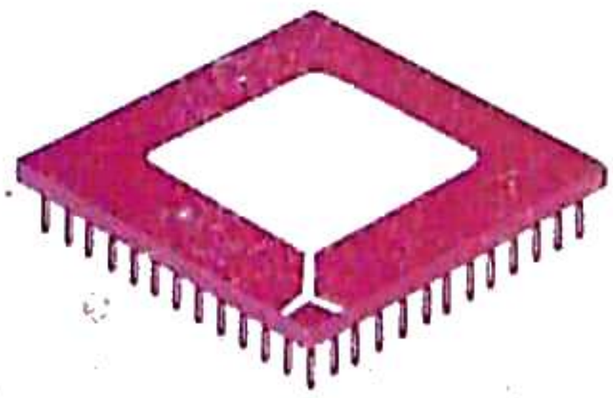
MTT31/CN
Rev. 3

MOTOROLA TECHNICAL TRAINING

DSP56000/1



Course Notes

MOTOROLA INC.
Semiconductor Products Sector

DANIEL A. JACOBY
ING. ELECTRONICO
AMENABAR 2355 - 1° A 782-6230

DSP56000/1
COURSE NOTES

Prepared by
MOTOROLA SEMICONDUCTOR PRODUCTS SECTOR
World Marketing Training Operations
Phoenix, Arizona

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 the Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© Copyright Motorola Inc. 1990, All Rights Reserved
PRINTED SEPT, 1991

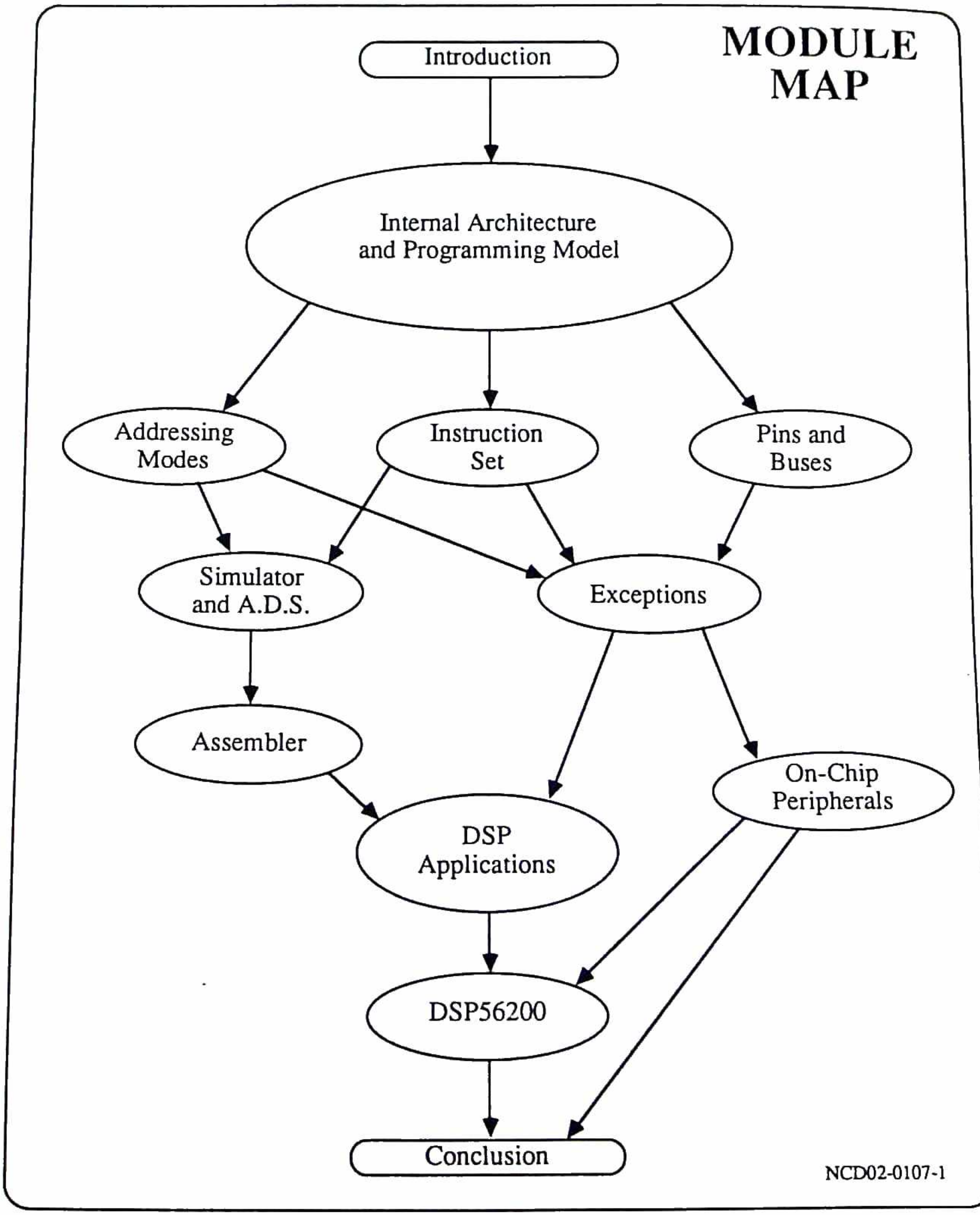
DANIEL A. JACOBY
ING. ELECTRONICO
AMENABAR 2255 - 1º A 788-8239

DSP56000/1
COURSE NOTES

Prepared by
MOTOROLA SEMICONDUCTOR PRODUCTS SECTOR
World Marketing Training Operations
Phoenix, Arizona

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 the Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© Copyright Motorola Inc. 1990, All Rights Reserved
PRINTED SEPT, 1991


MODULE MAP
Introduction
Internal Architecture and Programming Model
Addressing Modes
Instruction Set
Pins and Buses
Simulator and A.D.S.
Exceptions
Assembler
On-Chip Peripherals
DSP Applications
DSP56200
Conclusion
NCD02-0107-1

COURSE SCHEDULE

DAY 1

Internal Architecture and Programming Model

Pins and Buses

Addressing Modes

DAY 2

Instructions

Applications Development System

A.D.S. Lab

DAY 3

Exceptions

Parallel I/O

Serial Communications Interface

Host Interface

Assembler

Assembler Lab

DAY 4

Synchronous Serial Interface

DSP Algorithms

DSP56200

Filter Design Package

DSP Lab

REFERENCE DOCUMENTS AND MATERIALS

1. DSP56000UM/AD – DSP56000 Digital Signal Processor User's Manual
2. DSP56001/D – DSP56001 Data Sheet
3. MTT31 Course Notes
4. DSP56200/D – DSP56200 Data Sheet
5. BR526 – DSP56000 CLASx Software Summary
6. BR-297 – Dr. BuB
7. APR1 – Digital Sine Wave Synthesis Using the DSP56001
8. BR283 DSP56200 Technical Summary
9. APR2 10 Band Graphic Equalizer
10. BR517 – DSP56000ADS Application Development System
11. BR522 – DSP320 to 56001 Translator Software

NCD02-0109

Literature Order Form

Document	Description	Price 1-9	Wt. lbs.	QtyAmt
DSP56000UM/AD	DSP56000 Digital Signal Processor User's Manual	3.85	2	
* BR505	DSP56001 Technical Summary	.15	–	
* BR282	DSP56000 Technical Summary	.15	–	
* BR283	DSP56200 Technical Summary	.15	–	
* BR284	DSP56000 Simulator/Assembler Technical Summary	.15	–	
* BR297	Dr. BuB Flyer	.15	–	
* BR517	DSP56000ADS Applications Development System	.20	–	
* BR522/D	DSP320 to DSP56001 Translator Software	.15	–	

* BR505, * BR282, * BR283, * BR284, * BR297, * BR517, * BR522/D, *APR1
— No charge for quantity under 10. Ten and up - priced as shown.

Postage and Handling – U.S., Canada, Mexico, Central America Surface – $1.00 per lb. Air – 2.00 per lb.	South America, Europe – Surface – $1.50 per lb. Air – 5.00 per lb.	
	Rest of World – Surface – $1.50 per lb Air – 7.00 per lb.	
	GRAND TOTAL $	

Documents will be sent best way surface unless specified for air delivery. Allow 2 to 3 weeks for delivery. USA purchase orders will be honored for orders of $ 50.00 or more if a credit line has previously been established. If not, please send check or money order. Mail with remittance to:

MOTOROLA SEMICONDUCTOR PRODUCTS INC.

NAME _____

COMPANY _____ DEPT. _____
(P.O. Box not deliverable)

ADDRESS _____

NCD02-0110

DSP56000/1 FEATURES

- 10.25 Million Instructions Per Second (MIPS)

- Single Cycle, non-pipelined Data ALU
  - 24 x 24 ⟶ 56 Bit Parallel Multiply/Accumulate
  - 10 Data Registers
  - 2 Data Bus Shifter/Limiters

- DSP Oriented Address ALU
  - 24 Address Registers
  - Dual Modulo Arithmetic Units
  - Linear, Modulo, and Bit Reversed Address Generation

- Advanced Program Controller
  - 15 Level Hardware Stack
  - Nested Hardware DO Loops
  - No Overhead Auto-Return Interrupts

- Highly Orthogonal Instruction Set
  - 62 Instruction Types
  - Makes Pipeline Invisible
  - Suitable for High Level Language (HLL) Compilers

- Multiple Buses
  - 4 Data Buses
  - 3 Address Buses
  - Programmable Off-Chip Access Times

- On-Chip Memory
  - Two Independent 256 x 24-Bit Data RAMs
  - Two Independent 256 x 24-Bit Data ROMs
  - 3.75K x 24-Bit Program ROM (DSP56000 Only)
  - 512 x 24-Bit Program RAM (DSP56001 Only)

- On-Chip Peripherals
  - 24 Programmable I/O Port Pins
    or a combination of I/O Port Pins and
  - 8-bit Parallel Host MPU/DMA Interface
  - Serial Communication Port with Baud Rate Generator
  - Synchronous Serial (Codec) Port with Clock Generator

NCD02-0112

DSP56200 FEATURES

CASCADABLE ADAPTIVE FINITE IMPULSE RESPONSE DIGITAL FILTER (CAFIR)

- Three Modes of Operation
  - Single FIR Filter
  - Dual FIR Filter (Two Independent FIR Filters)
  - Single Adaptive FIR Filter
- High Performance Hardware
  - 24 x 16 Bit Multiplication with 40-Bit Accumulation
  - 10.25 MHz Internal Operation
  - 28-Pin Dual-in-Line Package
- Architecture Optimized for Digital Filtering
  - Three Execution Units Operate in Parallel
  - Multiple Internal Buses
- Digital Filtering Features
  - 16-Bit Rounding Option on Filtered Output
  - dc Tap Option
  - Programmable Filter Tap Length (4 to 256 Taps)
  - Programmable Loop Gain
  - Progammable Coefficient Leakage Term
  - Adaptation Disable Capability
  - LMS Adaptation Algorithm
- Unlimited Cascadability Offers
  - Longer-Tap Filters
  - Higher Sampling Rates
- High Throughput Rates – Examples of Possible Configurations
  - 227 kHz FIR Filter (32 Taps, 1 DSP56200)
  - 37 kHz FIR Filter (256 Taps, 1 DSP56200)
  - 115 kHz Adaptive Filter (256 Taps, 8 Cascaded DSP56200s)
  - 19 kHz Adaptive Filter (256 Taps, 1 DSP56200)
  - Many Other Configurations Possible
- Simple Interface to Popular Hosts
  - Microprocessors
  - Microcomputers
  - General Purpose Digital Signal Processors

NCD02-0114-1

DSP56200 FEATURES

CASCADABLE ADAPTIVE FINITE IMPULSE RESPONSE DIGITAL FILTER (CAFIR)

- Three Modes of Operation
  - Single FIR Filter
  - Dual FIR Filter (Two Independent FIR Filters)
  - Single Adaptive FIR Filter

- High Performance Hardware
  - 24 x 16 Bit Multiplication with 40-Bit Accumulation
  - 10.25 MHz Internal Operation
  - 28-Pin Dual-in-Line Package

- Architecture Optimized for Digital Filtering
  - Three Execution Units Operate in Parallel
  - Multiple Internal Buses

- Digital Filtering Features
  - 16-Bit Rounding Option on Filtered Output
  - dc Tap Option
  - Programmable Filter Tap Length (4 to 256 Taps)
  - Programmable Loop Gain
  - Progammable Coefficient Leakage Term
  - Adaptation Disable Capability
  - LMS Adaptation Algorithm

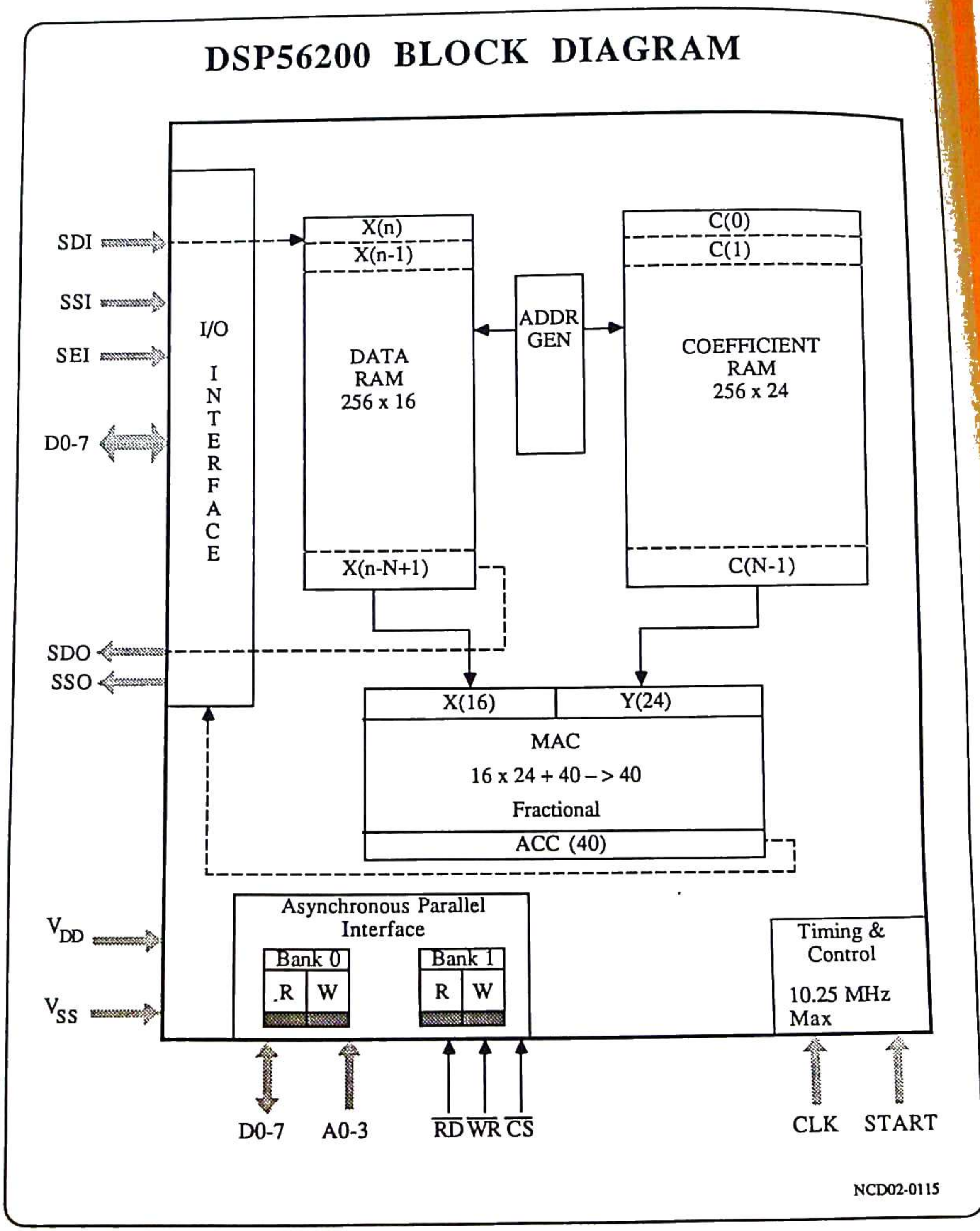
- Unlimited Cascadability Offers
  - Longer-Tap Filters
  - Higher Sampling Rates

- High Throughput Rates – Examples of Possible Configurations
  - 227 kHz FIR Filter (32 Taps, 1 DSP56200)
  - 37 kHz FIR Filter (256 Taps, 1 DSP56200)
  - 115 kHz Adaptive Filter (256 Taps, 8 Cascaded DSP56200s)
  - 19 kHz Adaptive Filter (256 Taps, 1 DSP56200)
  - Many Other Configurations Possible

- Simple Interface to Popular Hosts
  - Microprocessors
  - Microcomputers
  - General Purpose Digital Signal Processors

NCD02-0114-1

DSP56200 BLOCK DIAGRAM



DSP56000/1

INTERNAL ARCHITECTURE

AND

PROGRAMMING MODEL

Objectives:

- Describe the Architecture and Programming Model of the DSP56000/1.
- Describe the numeric representation of the DSP56000/1.
- Describe the operation of the DSP56000/1 Status Register.
- Describe the Memory Modes of the DSP56000/1.

NCD02-0200-2

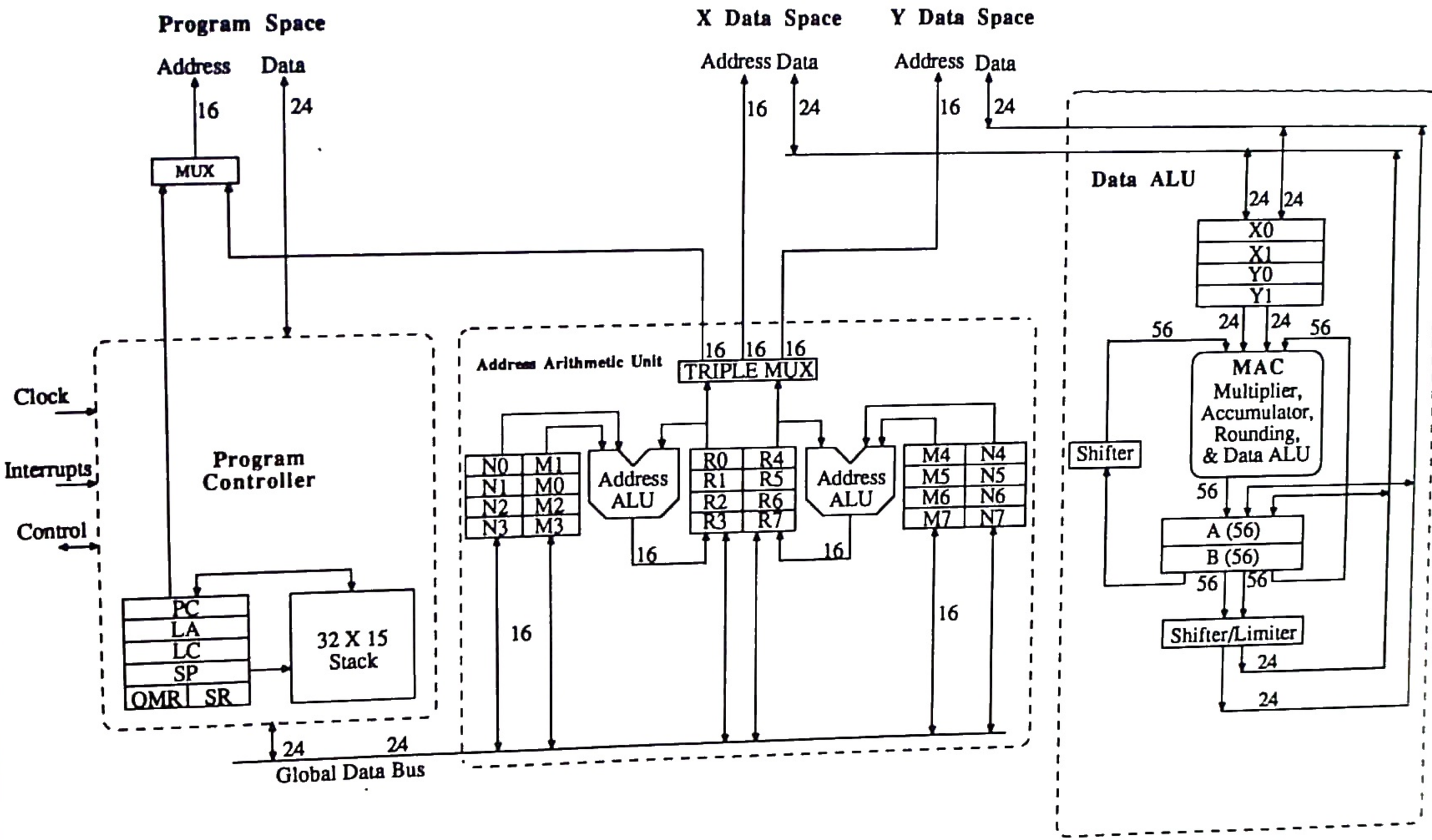
Table of Contents

DSP56000/1 Block Diagram 2-5
DSP56000/1 Programming Model 2-6
Core Processor 2-7
Program Controller 2-8
Hardware Stack 2-8
Stack Pointer Values 2-9
Operating Mode Register 2-9
Status Register 2-10
Loop Registers 2-10
Core Processor 2-11
Data ALU Registers 2-12
Data ALU 2-13
Operand Types 2-14
Fractional vs. Integer 2-14
Fractional vs. Integer Multiplication 2-15
Fractional vs. Floating Point 2-15
Word Operand 2-16
Fractional Examples 2-16
Memory Types 2-17
X Y Memory 2-17
Long word Operand 2-18
L Memory 2-18
Accumulator Operand 2-19
Accumulator Operand Examples 2-19
Data Transfers 2-20
Word Transfers 2-20
Long word Transfers 2-21
Hitting the Rail 2-21
Saturation Arithmetic 2-22
Limiting Cases 2-23
X and Y Limiting 2-24
Rounding 2-24
Convergent Rounding 2-25
Status Register 2-26
Z, N, and C Condition Code Bits 2-26
V - Overflow Bit 2-27
E - Extention Bit 2-27
U - Unnormalized Bit 2-28
Shifter / Limiters 2-28

Table of Contents

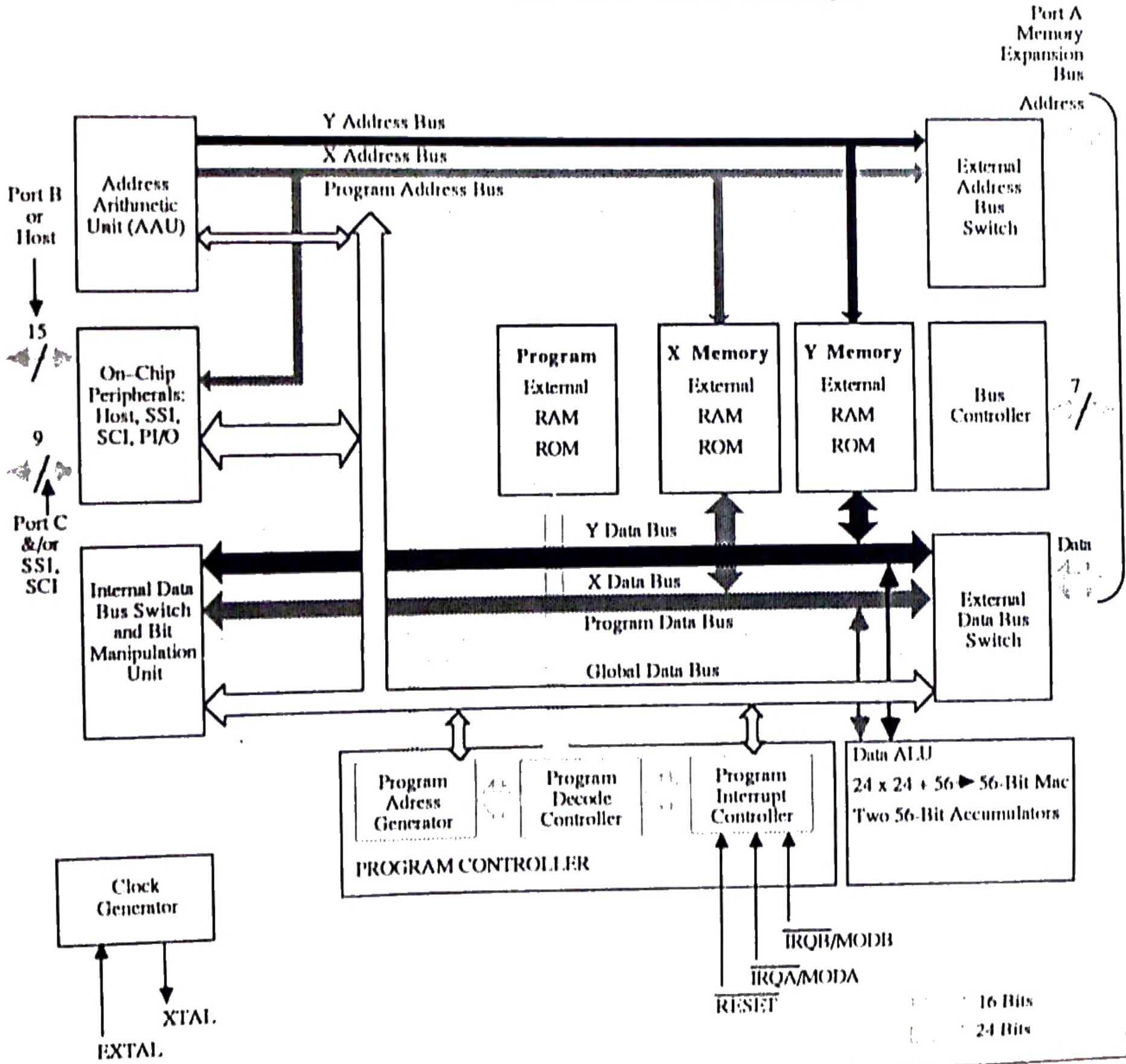
Scaling Mode Bits 2-29
Scaling and Convergent Rounding 2-29
Scaling and Extension Bit 2-30
Scaling and Unnormalize Bit 2-30
DSP56000 Memory Map Mode 0 2-31
DSP56000 Modes 2-32
DSP56001 Memory Map Mode 0 2-33
DSP56001 Memory Map Mode 2 2-34
DSP56001 Memory Map Mode 3 2-35
DSP56001 Modes 2-36
Parallel Bus Transfers 2-37
Instruction Pipeline 2-38

CORE PROCESSOR

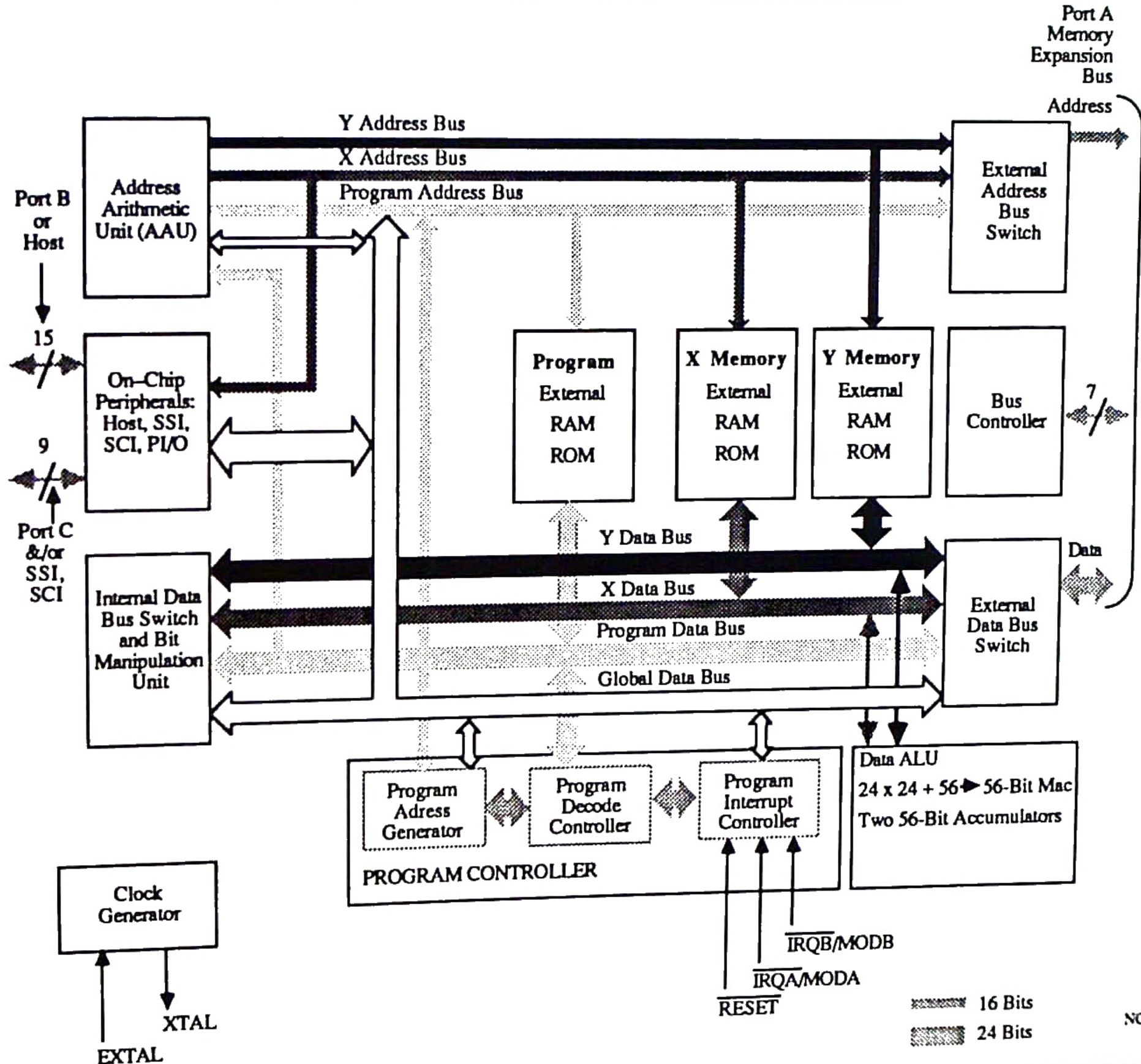


NCD02-0214

DSP56000/1 BLOCK DIAGRAM



DSP56000/1 BLOCK DIAGRAM



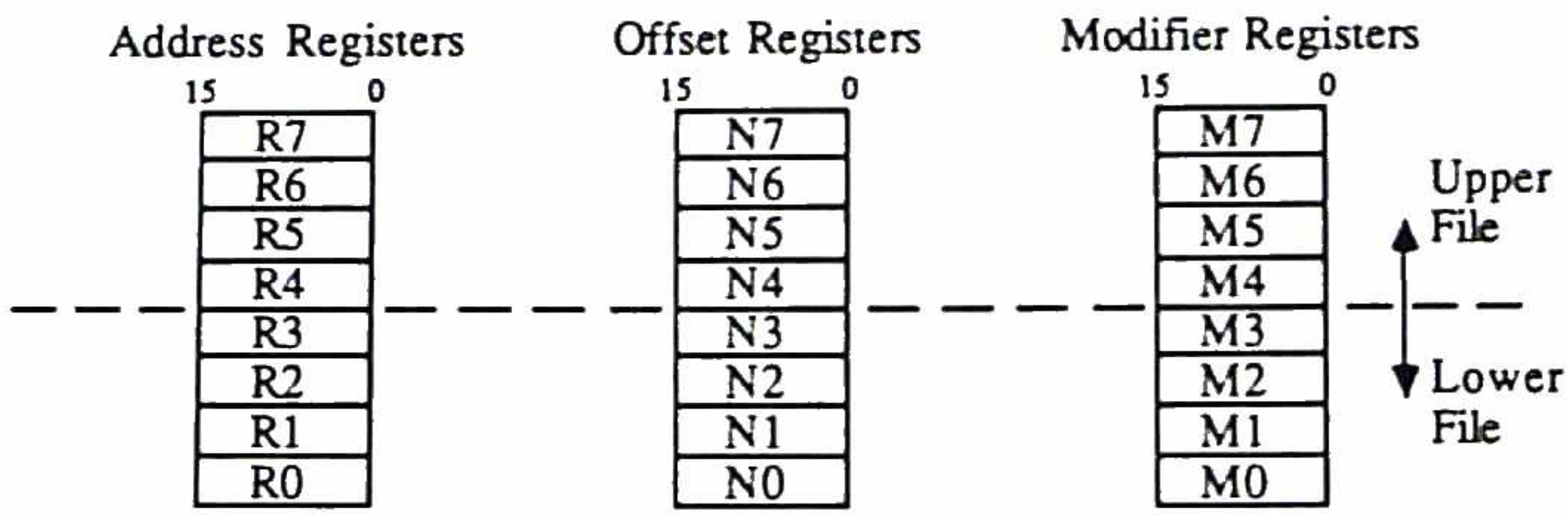
DSP56000/1 PROGRAMMING MODEL

Data ALU

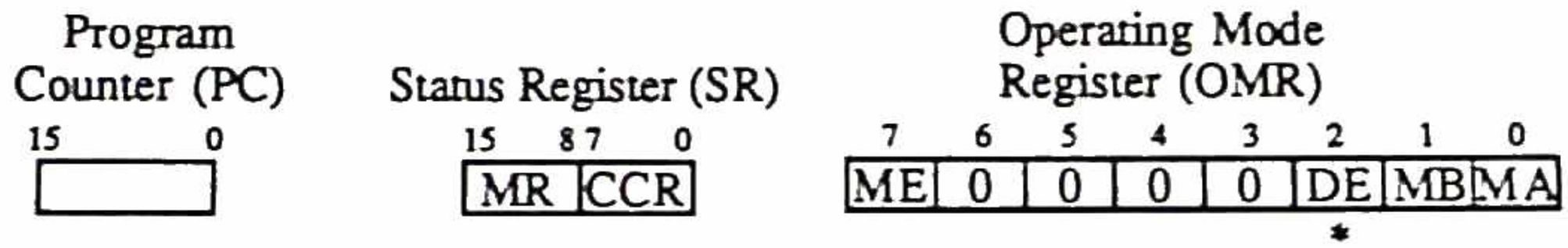
Input Registers

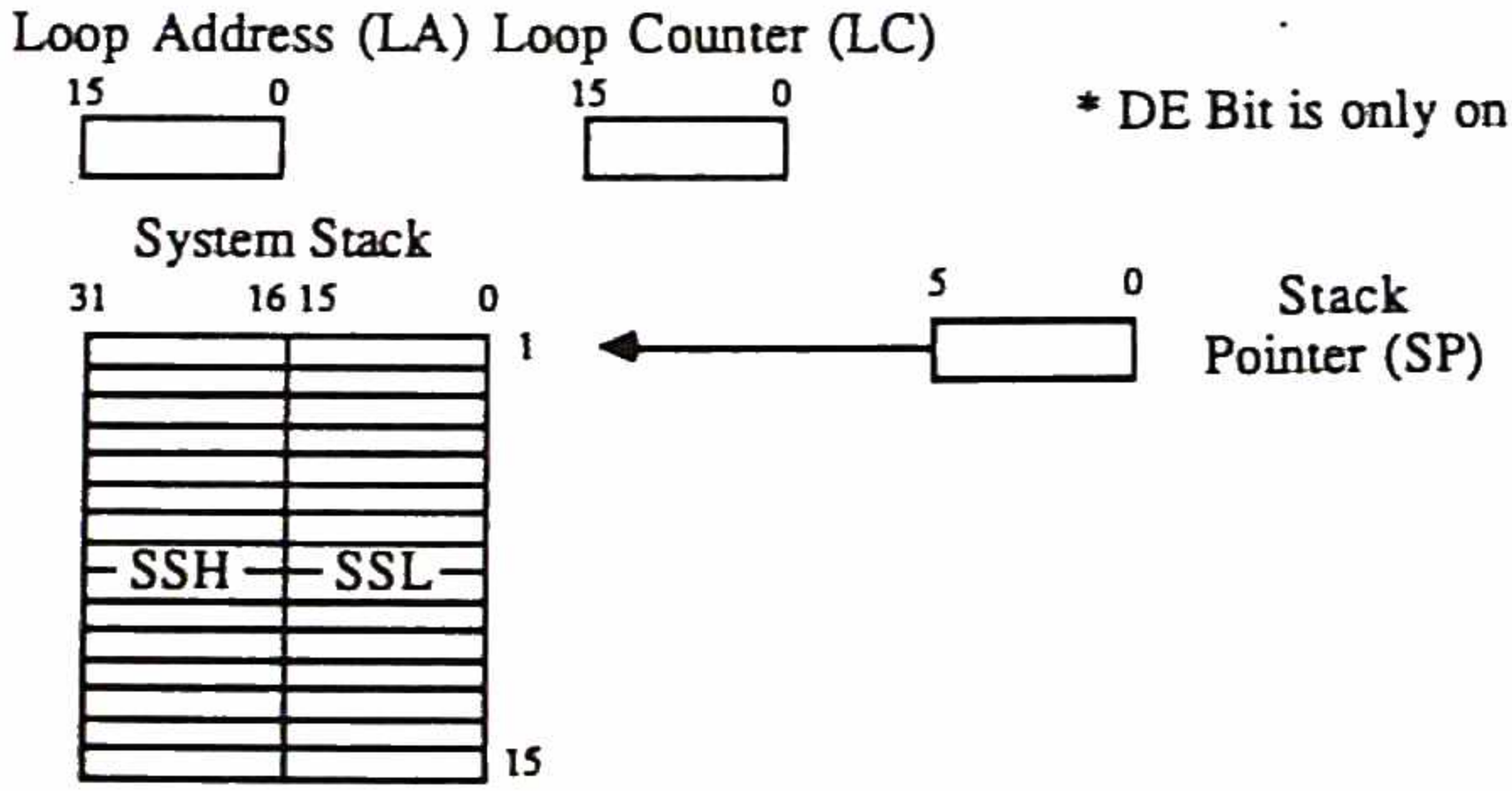
Accumulator Registers

Address ALU



Program Controller

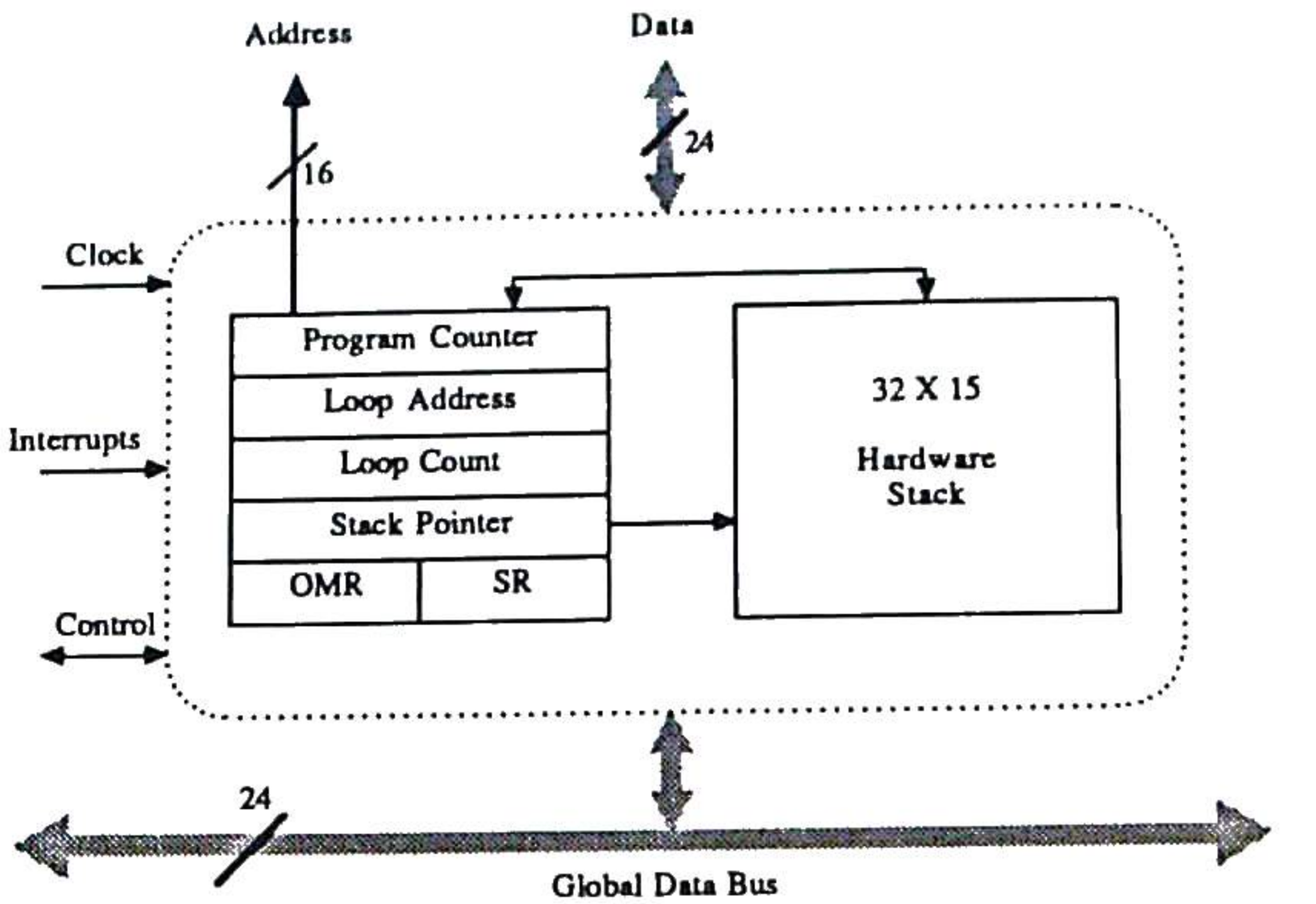




* DE Bit is only on the DSP56001

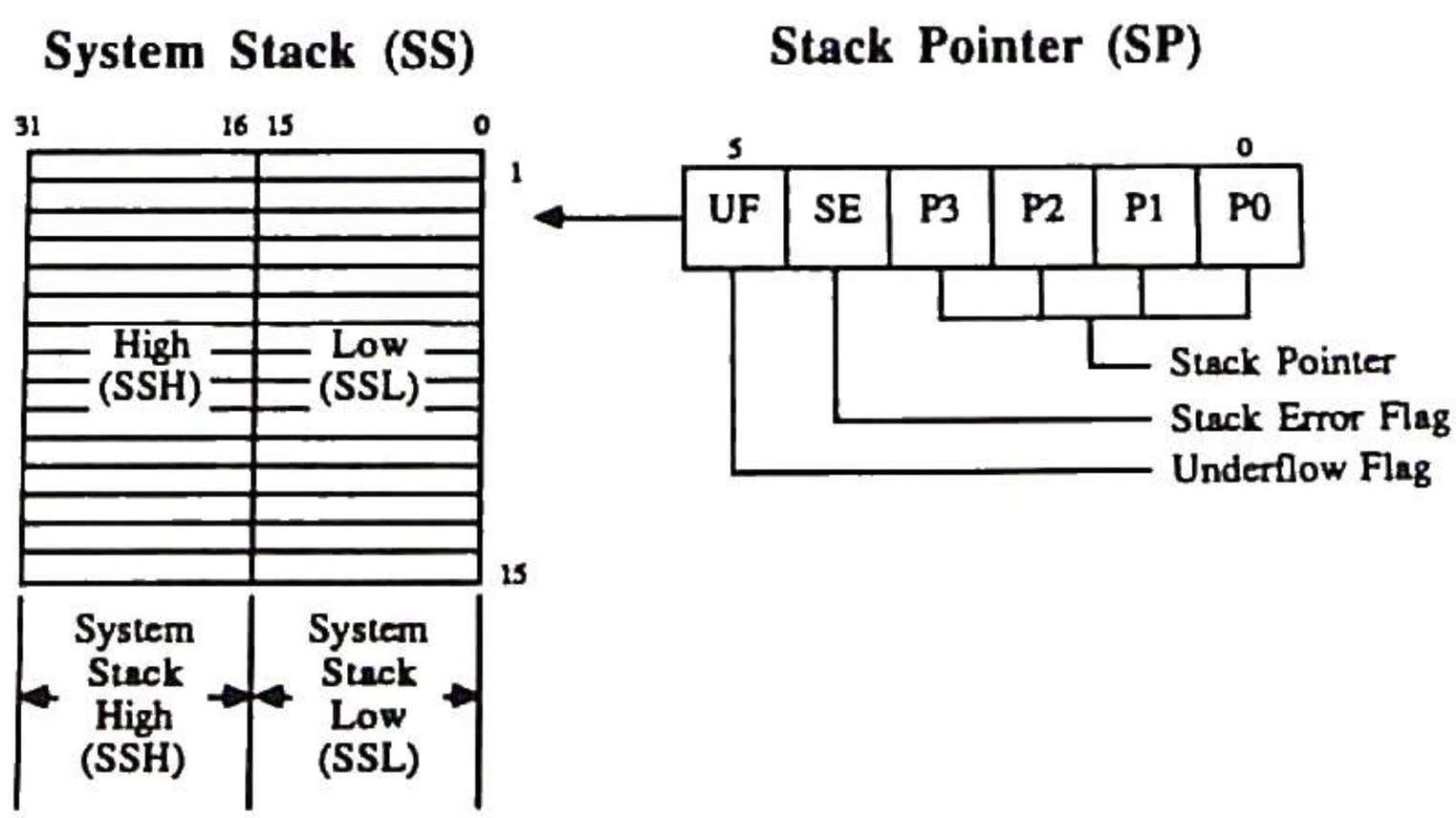
NCD02-0213

PROGRAM CONTROLLER



NCD02-0215

HARDWARE STACK



Uses:

- Subroutines
- Long Interrupts
- Hardware "DO LOOPS"
- Not recommended for data storage

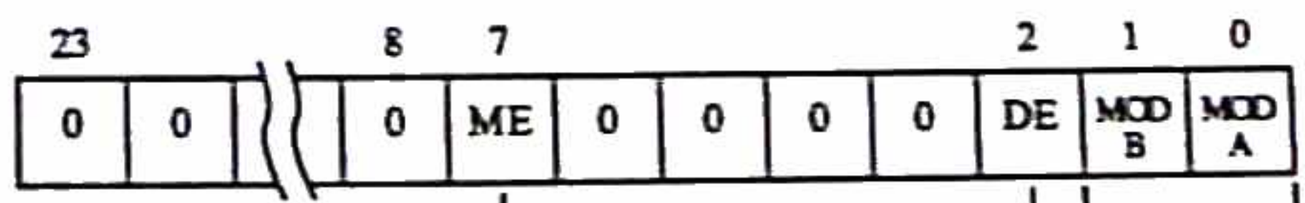
NCD02-0216

STACK POINTER VALUES

UF	SE	P3	P2	P1	P0		
1	1	1	1	1	0	← Stack Underflow condition after double pull	Stack Error Exception
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	Stack Error Exception
0	1	0	0	0	1	← Stack Overflow condition after double push	

NCD00-0217

OPERATING MODE REGISTER (OMR)



Memory Ready Enable
0 = Bus Request/Bus Grant Arbitration (RESET)
1 = Memory Ready/Memory Ready Strobe

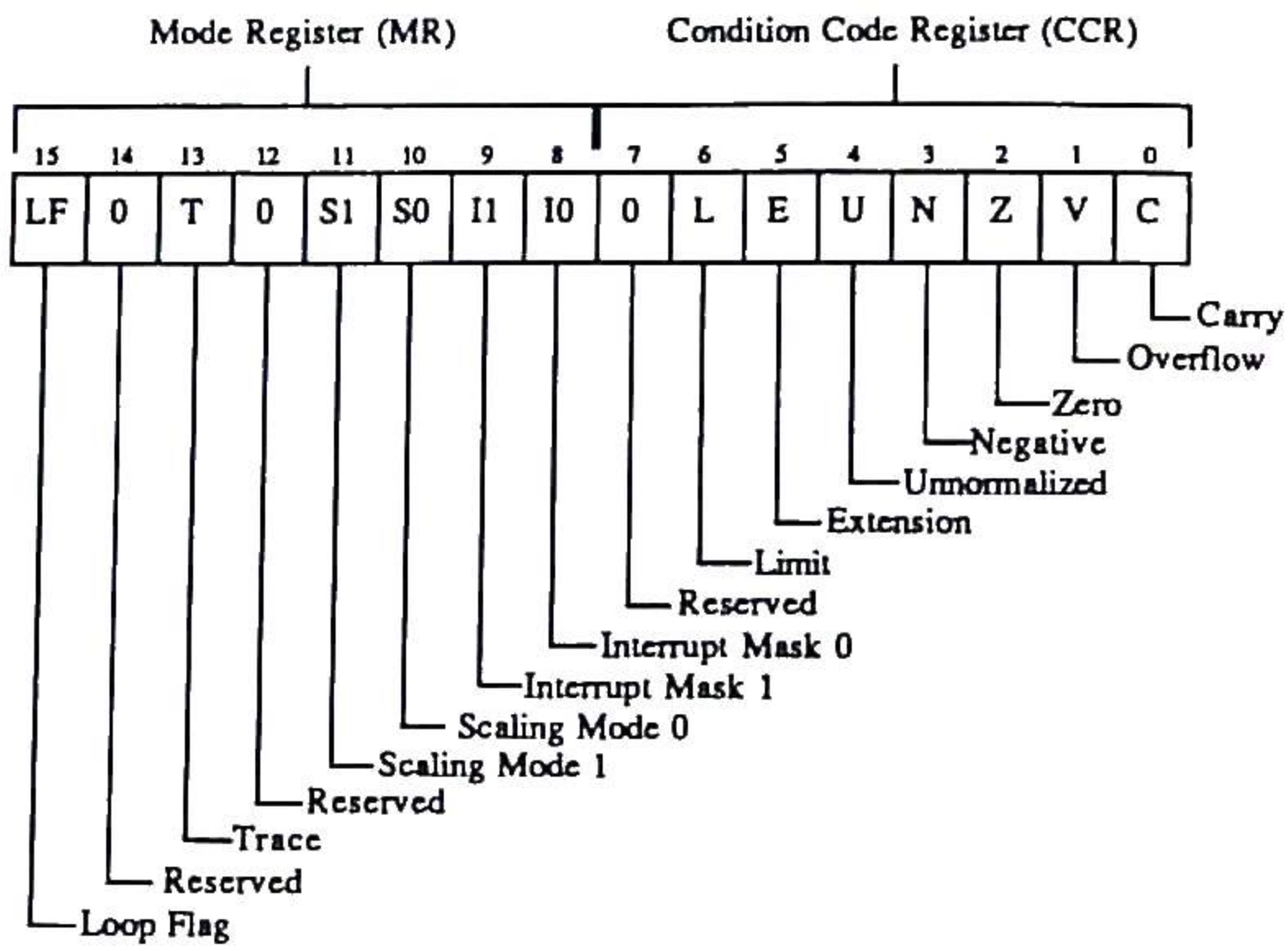
Data ROM Enable (DSP56001 Only)
0 = Disable ROM in X and Y memory (RESET)
1 = Enable ROM in X and Y memory

RESET Condition
MODA = IRQA
MODB = IRQB
DE = 0
ME = 0

All Control Bits in the OMR are Software Programmable

NCD00-0218

STATUS REGISTER

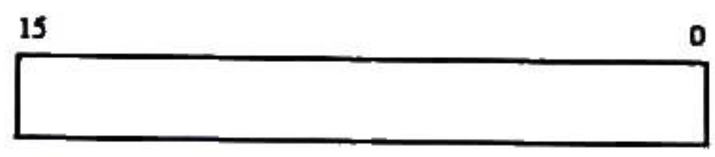


NCD02-0219

LOOP REGISTERS

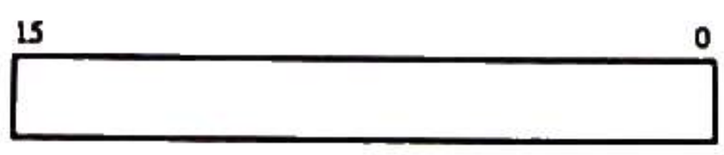
- Used by Hardware DO-Loops.

Loop Address



Where to end the Loop

Loop Count

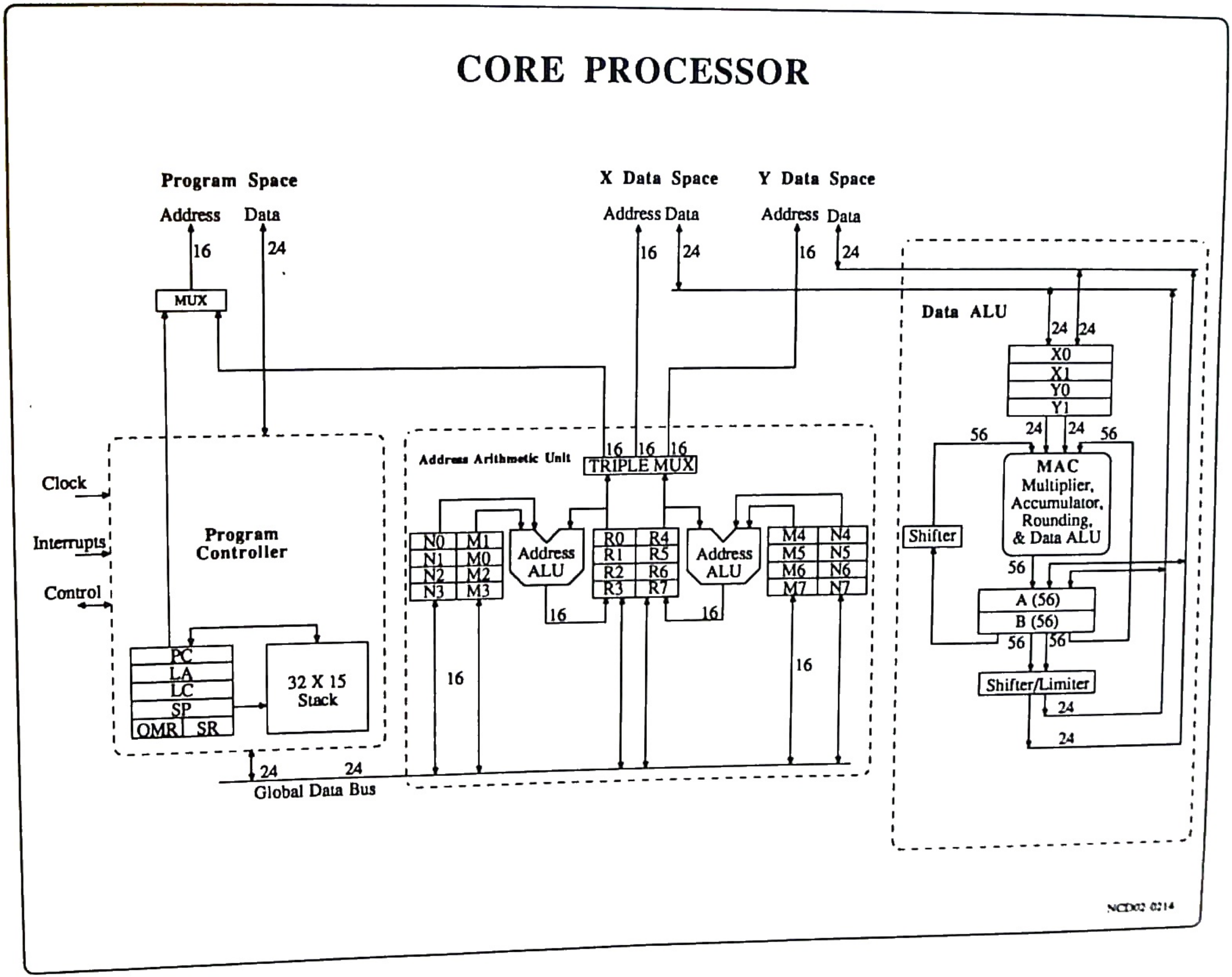


Number of Iteration

- REPeat Instruction uses Loop Count Register

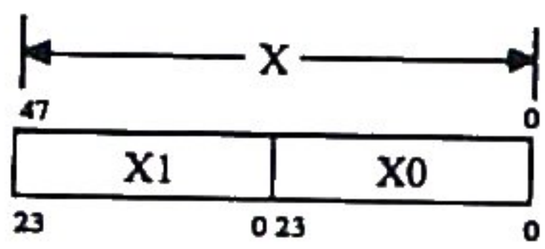
NCD02-0220

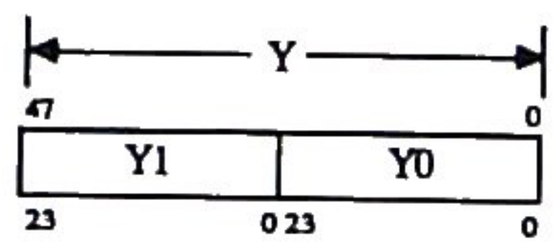
CORE PROCESSOR



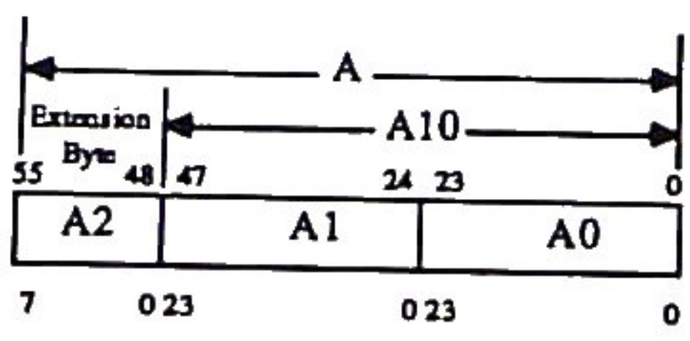
DATA ALU REGISTERS

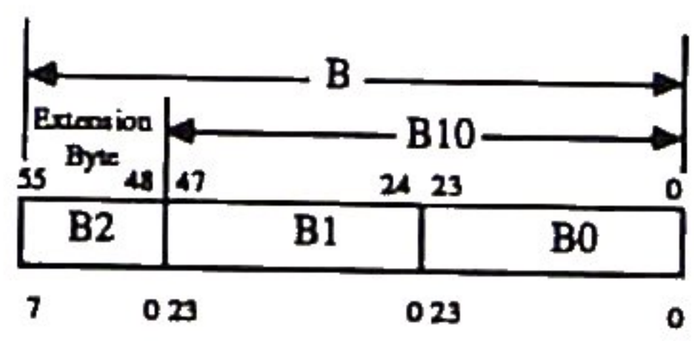
Data ALU Input Registers



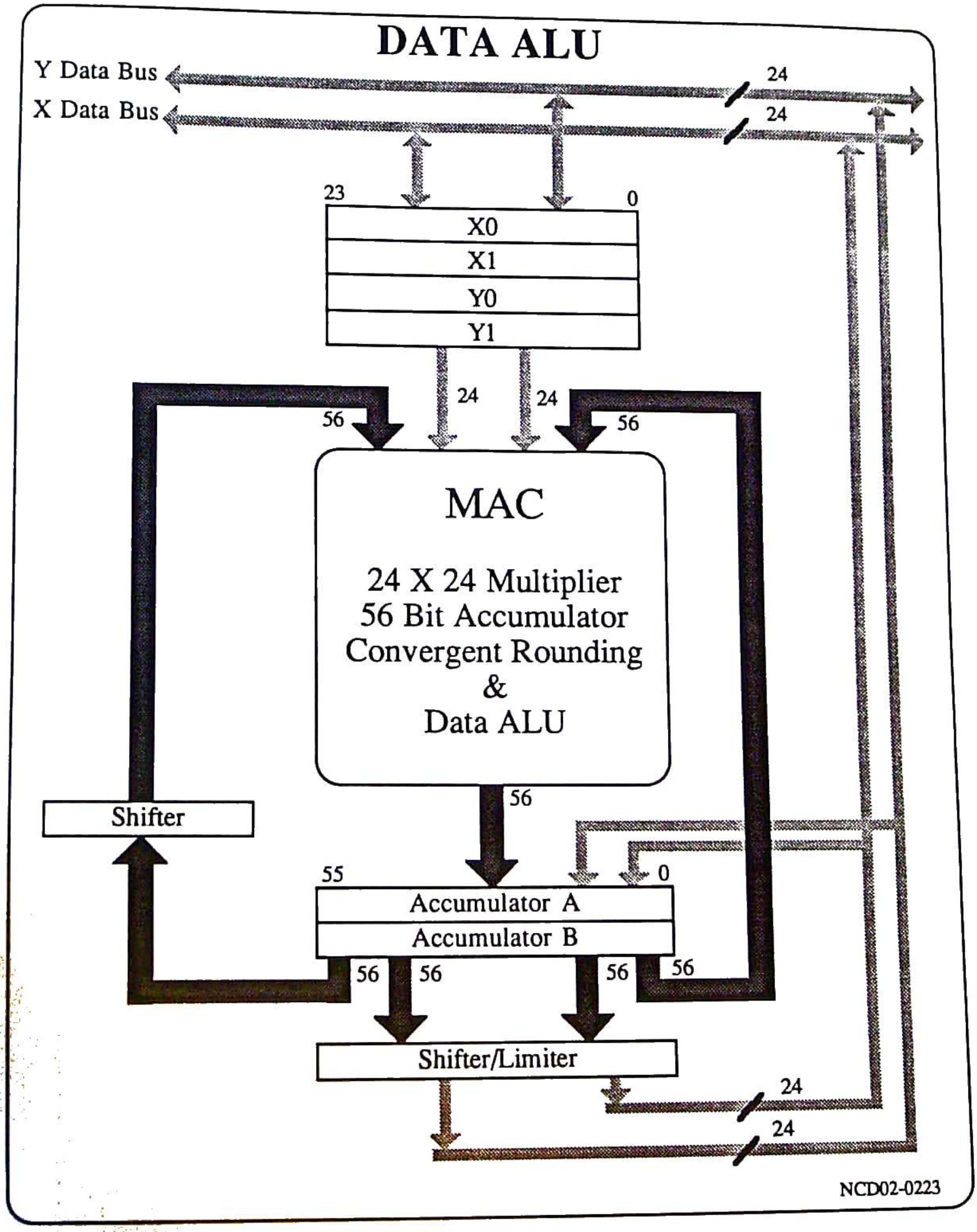


Data ALU Accumulators

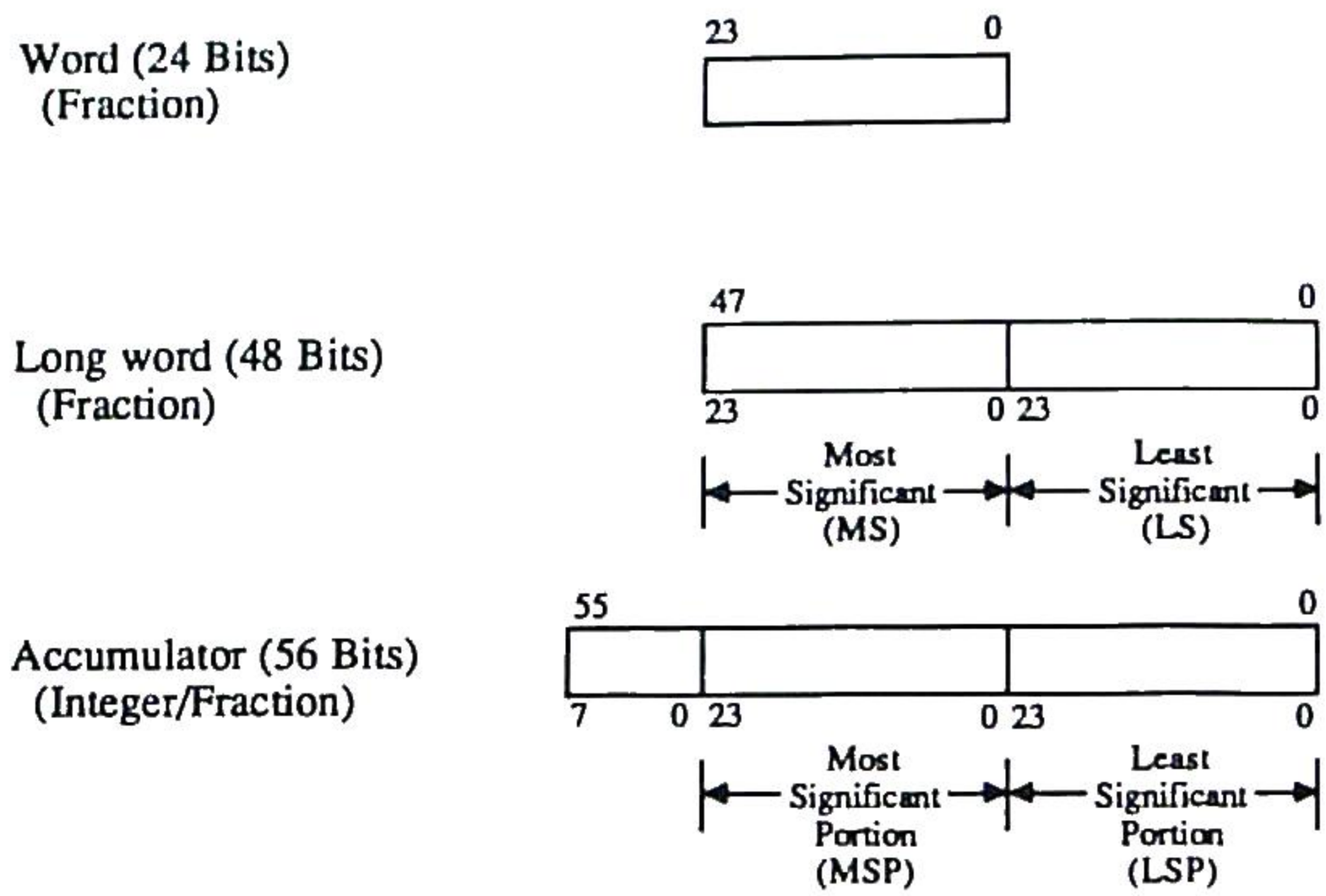




NCD02-0222

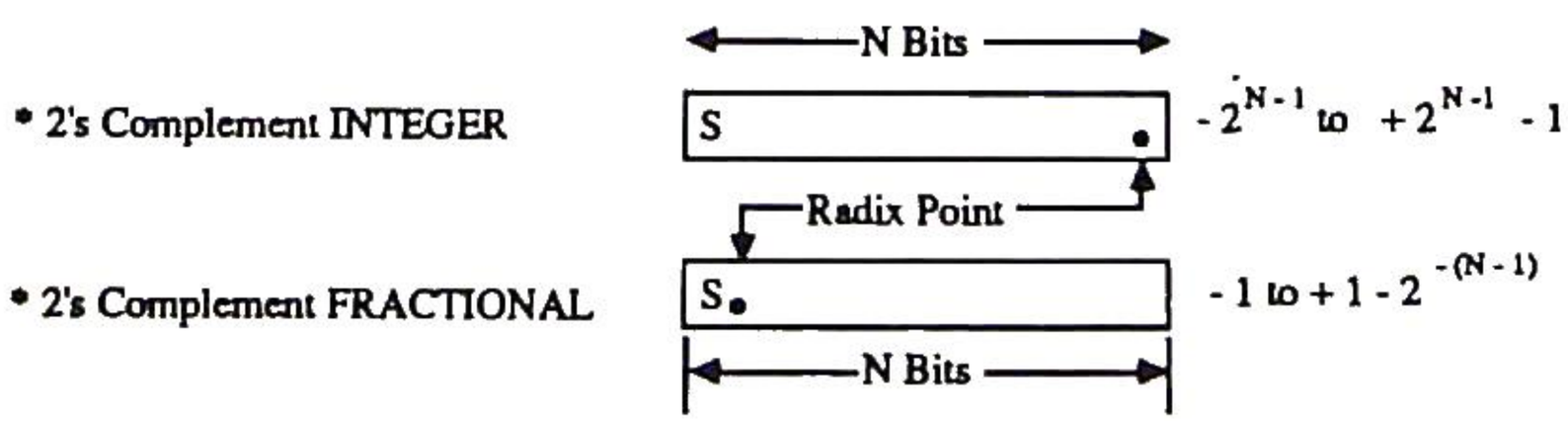

DATA ALU
Y Data Bus
X Data Bus
24
24
23
0
X0
X1
Y0
Y1
24
24
56
56
MAC
24 X 24 Multiplier
56 Bit Accumulator
Convergent Rounding
&
Data ALU
Shifter
56
55
0
Accumulator A
Accumulator B
56
56
56
56
Shifter/Limiter
24
24
NCD02-0223

OPERAND TYPES



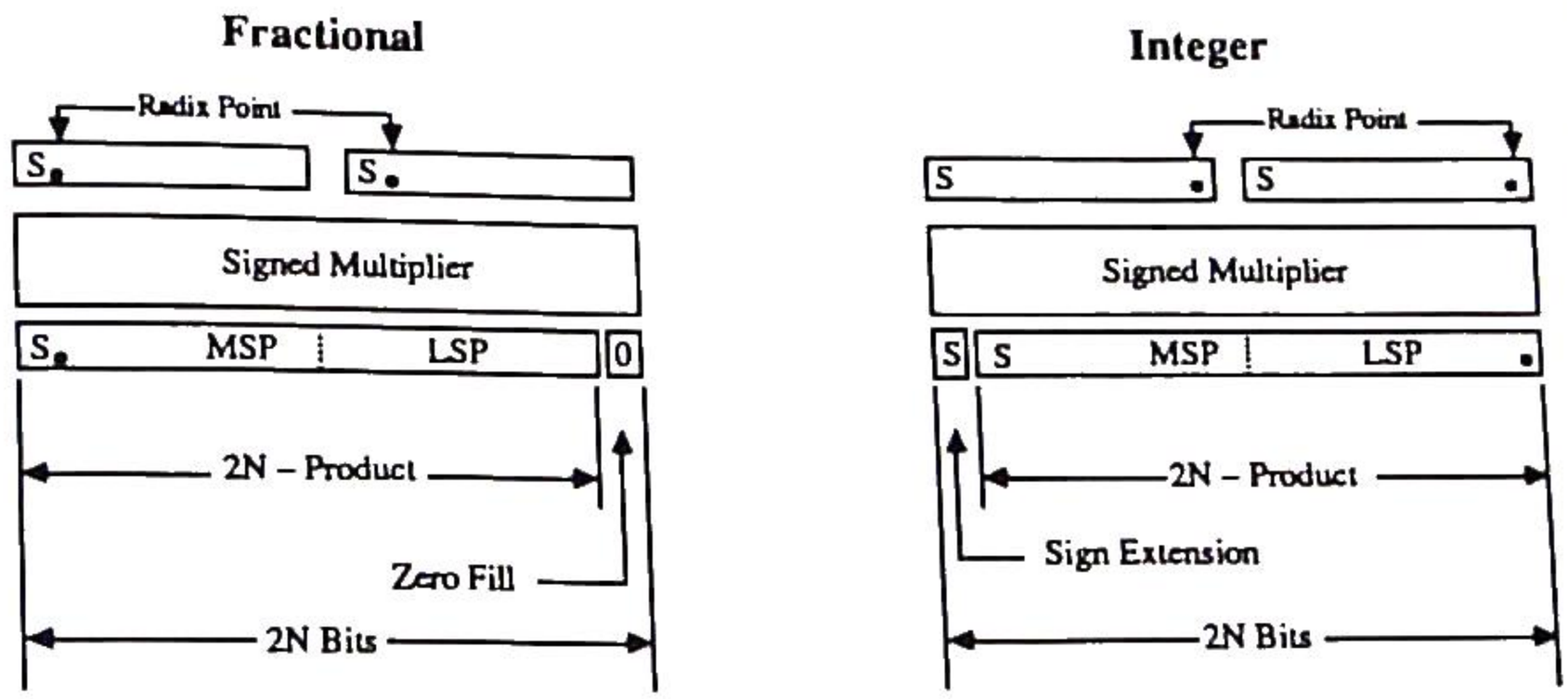
NCD02-0224

FRACTIONAL VS. INTEGER



NCD02-0225

FRACTIONAL VS. INTEGER MULTIPLICATION



NOTE: Fractional numbers tend to be self-normalizing.

