | 12 | 12 | 21 |

Fill in the new values of the Address registers after the execution of this code sequence. (NOTE: All numbers are in Base Ten and the problem should be worked in Base Ten.)

```
MOVE X:(R0)+N0,X0   Y:(R4)+N4,Y0
MOVE X:(R1)+N1,X0   Y:(R5)+N5,Y0
MOVE X:(R2)+N2,X0   Y:(R6)+N6,Y0
MOVE X:(R3)+N3,X0   Y:(R7)+N7,Y0
```

| | Address Registers R |
|---|---|
| 0 | 1 |
| 1 | 5 |
| 2 | 2 |
| 3 | 1 |
| 4 | 1 |
| 5 | 3 |
| 6 | 0 |
| 7 | 2 |

MTA5-9275

# ANSWERS TO EXERCISE 11

## DSP INSTRUCTIONS AND ADDRESSING MODES

1. Given the four tap finite impulse response filter and its four coefficients C(0) through C(3);

$$Y(n) = \sum_{K=0}^{3} C(K)\ X(n-K)$$

C(0) = 0.5
C(1) = -0.3
C(2) = -0.6
C(3) = 0.4

For the input sequence X(n) on the left, find the output sequence Y(n) for the requested values.

X(0) = 0.1

X(1) = 0.2

X(2) = 0.3

X(3) = 0.4 Y(3) = 0.03

X(4) = 0.5 Y(4) = 0.03

X(5) = 0.4 Y(5) = -0.07

X(6) = 0.3 Y(6) = -0.11

X(7) = 0.2 Y(7) = -0.03

MTA5-9280

# ANSWERS TO EXERCISE 12

## DSP INSTRUCTIONS AND ADDRESSING MODES

1. Given the initial conditions of Address Arithmetic Unit Registers R0,N0, and M0;

R0 [ 0000 ]  N0 [ $80 ]  M0 [ 0000 ]

Show the new values of R0 for the code sequence.:

```
REP #5
MOVE X:(R0)+N0,X0
```

R0 [ $80 ]

R0 [ $40 ]

R0 [ $C0 ]

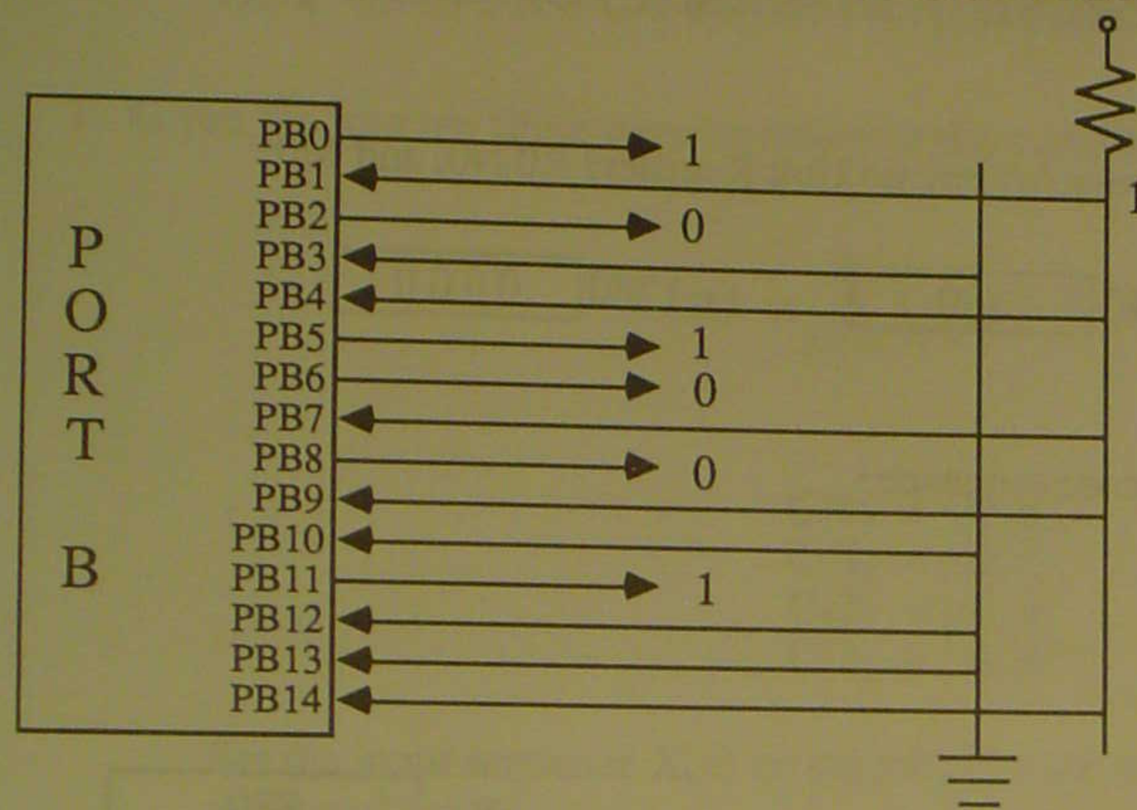
R0 [ $20 ]