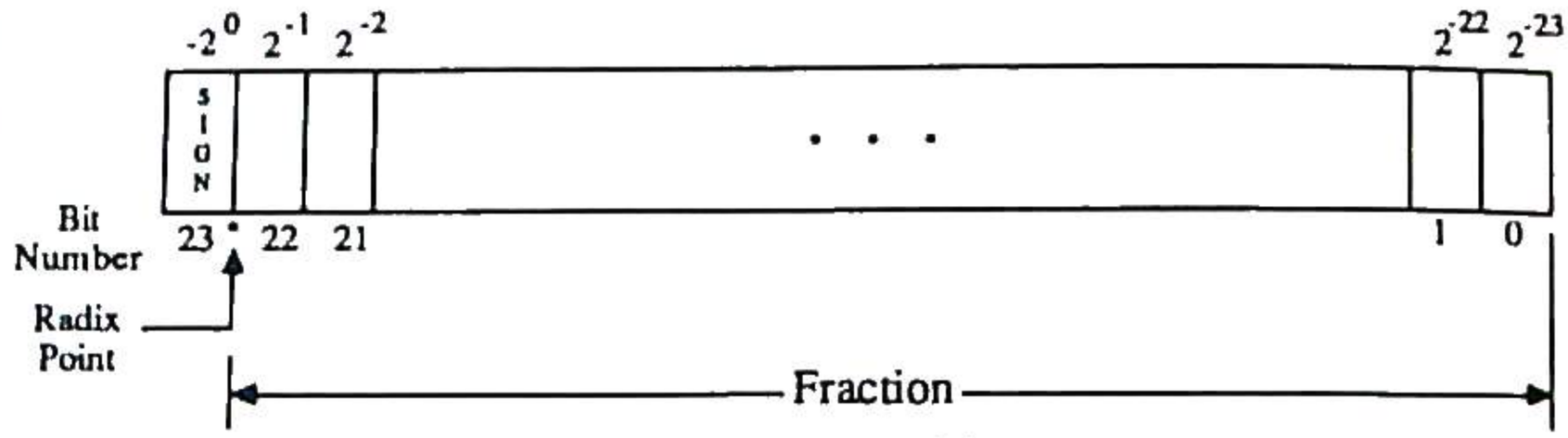
NCD02-0026-1

FRACTIONAL VS. 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

NCD02-0027

WORD OPERAND (24 BITS)



Largest Value

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

Smallest Value

-1 = $800000

Dynamic Range

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

Storage Locations

Program X and Y Memory
Registers X1, X0, Y1, Y0
Accumulators A1, A0, B1, B0

NCD02-0228

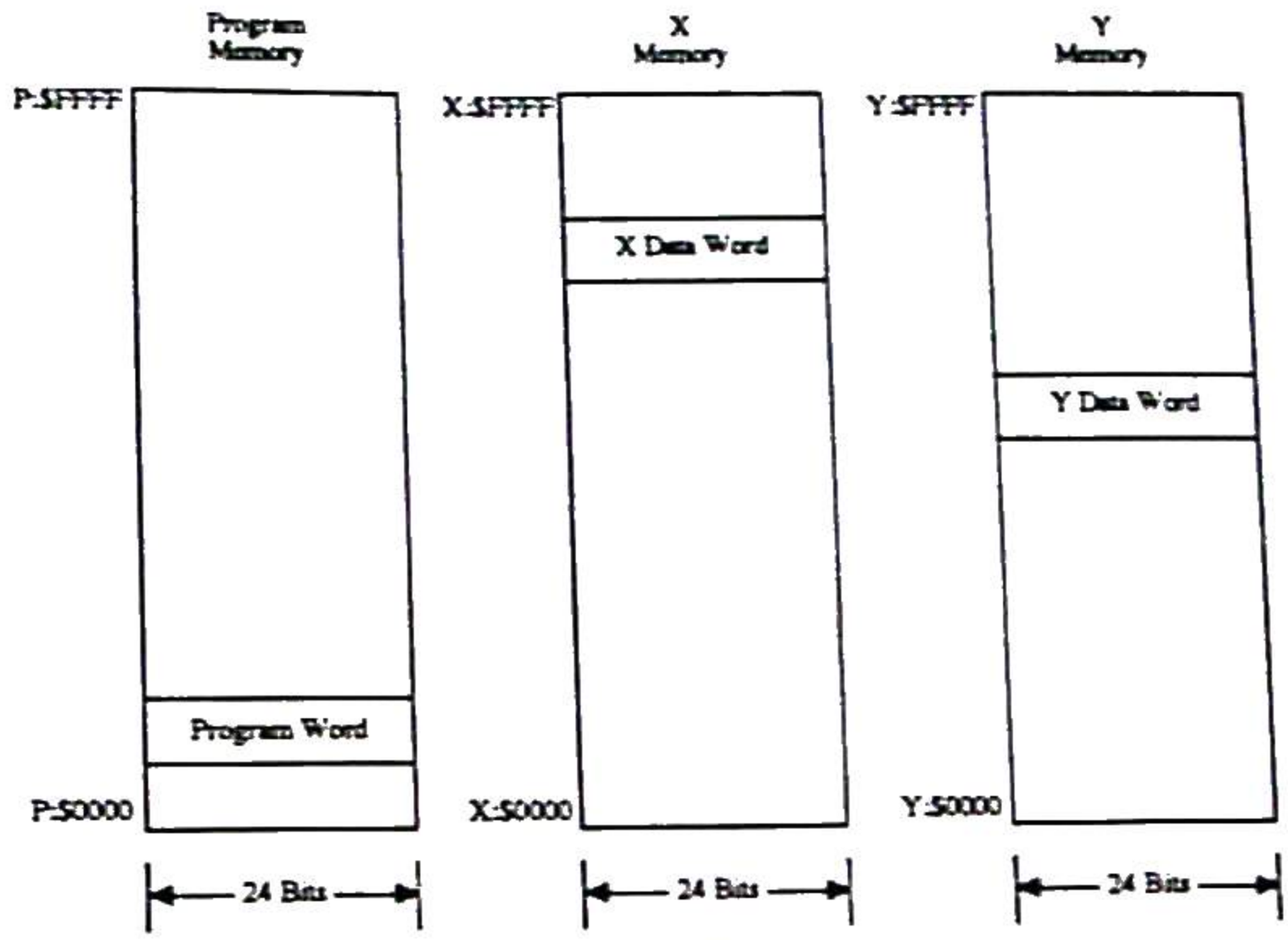
FRACTIONAL EXAMPLES

Decimal Fraction	24-Bit Binary Value	Hexadecimal Coefficient
0.750	0.11000000 00000000 00000000	$600000
0.500	0.10000000 00000000 00000000	$400000
0.250	0.01000000 00000000 00000000	$200000
0.125	0.00100000 00000000 00000000	$100000
-0.125	1.11100000 00000000 00000000	$F00000
-0.750	1.01000000 00000000 00000000	$A00000

Decimal Fraction	Hexadecimal Coefficient
0.00784313678741455	$010101
-0.00784313678741455	$FEFEFF

NCD02-0229-1

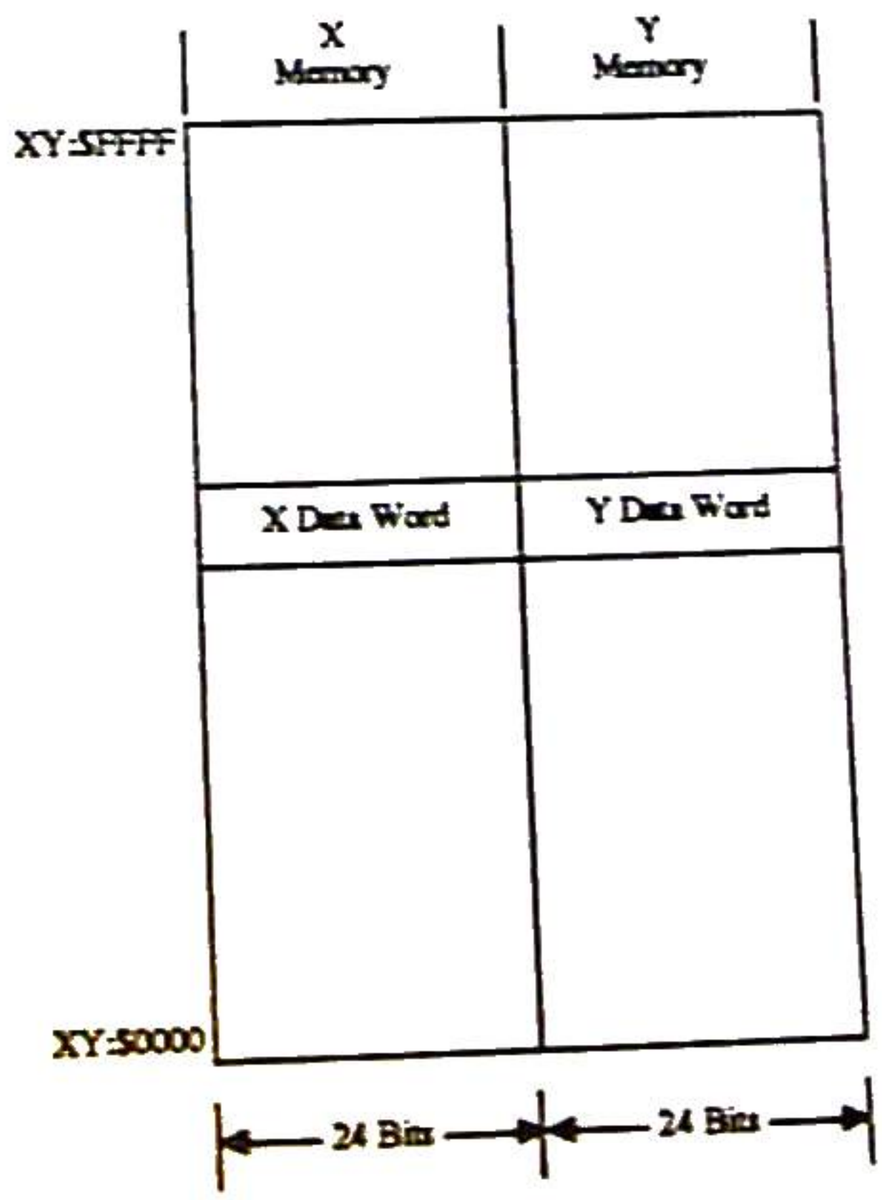
MEMORY TYPES



Three 64 K x 24 Bit Memory Spaces

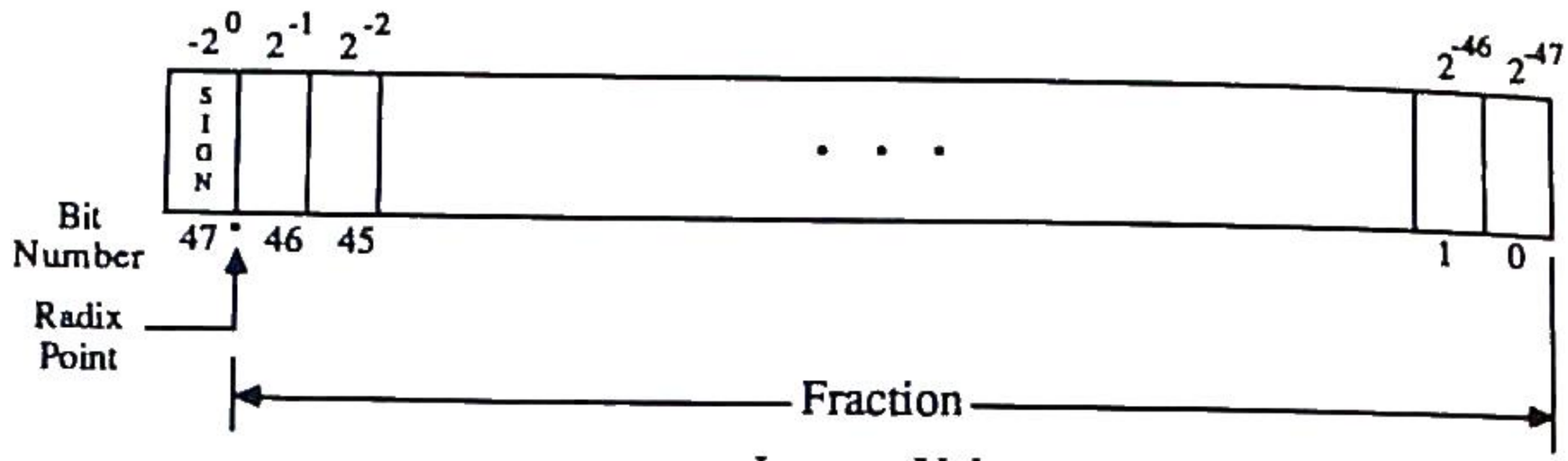
NCD02-0230

X Y MEMORY



NCD02-0231

LONG WORD OPERAND
(48 BITS)



Largest Value

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

Smallest Value

-1 = $800000000000

Dynamic Range

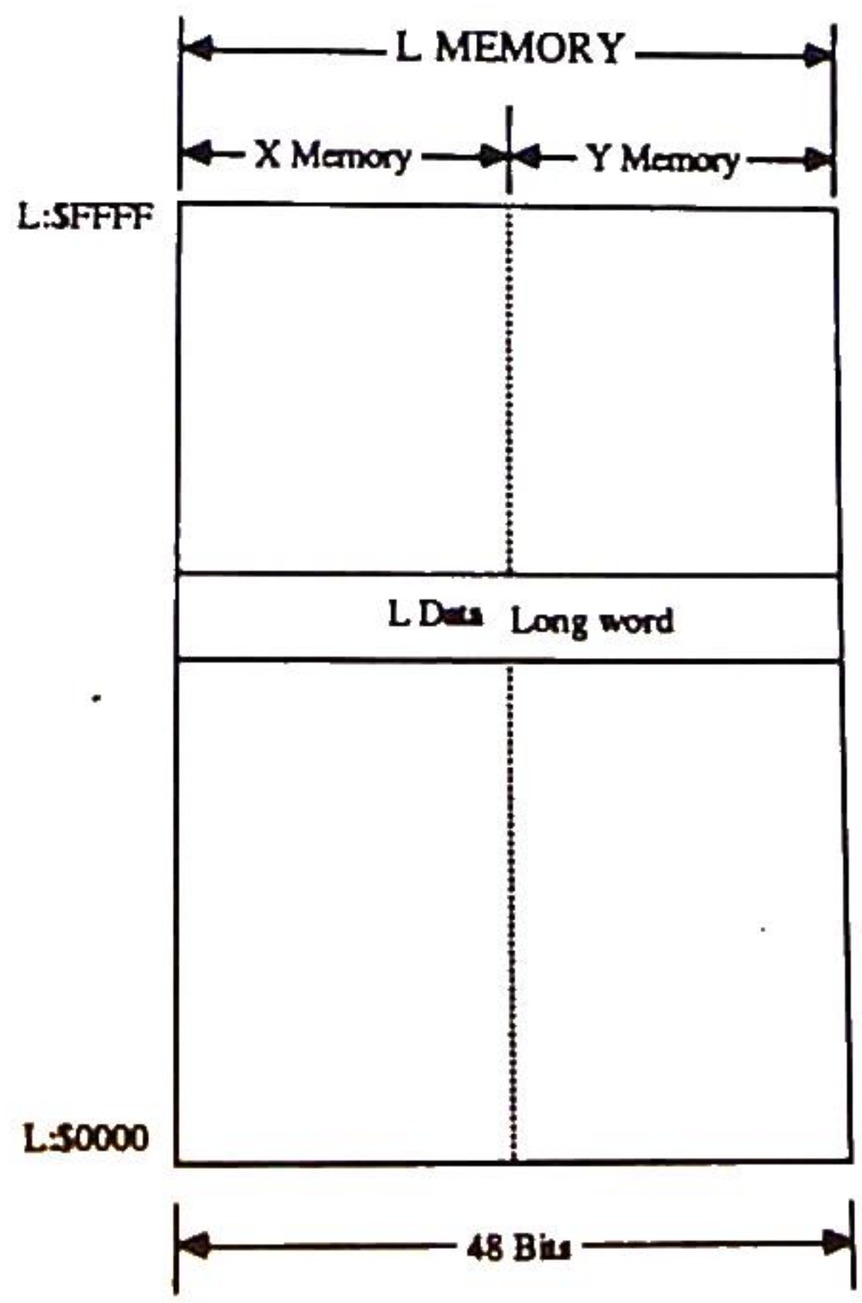
$289dB = 20\log_{10}(2^{48})$

Storage Locations

L Memory
Registers A10 and B10
Input registers X and Y

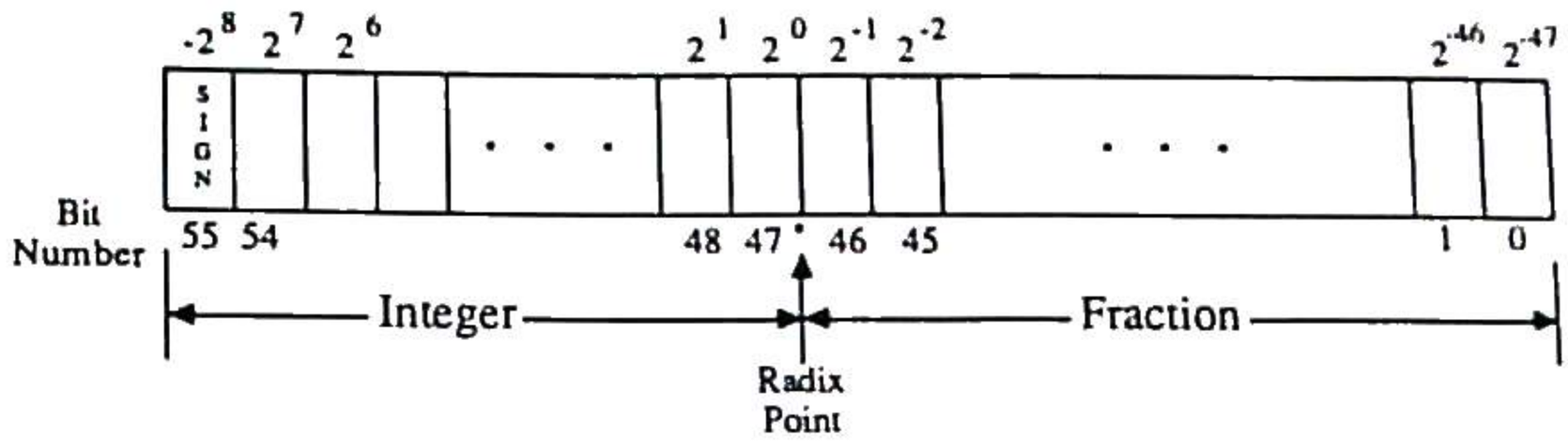
NCD02-0232-1

L MEMORY



NCD02-0233

ACCUMULATOR OPERAND (56 BITS)



Largest Value

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

Smallest Value

-256 = $80000000000000

Dynamic Range

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

Storage Locations

Accumulators A & B

NCD02-0234

ACCUMULATOR OPERAND EXAMPLES

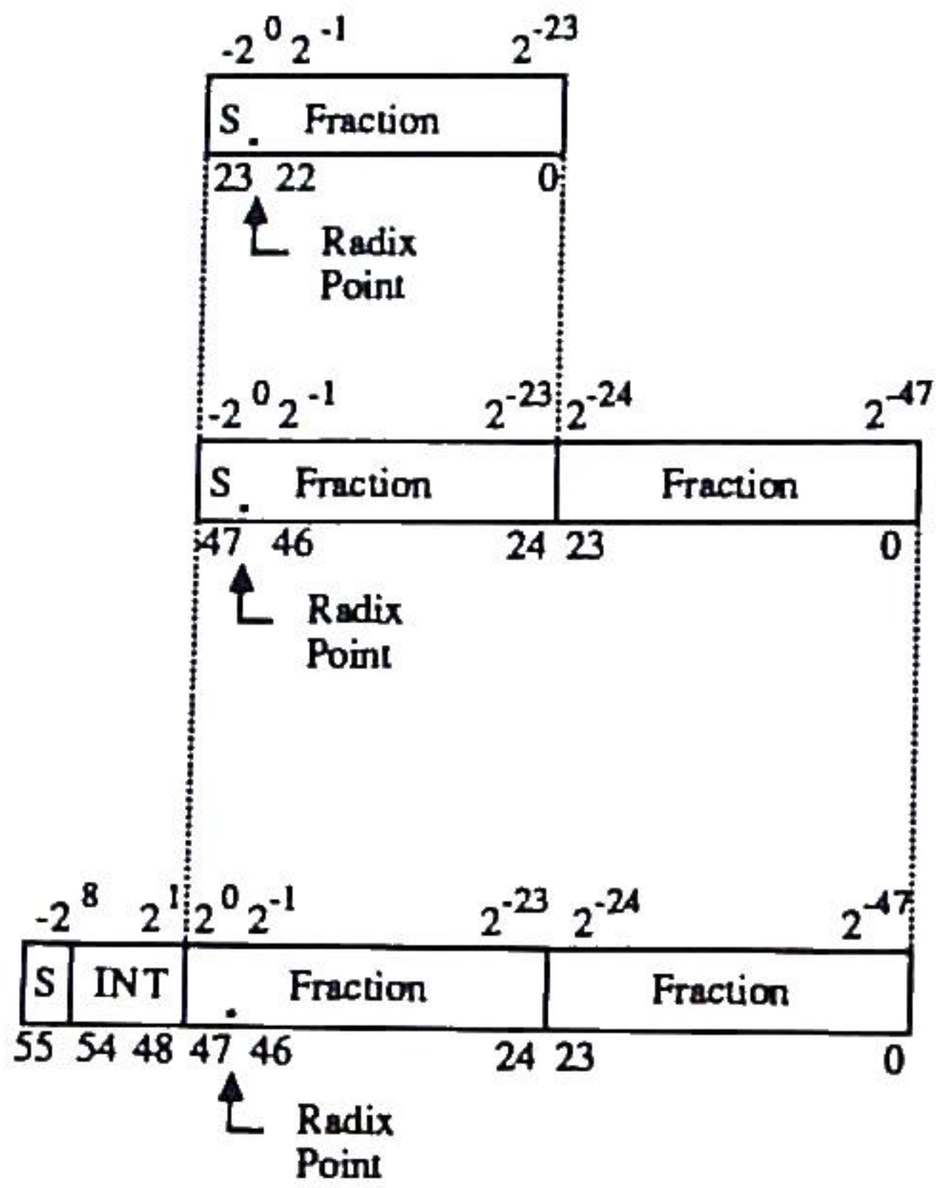
Decimal Fraction	56-Bit Binary Value	Hexadecimal Coefficient
12.5	000001100.1000000 • • • 0000	$06400000000000
1.5	000000001.1000000 • • • 0000	$00C00000000000
0.5	000000000.1000000 • • • 0000	$00400000000000
-0.5	111111111.1000000 • • • 0000	$FFC00000000000
-1.5	111111110.1000000 • • • 0000	$FF400000000000
-12.5	111110011.1000000 • • • 0000	$F9C00000000000

Decimal Fraction	Hexadecimal Coefficient
2.007843137254902	$01010101010101
-30.117647058823540	$F0F0F0F0F0F0F0

NOTE: RADIX Point Does not align on a hex Boundary

NCD02-0235-1

DATA TRANSFERS



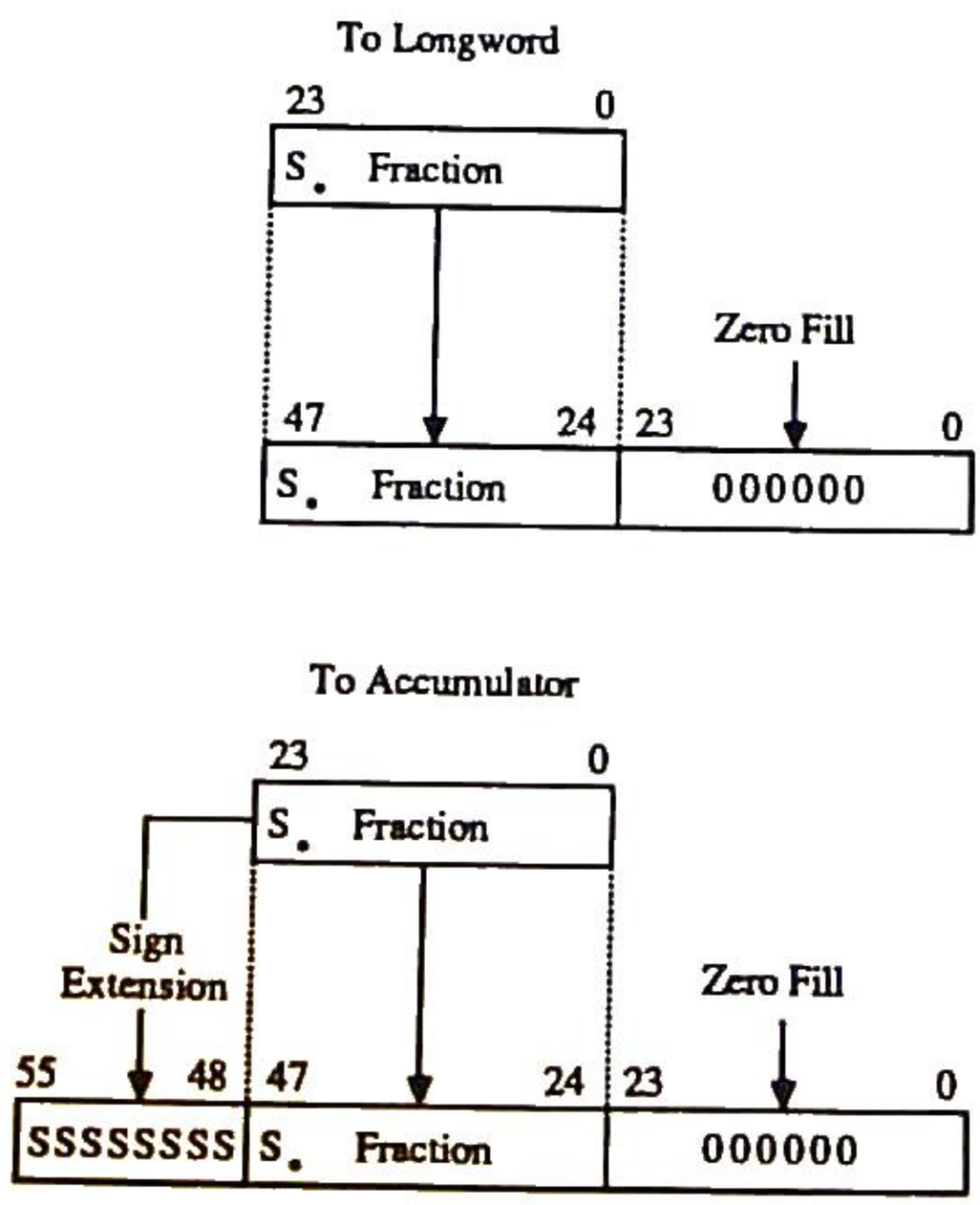
Word Operand
Program, X, or Y Memory
X1,X0
Y1,Y0
A1,A0
B1,B0

Long Word Operand
L Memory
X = X1:X0
Y = Y1:Y0
A10 = A1:A0
B10 = B1:B0

Accumulator Operand
Accumulators
A,B

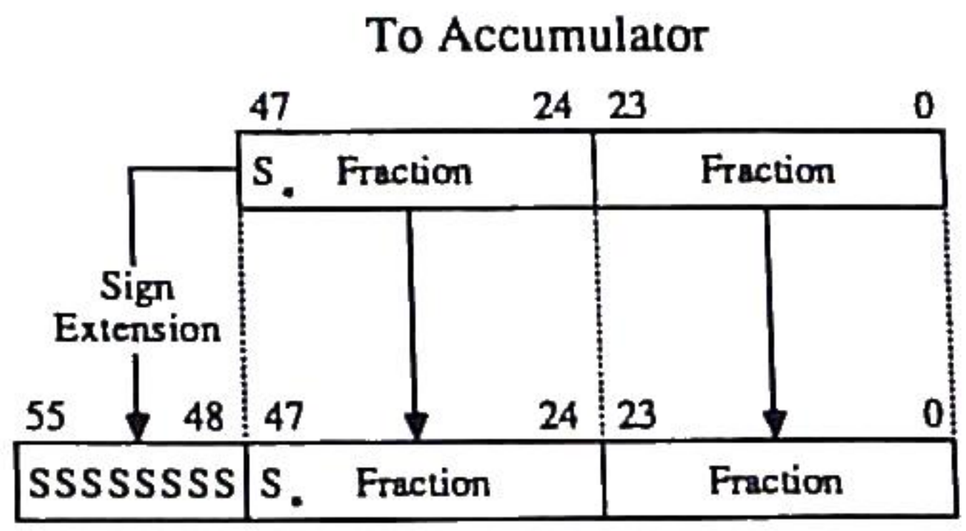
NCD02-0236-1

WORD TRANSFERS



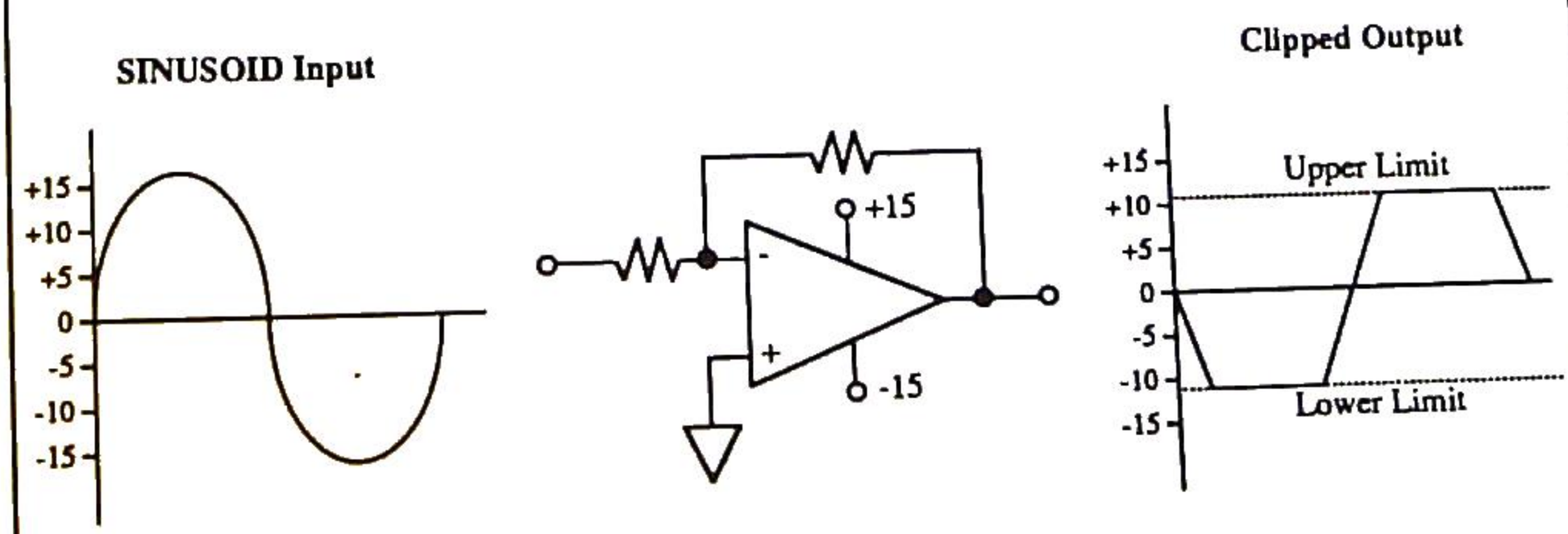
NCD02-0237

LONG WORD TRANSFERS



NCD02-0238

HITTING THE RAIL

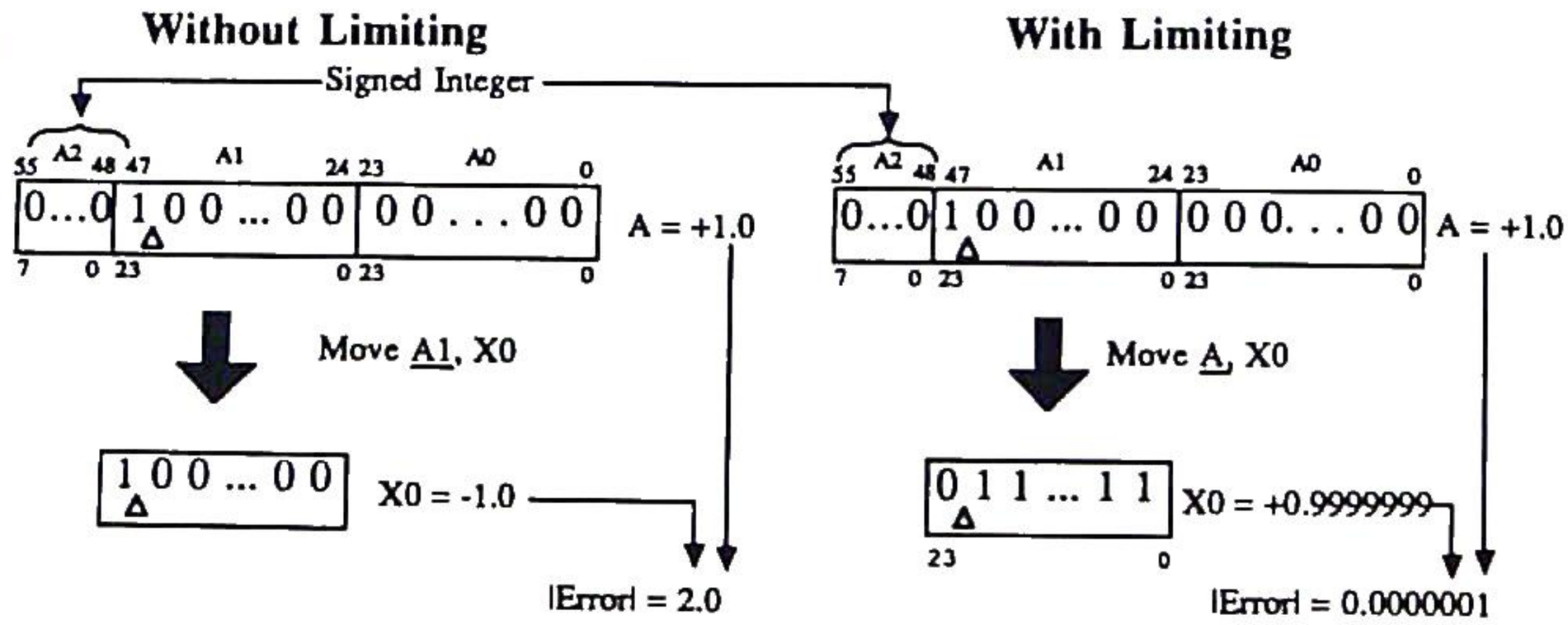


The OP-AMP has been driven into

"SATURATION"

NCD02-0239

SATURATION ARITHMETIC (LIMITING)



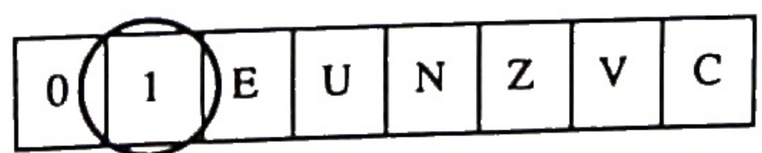
- Minimizes overflow errors when reading Accumulator Data.
- Limiting occurs automatically when transferring from Accumulator A or B into 24 or 48 bit destination.
- Latched in Condition Code Register.
- Contents of A or B are not changed.

NCD02-0240

LIMITING CASES

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

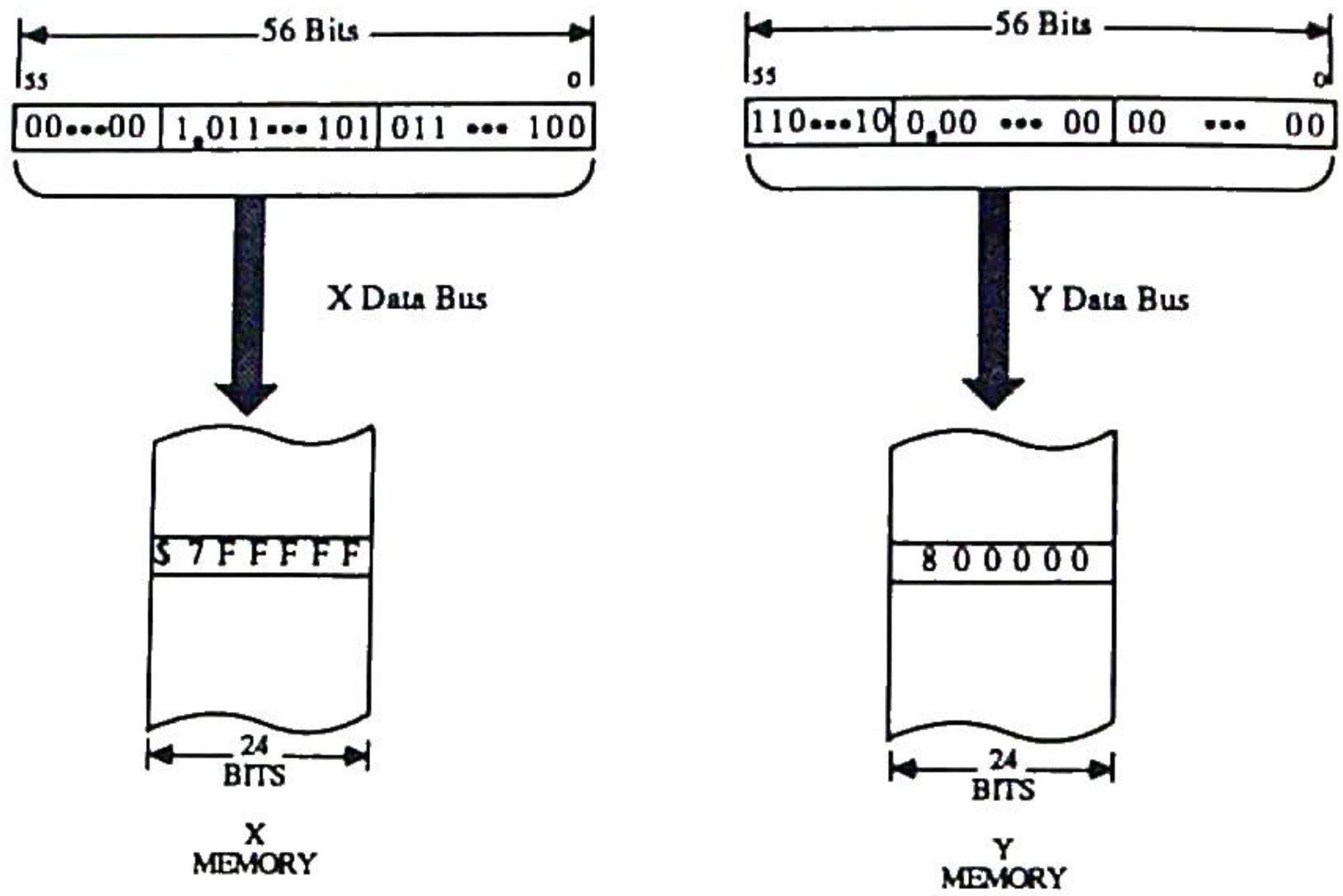
Condition Code Register



If Limiting or Overflow occurs then
Limit Bit is set and latched.
Must be cleared by software.

NCD02-0241-1

X AND Y LIMITING



NCD02-0242-1

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

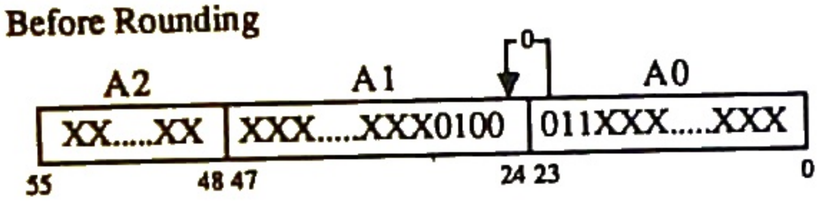
The Solution: Convergent Rounding or Round to Nearest Even.

NCD02-0243

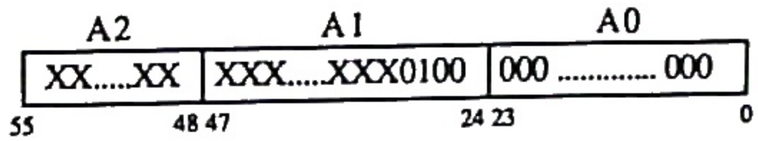
CONVERGENT ROUNDING

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

Before Rounding

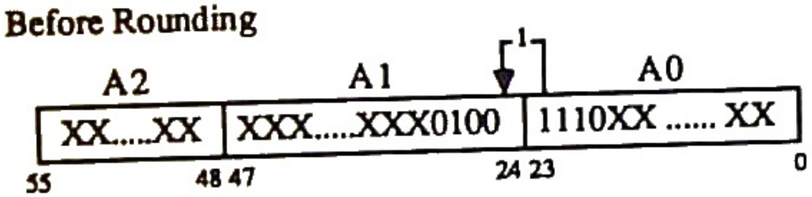


After Rounding

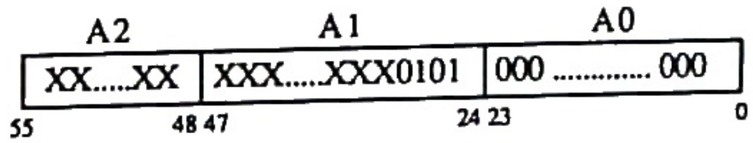


Case II: If A0 > $800000 (1/2), then round up (add 1 to A1)

Before Rounding

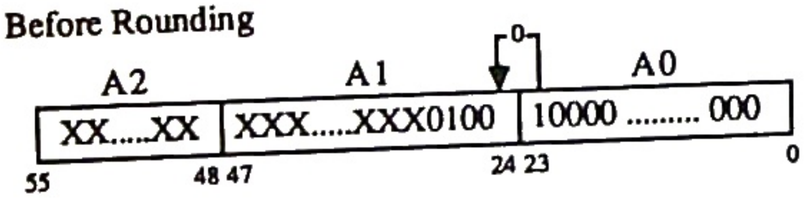


After Rounding

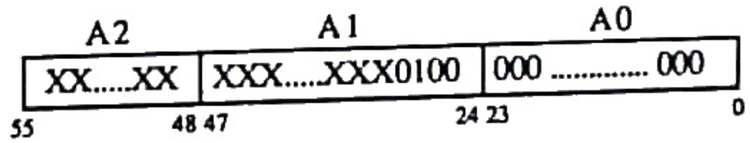


Case III: If A0 = $800000 (1/2), and the least significant bit of A1 = 0, then round down (add nothing)

Before Rounding

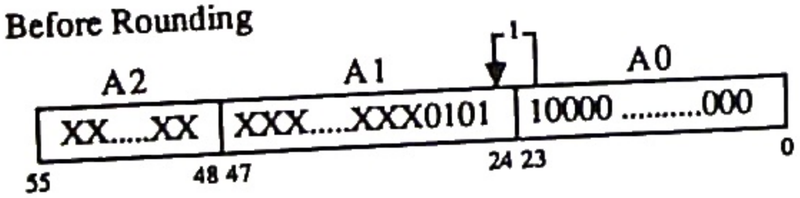


After Rounding

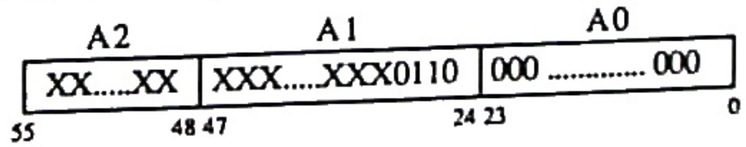


Case IV: If A0 = $8000000 (1/2), and the least significant bit of A1 = 1, then round up (add one)

Before Rounding



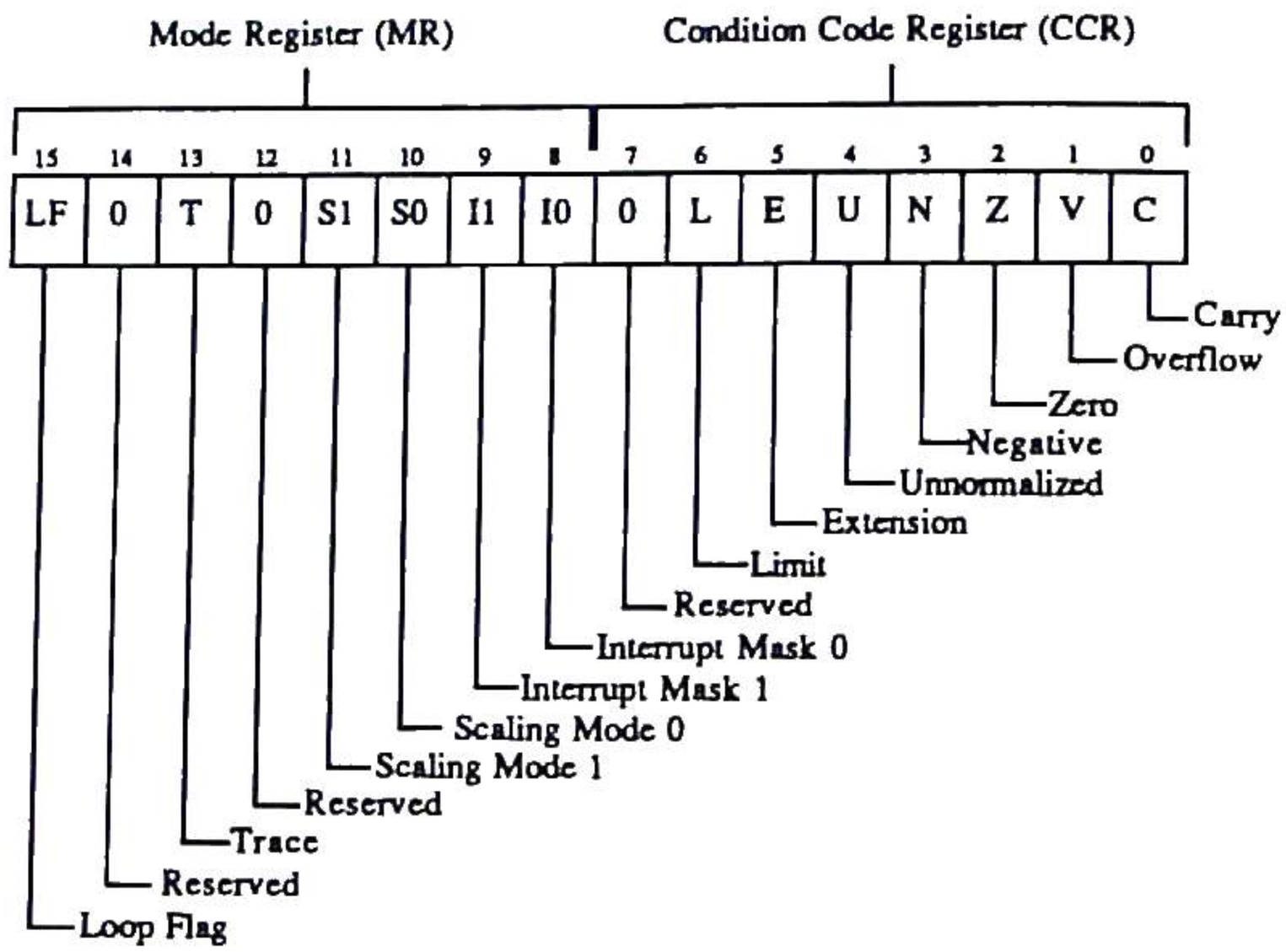
After Rounding



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

NCD02-0244

STATUS REGISTER

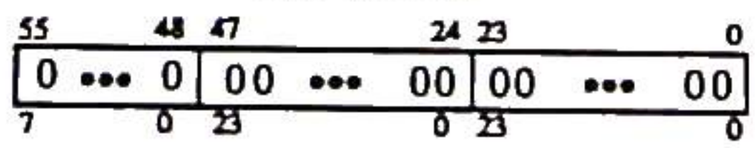


NCD02-0219

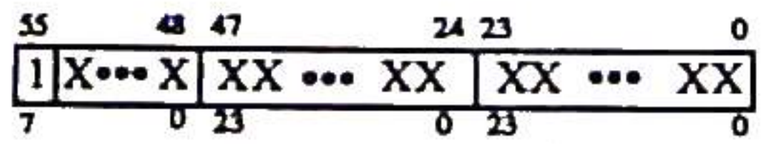
Z, N, AND C CONDITION CODE BITS

A or B Accumulator

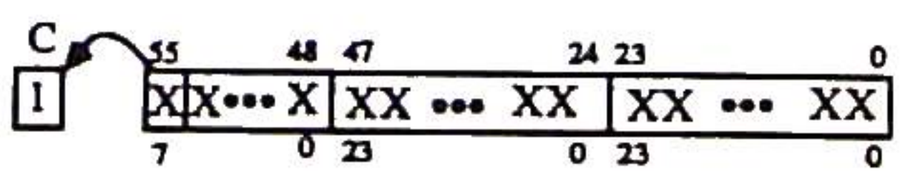
- Zero — Z = 1



- Negative — N = 1



- Carry (Borrow) — C = 1



- Z, N, and C are cleared otherwise

NCD02-0246

V – OVERFLOW BIT

- The Result Cannot be Represented in 56 Bits

$00	000000	000001

+

$7F	FFFFFF	FFFFFF

=

$80	000000	000000

- OVERFLOW V = 1

- OVERFLOW also Sets and Latches the Limit "L" Bit.

NCD02-0247

E – EXTENSION BIT

- E = 0 If all the Bits of the Signed Integer Portion are the same.
  The number (n) is a fraction: $+1 > n \geq -1$

Case I

55 … 48	47 … 24	23 … 0
11111111	1.XX • • •XX	XX • • • XX

Radix Point

Case II

55 … 48	47 … 24	23 … 0
00000000	0.XX • • • XX	XX • • • XX

- E = 1 For all other cases

  The number (n) is an Integer plus Fraction:

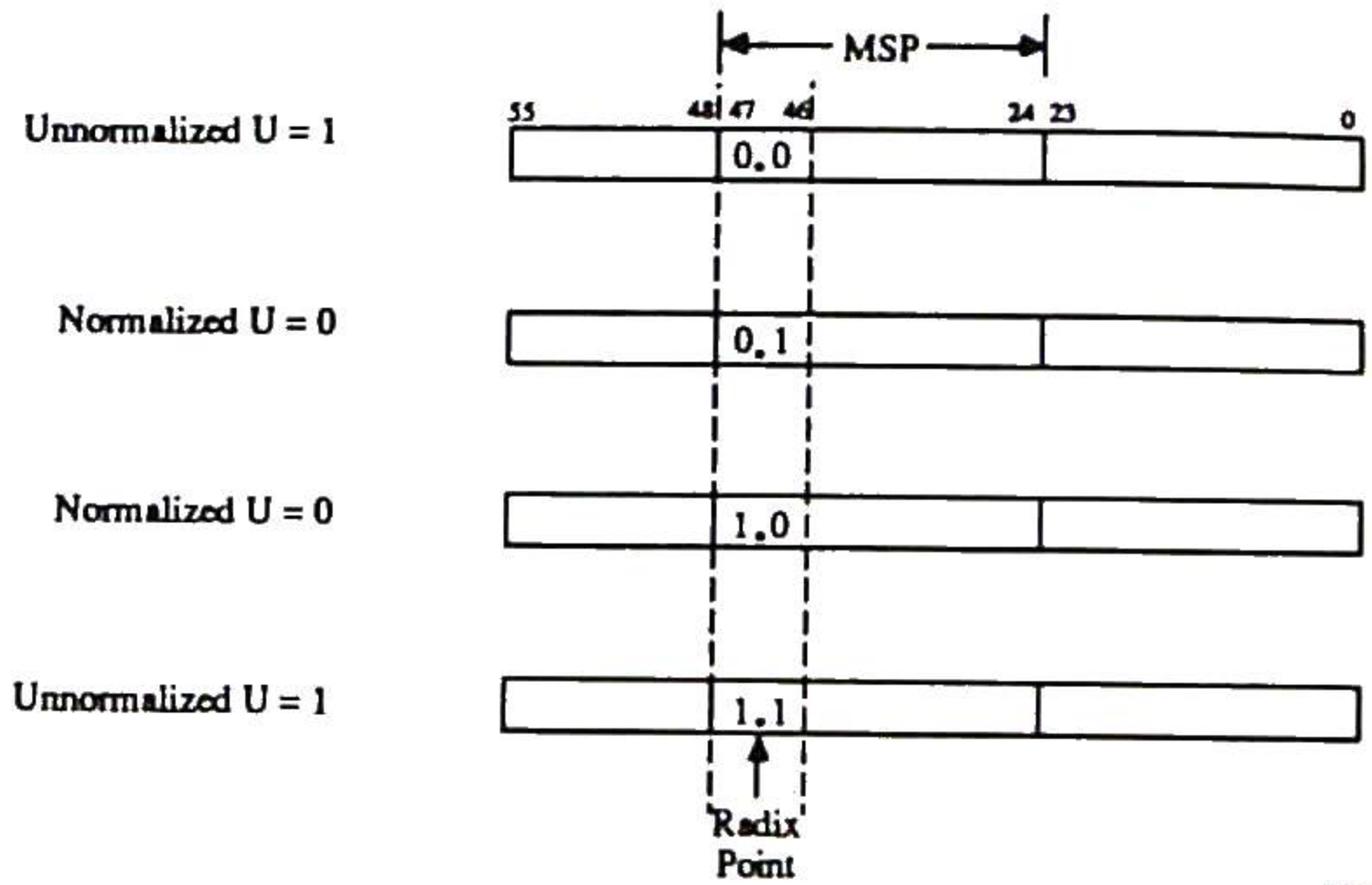
  $+256 > n \geq -256$

NCD02-0248

U – UNNORMALIZED BIT

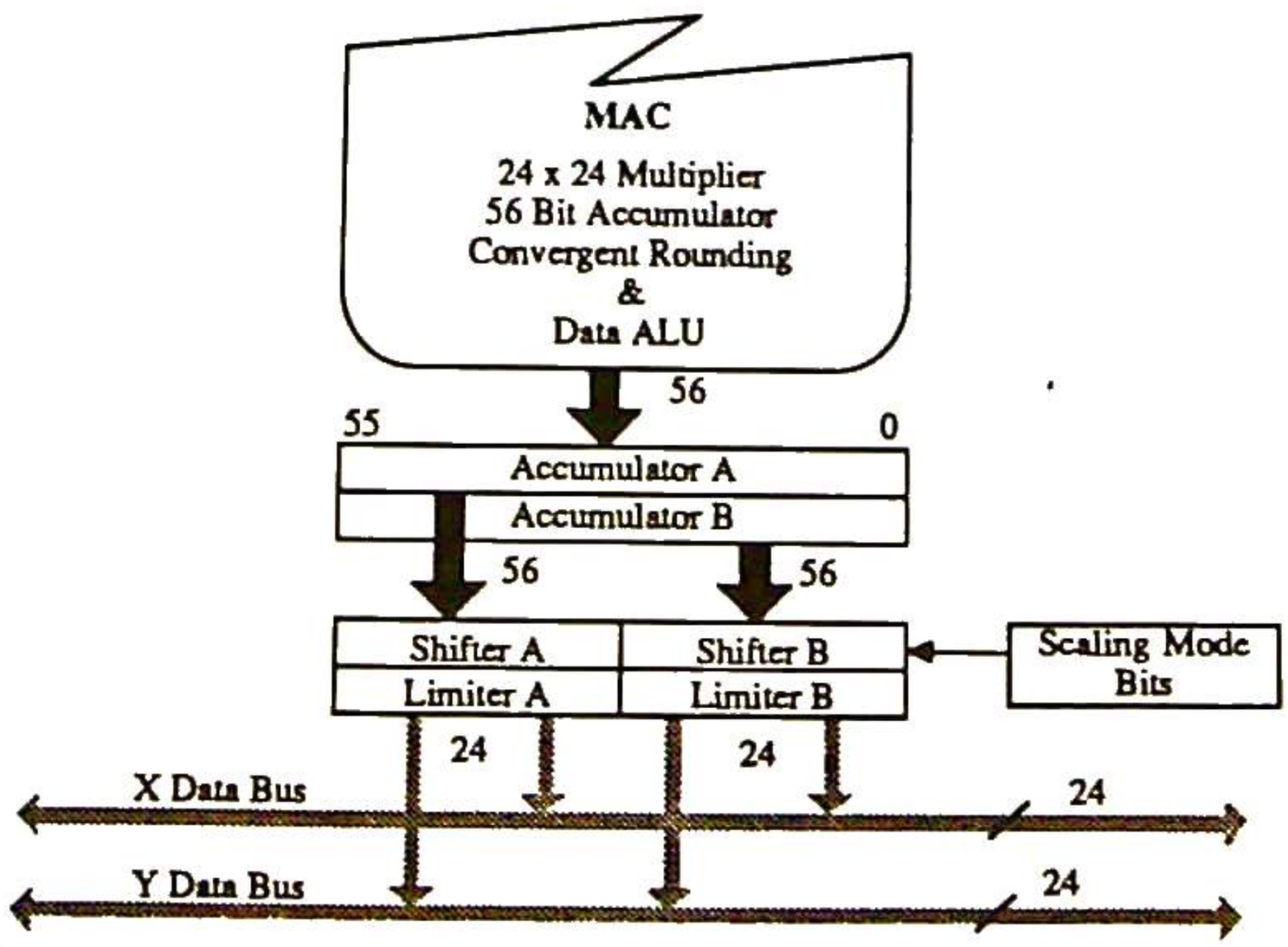
- U = 1 If the two most significant bits of the MSP are the same.

Cleared otherwise



NCD02-0249

SHIFTER / LIMITERS

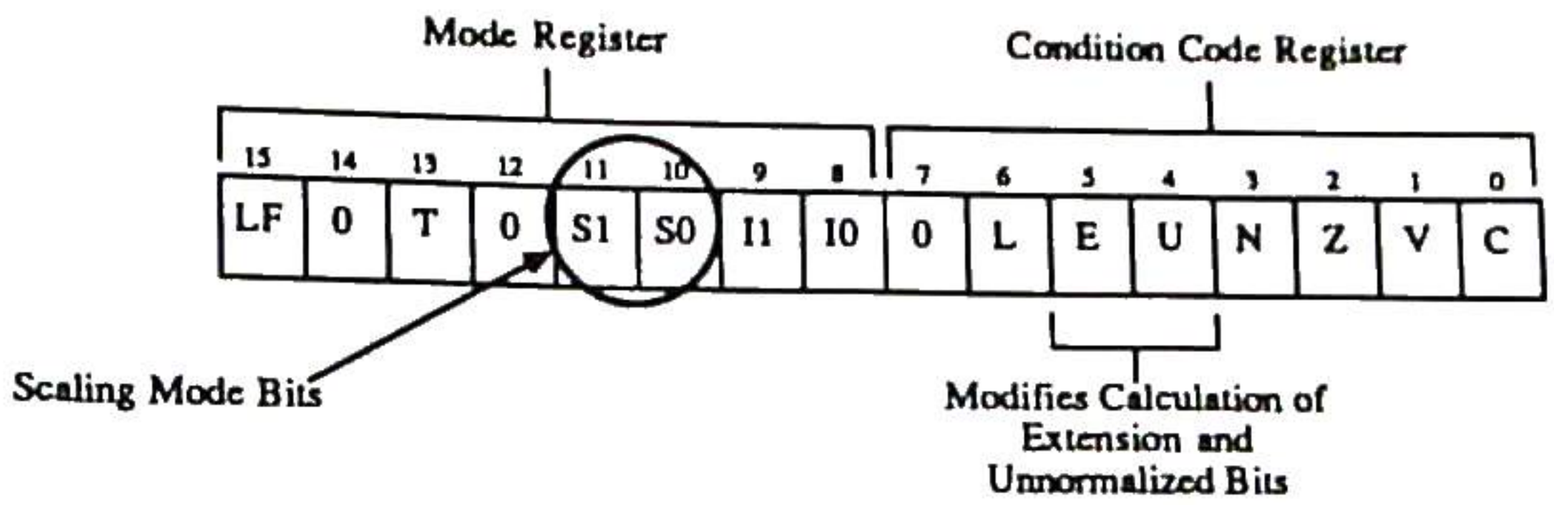


- SHIFTER Allows Dynamic Scaling without Changing Program Code.
- LIMITING Follows the Scaling Function.
- SHIFT/LIMITER Affects 24 or 48 Bit Data Read from A or B Accumulators onto the X or Y Data Buses
- Contents of Accumulators A or B are Not Changed.

NCD02-0250

SCALING MODE BITS

Status Register

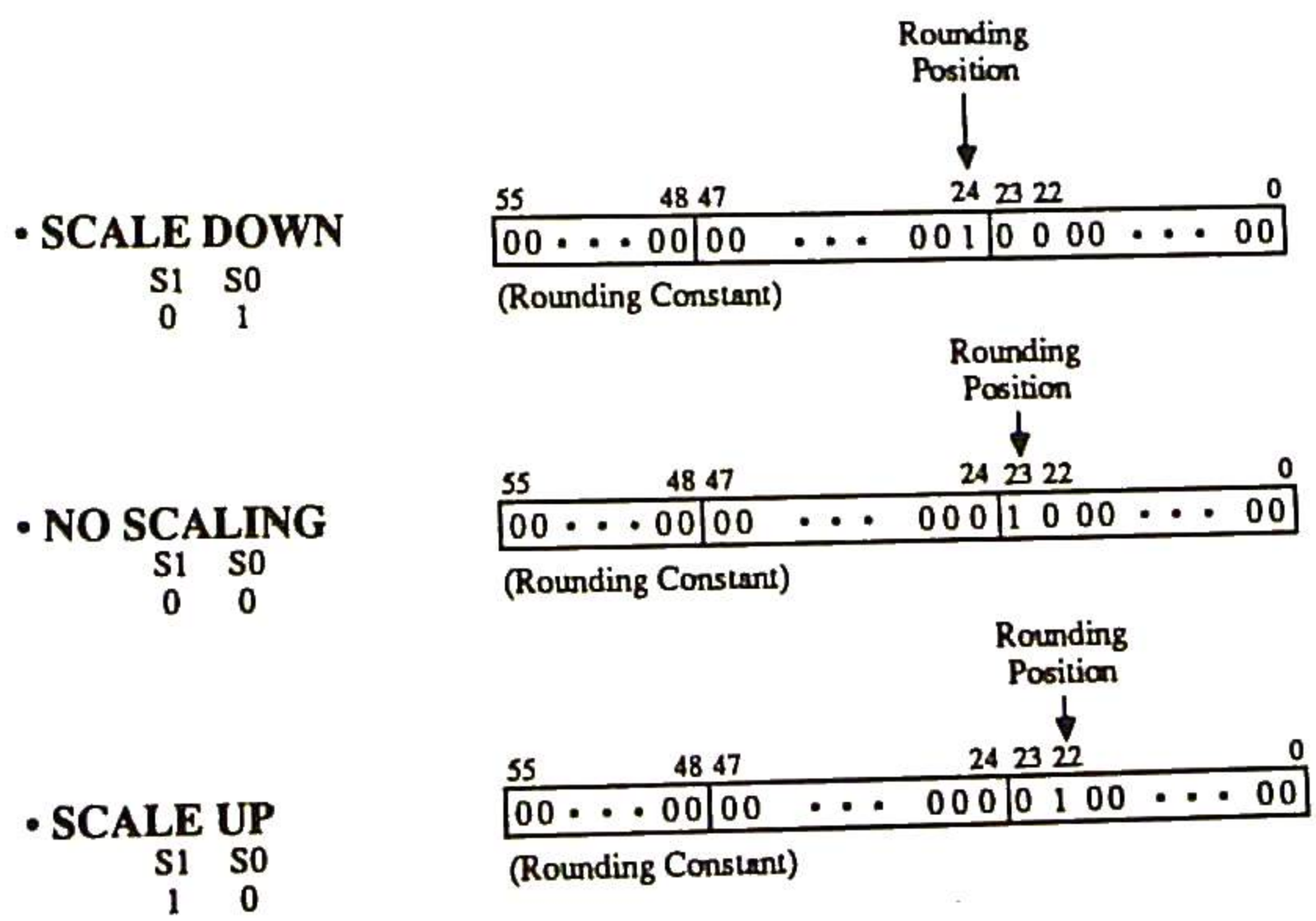


S1	S0	Rounding bit	Scaling Mode
0	0	23	No Scaling
0	1	24	Scale Down (1 bit arithmetic right shift)
1	0	22	Scale Up (1 bit arithmetic left shift)
1	1	--	Reserved for future expansion

- Scaling bits are cleared during reset and long interrupt service routine.
- Affects Convergent Rounding

NCD02-0251

SCALING AND CONVERGENT ROUNDING

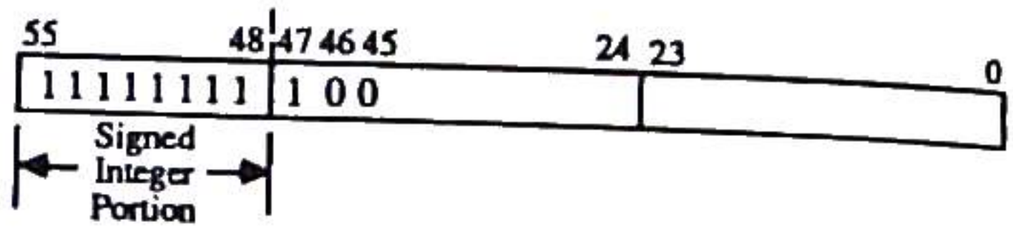


NCD02-0252

SCALING AND EXTENSION BIT

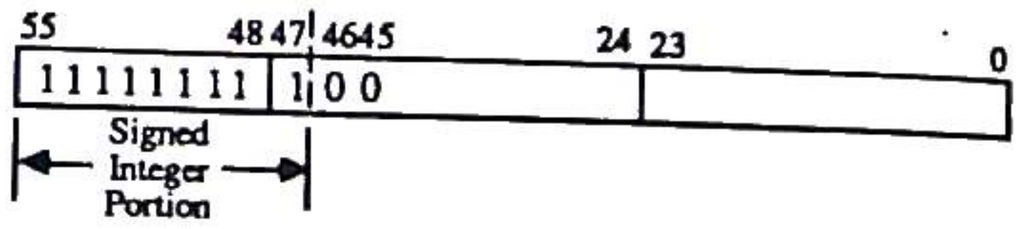
- **SCALE DOWN**

  S1 S0
  0 1
  E = 0



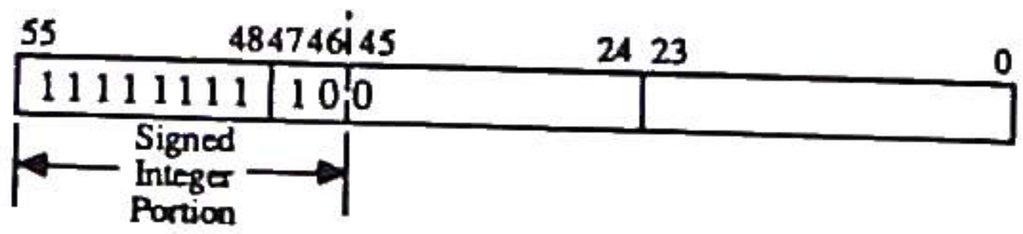
- **NO SCALING**

  S1 S0
  0 0
  E = 0



- **SCALE UP**

  S1 S0
  1 0
  E = 1



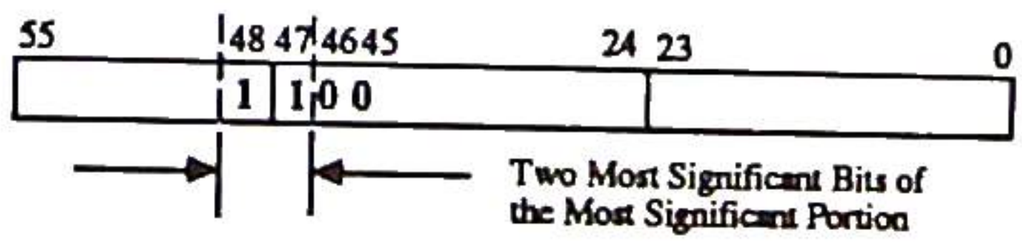
- E = 0 If all the Bits in the Signed Integer Portion are the Same
  else E = 1

NCD02-0253

SCALING AND UNNORMALIZE BIT

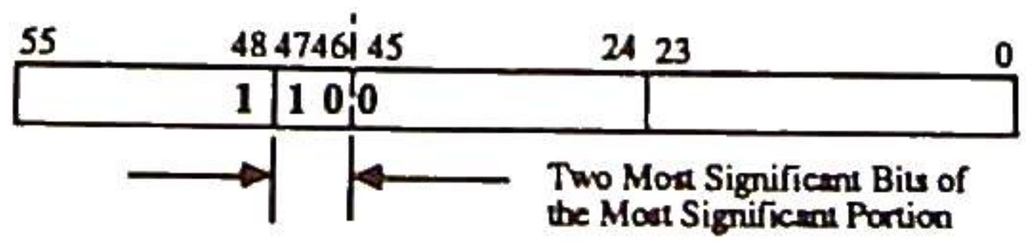
- **SCALE DOWN**

  S1 S0
  0 1
  U = 1



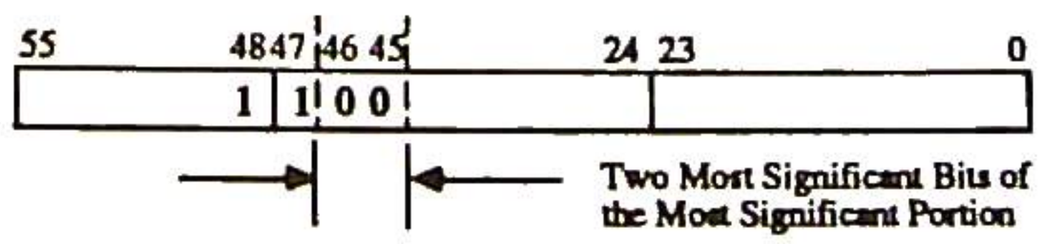
- **NO SCALING**

  S1 S0
  0 0
  U = 0



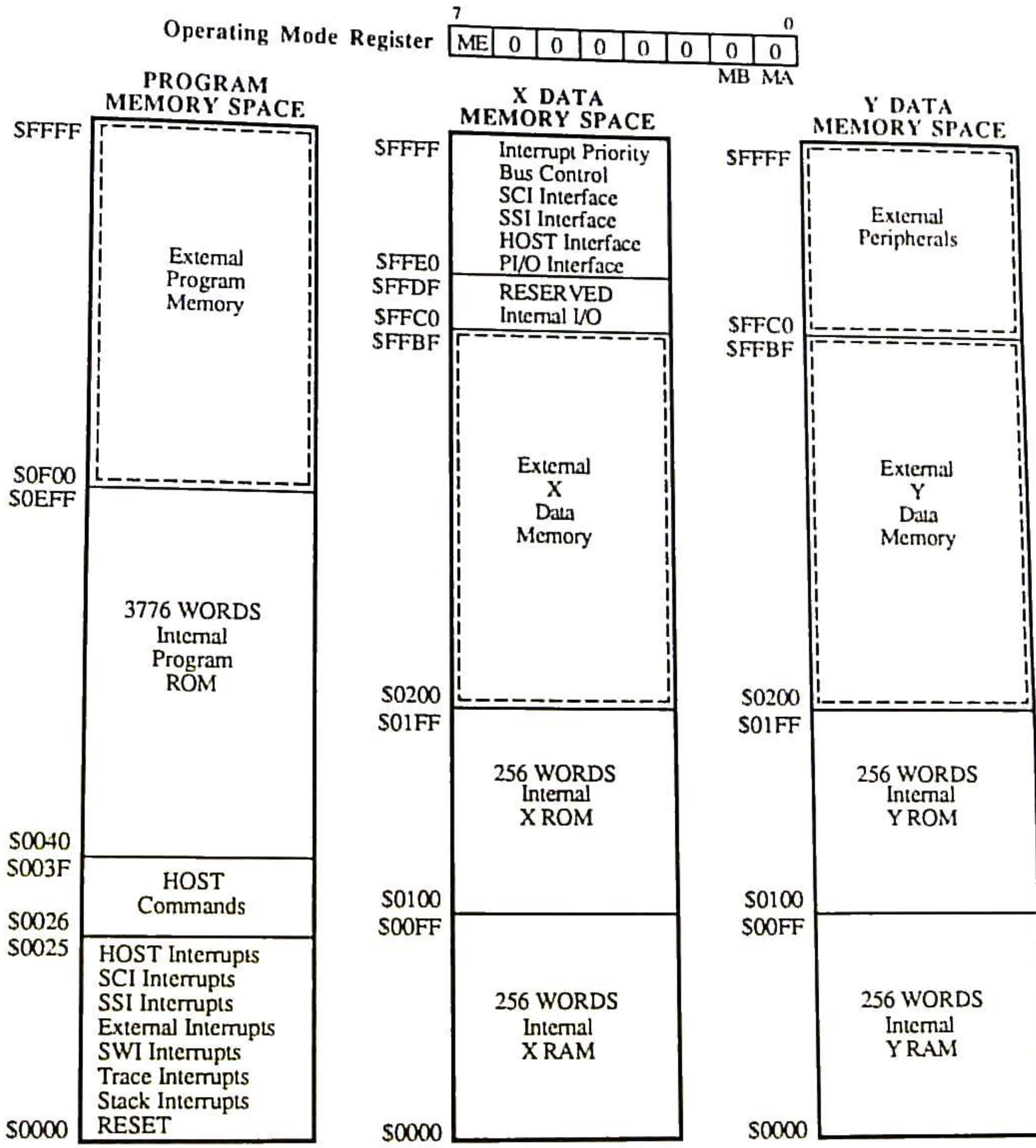
- **SCALE UP**

  S1 S0
  1 0
  U = 1



NCD02-0254-1

DSP56000 MEMORY MAP MODE 0 - NON-EXPANDED AND INTERNAL RESET

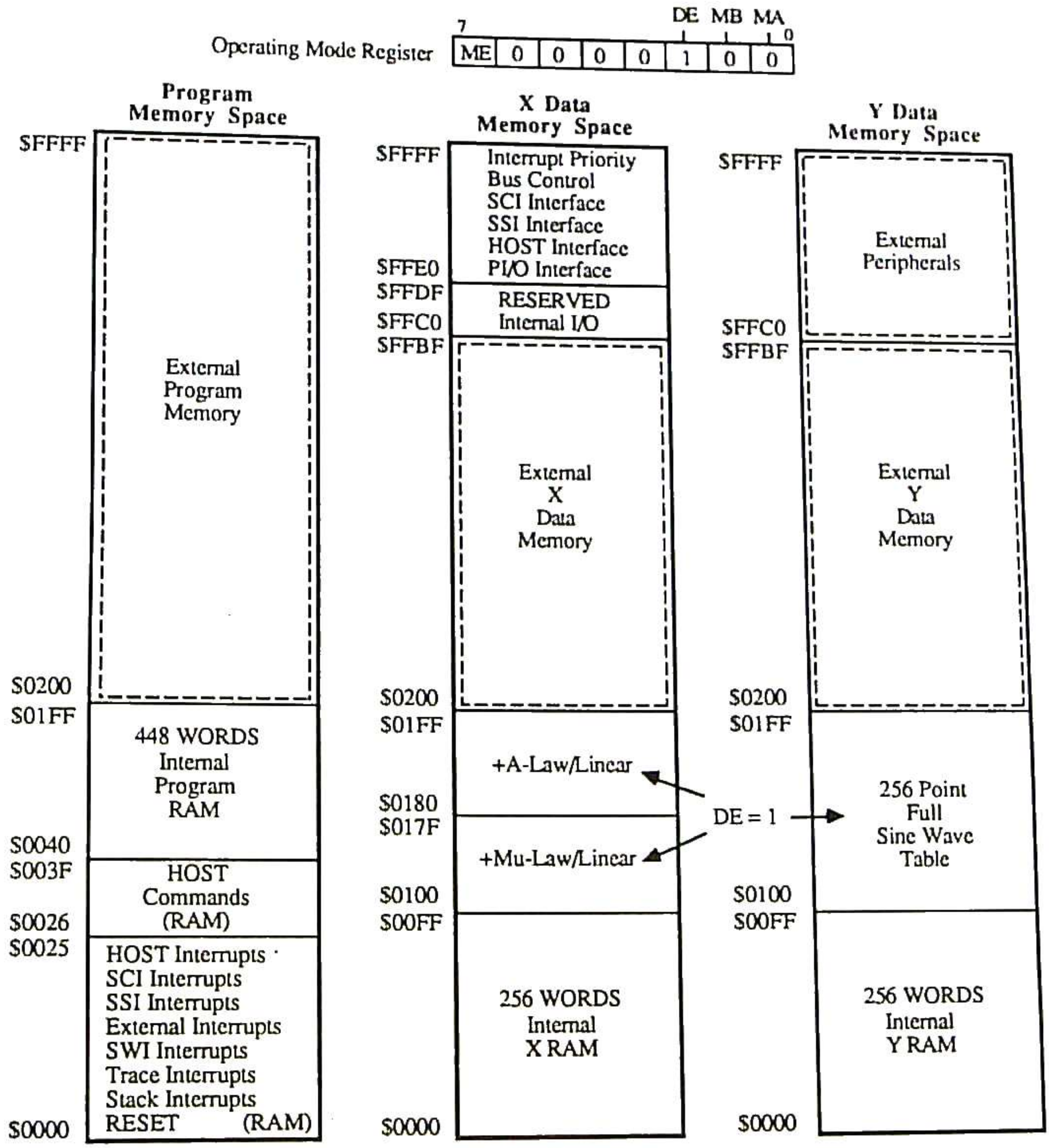


DSP56000 MODES

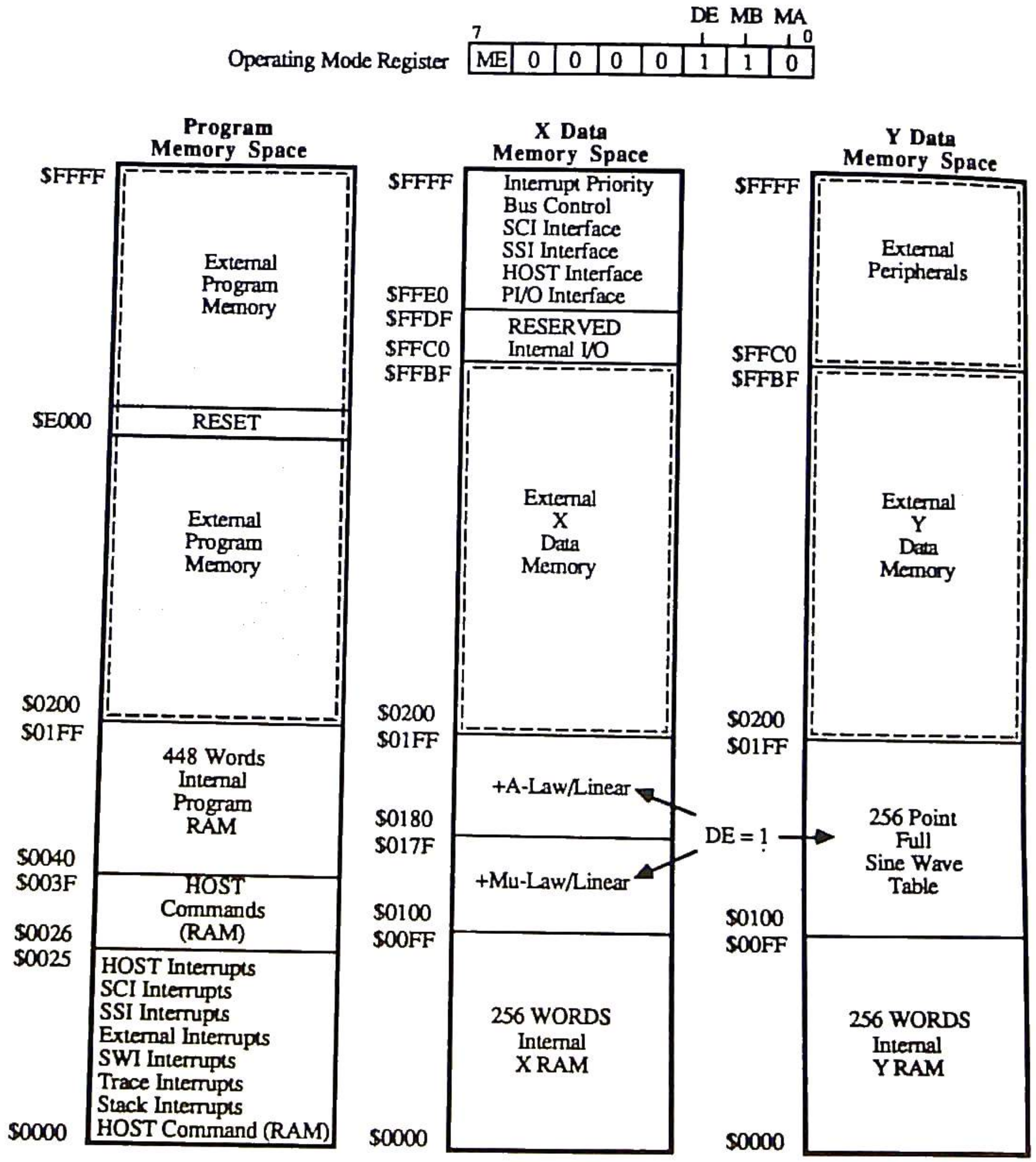
Operating Mode	Mode Pins		DSP56000 Program Memory Map		
	MB	MA	Internal ROM	External	Reset
0 Non-Expanded	0	0	$0000 to $0EFF	$0F00 to $FFFF	Internal at $0000
1 Undefined	0	1	Undefined		
2 Normal Expanded	1	0	$0000 to $0EFF	$0F00 to $FFFF	External at $E000
3 Development	1	1	Disabled	$0000 to $FFFF	External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

NCD02-0257

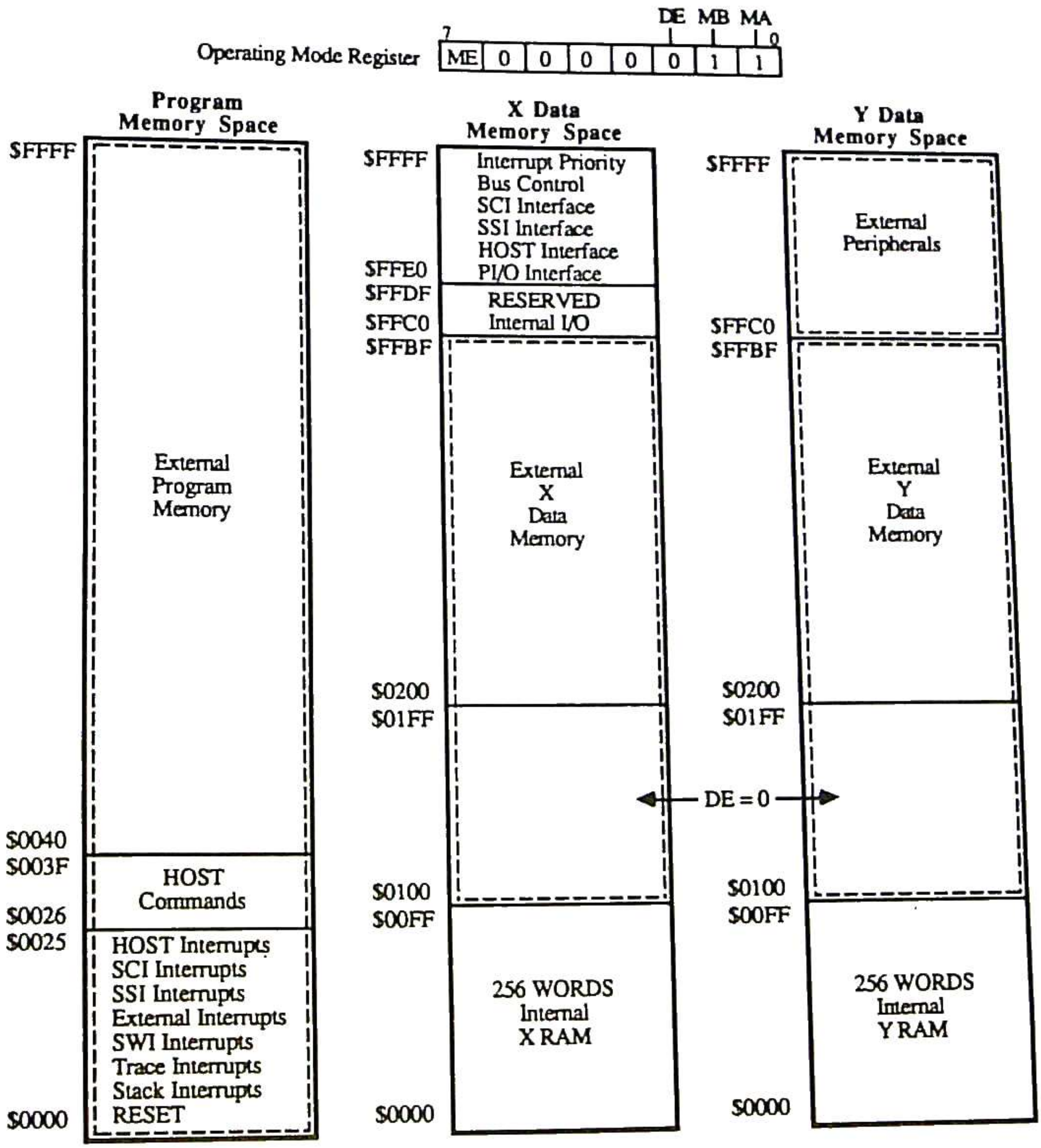
DSP56001 MEMORY MAP MODE 0 - NON-EXPANDED AND INTERNAL RESET



DSP56001 MEMORY MAP MODE 2 - NORMAL EXPANDED AND EXTERNAL RESET



DSP56001 MEMORY MAP
MODE 3 - DEVELOPMENT MODE
EXTERNAL PROGRAM MEMORY



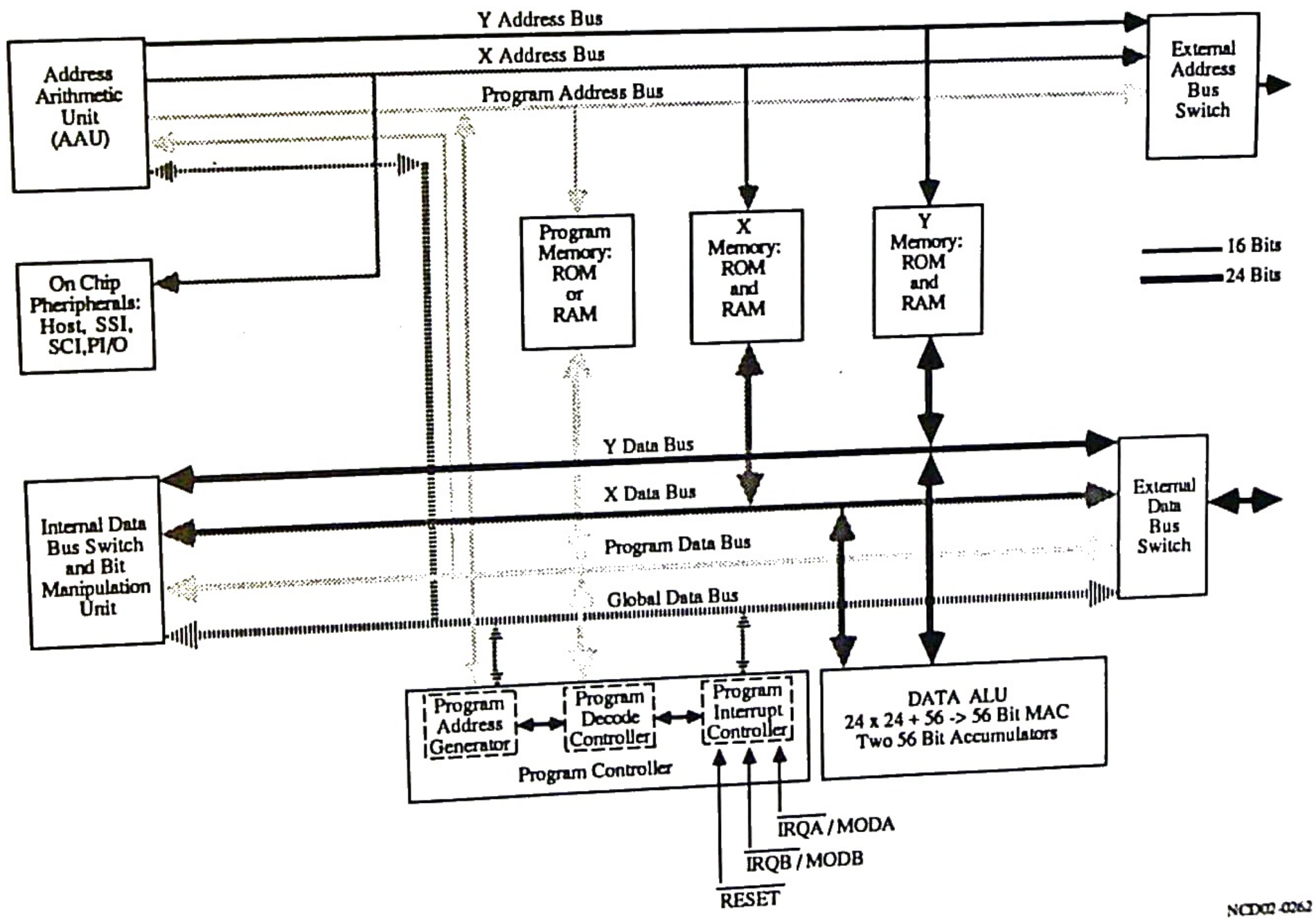
DSP56001 MODES

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 External: $0100 to $FFBF On Chip I/O: $FFC0 to $FFFF	Internal RAM: $0000 to $00FF External: $0100 to $FFFF
1	Internal RAM: $0000 to $00FF Internal ROM: $0100 to $01FF External: $0200 to $FFBF On Chip I/O: $FFC0 to $FFFF	Internal RAM: $0000 to $00FF Internal ROM: $0100 to $01FF External: $0200 to $FFFF

NCD02-0261

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	•	•	•

NCD02-0263

DSP56000/1
PINS AND BUSES

Objectives:

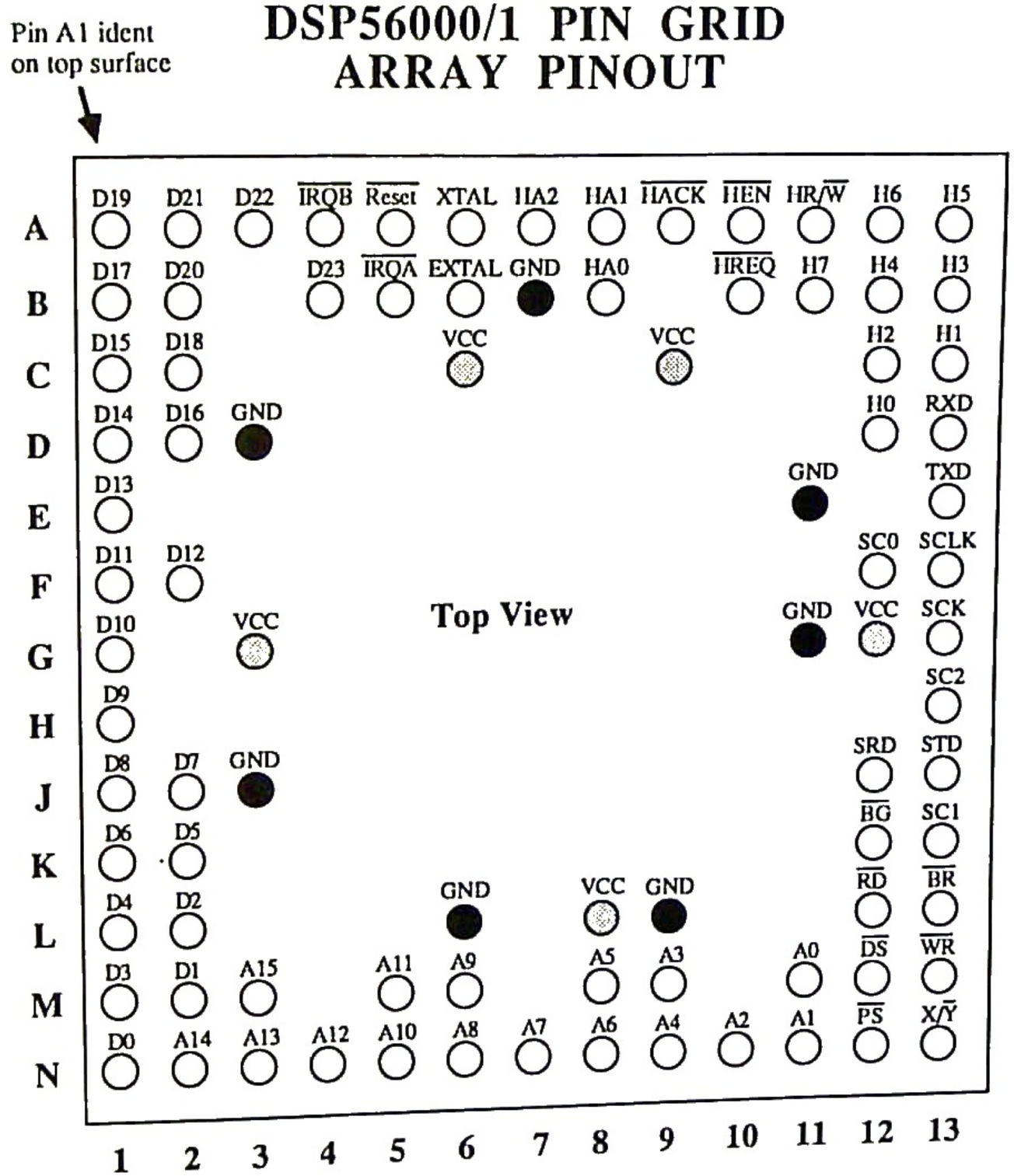
- Describe the Power and Clock Requirements.
- Describe the RESET Sequence.
- Draw a schematic showing how to interface external memory.
- Describe Wait States.
- Describe a Shared Memory System.

NCD02-0300-1

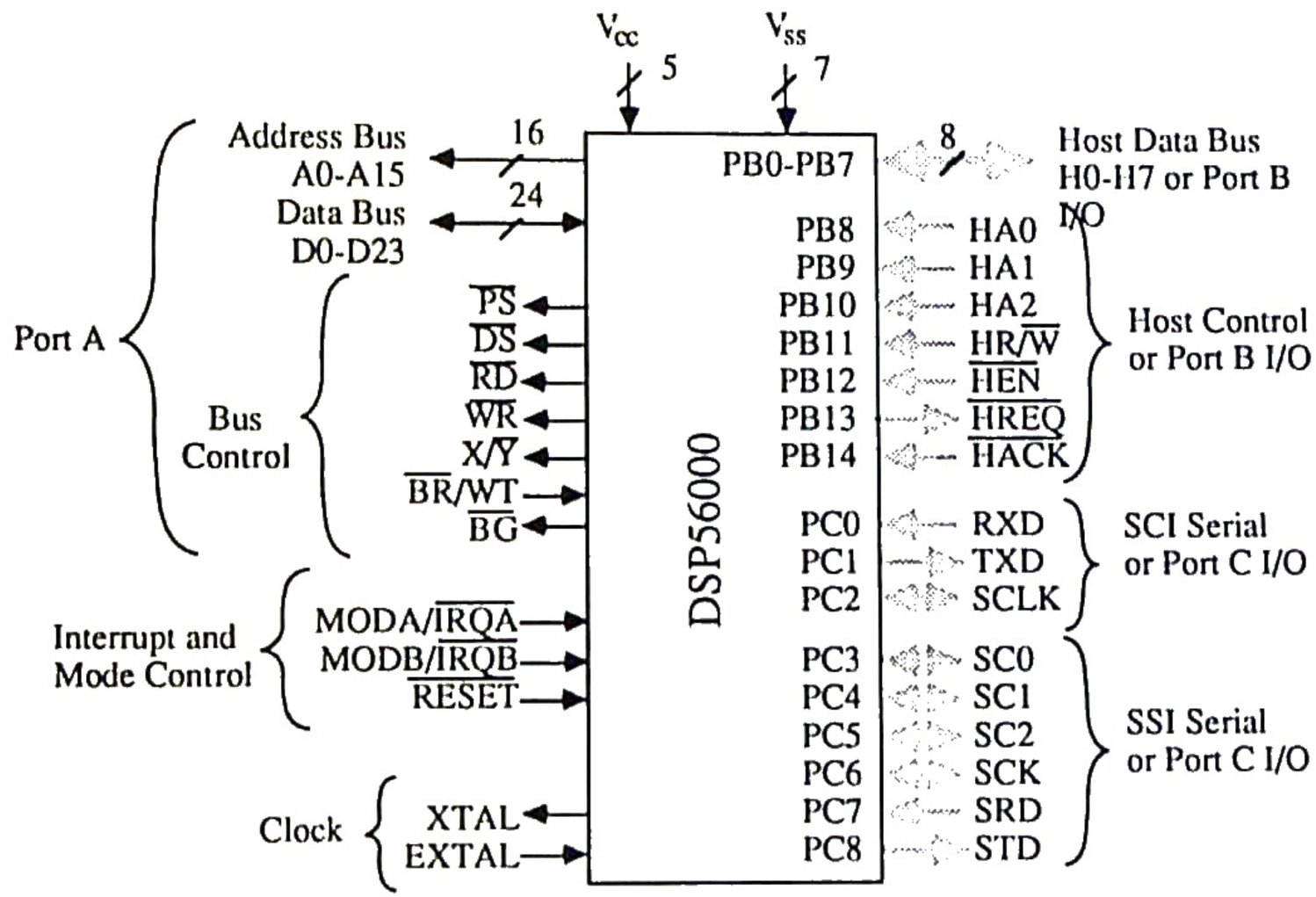
Table of Contents

DSP56000/1 Pin Grid Array Pinout 3-5
Functional Signal Groups 3-6
Signal State Definitions 3-7
PORT A - External BusSwitch 3-8
Power and Ground 3-9
On Chip Oscillator 3-10
External Oscillator 3-10
Asynchronous Reset Timing 3-11
Synchronous Reset Timing 3-11
Operating Mode Select 3-12
Reset Circuits 3-13
Instruction Cycle Timing 3-14
Read Data Timing 3-14
Write Data Timing 3-15
Read/Modify/Write Data Timing 3-15
External Bus Timing Detail 3-16
External Bus AC Electrical Characteristics 3-17
External Program Space 3-18
External X and Y Data Space 3-18
Single Chip External Memory Example 3-19
Port A Bootstrap Mode 3-19
Port A Bootstrap with X-Y RAM 3-20
Bus Control Register 3-21
Two Wait State Timing 3-22
Mixed Speed Expanded System 3-23
Shared Memory System 3-24
Memory Ready Enable Bit 3-24
Bus Arbitration Timing 3-25
Bus Mastership 3-26
Memory Ready 3-27
Memory Ready Timing 3-28
Memory Ready Vs. The Bus Control Register 3-28

DSP56000/1 PIN GRID ARRAY PINOUT



FUNCTIONAL SIGNAL GROUPS



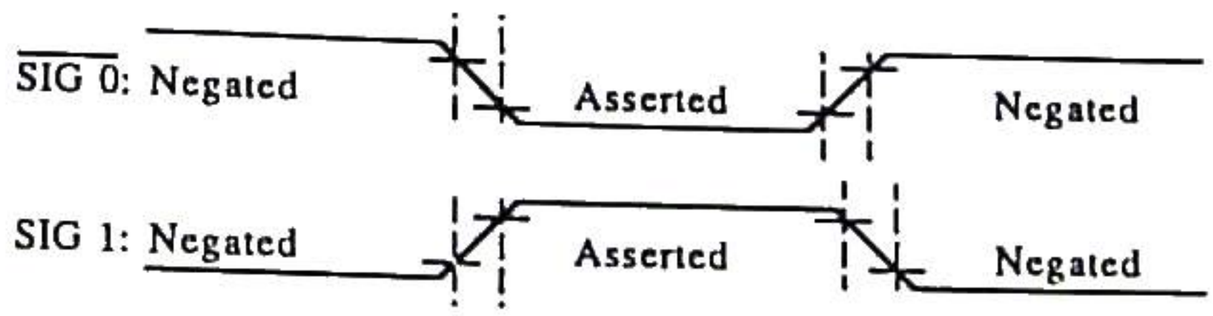
PGA Sockets Available From

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	Local Distribution	Zero Insertion Force

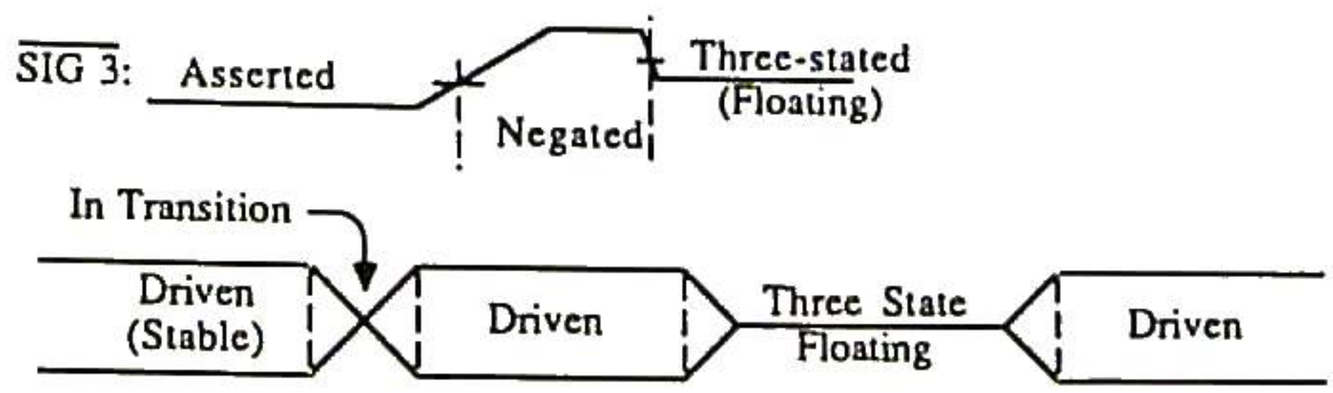
SIGNAL STATE DEFINITIONS

Assertion (ASSERT)
A signal (pin) is active or true independent of the actual voltage level.

Negation (NEGATE)
A signal (pin) is inactive or false independent of the actual voltage level.

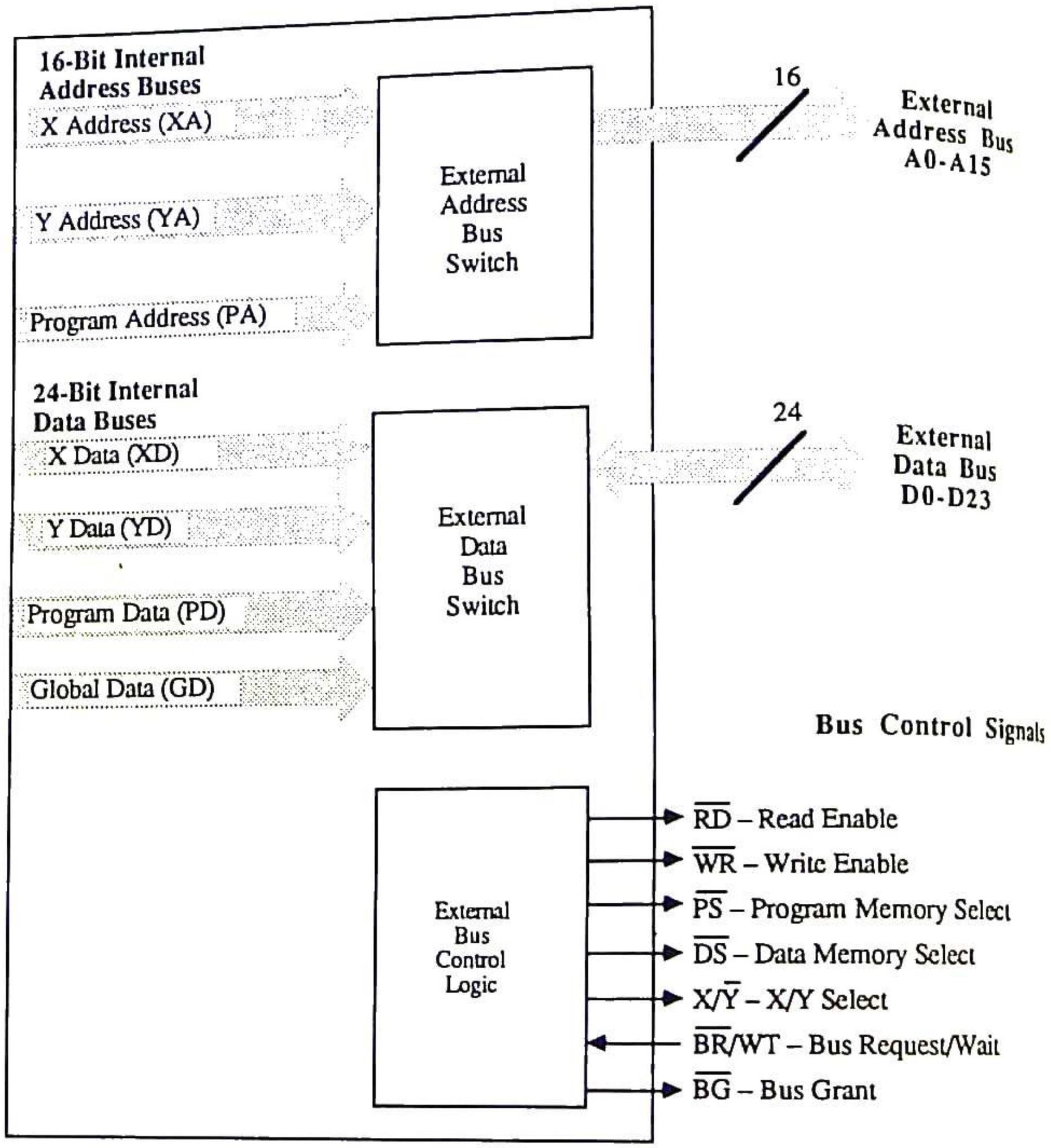


Rescindable (RESCIND)
A signal (pin) is first negated and then three-stated.

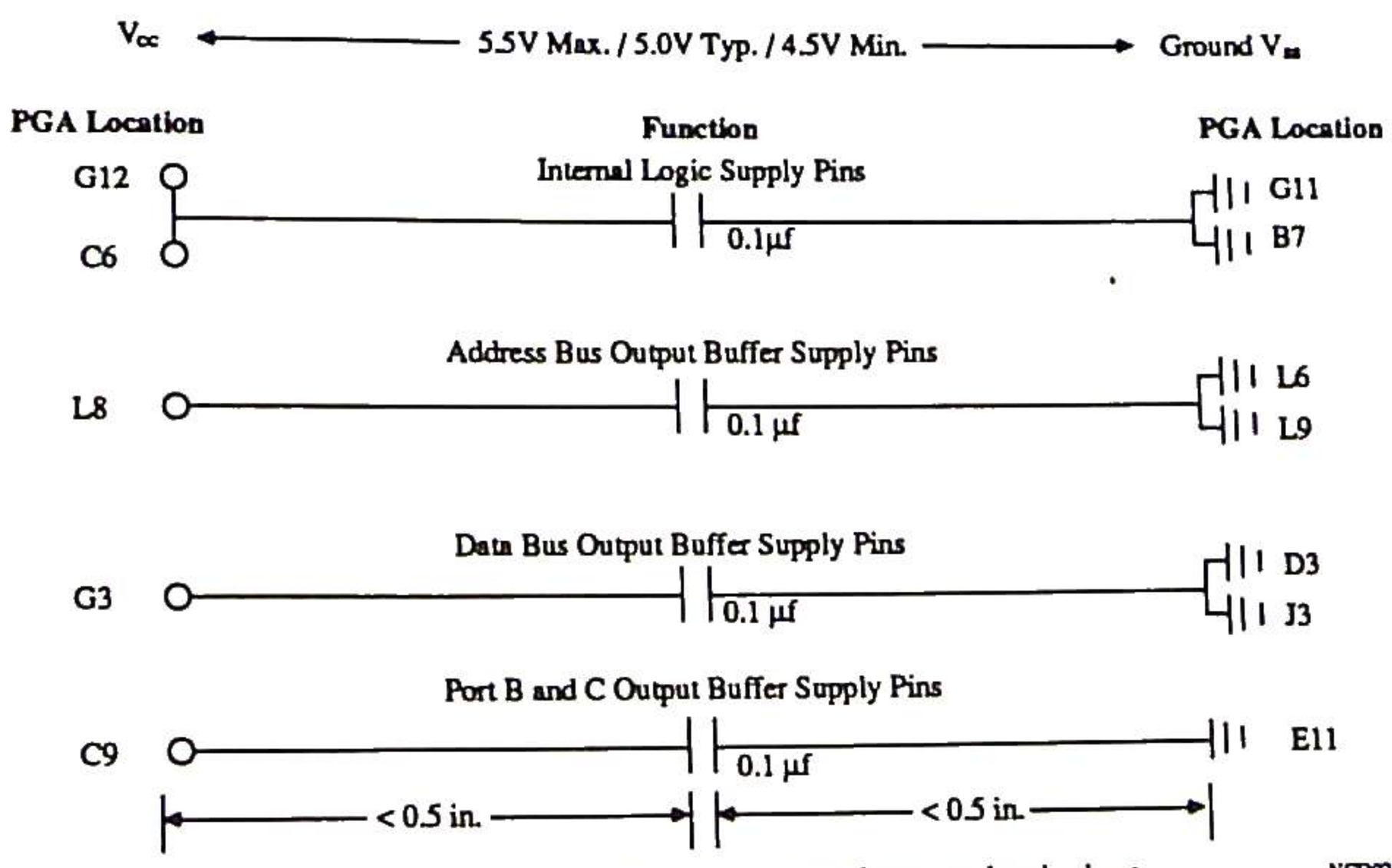


NCD02-0303-1

PORTA-EXTERNAL BUS SWITCHES



POWER AND GROUND

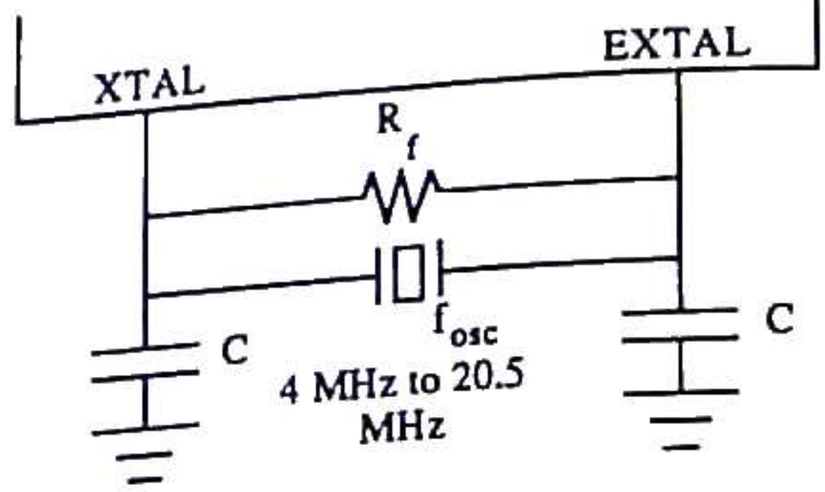


NOTE: 0.1 µf decoupling capacitors are recommended to control undershoot.

NCD02-0305-1

ON CHIP OSCILLATOR

Crystal Oscillator Circuit



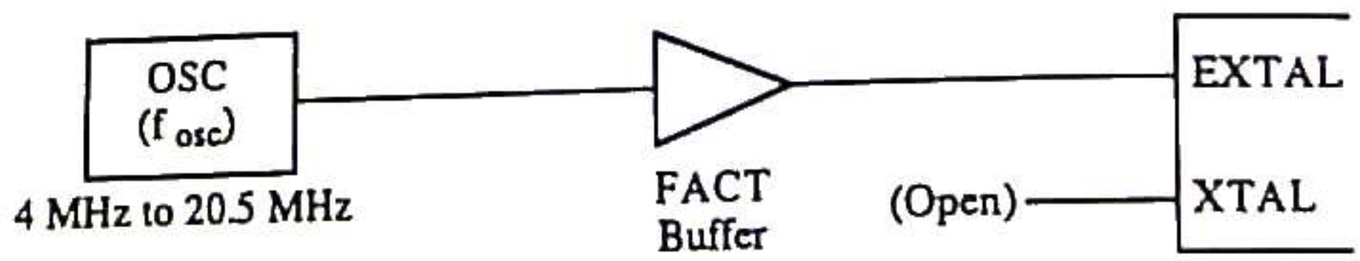
Recommended Component Values

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

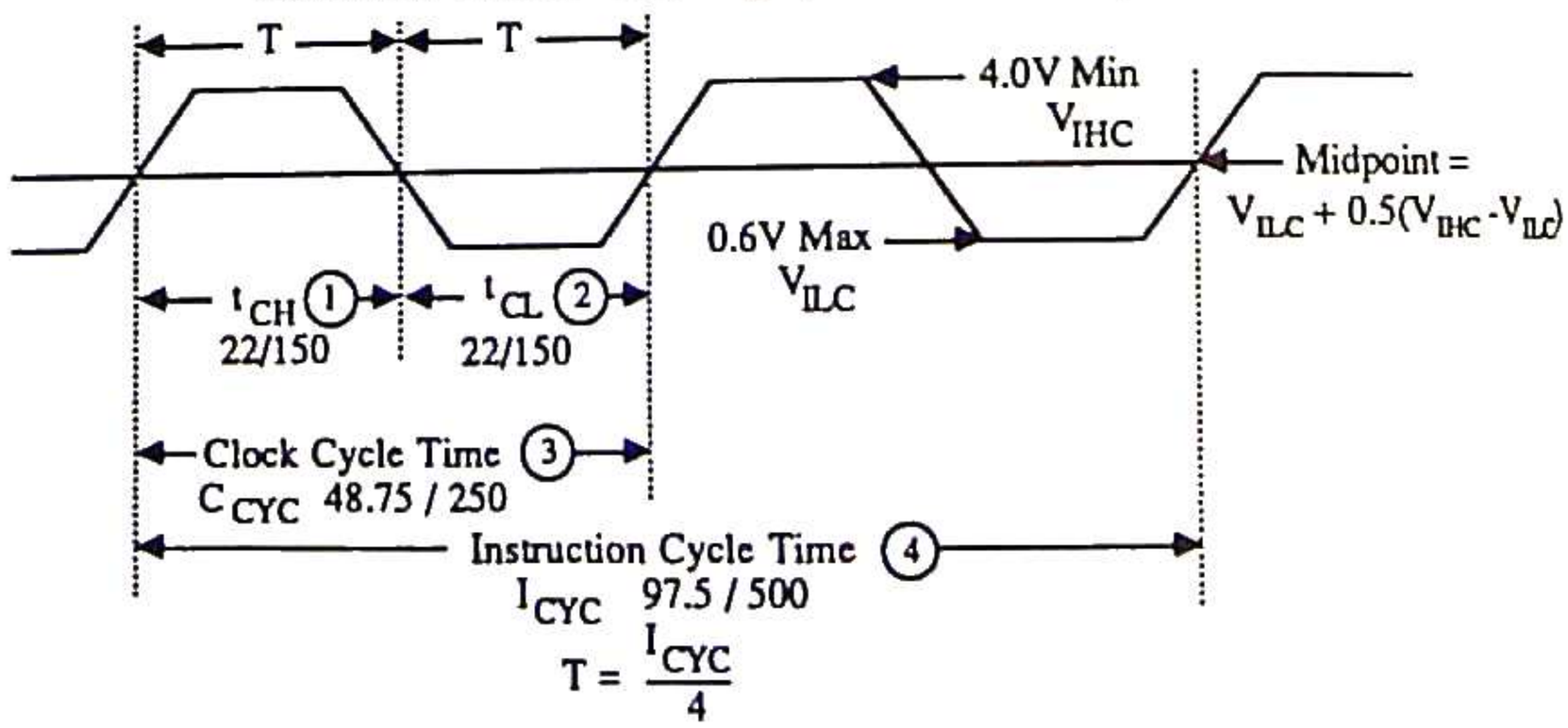
- 150,000T Stabilization Delay is Required after
  - Power on Reset and
  - When Recovering from STOP Mode

NCD02-036

EXTERNAL OSCILLATOR



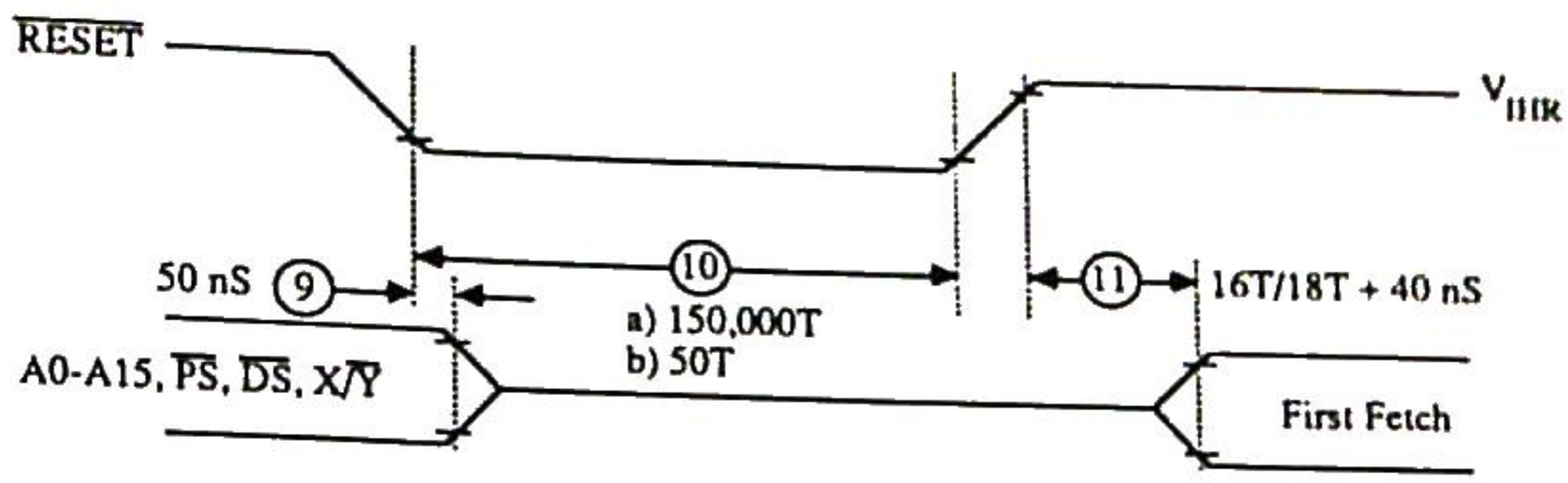
EXTAL Clock Timing (Min/Max nS)



- 50 T Circuit Stabilization Delay is Required
  - After Power on Reset and
  - When Recovering from STOP Mode.

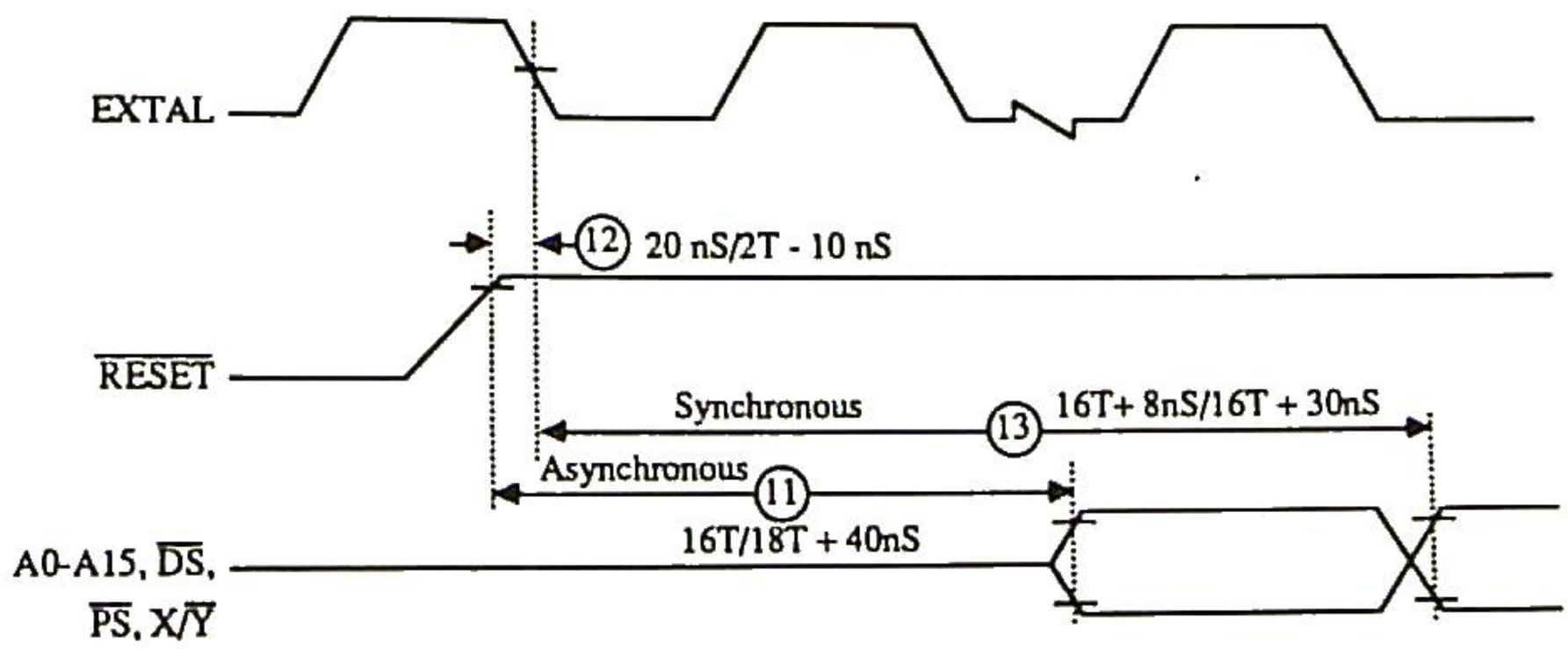
NCD02-037

ASYNCHRONOUS RESET TIMING



NCD02-0308

SYNCHRONOUS RESET TIMING



NCD02-0309

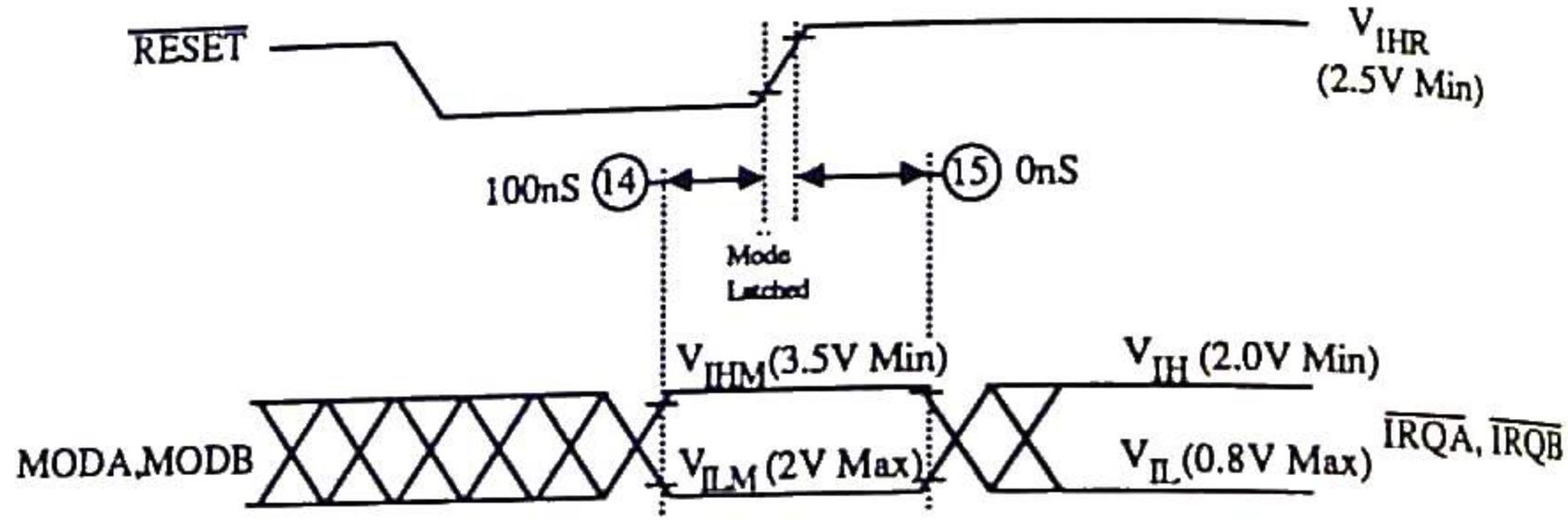
OPERATING MODE SELECT

Operating Mode Register

7							0
ME	0	0	0	0	0	MB	MA

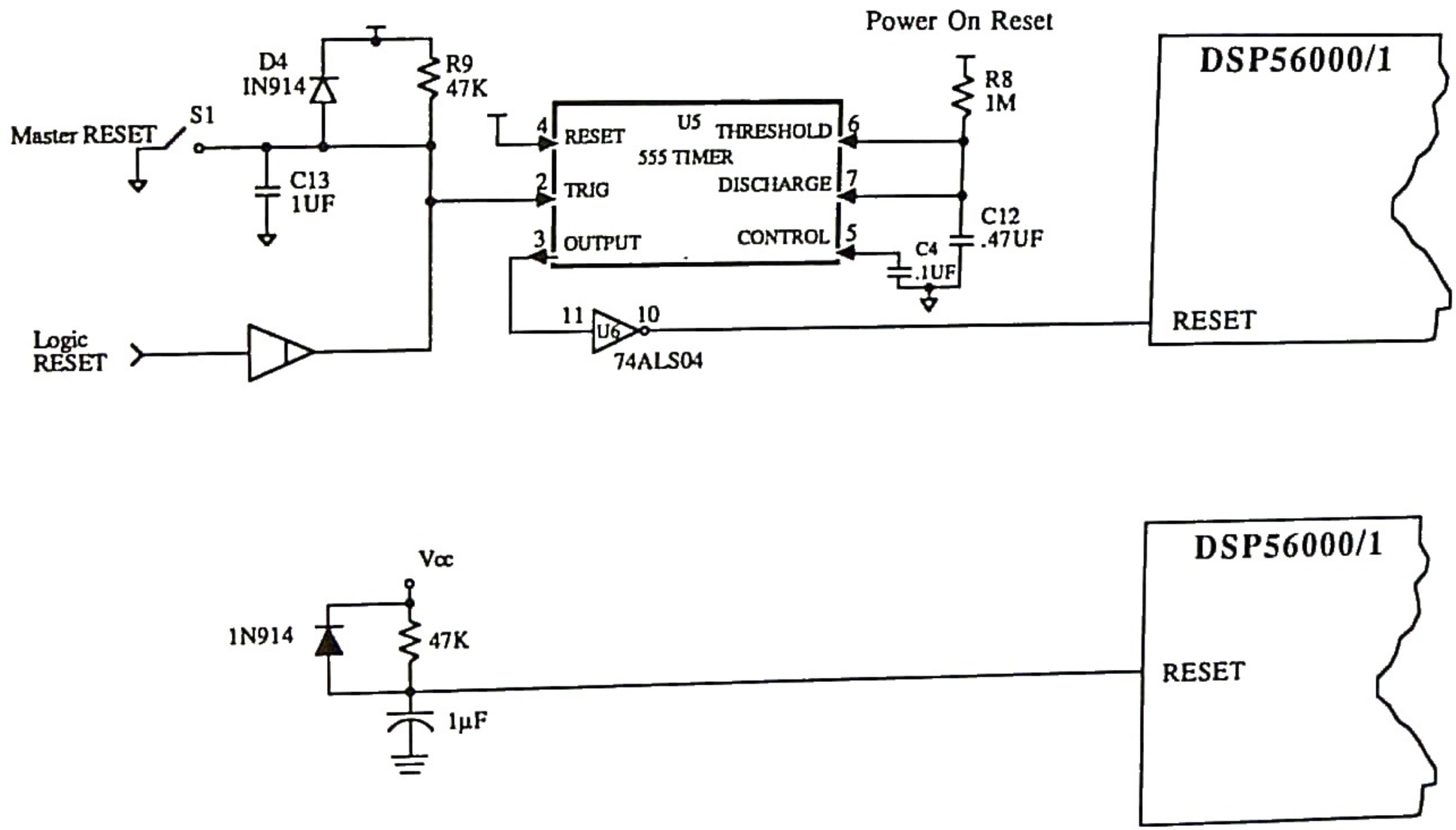
DE – DSP56001 Only

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

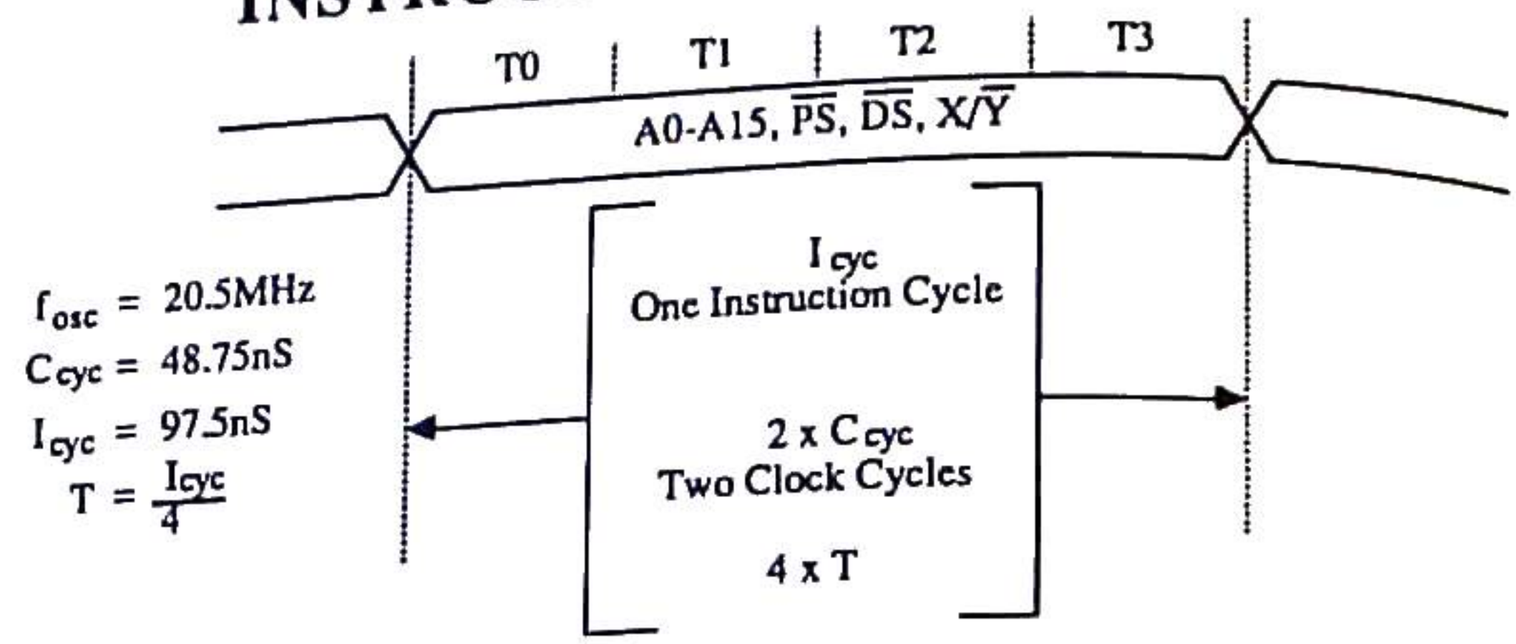


NCD02-0310

RESET CIRCUITS



INSTRUCTION CYCLE TIMING

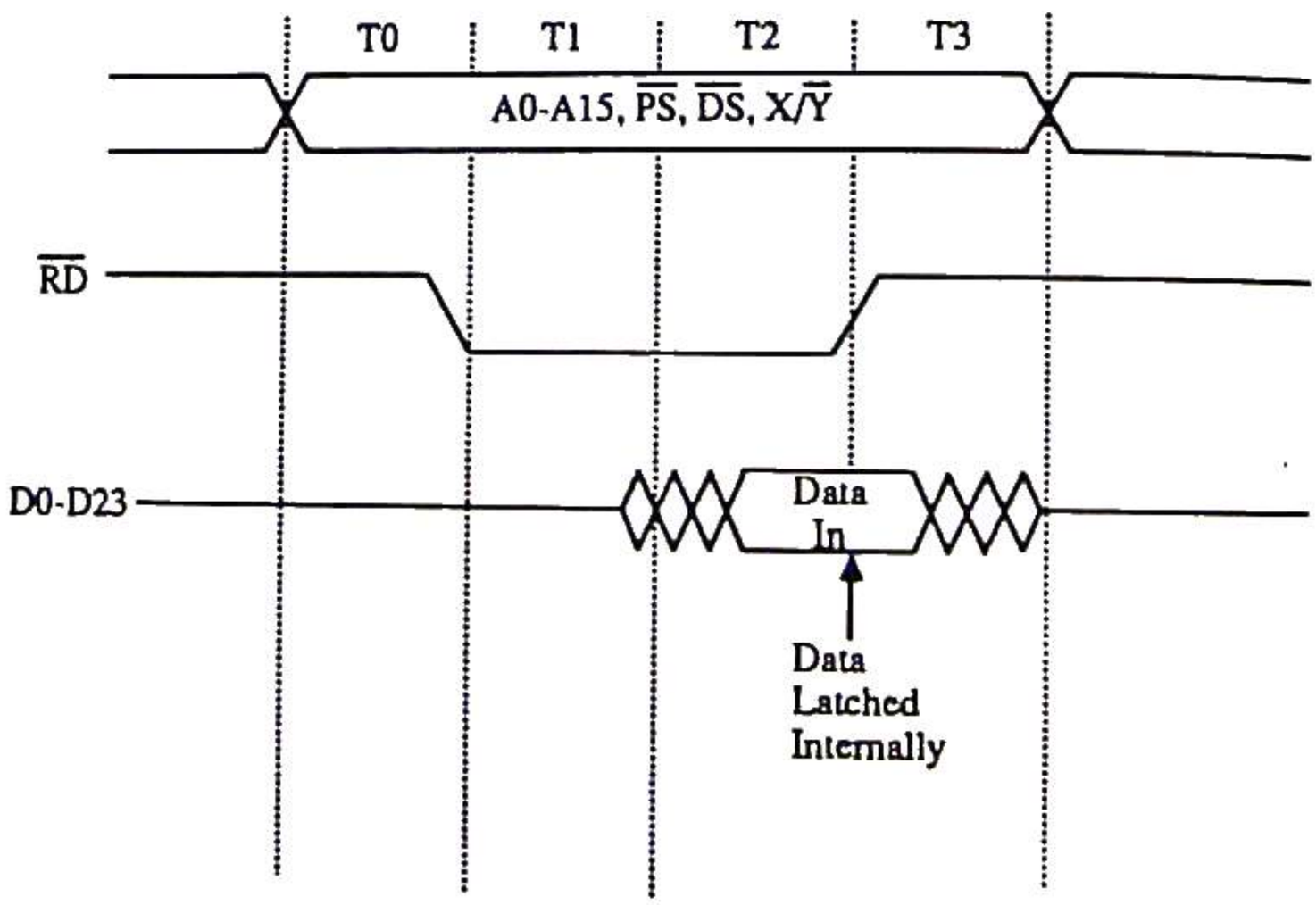


f_{osc} = 20.5MHz
C_{cyc} = 48.75nS
I_{cyc} = 97.5nS
$T = \frac{I_{cyc}}{4}$

$\overline{PS}$	$\overline{DS}$	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

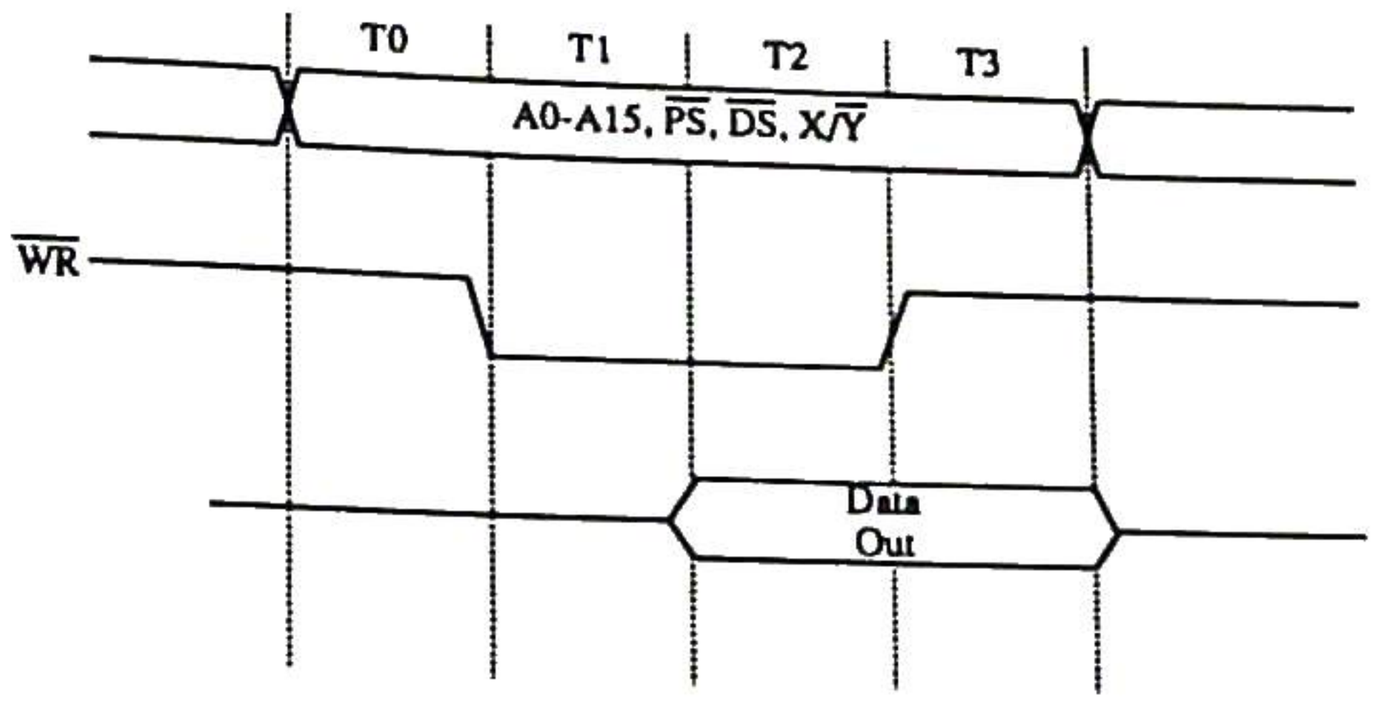
NCD02-0312

READ DATA TIMING



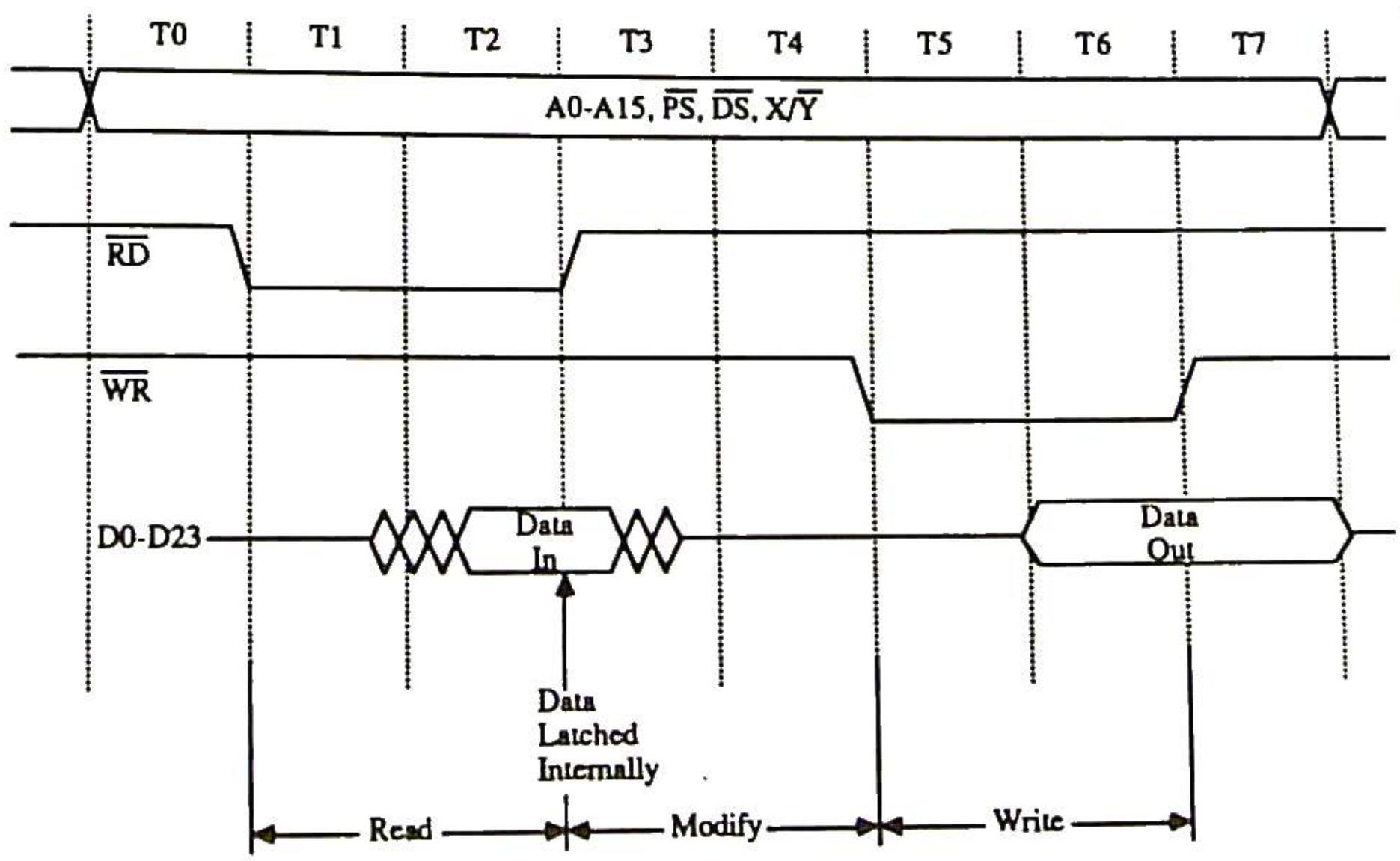
NCD02-0313

WRITE DATA TIMING



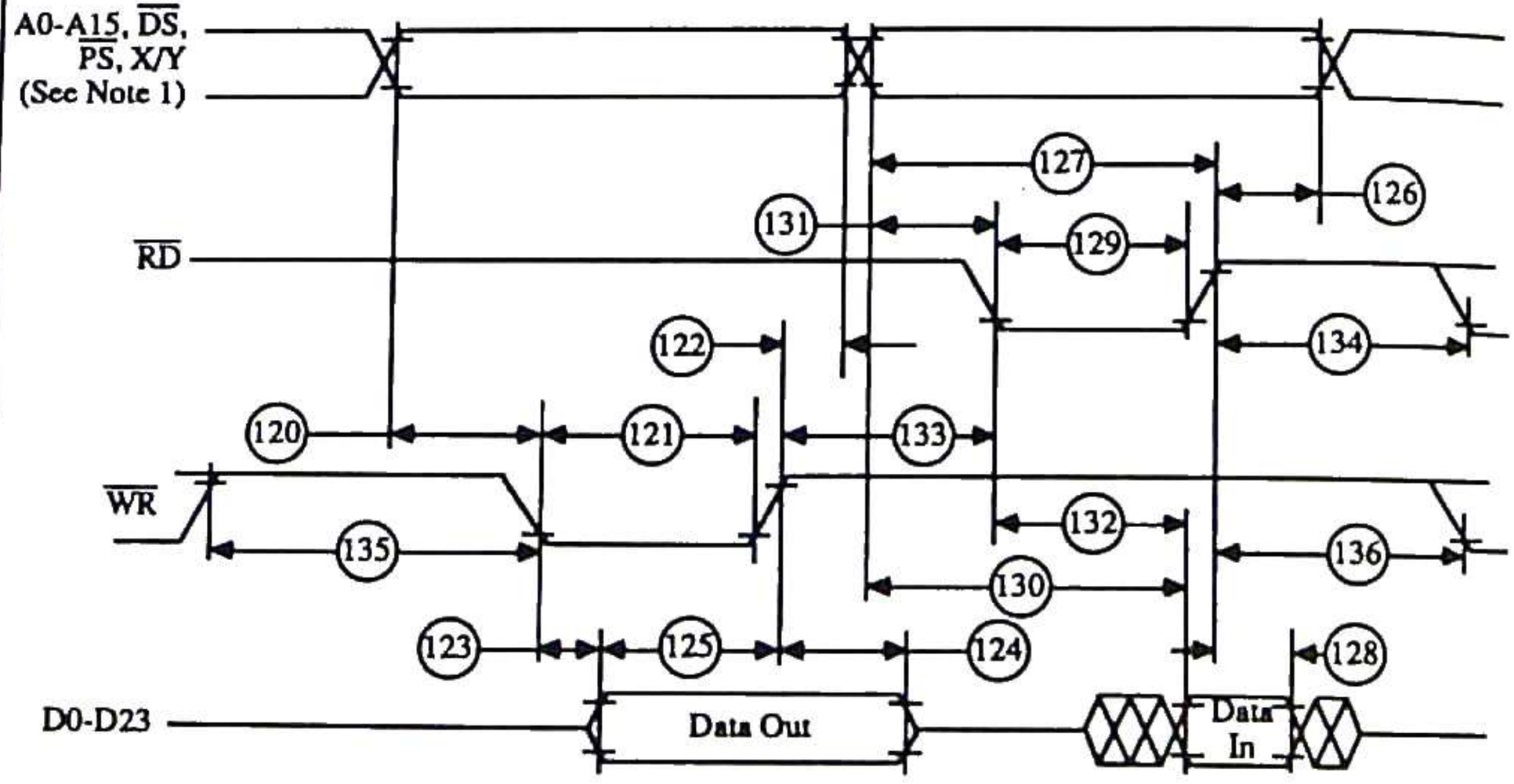
NCD02-0314

READ-MODIFY-WRITE DATA TIMING



NCD02-0315-1

EXTERNAL BUS TIMING DETAIL



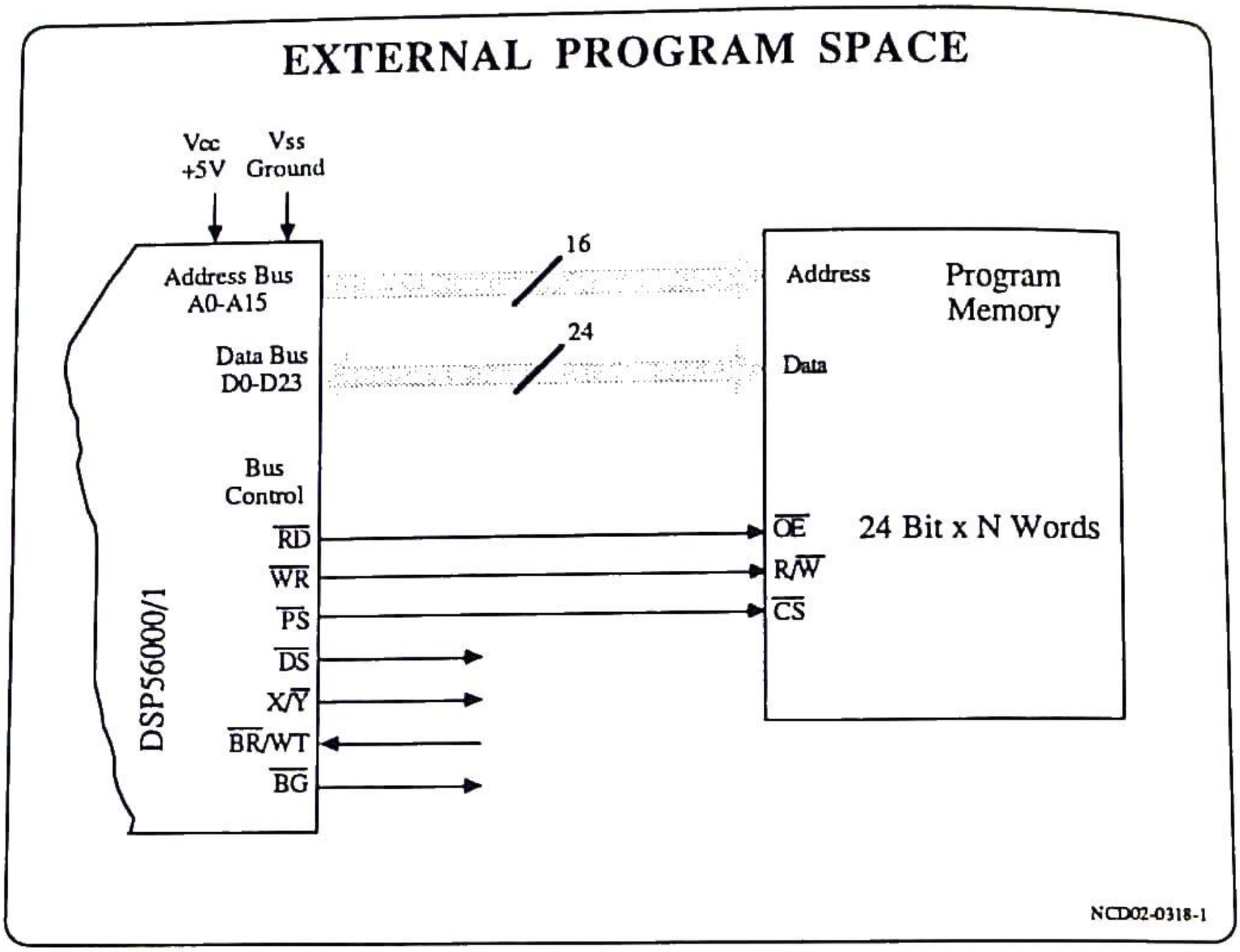
Note: 1. During Read-Modify-Write instructions, these address lines do not change state.

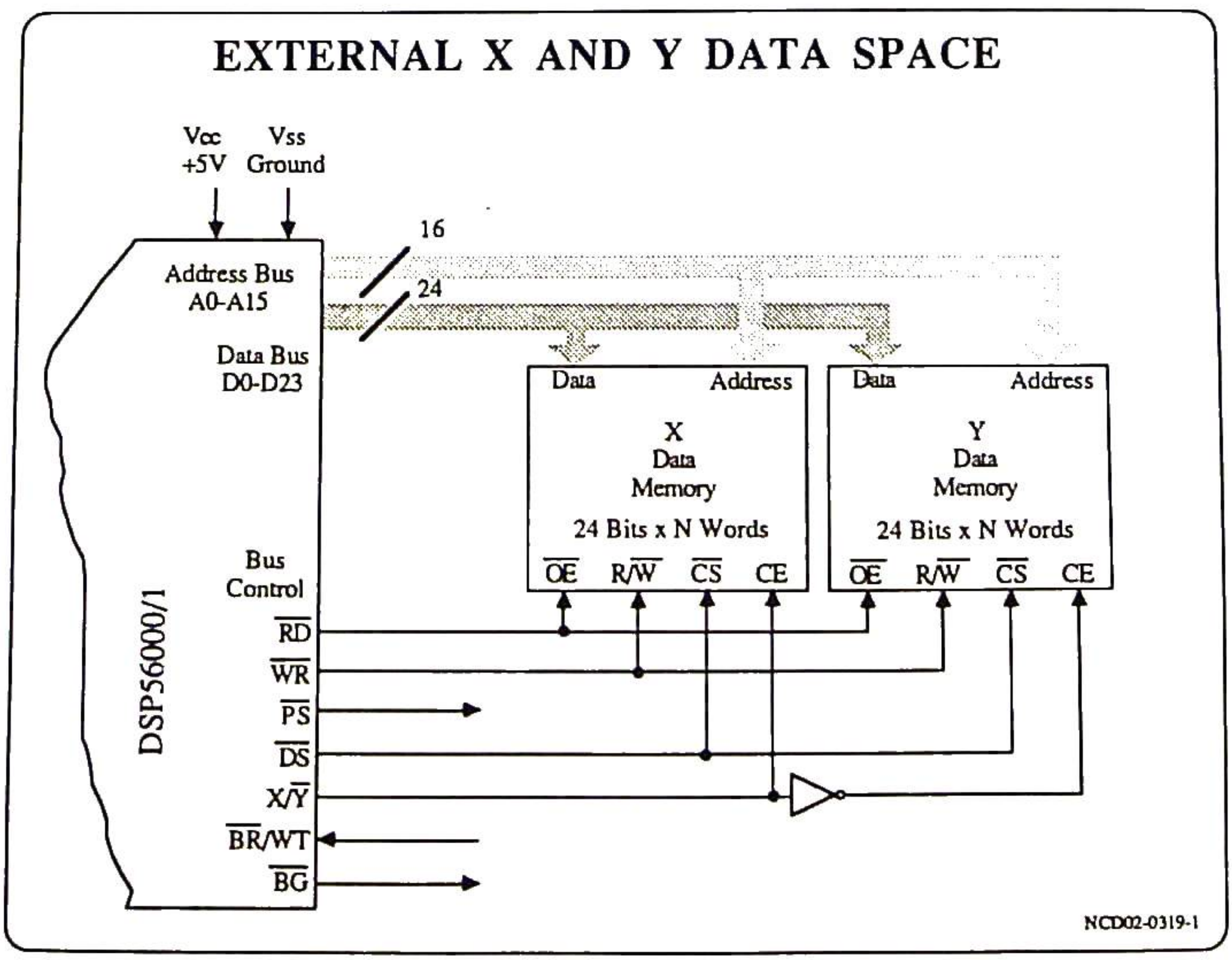
NCD02-0316-1

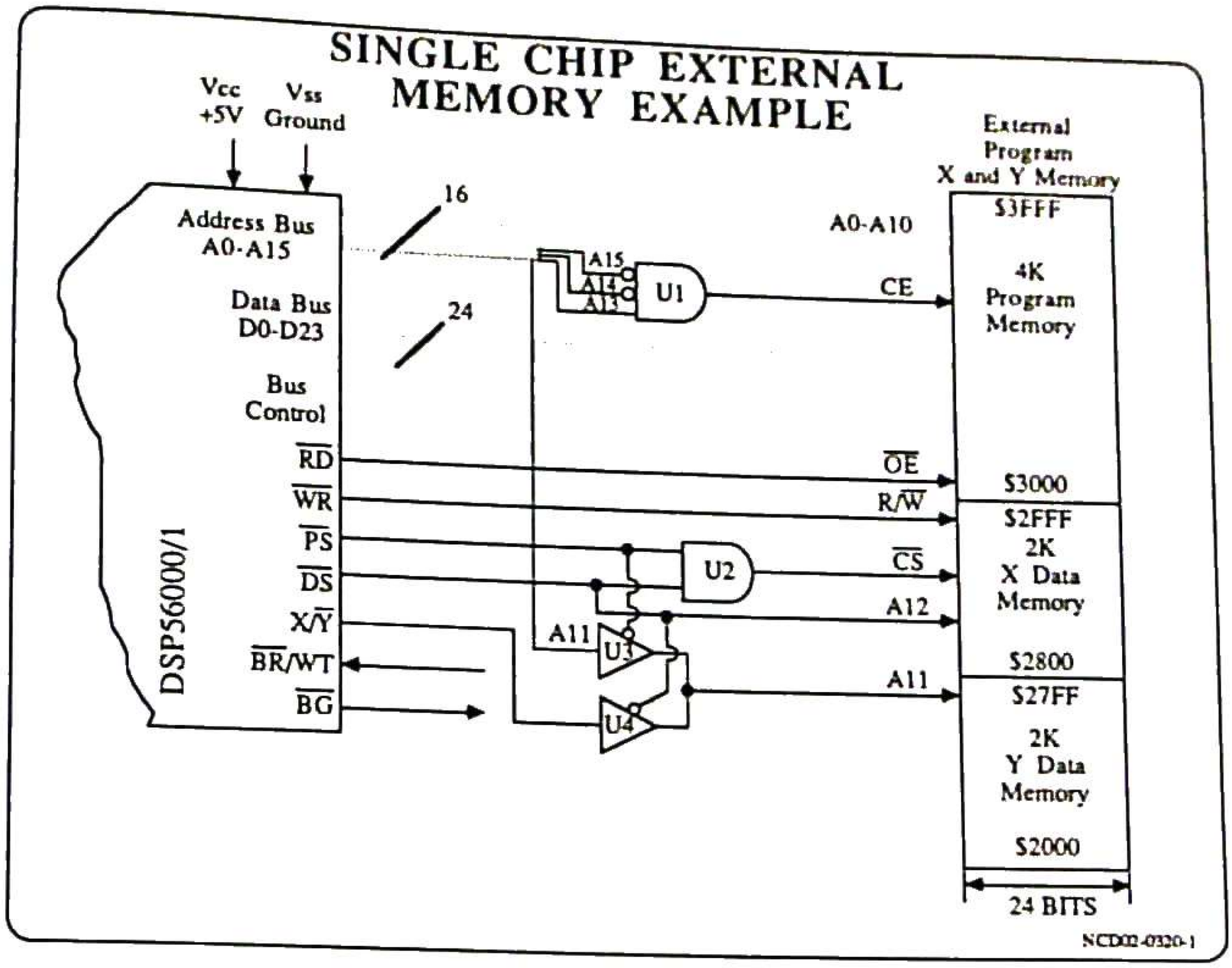
EXTERNAL BUS AC ELECTRICAL CHARACTERISTICS

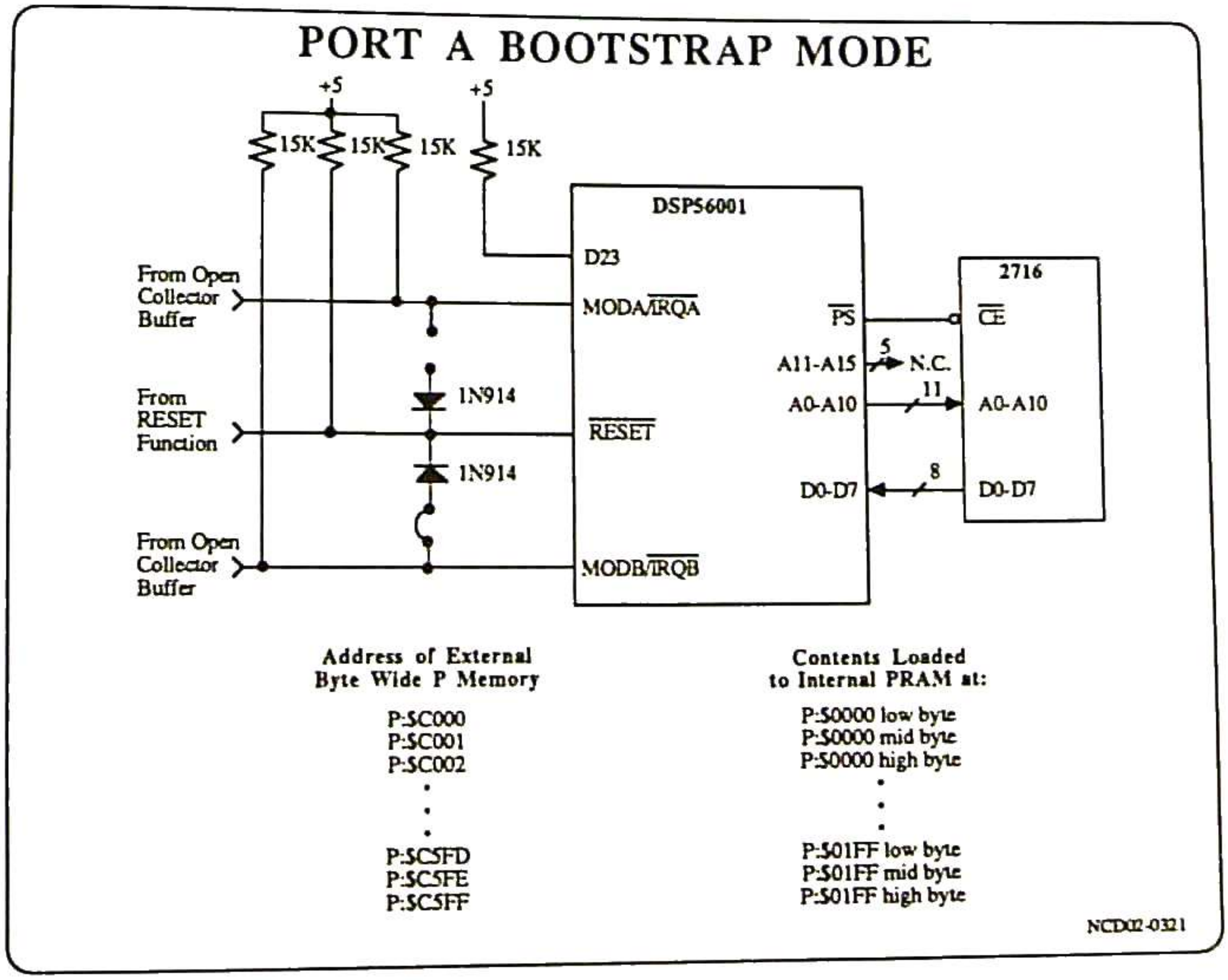
Num	Characteristic		Min	Max	Unit
120	Address Valid to $\overline{WR}$ Assertion	WS = 0 WS > 0	T–9 2T–9	T 2T	ns
121	$\overline{WR}$ Assertion Width	WS = 0 WS > 0	2T–9 (2•WS+1)•T–9	– –	ns
122	$\overline{WR}$ Deassertion to Address Not Valid		T–12	–	ns
123	$\overline{WR}$ Assertion to Data Out Valid	WS = 0 WS > 0	T–9 0	T+10 10	ns
124	Data Out Hold Time from $\overline{WR}$ Deassertion		T–9	T+7	ns
125	Data Out Setup Time to $\overline{WR}$ Deassertion	WS = 0 WS > 0	T–4 (2•WS+1)•T–4	– –	ns
126	$\overline{RD}$ Deassertion to Address Not Valid		T–9	–	ns
127	Address Valid to $\overline{RD}$ Deassertion	WS = 0 WS > 0	3T–8 (2•WS+3)•T–8	– –	ns
128	Input Data Hold Time to $\overline{RD}$ Deassertion		0	–	ns
129	$\overline{RD}$ Assertion Width	WS = 0 WS > 0	2T–9 (2•WS+2)•T–9	– –	ns
130	Address Valid to Input Data Valid	WS = 0 WS > 0	– –	3T–18 (2•WS+3)•T–18	ns
131	Address Valid to $\overline{RD}$ Assertion		T–9	T	ns
132	$\overline{RD}$ Assertion to Input Data Valid	WS = 0 WS > 0	– –	2T–14 (2•WS+2)•T–14	ns
133	$\overline{WR}$ Deassertion to $\overline{RD}$ Assertion	WS = 0 WS > 0	2T–15 (2•WS+2)•T–15	– –	ns
134	$\overline{RD}$ Deassertion to $\overline{RD}$ Assertion	WS = 0 WS > 0	2T–10 (2•WS+2)•T–10	– –	ns
135	$\overline{WR}$ Deassertion to $\overline{WR}$ Assertion	WS = 0 WS > 0	2T–15 (2•WS+3)•T–15	– –	ns
136	$\overline{RD}$ Deassertion to $\overline{WR}$ Assertion	WS = 0 WS > 0	2T–10 (2•WS+3)•T–10	– –	ns

NCD02-0317

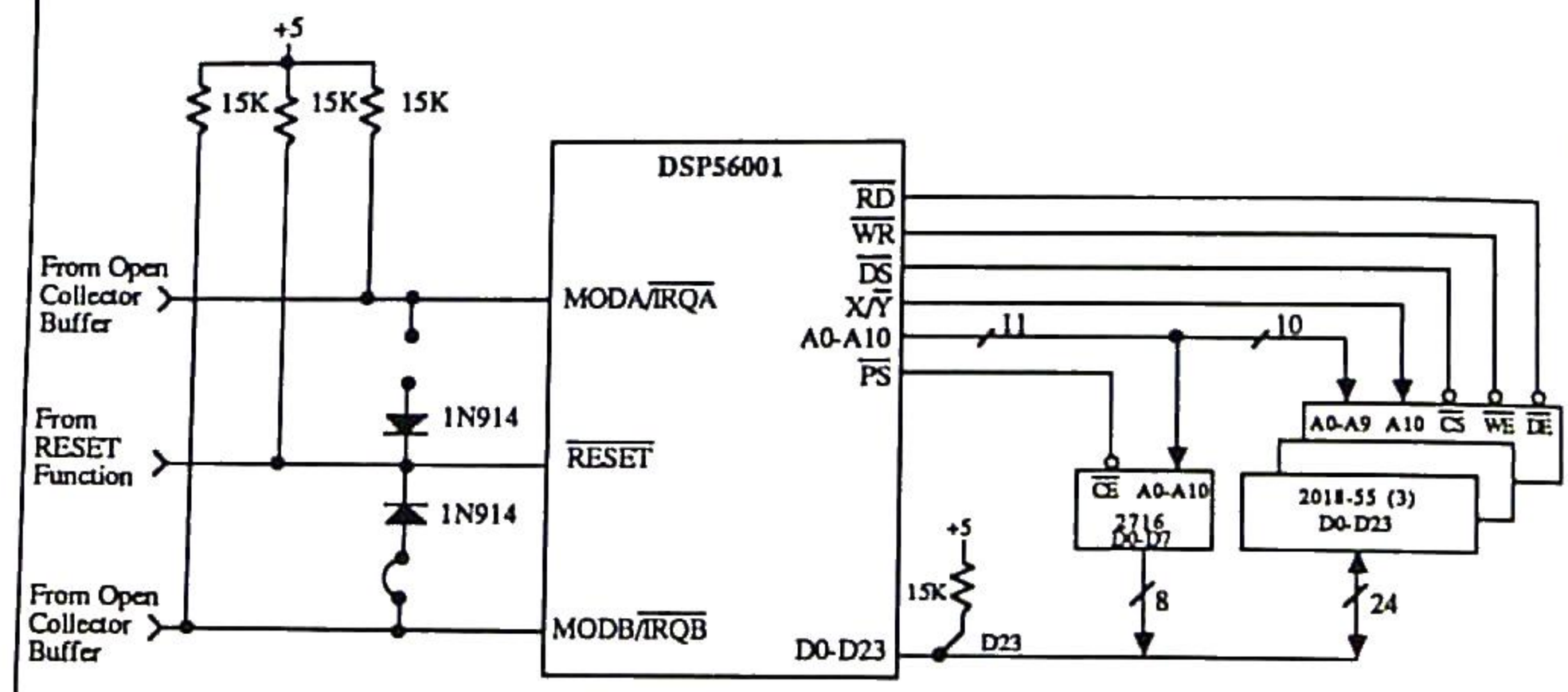

EXTERNAL PROGRAM SPACE
Vcc
+5V
Vss
Ground
Address Bus
A0-A15
Data Bus
D0-D23
Bus
Control
RD
WR
PS
DS
X/Y
BR/WT
BG
DSP56000/1
16
24
Address
Program
Memory
Data
OE
R/W
CS
24 Bit x N Words
NCD02-0318-1


EXTERNAL X AND Y DATA SPACE
Vcc
+5V
Vss
Ground
Address Bus
A0-A15
Data Bus
D0-D23
Bus
Control
RD
WR
PS
DS
X/Y
BR/WT
BG
DSP56000/1
16
24
Data
Address
X
Data
Memory
24 Bits x N Words
OE
R/W
CS
CE
Data
Address
Y
Data
Memory
24 Bits x N Words
OE
R/W
CS
CE
NCD02-0319-1


SINGLE CHIP EXTERNAL MEMORY EXAMPLE
Vcc +5V
Vss Ground
Address Bus A0-A15
Data Bus D0-D23
Bus Control
RD
WR
PS
DS
X/Y
BR/WT
BG
DSP56000/1
16
24
A15
A14
A13
U1
U2
U3
U4
A11
A0-A10
CE
OE
R/W
CS
A12
A11
External Program X and Y Memory
$3FFF
4K Program Memory
$3000
$2FFF
2K X Data Memory
$2800
$27FF
2K Y Data Memory
$2000
24 BITS
NCD02-0320-1

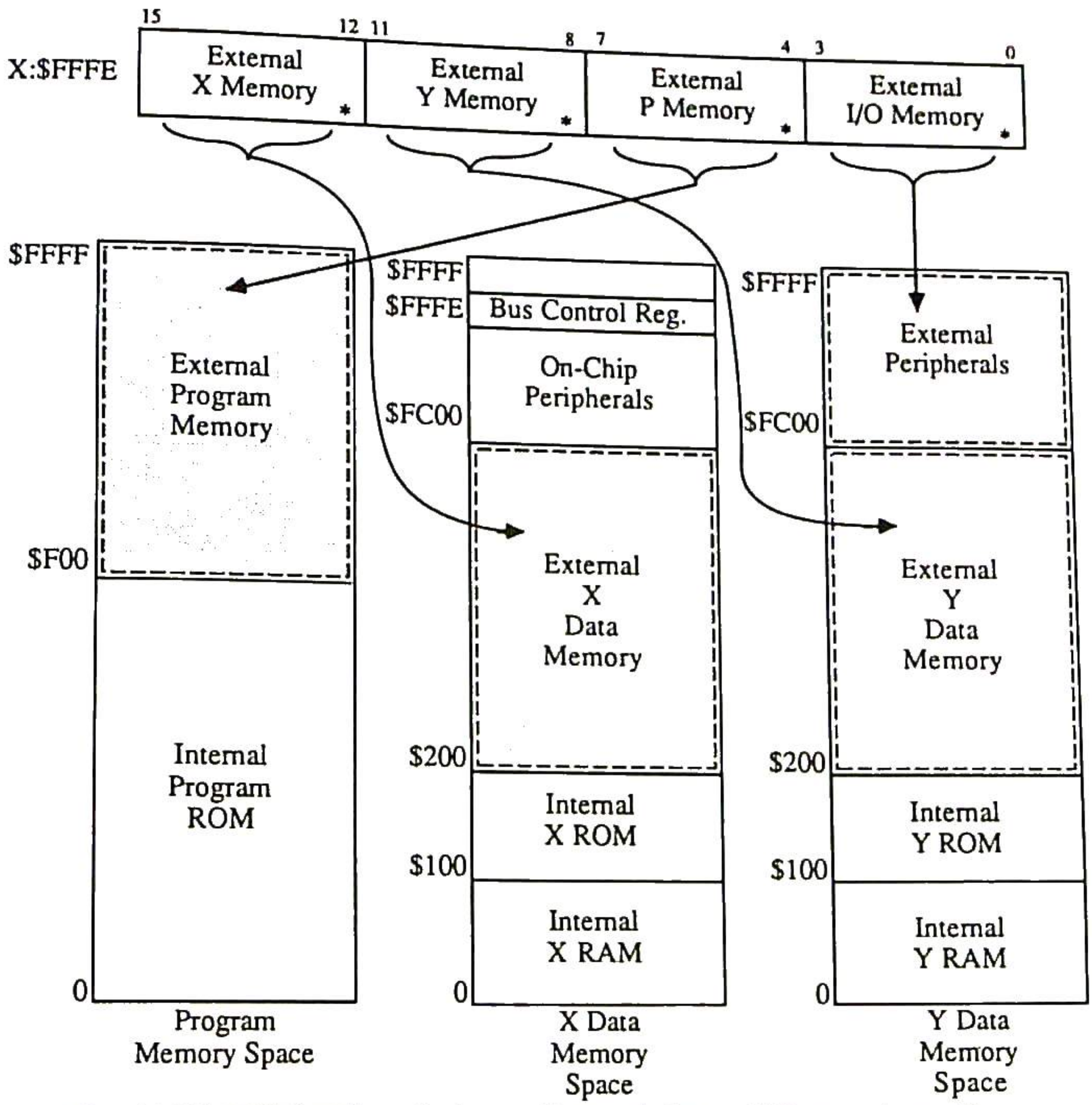

PORT A BOOTSTRAP MODE
+5
+5
15K
15K
15K
15K
DSP56001
D23
From Open Collector Buffer
MODA/IRQA
PS
CE
A11-A15
5
N.C.
From RESET Function
1N914
A0-A10
11
A0-A10
RESET
1N914
D0-D7
8
D0-D7
From Open Collector Buffer
MODB/IRQB
2716
Address of External Byte Wide P Memory
P:$C000
P:$C001
P:$C002
P:$C5FD
P:$C5FE
P:$C5FF
Contents Loaded to Internal PRAM at:
P:$0000 low byte
P:$0000 mid byte
P:$0000 high byte
P:$01FF low byte
P:$01FF mid byte
P:$01FF high byte
NCD02-0321

PORT A BOOTSTRAP WITH X-Y RAM



NCD02-0322

BUS CONTROL REGISTER (BCR)



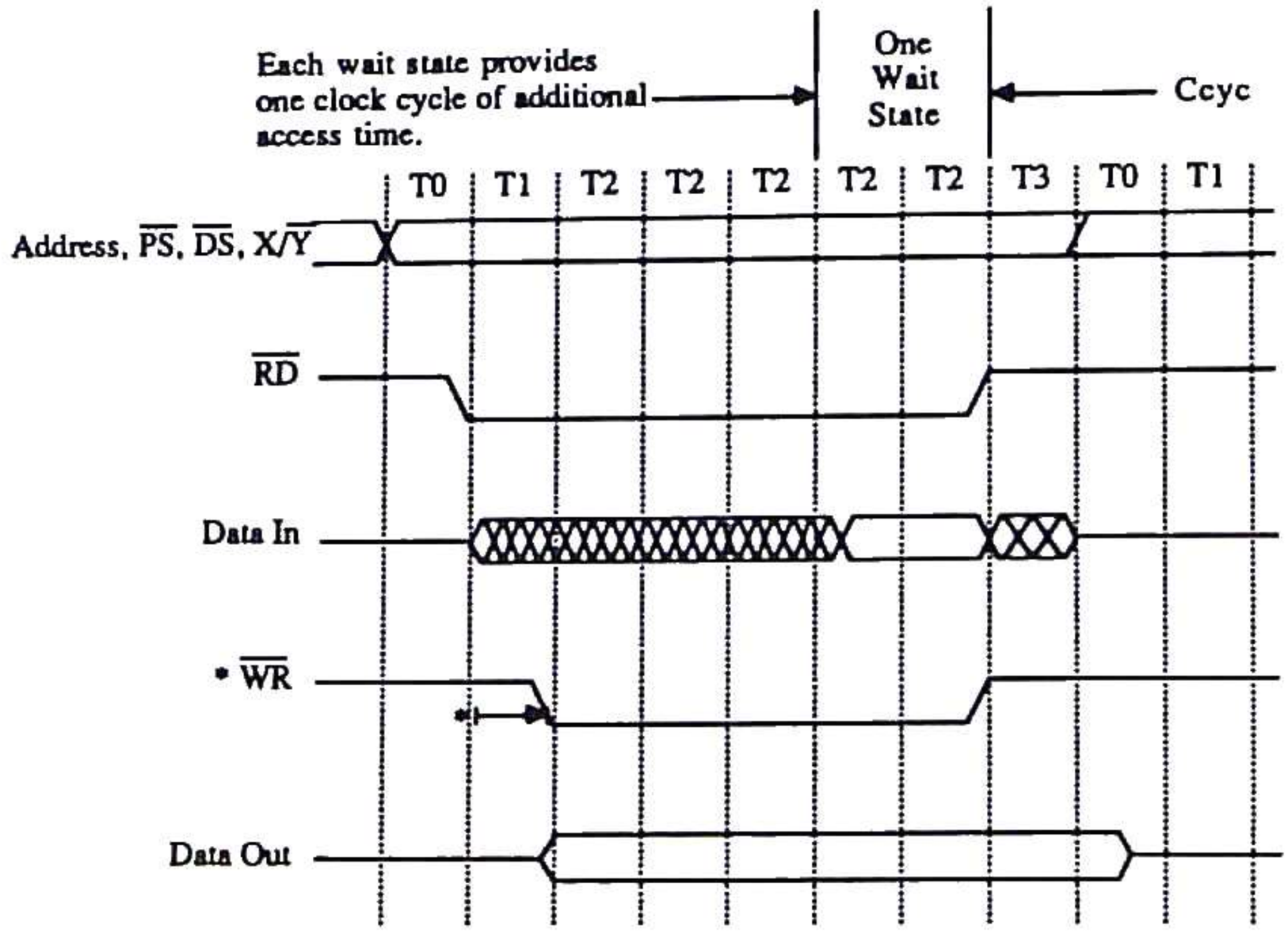
* 0 to 15 WAIT STATES can be inserted into each External Memory Access Category.