R0 [ $A0 ]

MTA5-9285

# ANSWERS TO EXERCISE 13

## ON CHIP PERIPHERALS



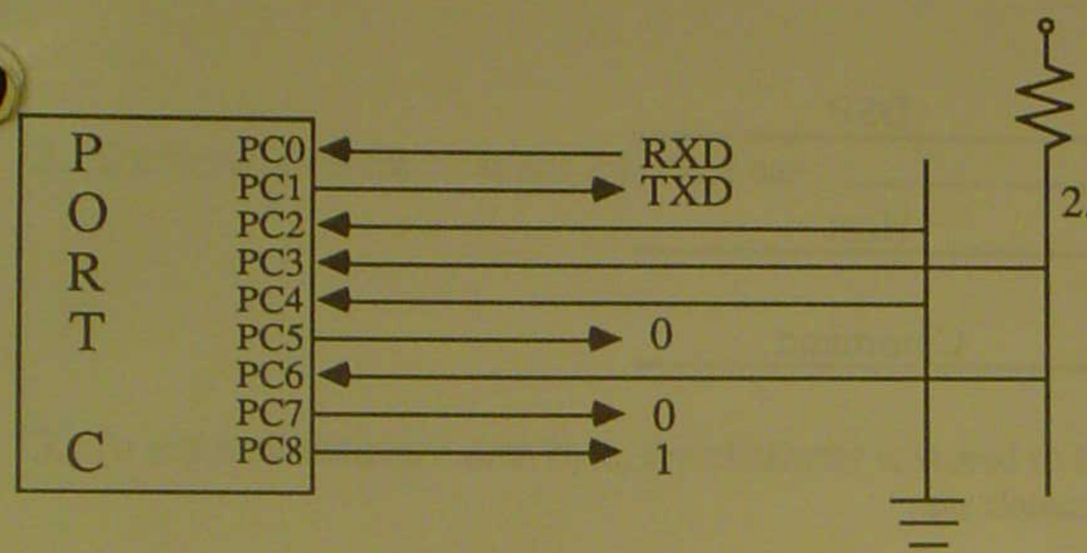

1. For this circuit , write the values in the PORT B Control Register, Data Direction Register and Data Register.

### X:$FFE0 PORT B Control Register (PBC)

| 23 | | | | | | | | | | | | | | | | | | | | | | | 16 | 15 | | | | | | | | | | | | | | | | | | |

# ANSWERS TO EXERCISE 13

## ON CHIP PERIPHERALS (Cont.)




2. For this circuit , write the values in the PORT C Control Register, Data Direction Register and Data Register. If desired use "X" for Don't Care.

### X:$FFE1 PORT C Control Register (PCC)

| 23 | | | | | | | | | | | | | | | 16 15 | CC 8 | CC 7 | CC 6 | CC 5 | CC 4 | CC 3 | CC 2 | CC 1 | CC 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### X:$FFE3 PORT C Data Direction Register (PCDDR)

| 23 | | | | | | | | | | | | | | | 16 15 | CD 8 | CD 7 | CD 6 | CD 5 | CD 4 | CD 3 | CD 2 | CD 1 | CD 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 1 | 1 | 0 | 1 | 0 | 0 | 0 | X | X |

### X:$FFE5 PORT C Data Register (PCD)

| 23 | | | | | | | | | | | | | | | 16 15 | PC 8 | PC 7 | PC 6 | PC 5 | PC 4 | PC 3 | PC 2 | PC 1 | PC 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | X | X |

MTA5-9255

# ANSWERS TO EXERCISE 14

## ON CHIP PERIPHERALS

1. The Host interface permits 8, 16, and 24 bit transfers.

2. The three types of Host transfers are:

   A. Host to DSP

   B. DSP to Host

   C. Host Command

3. The three types of Host handshaking protocols are:

   A. Software Polled

   B. Interrupt Driven

   C. Direct Memory Access

4. The Host interface has 4 user defined Host status flags.

5. The Host Vector in the Command Vector Register is multiplied by 2 to form the exception vector address. It is initialized to $12 on RESET.