On RESET BCR = $FFFF.

1 WAIT STATE = Ccyc = Icyc / 2

NCD02-0323

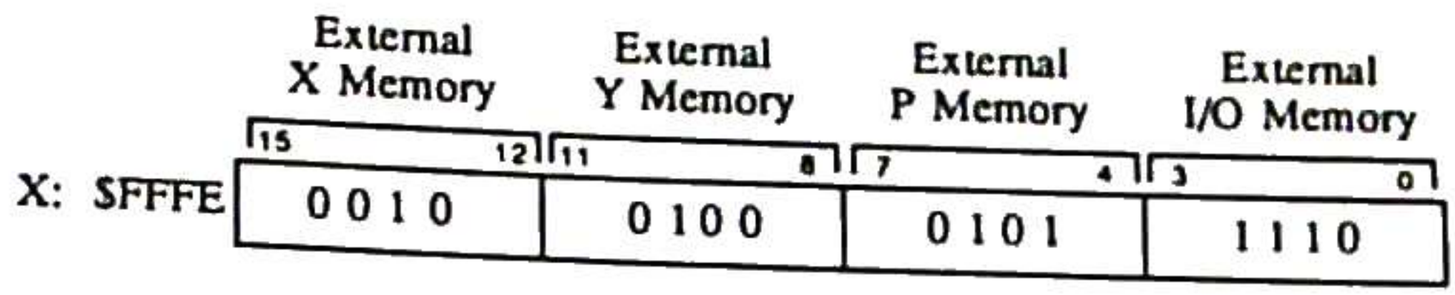
TWO WAIT STATE TIMING
(BUS CONTROL REGISTER = 0010)



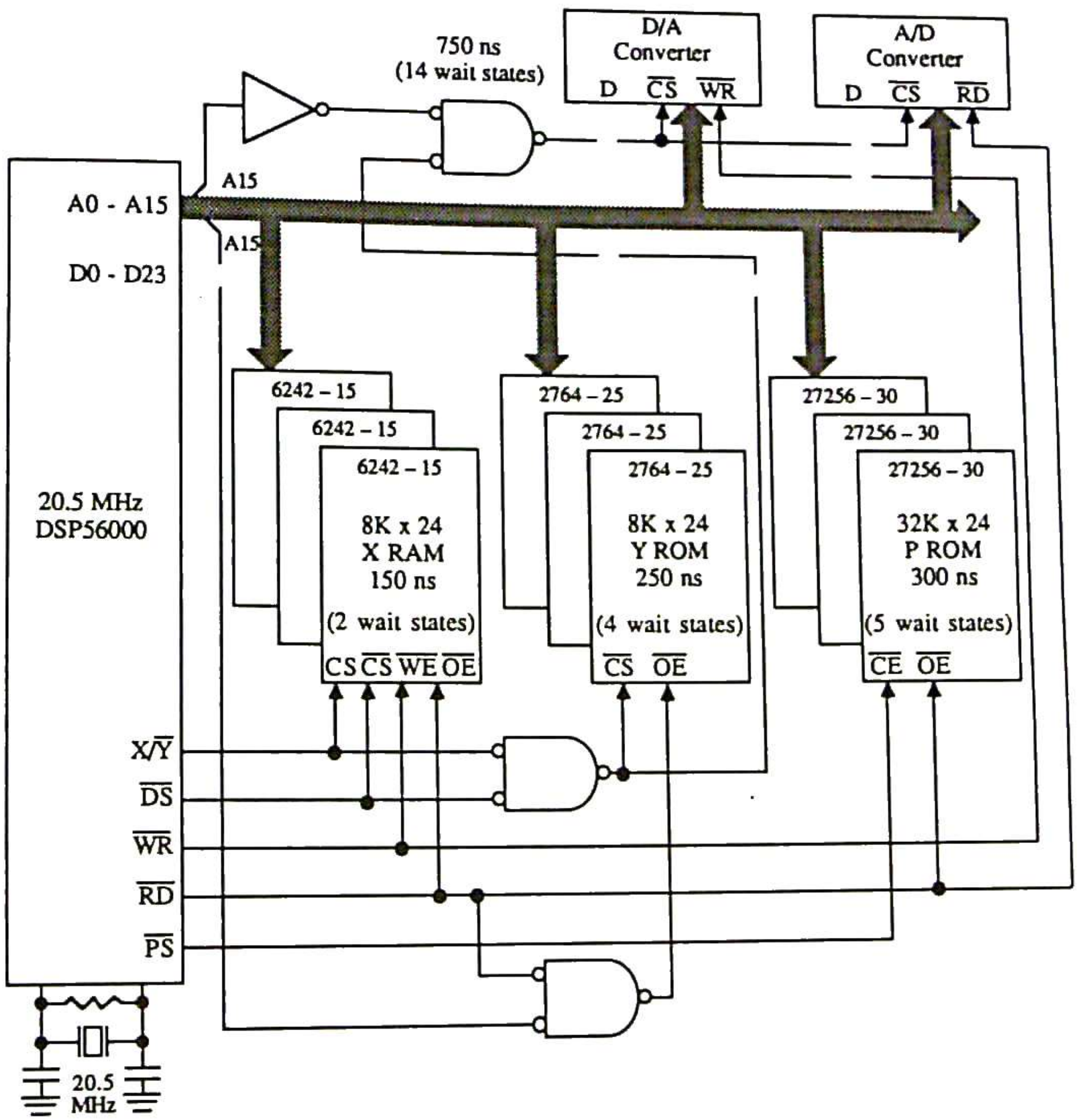
* $\overline{WR}$ is delayed one internal "T" phase when one or more WAIT STATES are inserted.

NCD02-0324-1

MIXED SPEED EXPANDED SYSTEM

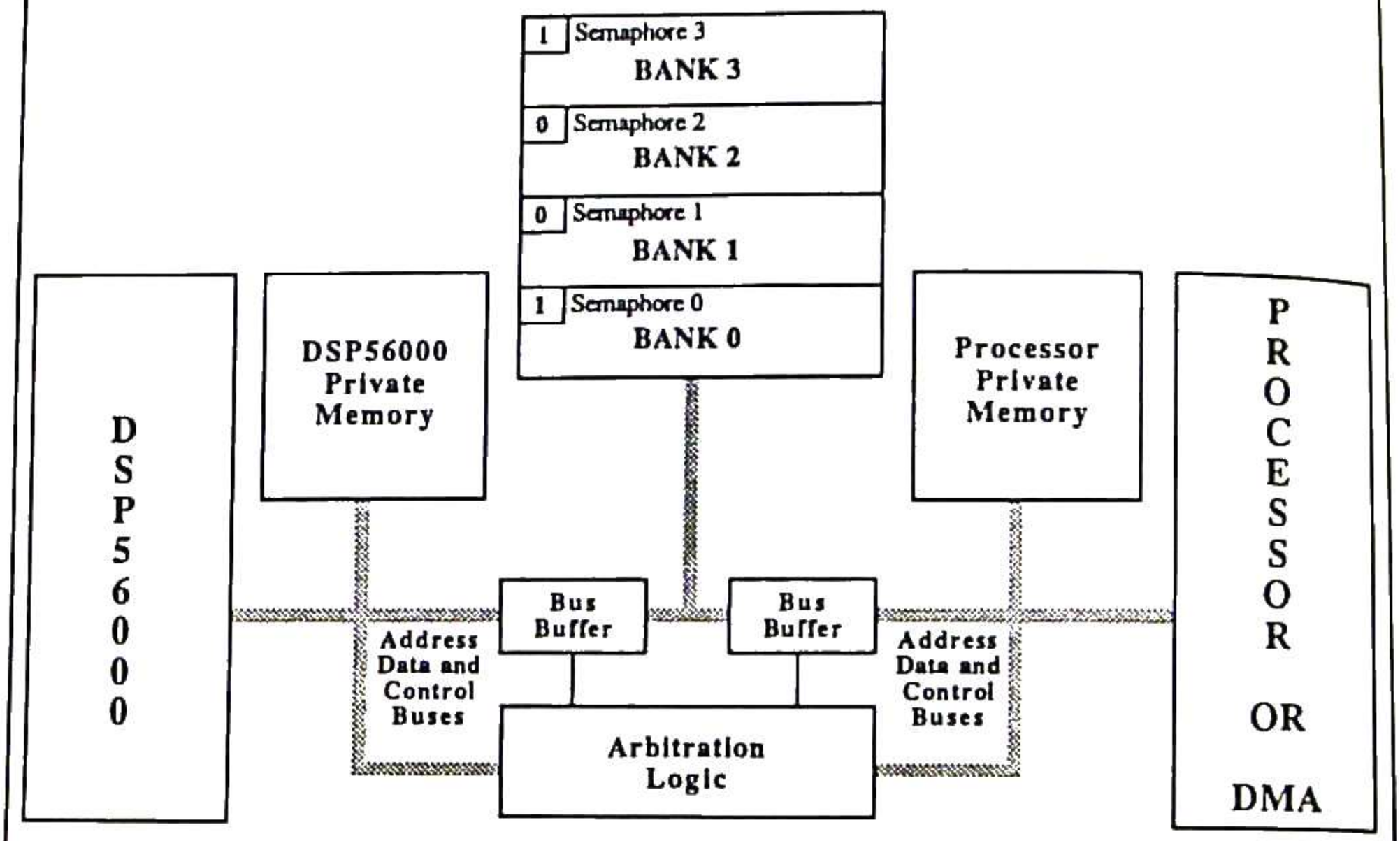


Port A – Bus Control Register (BCR)



NCD02-0325

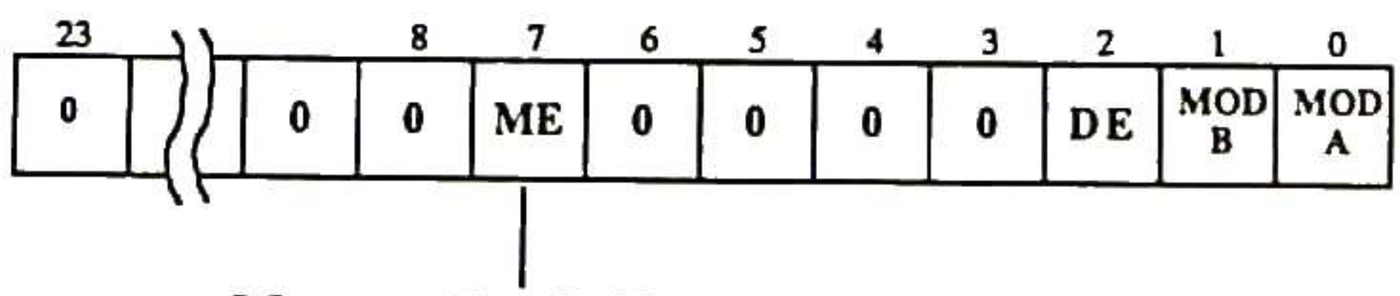
SHARED MEMORY SYSTEM



NCD02-0326

MEMORY READY ENABLE BIT

OPERATING MODE REGISTER



Memory Ready Enable

0 = Bus Request /Bus Grant Arbitration (RESET)

USED BY A HOST TO TRI-STATE THE DSP56000 EXTERNAL BUS

1 = Memory Ready/Memory Ready Strobe

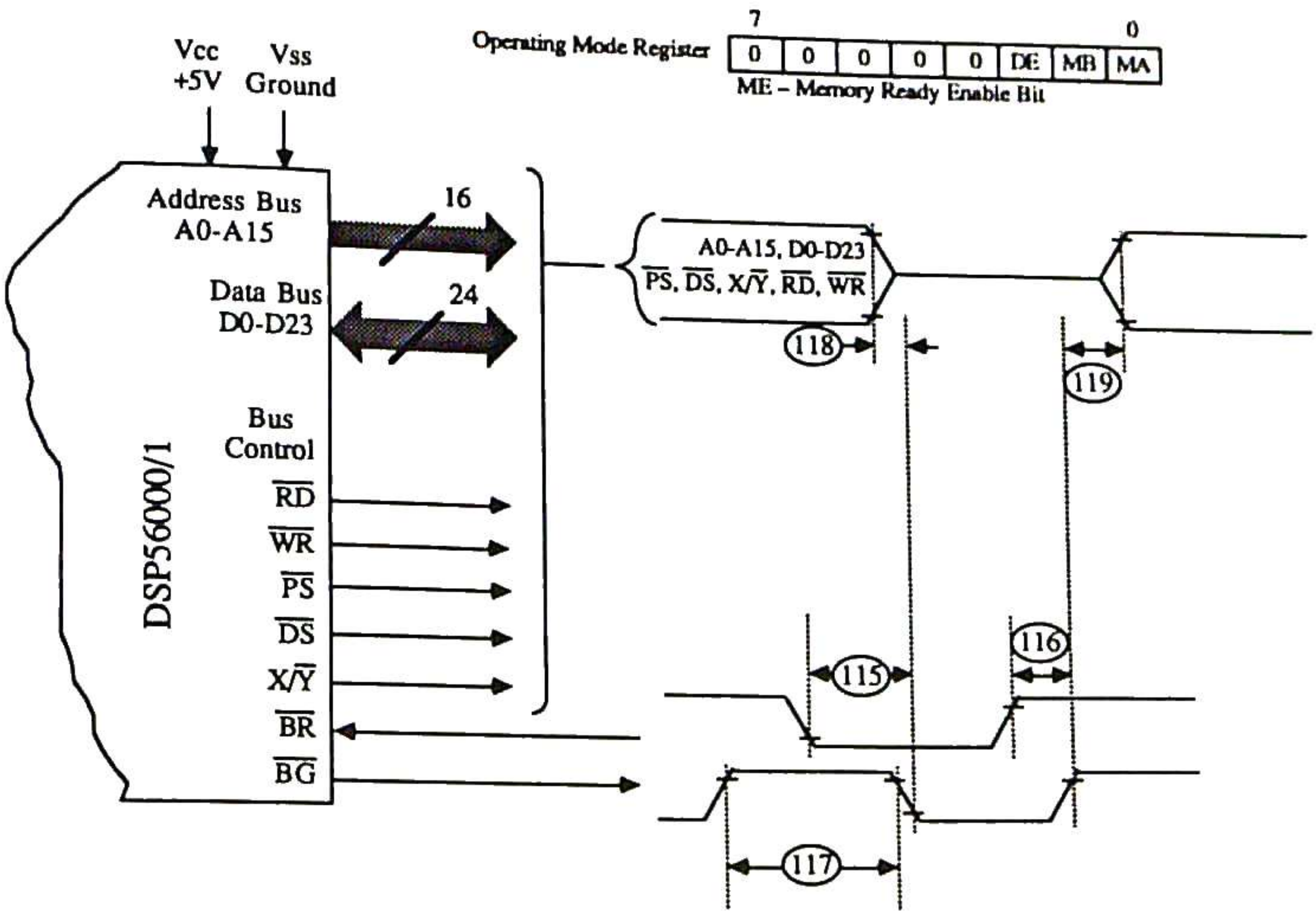
USED BY A HOST TO DELAY OR STRETCH THE CURRENT DSP56000 BUS CYCLE

THE TWO SYSTEMS ARE MUTUALLY EXCLUSIVE

All Control Bits in the OMR are Software Programmable

NCD02-0327

BUS ARBITRATION TIMING

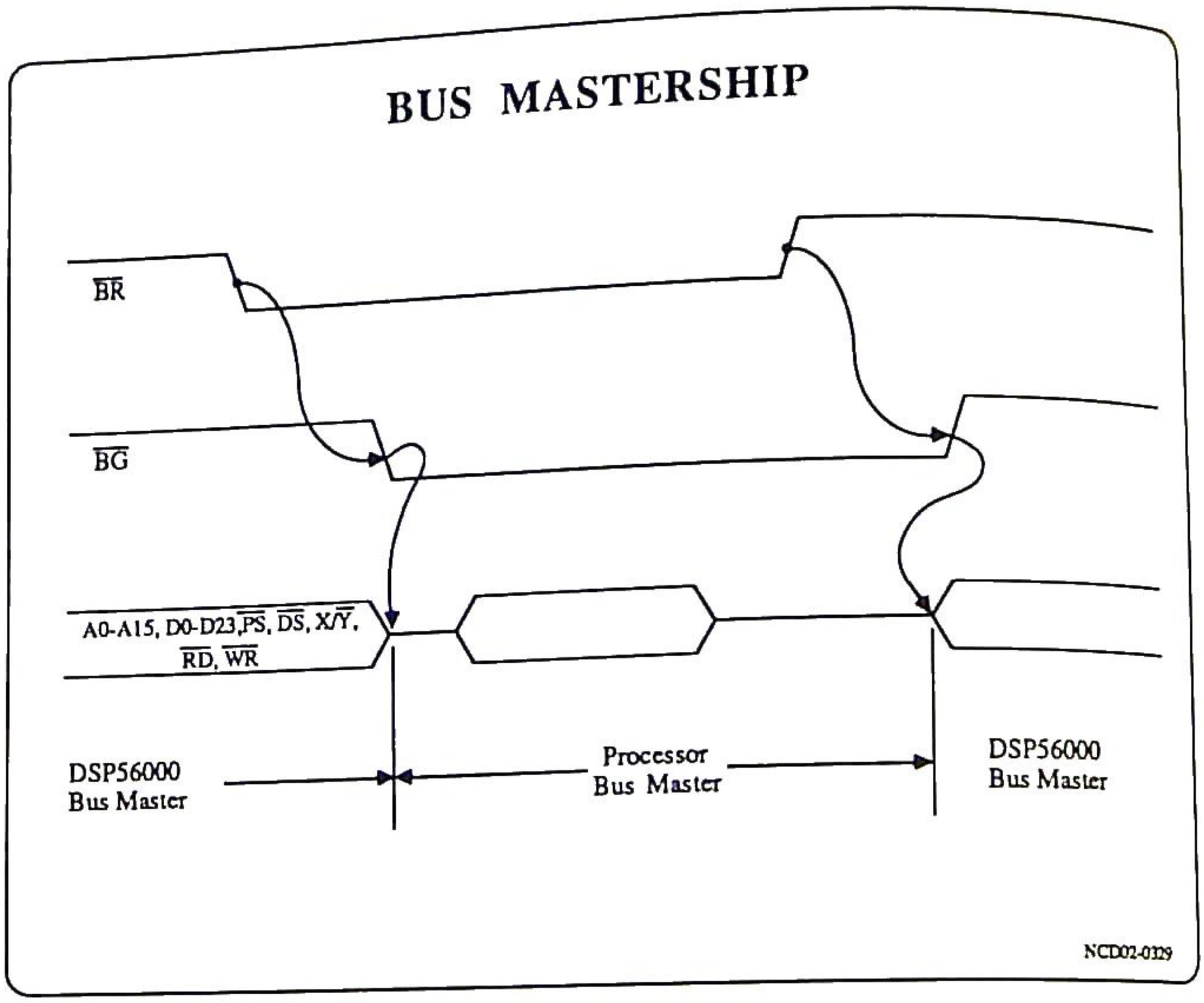


Num	Characteristics		Min	Max	Unit
115	Delay from $\overline{BR}$ Assertion to $\overline{BG}$ Assertion	(See Note 1)	3T	9T+20	ns
		(See Note 2)	3T	9T+2T•WS+20	
		(See Note 3)	3T	13T+4T•WS+20	
		(See Note 4)	Infinity	–	
		(See Note 5)	Infinity	–	
116	Delay from $\overline{BR}$ Deassertion to $\overline{BG}$ Deassertion		5T	9T+20	ns
117	$\overline{BG}$ Deassertion Duration		4T–10	–	ns
118	Delay from Address, Data, and Control Bus High Impedance to $\overline{BG}$ Assertion		0	–	ns
119	Delay from $\overline{BG}$ Deassertion to Address, Data, and Control Bus Enabled		–	-10	ns

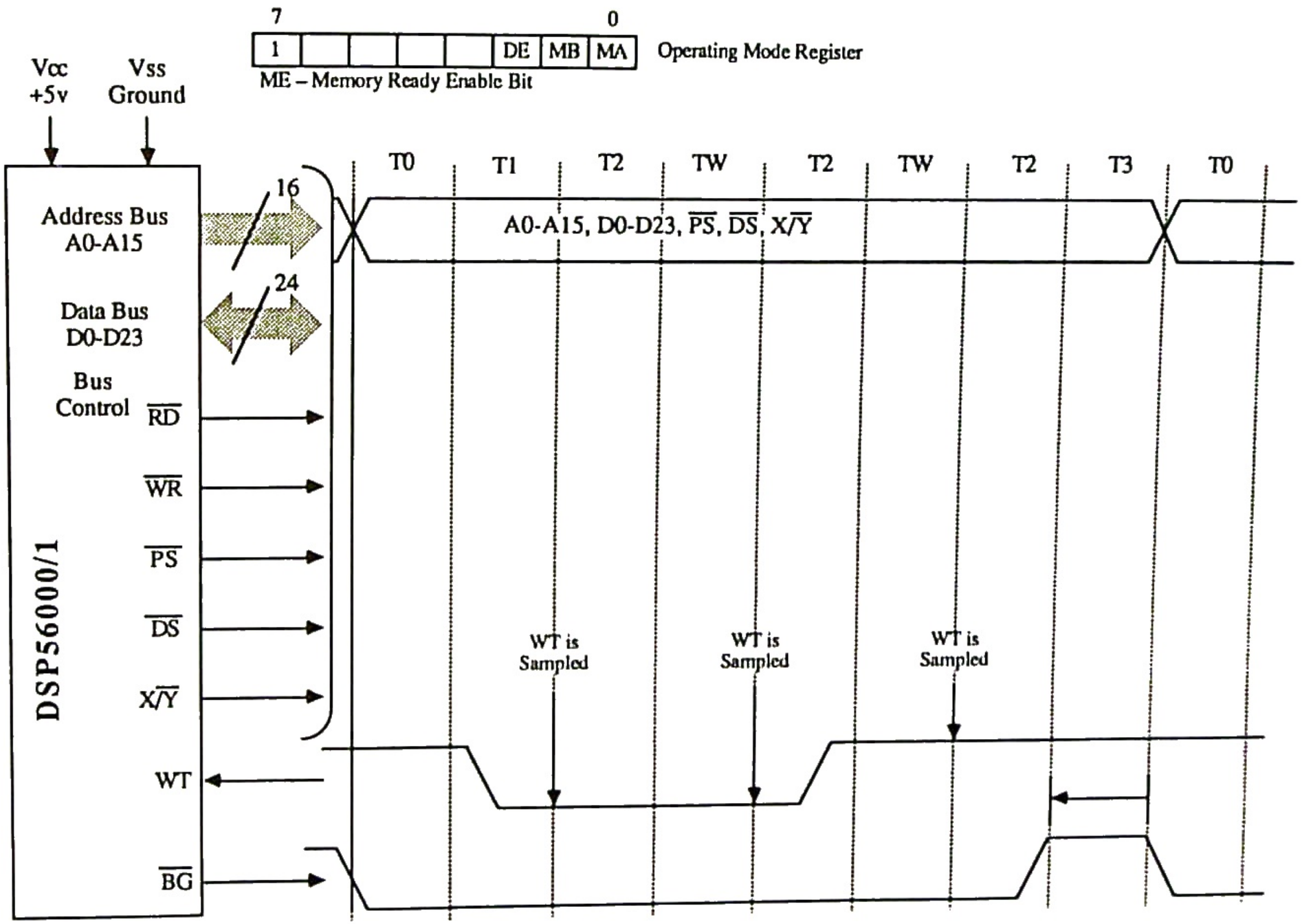
NOTES:

1. With no external access from the DSP.
2. During external read or write access.
3. During external read-modify-write access.
4. During STOP mode – external bus will not be released and $\overline{BG}$ will not go low.
5. During WAIT mode – external bus will not be released and $\overline{BG}$ will not go low.

NCD02-0328-1

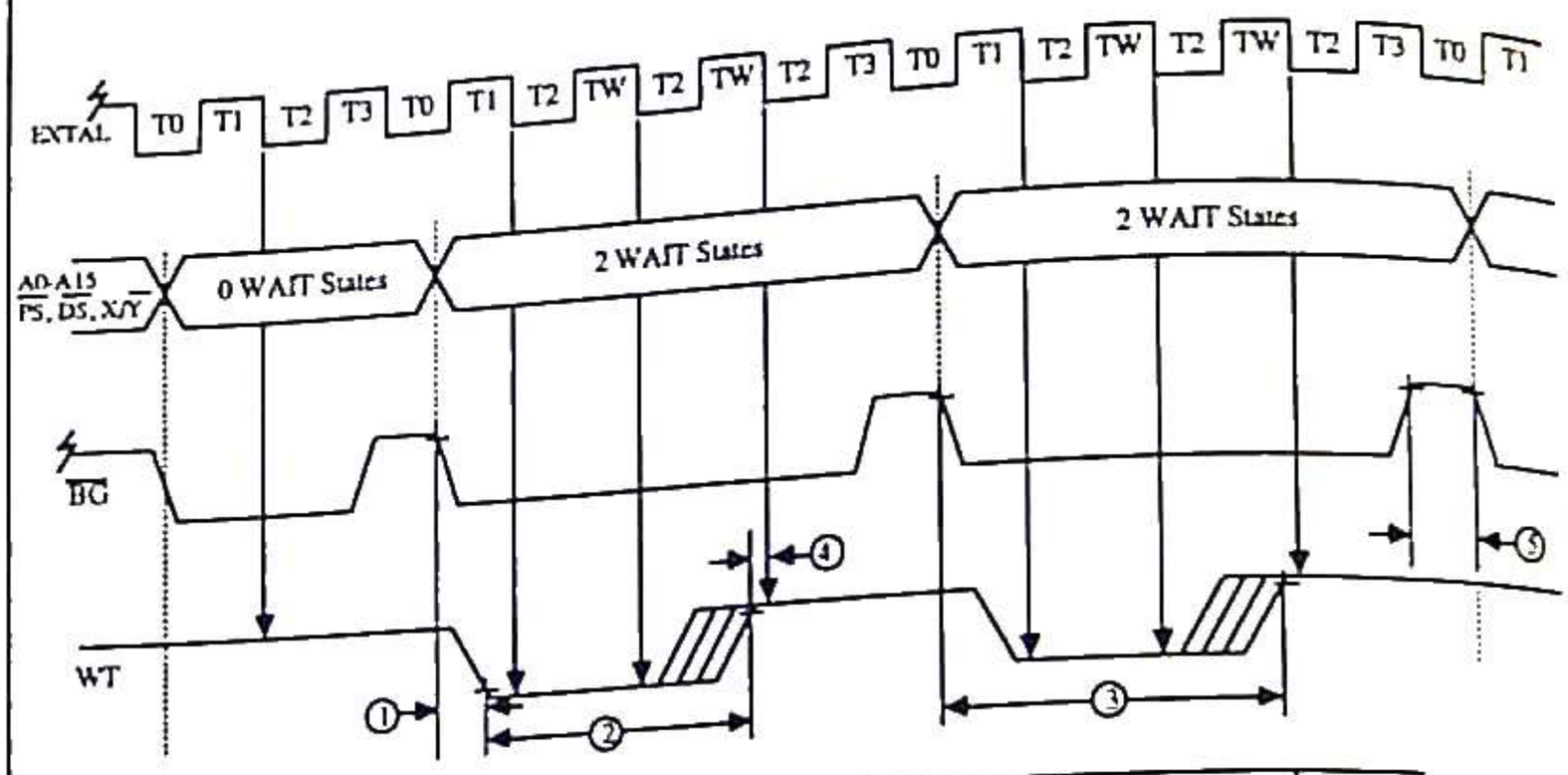

BUS MASTERSHIP
BR
BG
A0-A15, D0-D23, PS, DS, X/Y,
RD, WR
DSP56000
Bus Master
Processor
Bus Master
DSP56000
Bus Master
NCD02-0329

MEMORY READY



NCD02 0330-4

MEMORY READY TIMING



Num	Characteristic	Min	Max	Units
1	$\overline{BG}$ Low to WT Low	–	2T-30	ns
2	$\overline{BG}$ Low to WT High	2T	–	ns
3	$\overline{BG}$ Low to WT High (2WS)	–	4T-30	ns
4	WT High to EXTAL falling edge setup time for synchronous operation	5	2T-20	ns
5	$\overline{BG}$ Minimum High	T–7	–	ns

NCD02-0331-2

MEMORY READY VS. THE BUS CONTROL REGISTER

Bus Control Register Value	WT Input Value	$\overline{WR}$ Falling Edge Occurs In	Number of Wait States
0	HIGH	T1	0
0	LOW	T1	Minimum of 1
1 to 15	HIGH	T2	Value Programmed in the B.C.R.
1 to 15	LOW	T2	1 Min. or Value in the B.C.R.

NCD02-0332-2

DSP56000/1
ADDRESSING MODES

Objectives:

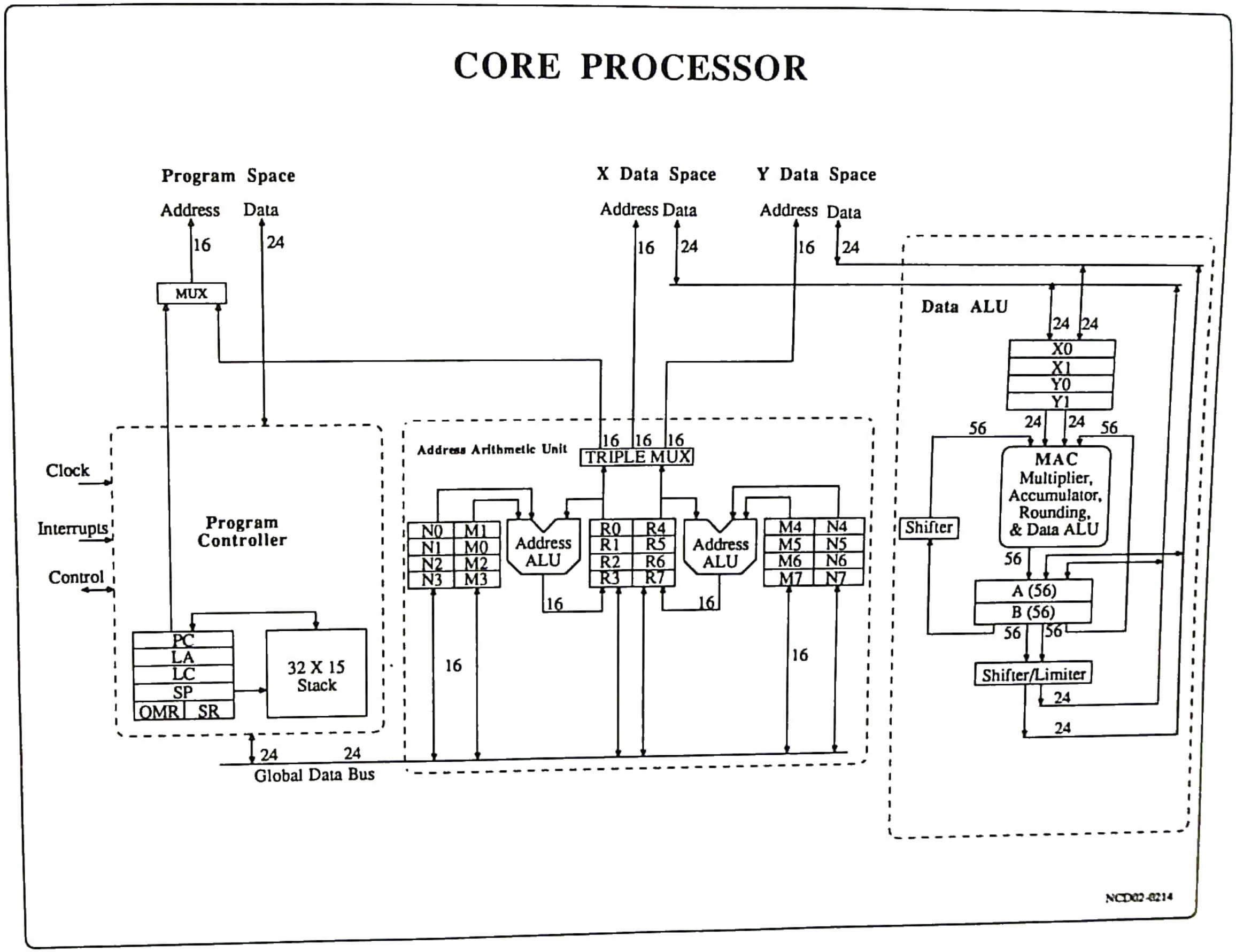
- Describe the function of the Address Arithmetic Unit and the basic MOVE instruction.
- Describe the Special Addressing Modes.
- Describe the Address Register Indirect.
- Describe the functions of the Modifier Registers.
  - Modulo Addressing
  - Reverse Carry Addressing

NCD02-0400-1

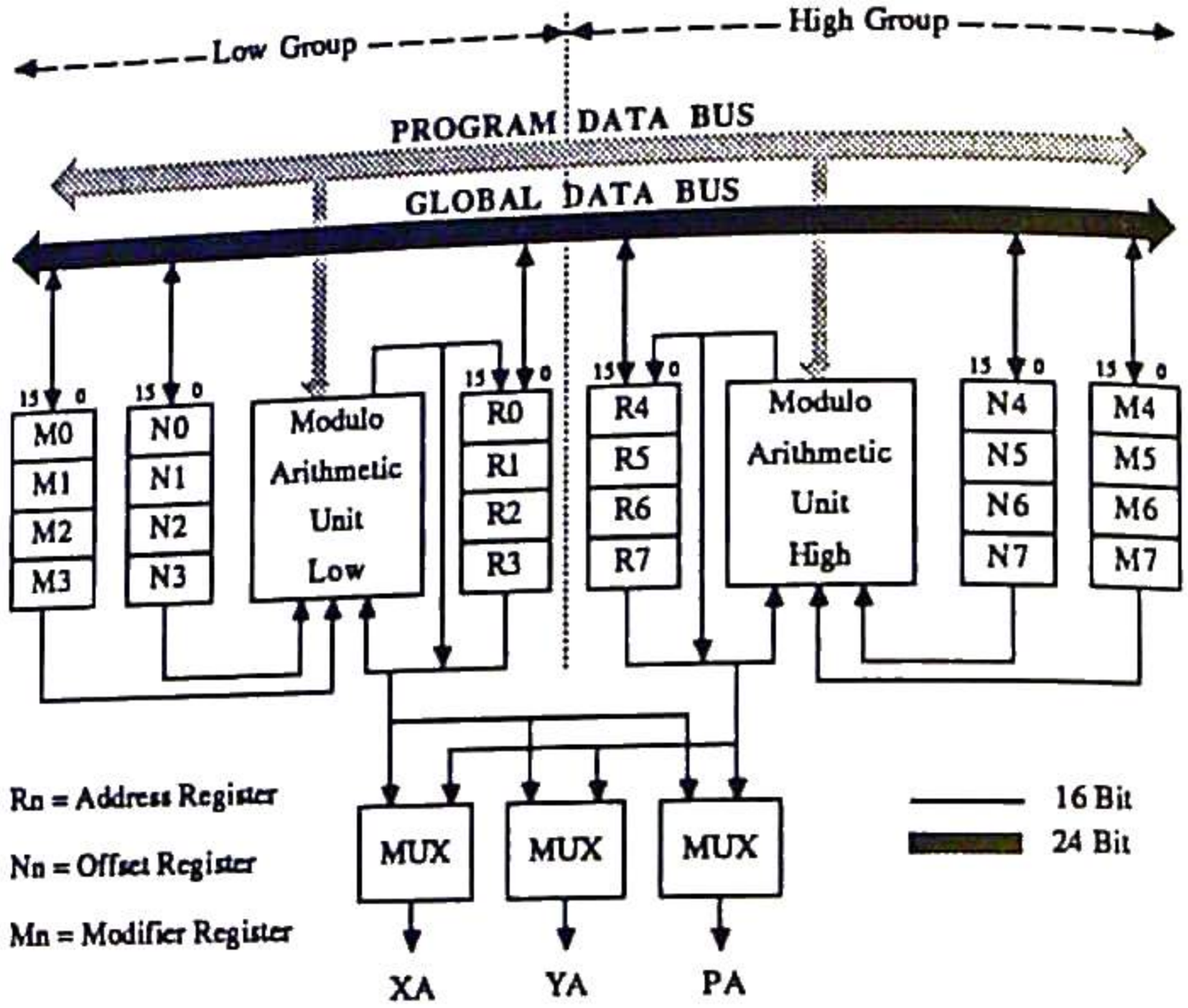
Table of Contents

Core Processor 4-5
Address Arithmetic Unit 4-6
AAU Programming Model 4-7
Operands I 4-8
Operands II 4-9
Addressing Modes 4-10
Addressing Categories 4-11
Basic MOVE Instruction 4-12
Register Direct Addressing 4-12
Special Addressing Modes 4-13
Implicit 4-13
Immediate Data 4-14
Immediate Data Into Accumulator 4-14
Immediate Short 4-15
Immediate Short Into Accumulator 4-15
Absolute Long 4-16
Absolute Short Data 4-16
Short Jump 4-17
I/O Short 4-18
Address Register Indirect Addressing Modes 4-18
No Update 4-19
Postincrement by One 4-19
Postdecrement by One 4-20
Postincrement by Nn 4-20
Postdecrement by Nn 4-21
Predecrement by One 4-21
Indexed by Offset Nn 4-22
Modifier Registers 4-23
Modifier Register Values 4-23
Modulo M Addressing Applications 4-24
Circular Queue 4-24
Modulo M Register Setup 4-25
Modulo M Addressing Example 4-25
Fast Fourier Transform Butterfly 4-26
8 Point FFT Transform 4-26
Bit Reversed Register Setup for N Point FFT 4-27
Bit Reversed / Reverse Carry Algorithm 4-27
Bit Reverse Addressing Example 4-28

CORE PROCESSOR

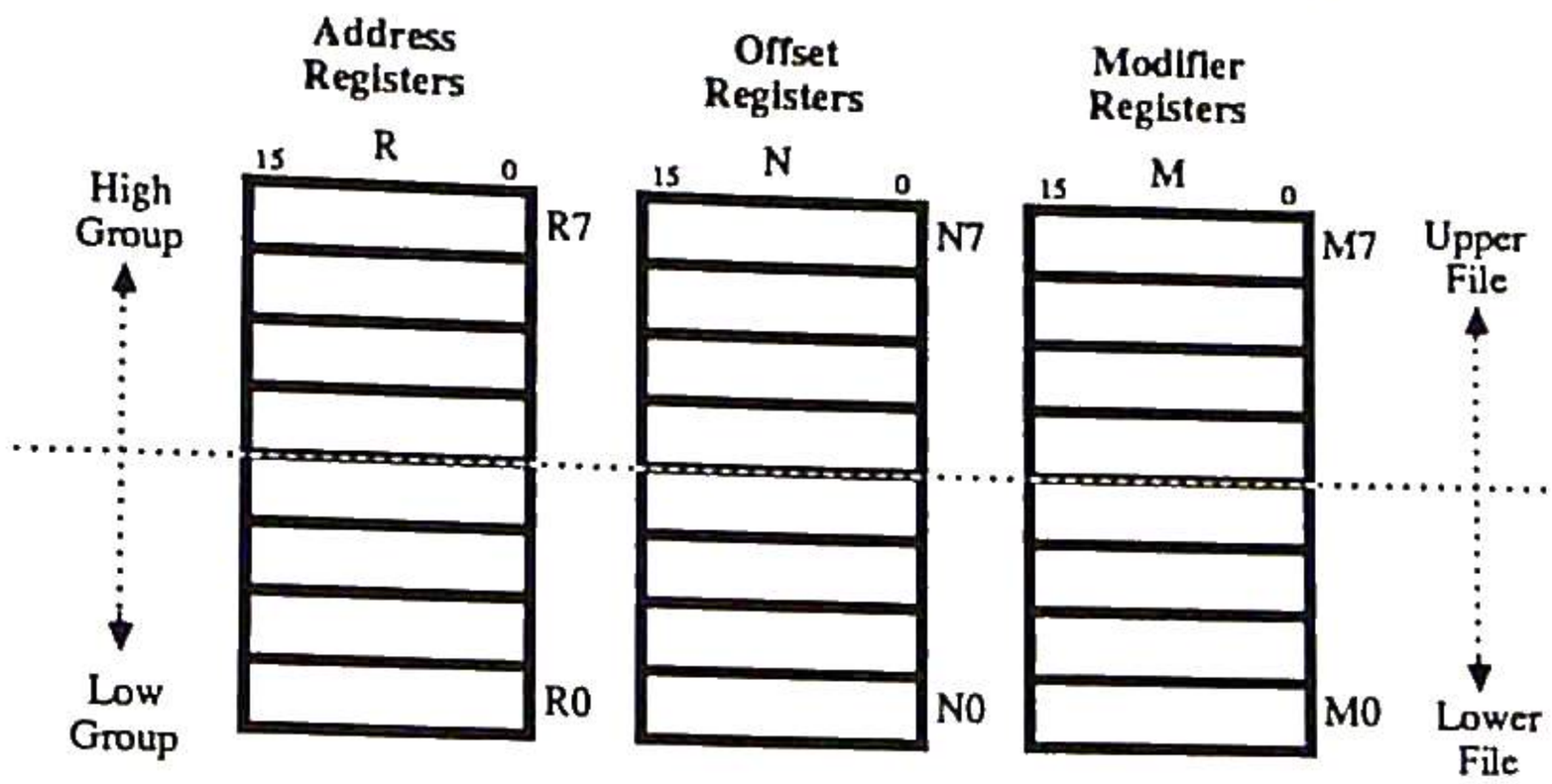


ADDRESS ARITHMETIC UNITS (AAU)



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

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.
- When Calculating Two Effective Addresses one must come from the lower file and the second must come from the upper file.
- When read over the 24 bit Global Data Bus, the register contents are zero extended.
- Due to pipelining, if an A.A.U. register (R,N,M) is changed with a MOVE instruction, the new contents will not be available for use as a pointer until the second following instruction. The assembler flags this with a warning.

NCD02-0404

OPERANDS I

Program Controller Registers

PC	Program Counter Register (16 Bits)
MR	Mode Register (8 Bits)
CCR	Condition Code Register (8 Bits)
SR	Status Register = MR:CCR (16 Bits)
OMR	Operating Mode Register (8 Bits)
LA	Hardware Loop Address Register (16 Bits)
LC	Hardware Loop Counter Register (16 Bits)
SP	System Stack Pointer Register (6 Bits)
SSH	Upper Portion of the Current Top of the Stack (16 Bits)
SSL	Lower Portion of the Current Top of the Stack (16 Bits)
SS	System Stack RAM = SSH:SSL (15 Locations by 32 Bits)

Data ALU Registers

Xn	Input Registers X1 or X0 (24 Bits)
Yn	Input Registers Y1 or Y0 (24 Bits)
An	Accumulator Registers A2, A1, A0 (A2 – 8 Bits, A1 and A0 –24 Bits)
Bn	Accumulator Registers B2, B1, B0 (B2 – 8 Bits, B1 and B0 –24 Bits)
X	Input Register X = X1:X0 (48 Bits)
Y	Input Register Y = Y1:Y0 (48 Bits)
A	Accumulator A = A2:A1:A0 (56 Bits) *
B	Accumulator B = B2:B1:B0 (56 Bits) *
AB	Accumulators A and B = A1:B1 (48 Bits) *
BA	Accumulators B and A = B1:A1 (48 Bits) *
A10	Accumulator A = A1:A0 (48 Bits)
B10	Accumulator B = B1:B0 (48 Bits)

*Note: In data move operations, shifting and limiting is performed when this register is specified as a source operand. When specified as a destination operand, sign extension and possibly zeroing are performed.

Address ALU Registers

Rn	Address Registers R0 thru R7 (16 Bits)
Nn	Address Offset Registers N0 thru N7 (16 Bits)
Mn	Address Modifier Registers M0 thru M7 (16 Bits)

NCD02-0405-1

OPERANDS II

Mode Register (MR) Symbols

LF	Loop Flag Bit Indicating When a DO Loop is in Progress
T	Trace Mode Bit Indicating if the Tracing Function has been Enabled
S1, S0	Scaling Mode Bits Indicating the Current Scaling Mode
I1, I0	Interrupt Mask Bits Indicating the Current Interrupt Priority Level

Condition Code Register (CCR) Symbols (standard definitions)

L	Limit Bit Indicating Arithmetic Overflow and/or Data Shifting/Limiting
E	Extension Bit Indicating if the Integer Portion of A or B is in Use
U	Unnormalized Bit Indicating if the A or B Result is Unnormalized
N	Negative Bit Indicating if Bit 55 of the A or B Result is Set
Z	Zero Bit Indicating if the A or B Result Equals Zero
V	Overflow Bit Indicating if Arithmetic Overflow has Occurred in A or B
C	Carry Bit Indicating if a Carry or Borrow Occurred in A or B Result

Address Operands

ea	Effective Address
eax	Effective Address for X Bus
eay	Effective Address for Y Bus
xxxx	Absolute Address (16 Bits)
xxx	Short Jump Address (12 Bits)
aa	Absolute Short Address (6 Bits, Zero Extended)
pp	I/O Short Address (6 Bits, One's Extended)
<...>	Specifies the Contents of The Specified Address
X:	X Memory Reference
Y:	Y Memory Reference
L:	Long Memory Reference = X:Y
P:	Program Memory Reference

Miscellaneous Operands

S,Sn	Source Operand Register
D,Dn	Destination Operand Register
D[n]	Bit N of D Destination Operand Register
#n	Immediate Short Data (5 Bits)
#xx	Immediate Short Data (8 Bits)
#xxx	Immediate Short Data (12 Bits)
#xxxxxx	Immediate Data (24 Bits)

NCD02-0406-1

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					

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

NCD02-047

ADDRESSING CATEGORIES

Addressing Mode	Mode MMM	Reg RRR	Addressing Categories U	P	M	A	Assembler Syntax
Register Direct							
Data or Control Register	–	–				X	
Address Register	–	–				X	Rn
Address Offset Register	–	–				X	Nn
Address Modifier Register	–	–				X	Mn
Address Register Indirect							
No Update	100	Rn		X	X	X	(Rn)
Postincrement by 1	011	Rn	X	X	X	X	(Rn)+
Postdecrement by 1	010	Rn	X	X	X	X	(Rn)-
Postincrement by Offset Nn	001	Rn	X	X	X	X	(Rn)+Nn
Postdecrement by Offset Nn	000	Rn	X		X	X	(Rn)-Nn
Indexed by Offset Nn	101	Rn			X	X	(Rn+Nn)
Predecrement by 1	111	Rn			X	X	-(Rn)
Special							
Immediate Data	110	100			X		#xxxxxx
Absolute Address	110	000			X	X	xxxx
Immediate Short Data	–	–					#xx
Short Jump Address	–	–				X	xxx
Absolute Short Address	–	–				X	aa
I/O Short Address	–	–				X	pp
Implicit	–	–				X	

Update Mode (U) — The Update Addressing mode is used to modify address registers without any associated data move.

Parallel Mode (P) — The Parallel Addressing mode is used in instructions where two effective addresses are required.

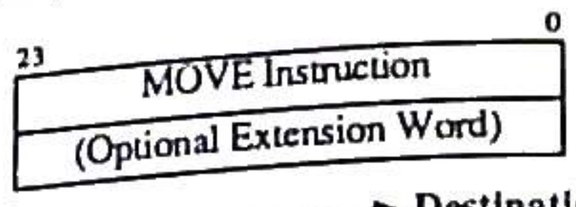
Memory Mode (M) — The Memory Addressing mode is used to refer to operands in memory using an effective addressing field.

Alterable Mode (A) — The Alterable Addressing mode is used to refer to alterable or writable registers or memory.

NCD02-0408

BASIC MOVE INSTRUCTION

MOVE Source 1, Destination 1 [Source 2, Destination 2]



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	

- MEMORY to MEMORY MOVES not supported (Except for MOVEP)

NCD02-045-1

REGISTER DIRECT ADDRESSING

- Directly Addressing

Data Registers

X, X1, X0
Y, Y1, Y0
A, A2, A1, A0
B, B2, B1, B0

Address Registers

R0-R7
N0-N7
M0-M7

Control Registers

SR, MR, CCR
SP
SSH, SSL
LA, LC
OMR

- Modifier Registers do not affect Register Direct Addressing

NCD02-045-1

SPECIAL ADDRESSING MODES

- Implicit
- Immediate Data
- Immediate Short Data
- Absolute Address
- Absolute Short Address
- Short Jump Address
- I/O Short Address
- Modifier Registers Do Not Affect Special Addressing Modes

NCD02-0411-1

IMPLICIT

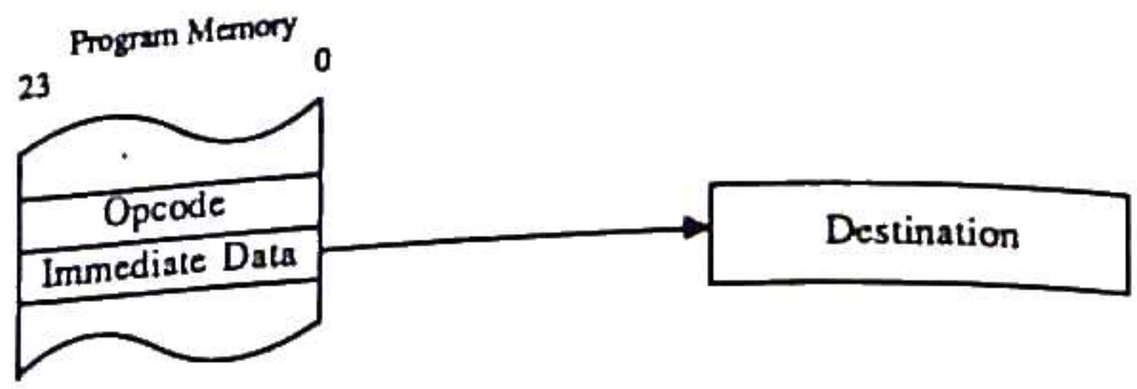
- Reference is embedded in the instruction
- Instruction Dependent
- Refers to "ALTERABLE" Registers or Memory
  - System Stack Reference
  - Program Controller Register Reference
  - Program Memory Reference

NCD02-0412

IMMEDIATE DATA

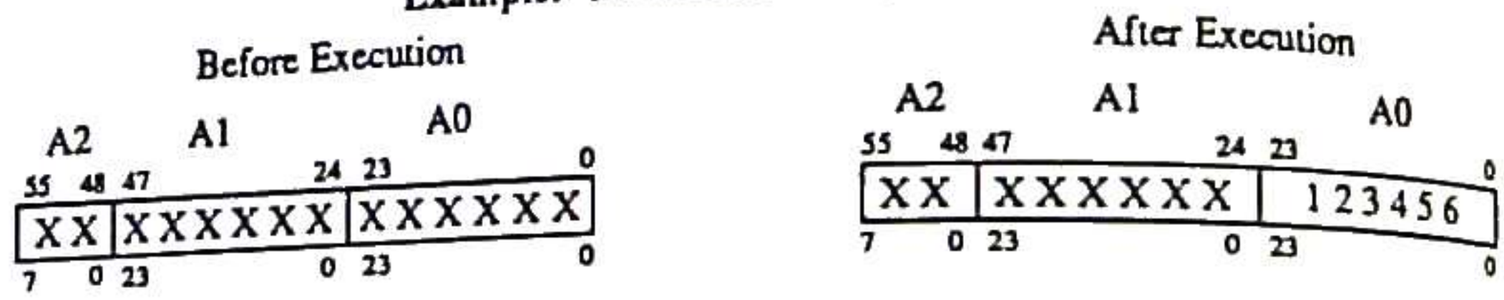
Assembler Syntax: #xxxxxx

Operands Referenced: P Memory



24 Bit Destination

Example: MOVE #$123456,A0

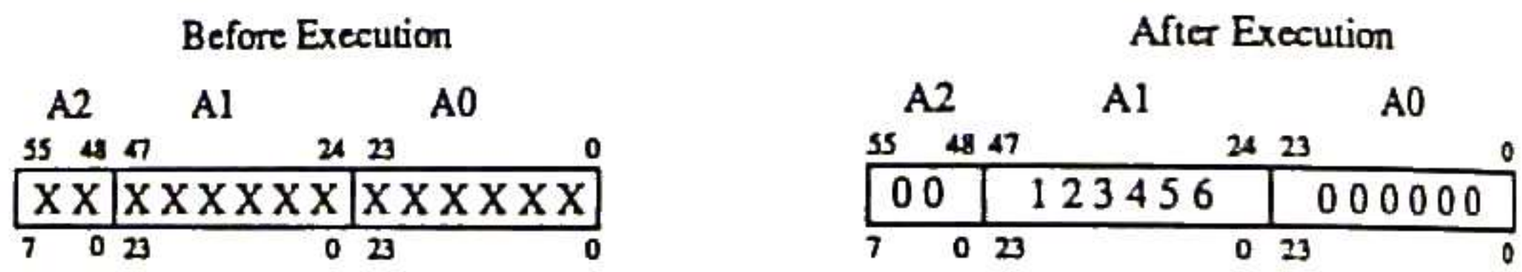


NCDD 0413

IMMEDIATE DATA INTO ACCUMULATOR

Immediate into 56 bit Accumulator

Example: MOVE #$123456,A



Negative Immediate into 56 bit Accumulator

Example: MOVE #$801234,A

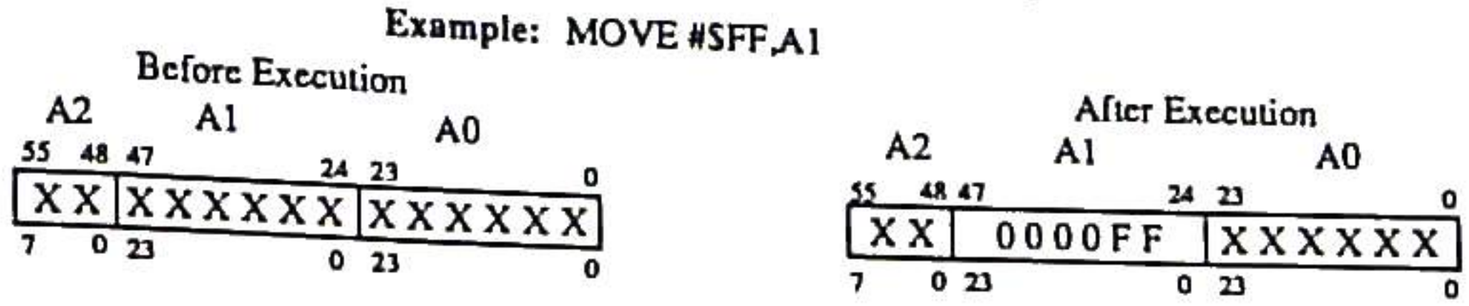
NCDD 0414

IMMEDIATE SHORT

Eight Bit Data
Assembler Syntax: #xx
Operands Referenced: P Memory

If Destination is A2, A1, A0, B2, B1, B0, R0-R7, N0-N7
then 8 bit Immediate Short Operand is treated as Unsigned Integer.

Example: MOVE #$FF,A1



If Destination is X1, X0, Y1, Y0, A,B
then 8 bit Immediate Short Operand is treated as Signed Fraction.

Positive Immediate into 24 bit register
Example: MOVE #$1F,Y1

NCD02-0415

IMMEDIATE SHORT INTO ACCUMULATOR

8 Bits, Sign Extended, Zero Filled

Positive Immediate into 56 bit accumulator
Example: MOVE #$1F,A

Negative Immediate into 56 bit accumulator
Example: MOVE #$83,B

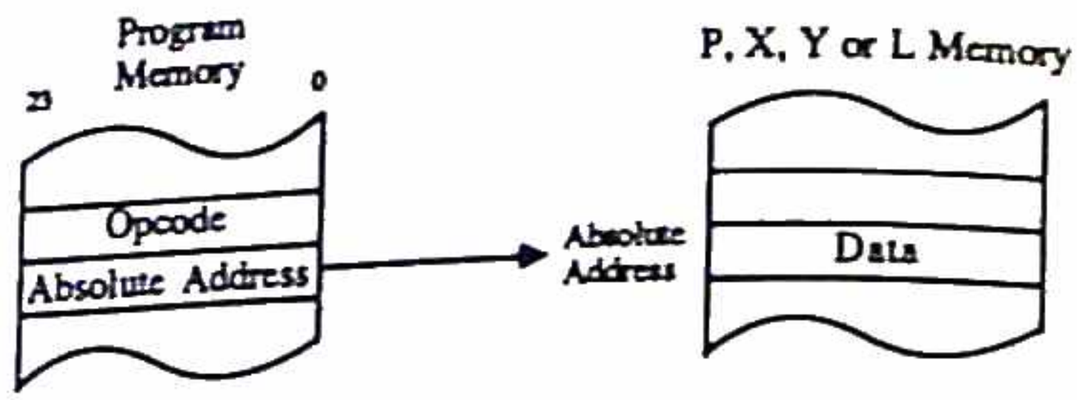
NCD02-0416

ABSOLUTE LONG

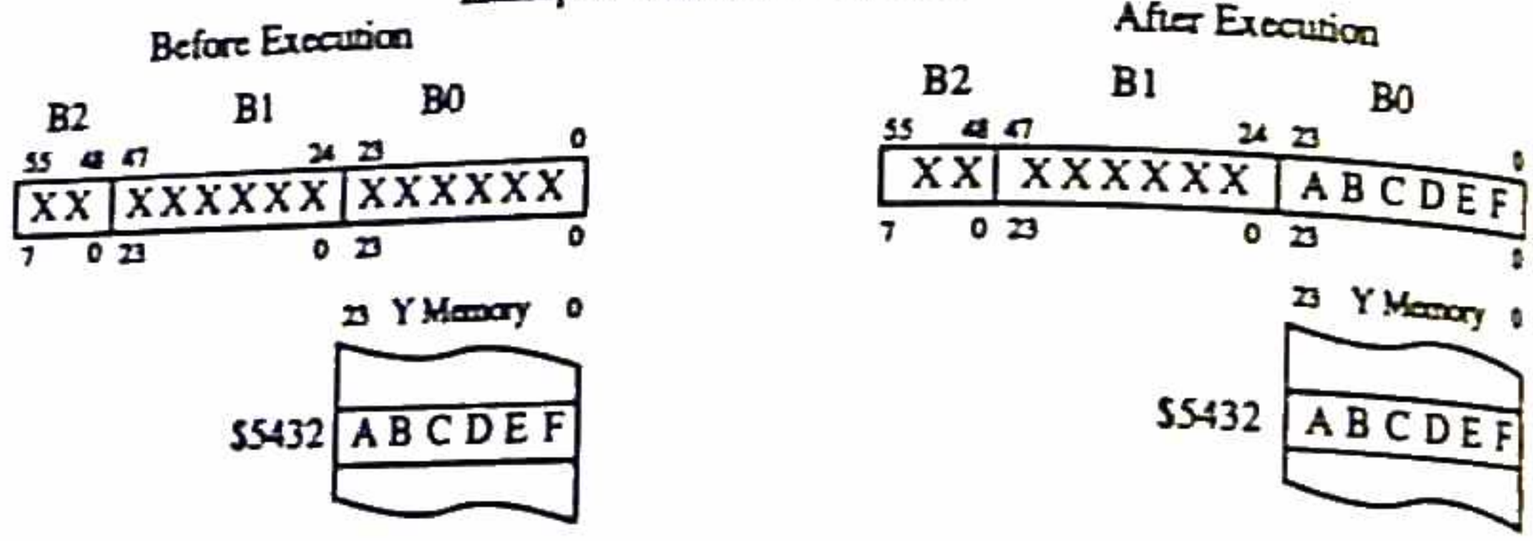
16 Bit Address

Assembler Syntax: xxxx

Operands Referenced: P, X, Y, L Memories



Example: MOVE Y:$5432,B0



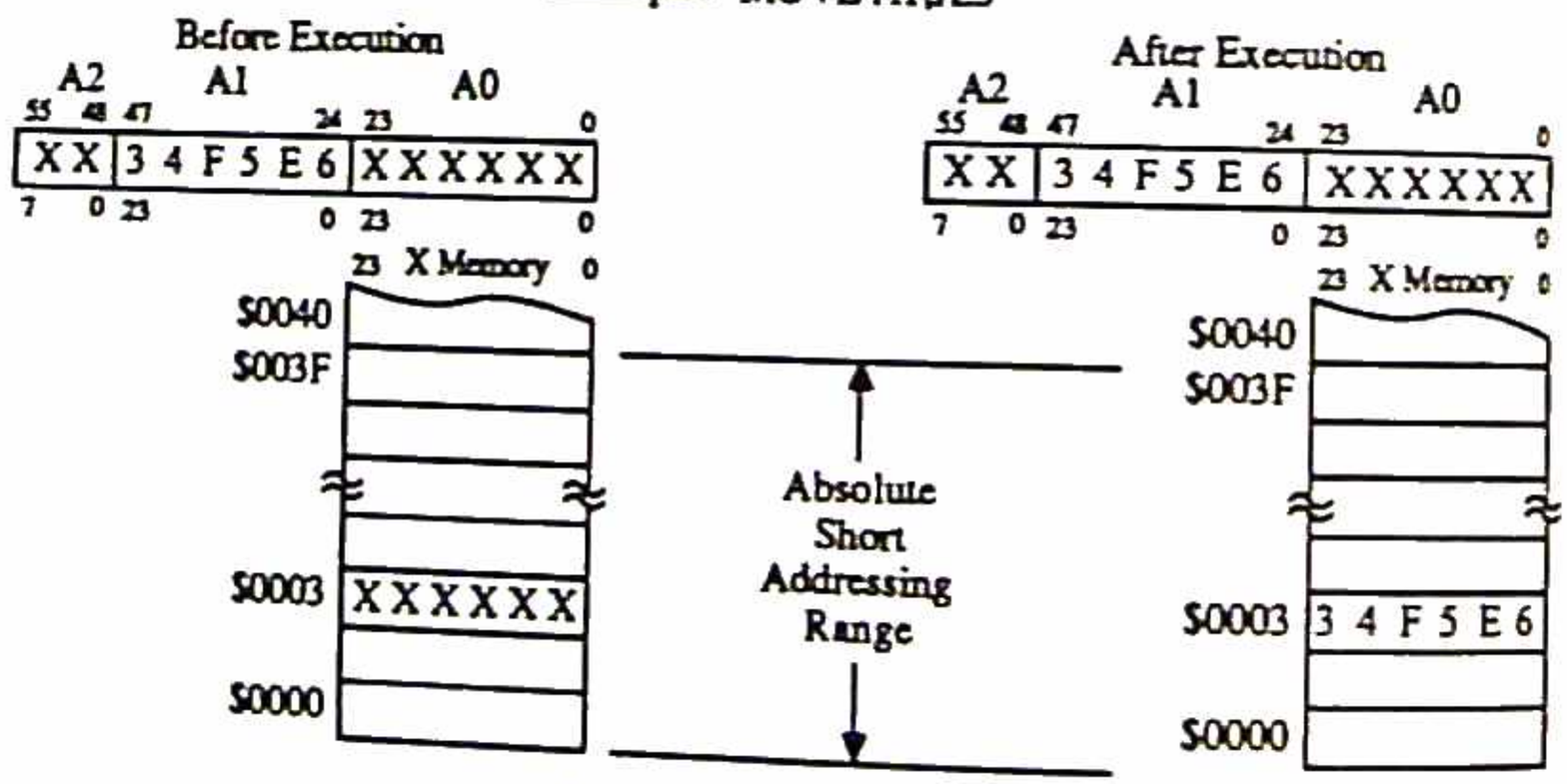
ABSOLUTE SHORT

6 Bits, Zero Extended

Assembler Syntax: aa

Operands Referenced: P, X, Y, L

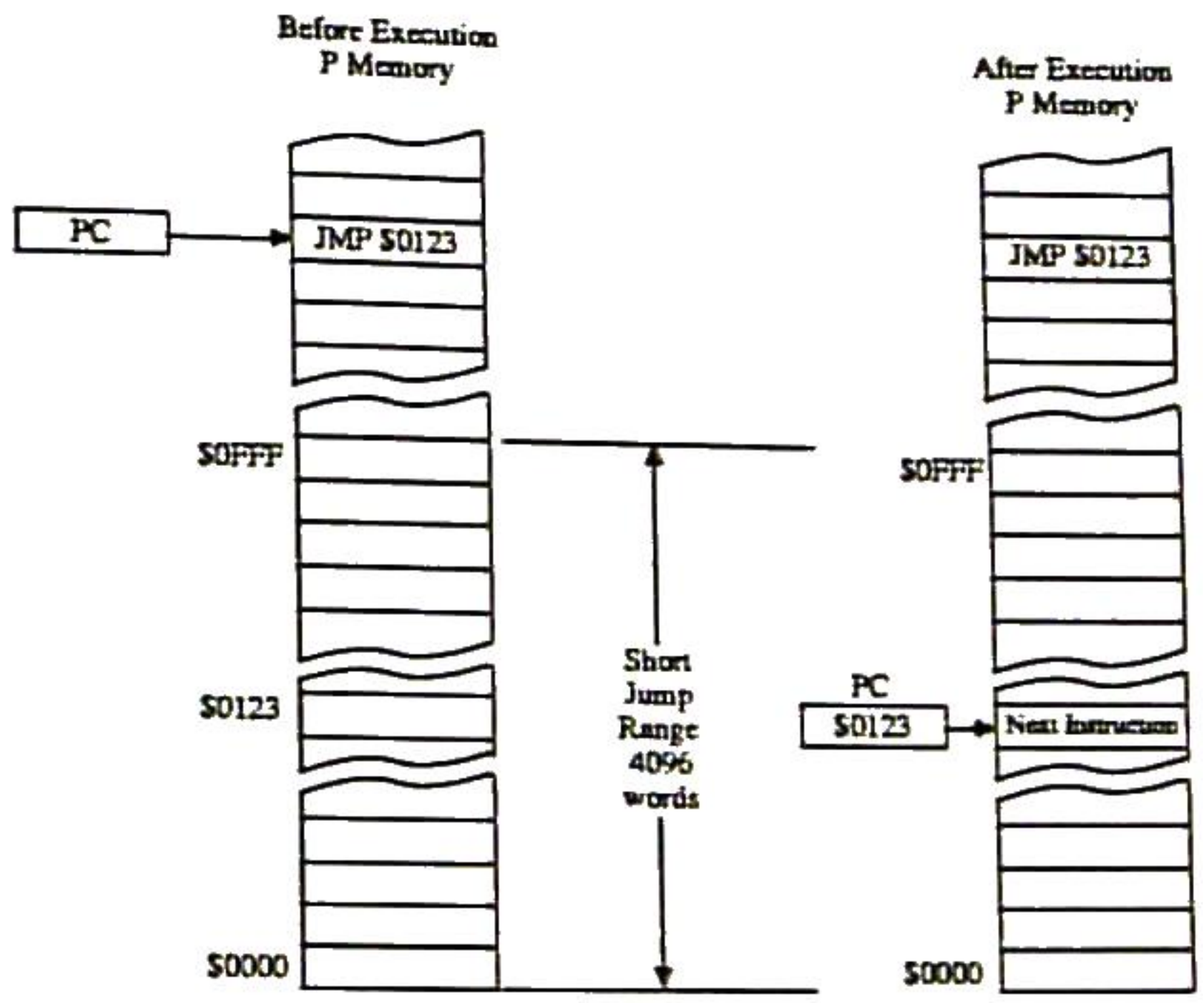
Example: MOVE A1,X:3



- Addresses Lowest 64 words
- Interrupt Vectors
- X, Y, L Data RAM
- No Extension Word
- One Word Instruction

SHORT JUMP

12 Bits, Zero Extended



- DSP56000 ROM P:$0000 – P:$0EFF
- DSP56001 RAM P:$0000 – P:$01FF
- Used by Conditional JUMPS and JUMP to Subroutines

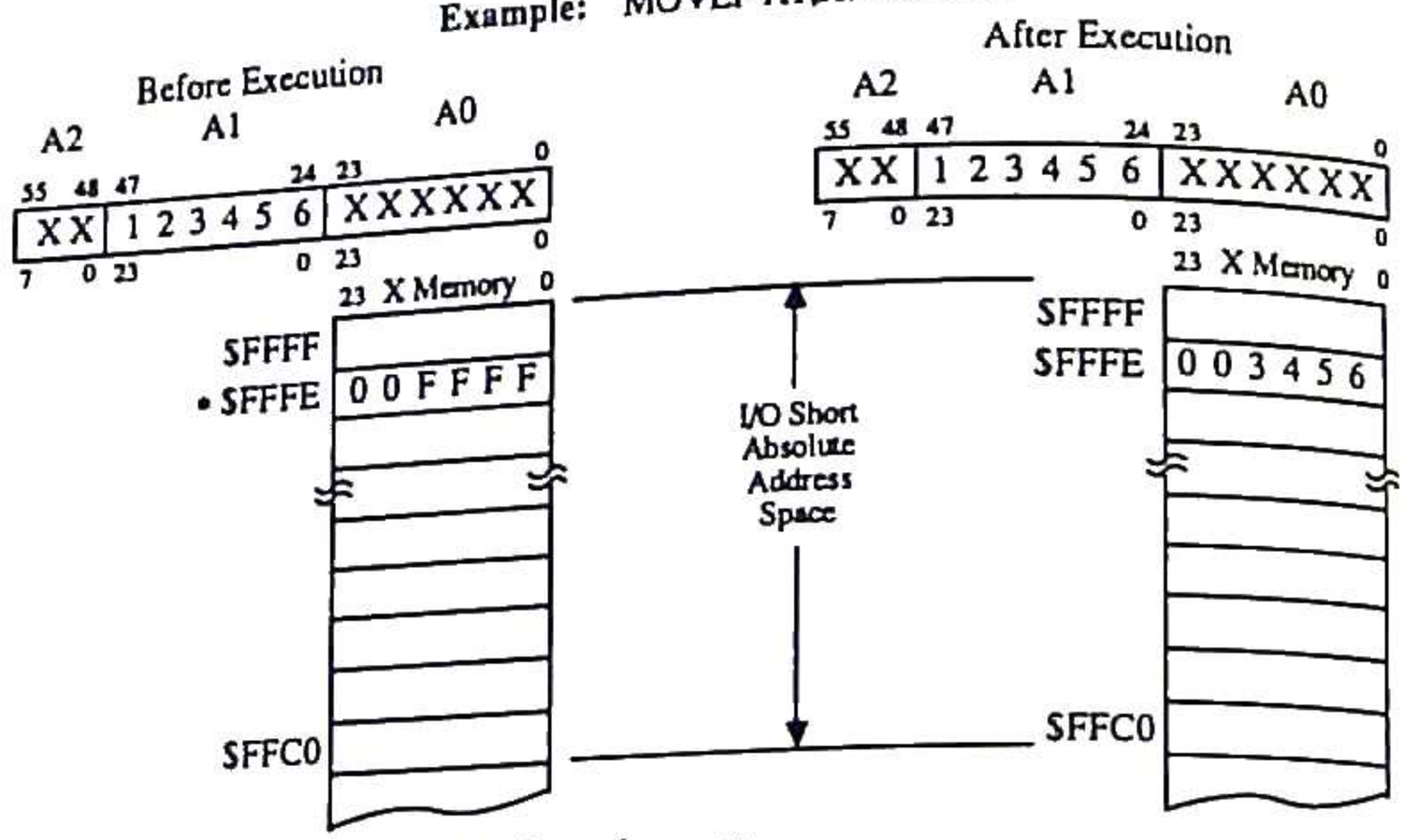
I/O SHORT

6 Bits, Ones Extended

Assembler Syntax: pp

Operands Referenced: X and Y, I/O Space

Example: MOVEP A1,X:<<$FFFE



• Contents of Bus Control Register after reset

• Upper 64 Words of X or Y Memory

• << indicates I/O short addressing

NCD02-0420-1

ADDRESS REGISTER INDIRECT ADDRESSING MODES

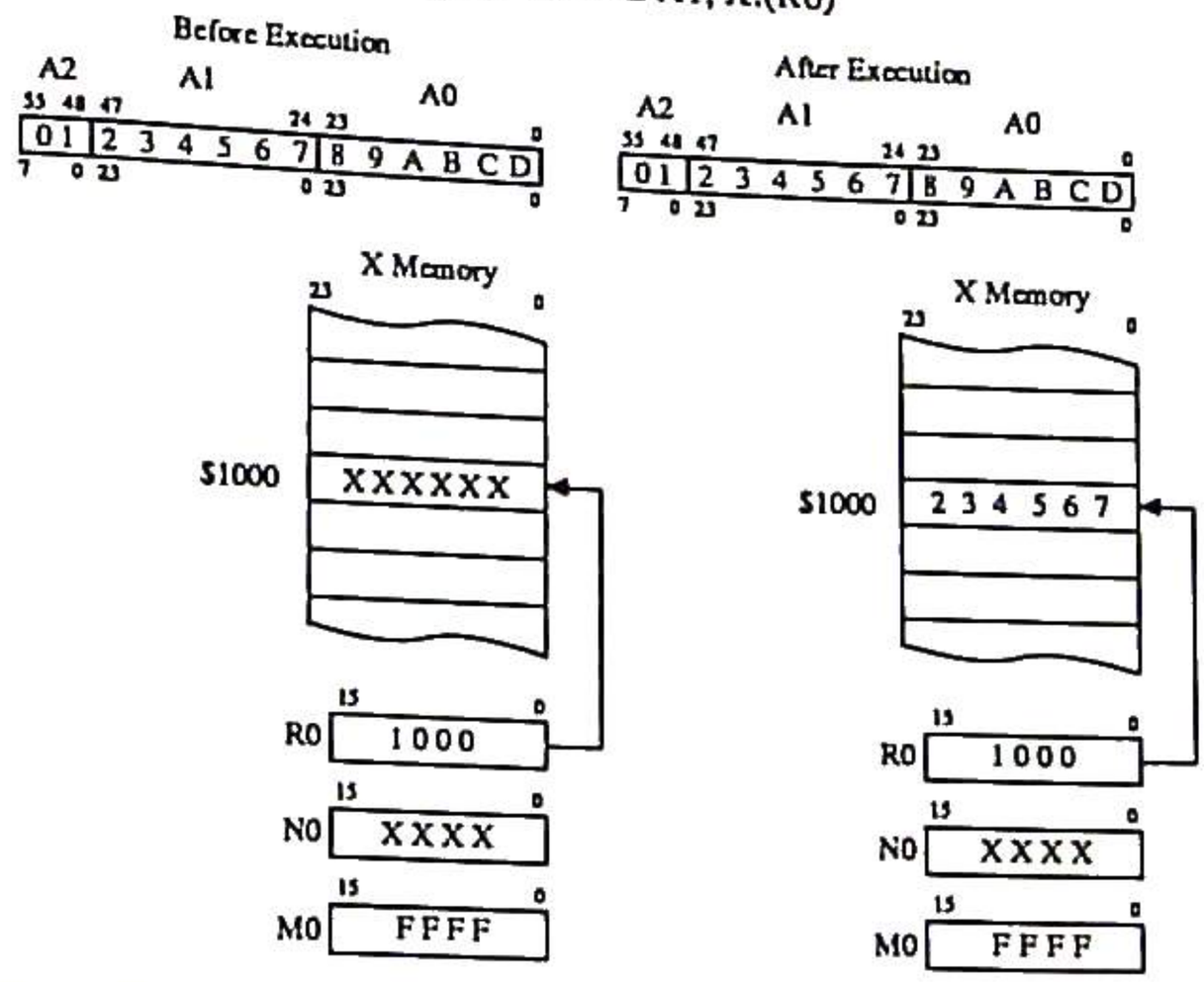
- No Update
- Postincrement by One
- Postincrement by Nn
- Postdecrement by One
- Postdecrement by Nn
- Indexed by Offset Nn
- Predecrement by 1
- All Modes of Address Register Indirect use the Mn Modifier Registers.