6. The Host interface can interrupt the Host processor by using the $\overline{\text{HREQ}}$ pin.

7. The Host interface can provide an interrupt vector on the host data bus when both $\overline{\text{HREQ}}$ and $\overline{\text{HACK}}$ are asserted. This feature is not available in the D.M.A. mode. The host exception vector is initialized to $0F on RESET.

MTA5-9260-1

# ANSWERS TO EXERCISE 15

## ON CHIP PERIPHERALS

1. The Serial Communications Interface supports ___5___ serial formats.

2. The Serial Communications Interface has ___5___ exception vectors.

3. To transmit data as a data byte, the character is stored in the ___STX___ register.

4. To transmit data as an address byte, the character is stored in the ___STXA___ register.

5. The three types of receive errors detected by the SCI are ___Framing___ ___Error___, ___Overrun___ ___Error___, and ___Parity___ ___Error___.

6. The two types of wake up modes are ___Address___ ___Bit___, and ___Idle___ ___Line___.

7. For a multidrop S.C.I. configuration, all the transmit and receive lines may be connected to a bus, providing that the ___Wire___ ___OR___ ___Mode___ ___Select___ bit is set in the Interface Control Register.

MTA5-9265

# ANSWERS TO EXERCISE 16

## ON CHIP PERIPHERALS

1. The synchronous serial interface uses ___6___ pins of port C.

2. The S.S.I. has a programmable word length and can transmit ___8___, ___12___, ___16___, and ___24___ bit data.

3. The synchronous serial interface has ___4___ exception vectors.

4. The data in the S.S.I. transmit and receive registers are ___Left___ justified and are broadcast with the ___Most___ significant bit first.

5. Program control register B to support the following configuration:

   Asynchronous mode
   Gated clock
   SC0 pin is an external receive clock input
   SC1 pin is an internal receive frame sync
   SC2 pin is an internal transmit frame sync
   SCK pin is an internal transmit clock output
   Normal Mode
   Word Wide Frame Sync Length

X:$FFED-SSI Control Register B (CRB)(Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | OF1 | OF0 |
| | | | | MOD | GCK | SYN | FSL | | | SCKD | SCD2 | SCD1 | SCD0 | | |

6. Program control register B to support the following configuration:

   Synchronous mode
   Continuous clock
   SC0 pin is Output Flag Zero
   SC1 pin is Output Flag One
   SC2 pin is an internal transmit frame sync
   SCK pin is an external transmit and receive clock input
   Network mode
   Bit wide frame sync length
   Output Flag Zero is set
   Output Flag One is cleared

X:$FFED-SSI Control Register B (CRB)(Read/Write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RIE | TIE | RE | TE | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| | | | | MOD | GCK | SYN | FSL | | | SCKD | SCD2 | SCD1 | SCD0 | OF1 | OF0 |

MTA5-9270-1

# LIST OF REFERENCES


1. Peled, Abraham and Liu, Bede. "Digital Signal Processing, Theory, Design and Implementation," John Wiley & Sons, 1976.

2. Papoulis, Anthanasios. "Signal Analysis, "McGraw Hill, 1977.

3. Hamming, R.W. "Digital Filters," Prentice-Hall, 1977.

4. Oppenheim, A.V. and Shafer, R.W. "Digital Signal Processing," Prentice-Hall, 1975.

# MTTA5 GLOSSARY

**Adaptive Filter** - A digital filter that can dynamically change its filter characteristics based on an input error signal. The technique used is typically a Least Means Squared (LMS) algorithm.

**Aliasing** - A problem of a high frequency input being interpreted as a lower frequency. Also called frequency folding. This is caused by sampling too slowly. (See Nyquist Criterion and Shannons Theorem)

**Amplitude** - The magnitude or maximum height of a signal.

**Attenuation** - A decrease in output signal magnitude relative to the input signal.

**Block Floating Point** - A technique of representing floating point numbers that have the same magnitude where by a group, or block, of normalized mantissas share one exponent.

**Butterfly, Fast Fourier Transform** - A special algorithm originally developed by Cooley and Tukey that permits efficient implementation of the Discrete Fourier Transform.

**CODEC (COder/DECoder)** - A special type of Analog to Digital and Digital to Analog converter used for speech and telephony applications. It is characterized by a non linear companding conversion function and typically a serial communications interface.