NCD02-0421

NO UPDATE

- Assembler Syntax: (Rn)
- Operands referenced: P, X, Y, XY, L memories.

Example: MOVE A1, X:(R0)

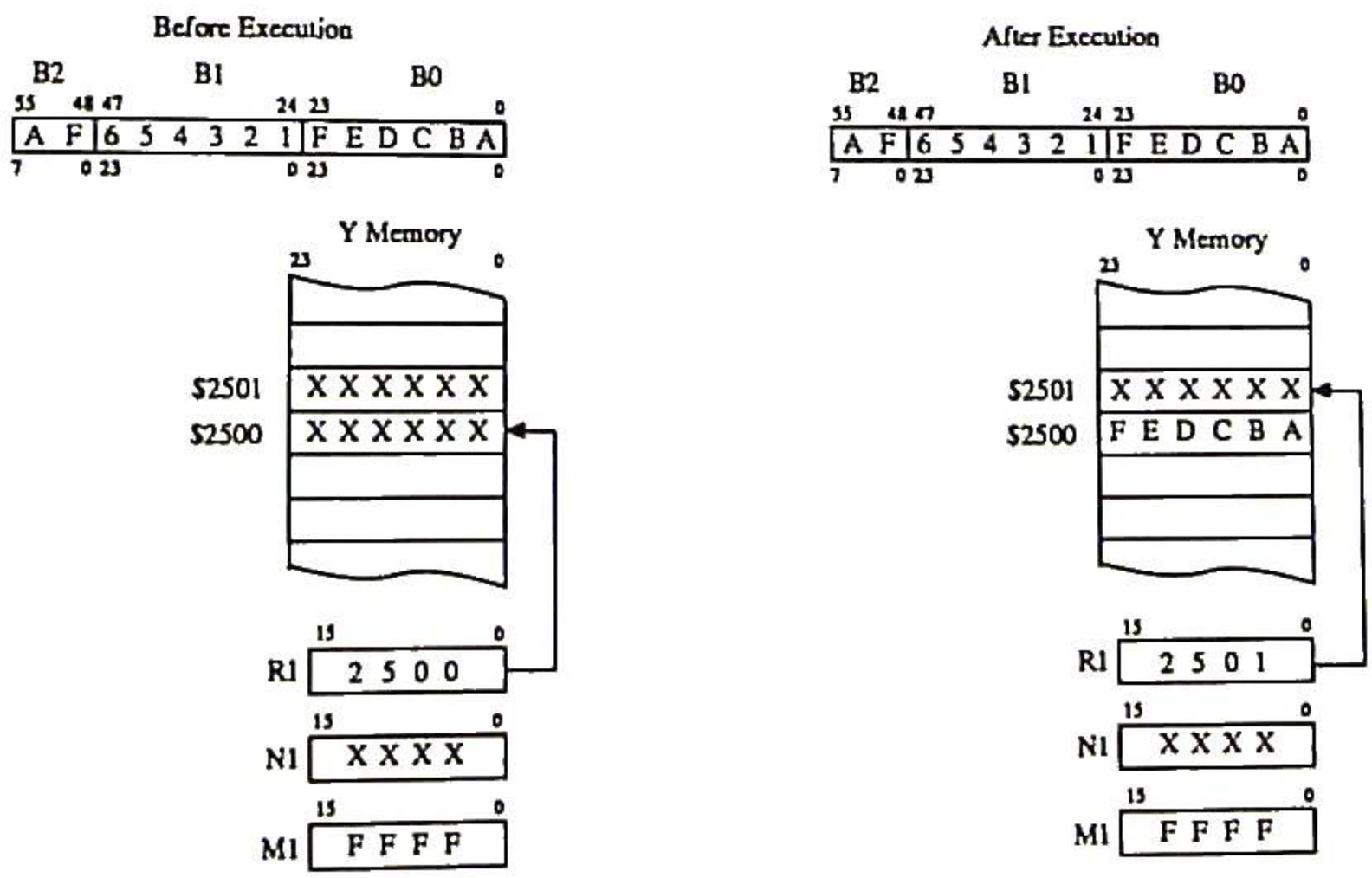


NCD02-0422

POST INCREMENT BY ONE

- Assembler Syntax: (Rn) +
- Operands referenced: P, X, Y, XY, L memories.

Example: MOVE B0,Y:(R1) +

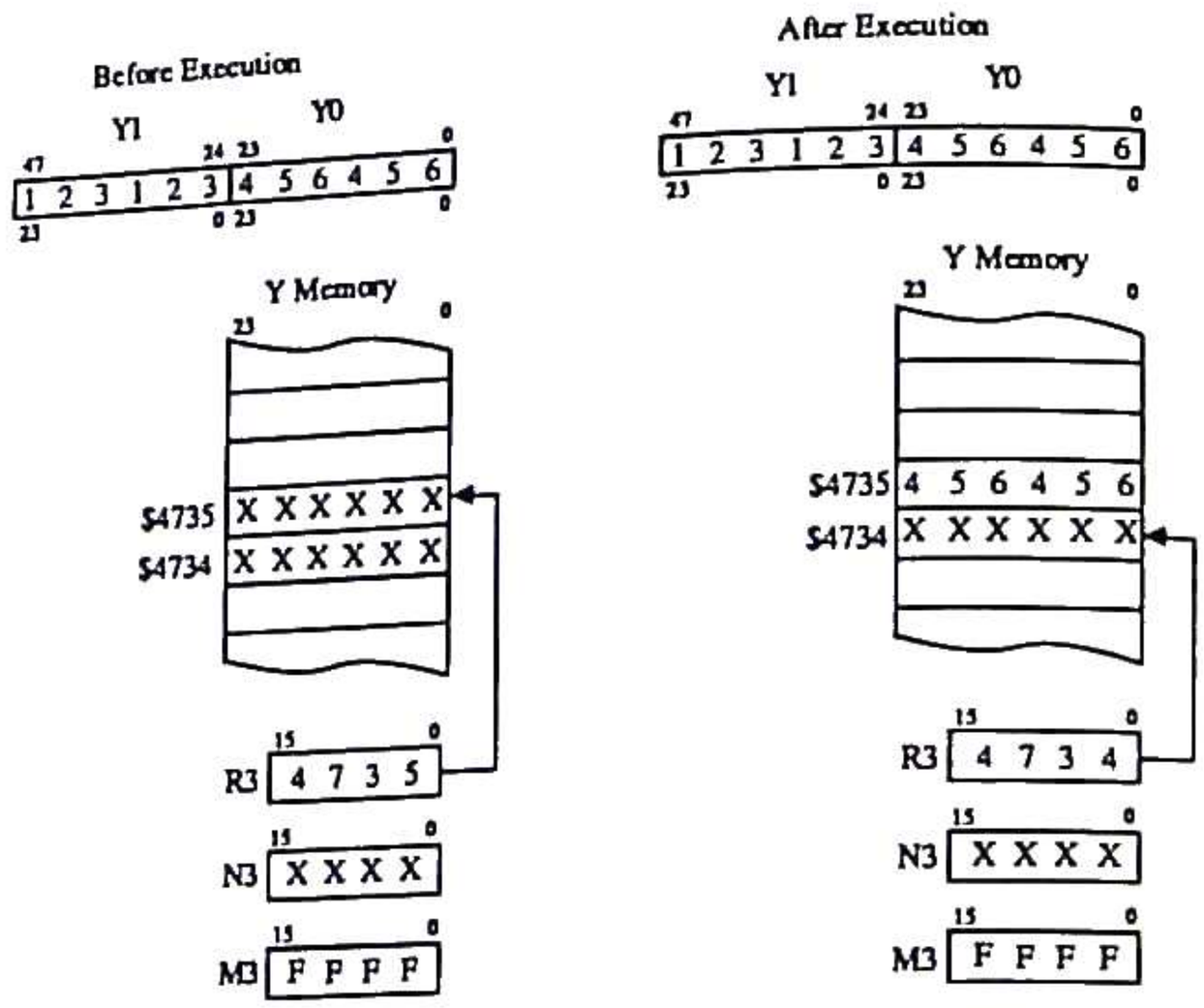


NCD02-0423

POSTDECREMENT BY ONE

- Assembler Syntax: (Rn) -
- Operands referenced: P, X, Y, XY, L memories.

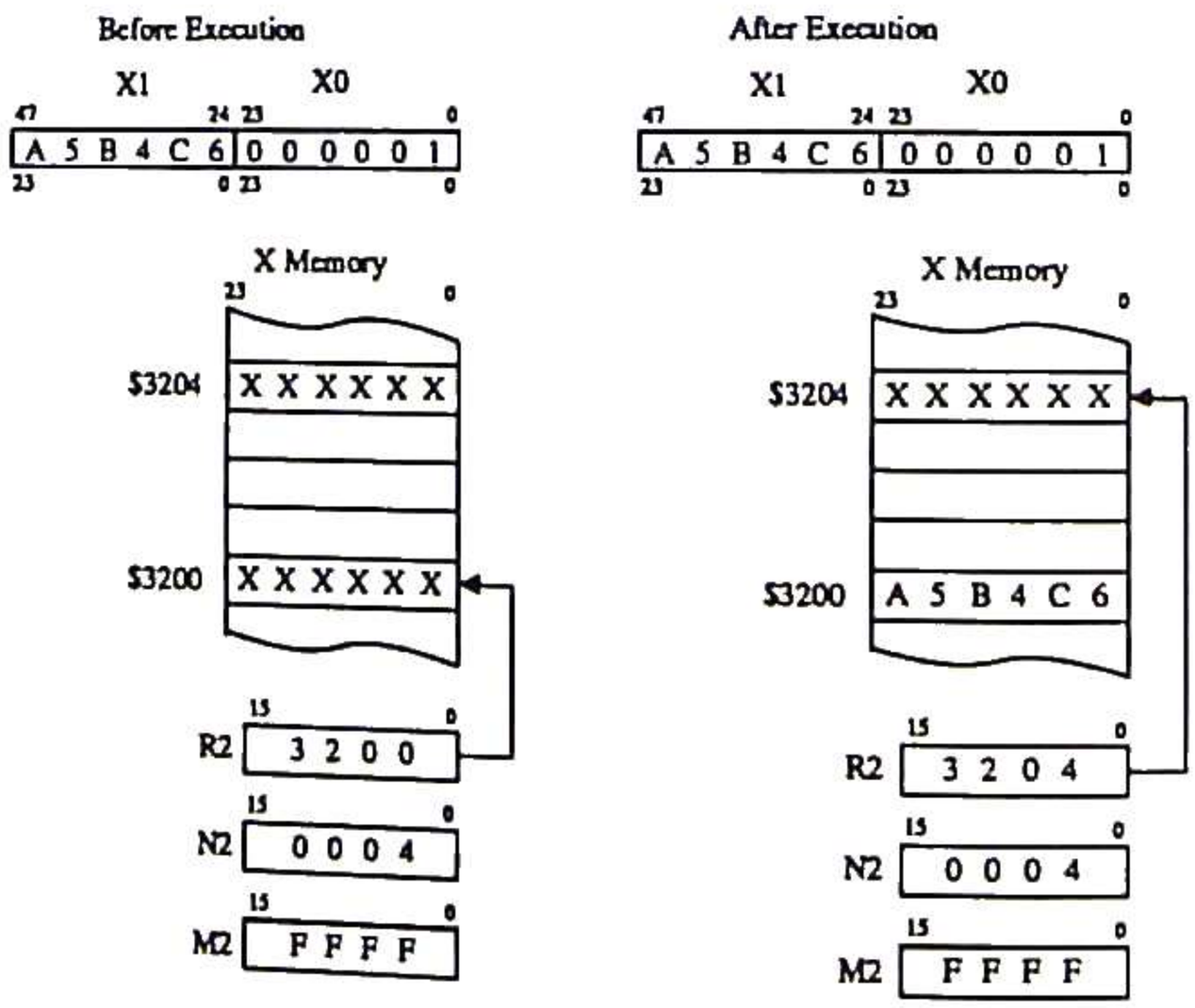
Example: MOVE Y0,Y:(R3) -



POSTINCREMENT BY OFFSET Nn

- Assembler Syntax: (Rn) + Nn
- Operands referenced: P, X, Y, L, XY memories.

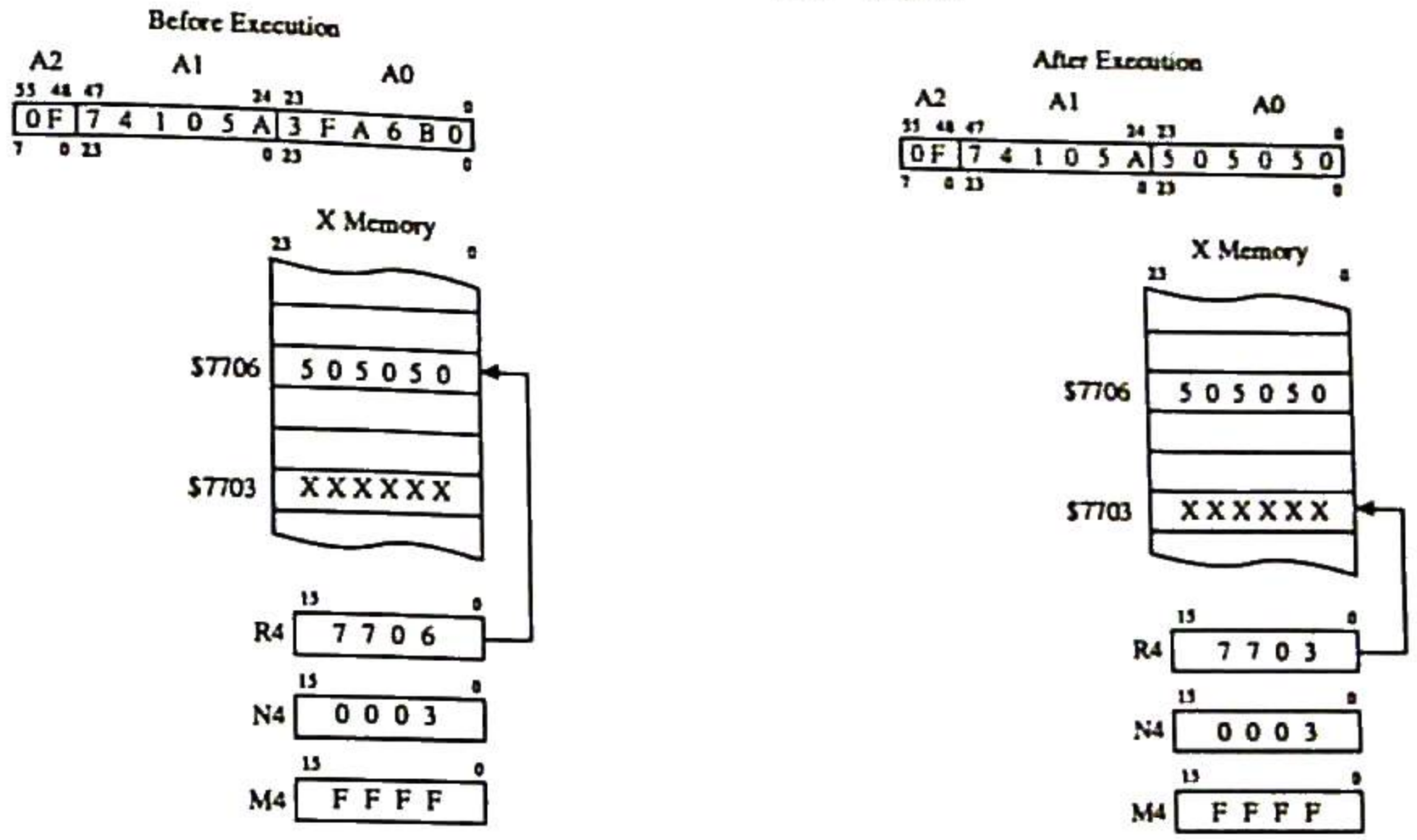
Example: MOVE X1, X:(R2) + N2



POSTDECREMENT BY OFFSET Nn

- Assembler Syntax: (Rn) - Nn
- Operands referenced: P, X, Y, L memories.

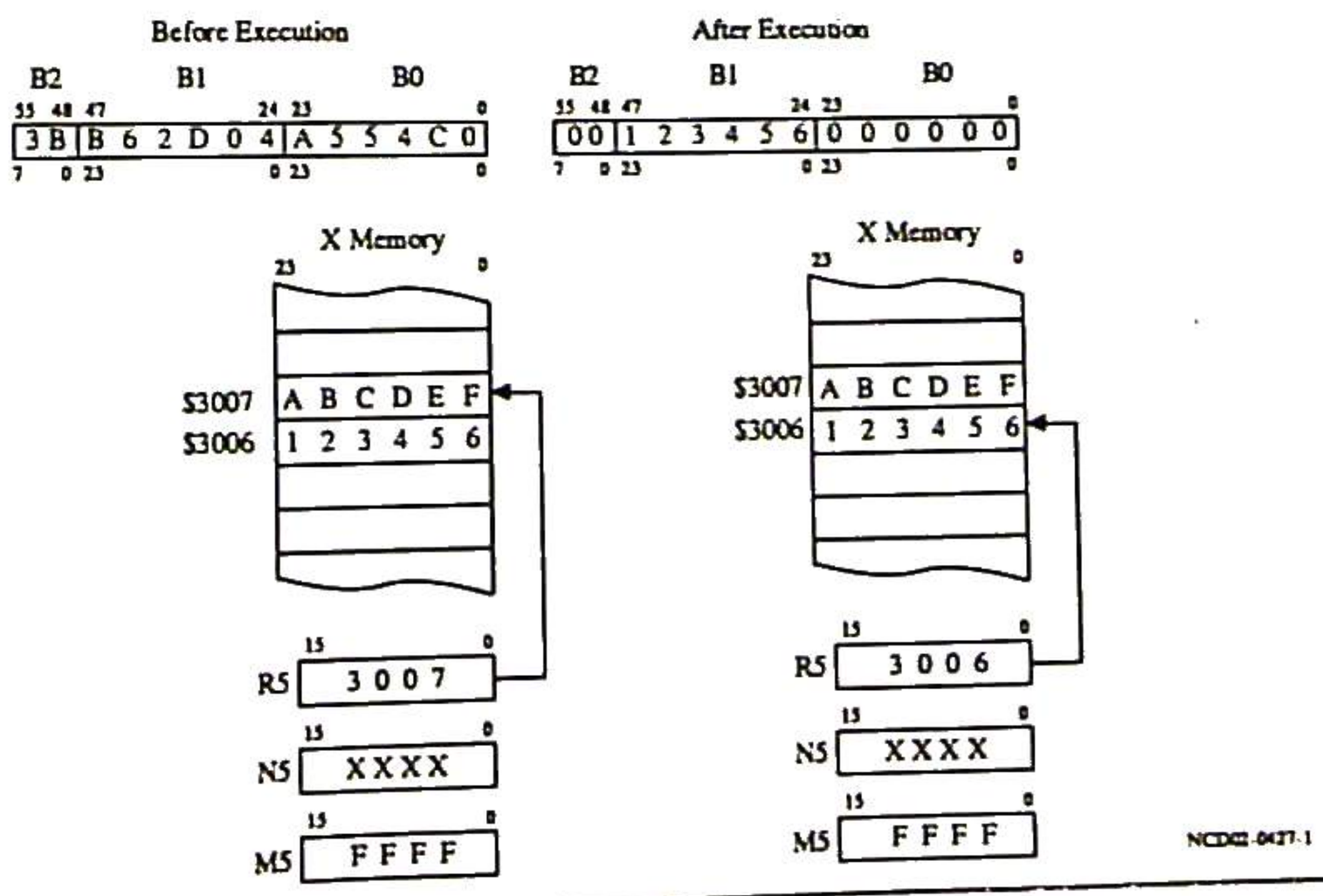
Example: MOVE X:(R4) - N4, A0



PREDECREMENT BY ONE

- Assembler Syntax: -(Rn)
- Operands referenced: P, X, Y, L memories.

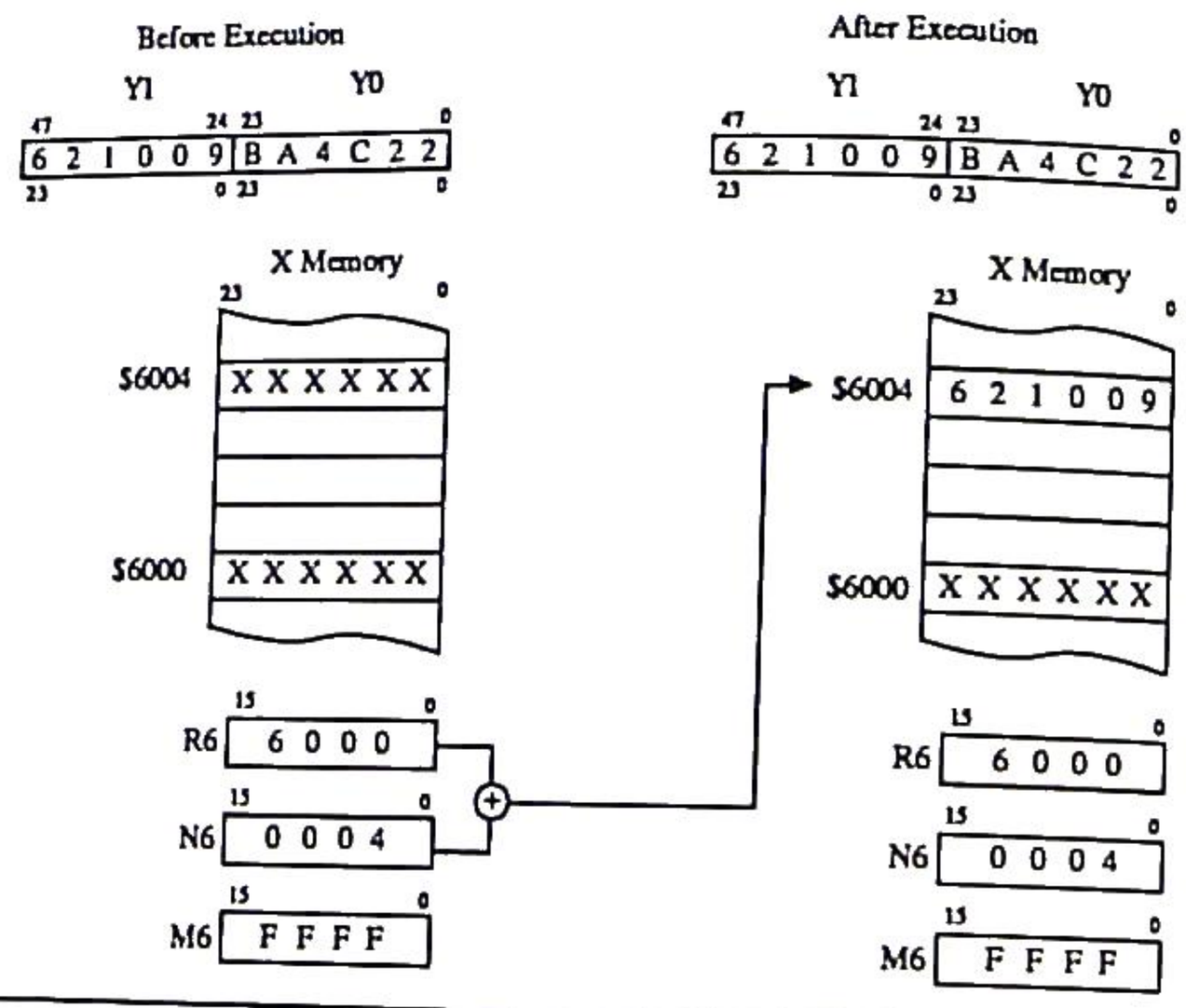
Example: MOVE X:- (R5), B1



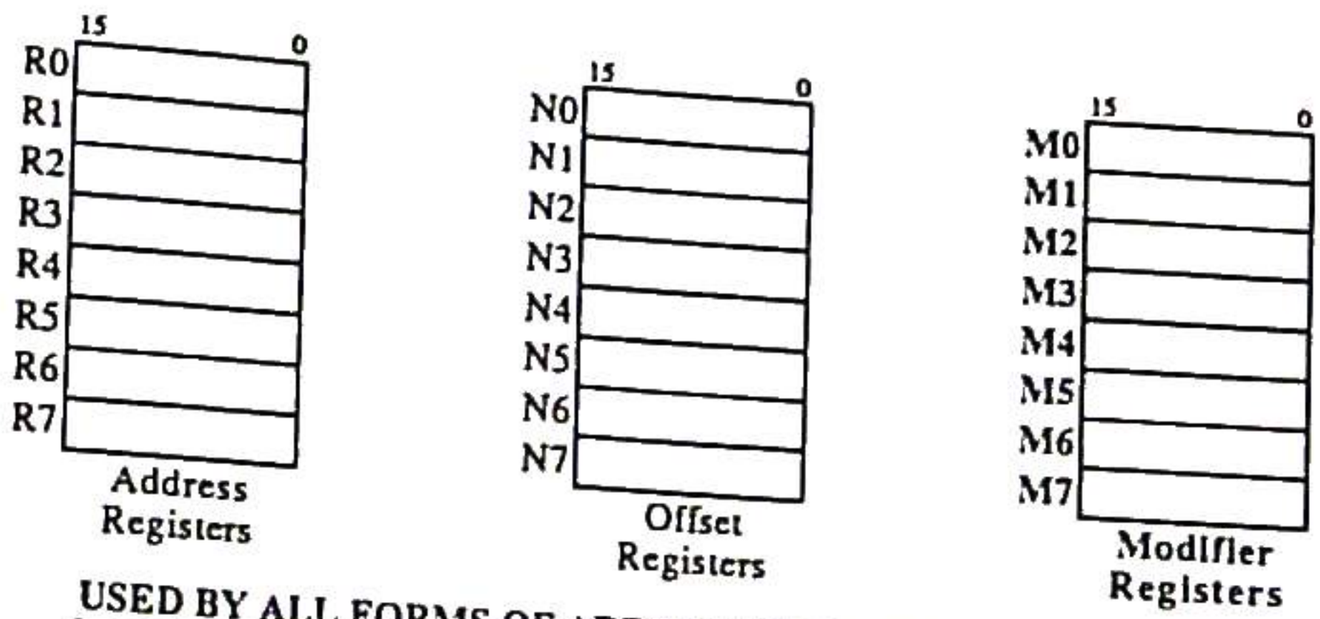
INDEXED BY OFFSET Nn

- Assembler Syntax: (Rn + Nn)
- Operands referenced: P, X, Y, L memories.

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



MODIFIER REGISTERS



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)

- Modifier register Mn alters the effective address that is derived from Rn and Nn.

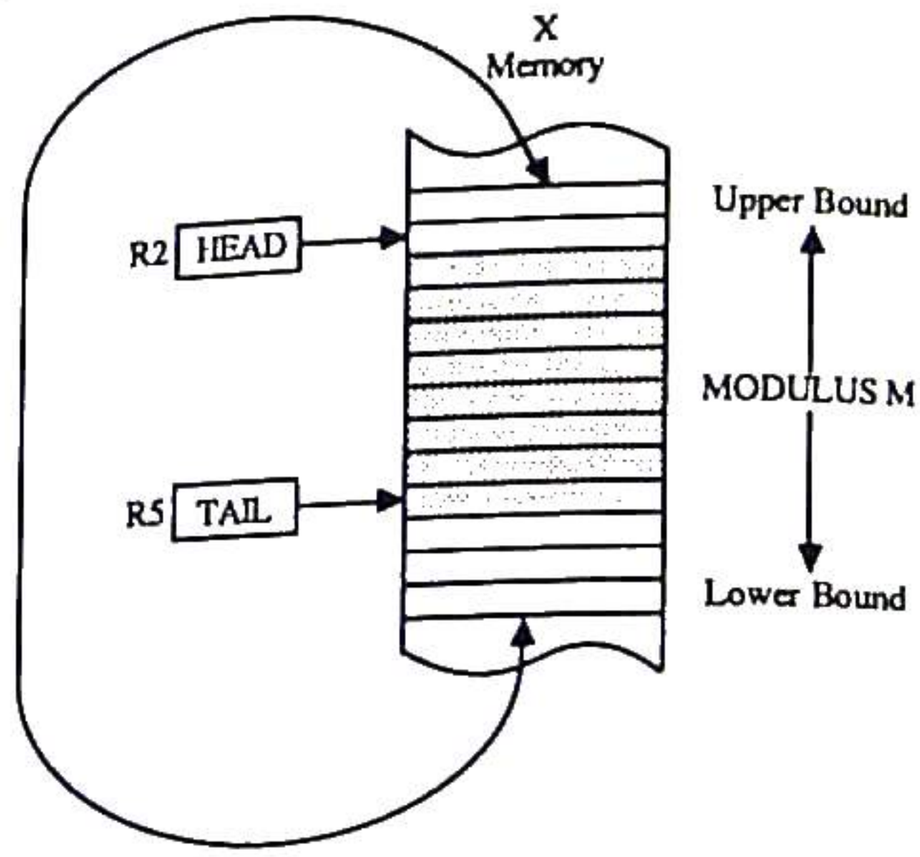
NCD02-0430-1

MODIFIER REGISTER VALUES

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

NCD02-0431

MODULO M ADDRESSING APPLICATIONS



Applications

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

CIRCULAR QUEUE

HEAD

R2 HEAD N2 XXXX M2 M-1

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

TAIL

R5 TAIL N5 XXXX M5 M-1

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

• CAUTION– This does not look out for Head and Tail Crashes.
(i.e. Queue Overflow and Underflow)

MODULO M REGISTER SETUP

1. Modifier Register Mn = M-1. — MODULUS =M

2. Lower Bound = $\overset{15}{X}X\ldots XX00\ldots00^{0}$ where $2^{J} \geq M$ — Beginning of Table (Minimize J)

3. Upper Bound = $\overset{15}{X}X\ldots XX00\ldots00^{0} + M - 1$ — End of Table

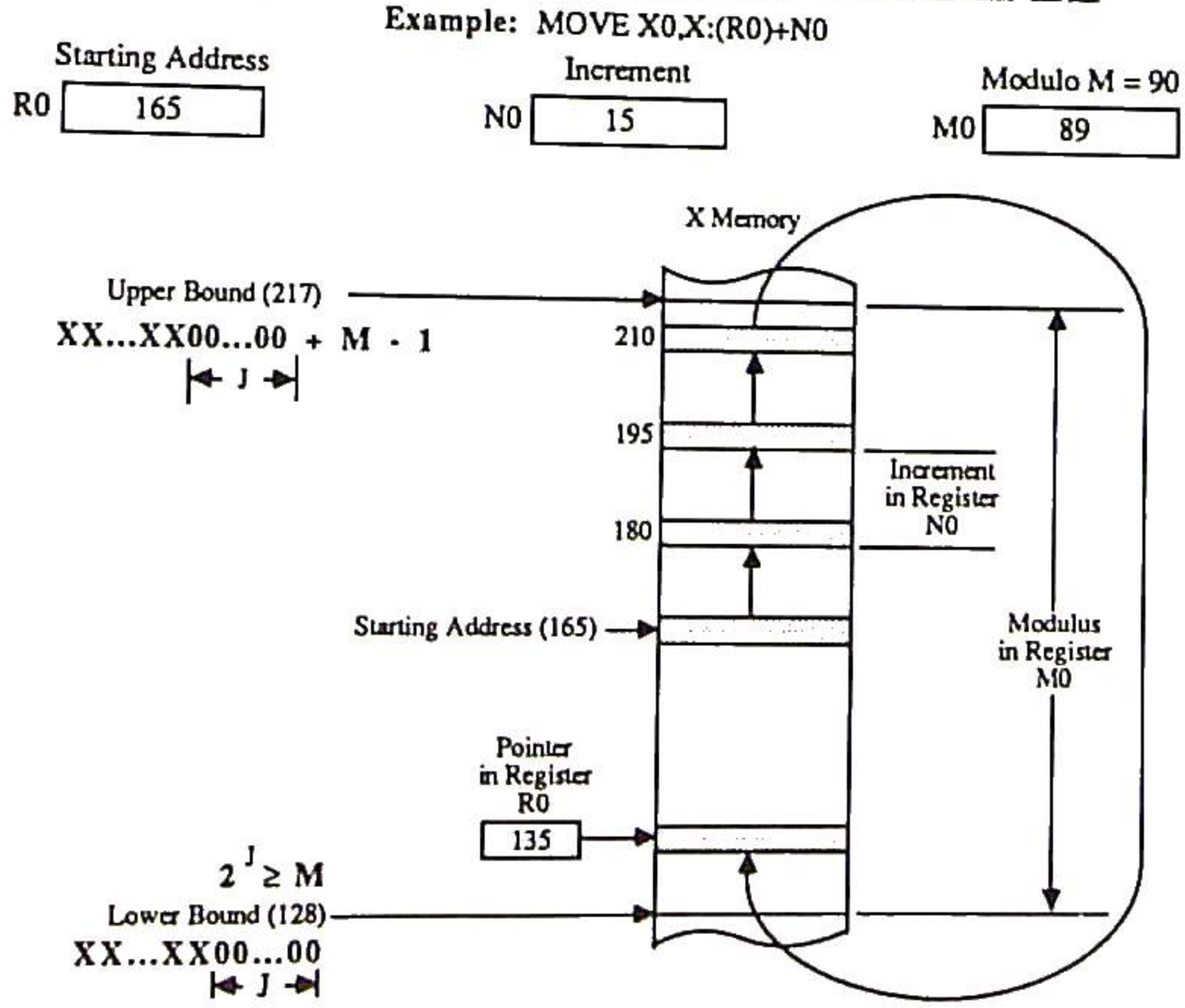
4. Lower Bound ≤ Address Register Rn ≤ Upper Bound — Starting Point within Table

5. Offset Register Nn = Increment ≤ M — Desired Increment (if any)

NCD02-0434

MODULO M ADDRESSING EXAMPLE

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

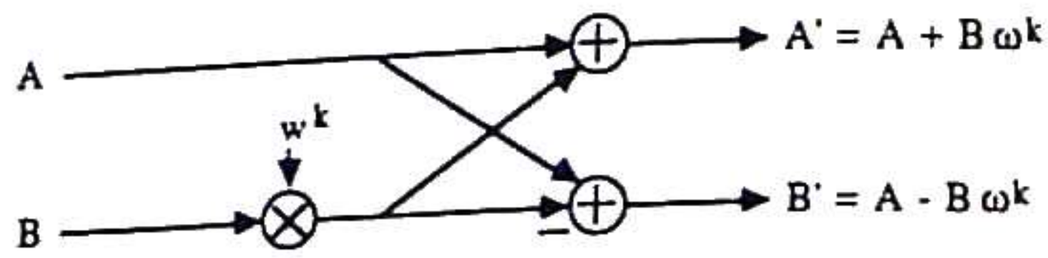


NOTE: This example is in decimal.

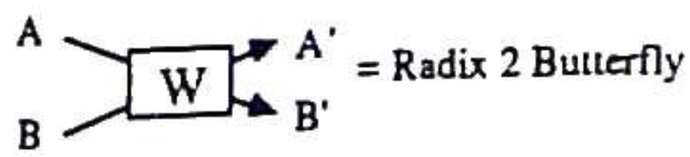
NCD02-0435

FAST FOURIER TRANSFORM BUTTERFLY

Radix 2, decimation in time, complex FFT butterfly

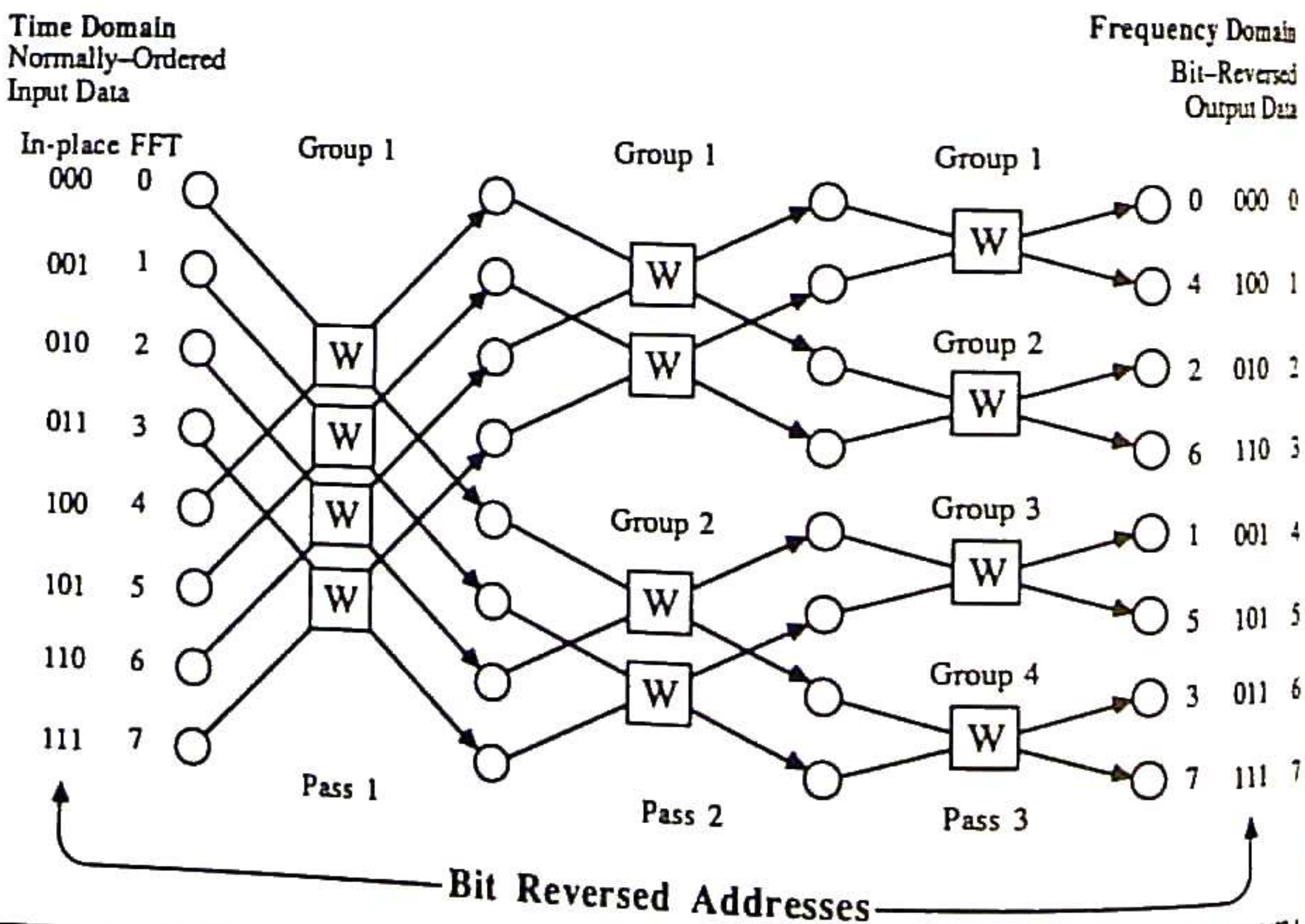


Schematic Symbol



NCD01-0436-1

8 POINT FFT TRANSFORM



NCD02-0437-1

BIT REVERSED REGISTER SETUP FOR N POINT FFT

1. Modifier Register Mn = $0000

   N = Table Size
   Select Bit Reverse Addressing Mode

2. Lower Bound = $\overset{15}{XX} \ldots \underbrace{XX00 \ldots 00}_{K}^{0}$ where $2^K \geq N$

   Beginning of Table
   (Minimize K)

3. Upper Bound = $\overset{15}{XX} \ldots \underbrace{XX00 \ldots 00}_{K}^{0} + 2^K - 1$

   End of Table

4. Lower Bound ≤ Address Register Rn ≤ Upper Bound

   Starting Point within Table

5. Offset Register Nn = 2^{K-1}

   Initialize Offset Register

NCD02-0438-1

BIT REVERSED/REVERSE CARRY ALGORITHM

N = 1024

K = 10

Lower Bound = 3 x N = 3 x 1024 = 3072

Upper Bound = 3 x N + 2^K – 1 = 3072 + 1024 – 1 = 4095

Bit Reversed

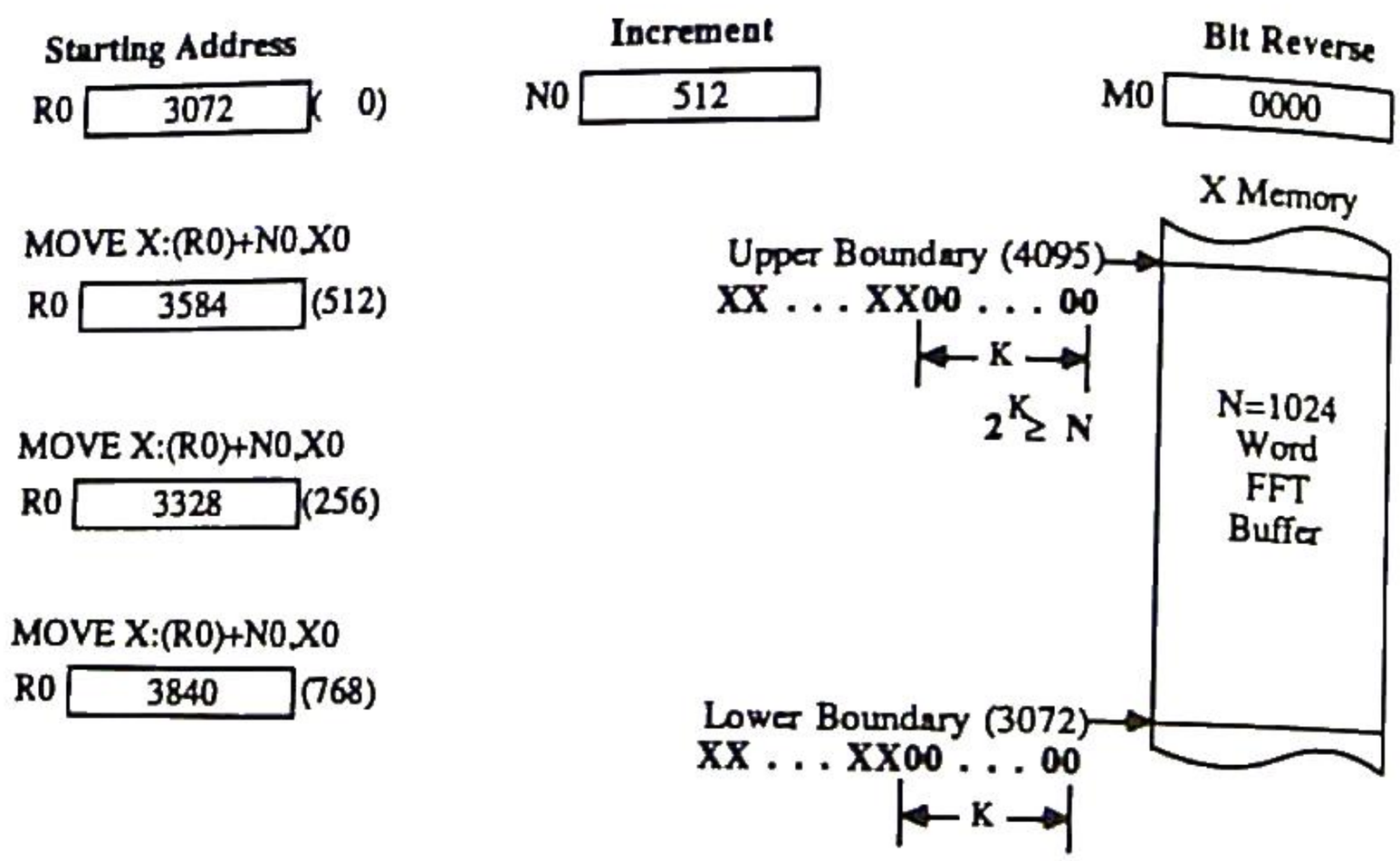
```
                    |<------ K ------>|
4041 = 0 0 0 0 1 1 | 1 1 1 1 0 0 1 0 0 1
Reverse K LSB's
3663 = 0 0 0 0 1 1 | 1 0 0 1 0 0 1 1 1 1
 +1     No Carry     Increment K LSB's
        Beyond K LSB'S
3664 = 0 0 0 0 1 1 | 1 0 0 1 0 1 0 0 0 0
Reverse K LSB's
3113 = 0 0 0 0 1 1 | 0 0 0 0 1 0 1 0 0 1
```

Reversed Carry

```
                    |<------ K ------>|
4041 = 0 0 0 0 1 1 | 1 1 1 1 0 0 1 0 0 1
Increment Kth Bit  | +1
Propagate Carry    |-------------------->
from Left to Right | 1 1 1 1 0 0 0 0 0 0
                   | ^ ^ ^ ^ ^ ^ ^ ^ ^ ^
3113 = 0 0 0 0 1 1 | 0 0 0 0 1 0 1 0 0 1
```

NCD02-0439-1

BIT REVERSED ADDRESSING EXAMPLE



NOTE: This example is in decimal.

NCD02-040-1

DSP56000/1
INSTRUCTION SET

Objectives:

- Describe the operation of each instruction.
- Write simple programs.

NCD02-0500-1

Table of Contents

DSP56000 Instruction Set
Parallel Data Move Instructions 5-5
Parallel Data Move Operations 5-5
Parallel Move Types 5-6
Parallel Move Rules 5-6
16 Bit Transfers 5-7
Immediate Short Data Move 5-8
Immediate Short Examples 5-9
Register to Register Data Move 5-9
Register To Register Examples 5-10
Address Register Update 5-10
Address Register Update Examples 5-11
X Memory Data Move 5-11
X Memory Data Move Examples 5-12
X Memory and Register Data Move 5-13
X Memory and Register Data Move Examples 5-14
Y Memory Data Move 5-14
Y Memory Data Move Examples 5-15
Register and Y Memory Data Move 5-16
Register and Y Memory Date Move Examples 5-17
Long Memory Data Move 5-18
Long Memory Data Move Examples 5-19
XY Memory Data Move 5-20
XY Memory Data Move Examples 5-20
Parallel Move Summary 5-21
MOVE Instruction 5-22
MOVE Control Register 5-22
MOVE Program Memory 5-23
MOVE Peripheral Data 5-23
MOVEC, MOVEM and MOVEP used with SSH 5-24
Load Updated Address 5-24
Load Updated Address Examples 5-25
Addition and Subtraction Arithmetic Instructions 5-25
Addition and Subtraction Arithmetic Instruction Examples 5-26
Shift and Add or Subtract Arithmetic Instructions 5-26
Shift and Add or Subtract Arithmetic Instruction Examples 5-27
Multiply Arithmetic Instructions 5-27
Multiply Arithmetic Instruction Examples 5-28
Multiply and Accumulate Instructions 5-28
Multiply and Accumulate Instruction Examples 5-29

Table of Contents (cont'd)

Arithmetic Shift 5-30
Compare Arithmetic Instructions 5-30
Affect of Arithmetic Instructions on CCR 5-31
Transfer Instructions 5-32
Transfer Example 5-33
Conditional Transfer Example 5-34
Arithmetic Instructions 5-34
Arithmetic Instruction Examples 5-35
Logical Instructions 5-35
Logical Instruction Examples 5-36
Immediate Logical Instructions 5-36
Immediate Logical Instruction Examples 5-37
Shift and Rotate Instructions 5-37
Shift and Rotate Instruction Examples 5-38
Repeat Next Instruction 5-38
Repeat Next Instruction Examples 5-39
Characteristics of the REP Instruction 5-40
Hardware DO Loops 5-41
DO Loop Example 5-41
END Current DO Loop 5-42
DO Don't Do's 5-43
Branching Instructions 5-44
Branching Addressing Modes and Conditions 5-45
Bit Manipulation Instructions 5-46
Bit Manipulation Examples 5-47
Bit Field Branching Instructions 5-48
Bit Field Branching Instruction Examples 5-49
Divide Iteration 5-50
One Quadrant Division Example 5-51
One Quadrant Division Accumulator Trace 5-52
Four Quadrant Division Example 5-53
Characteristics of the Divide Iteration 5-53
Miscellaneous Instructions 5-54
Binary to Gray Code Converter 5-54
Random Number Generator 5-55
Square Root 5-55

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.

NCD02-0501

PARALLEL DATA MOVE INSTRUCTIONS

- **Syntax:**

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

Specifies:

- Data ALU Operation
- Logical Operation
- Convergent Rounding

Supports:

- Condition Code Bits 0 through 5

Specifies:

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

Supports:

- Scaling
- Limiting
- Sign extension and least significant zero fill
- Duplicate sources
- Duplicate destinations <u>not allowed</u>

NCD02-0502

PARALLEL DATA MOVE OPERATIONS

Parallel Data Move

MACR -X0, Y0, A

1. Negate X0
2. 24 x 24 Bit Multiply
   X0 x Y0
3. 56 Bit Sum into Accumulator A
4. Convergently Round A
5. Update Condition Code Register

A,X:(R0)+N0

6. MOVE A to X Memory
7. Scale
8. Limit (If Necessary)
9. Calculate Next Address (R0)+ N0
10. Use Modifier Register M0 For
    - Linear Addressing
    - Modulo M
    - Bit Reverse

Y:(R4)-N4,B

11. MOVE Y Memory to B
12. Sign extend and Zero fill
13. Calculate Next Address (R4) - N4
14. Use Modifier Register M4 For
    - Linear Addressing
    - Modulo M
    - Bit Reverse

• This All Occurs In 97.5 nS.

NCD02-0503

PARALLEL MOVE TYPES

- No Parallel Move
- Immediate Short Data Move
- Register to Register Data Move
- Address Register Update
- X Memory Data Move
- X Memory and Register Data Move
- Y Memory Data Move
- Register and Y Memory Data Move
- Long Memory Data Move
- XY Memory Data Move

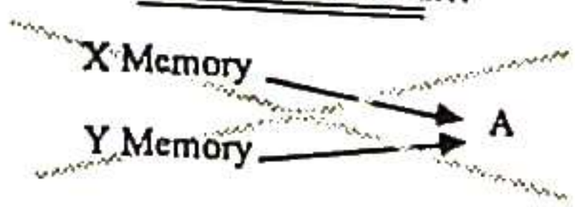
NCD02-0504

PARALLEL MOVE RULES

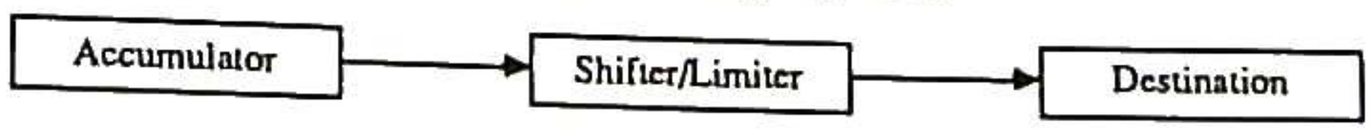
- Duplicate Sources Are Permitted

A ⟶ X Memory and A ⟶ Y Memory

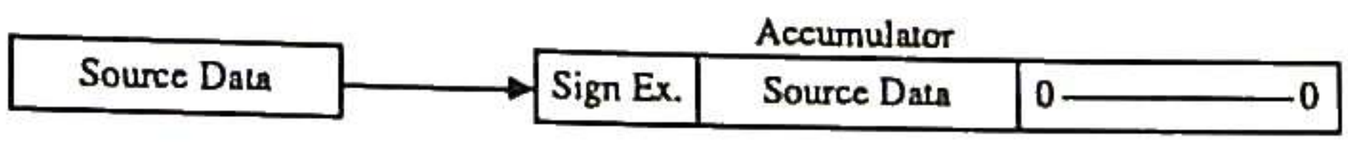
- Duplicate Destinations Are Not Permitted!!!



- If An Accumulator is a Source, Then Shifting or Limiting may occur.



- If An Accumulator is a Destination Then Sign Extension and Least Significant Zero Fill Occurs.



- Due to pipelining, If an Address Register is changed using a Move Type Instruction, the new contents of the destination Address Register will not be available for use during the folowing instruction.

  – There is a single instruction pipeline delay.

  – This is flagged as a warning by the assembler.

NCD02-0524

PARALLEL DATA MOVE OPERATIONS

Parallel Data Move

MACR -X0, Y0, A

1. Negate X0
2. 24 x 24 Bit Multiply
   X0 x Y0
3. 56 Bit Sum into Accumulator A
4. Convergently Round A
5. Update Condition Code Register

A,X:(R0)+N0

6. MOVE A to X Memory
7. Scale
8. Limit (If Necessary)
9. Calculate Next Address (R0)+ N0
10. Use Modifier Register M0 For
    - Linear Addressing
    - Modulo M
    - Bit Reverse

Y:(R4)-N4,B

11. MOVE Y Memory to B
12. Sign extend and Zero fill
13. Calculate Next Address (R4) - N4
14. Use Modifier Register M4 For
    - Linear Addressing
    - Modulo M
    - Bit Reverse

• This All Occurs In 97.5 nS.

NCD02-0503

PARALLEL MOVE TYPES

- No Parallel Move
- Immediate Short Data Move
- Register to Register Data Move
- Address Register Update
- X Memory Data Move
- X Memory and Register Data Move
- Y Memory Data Move
- Register and Y Memory Data Move
- Long Memory Data Move
- XY Memory Data Move

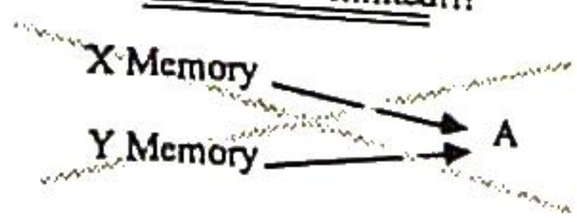
NCD02-0504

PARALLEL MOVE RULES

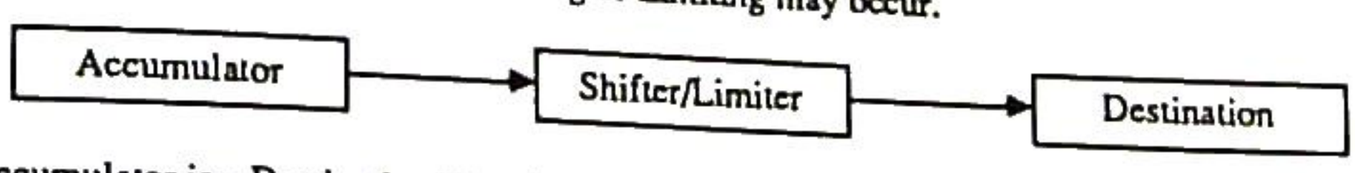
- Duplicate Sources Are Permitted

A ⟶ X Memory and A ⟶ Y Memory

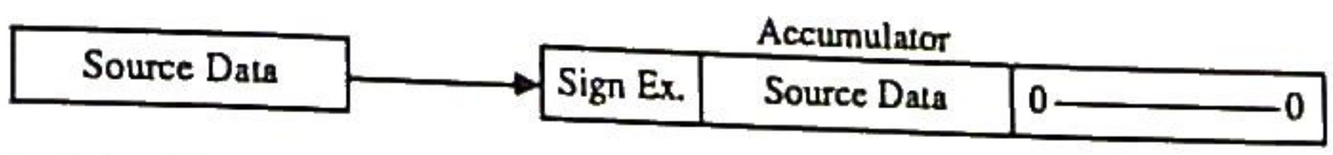
- Duplicate Destinations Are Not Permitted!!!



- If An Accumulator is a Source, Then Shifting or Limiting may occur.



- If An Accumulator is a Destination Then Sign Extension and Least Significant Zero Fill Occurs.



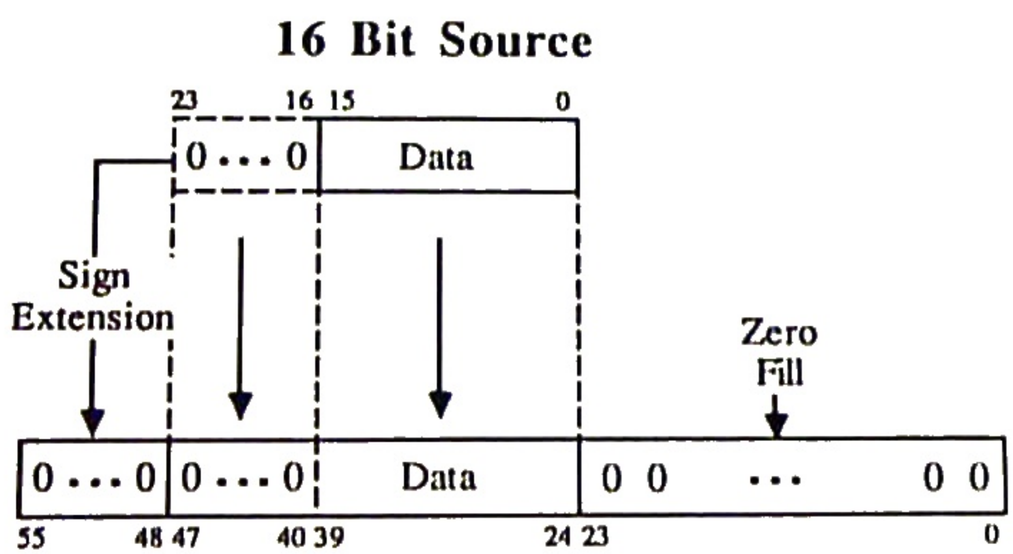
- Due to pipelining, If an Address Register is changed using a Move Type Instruction, the new contents of the destination Address Register will not be available for use during the folowing instruction.

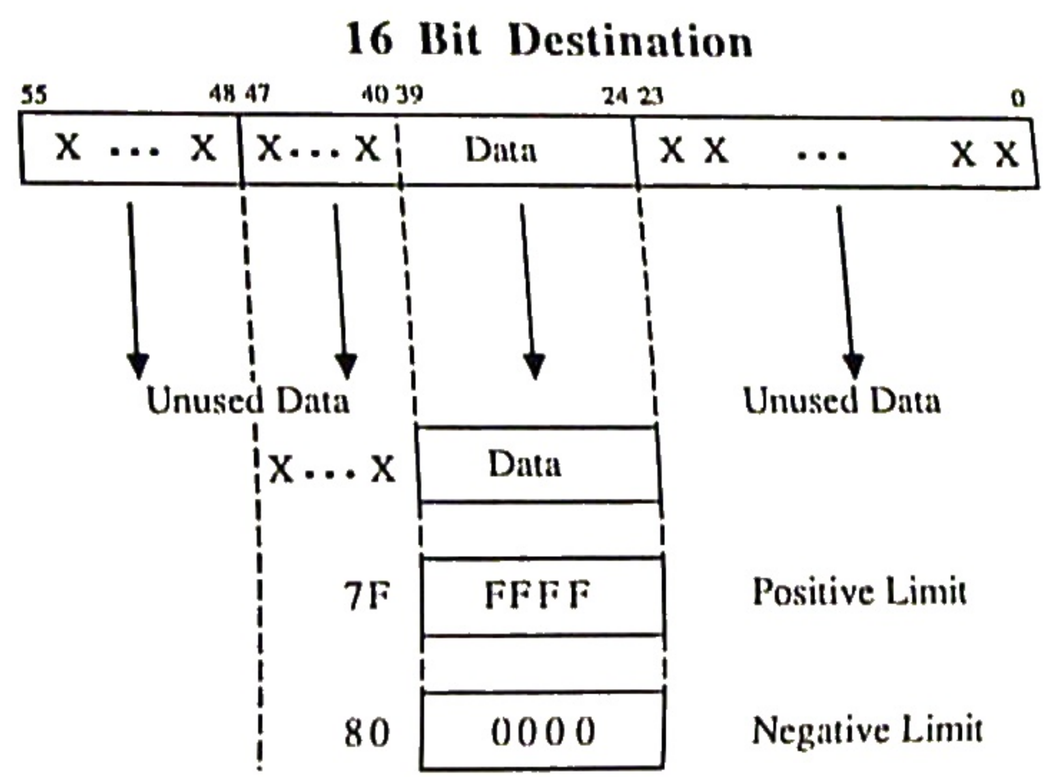
  – There is a single instruction pipeline delay.

  – This is flagged as a warning by the assembler.

NCD02-0524

16 BIT TRANSFERS





IMMEDIATE SHORT DATA MOVE

Assembler Syntax:

Opcode	Operand	PARALLEL MOVE
		#xx, Destination

Addressing Modes

Source Operand		Destination Registers	
Immediate Short (8 Bits)	#xx	A2, A1, A0 B2, B1, B0 R0 – R7 N0 – N7	Unsigned Integer Transfer to Least Significant Portion
		X1, X0 Y1, Y0 A,B	Signed Fraction Transfer

NCD02-0505

IMMEDIATE SHORT EXAMPLES

Unsigned Integer

Example 1:

Opcode	Operand	PARALLEL MOVE
ADD	A,B	#$25,A0

Before Execution

A	01	000001	000001
B	06	FFFFFE	FFFFFE

After Execution

A	01	000001	000025
B	07	FFFFFF	FFFFFF

Signed Fraction

Example 2:

Opcode	Operand	PARALLEL MOVE
ADD	B,A	#$C7,B

Before Execution

A	01	000001	000001
B	01	000001	000001

After Execution

A	02	000002	000002
B	FF	C70000	000000

NCD02-0506

REGISTER TO REGISTER DATA MOVE

Assembler Syntax:

Opcode	Operand	PARALLEL MOVE
		Source, Destination

Addressing Modes

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

NCD02-0507

REGISTER TO REGISTER EXAMPLES

Example 1:

Opcode	Operand	PARALLEL MOVE
ADD	B,A	A,N7

Before Execution

A	00	123456	ABCDEF
B	12	654321	9ABCDE
N7		0000	

After Execution

A	12	777778	468ACD
B	12	654321	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	

Limiting has occurred

NCD02-0508

ADDRESS REGISTER UPDATE

Assembler Syntax: Opcode	Operand	PARALLEL MOVE <Effective Address>

Addressing Modes

Effective Address:		Address Register
Postincrement by 1	(Rn)+	R0-R7
Postdecrement by 1	(Rn)-	
Postincrement by Offset Nn	(Rn)+Nn	
Postdecrement by Offset Nn	(Rn)-Nn	

NCD02-0509

ADDRESS REGISTER UPDATE EXAMPLES

Example 1: Opcode: ADD; Operand: A,B; PARALLEL MOVE: (R1)+N1

Before Execution

A	11	222222	333333
B	44	555555	666666
R1	2000		
N1	F000		

After Execution

A	11	222222	333333
B	55	777777	999999
R1	1000		
N1	F000		

Example 2: Opcode: ADD; Operand: A,B; PARALLEL MOVE: (R0)-

Before Execution

A	12	345678	9ABCDE
B	12	345678	9ABCDE
R0	2000		

After Execution

A	12	345678	9ABCDE
B	24	68ACF1	3579BC
R0	1FFF		

NCD02-0510

X MEMORY DATA MOVE

Assembler Syntax:		PARALLEL MOVE
Opcode	Operand	X:<Effective Address>,Destination
Opcode	Operand	Source,X:<Effective Address>
Opcode	Operand	#xxxxxx, Destination

Addressing Modes

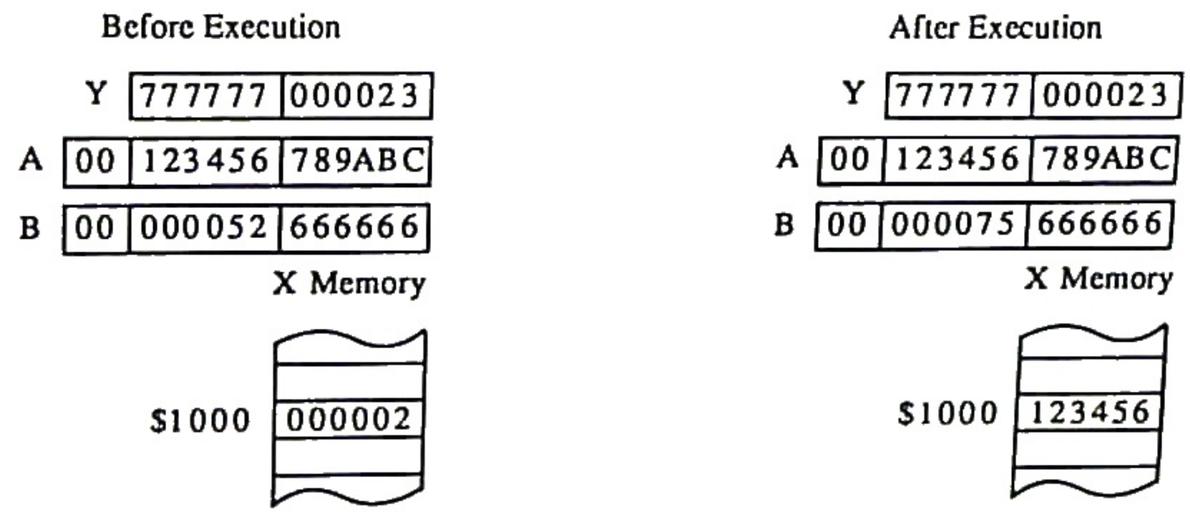
Effective Address:		Source or Destination
Absolute Short Address	aa	X1,X0
Absolute Address	xxxx	Y1,Y0
Address Register Indirect with;		A,A2,A1,A0
No Update	(Rn)	B,B2,B1,B0
Postincrement by 1	(Rn)+	R0-R7
Postdecrement by 1	(Rn)-	N0-N7
Postincrement by Offset Nn	(Rn)+Nn	
Postdecrement by Offset Nn	(Rn)-Nn	
Indexed by Offset Nn	(Rn+Nn)	
Predecrement by 1	-(Rn)	

NCD02-0511

X MEMORY DATA MOVE EXAMPLES

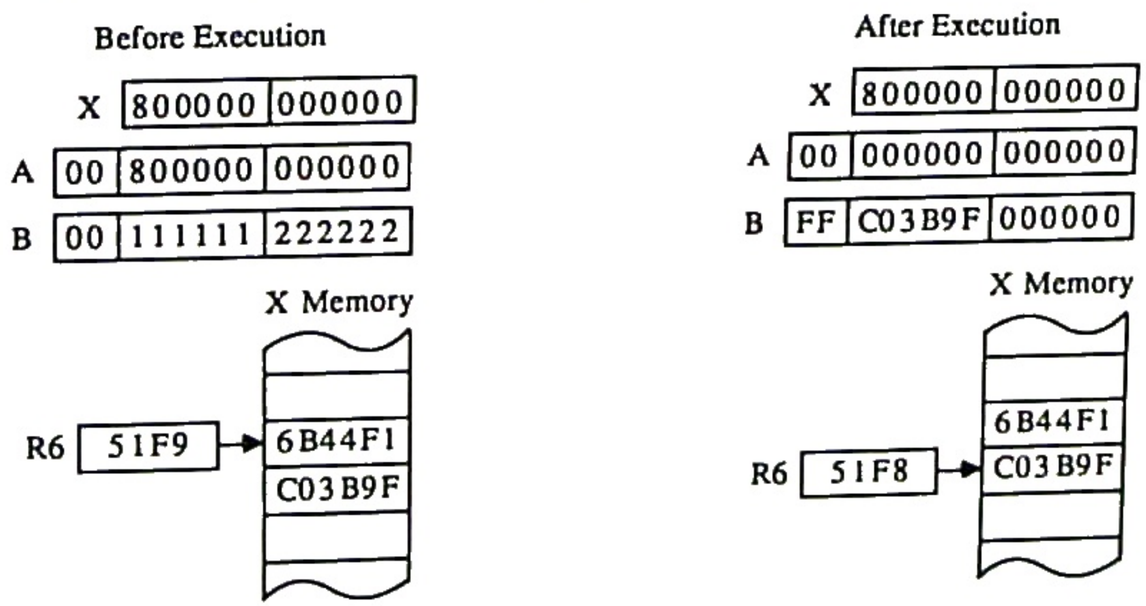
Example 1:

Opcode	Operand	PARALLEL MOVE
ADD	Y0,B	A,X:$1000



Example 2:

Opcode	Operand	PARALLEL MOVE
ADD	X,A	X:-(R6), B



NCD02-0512

X MEMORY AND REGISTER DATA MOVE

PARALLEL MOVE

Assembler Syntax:			
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

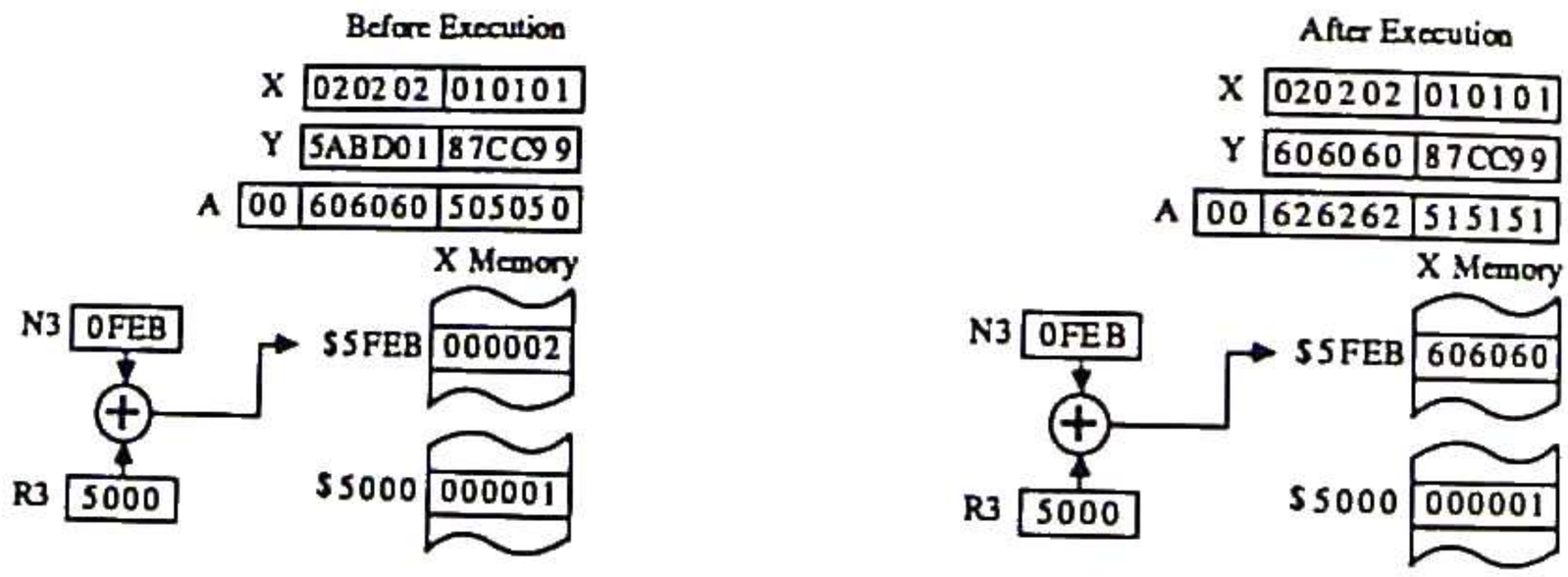
Addressing Modes

Effective Address:		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)			

NCD02-0513

X MEMORY AND REGISTER DATA MOVE EXAMPLES

Example 1: Opcode ADD, Operand X,A — PARALLEL MOVE A,X:(R3+N3) A,Y1



Example 2: Opcode ADD, Operand Y1,B — PARALLEL MOVE X:-(R5),A A,Y0

NCD02-0514

Y MEMORY DATA MOVE

Assembler Syntax:		PARALLEL MOVE
Opcode	Operand	Y:<Effective Address>,Destination
Opcode	Operand	Source,Y:<Effective Address>
Opcode	Operand	#xxxxxx, Destination

Addressing Modes

Effective Address :		Source or Destination
Absolute Short Address	aa	X1,X0
Absolute Address	xxxx	Y1,Y0
Address Register Indirect with;		A,A2,A1,A0
No Update	(Rn)	B,B2,B1,B0
Postincrement by 1	(Rn)+	R0-R7
Postdecrement by 1	(Rn)-	N0-N7
Postincrement by Offset Nn	(Rn)+Nn	
Postdecrement by Offset Nn	(Rn)-Nn	
Indexed by Offset Nn	(Rn+Nn)	
Predecrement by 1	-(Rn)	

NCD02-0515

Y MEMORY DATA MOVE

Assembler Syntax:		PARALLEL MOVE
Opcode	Operand	Y:<Effective Address>,Destination
Opcode	Operand	Source,Y:<Effective Address>
Opcode	Operand	#xxxxxx, Destination

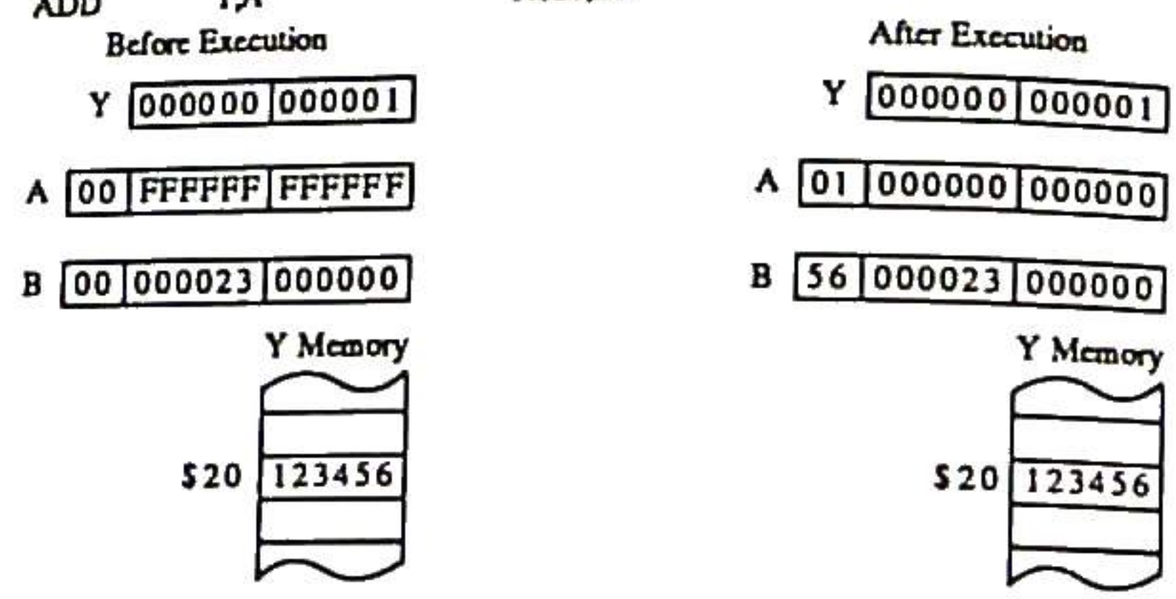
Addressing Modes

Effective Address :		Source or Destination
Absolute Short Address	aa	X1,X0
Absolute Address	xxxx	Y1,Y0
Address Register Indirect with;		A,A2,A1,A0
No Update	(Rn)	B,B2,B1,B0
Postincrement by 1	(Rn)+	R0-R7
Postdecrement by 1	(Rn)-	N0-N7
Postincrement by Offset Nn	(Rn)+Nn	
Postdecrement by Offset Nn	(Rn)-Nn	
Indexed by Offset Nn	(Rn+Nn)	
Predecrement by 1	-(Rn)	

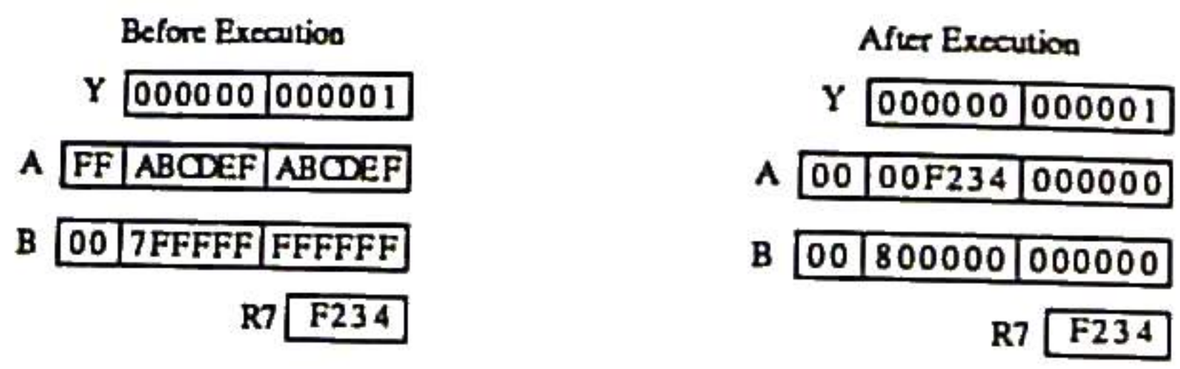
NCD02-0515

Y MEMORY DATA MOVE EXAMPLES

Example 1: Opcode ADD, Operand Y,A — PARALLEL MOVE Y:$20,B2



Example 2: Opcode ADD, Operand Y,B — PARALLEL MOVE R7,A



NCD02-0516

REGISTER AND Y MEMORY DATA MOVE

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:	
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
Y1, Y0
A, B

NCD02-0517

REGISTER AND Y MEMORY DATA MOVE

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

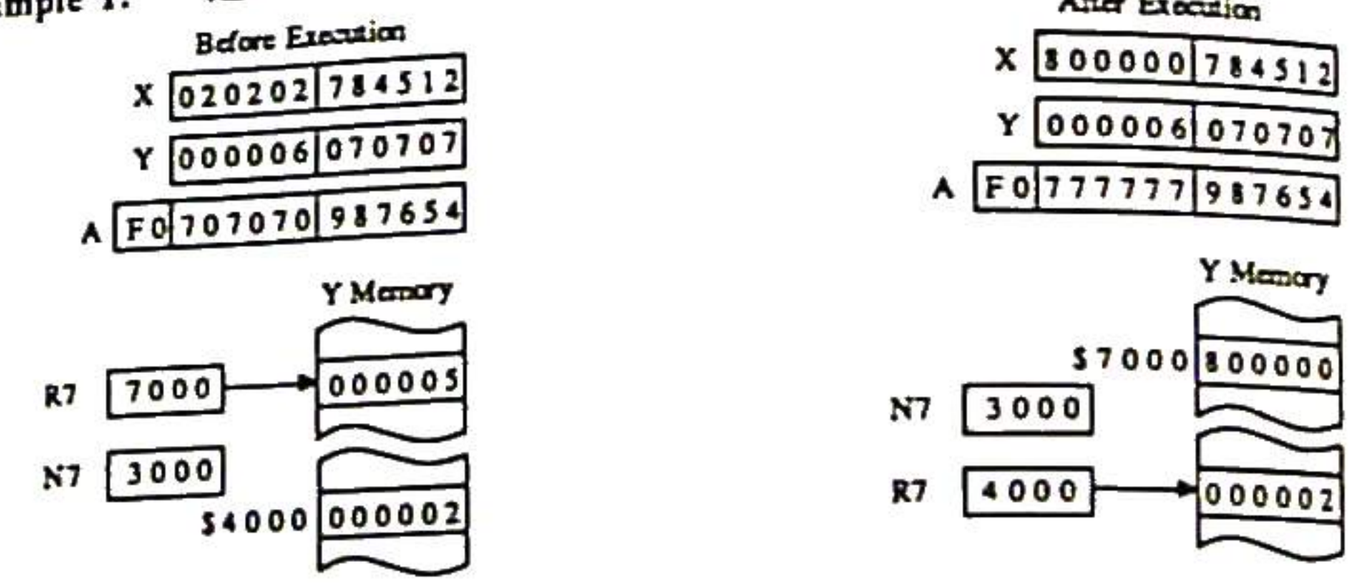
Y Effective Address:	
Absolute Short	aa
Absolute Address	xxxx
Address Register Indirect with;	
No Update	(Rn)
Postincrement by 1:	(Rn)+
Postdecrement by 1:	(Rn)-
Postincrement by Offset Nn	(Rn)+Nn
Postdecrement by Offset Nn	(Rn)-Nn
Indexed by Offset Nn	(Rn+Nn)
Predecrement by 1	-(Rn)

Source 2 or Destination 2
Y1, Y0
A, B

NCD02-0517

REGISTER AND Y MEMORY DATA MOVE EXAMPLES

Example 1: Opcode ADD, Operand Y0,A — PARALLEL MOVE A,X1 A,Y:(R7)-N7



NOTE: Negative Limiting has occurred in the transfer from "A" to X1 and Y:$7000.

Example 2: Opcode ADD, Operand X0,A — PARALLEL MOVE A,X0 A,Y:(R2)+

NOTE: Positive Limiting has occurred in the transfer from "A" to Y:$913D.

LONG MEMORY DATA MOVE

Assembler Syntax:		PARALLEL MOVE
Opcode	Operand	L:<Effective Address>,Destination
Opcode	Operand	Source,L:<Effective Address>

Addressing Modes

L Effective Address:	
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 or Destination

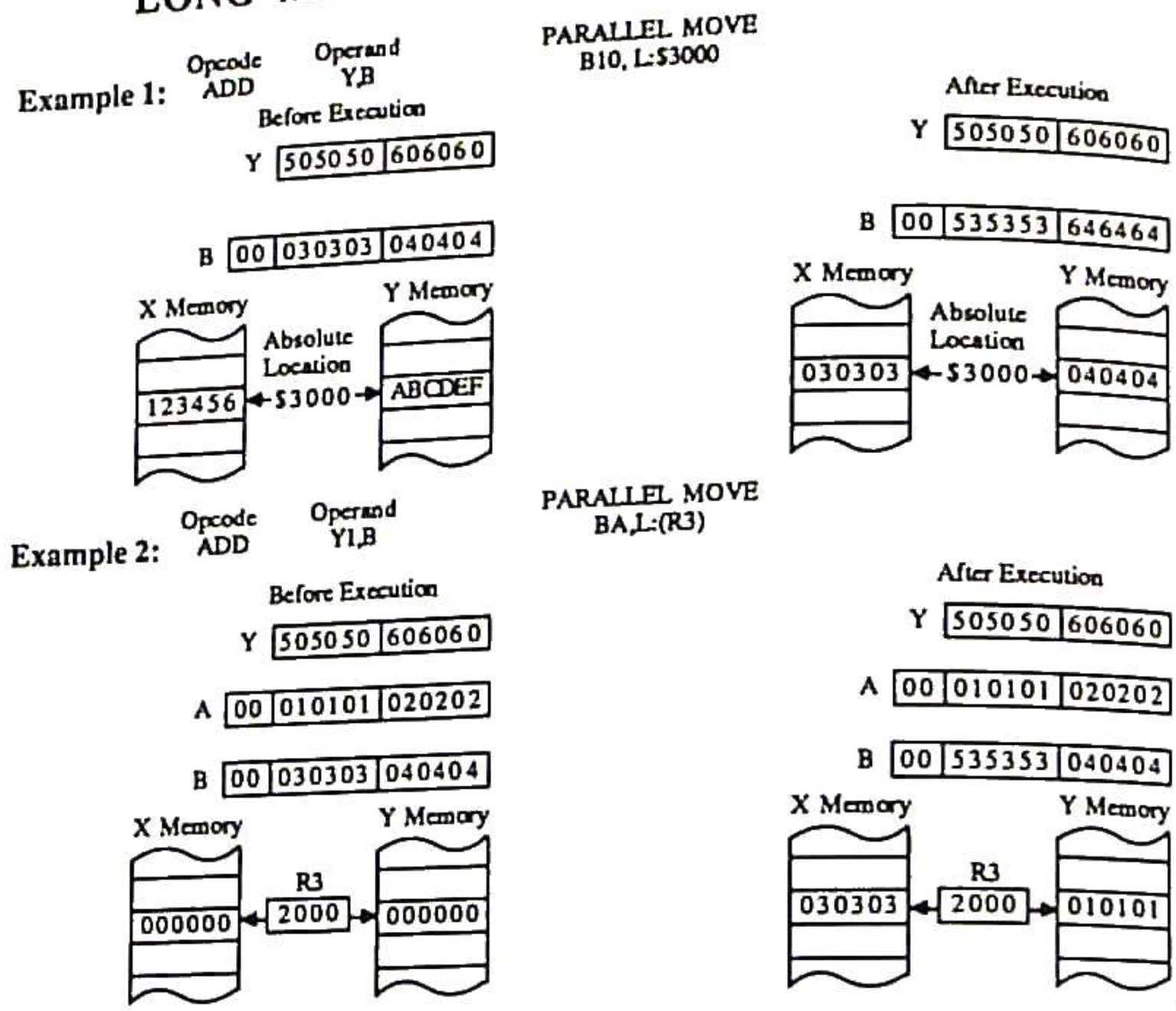
X,Y, A,A10, B,B10 — One 48 Bit Move

AB,BA — Two 24 Bit Moves

- The Only Parallel Move that References A10, B10, X, Y, AB, and BA.
- Useful for:
  - Double Precision, 48 Bit Long Word Transfers.
  - Complex Number (Real:Imaginary), Two Word Transfers.

NCD02-0519

LONG MEMORY DATA MOVE EXAMPLES



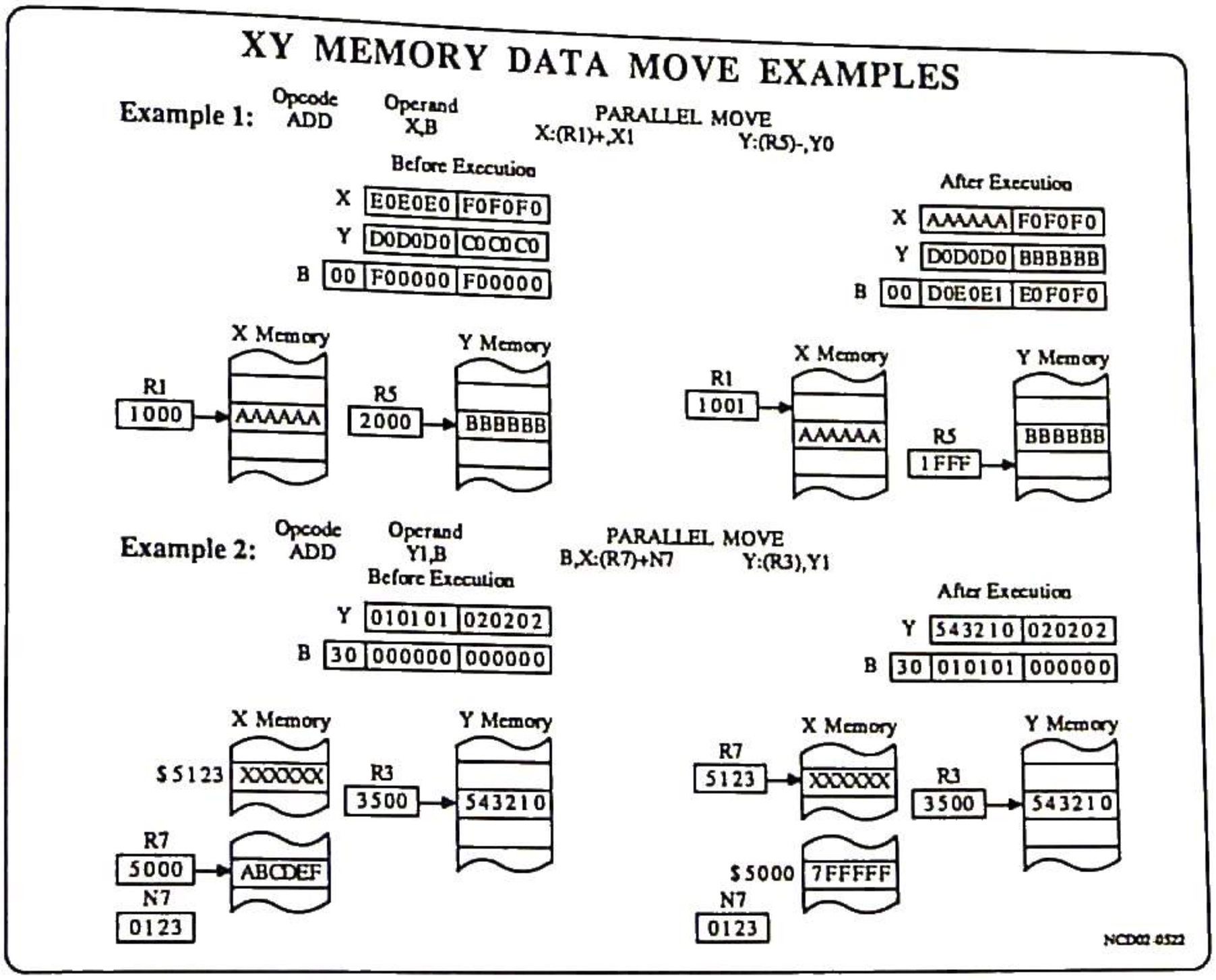
XY MEMORY DATA MOVE

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:		Source 1 or Destination 1	Source 2 or Destination 2
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		

XY MEMORY DATA MOVE EXAMPLES



PARALLEL MOVE SUMMARY

	Opcode/Operands	Parallel Move Examples	
Immediate Short Data	ADD X0,A	#$05,Y1	
Register to Register	ADD X0,A	A1,Y0	
Address Register Update	ADD X0,A	(R0)+N0	
X Memory	ADD X0,A	X0,X:(R3)+	
X Memory plus Register	ADD X0,A	X:(R4)–,X1	A,Y0
Y Memory	ADD X0,A	Y:(R6)+N6,X0	
Y Memory plus Register	ADD X0,A	A,X0	B,Y:(R0)
L Memory	ADD X0,A	L:-(R2),AB	
XY Memory	ADD X0,A	X1,X:(R1) +N1	Y1,Y:(R5)+N5

NCD02-0523

MOVE INSTRUCTION

Mnemonic	Operation	Assembler Syntax
MOVE	Move Data Registers S → D	MOVE S,D

Operand Combinations

Source	Destination
All Parallel Move Combinations	

• Equivalent to a Data ALU NOP.

NCD02-0526

MOVE CONTROL REGISTER

Mnemonic	Operation	Assembler Syntax
MOVEC	Move Data Registers S → D	MOVE(C) S,D

Operand Combinations

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

Example: MOVE SSL,LC

• Refer to Instruction Set Details for Restrictions.

NCD02-0527

MOVE PROGRAM MEMORY

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

Operand Combinations

PROGRAM Memory, Effective Address		Source or Destination
Absolute Short Address	aa	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
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 1: MOVE M3, P:(R3)-N3

Example 2: MOVE P:$0000, LC

• Refer to Instruction Set Details for Restrictions.

NCD02-0528

MOVE PERIPHERAL DATA

Mnemonic	Operation	Assembler Syntax

MOVEP	Move Peripheral Data
	S → X or Y Peripheral
	X OR Y Peripheral → D
		MOVEP S,X:<pp>
		MOVEP S,Y:<pp>
		MOVEP X:<pp>,D
		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	(Ra)
Postincrement by 1	(Ra)+
Postdecrement by 1	(Ra)-
Postincrement by Offset Nn	(Ra)+Nn
Postdecrement by Offset Nn	(Ra)-Nn
Indexed by Offset Nn	(Ra+Nn)
Predecrement by 1	-(Ra)

X OR Y
I/O Space

Example 1: MOVEP X:(R3+N3), Y:$FFC0

Example 2: MOVEP X:$FFFF, A

• The Only Memory to Memory Transfer Instruction.

NCD02 0529

MOVEC MOVEM AND MOVEP USED WITH SSH

SSH As a Source

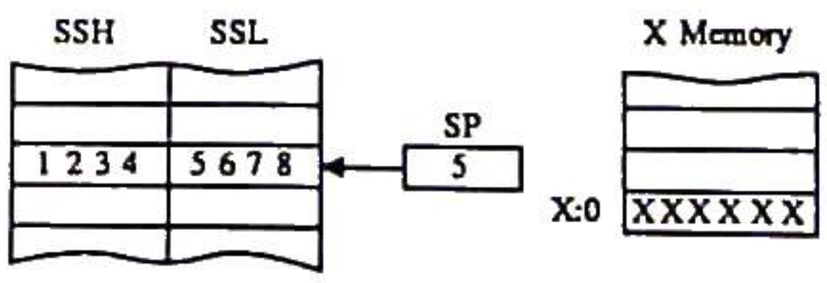
SSH → Destination
SP -1 → SP

SSH As a Destination

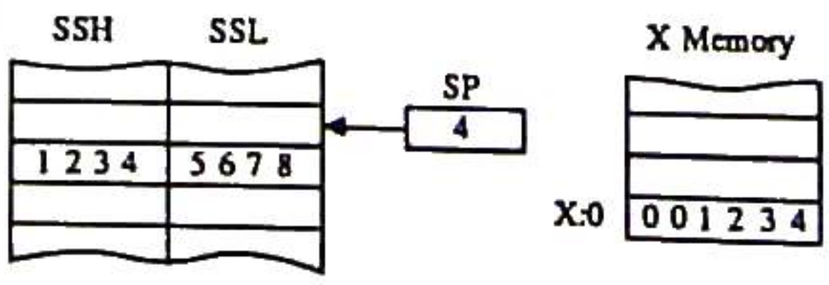
S P + 1 → SP
Source → SSH

Example 1: MOVEC SSH, X:0

Before Execution

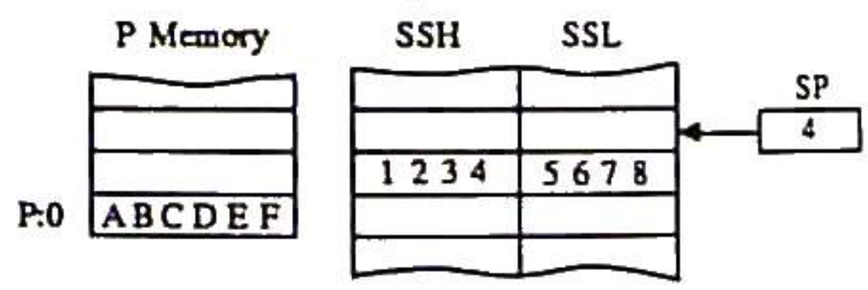


After Execution

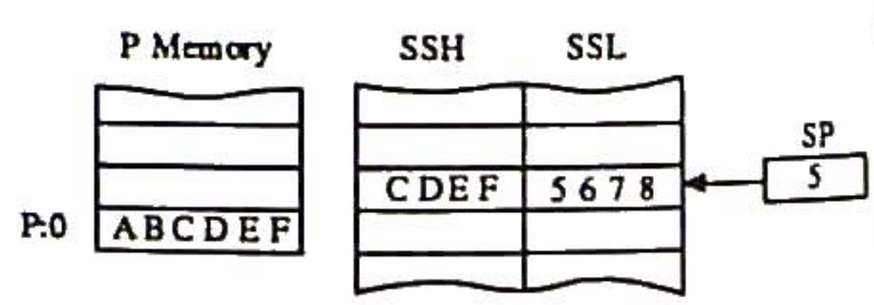


Example 2: MOVEM P:0, SSH

Before Execution



After Execution



NCD02 0530

LOAD UPDATED ADDRESS

Mnemonic	Operand Size	Operation	Assembler Syntax
LUA	16	Load Updated Address <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	

NOTE: The Source Effective Address is NOT Updated.

NCD02-0531

LOAD UPDATED ADDRESS EXAMPLES

Example 1: LUA (R4)-,R3

	Before Execution	After Execution
R3	X X X X	4 F F F
R4	5 0 0 0	5 0 0 0

Example 2: LUA (R1)-N1,N7

	Before Execution	After Execution
R1	7 A 3 D	7 A 3 D
N1	6 9 D C	6 9 D C
N7	X X X X	1 0 6 1

NCD02-0532

ADDITION AND SUBTRACTION ARITHMETIC INSTRUCTIONS

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

Operand Combinations

A,B	Y, A or B
B,A	Y1,A or B
X,A or B	Y2,A or B
X1,A or B	
X2,A or B	

NOTE: Adding or Subtracting an Accumulator from itself is NOT Allowed.

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

Operand Combinations

X,A	X,B
Y,A	Y,B

NCD02-0533-1

ADDITION AND SUBTRACTION ARITHMETIC INSTRUCTION EXAMPLES

Example 1: SUB X,B [Optional Parallel Move]

Before Execution

X [7FFFFF | FFFFFF]

B [FF | FFFFFF | FFFFFF]

After Execution

X [7FFFFF | FFFFFF]

B [FF | 800000 | 000000]

Example 2: ADC Y,A [Optional Parallel Move]

Before Execution

Carry [1]

Y [7FFFFF | FFFFFF]

A [0F | 800000 | 000000]

After Execution

Carry [0]

Y [7FFFFF | FFFFFF]

A [10 | 000000 | 000000]

NCD02-0534

SHIFT AND ADD OR SUBTRACT ARITHMETIC INSTRUCTIONS

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

Operand Combinations

Source	Destination
A	B
B	A

- The SHIFT Occurs Prior to the Arithmetic Operation.
- These Instructions are useful for Efficient Divide and Decimation in Time (DIT) Fast Fourier Transform (FFT) Algorithms.

NCD02-0535

SHIFT AND ADD OR SUBTRACT ARITHMETIC INSTRUCTION EXAMPLES

Example 1: ADDR A,B [Optional Parallel Move]

	Before Execution	After Execution
A	00 | 13579B | 000000	00 | 13579B | 000000
B	C0 | 000000 | 2468AC	E0 | 13579B | 123456

Example 2: SUBL A,B [Optional Parallel Move]

	Before Execution	After Execution
A	00 | 004000 | 000000	00 | 004000 | 000000
B	00 | 005000 | 000001	00 | 006000 | 000002

NCD02-0536

MULTIPLY ARITHMETIC INSTRUCTIONS

Mnemonic	Operand Size	Operation	Assembler Syntax
MPY	24, 56	Signed Multiply ± (S1 • S2) → D	MPY (±) S1, S2, D [Parallel Move]
MPYR	24, 56	Signed Multiply and Round ± (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 <u>not</u> valid.

NCD02-0537-1

MULTIPLY ARITHMETIC INSTRUCTION EXAMPLES

Example 1: MPY -X0,X0,A [Optional Parallel Move]

Before Execution

X | ABCDEF | 400000 | X0 = 1/2

A | 12 | 345678 | 9ABCDE

After Execution

X | ABCDEF | 400000

A | FF | E00000 | 000000 | A = -1/4

Example 2: MPY Y1, Y0, B [Optional Parallel Move]

Before Execution

Y | 400000 | 000007

Y1 = 1/2

B | 12 | 345678 | ABCDEF

After Execution

Y | 400000 | 000007

B | 00 | 000003 | 800000

Example 3: MPYR Y1, Y0, B [Optional Parallel Move]

Before Execution

Y | 400000 | 000007

Y1 = 1/2

B | 12 | 345678 | ABCDEF

After Execution

Y | 400000 | 000007

B | 00 | 000004 | 000000

NCD02-0538

MULTIPLY AND ACCUMULATE INSTRUCTIONS

Mnemonic	Operand Size	Operation	Assembler Syntax
MAC	24, 56	$D \pm (S1 \bullet S2) \longrightarrow D$	MAC ± S1, S2, D [Parallel Move]
MACR	24, 56	$D \pm (S1 \bullet S2) + r \longrightarrow D$	MACR ± 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 Source 1 and Source 2 are valid. X1 * X1 and Y1 * Y1 are <u>not</u> valid.
- These Instructions are the Basis for <u>All</u> Digital Filtering Algorithms!

NCD02-0539

MULTIPLY AND ACCUMULATE INSTRUCTION EXAMPLES

Example 1: MAC X1, Y1, A [Optional Parallel Move]

Before Execution

X	000002	000000
Y	000003	000000
A	00 000004	000000

After Execution

X	000002	000000
Y	000003	000000
A	00 000004	00000C

Example 2: MACR X0, Y0, B [Optional Parallel Move]

Before Execution

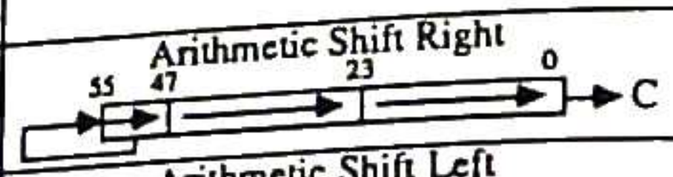
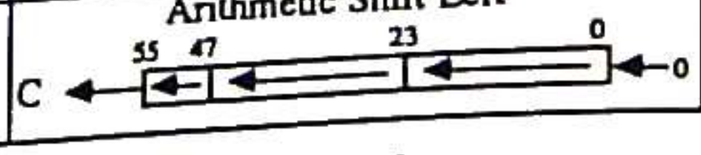
X	200000	400000
		X0 = 1/2
Y	010000	000003
B	00 000004	000000

After Execution

X	200000	400000
Y	010000	000003
B	00 000006	000000

NCD02-0540

ARITHMETIC SHIFT

Mnemonic	Operand Size	Operation	Assembler Syntax
ASR	56	Arithmetic Shift Right 	ASR D [Parallel Move]
ASL	56	Arithmetic Shift Left 	ASL D [Parallel Move]

Destination
A,B

Example 1: ASR A

Before Execution

A | 88 | 888888 | 111111 | Carry 0

After Execution

A | C4 | 444444 | 088888 | Carry 1

Example 2: ASL B

Before Execution

Carry 0 | B | C0 | F11111 | 888888

After Execution

Carry 1 | B | 81 | E22223 | 111110

NCD02-0541

COMPARE ARITHMETIC INSTRUCTIONS

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

NCD02-0542

AFFECT OF ARITHMETIC INSTRUCTIONS ON CONDITION CODES

Instruction Syntax:

Opcode Operand — Condition Code Register

Parallel Move — Limit Bit

Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LF	0	T	0	S1	S0	I1	I0	0	L	E	U	N	Z	V	C
MR (No Effect)								CCR							

L - Set if limiting (parallel move) or overflow has occured in result (Latched Bit)

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 occured in result

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

• The definition of the E and U bits varies according to the scaling mode being used.

NCD02-0543

TRANSFER INSTRUCTIONS

Mnemonic	Operand Size	Operation	Assembler Syntax
TFR	24, 56	Transfer Data ALU Register S → D	TFR S,D [Parallel Move]
Tcc	16, 24, 56	Conditionally Transfer Data ALU Registers If cc Then S1 → D1 or 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	Z= 0
NR -	normalized	$Z+(\overline{U} \cdot \overline{E}) = 1$
PL -	plus	N= 0
NN -	not normalized	$Z+(\overline{U} \cdot \overline{E}) = 0$

NCD02-0544

TRANSFER EXAMPLE

Example: TFR A, B A, X: (R6) + N6 B, Y:(R3)-

Status Register

15							8	7							0
LF	0	T	0	I	0	I1	I0	0	L	E	U	N	Z	V	C

S1, S0 – Scale Up

Before Execution

A | 00 | 4080A0 | C0E000

B | 00 | 203050 | A0B0C0

R6 | 7101 R3 | 0001

N6 | FFFF

X Memory $7101: XXXXXX

Y Memory $0001: XXXXXX

After Execution

A | 00 | 4080A0 | C0E000

B | 00 | 4080A0 | C0E000

R6 | 7100 R3 | 0000

N6 | FFFF

X Memory $7101: 7FFFFF

Y Memory $0001: 4060A1

- Limiting has occurred on the transfer to X Memory.
- TFR using the Parallel MOVE option creates a three Move Instruction.

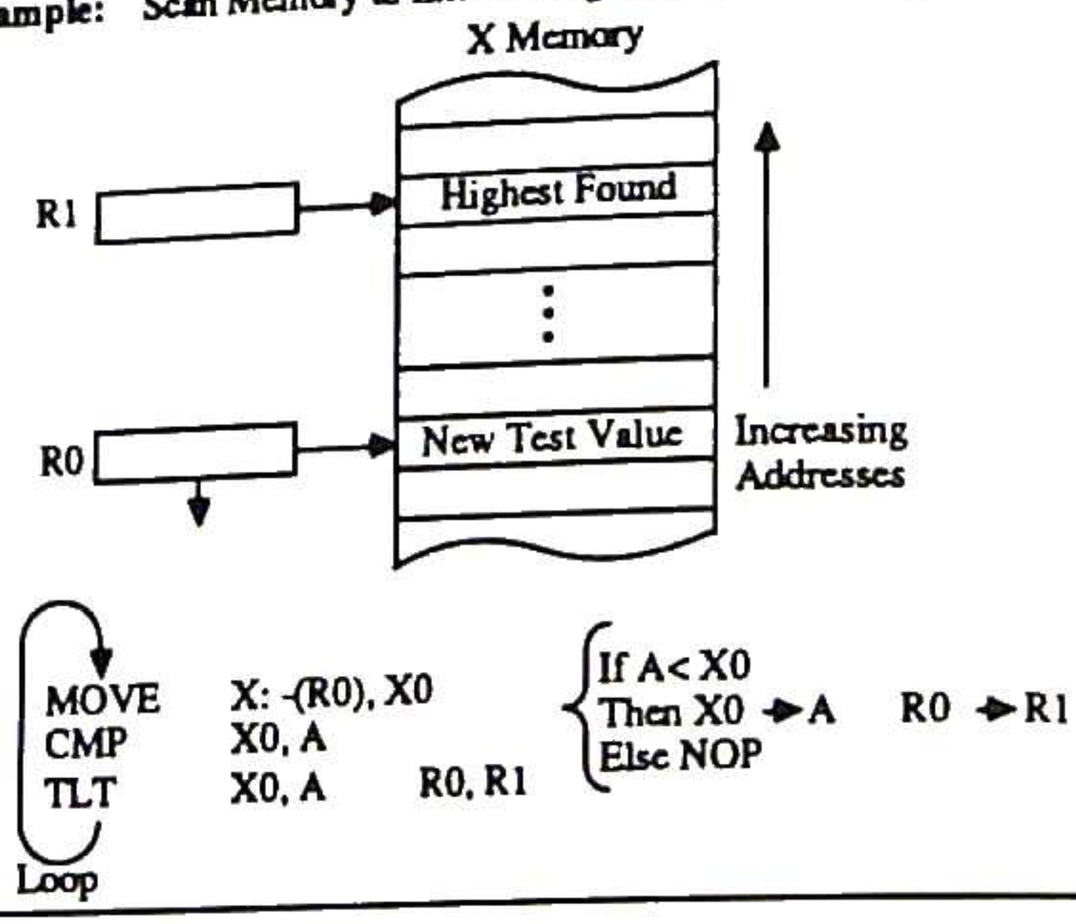
NCD02-0545-1

CONDITIONAL TRANSFER EXAMPLE

- 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: Scan Memory to find the largest value in an array.



NCD02-0546

ARITHMETIC INSTRUCTIONS

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

Operand Combinations

Destination	Register Rn
A,B	R0-R7

NCD02-0547-1

ARITHMETIC INSTRUCTION EXAMPLES

Example 1: NEG A

[Optional Parallel Move]

Before Execution

A | 00 | 0012F0 | 000000

After Execution

A | FF | FFED10 | 000000

Example 2: Normalize Accumulator A.

	R0	A
NORM R0, A	0000	001000 001248
NORM R0, A	FFFF	002000 002490
NORM R0, A	FFFE	004000 004920
	FFFE	004000 004920

- R0 Contains the Number of Bits shifted to Normalize the Accumulator
  If R0<0 then the Shift was to the left.
  If R0>0 then the Shift was to the right.
- Only does one bit of normalization at a time.
  Maximum to normalization is >46.

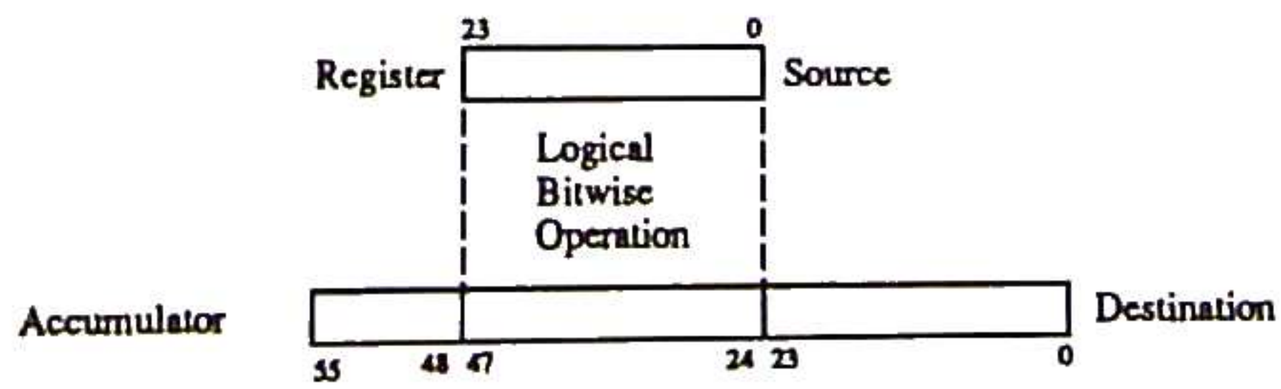
NCD02-0548-1

LOGICAL INSTRUCTIONS

Mnemonic	Operand Size	Operation	Assembler Syntax
AND	24	Logical AND S • D [47:24] → D [47:24]	AND S, D [Parallel Move]
OR	24	Logical Inclusive OR S + D [47:24] → D [47:24]	OR S, D [Parallel Move]
EOR	24	Logical Exclusive OR S ⊕ D [47:24] → D [47:24]	EOR S, D [Parallel Move]
NOT	24	Logical Complement $\overline{D\,[47:24]}$ → D [47:24]	NOT D [Parallel Move]

Operand Combinations

Source	Destination
X1, X0 Y1, Y0	A, B



NCD02-0549

LOGICAL INSTRUCTION EXAMPLES

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

NCD02-0550

IMMEDIATE LOGICAL INSTRUCTIONS

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

Operands
Destinations

MR
CCR
OMR

- Does not support parallel moves.
- If destination is the MR or OMR, then the CCR is not affected.
- If the CCR is the destination, then bits 0 through 6 are set or cleared according to the operation.

NCD02-0551

IMMEDIATE LOGICAL INSTRUCTION EXAMPLES

Example 1: ANDI #%11111100, OMR

- Puts processor in Mode 0
- Does not change other bits

Example 2: ORI #%00000010, OMR

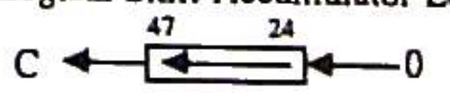
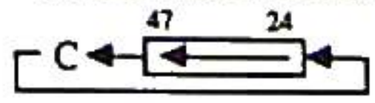
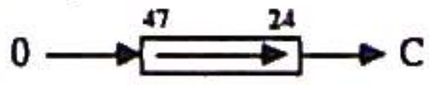
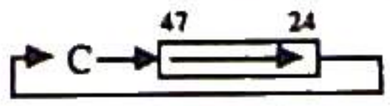
- Sets MB Bit, all others are unchanged

Example 3: ORI #%00001111, CCR

- Sets N, Z, V, and C bits in the CCR.
- The L, E, and U bits are unchanged.

NCD02-0552-1

SHIFT AND ROTATE INSTRUCTIONS

Mnemonic	Operand Size	Operation	Assembler Syntax
LSL	24	Logical Shift Accumulator Left 	LSL D [Parallel Move]
ROL	24	Rotate Accumulator Left 	ROL D [Parallel Move]
LSR	24	Logical Shift Accumulator Right 	LSR D [Parallel Move]
ROR	24	Rotate Accumulator Right 	ROR D [Parallel Move]

Operands
Destination
A, B

NCD02-0553

SHIFT AND ROTATE INSTRUCTION EXAMPLES

Example 1: LSR A

Before Execution

A | 80 | 822222 | 111111

Carry 1

After Execution

A | 80 | 411111 | 111111

Carry 0

Example 2: ROL B

Before Execution

B | F0 | 00000E | F0000F

Carry 1

After Execution

B | F0 | 00001D | F0000F

Carry 0

NCD02-0554-1

REPEAT NEXT INSTRUCTON

Mnemonic	Operation	Assembler Syntax
REP	LC → TEMP; #xxx → LC Repeat Next Instruction Until LC = 1 TEMP → LC	REP #xxx REP X:<ea> REP Y:<ea> 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
(12 Bits, Zero Extended)	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

NCD02-0561

REPEAT NEXT INSTRUCTION EXAMPLES

Example 1:

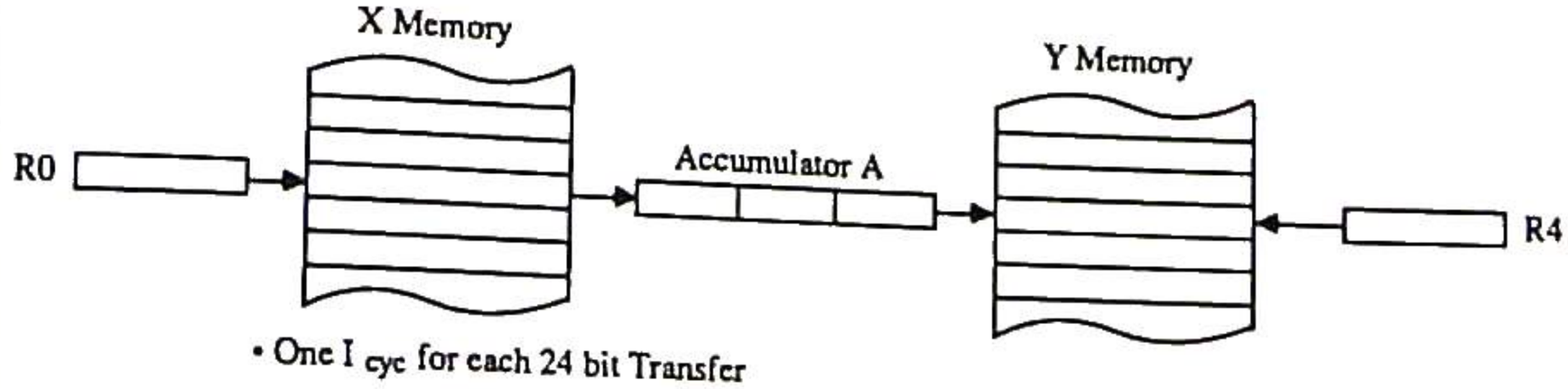
```
REP   X0                                  ; Take the 16 Least Significant bits of X0
MAC   X1,Y1,A   X:(R1)+,X1   Y:(R4)+,Y1   ; and repeat that many times
                                          ; X1*Y1+A ---> A, update X1,Y1
```

Example 2: Multiple Bit Shifts

```
REP #N
ASR   A
                                          ; N is the number of right shifts
```

Example 3: Memory to Memory Block Move

```
REP #N
MOVE  X:(R0)+,A      A, Y:(R4)+ ;         ; N is the size of the block moved.
```



• One I_{cyc} for each 24 bit Transfer

NCD02-0562

CHARACTERISTICS OF THE REP INSTRUCTION

- Only single word instructions are repeatable.
- REP and the following instruction are fetched once and stored internally.
- REP is NOT Interruptable!
- LC = 0 specifies the maximum repetition of 65,536 times.
- If LC = SSH, then SSH is read and the SP is postdecremented.
- For dynamic LC sources that are greater than 16 bits, use the appropriate 16 bit destination transfer rules. i.e. Scaling, Limiting, Truncation
- REP cannot be the last instruction of a DO loop. i.e. at LA
- The following instructions are NOT Repeatable:

DO
Jcc
JCLR
JMP
JSET
JScc
JSCLR
JSR
JSSET
REP
RTI
RTS
STOP
SWI
WAIT

NCD02-0563

HARDWARE DO LOOPS

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

#xxx Static Loop Count

Immediate Short Data #xxx
(12 bits, Zero Extended)

X or Y Dynamic Loop Count Effective Address MODES:

	ea
Absolute Short Address	
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

NCD02-0564

DO LOOP EXAMPLE

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 LOOP4
                  ⋮
        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 INSTRUCTON IN LOOP1
END1                                ;FIRST INSTRUCTION AFTER LOOP 1
```

NCD02-0565

END CURRENT DO LOOP

Mnemonic	Operation	Assembler Syntax
ENDDO	SSL(LF) → SR; SP-1 → SP 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
DO               JNE      ONWARD       ;go to ONWARD if LC not equal to Y1
Loop             ENDDO                 ;LC equal to Y1, restore all DO registers  <---- Abort DO Loop
                 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)
```

NCD02-0566

DO DON'T DO'S

Immediately before DO	MOVEC to LA, LC, SSH, SSL OR SP MOVEM to LA, LC, SSH, SSL, or SP MOVEP to LA, LC, SSH, SSL or SP MOVEC from SSH MOVEM from SSH MOVEP from SSH
At LA-2, LA-1 and LA	DO MOVEC from SSH MOVEM from SSH MOVEP from SSH MOVEC to LA, LC, SR, SP, SSH or SSL MOVEM to LA, LC, SR, SP, SSH or SSL MOVEP to LA, LC, SR, SP, SSH or SSL ANDI MR ORI MR
At LA-1	ENDDO
At LA	any two-word instruction* Jcc JMP JScc JSR REP RESET RTI RTS STOP WAIT
Other Restrictions	DO SSH,xxxx JSR to (LA) whenever the Loop Flag (LF)is set JScc to (LA) whenever the Loop Flag (LF) is set JSCLR to (LA) whenever the Loop Flag (LF) is set JSSET to (LA) whenever the Loop Flag (LF) is set A DO instruction cannot be repeated using the REP instruction.

* This restriction applies to the situation in which the DSP56000/1 Simulator's single line assembler is used to change the last instruction in a DO loop from a one-word instruction to a two-word instruction.

NCD02-0567

BRANCHING INSTRUCTIONS

Mnemonic	Operation	Assembler Syntax
JMP	Jump <ea> → PC	JMP <ea>
Jcc	Conditional Jump If cc then <ea> → PC Else PC+1 → PC	Jcc <ea>
JSR	Jump to Subroutine SP + 1 → SP PC → SSH SR → SSL <ea> → PC	JSR <ea>

JScc	Conditional Jump to Subroutine
	If cc then
	SP + 1 → SP
	PC → SSH
	SR → SSL
	<ea> → PC
	Else
	PC + 1 → PC
		JScc <ea>

RTS	Return From Subroutine SSH → PC SP - 1 → SP	RTS
RTI	Return From Interrupt SSH → PC SSL → SR SP - 1 → SP	RTI

NCD02-0555-1

BRANCHING ADDRESSING MODES AND CONDITIONS

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	$Z+(\overline{U} \cdot \overline{E})=0$

NCD02-0556

BIT MANIPULATION INSTRUCTIONS

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

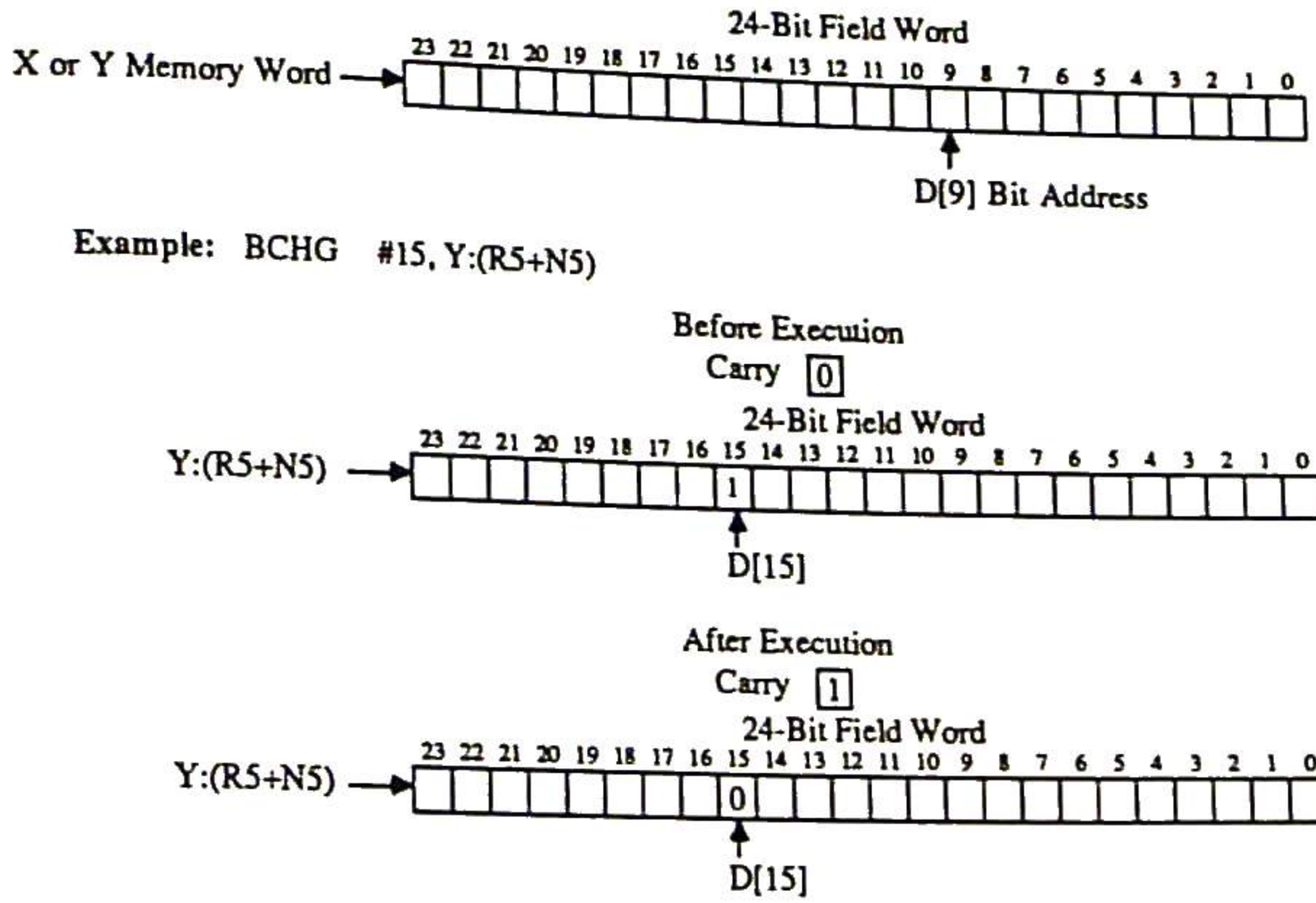
• Generates Indivisible Read-Modify-Write Bus Cycle

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)

• D[n] is the nth bit of the 24 bit destination operand field. $0 \le n \le 23$

NCD02-0357

BIT MANIPULATION EXAMPLE



NCD02-0558

BIT FIELD BRANCHING INSTRUCTIONS

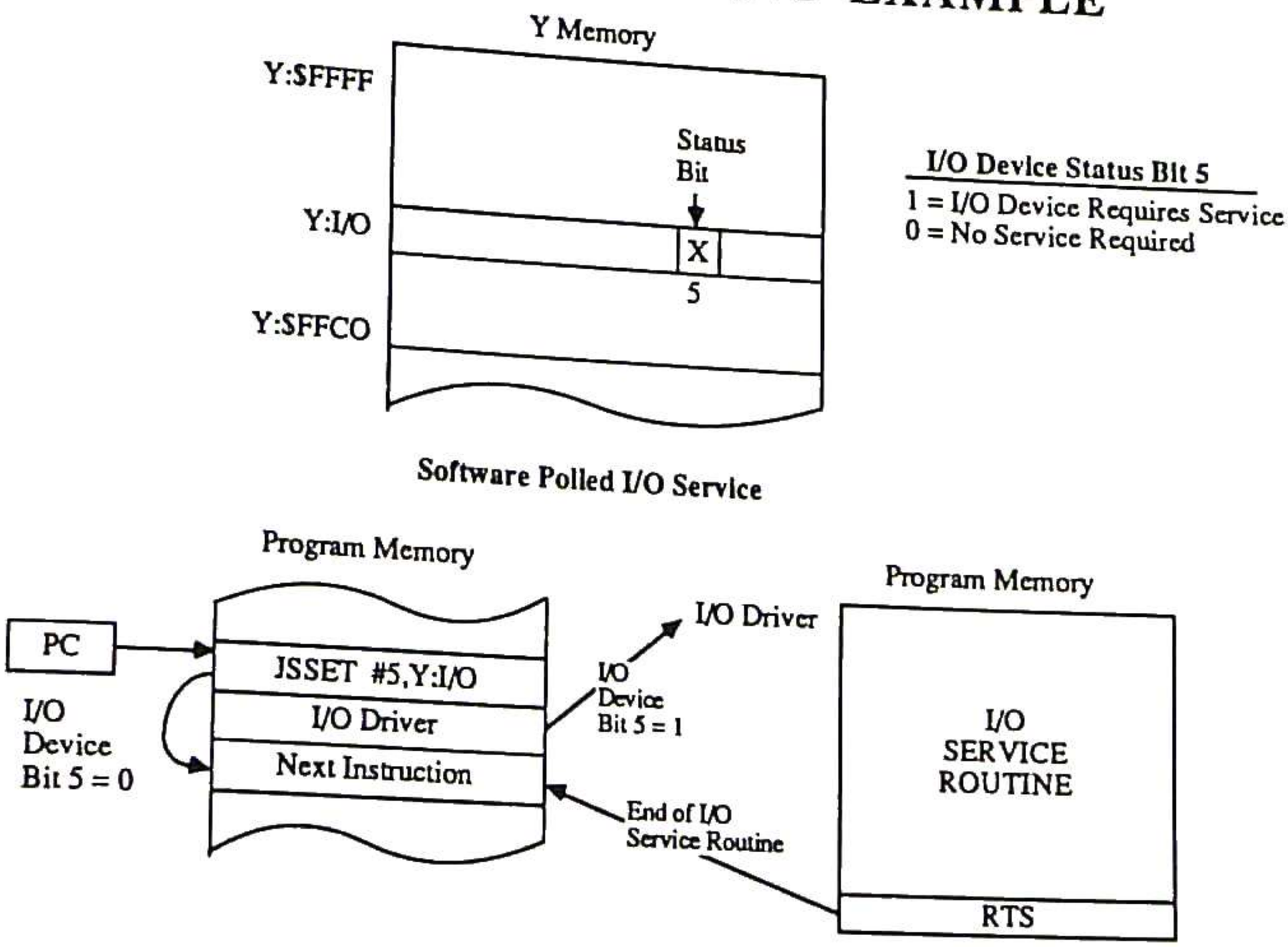
Mnemonic	Operation	Assembler Syntax
JSET	Jump if Bit Set If S[n] =1 Then xxxx → PC Else PC+1 → PC	JSET #n,X:<ea>, xxxx JSET #n,Y:<ea>, xxxx
JCLR	Jump if Bit Clear If S[n] =0 Then xxxx → PC Else PC+1 → PC	JCLR #n,X:<ea>, xxxx JCLR #n,Y:<ea>, xxxx
JSSET	Jump to Subroutine if Bit Set If S[n] =1 Then SP+1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

NCD02-059

BIT FIELD BRANCHING EXAMPLE



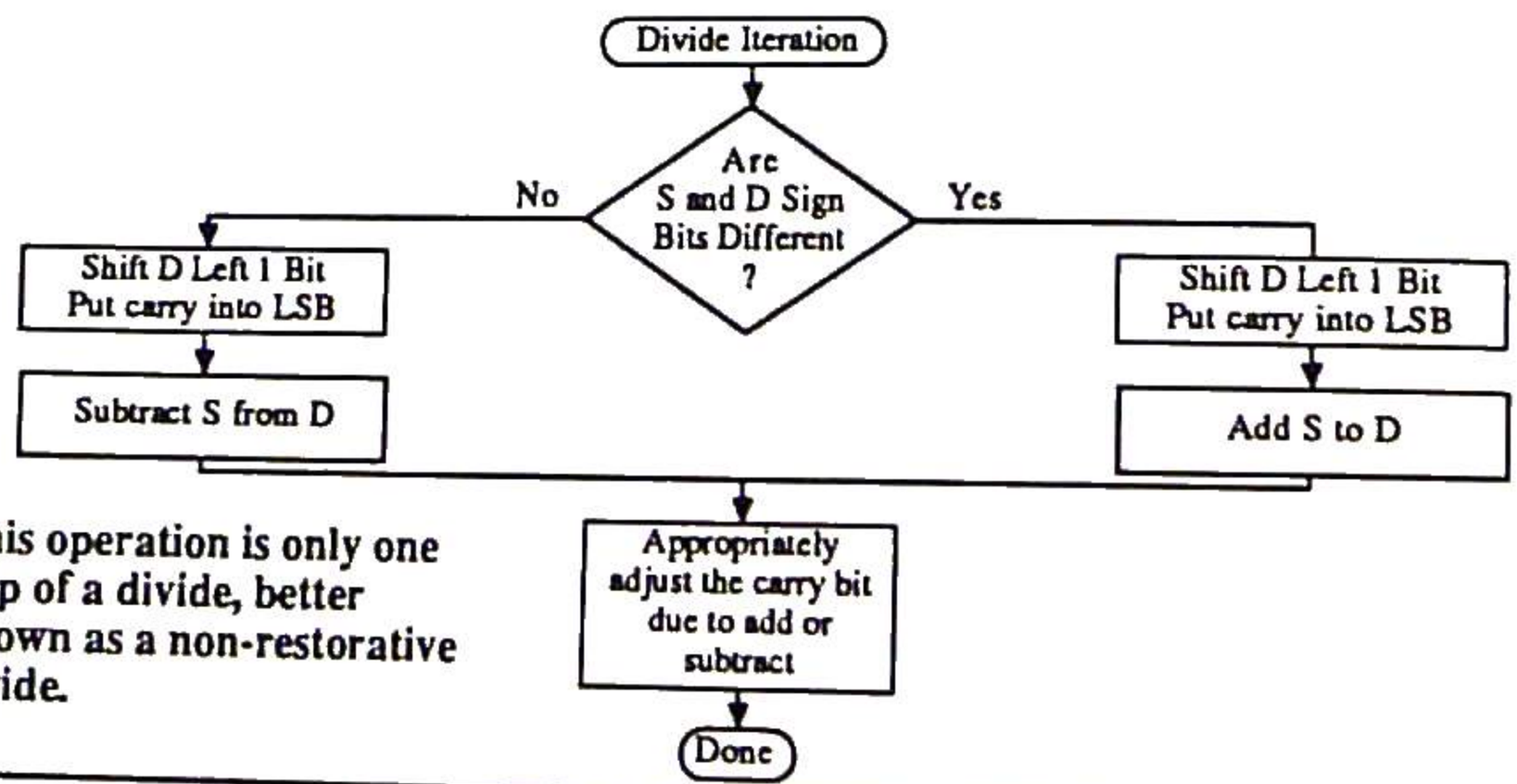
NCD02-0560

DIVIDE ITERATION

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

Operand Combinations

Source	Destinations
X1, X0	A,B
Y1, Y0	A,B



- This operation is only one step of a divide, better known as a non-restorative divide.

NCD02-0569-2

ONE QUADRANT DIVISION EXAMPLE

Example: 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
```

Before Execution

X	XXXXXX	600000

X0 = 3/4 = 0.75

A	00	400000	000000

A = 1/2 = 0.5

After Execution

$$\frac{A}{X0} = \frac{\frac{1}{2}}{\frac{3}{4}} = \frac{2}{3} = 0.666666\ldots$$

		L.S.P. Remainder	Quotient
X	XXXXXX	600000	
A	00	400000	555555

$0.000000400000

Quotient = $0.555555 = 0.666666626930237

Remainder = $0.000000400000 = 0.000000029802322

NCD02-0570-1

ONE QUADRANT DIVISION ACCUMULATOR TRACE

	A2	A1	A0
Start	00	400000	000000
DIV X0,A 1	00	200000	000000
2	FF	E00000	000001
3	00	200000	000002
4	FF	E00000	000005
5	00	200000	00000A
6	FF	E00000	000015
7	00	200000	00002A
8	FF	E00000	000055
9	00	200000	0000AA
10	FF	E00000	000155
11	00	200000	0002AA
12	FF	E00000	000555
13	00	200000	000AAA
14	FF	E00000	001555
15	00	200000	002AAA
16	FF	E00000	005555
17	00	200000	00AAAA
18	FF	E00000	015555
19	00	200000	02AAAA
20	FF	E00000	055555
21	00	200000	0AAAAA
22	FF	E00000	155555
23	00	200000	2AAAAA
24	FF	E00000	555555
ADD X0, A	00	400000	555555
		L.S.P. Remainder	Quotient

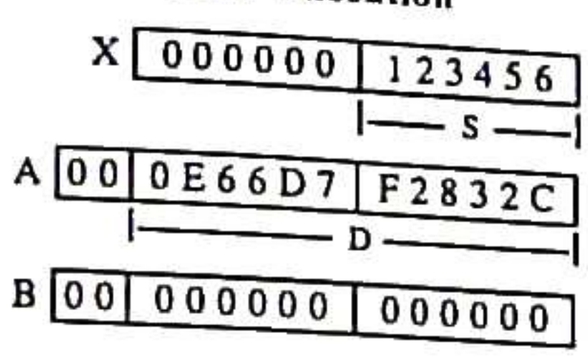
NCD02-0571

FOUR QUADRANT DIVISION EXAMPLE

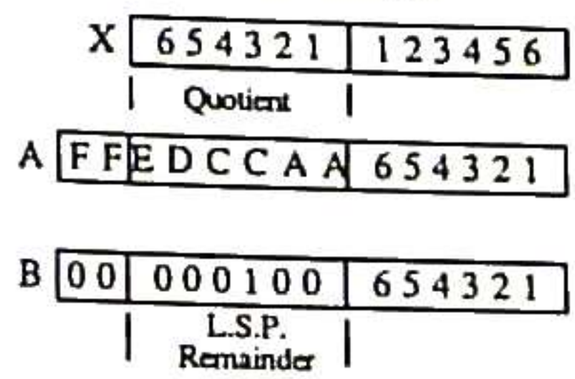
Example: (24-bit signed quotient, 48-bit signed remainder)

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

Before Execution



After Execution



NCD02-0572

CHARACTERISTICS OF DIVIDE ITERATION

- The 48 bit dividend (destination) must be a positive fraction, sign extended to 56 bits.
- The 24 bit divisor (Source) must be a positive fraction.
- Valid results are obtained only when the operands are interpreted as fractions and $|D| < |S|$.
- Each DIV operation calculates one bit of quotient using a non restoring fractional division algorithm.
- For an N bit quotient, DIV is executed N times ($1 \le N \le 24$) and produces a 48 bit remainder with 48-N bits of precision.
- In DSP, there is not a real need for a general purpose divide, other than dissemination in time. But DSP is a series of summations, delays, and multiplies.

For more information

Theory and Application of Digital Signal Processing, by Rabner and Gold (Prentice-Hall, 1975) pp. 524-530.

Computer Architecture and Organization, by John Hayes, (McGraw Hill, 1978) pp. 190-199.

Computer Arithmetic: Principals, Architecture, and Design, by Kai Hwang (John Wiley and Sons, 1979), pp. 213-223.

NCD02-0573-2

MISCELLANEOUS INSTRUCTIONS

Mnemonic	Operation	Assembler Syntax
NOP	No Operation PC+1 → PC	NOP
RESET	Reset on Chip Peripheral Devices 1 → I1 1 → I0 0 → X:$FFFF 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 Clock Oscillator	STOP
SWI	Software Interrupt Begin SWI Exception Processing	SWI

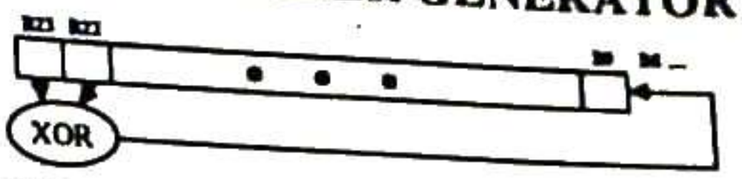
NCD02-0568-1

BINARY TO GRAY CODE CONVERTER

```
          org     p:$100
start
          clr     b          #>1,x1          ;clear count, load increment
          move               #0,r0           ;pointer to memory
          move               #-1,m0          ;linear addressing
          do      #64,_endtest
          tfr     b,a                        ;value to convert
;
;   This converts a binary value in register A1 to a GRAY coded
;   number in A1.  Register X0 is used as a temporary register.
;
          lsr     a          a1,x0           ;shift bits, copy a
          eor     x0,a                       ;xor adjacent bits to do gray code
          add     x1,b       a1,x:(r0)+      ;inc b, save result in memory
_endtest
          end

      display x:0#64
      X:$0000 000000 000001 000003 000002 000006 000007 000005 000004
      X:$0008 00000C 00000D 00000F 00000E 00000A 00000B 000009 000008
      X:$0010 000018 000019 00001B 00001A 00001E 00001F 00001D 00001C
      X:$0018 000014 000015 000017 000016 000012 000013 000011 000010
      X:$0020 000030 000031 000033 000032 000036 000037 000035 000034
      X:$0028 00003C 00003D 00003F 00003E 00003A 00003B 000039 000038
      X:$0030 000028 000029 00002B 00002A 00002E 00002F 00002D 00002C
      X:$0038 000024 000025 000027 000026 000022 000023 000021 000020
```

NCD02-0574

RANDOM NUMBER GENERATOR




```
;
;  15 BIT PSEUDO RANDOM NUMBER GENERATOR
;
          ORG     X:0
NSTATE    DS      1                                  ;STATE OF NOISE GENERATOR (SHIFT REG)
          ORG     P:$100
INIT
          MOVE                    #$FF,X0            ;INITIAL STATE OF NOISE GENERATOR
          MOVE                    X0,X:NSTATE
GET
          JSR     NOISE                              ;GENERATE NOISE SAMPLE
          MOVE                    X:NSTATE,A         ;GET NOISE
          MOVEP                   A1,Y:$FFFF         ;OUTPUT NOISE SAMPLE
          JMP     GET                                ;LOOP FOREVER
;
;       NOISE GENERATION SUBROUTINE
;
NOISE
                                                     ;DO NOISE GENERATION LOOP
          CLR     B               X:NSTATE,A         ;GET A ZERO IN B
          TST     A               #>$000100,X0       ;CHECK A, GET A 1 IN LSB POSITION
          TNR     X0,B                               ;IF NORMALIZED, THEN XOR=1
          ADD     B,A                                ;SET LSB
          ASL     A                                  ;SHIFT BITS
          MOVE                    A1,X:NSTATE        ;SAVE STATE
          RTS                                        ;RETURN WITH STATE
          END
```

FE0000	001000	400000	060000	005000	400600
FC0000	002000	800200	0C0000	00A000	800E00
F80000	004000	000600	180000	014000	001E00
F00000	008000	000C00	300000	028000	003C00
E00000	010000	001800	600000	050000	007800
C00000	020000	003000	C00200	0A0000	00F000
800000	040000	006000	800400	140000	01E000
000200	080000	00C000	000A00	280000	03C000
000400	100000	018000	001400	500000	078000
000800	200000	030000	002800	A00200	0F0000

NCD02-0575

SQUARE ROOT

```
;
;       Does a square root by polynomial aproximation, 10 bit accuracy.
;                        c2              c1              c0
;       sqrt(x) = -.1985987*x*x + 0.8803385*x + 0.3185231
;
;       valid for:  .5<= x < 1.0
;       input value in x0, output in register A.
;       r1 initially points to the coefficients in y memory.
;
          org     y:0                                ;location in y memory of input
datin     ds      1                                  ;location in y memory of output
datout    ds      1
pcoef     dc      .8803385,-.1985987,.3185231        ;c1, c2, c0
;                                                    ;|
          org     p:$100                             ;r1, point to polynomial coefficients
start     move                    #pcoef,r1
          move                            y:datin,x0 ;get input value
;   polynomial evaluation
          mpyr    x0,x0,a         y:(r1)+,y0         ;a = x**2, c1 -> y0
          mpy     x0,y0,a   a,x1  y:(r1)+,y0         ;a = c1*x, x**2 -> x1, c2 -> y0
          macr    x1,y0,a         y:(r1)+,y0         ;a = c2*(x**2) + c1*x, c0 -> y0
          add     y0,a                               ;add c0
;                                 a,y:datout         ;write result to y memory
          move
          end

change y:0 .75
display y:0#5
Y:$0000 600000 000000 70AEEF E69451 28C55D
evaluate y:0
 Hex:600000 Dec:6291456 Fract:0.7500000 Bin:011000000000000000000000

          (Execute Square Root Program)

display y:0#5
Y:$0000 600000 6EFBFE 70AEEF E69451 28C55D
evaluate y:1
 Hex:6EFBFE Dec:7273470 Fract:0.8670652 Bin:011011101111101111111110
```

NCD02-0574-1

DSP56000
APPLICATIONS DEVELOPMENT
SYSTEM AND SIMULATOR

Objectives:

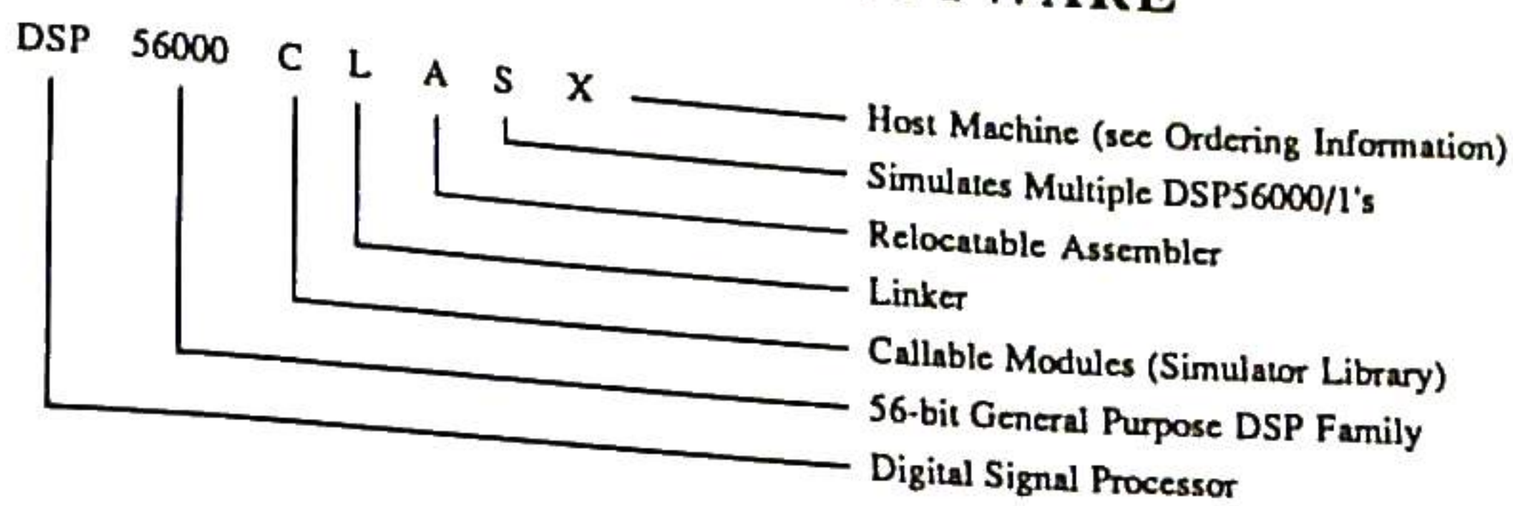
- Demonstrate how to start up the ADS.
- Describe the operation of each Data Manipulation Command.
- Describe the operation of each Debugging Command.

NCD02-0700-1

Table of Contents

Development Software
Development Hardware 6-5
Multiple ADM Hookup 6-5
Getting Started 6-6
Command Groups 6-6
Help 6-7
Command Parameters 6-7
Change 6-8
Change Examples 6-8
Display 6-9
Display Examples 6-9
Radix 6-10
Evaluate 6-10
Copy 6-11
Force 6-11
Assemble 6-12
Disassemble 6-12
Trace 6-13
Step 6-13
Go 6-14
Breakpoint 6-14
Debugging 6-15
6-16

DEVELOPMENT SOFTWARE

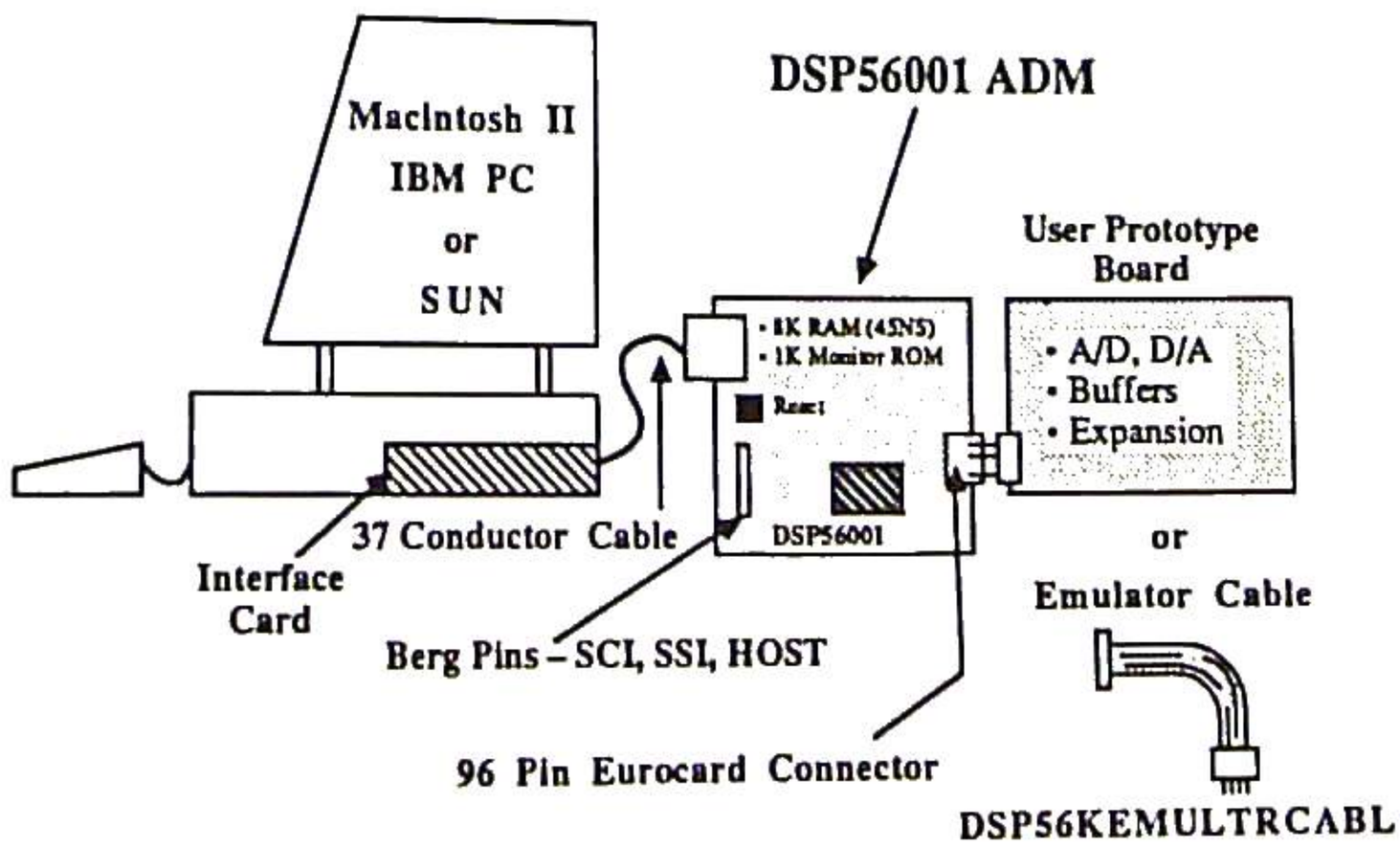


The ordering information is:

Host Platform	Operating System	Order Number
IBM PC	DOS 2.x, 3.x	DSP56000CLASA
Macintosh II	MAC OS 4.1	DSP56000CLASB
SUN-3	BSD 4.2	DSP56000CLASC
VAX	VMS 4.5	DSP56000CLASD
VAX	BSD 4.2	DSP56000CLASE

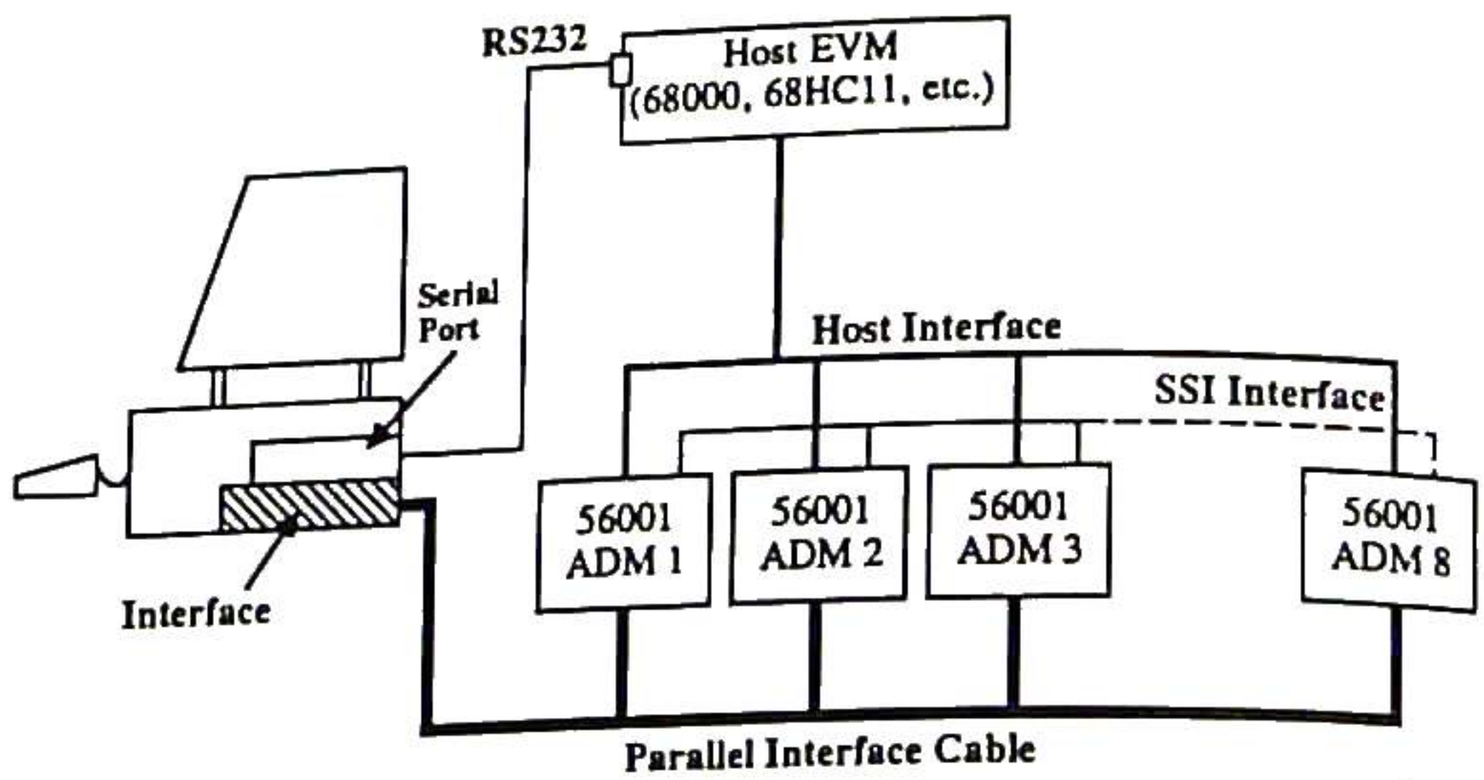
NCD02-0701

DEVELOPMENT HARDWARE



NCD02-0702

MULTIPLE ADM HOOKUP



NCD02-0703

GETTING STARTED

Simulator

```
C>SIM56000<CR>


MOTOROLA DSP56000 SIMULATOR:  VERSION 2.00 9-01-87
0>
asm  break  copy  disassemble  display  <space>=more
```

Applications Development System

```
C>ADS56000<CR>


ADM0>
asm  break  change  copy  device  disassemble  <space>=more
```

NCD02-0704

COMMAND GROUPS

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		

NCD02-0705

HELP

```
help
   --------- DSP56001 APPLICATION DEVELOPMENT SYSTEM COMMANDS ---------
 ASM [Dx] [(beginning at) addr] [assembler_mnemonic]
 BREAK [D0..7] [#bn] [addr|OFF] T[expr] [H(halt)/In(inc. CNTn)/N(note)/S(show)]
 CHANGE [D0..7] [reg[_block]/addr[_block] (to)[expression] [VERIFY]
 COPY (from) [Dx] addr[_block] (to)[D0..7] addr
 DISPLAY [D0..7] [ON/OFF/ALL/IO] [reg[_block/_group]/addr[_block]]...
 DISPLAY W(active ADMs)/V(version)
 DEVICE D0..7 [ON/OFF]
 DISASSEMBLE [D0..7] [addr[_block]]
 EVALUATE [Dx] [B(binary)/D(decimal)/F(fractional)/H(hexadecimal)] expression
 FORCE [D0..7] R(hardware reset)|B(Return to monitor)
 GO [D0..7] [(from)addr/R(reset)] [(to break #)#bn] [(occurrence):count]
 HELP [Dx] [command|register]
 INPUT [D0..7] [#number] OFF/TERM/filename
 LOAD [D0..7|S(state)] (from) filename
 LOG [D0..7] [OFF] [C(commands)/S(session) [filename]]
 OUTPUT [D0..7] [#number] OFF/TERM/filename
 PATH [D0..7] [pathname]
 PC (interrupt select)[IRQ3|IRQ4|IRQ7] (address)[IO [$100|$200|$300]]
 QUIT
 RADIX [B(binary)/D(decimal)/H(hexadecimal)]
 SAVE [D0..7] S(state)/addr_block... filename
 STEP [D0..7] [count]
 SYSTEM [system_command [parameter_list]]
 TRACE [D0..7] [count]
 WAIT [count(seconds)]
  Macro filename
  ;comment string entry
 Note: Trace/Step/Go commands use ADS Interrupts to PC
```

NCD02-0706

COMMAND PARAMETERS

radix = % for binary value, ' for decimal value, $ for hexadecimal

% 101101011100 '2908 $B5C

address = P:location|X:location|Y:location

P:$40 X:$100 Y:0

address_block = address..location|address#count

P:$40..$50 Y:0..100 X:$100#5

bn = (break number) decimal integer constant in the range 1 to 99.

b1 b14 b99

count = positive integer expression in range 1 to $7fffffff.

5 $10 % 1101

reg = A0|A1|A2|B0|B1|B2|BCR|HTX|HRX|HSR|HCR|HPICR|HPISR|HPIVR|
HPRXH|HPRXM|HPRXL|HPTXH|HPTXM|HPTXL|IPR|LA|M0|M1|
M2|M3|M4|M5|M6|M7|N0|N1|N2|N3|N4|N5|N6|N7|OMR|PBC|PBD|PBDDR|
PC|PCD|PCDDR|R0|R1|R2|R3|R4|R5|R6|R7|SCR|SCCR|SP|SR|SRX|
SSH|SSICA|SSICB|SSIRX|SSISR|SSITX|SSL|STX|TSR|X0|X1|Y0|Y1

reg_block = reg..reg

r0..r7 m3..m6 a0..a2

reg_group = ALL|DSP|HOST|IO|PORTB|PORTC|SCI|SSI

NCD02-0707

CHANGE

help change

```
---------- CHANGE: Change Register or Memory Value ----------

CHANGE [D0..7] [reg[_block]/addr[_block] (to)[expression] [VERIFY]

 change pc
 Display register values of the default ADM individually starting with
 the program counter and prompt the user for new values.

 change d2 x:$55
 Display ADM 2 x memory location hexadecimal 55 and prompt user for new value.

 change d3,4,7 p:$20 $123456
 Change ADM 3,4, and 7 p memory address hexadecimal 20 to hexadecimal 123456.

 change r0..r7 0 p:$30..$300 0 x:$fffe $55 pc 100
 Change the default ADM registers r0 through r7 to 0, p memory addresses
 hexadecimal 30 through 300 to 0, x memory address hex fffe to hex 55 and
 the program counter to 100 decimal.
```

NCD02-0708

CHANGE EXAMPLES

```
change x:0
change X:$0000 $000001 ;Dec:1   Fract:0.0000001
change X:$0001 $000002 ;Dec:2   Fract:0.0000002
change X:$0002 $000003 ;Dec:3   Fract:0.0000004
change X:$0003 $000004 ;Dec:4   Fract:0.0000005
change X:$0004 $000005 ;Dec:5   Fract:0.0000006
change X:$0005 $000006 ;Dec:6   Fract:0.0000007
change X:$0006 $000007 ;Dec:7   Fract:0.0000008
change X:$0007 $000008 ;Dec:8   Fract:0.0000010

display
   b2=     87     b1=429087     b0=087691
Y:$0000 259870
ADM#0 P:$0043 000000              = NOP

change b2 $33 b1 $444444 y:0 $ffffff b0 $555555

display
   b2=     33     b1=444444     b0=555555
Y:$0000 FFFFFF
ADM#0 P:$0043 000000              = NOP
```

NCD02-0709

DISPLAY

```
DISPLAY [D0..7] [ON/OFF/ALL/IO] [reg[_block/_group]/addr[_block]]...

---------- Register and Memory Display Modes ----------

  ON  Always display the following registers and memory locations.
  OFF Never display the following registers and memory locations.

---------- Examples ----------

display on
Enable all registers for display

display on p:0..20 x:30..40 y:$100
Display enable p memory address block 0 to 20, x memory address block
30 to 40 and y memory address hexadecimal 100.

display off
display on dsp host sci
Disable all display, then enable display of the DSP56000 programming model
registers, the HOST peripheral registers, and the SCI peripheral registers.

---------- Register Group Names ----------

  ALL   all registers.
  DSP   all of the DSP56000 programming model registers.
  IO    all peripheral register names.
  HOST  all registers of the Host peripheral interface.
  PORTB all registers of the Port B interface.
  PORTC all registers of the Port C interface.
  SCI   all registers of the SCI peripheral interface.
  SSI   all registers of the SSI peripheral interface.

display p:0..300
Immediate display of p memory addresses 0 through 300.

display all
Immediate display of all registers plus the enabled memory locations.
```

NCD02-0710

DISPLAY EXAMPLES

```
display

                x1=000000   x0=000000   r7=  0000   n7=  0000   m7=  FFFF
                y1=000000   y0=000000   r6=  0000   n6=  0000   m6=  FFFF
   a2=    00    a1=000000   a0=FFFFFF   r5=  0000   n5=  0000   m5=  FFFF
   b2=    00    b1=000000   b0=000000   r4=  0000   n4=  0000   m4=  FFFF
                                        r3=  0000   n3=  0000   m3=  FFFF
   pc=  0040    sr=  0300   omr=   02   r2=  0000   n2=  0000   m2=  FFFF
   la=  0000    lc=  000A               r1=  0000   n1=  0000   m1=  FFFF
  ssh=  FFFF   ssl=  FFFF   sp=    00   r0=  0000   n0=  0000   m0=  FFFF
 cnt1=000000  cnt2=000000  cnt3=000000  cnt4=000000


display p:$e000..$e00f x:$100#8 y:$180#8
P:$E000 0AF080 00E010 0AF080 00E28A 0AF080 00E2DD 0AF080 00E2A8
P:$E008 0AF080 00E2FB 000000 000000 000000 000000 000000 000000
X:$0100 7D7C00 797C00 757C00 717C00 6D7C00 697C00 657C00 617C00
Y:$0180 000000 FCDBD5 F9B827 F69570 F3742D F054D9 ED37F0 EA1DEC
```

NCD02-0711

RADIX

```
----------- RADIX: Change Default Number Base -----------

RADIX [B(binary)/D(decimal)/H(hexadecimal)]

 radix
 Display the default radix currently enabled.

 radix h
 Change default radix entry to hexadecimal.  Hexadecimal constant entries
 no longer require a preceding dollar sign ($) , but any decimal constants
 will require a preceding grave accent (`).
```

NCD02-0712

EVALUATE

---------- EVALUATE: Evaluate an Expression ----------

EVALUATE [Dx] [B(binary)/D(decimal)/F(fractional)/H(hexadecimal)] expression

Following is a list of valid operators useable in an expression:

<	less than	&&	logical 'and'
<=	less than or equal to	||	logical 'or'
==	equal to	!	logical 'negate'
>=	greater than or equal to	&	bitwise 'and'
>	greater than	|	bitwise 'or'
!=	not equal to	~	bitwise one's complement
+	addition	^	bitwise 'exclusive or'
-	subtraction	<<	shift left
/	division	>>	shift right

evaluate r0+p:$50
Add the value in r0 register to the value in program memory address hexadecimal 50 and display the result using the default radix.

evaluate h %10101010&p:r0
Calculate the bitwise AND of the program memory address specified by the value in r0 register and the binary value 10101010 and display the result in hexadecimal.

---------- Examples ----------

evaluate 10
Hex:00000A Dec:000010 Fract:0.0000012 Bin:000000000000000000001010

evaluate 0.31415926
Hex:28365F Dec:2635359 Fract:0.3141593 Bin:001010000011011001011111 NCD02-0713

COPY

---------- COPY: Copy a Memory Block ----------

COPY (from) [Dx] addr[_block] (to)[D0..7] addr
copy p:$100..$130 p:$200
Copy the default ADM program memory values located from hexadecimal 100 through hexadecimal 130 to program memory starting at hexadecimal 200.

copy x:0#100 p:0
Copy one hundred memory locations of the default ADM beginning at x memory location 0 to p memory beginning at location 0.

copy d1 p:0..20 d2..5 p:40
Copy ADM 1 p memory locations 0 through 20 to ADMs 2,3,4 and 5 p memory locations 40 through 60.

---------- Examples ----------

display x:0..7 y:$100..$107
```
X:$0000 000001 000002 000003 000004 000005 000006 000007 000008
Y:$0100 000000 03242B 0647D9 096A90 0C8BD3 0FAB27 12C810 15E214
```

copy y:$100..$107 x:0

display x:0#8 y:0#8
```
X:$0000 000000 03242B 0647D9 096A90 0C8BD3 0FAB27 12C810 000008
Y:$0000 000000 000000 000000 000000 000000 000000 000000 000000
```

NCD02-0714

FORCE

---------- FORCE: Assert Reset or Break on an ADM ----------

FORCE [D0..7] R(hardware reset)|B(Return to monitor)

force r
Force a reset on the current default ADM.

force b
Force a break on the current default ADM.

force d1,4,5 b
Force ADMs 1,4 and 5 to break

NCD02-0715

ASSEMBLE

---------- ASM: Single Line Assembler ----------

ASM [Dx] [(beginning at) addr] [assembler_mnemonic]

asm p:$50
Start interactive assembler at program memory address 50 (hex) of default ADM.

asm x:0 move r0,a1
Assemble single instruction at x memory address 0 of default ADM.

asm
Start assembler at current program counter value of default ADM.

asm d3 p:100
Start assembler at program memory address 100 (decimal) of ADM 3.

---------- Example ----------

```
asm p:$40
ADM#0 P:$0040 57F400 123456 = MOVE #$123456,B
ADM#0 P:$0042 5F0000        = MOVE B,Y:0
ADM#0 P:$0043 20003A        = ASL B
ADM#0 P:$0044 20003A        = ASL B
ADM#0 P:$0045 000000        = NOP
ADM#0 P:$0046 FFFFFF        = DC $FFFFFF
```

(To terminate assembly, press <esc>

NCD02-0716

DISASSEMBLE

```
----------- DISASSEMBLE: Object Code Disassembler -----------
DISASSEMBLE [D0..7] [addr[_block]]

 disassemble
 Disassemble 19 instructions starting at the current program counter value.

 disassemble y:5
 Disassemble 19 instructions starting at address map y:5.

 disassemble p:0..20
 Disassemble program memory address block 0 to 20.

 disassemble x:$50#10
 Disassemble 10 instructions starting at x memory map hex 50.