**Convergent Rounding** - A technique for rounding numbers without inducing bias into the result. Also called rounding without bias or round to nearest even.



**Decimation in Frequency** - A reduction in sampling of a discrete frequency signal. This technique is used in the implementation of the Butterfly Discrete Fourier Transform.

**Decimation in Time** - A reduction in sampling rate of a discrete time signal. This technique is used in the implementation of the Butterfly Discrete Fourier Transform.

**Digital Signal Processing** - The processing of signals in discrete time and amplitude using discrete digital logic or computers.

**Discrete Fourier Transform (D.F.T.)** - A mathematical relationship that maps a signal in the discrete time domain into the frequency domain and back again. It is based on the continuous Fourier Transform.

**Fast Fourier Transform (F.F.T.)** - A mathematical relationship that maps a signal in the continuous time domain into the frequency domain.

**Filter Coefficients** - Numbers that define a filter's characteristics. These numbers represent the Fourier transform of the desired filter transfer function.

# MTTA5 GLOSSARY
## (Cont.)

**Finite Impulse Response Filter (F.I.R.)** - A digital filter that is non-recursive and contains no poles.

**Fixed Point Arithmetic** - A method of representing numbers and performing mathematical operations where the number takes on a fixed dynamic range. A fixed point number has only one part, the mantissa or signicand.

**Floating Point Arithmetic** - A method of representing numbers and performing mathematical operations using real numbers that can take on a wide dynamic range of integer and fractional representations. A floating point number consists of a mantissa or signicand, and an exponent or exrad.

**Fractional Arithmetic** - A method of representing numbers and performing mathematical operations using fractions that take on the range of values between +1 and -1. In the DSP56000, -1 is included in this range.

**Frequency Domain** - A way of representing a signal as a spectrum of its component frequencies.

**Harvard Architecture** - A Non-Von Neumann architecture distinguished by multiple bus architectures and the separation of instruction and data in separate memories.

**Infinite Impulse Response Filter (I.I.R.)** - A digital filter that is recursive and contains both poles and zeros.

**Integer Arithmetic** - A method of representing numbers and performing mathematical operations on integer whole numbers. In a digital computer this is the traditional binary arithmetic form.

**Modulo (Modular) Arithmetic** - The arithmetic of residues. The representation of an integer by dividing it by the modulus M and using the remainder where all integer values N fall in the range of $0 \leq N < M$. For example, hexadecimal notation is a modulus 16 arithmetic. Any integer converted to Modulus 16 arithmetic fall in the range of 0 to 15 ($0 to $F).

**Multiply/Accumulate (M.A.C.)** - The basic mathematical function use in digital filtering. Most times it involves multiplying the input data by a coefficient and then summing the result with previous multiply/accumulate cycles.

**Nyquist Criterion** - The input frequency cannot contain any frequency greater than one-half of the sampling rate. (See Shannons Theorem)

**Passband** - The frequency range of no signal attenuation. Signals in this range pass through the filter unaltered, except possibly for some gain in the passband.

# MTTA5 GLOSSARY
## (Cont.)

**Passband Ripple** - The maximum amount of excursion in the passband from the desired output magnitude.

**Quantization** - The process of representing a signal in continuous time or amplitude in discrete time or amplitude.

**Residue** - The remainder portion of a division by a modulus M. (See Modulus Arithmetic)

**Sampling** - The chopping up, or quantizing, a continuous time signal into discrete time elements.

**Sampling Rate** - The rate at which the input signal value is sampled. This is usually performed by an Analog to Digital converter.

**Sampling Theorem** - See Nyquist Criterion or Shannons Theorem.

**Saturation Arithmetic** - A technique to minimize the effects of overflow errors by substituting the largest allowable numeric value that is closest to the overflow result, for the overflowed result.

**Shannons Theorem** - In a band of frequencies of bandwidth W, all the information content of the signal is essentially conveyed with a sampling frequency of 2W. (See Nyquist Criterion)

**Spectrum** - A collection of frequencies.

**Stopband** - The frequency range of signal attenuation.

**Stopband Attenuation** - The minimum amount of attenuation in the stopband.

**Tap** - Delays in a digital filter. The number of taps equals the number of filter coefficients and also the number of sampled input values processed by the filter for each output point.

**Time Domain** - Representing the amplitude of a signal as a function of time.

**Von Neumann Architecture** - The traditional type of computer architecture whereby instructions and data are stored in the same memory.

**Z-Transform** - A mathematical relationship that simplifies discrete time system analysis similar to the way the Laplace transform eases analysis of continuous time system evaluation.