---------------------------- Example ---------------------------

disassemble p:$40#5
ADM#0 P:$0040 57F400 123456 = MOVE #>$123456,B
ADM#0 P:$0042 5F0000        = MOVE B,Y:<$0
ADM#0 P:$0043 20003A        = ASL B
ADM#0 P:$0044 20003A        = ASL B
ADM#0 P:$0045 000000        = NOP
```

NCD02-0717

TRACE

```
----------- TRACE: Trace Through DSP56000 Program -----------

TRACE [D0..7] [count]

 trace
 Execute one instruction and display the enabled registers and memory
 blocks.

 trace 20
 Execute 20 instructions and display the enabled registers and memory
 blocks after each instruction.

 trace d2,4 5
 Execute 5 instructions on ADMs 2 and 4 from their current program counter
 and display their enabled registers and memory blocks after each instruction.

 ------------------------- Example --------------------------

 display
    b2=    33    b1=444444    b0=555555
 Y:$0000 FFFFFF
 ADM#0 P:$0040 57F400 123456 = MOVE #>$123456,B

 trace 2
 ADM#0 TRACE COUNT=1
    b2=    00    b1=123456    b0=000000
 Y:$0000 FFFFFF
 ADM#0 P:$0042 5F0000        = MOVE B,Y:<$0

 ADM#0 TRACE COUNT=0
    b2=    00    b1=123456    b0=000000
 Y:$0000 123456
 ADM#0 P:$0043 000000        = NOP
```

NCD02-0718

STEP

```
----------- STEP: Step Through DSP56000 Program -----------

STEP [D0..7] [count]

 step
 Step one instruction and display the enabled registers and memory blocks.

 step $50
 Execute hex 50 instructions and display the enabled registers and memory
 blocks at the end of the hex 50th instruction.

 step d1..4 10
 Execute 10 instructions on ADMs 1,2,3 and 4 siumultaneously and
 display the enabled registers and memory blocks of each when they have
 each completed their 10 instructions.
```

NCD02-0719

GO

```
----------- GO: Execute DSP56000 Program -----------

 GO [D0..7] [(from)addr/R(reset)] [(to break #)#bn] [(occurrence):count]

  go
  Start DSP56000 program execution from the current program counter value
  of the default ADM.

  go $100
  Load the default ADM program counter with hexidecimal 100 and start
  DSP56000 program execution.

  go p:$100
  Load the default ADM program counter with the value in program memory
  address hexidecimal 100 and start DSP56000 program execution.

  go r
  Start DSP56000 program execution of the default ADM by asserting
  its reset line.

  go d1..3
  Start DSP56000 program execution of ADMs 1,2 and 3 at their current
  program counter addresses.

  go d1 100 #5 :3
  Start DSP56000 program execution of ADM 1 from address P:100.  Halt on
  the third occurrence of breakpoint number 5.
```

NCD02-0720

BREAK POINT

```
BREAK [D0..7] [#bn] [addr|OFF] T[expr] [H(halt)/In(inc. CNTn)/N(note)/S(show)]
```

Following is a list of valid operators useable in an expression:

<	less than	&&	logical 'and'
<=	less than or equal to	||	logical 'or'
==	equal to	!	logical 'negate'
>=	greater than or equal to	&	bitwise 'and'
>	greater than	|	bitwise 'or'
!=	not equal to	~	bitwise one's complement
+	addition	^	bitwise 'exclusive or'
-	subtraction	<<	shift left
/	division	>>	shift right

---------- Breakpoint Actions ----------

H	Halt execution. This is the default.
In	Increment counter variable CNTn (n=1/2/3/4).
N	Note - display the breakpoint expression and continue.
S	Show the enabled register/memory set and continue.

---------- Examples ----------

```
break
Display all currently enabled breakpoints.

break off
Disable all currently enabled breakpoints.

break #1 off
Remove breakpoint number 1.

break p:500
The default ADM will halt program execution when the program counter reaches
the instruction at address 500.

break p:300 t(lc<10)&&(r0>100)
The default ADM will test the conditions of the expression when the
program counter reaches the instruction at address 300.  If the condition
is true the ADM will halt program execution and display its registers.
```

NCD02-0721

```
                                                        ;All Registers off
display off                                             ;Show only Acc. B
display on b2 b1 b0                                     ;Initialize Acc. B
change b2 $22 b1 $111111 b0 $abcdef                     ; and show it
display
   b2=     22     b1=111111     b0=ABCDEF
ADM#0 P:$0041 FFFFFF            = DC $FFFFFF

                                                        ;Assemble Program
asm p:$40                                               ; in Program memory
asm d0 P:$0040 move #$123456,b                          ; at hex address 40
asm d0 P:$0042 asl b                                    ;
asm d0 P:$0043 asl b                                    ;
asm d0 P:$0044 asl b                                    ;
asm d0 P:$0045 asl b                                    ;
asm d0 P:$0046 nop                                      ;Type <esc> to stop
asm d0 P:$0047 nop

                                                        ;Start of program
change pc $40                                           ;Disassemble program
disassemble p:$40#8
ADM#0 P:$0040 57F400 123456 = MOVE #>$123456,B
ADM#0 P:$0042 20003A        = ASL B
ADM#0 P:$0043 20003A        = ASL B
ADM#0 P:$0044 20003A        = ASL B
ADM#0 P:$0045 20003A        = ASL B
ADM#0 P:$0046 000000        = NOP
ADM#0 P:$0047 000000        = NOP
ADM#0 P:$0048 FFFFFF        = DC $FFFFFF

                                                        ;Start value of B
display
   b2=     22     b1=111111     b0=ABCDEF
ADM#0 P:$0040 57F400 123456 = MOVE #>$123456,B

                                                        ;Trace instruction 1
trace
ADM#0 TRACE COUNT=0
   b2=     00     b1=123456     b0=000000
ADM#0 P:$0042 20003A        = ASL B

                                                        ;Trace instruction 2
trace
ADM#0 TRACE COUNT=0
   b2=     00     b1=2468AC     b0=000000
ADM#0 P:$0043 20003A        = ASL B

                                                        ;Set breakpoint
break p:$46                                             ;Go to breakpoint
go
ADM#0 break #1
   b2=     01     b1=234560     b0=000000
ADM#0 P:$0046 000000        = NOP
```

NCD02-07

DSP56000/1
EXCEPTIONS

Objectives:

- Describe Interrupt Sources.
- Describe Interrupt types.
- Describe Interrupt Masks and Priorities.

NCD02-0600-1

Table of Contents

DSP56000 Processing States 7-5
Vector Table of Exception Sources 7-6
RESET 7-7
Stack Error 7-8
Characteristics of Stack Error 7-9
WAIT Exception 7-9
Exiting Wait State 7-10
STOP Exception 7-10
Exiting STOP State via RESET 7-11
Exiting STOP State via IRQA 7-11
Exiting from STOP via IRQA Timing 7-12
WAIT vs STOP 7-12
Program Debugging Using Trace Mode 7-13
Entering Trace Mode 7-13
Tracing 7-14
Characteristics of Trace 7-14
Fast Interrupt 7-15
Characteristics of Fast Interrupts 7-15
Long Interrupt 7-16
Characteristics of Long Interrupts 7-17
RTI - Return From Interrupt 7-17
Interrupt Instruction Timing 7-18
Level Sensitive Fast Interrupts 7-18
Negative Edge Triggered Interrupt 7-19
Interrupt Examples 7-19
Long Software Interrupt 7-20
Fast Software Interrupt 7-20
Interrupt Priority Level Mask Bits 7-21
Interrupt Priority System 7-22
Interrupt Priority Register 7-23
Exception Priorities Within an IPL 7-23

NCD02-0699

DSP56000 PROCESSING STATES

- Normal
  - Instruction Execution

- Reset – Level Sensitive
  - Assertion of $\overline{\text{RESET}}$ Pin.

- Wait
  - Halt the processor core.
  - Clock and peripherals are active.

- Stop
  - Halt everything!

- Exception – Three levels of Software programmable priority
  - External Interrupts
    - IRQA – Level Sensitive or Edge Sensitive
    - IRQB – Level Sensitive or Edge Sensitive
  - Internal
    - Software Interrupt
    - On Chip Peripheral
    - Error Condition
    - Tracing

NCD02-0601

VECTOR TABLE OF EXCEPTION SOURCES

Program Memory Space

Exception Starting Address	Exception Source	
$0000	Hardware $\overline{\text{RESET}}$	Two Words Per Vector
$0002	Stack Error	
$0004	Trace	External Interrupts
$0006	SWI (Software Interrupt)	
$0008	$\overline{\text{IRQA}}$ External Hardware Interrupt	
$000A	$\overline{\text{IRQB}}$ External Hardware Interrupt	Synchronous Serial Interface
$000C	SSI Receive Data	
$000E	SSI Receive Data with Exception Status	
$0010	SSI Transmit Data	Internal Interrupts
$0012	SSI Transmit Data with Exception Status	
$0014	SCI Receive Data	Synchronous Communications Interface
$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	

- The Vector table contains Instructions, NOT Addresses!

- Each Vector location can contain;
  - Two, One Word Instructions
  - One, Two Word Instruction

NCD02-0602

RESET

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

1. Reset on chip peripherals.

RESET → Port B & C, SCI, SSI, HOST

2. Set Modifier Registers to $FFFF.

M0 – M7 $FFFF

3. Clear Interrupt Priority Register.

X:$FFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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.

X:$FFFE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 $\overline{\text{RESET}}$

ME					DE	MODB	MODA
0	0	0	0	0	0	X	X

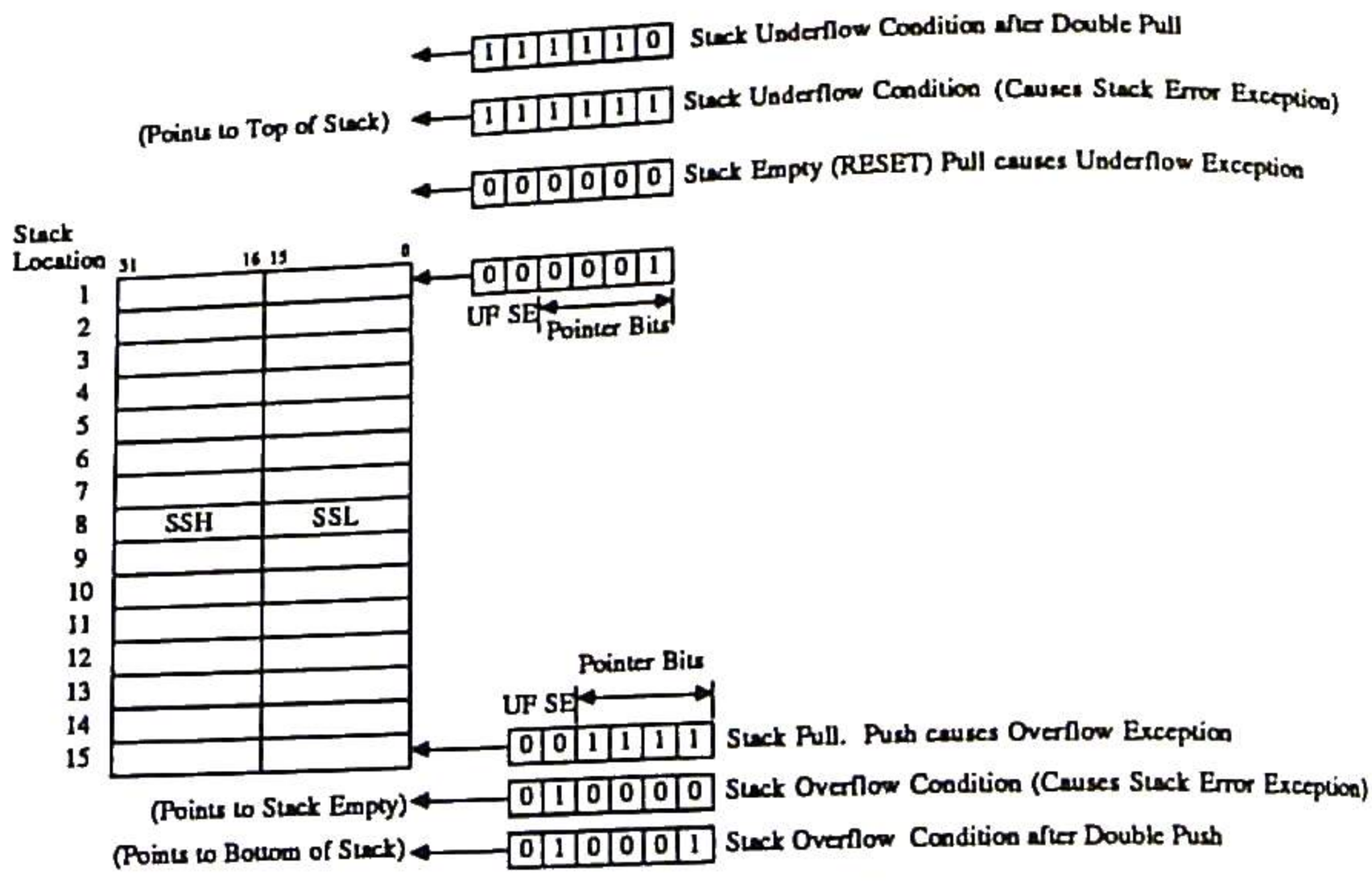
9. Load Operating Mode Register from Mode Pins.

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

NCD02-0603

JMP START

STACK ERROR

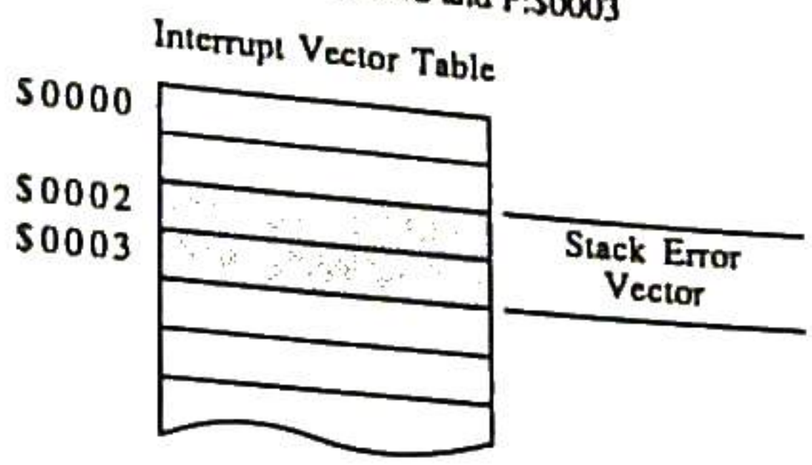


- A transition of the Stack Error bit from 0 to 1 causes a Stack Error Exception.
- DO Loops are sources of Double pulls and double pushes.

NCD01-0604

CHARACTERISTICS OF STACK ERROR

- The Stack Error Vector occupies address P:$0002 and P:$0003



- Occurs on Stack Overflow or Underflow.
- Stack Error Bit (SE) changing from 0 to 1 creates a non maskable level 3, (Highest Priority), Stack Error Exception.
- A Stack Error Exception does NOT occur when:
  - The Stack is empty (SP= 0) and an instruction reads the stack without postdecrementing the SP
  - The Stack is empty and overflowed (SP=$10) and an instruction writes the stack without preincrementing the SP.
  - Both conditions create an unsuccessful read or write because it is a non existing stack location.

Examples: a) DO SSL, xxxx

b) REP SSL

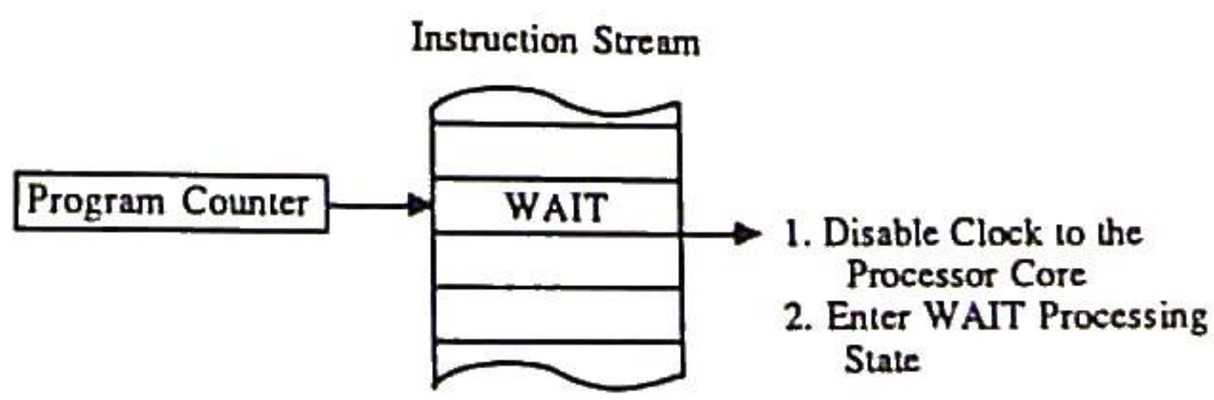
c) MOVEC SSL, xxxx

d) MOVEP xxxx, SSL

NCD02-0605

WAIT EXCEPTION

Entering Wait State

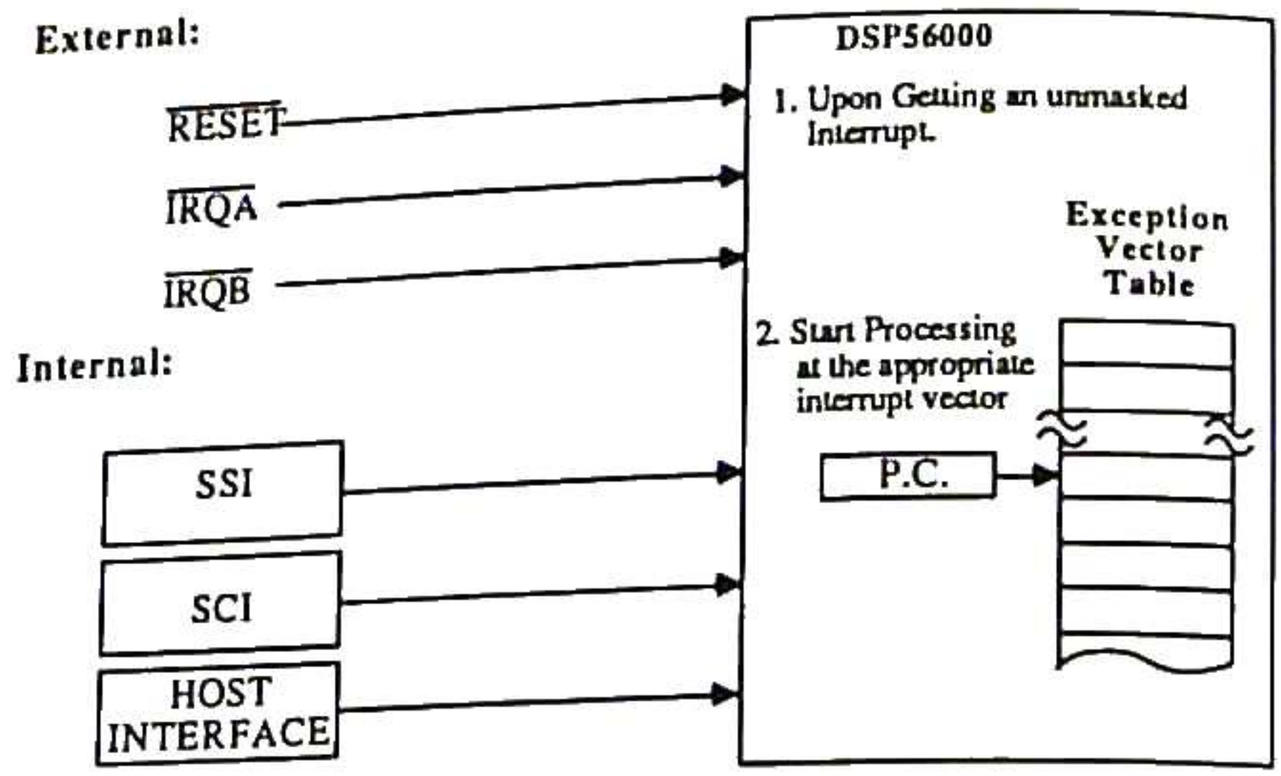


- On-chip Clock oscillator and peripherals remain active
- Low Power Stand by mode
- Machine state is maintained, all on chip RAM, Processor and Peripheral Registers
- Exit WAIT processing state via any interrupt.

NCD02-0606-1

EXITING WAIT STATE

- RESET – Begin Reset Processing
- Any Unmasked Internal or External Hardware Interrupt.

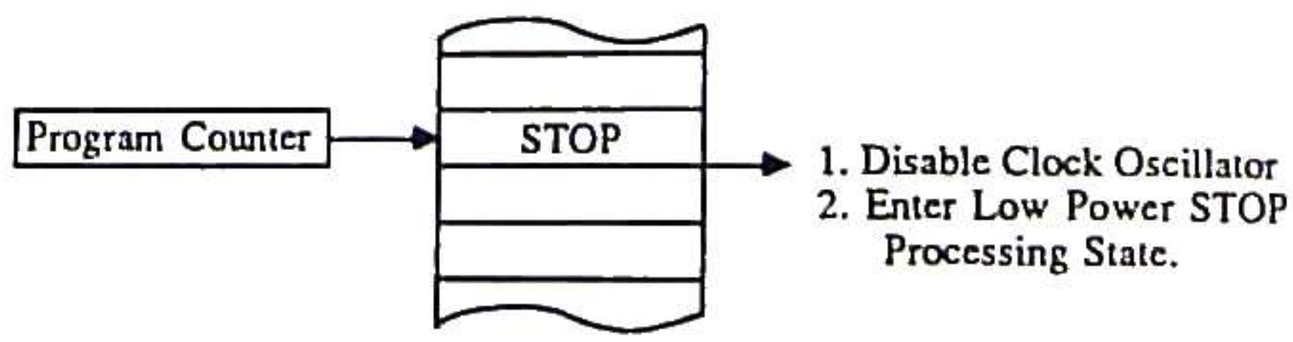


NCD02-0607

STOP EXCEPTION

Entering Stop State

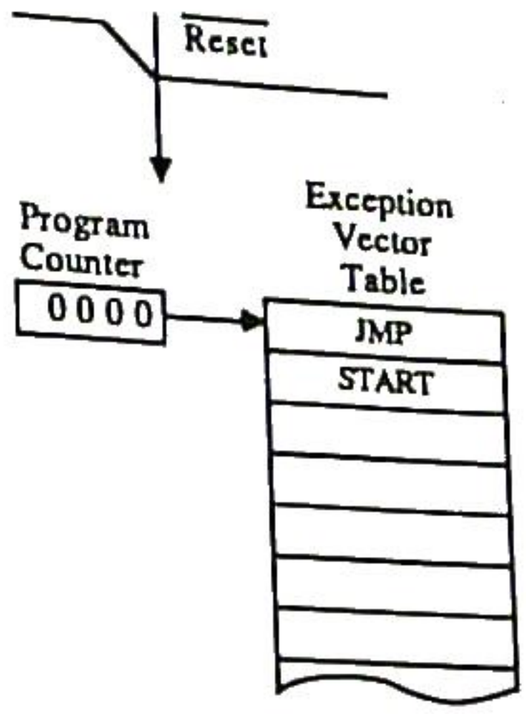
Instruction Stream



- Entire DSP56000 is halted: Processor, Oscillator and Peripherals
- Lowest power saving stand by mode
- Machine State is maintained, all on chip RAM, Processor and Peripheral registers.
- Exit STOP processing state via two methods

NCD02-0608-1

EXITING STOP STATE VIA RESET



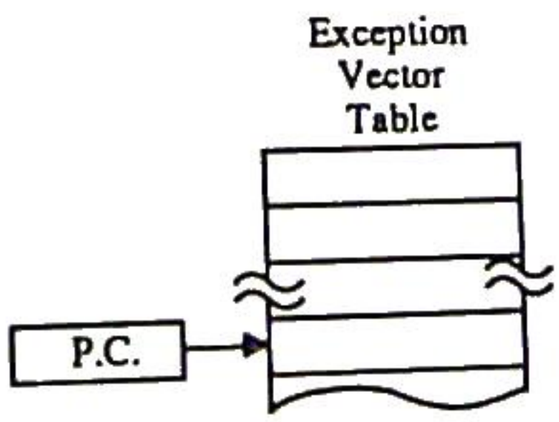
1. Enables Clock and Peripherals.
2. Starts RESET Exception Processing.

NCD02-0609

EXITING STOP STATE VIA IRQA

Case I: Interrupt Pending

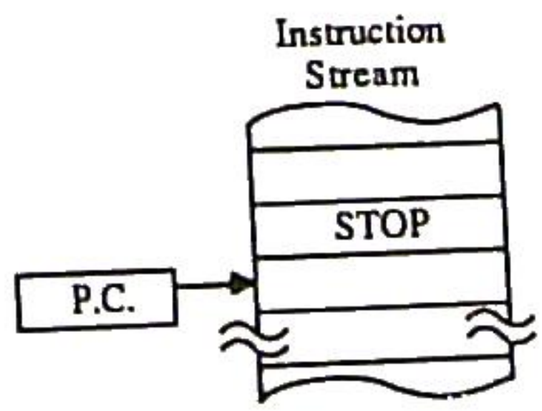
Service the Highest Pending Interrupt.



NOTE: IRQA does not need to be the pending interrupt. Any pending interrupt will cause exception processing for that interrupt.

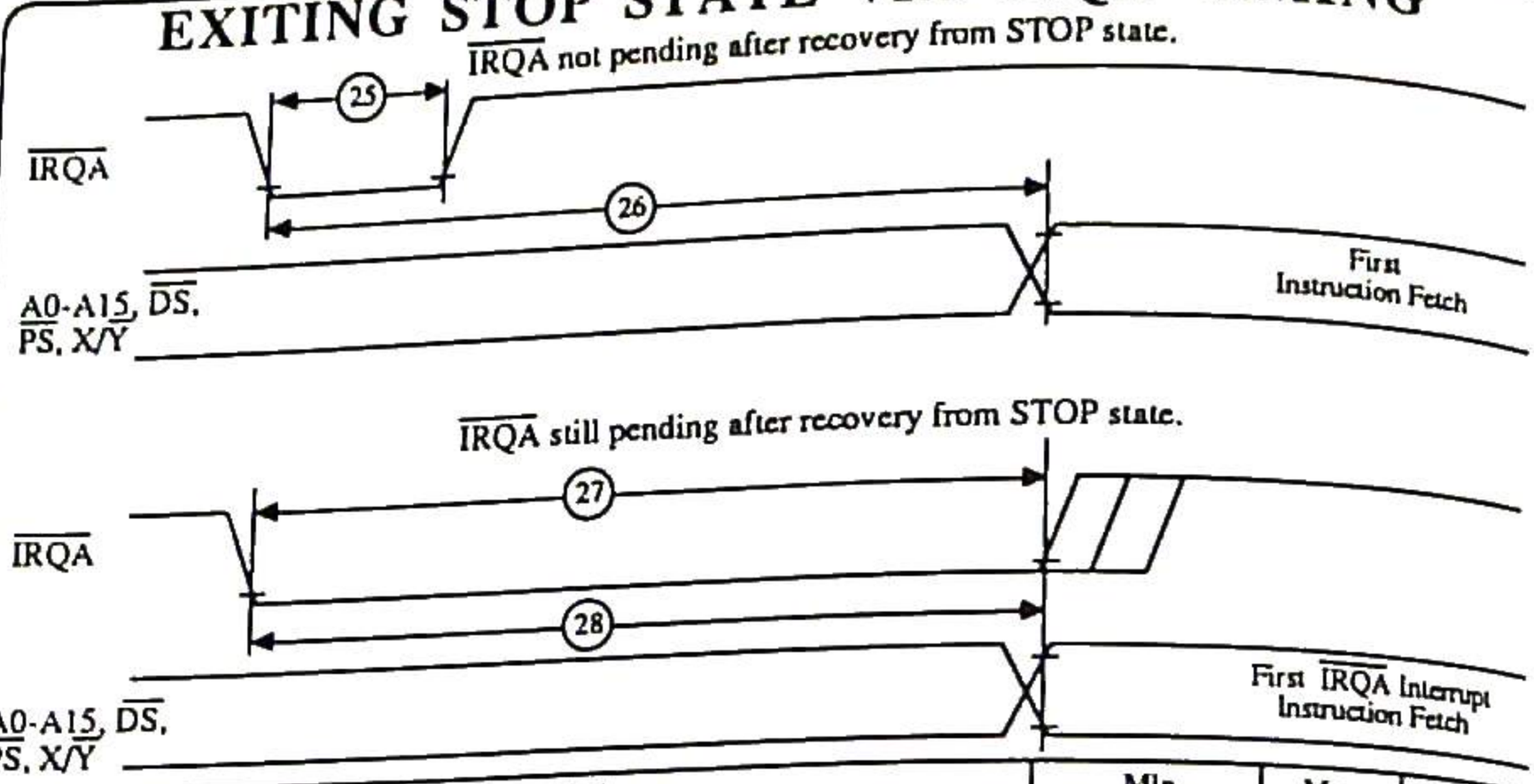
Case II: No Interrupt Pending

Begin execution with the instruction that immediately follows the STOP instruction.



NCD02-0611

EXITING STOP STATE VIA IRQA TIMING



Num	Characteristic		Min	Max	Unit
25	Duration for IRQA Assertion to Recover from STOP State		25		ns
26	Delay from IRQA Assertion to Fetch of First Interrupt Instruction	Stable External Clock	128000T + 17T	–	ns
		Internal Crystal Oscillator Clock	150000T	–	ns
27	Duration for Level Sensitive IRQA Assertion to Fetch of First Interrupt Instruction	External Clock	128000T + 17T	–	ns
		Internal Clock	150000T	–	ns
28	Delay from Level Sensitive IRQA Assertion to Fetch of First Interrupt Instruction	External Clock	128000T + 17T	–	ns
		Internal Clock	150000T	–	ns

NCD02-0610-2

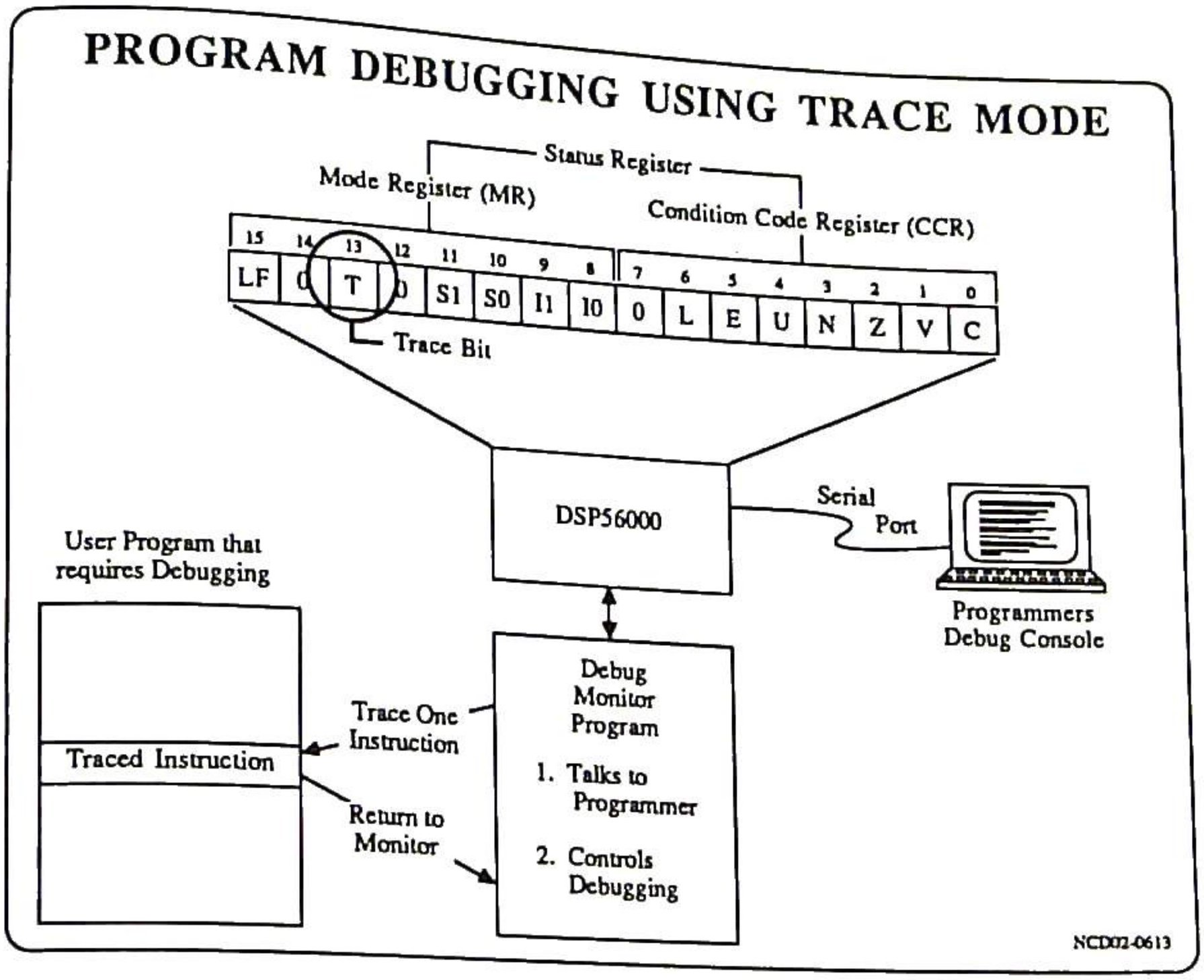
WAIT VS. STOP

- Useful for low power or battery powered applications
- Trade off is power dissipation vs. recovery time

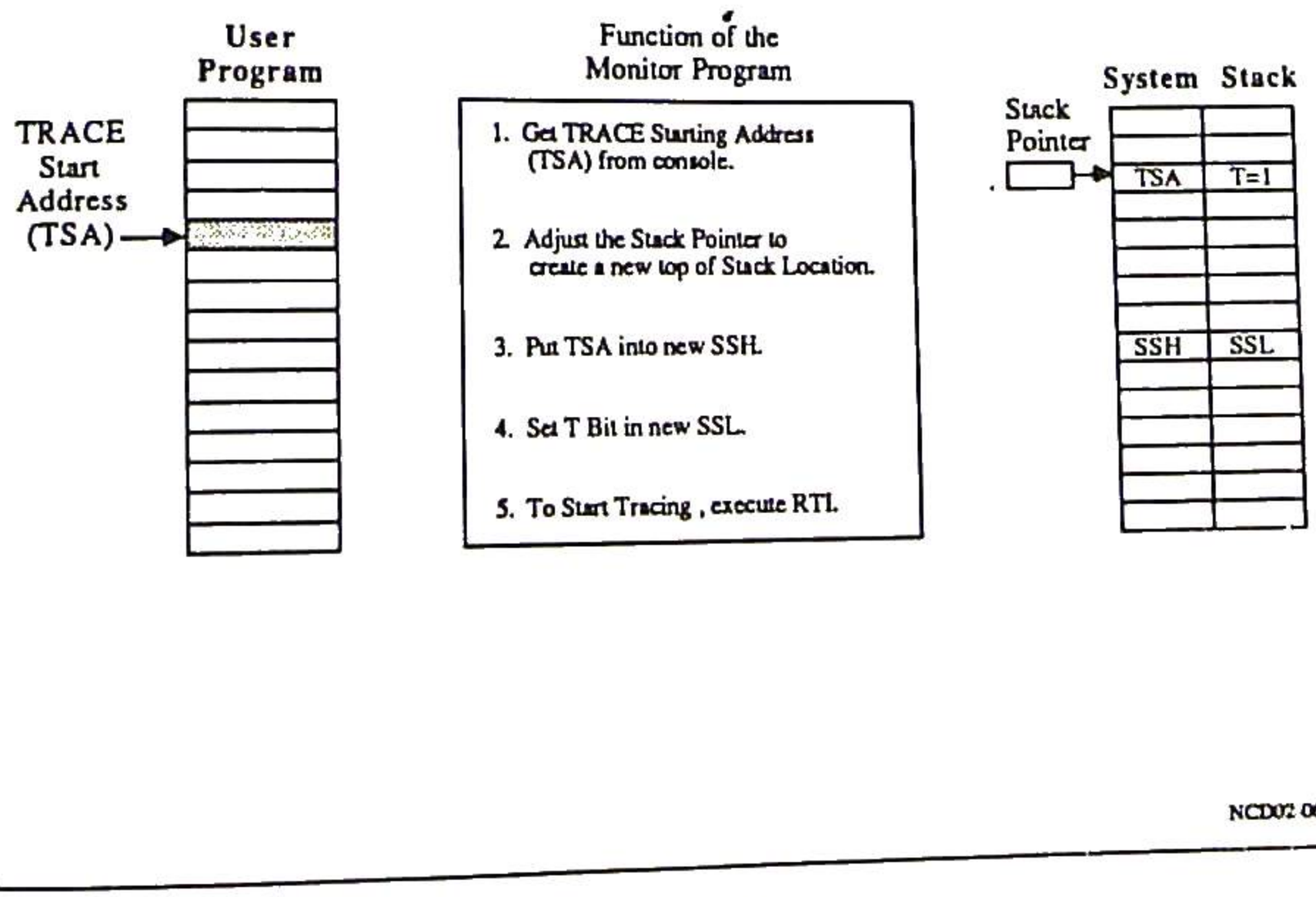
Processing State	Current Consumption (TYP) @ 20 MHz	Recovery From Exception
NORMAL	90 mA	16T – 18T
WAIT	30 mA	20T
STOP	100 μA	128,000T–150,000T

NCD02-0612

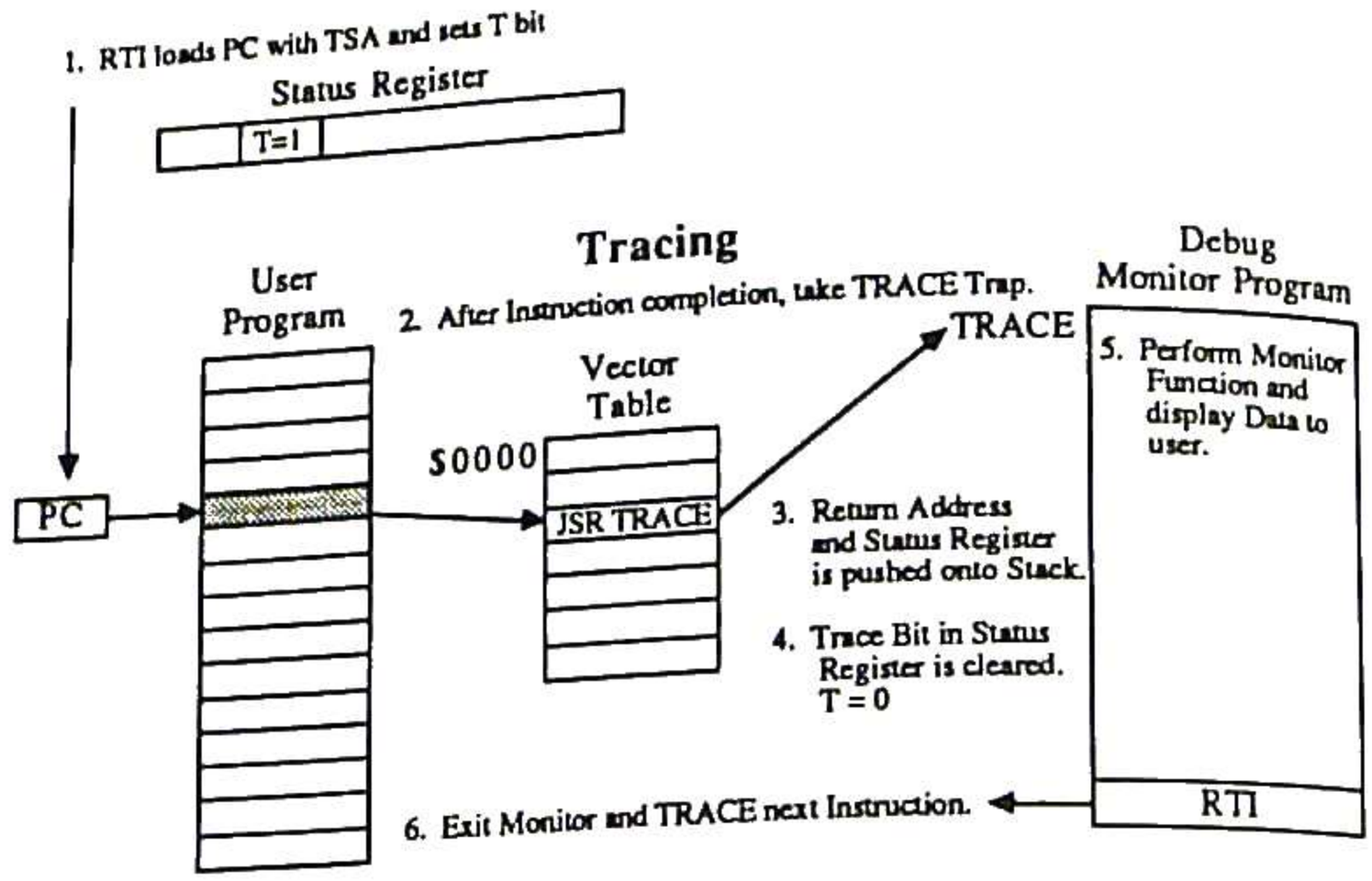
PROGRAM DEBUGGING USING TRACE MODE



ENTERING TRACE MODE



TRACING



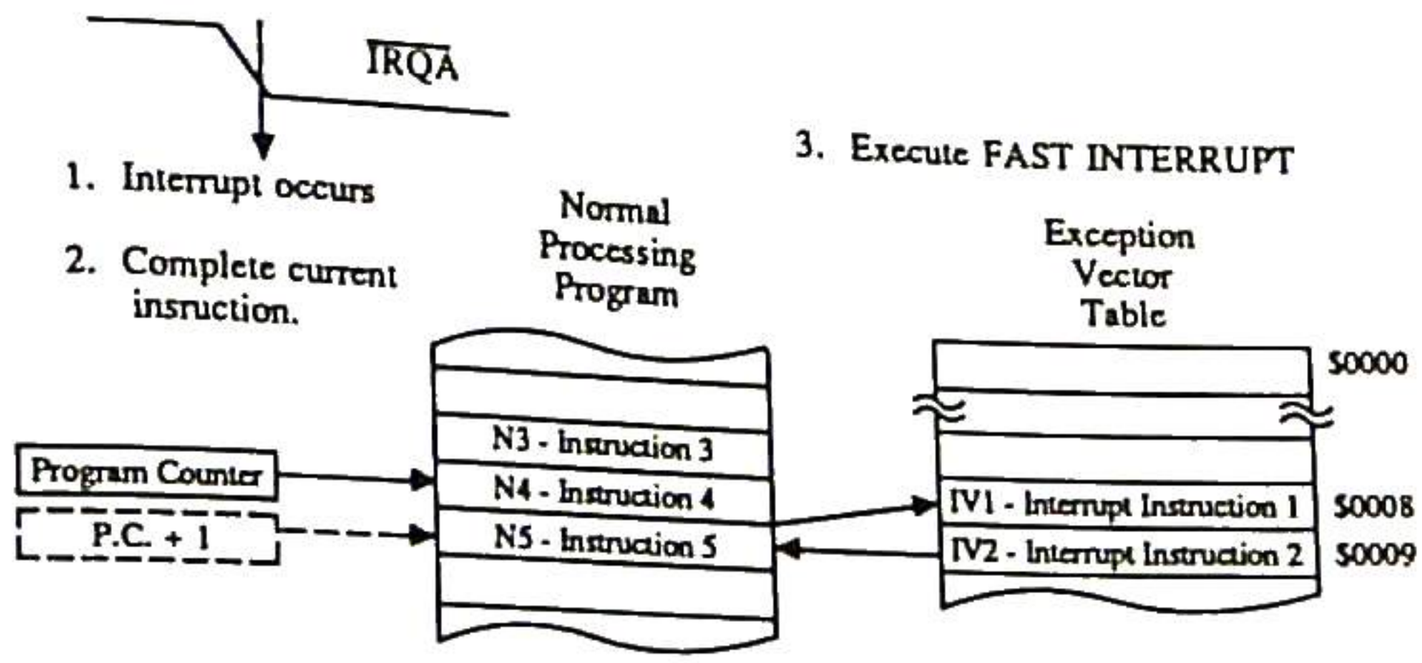
NCD02-0615

CHARACTERISTICS OF TRACE

- Useful for debugging.
- Traps out to Trace Exception Vector after execution of each instruction.
- Trace is a NON Maskable level 3 exception.
- Masks out all external and Peripheral Interrupts except for TRACE, STACK ERROR, and RESET.
- Fast Interrupt routines cannot be traced.

NCD02-0631

FAST INTERRUPT



- A Zero Overhead Interrupt

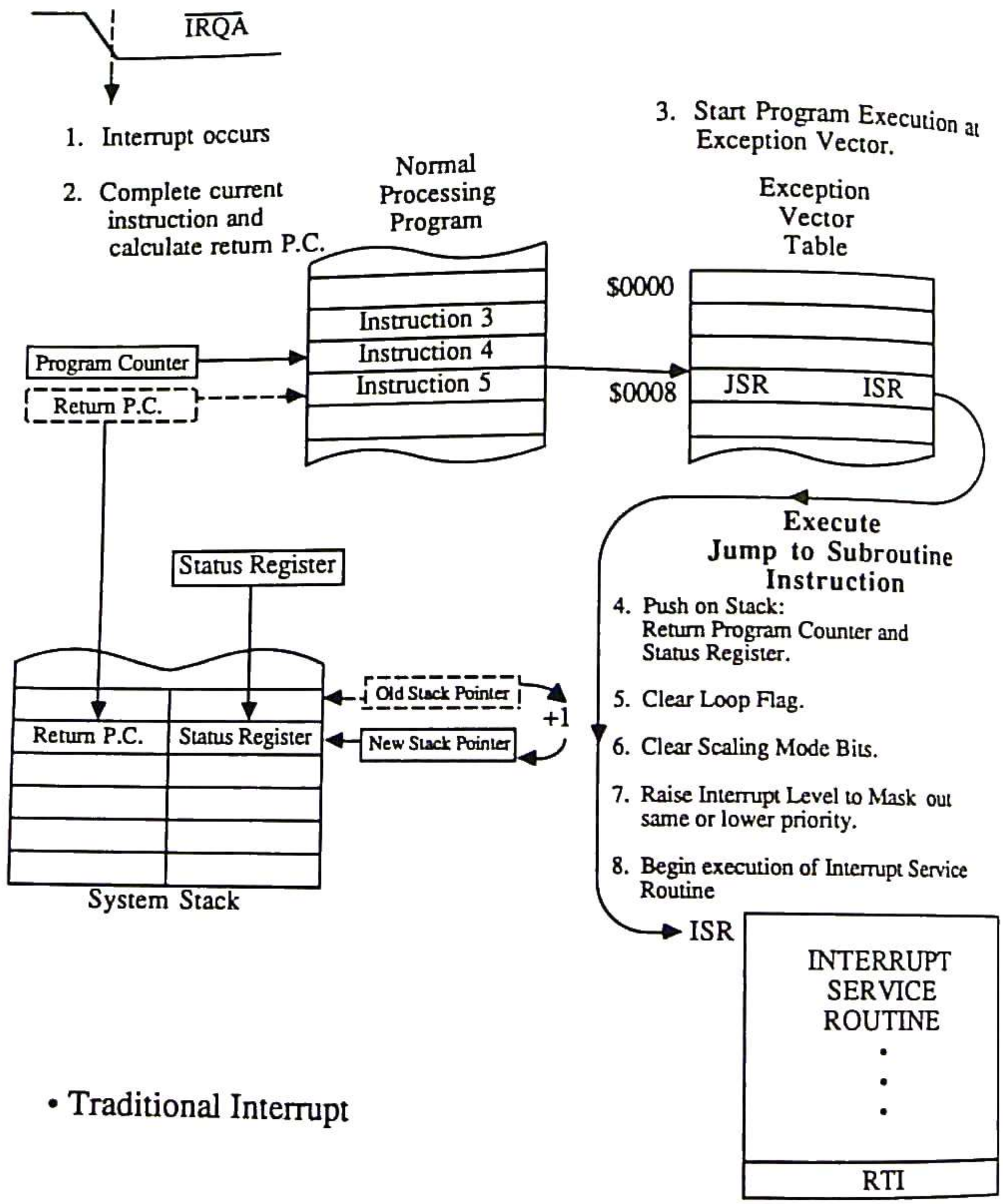
NCD02-0616

CHARACTERISTICS OF FAST INTERRUPTS

- Fast interrupts consist of a single two word instruction, or two, one word instructions.
- Any exception source can be a Fast interrupt except RESET.
- The Program Counter is never updated to the interrupt routine
- 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.
- They cannot be traced.
- The scaling mode in effect before the fast interrupt, apply to the fast interrupt.
- A Fast interrupt MUST NOT use a "Change of Flow" instruction.
- A jump to subroutine instruction within the fast interrupt routine constitutes a Long Interrupt Routine.

NCD02-0617

LONG INTERRUPT



- Traditional Interrupt

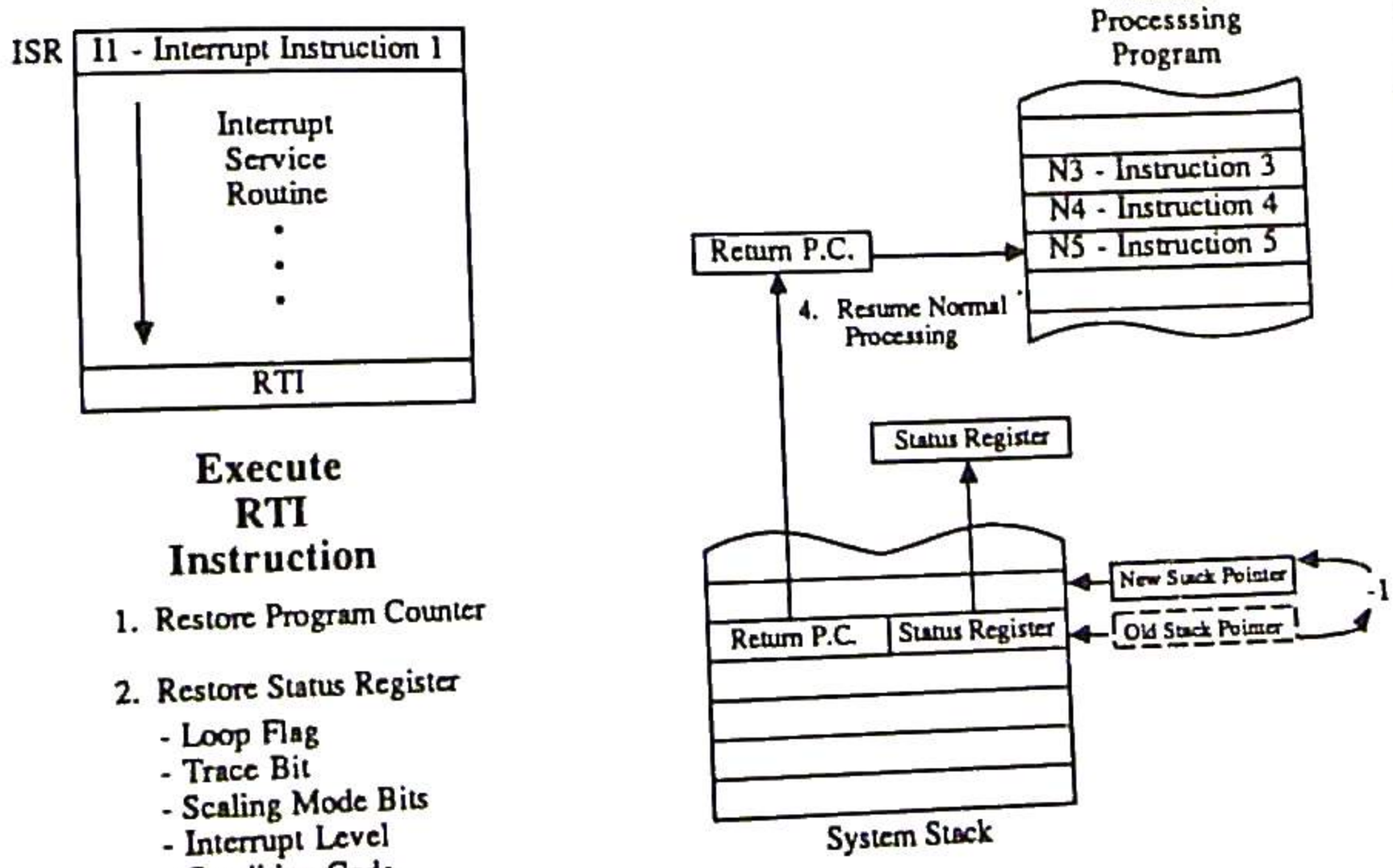
NCD02-0615

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.

NCD02-0632

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

NCD02-0620

INTERRUPT INSTRUCTION TIMING

Fast Interrupt

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)

Long Interrupt

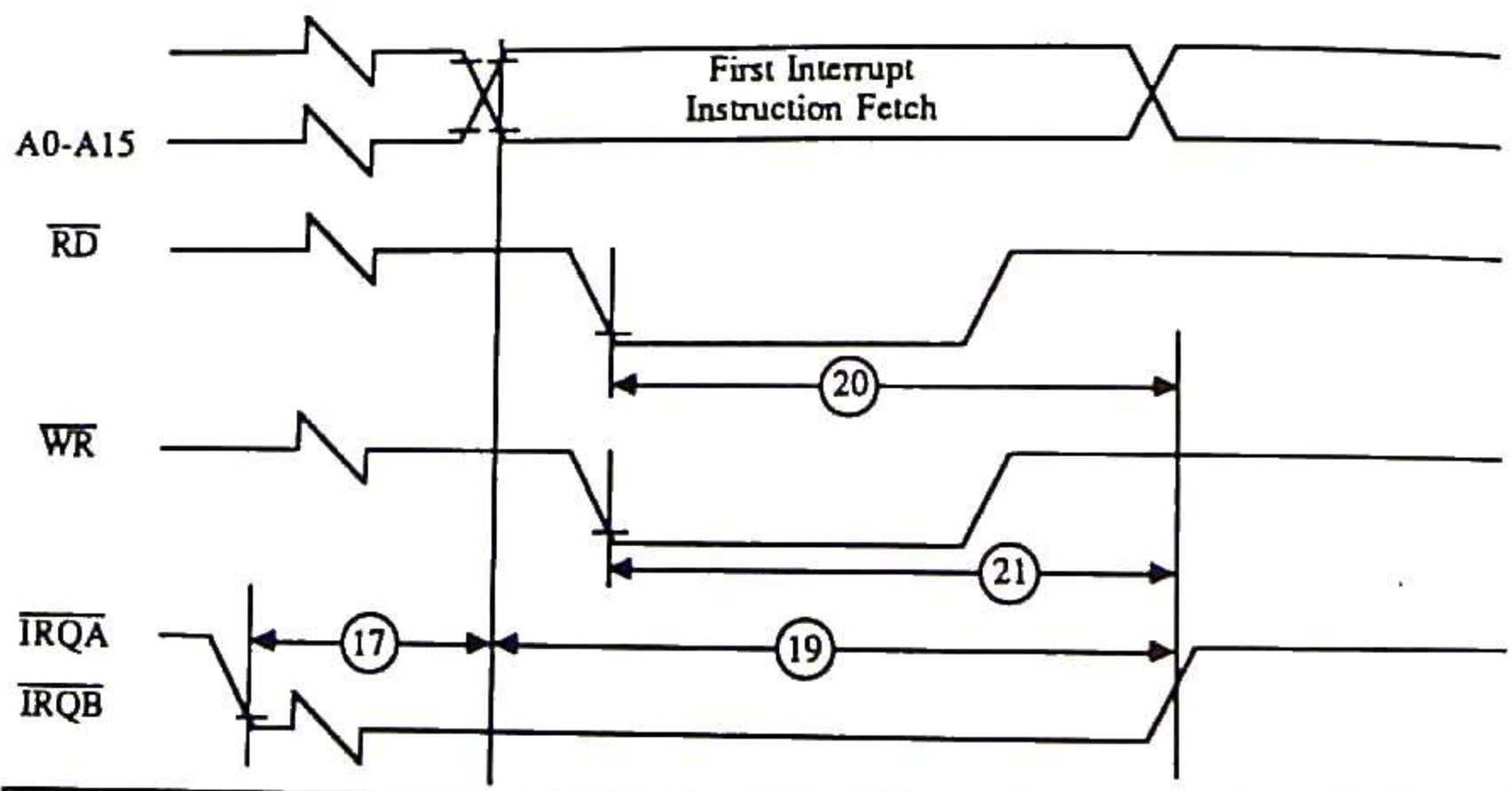
Minimum Interrupt Cycle Time

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

NCD02-0621

LEVEL SENSITIVE FAST INTERRUPTS

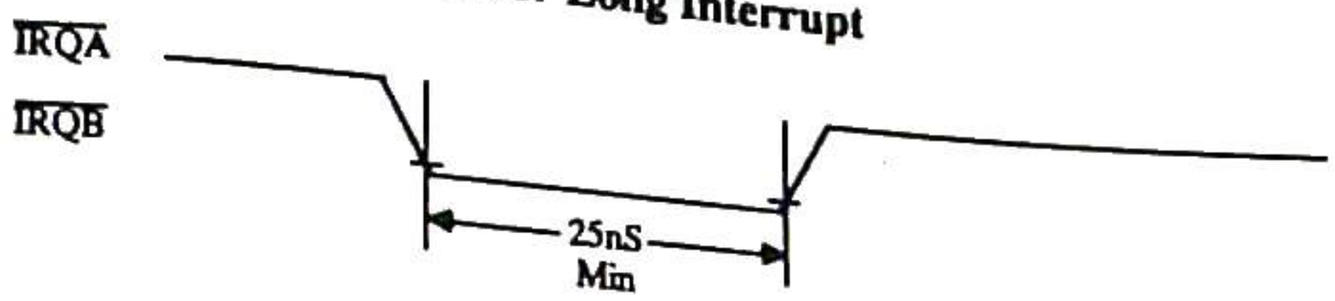


Num	Characteristic		Min	Max	Unit
17	Delay from $\overline{IRQA}$, $\overline{IRQB}$ Assertion to External Memory Access Address Out Valid Caused by First Interrupt Instruction Fetch		19T	–	ns
19	Delay from Address Output Valid Caused by First Interrupt Instruction Fetch to Interrupt Request Deassertion for Level Sensitive Fast Interrupts		–	5T – 2T•WS – 44	ns
20	Delay from $\overline{RD}$ Assertion to Interrupt Request Deassertion for Level Sensitive Fast Interrupts		–	4T – 2T•WS – 40	ns
21	Delay from $\overline{WR}$ Assertion to Interrupt Request Deassertion for Level Sensitive Fast Interrupts	WS = 0 WS = 0	– –	4T – 40 3T – 2T•WS – 40	ns ns

NCD02-0622

NEGATIVE EDGE TRIGGERED INTERRUPT

Fast or Long Interrupt



- Latched as pending on the first High to Low transition.
- Cleared automatically when interrupt is serviced or by hardware RESET.

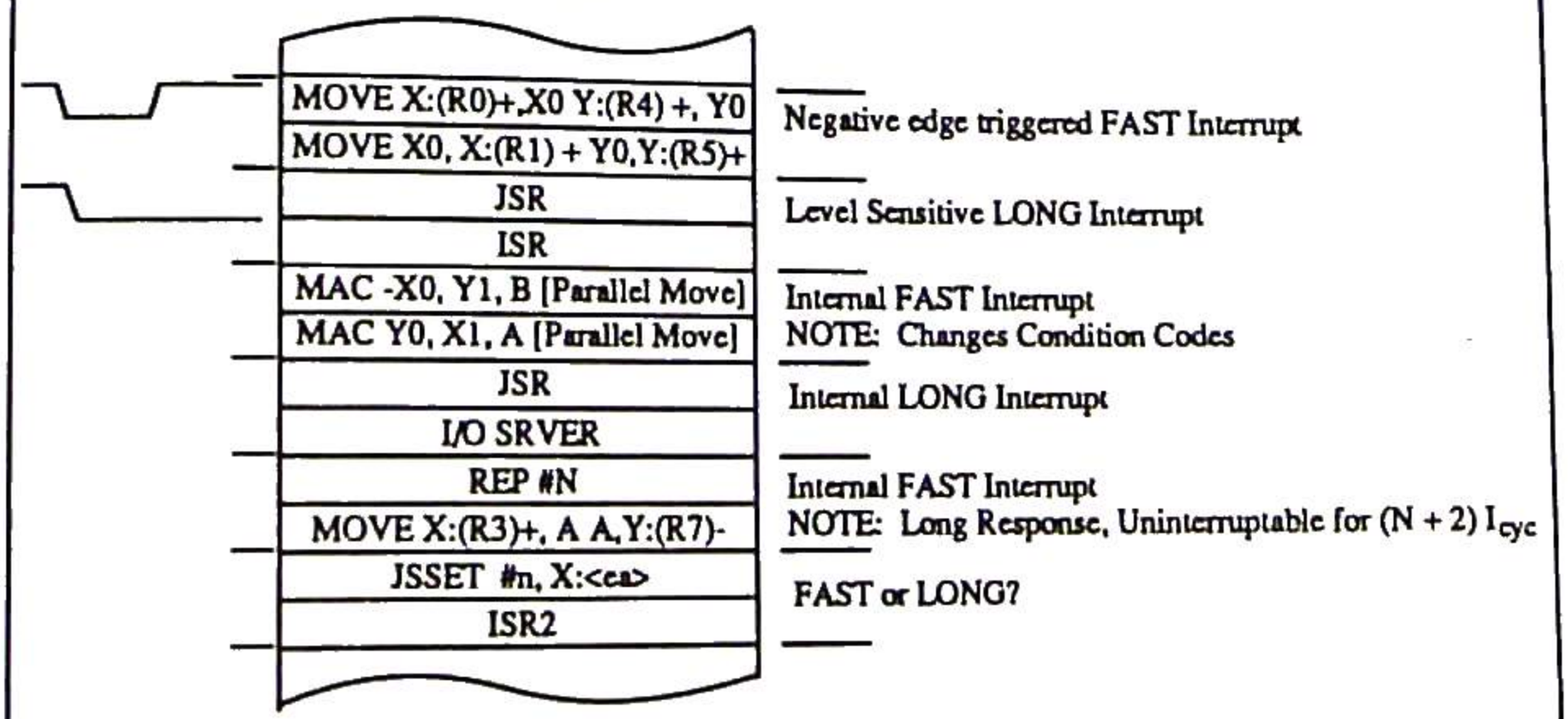
Recommendations for easy Interrupt Implementation

- Use Edge triggered for Fast Interrupts
  - Faster Response
  - Timing is easier to meet
- Use Level Sensitive for Long Interrupts
  - Multiple devices can be Wire-ORed together

NCD02-0623

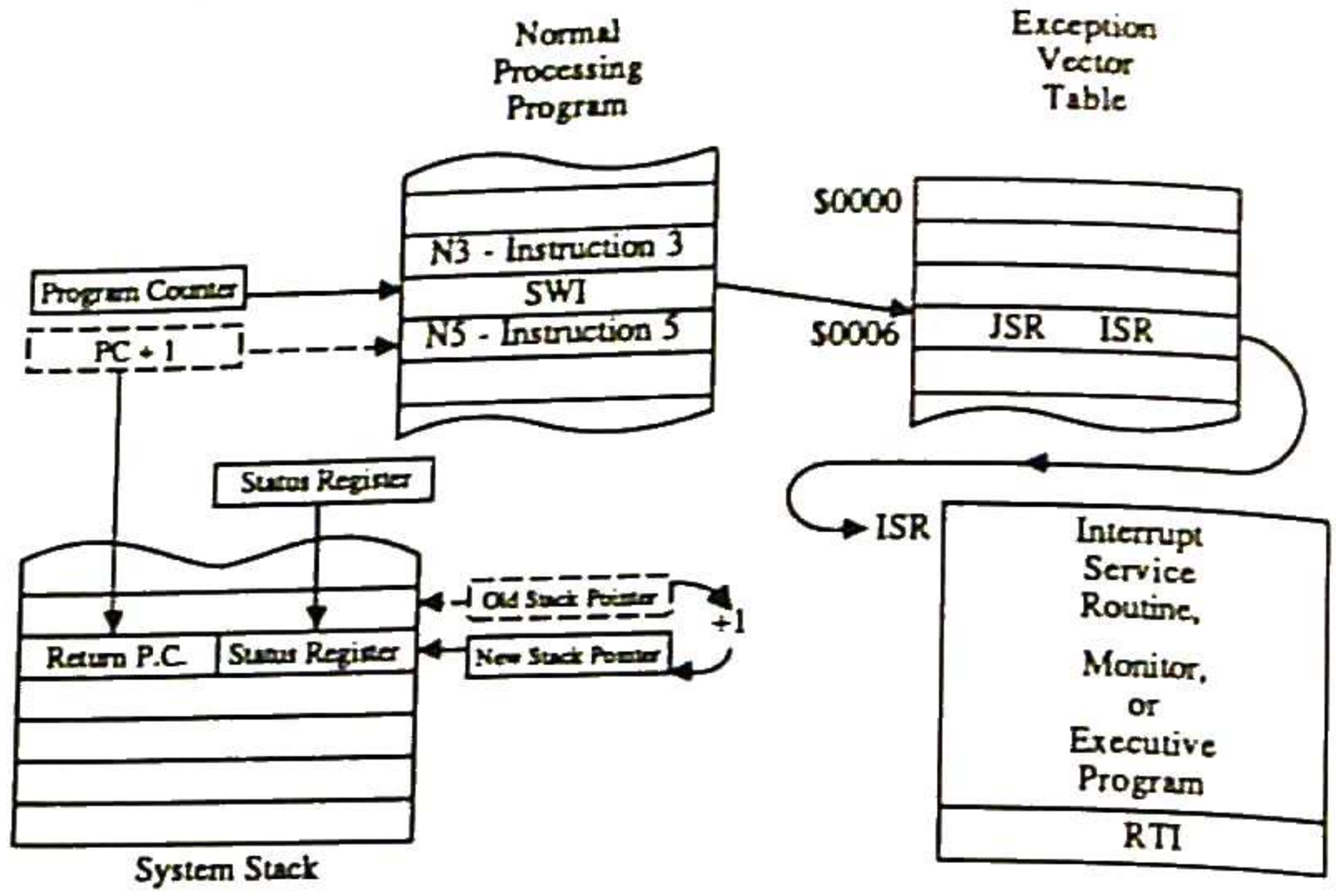
INTERRUPT EXAMPLES

Vector Table



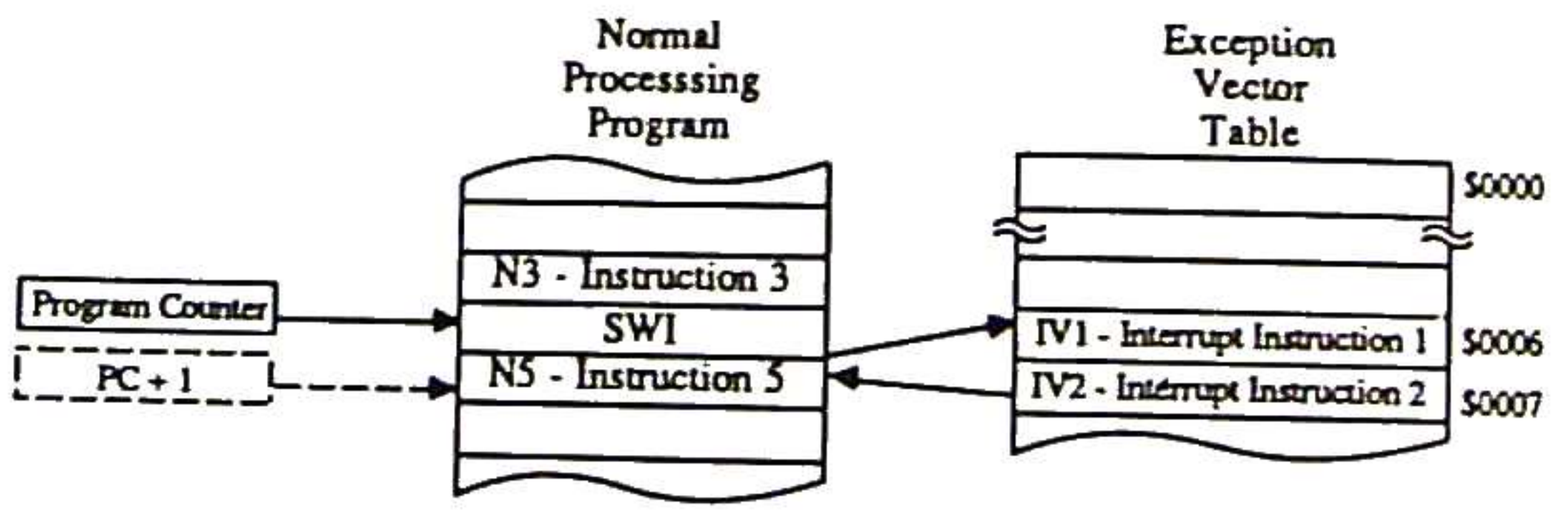
NCD02-0624-1

LONG SOFTWARE INTERRUPT



NCD02-0625

FAST SOFTWARE INTERRUPT

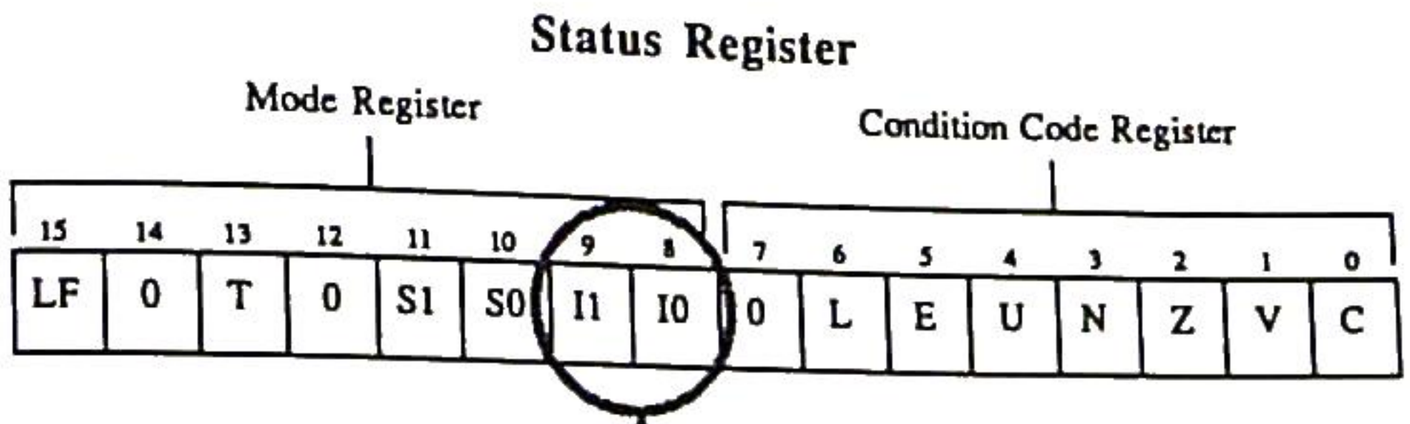


Characteristics of Software Interrupts

- 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 or Debugging Interrupt Routines.

NCD02-0626

INTERRUPT PRIORITY LEVEL MASK BITS

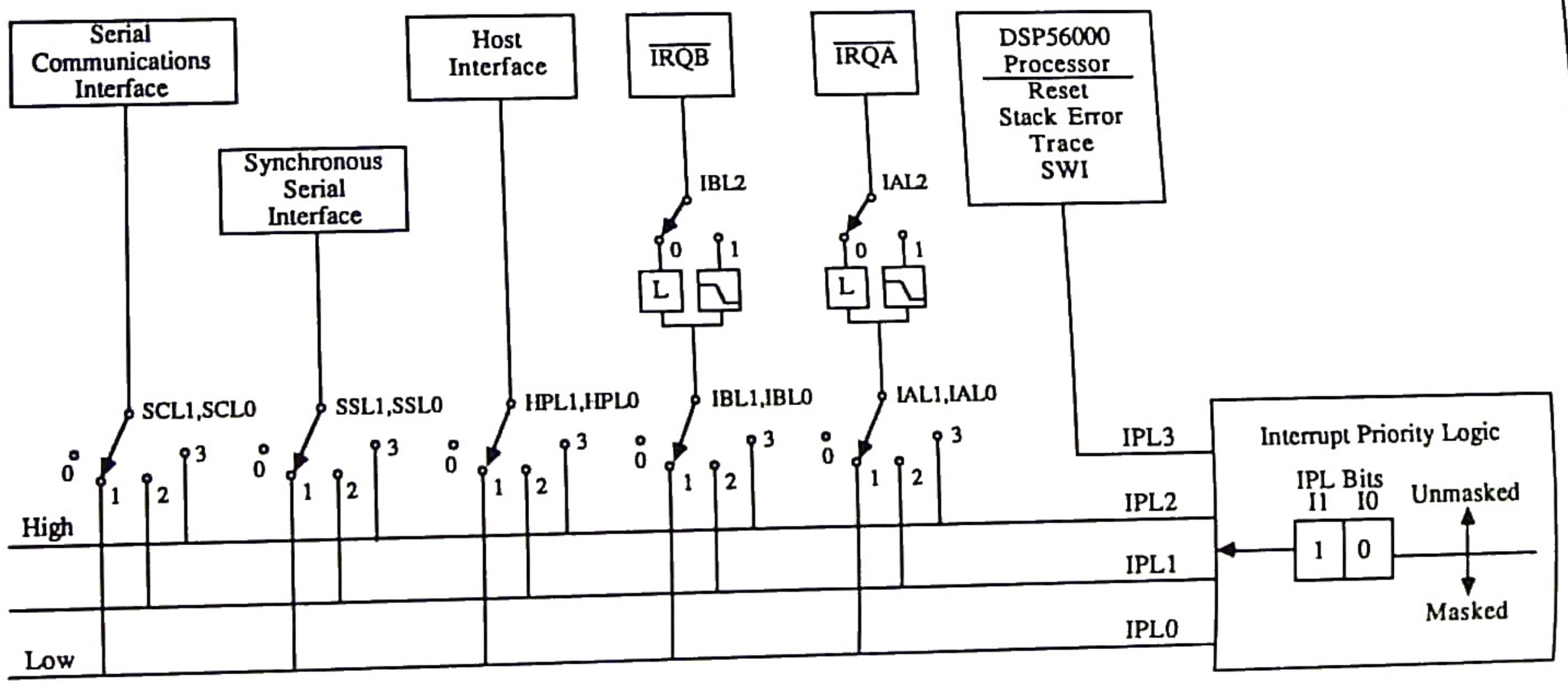


Priority	I1	I0	Exceptions Permitted	Exceptions Masked
Low	0	0	IPL 0, 1, 2, 3	None
↓	0	1	IPL 1, 2, 3	IPL 0
	1	0	IPL 2, 3	IPL 0, 1
High	1	1	IPL 3	IPL 0, 1, 2

- Four Levels of Prioritized Interrupts
- Set to Highest Level on Hardware or Software RESET

NCD02-0627

INTERRUPT PRIORITY SYSTEM



INTERRUPT PRIORITY REGISTER

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:SFFFF	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

- On Hardware or Software RESET, the Interrupt Priority Register is Cleared i.e. All Interrupts are Masked.

External Interrupts are Level Sensitive.

NCD02-0629

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	

NCD02-0630

DSP56000/1
ON CHIP PERIPHERALS

Objectives:

- Describe how to perform Parallel I/O.
- Describe how to transmit and receive a character using the Serial Communications Interface.
- Describe the Host Interface and Host Transfers.
- Describe a Synchronous Serial Interface application.
- Describe the function of the SSI registers plus the control and status bits.

NCD02-1000-1

Table of Contents

On Chip Peripheral Features
Input/Output Block Diagram
On Chip Peripheral Memory Map 8-5
Ports B and C Control and Direction Bits 8-6
Parallel Port B Registers 8-7
Parallel Port B Pinout 8-8
Parallel Port C Registers 8-9
Parallel Port C Pinout 8-9
Serial Communications Interface Features 8-10
SCI Control and Status Register Set 8-10
SCI Transmit and Receive Register Set 8-11
SCI RESET 8-12
Serial Formats 8-13
SCI Synchronous Master Mode 8-13
SCI Synchronous Master Timing 8-14
SCI Synchronous Slave Mode 8-15
SCI Synchronous Slave Timing 8-16
Characteristics of SCI Synchronous Mode 8-17
SCI Receiver Initialization 8-18
SCI Receive a Character 8-18
SCI Receive a Character with Exception 8-19
SCI Transmitter Initialization 8-19
SCI Transmit a Character 8-20
Transmitting Marks and Spaces 8-21
SCI Clock and Timer 8-21
Common BAUD Rates 8-22
SCI Asynchronous Mode Timing 8-23
11 Bit Multidrop Mode 8-24
Transmitting Data and Address Characters 8-24
WIRED-OR Mode 8-25
Idle Line Wake Up 8-25
Address Mode Wake Up 8-26
Host Interface Features 8-27
Host Interface Register Map 8-27
Host Transfer Timing 8-28
MC68HC11 to DSP56000 Host Interface 8-29
MC68000 to DSP56000 Host Interface 8-29
Host Interface Block Diagram 8-30
Host Interface Programming Model 8-31
DSP56000 Host Interface Programming Model 8-32
Host Command 8-33
Host Command Programming 8-34
Host Flag 8-35
Bootstrap Using Host Flags 8-36
Transmit/Receive Byte Registers 8-37

Table of Contents (Cont'd)

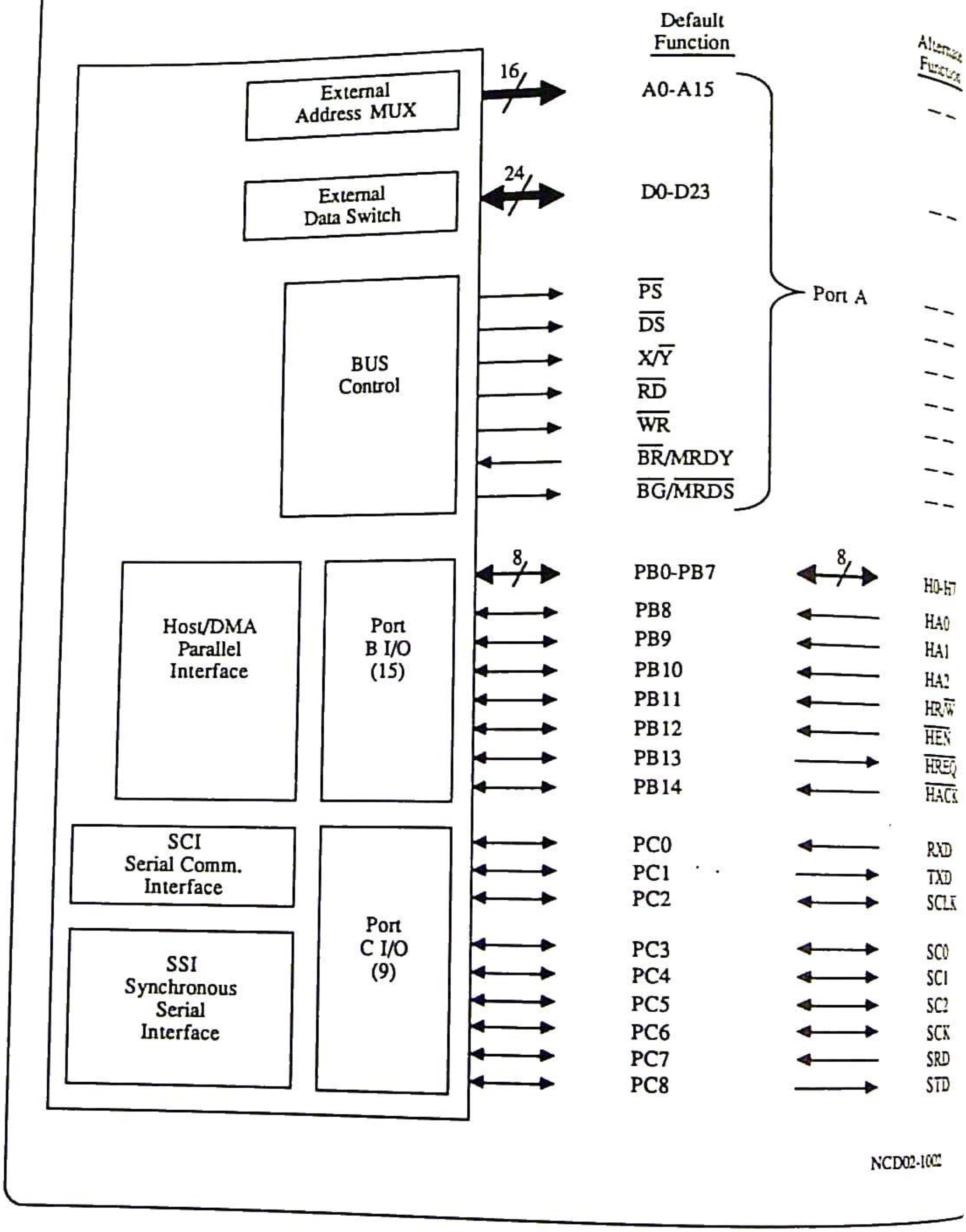
Bits Used for Host to DSP56000 Transfers 8-40
Data Transfer From DSP56000 to Host 8-41
Bits Used for DSP56000 to Host Transfers 8-42
Data Transfer for DSP56000 to Host 8-43
Host Read Cycle Timing 8-44
Host Write Cycle Timing 8-45
Host Exception 8-46
Host Interrupt Vector Read 8-47
Host Interface DMA Mode 8-48
DMA Transfer and Host Interrupts 8-48
Host Mode Bits with TREQ and RREQ 8-49
Host Mode and Initialize Bits 8-50
Host to DSP56000 DMA Procedure 8-50
Host DMA Write Cycle Timing 8-51
DSP to Host DMA Procedure 8-52
Host DMA Read Cycle Timing 8-53
Synchronous Serial Interface Features 8-54
SSI Operating Modes 8-55
SSI Initialization 8-55
Serial Control Direction Bits 8-56
SSI Control and Status Register Set 8-57
SSI Transmit and Receive Register Set 8-57
Synchronous Communications 8-58
Clock Modes 8-58
Frame Sync Length 8-59
SYN Bit 8-60
MODE Bit 8-61
SSI Clock 8-62
Normal Continuous Periodic Mode 8-63
Normal Mode Example 8-63
Normal Mode 8-64
SSI Time Division Multiplexed 8-64
Network Mode TDM 8-65
TDM Network Software Flow Chart 8-66
On Demand Example 8-67
Network Mode On Demand Data Driven 8-67
Output Flag Example 8-68
Output Flags 8-68
Input Flags 8-69
Control Pin Functions 8-70
SSI Exceptions 8-71

ON CHIP PERIPHERAL FEATURES

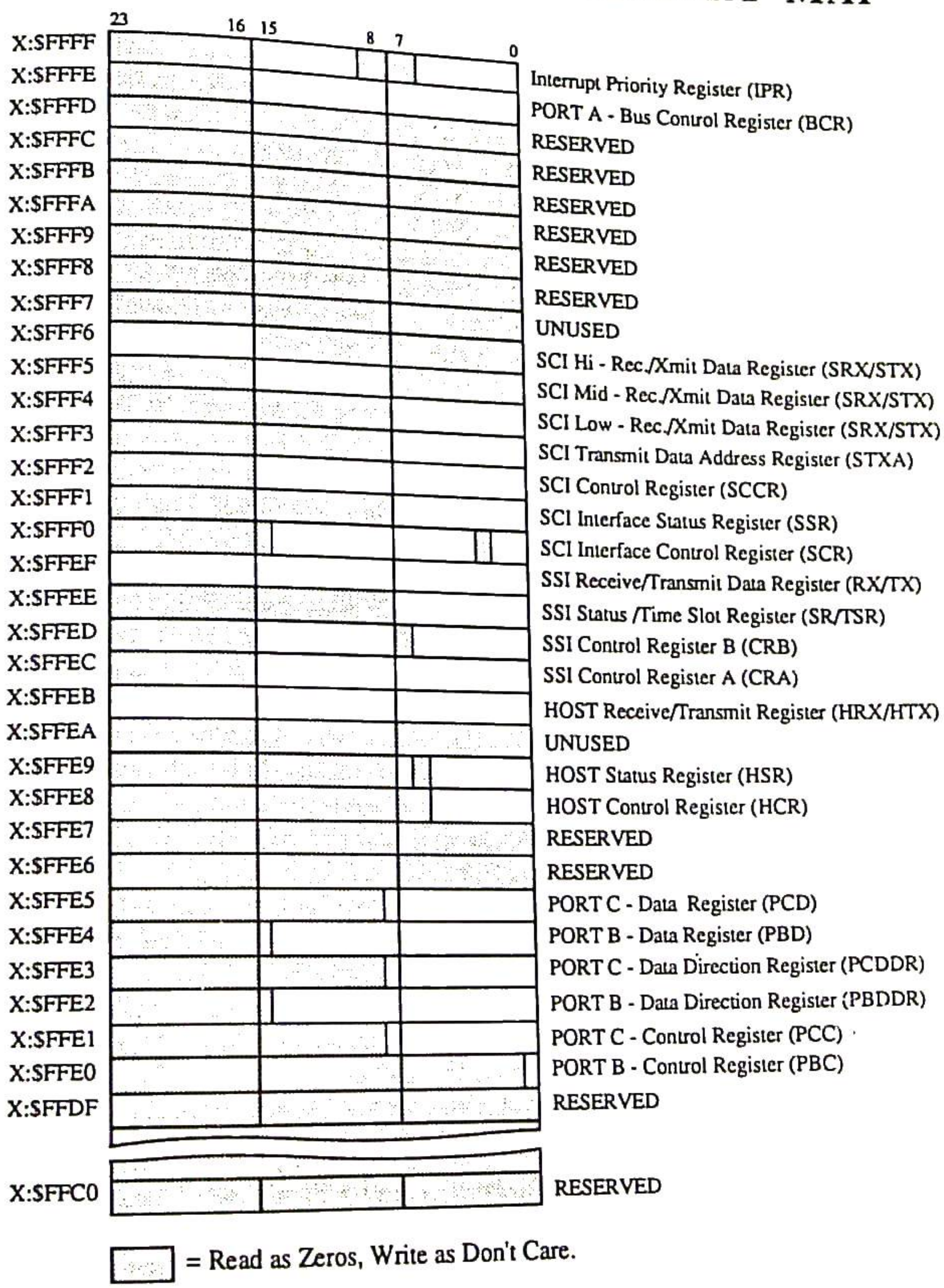
- 24 Separate Input/Output Pins
- Software Programmable as General Purpose Parallel I/O or Alternate Peripheral Function.
  - HOST Interface
  - Serial Communications Interface (SCI)
  - Synchronous Serial Interface (SSI)
- All Peripherals are Memory Mapped so Standard MOVE Instructions can be used.
- 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.

NCD02-1001

INPUT/OUTPUT BLOCK DIAGRAM



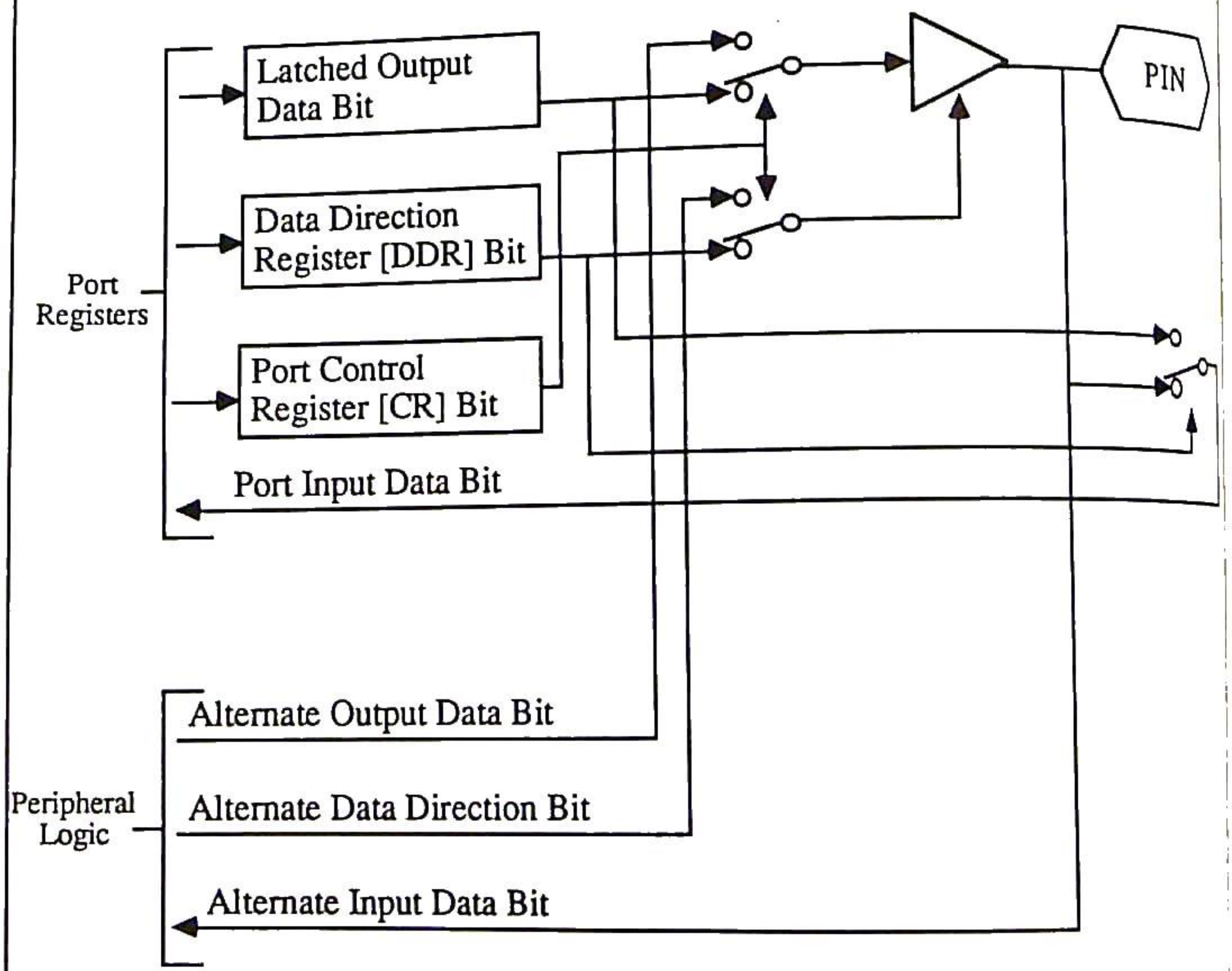
ON CHIP PERIPHERAL MEMORY MAP



NCD02-1003-1

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



NCD02-1004

PARALLEL PORT B REGISTERS

X:$FFE0 PORT B Control Register (PBC)

23								16	15							8	7							0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BC 0

BC 0	FUNCTION
0	Parallel I/O
1	HOST Interface

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

23								16	15							8	7							0
0	0	0	0	0	0	0	0	0	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	

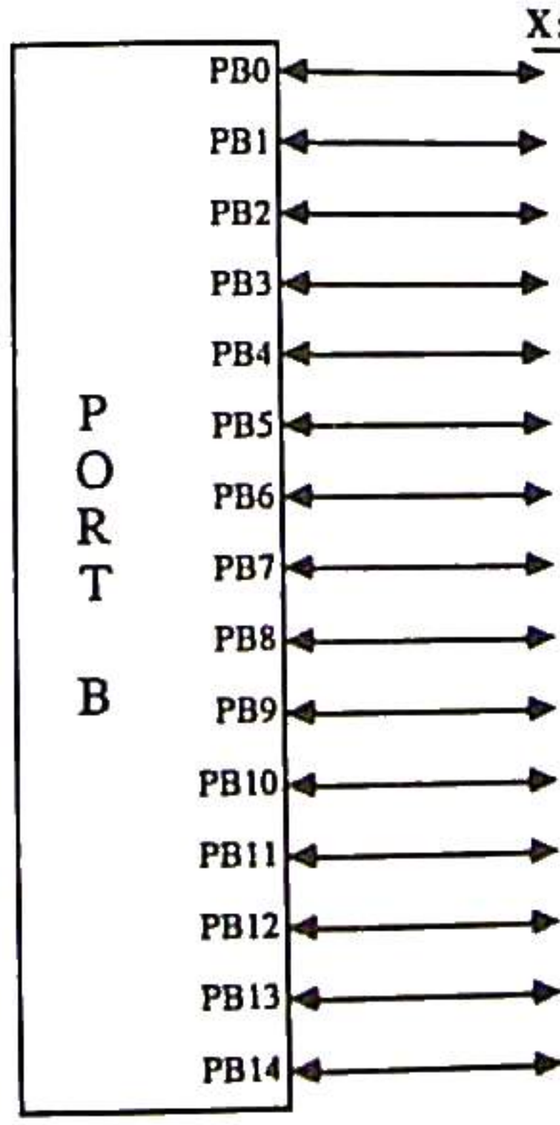
BDx	DATA DIRECTION
0	Input
1	Output

X:$FFE4 PORT B Data Register (PBD)

23								16	15							8	7							0
0	0	0	0	0	0	0	0	0	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	

NCD02-1005

PARALLEL PORT B PINOUT

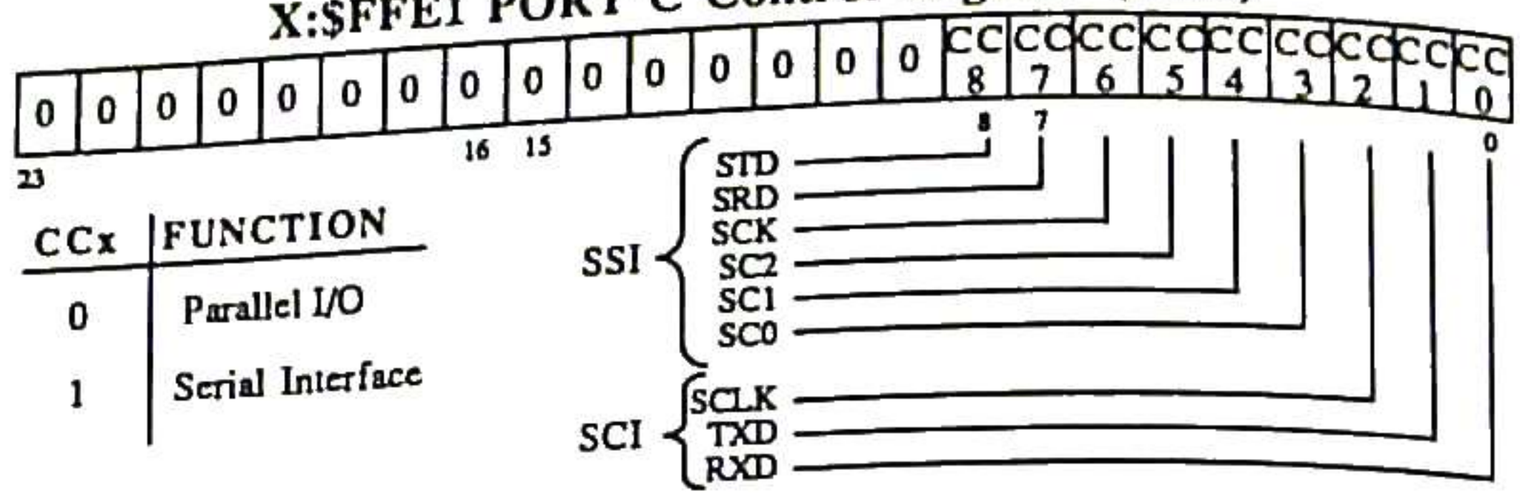


	Enabled by bit in X:$FFE0	Direction Selected by X:$FFE2	Input/Output Performed by X:$FFE4	Pin Grid Array Number
PB0	BC0	BD0	PB0	D12
PB1	BC0	BD1	PB1	C13
PB2	BC0	BD2	PB2	C12
PB3	BC0	BD3	PB3	B13
PB4	BC0	BD4	PB4	B12
PB5	BC0	BD5	PB5	A13
PB6	BC0	BD6	PB6	A12
PB7	BC0	BD7	PB7	B11
PB8	BC0	BD8	PB8	B8
PB9	BC0	BD9	PB9	A8
PB10	BC0	BD10	PB10	A7
PB11	BC0	BD11	PB11	A11
PB12	BC0	BD12	PB12	A10
PB13	BC0	BD13	PB13	B10
PB14	BC0	BD14	PB14	A9

NCD02-1006

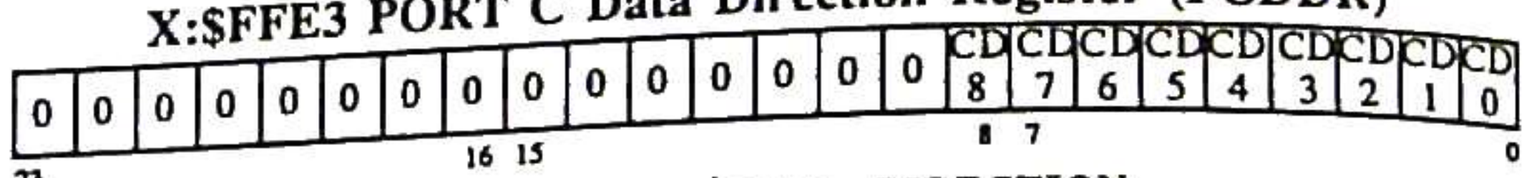
PARALLEL PORT C

X:$FFE1 PORT C Control Register (PCC)



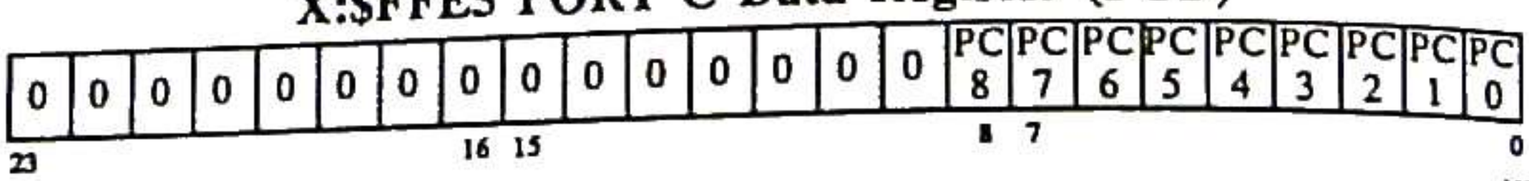
CCx	FUNCTION
0	Parallel I/O
1	Serial Interface

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



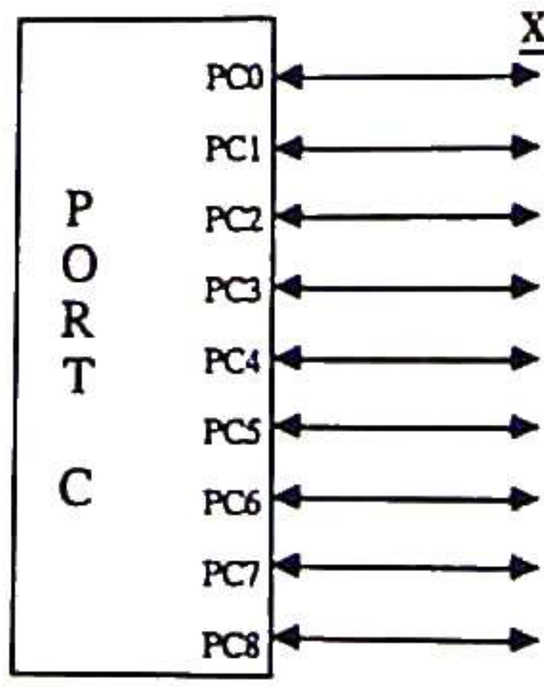
CDx	DATA DIRECTION
0	Input
1	Output

X:$FFE5 PORT C Data Register (PCD)



NCD02-1007

PARALLEL PORT C PINOUT



Enabled by bits in X:$FFE1	Direction Selected by X:$FFE3	Input/Output Performed by X:$FFE5	Pin Grid Array Number
CC0	CD0	PC0	D13
CC1	CD1	PC1	E13
CC2	CD2	PC2	F13
CC3	CD3	PC3	F12
CC4	CD4	PC4	K13
CC5	CD5	PC5	H13
CC6	CD6	PC6	G13
CC7	CD7	PC7	J12
CC8	CD8	PC8	J13

NCD02-1008

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

NCD02-1038

SCI CONTROL AND STATUS REGISTER SET

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

23	22	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	TMIE	TIE	RIE	ILIE	TE	RE	WOMS	RWU	WAKE	SBK	0	WDS2	WDS1	WDS0

Word Select Bits

Timer Interrupt Enable
Transmit Interrupt Enable
Receive Interrupt Enable
Idle Line Interrupt Enable
Transmitter Enable
Receiver Enable
Wired-OR Mode Select
Receiver Wake-up Enable
Wake-up Mode Select
Send Break

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

23	22	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	R8	FE	PE	OR	IDLE	RDRF	TDRE	TNRE

Received Bit 8
Framing Error Flag
Parity Error Flag
Over Run Error Flag
Transmitter Empty
Transmitter Data Register Empty
Receive Data Register Full
Idle Line Flag

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

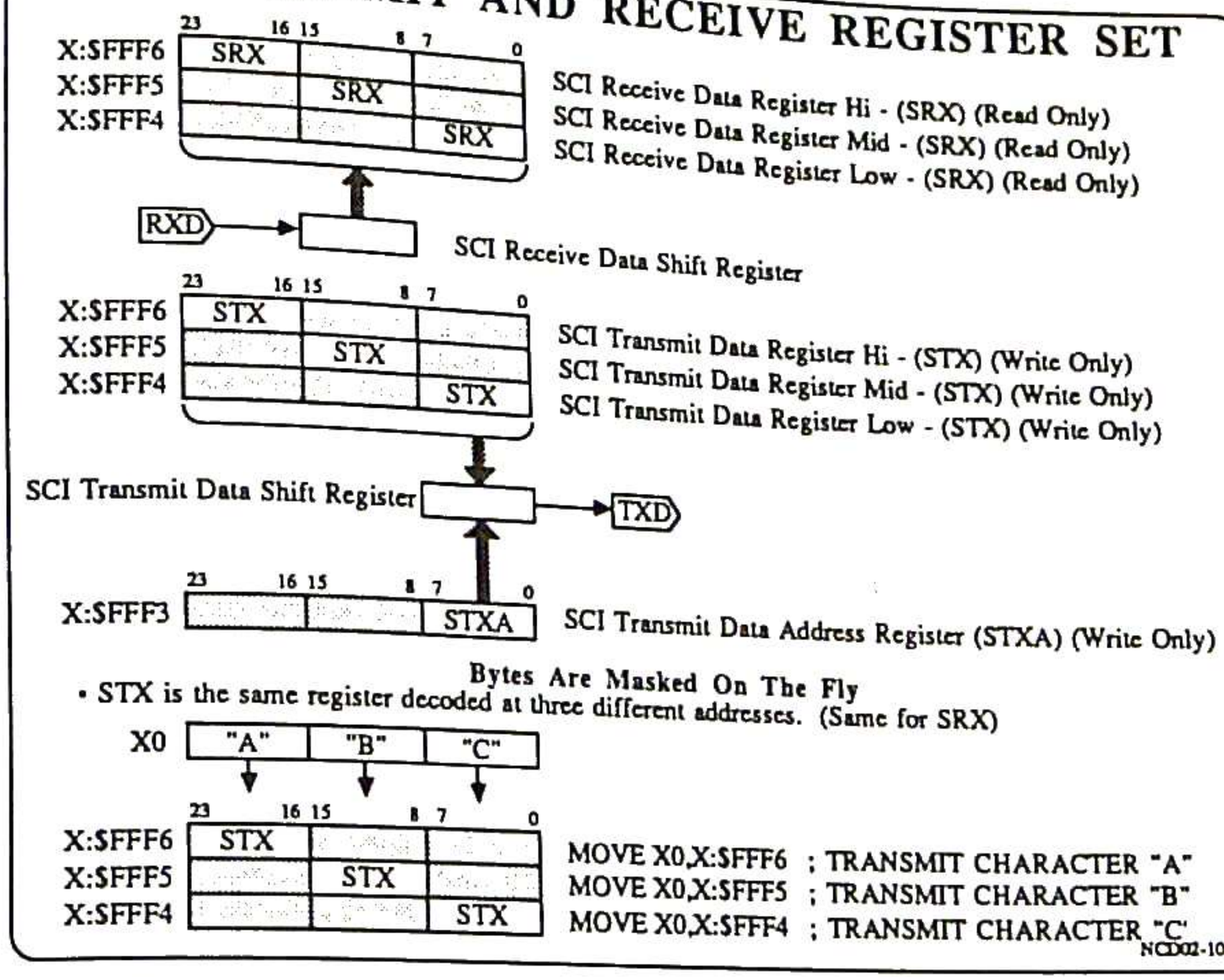
23	22	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	TCM	RCM	SCP	COD	CD11	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0

Clock Divider Bits

Clock Output Divider
SCI – Clock Prescaler
Receive Clock Source Bit
Transmit Clock Source Bit

NCD02-1019

SCI TRANSMIT AND RECEIVE REGISTER SET

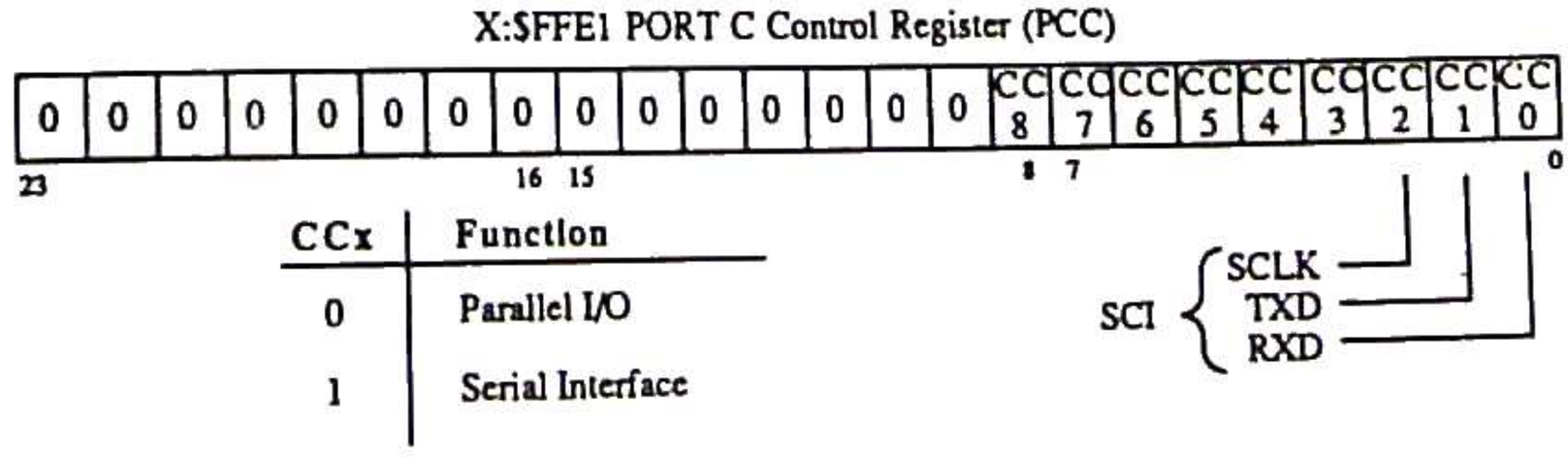


SCI RESET

1. Perform Hardware or Software RESET
2. Program SCI Control Registers
3. Configure at least one Port C Control bit as SCI Serial Interface.

Parallel Port C

X:$FFE1 PORT C Control Register (PCC)



CCx	Function
0	Parallel I/O
1	Serial Interface

4. SCI is now active.

Characteristics of SCI RESET

- At any time during program execution, the SCI may be put in the RESET state by clearing CC2, CC1, and CC0.
- All status bits of the SCI interface will be put to their RESET condition state.
- The contents of the SCI interface control and clock control registers are not affected.
- This permits separate resetting of the SCI.

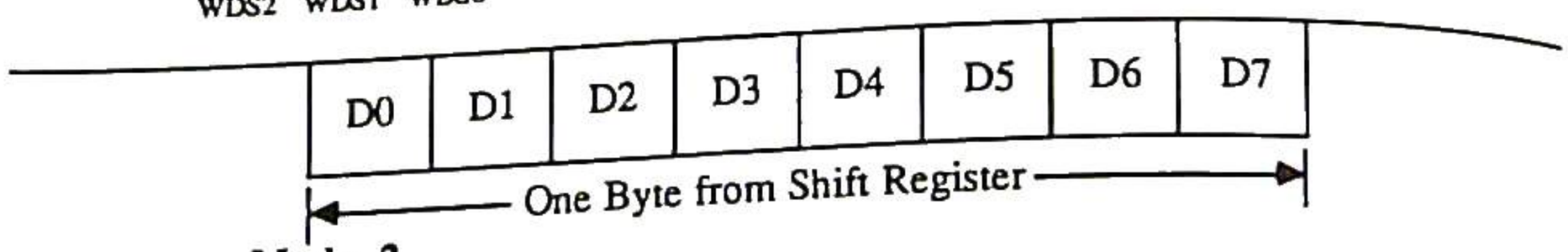
NCD02-1041

SERIAL FORMATS

Mode 0

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

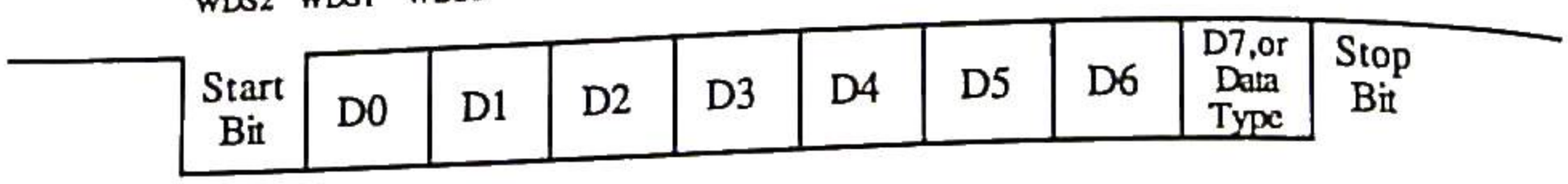
8 Bit Synchronous Data (Shift Register Mode)



Mode 2

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

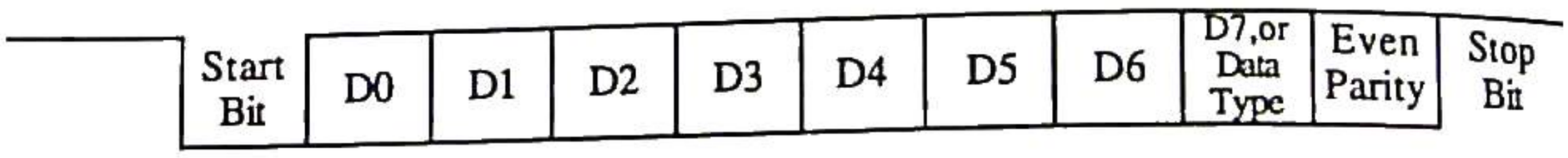
10 Bit Asynchronous (1 Start Bit - 8 Data Bits - 1 Stop Bit)



Mode 4

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

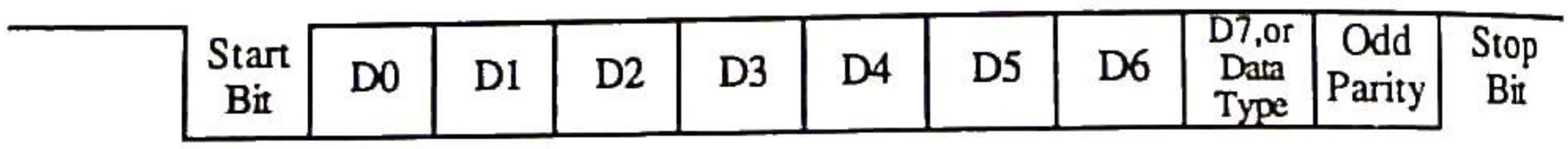
11 Bit Asynchronous
(1 Start Bit - 8 Data Bits - Even Parity Bit - 1 Stop Bit)



Mode 5

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

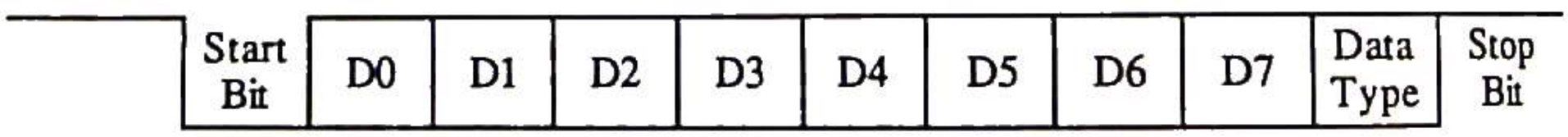
11 Bit Asynchronous
(1 Start Bit - 8 Data Bits - Odd Parity Bit - 1 Stop Bit)



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)



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

NCD02-1042

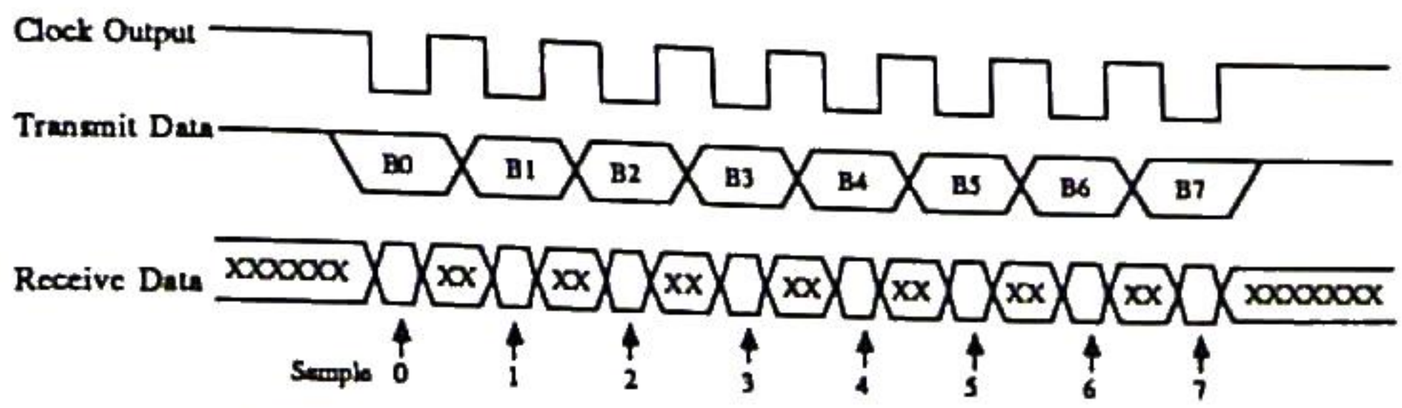
SCI SYNCHRONOUS MASTER MODE

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

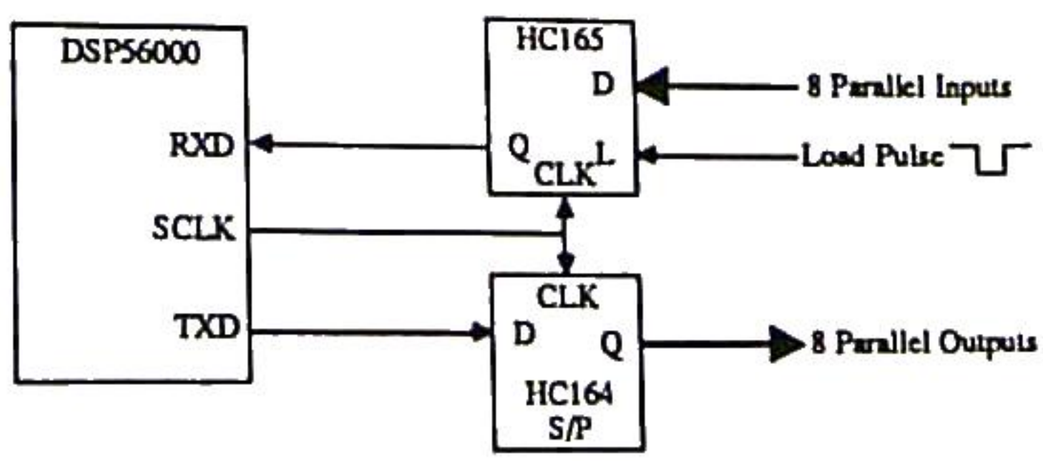
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	TMIE	TIE	RIE	ILIE	TE	RE	WOMS	RWU	WAKE	SBK	0	0	0	0
													WDS2	WDS1	WDS0

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
0	0	SCP	COD	CD11	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0
TCM	RCM														



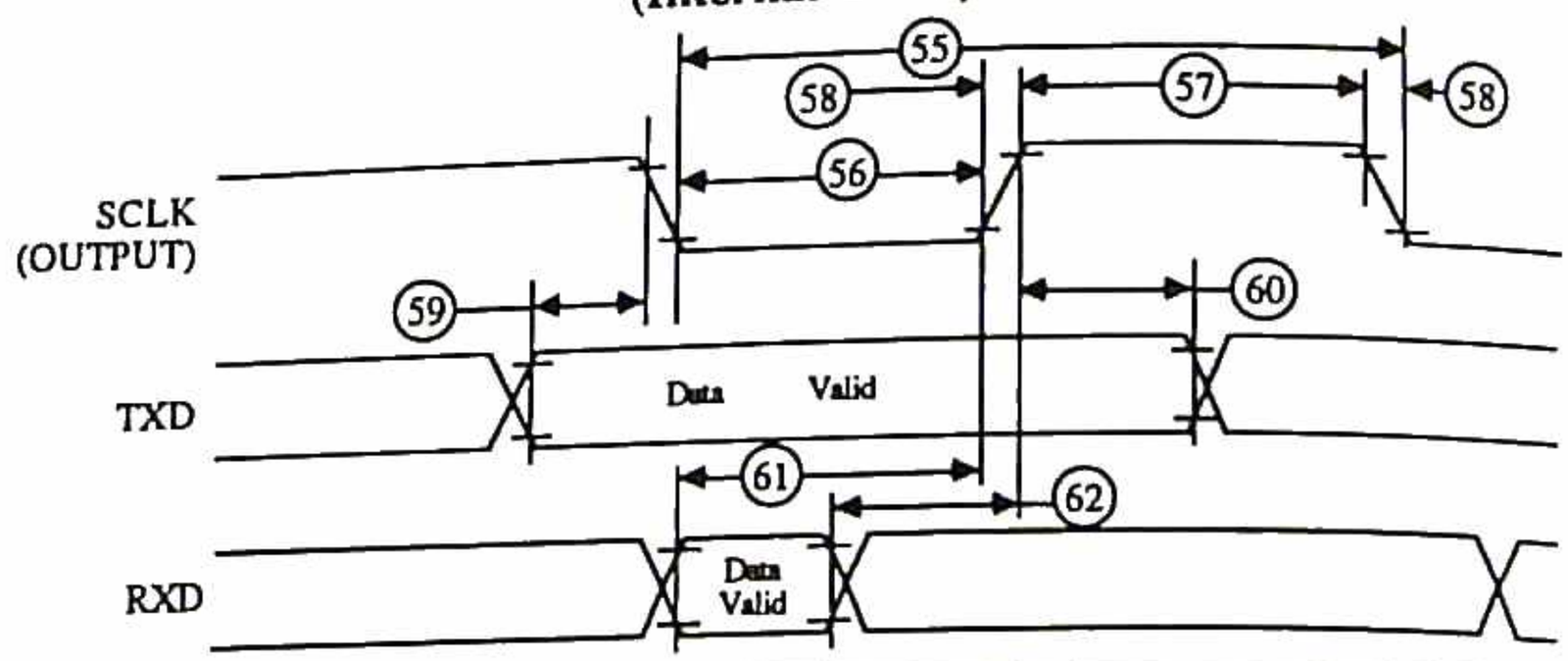
Example: Economical Shift Register I/O



NCD02-1043

SCI SYNCHRONOUS MASTER TIMING

(Internal Clock)



Num.	Characteristic	Min	Max	Unit
55	Synchronous Clock Cycle (see Note 1)	16T	–	ns
56	Clock Low Period	$t_{SCC}/2$ -20	–	ns
57	Clock High Period	$t_{SCC}/2$ -20	–	ns
58	Output Clock Rise Fall Time	–	20	ns
59	Output Data Setup to Clock Falling Edge (Internal Clock)	$t_{SCC}/4$ + T - 50	–	ns
60	Output Data Hold After Clock Rising Edge (Internal Clock)	$t_{SCC}/4$ - T - 15	–	ns
61	Input Data Setup Time Before Clock Rising Edge (Internal Clock)	$t_{SCC}/4$ + T + 45	–	ns
62	Input Data Not Valid Before Clock Rising Edge (Internal Clock)	–	$t_{SCC}/4$ - 10	ns

NOTE:
1. Synchronous clock cycle, t_{SCC}, (for internal clock) is determined by the SCI clock control register and I_{cyc}.

NCD02-1044

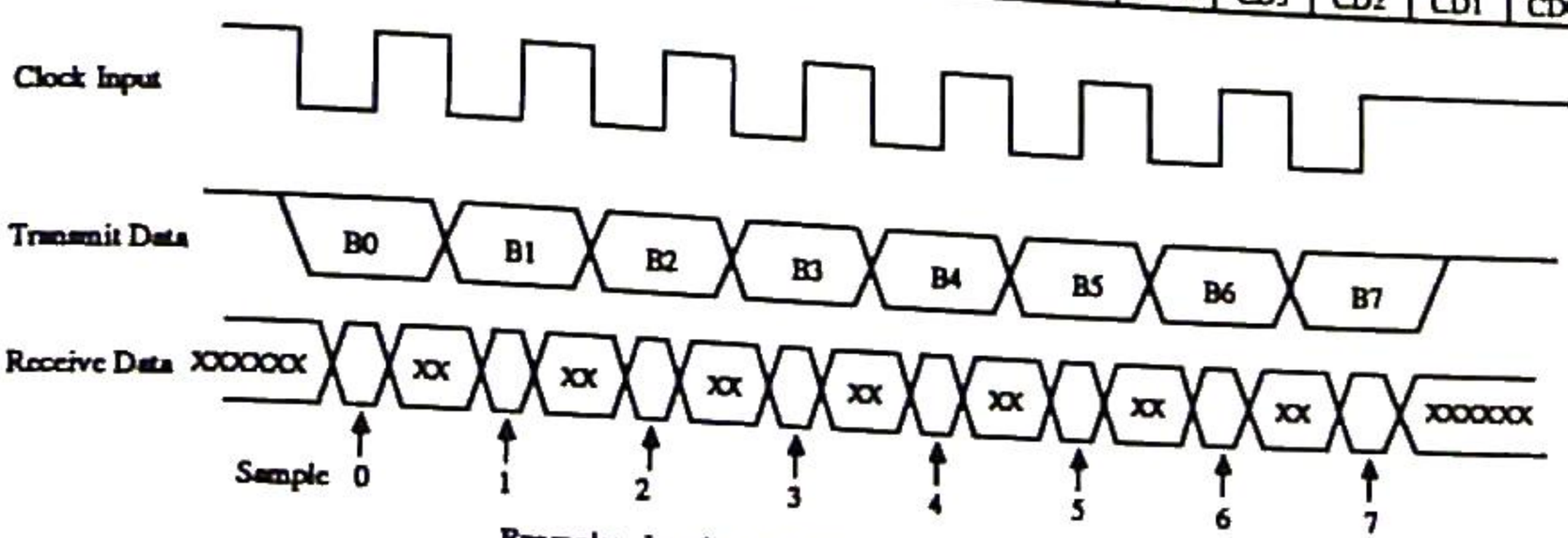
SCI SYNCHRONOUS SLAVE MODE

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

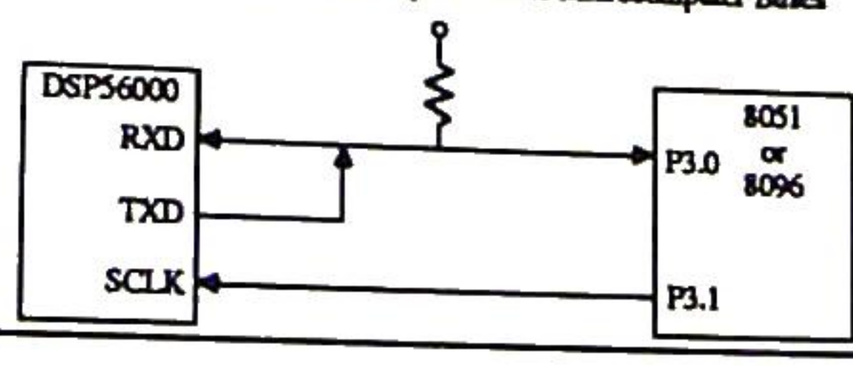
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	TMIE	TIE	RIE	ILIE	TE	RE	WOMS	RWU	WAKE	SBK	0	0	0	0
													WDS2	WDS1	WSD0

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
1	1	SCP	COD	CD11	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0
TCM	RCM														

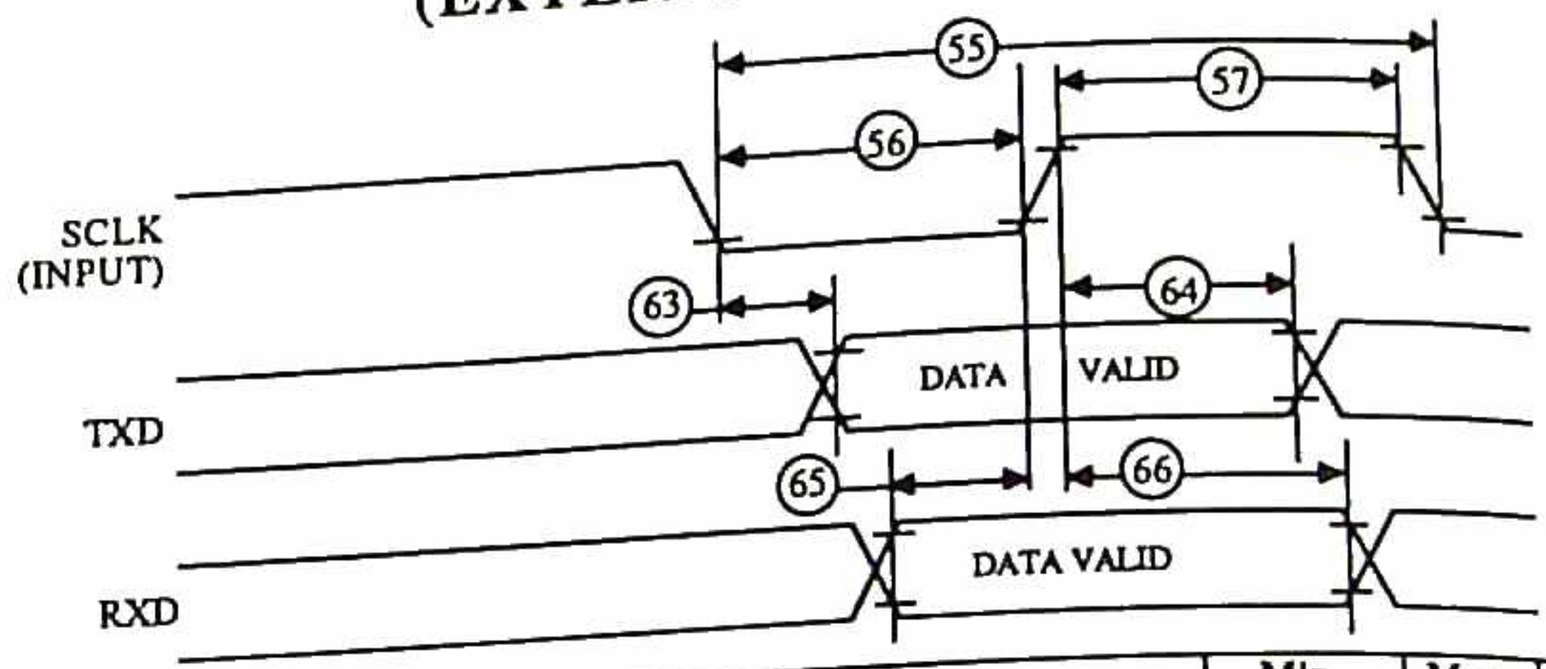


Example: Interface to Synchronous Microcomputer Buses



NCD02-1045-1

SCI SYNCHRONOUS SLAVE TIMING (EXTERNAL CLOCK)



Num.	Characteristic	Min	Max	Unit
55	Synchronous Clock Cycle (see Note 1)	16T	–	ns
56	Clock Low Period	16T/2-20	–	ns
57	Clock High Period	16T/2-20	–	ns
63	Clock Falling Edge to Output Data Valid (External Clock)	–	63	ns
64	Output Data Hold After Clock Rising Edge (External Clock)	2T + 12	–	ns
65	Input Data Setup Time Before Clock Rising Edge (External Clock)	30	–	ns
66	Input Data Hold Time After Clock Rising Edge (External Clock)	40	–	ns

NOTE:
1. Synchronous clock cycle, t_{SCC}, (for internal clock) is determined by the SCI clock control register and I_{cyc}.

NCD02-1046

CHARACTERISTICS OF SCI SYNCHRONOUS MODE

- Common clock to both Transmit and Receive shift registers

- As a controller:
  - Synchronous Master
  - TCM = RCM = 0
  - SCLK is an output
  - Outputs a clock when data is present
  - Transmitted data is stable during both the falling and rising clock edge
  - Received data is sampled in the middle of the clock low time

- As a peripheral:
  - Synchronous Slave
  - TCM = RCM = 1
  - SCLK is an input
  - Data is clocked out on the falling edge of SCLK
  - Data is clocked in on the rising edge of SCLK

- WARNING: TCM ≠ RCM is not permitted in the synchronous mode, and the results are undefined.

NCD02-1047

SCI RECEIVER INITIALIZATION

1. Hardware or Software RESET
2. Program SCR with desired mode and features.
3. Turn on Receiver (RE = 1).
4. 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 — Receiver Interrupt Enable

RE — Receiver Enable

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

X:$FFE1 PORT C Control Register (PCC)

23							16	15							8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	1

← SCI →

CCx	Function
0	Parallel I/O
1	Serial Interface

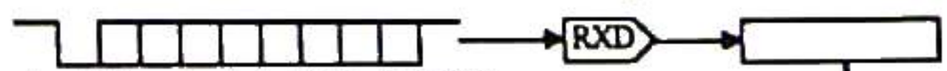
RXD — Receive Data Pin Enable

- If RE is cleared while a valid character is being received, the reception of the character will be completed before the receiver is disabled.

NCD02-1048

SCI RECEIVE A CHARACTER

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



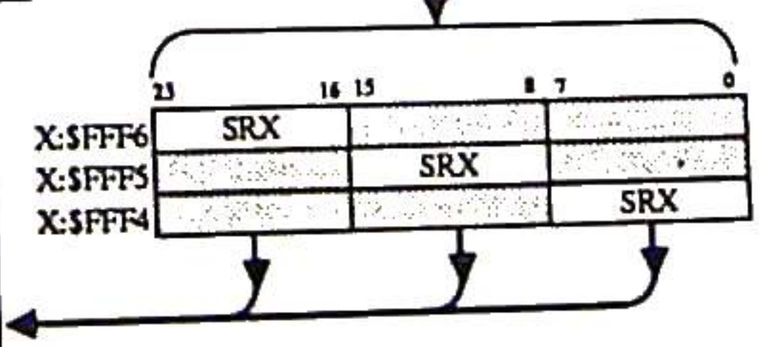
2. Transferring the received Character into SRX sets RDRF in the SSR.

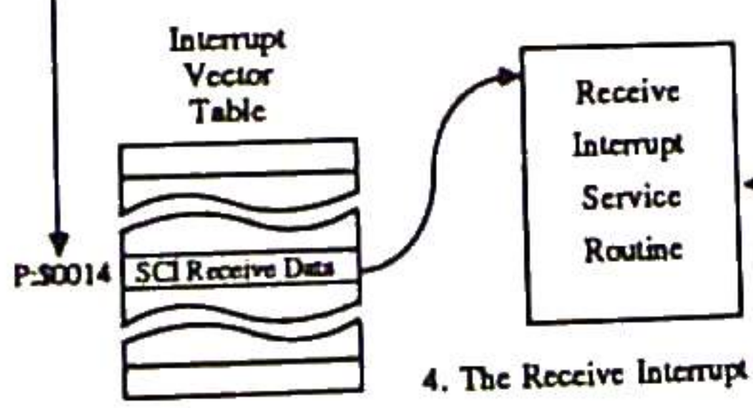
X:$FFF1 - SCI Interface Status Register (SSR) (Read Only)

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

RDRF

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





5. Reading SRX clears RDRF in the SSR.

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

NCD02-1049

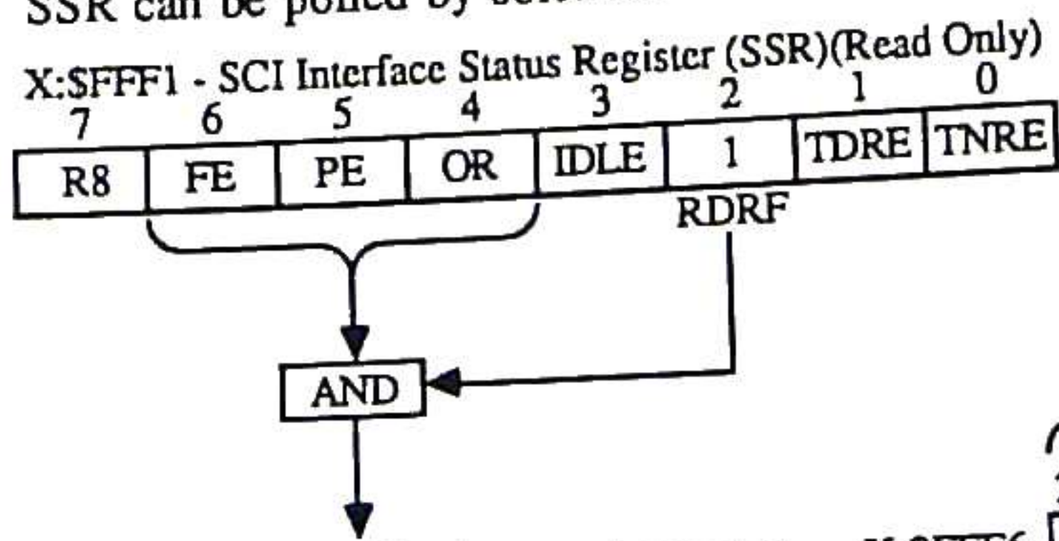
SCI - RECEIVE A CHARACTER WITH EXCEPTION

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

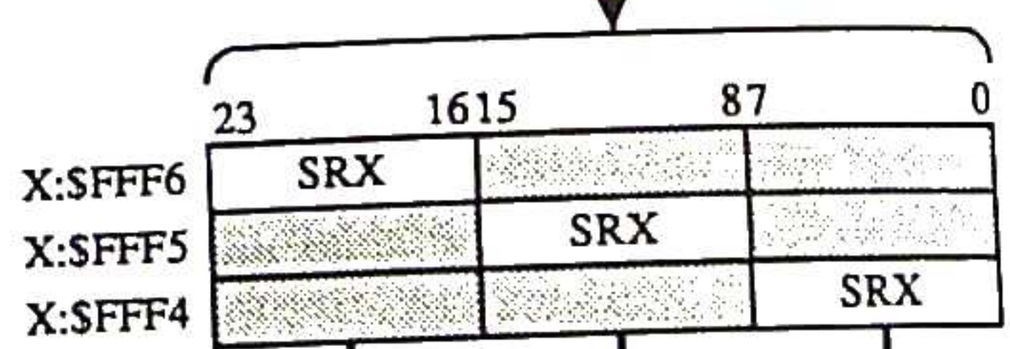
   - Framing Error (FE = Bit 6 in SSR)
   - Parity Error (PE = Bit 5 in SSR)
   - Overrun Error (OR = Bit 4 in SSR)

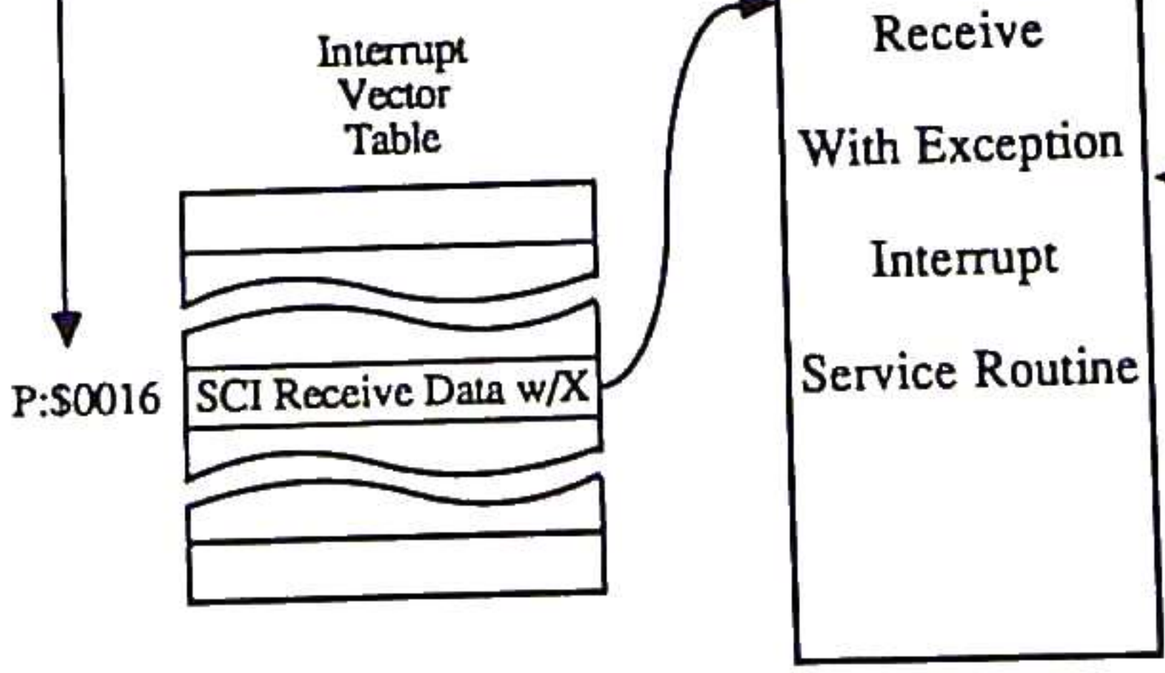
2. This sets RDRF and FE, PE, or OR in SSR.

3. SSR can be polled by software



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





5. Read SSR, followed by

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

NCD02-1050

SCI TRANSMITTER INITIALIZATION

1. Hardware or Software RESET
2. Program SCR with Desired Mode and Features.
3. Turn on Transmitter (TE = 1)
4. 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

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

X:$FFE1 Port C Control Register (PCC)

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

← SCI →

TXD — Transmit Data Pin Enable

CCx	FUNCTION
0	PARALLEL I/O
1	SERIAL INTERFACE

6. The transmitter will first broadcast a preamble of ones before beginning data transmission
   10 ones will be transmitted for the 10 bit asynchronous mode
   11 ones will be transmitted for the 11 bit asynchronous mode

- If TE is cleared while transmitting a character, the transmission of the character will be completed before the transmitter is disabled.

NCD02-1051

SCI TRANSMIT A CHARACTER

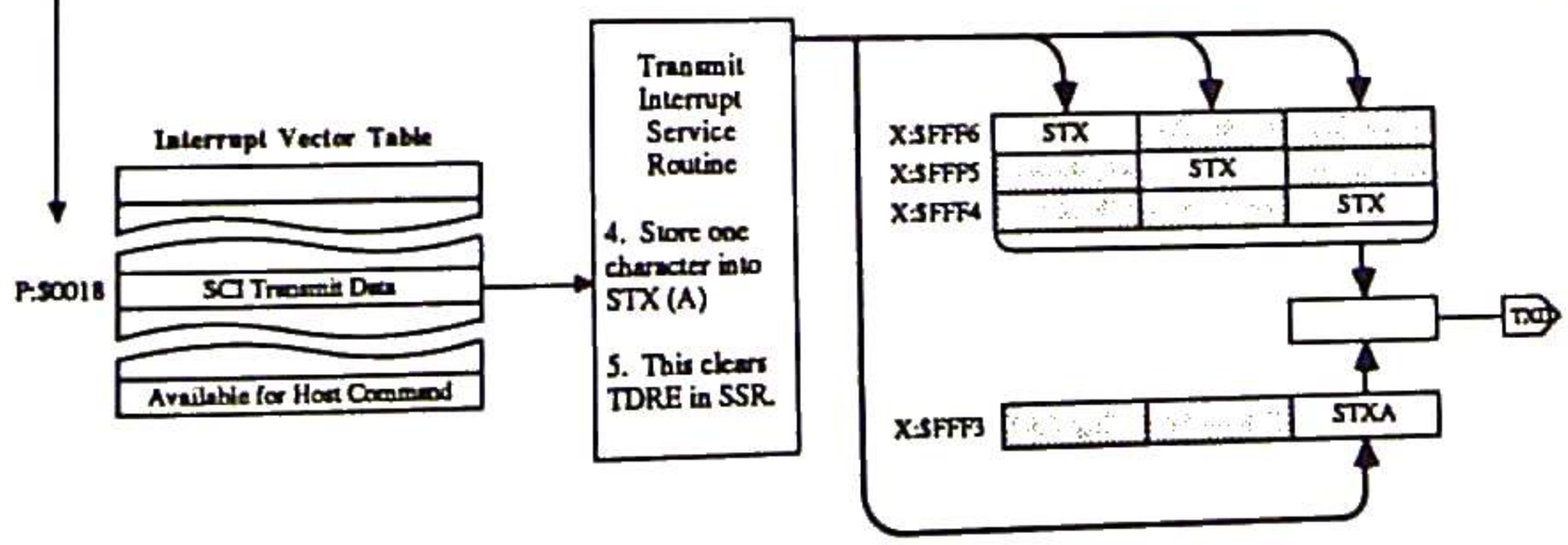
X:$FFF1 - SCI Interface Status Register (SSR)(Read Only)

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

TDRE — Transmit Data Register Empty

TRNE — Transmitter Empty

1. When STX is empty, then TDRE = 1
2. When STX is empty and the Transmit Data Shift Register is empty then TRNE = 1
3. If TIE = 1 in SCR and TDRE = 1 in SSR, then an interrupt is generated.



6. The character in STX is copied into transmit data shift register.
   TRNE is cleared
   TDRE is set
   Go to Step 3.

NCD02-1052

TRANSMITTING MARKS AND SPACES

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

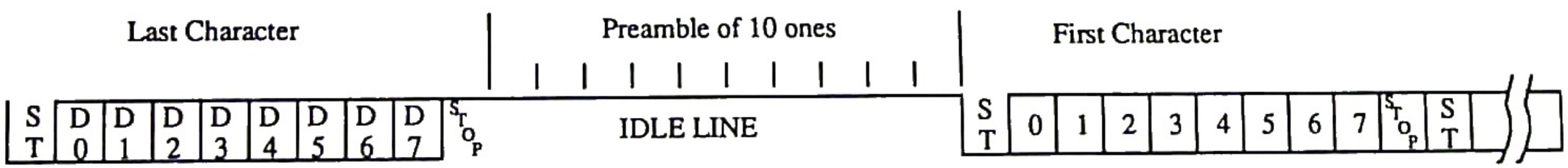
Toggle (1-0-1) to send a character time of all ones (marks)

Toggle (0-1-0) to send a character time of all zeros (spaces)

- 10 or 11 ones/zeros will be sent depending on the word length specified by WDS2, WDS1, WDS0.

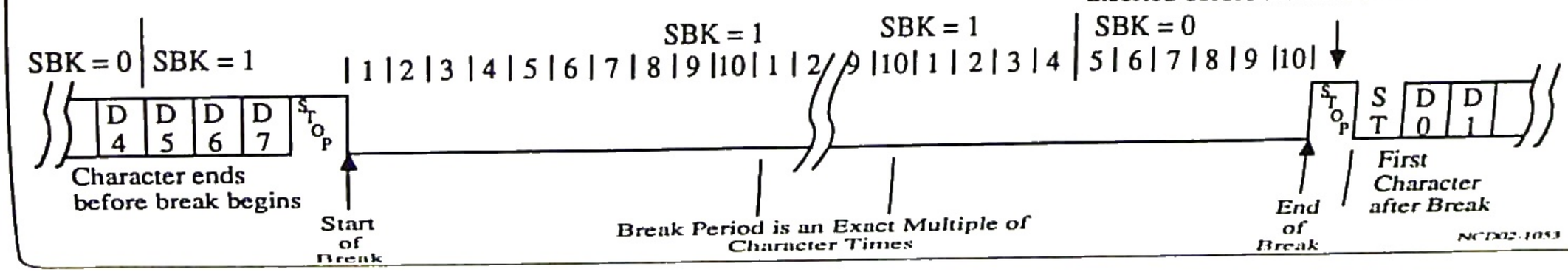
Marks (Ones)

1. Write the last byte to STX
2. Wait for TDRE = 1. The last byte is now in the transmit shift register
3. Clear TE and set back to one. This queues the preamble to follow the last byte
4. Write the first byte to follow the preamble into SRX

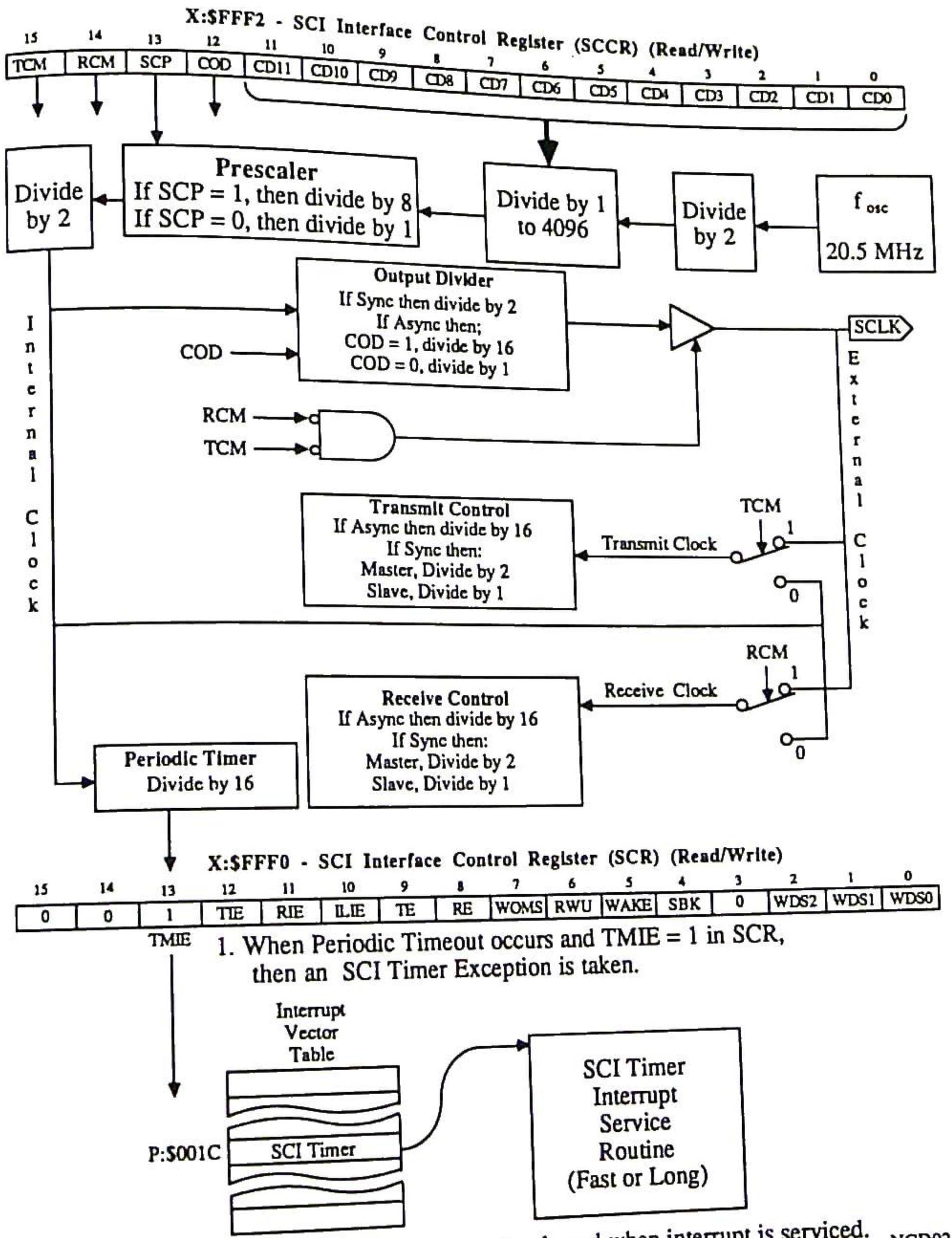


SPACES (ZEROS)

A stop bit at the end of the break will be inserted before the next character starts



SCI - CLOCK AND TIMER

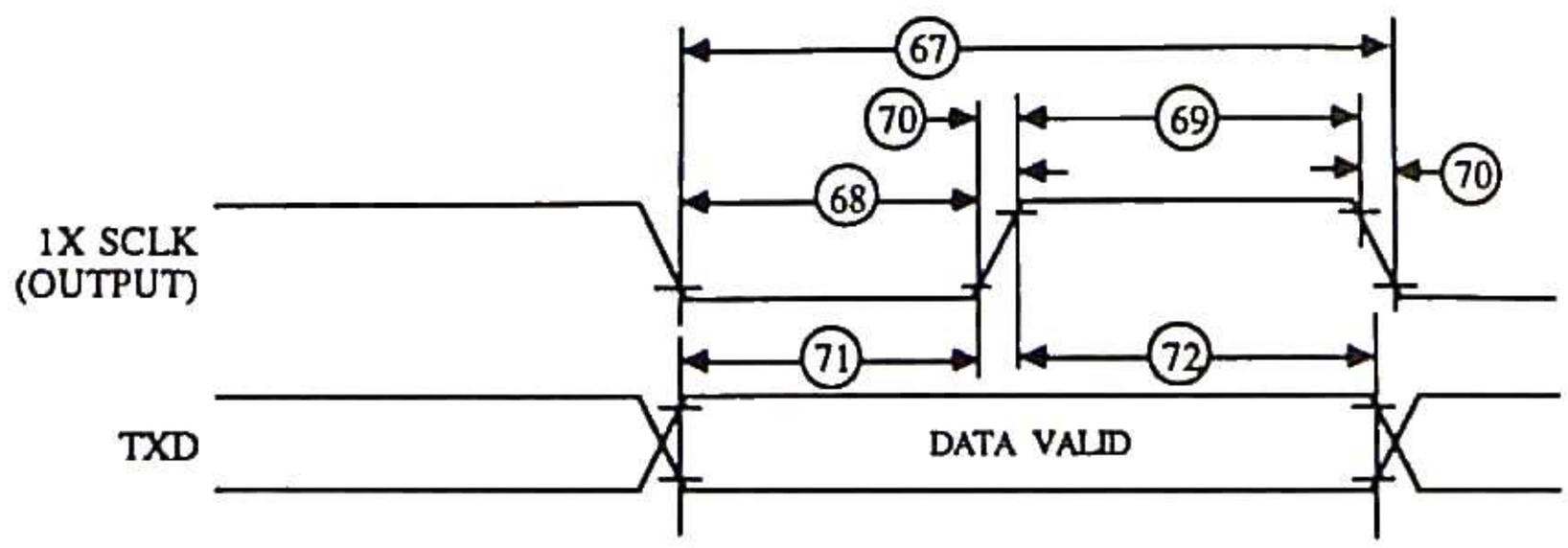


COMMON BAUD RATES

BAUD Rate	After Prescaler	SCP Bit	After 12 Bit Divider	Divider Bits CD0 - CD11	Fosc
350000	11,200,000	0	11,200,000	1	20,500,000
9600	307,200	0	307,200	33	20,275,200
4800	153,600	0	153,600	66	20,275,200
2400	76,800	0	76,800	132	20,275,200
1200	38,400	0	38,400	264	20,275,200
300	9,600	0	9,600	1056	20,275,200
9600	307,200	1	2,457,600	4	19,660,800
4800	153,600	1	1,228,800	8	19,660,800
2400	76,800	1	614,400	16	19,660,800
1200	38,400	1	307,200	32	19,660,800
300	9,600	1	76,800	128	19,660,800

NCD02-1055

SCI ASYNCHRONOUS MODE TIMING



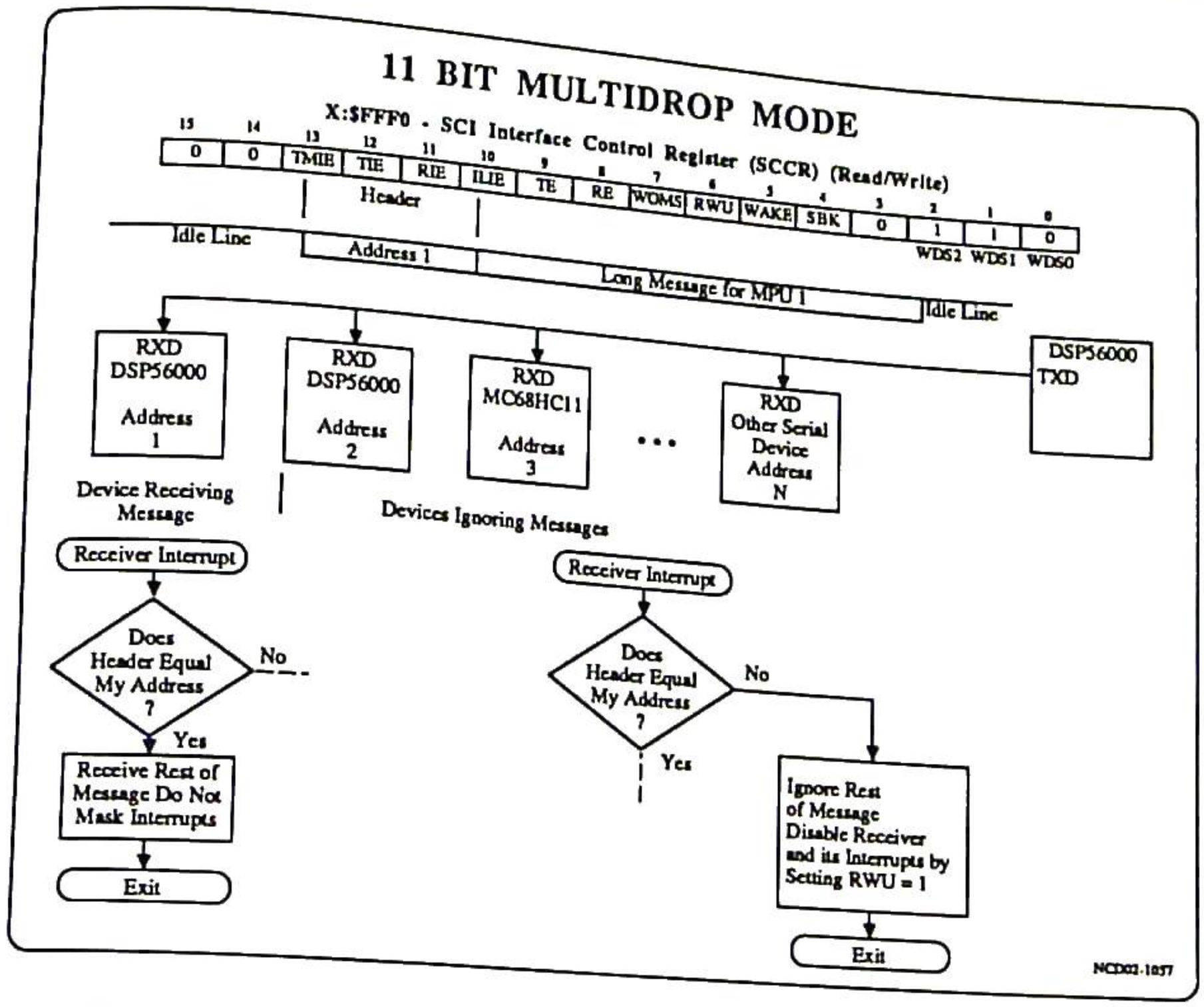
NOTE: In the wire-OR mode TXD can be pulled up by 1 kΩ.

Num.	Characteristic	Min	Max	Unit
67	Asynchronous Clock Cycle (see Note 2)	128T	–	ns
68	Clock Low Period	t_{ACC} / 2 - 20	–	ns
69	Clock High Period	t_{ACC} / 2 - 20	–	ns
70	Output Clock Rise/Fall Time	–	20	ns
71	Output Data Setup to Clock Rising Edge (Internal Clock)	t_{ACC} / 2 - 100	–	ns
72	Output Data Hold After Clock Rising Edge (Internal Clock)	t_{ACC} / 2 - 100	–	ns

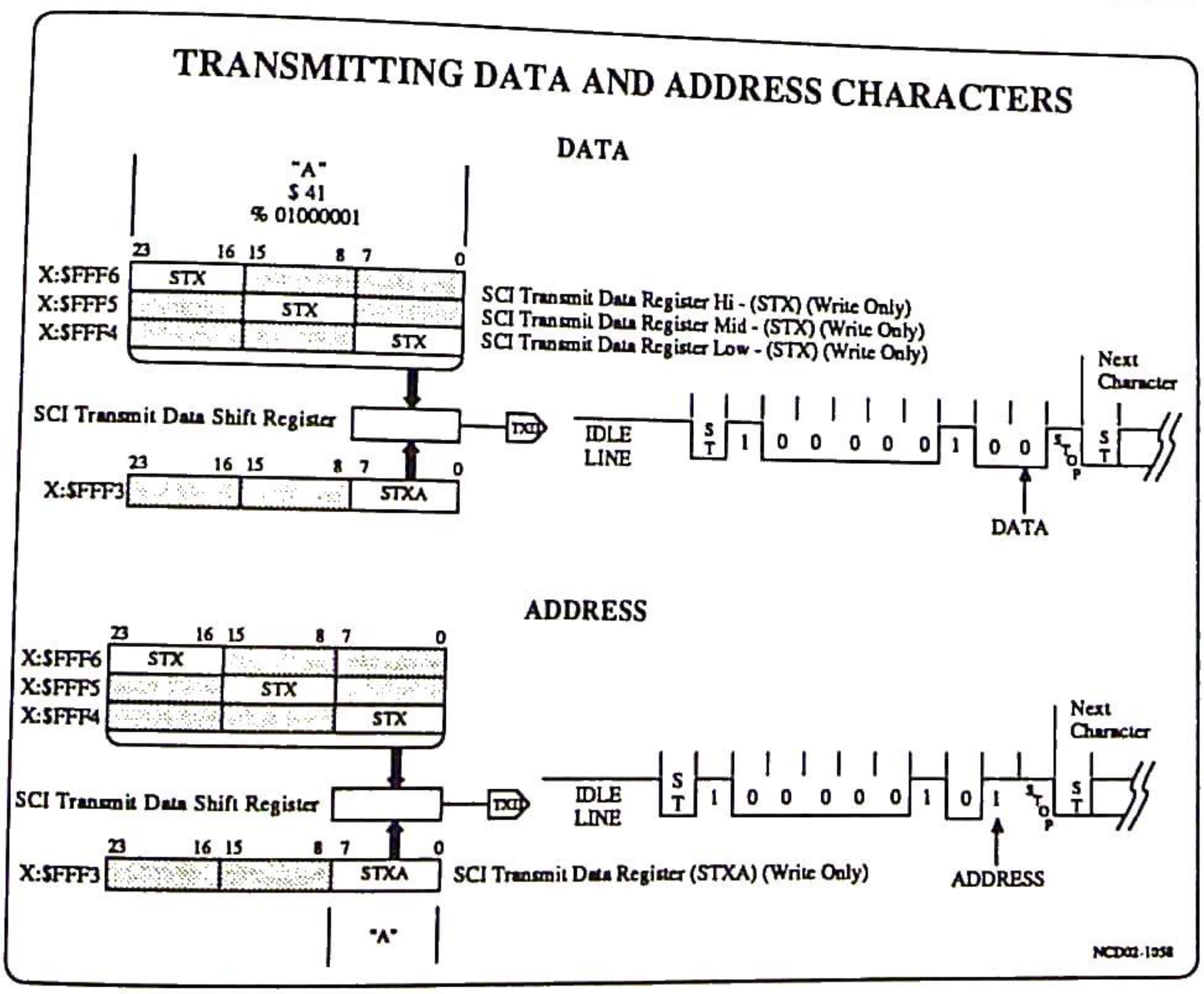
NOTE:
2. Asynchronous clock cycle, t_{ACC} (for internal clock) is determined by the SCI clock control register and I_{cyc}.

NCD02-1056

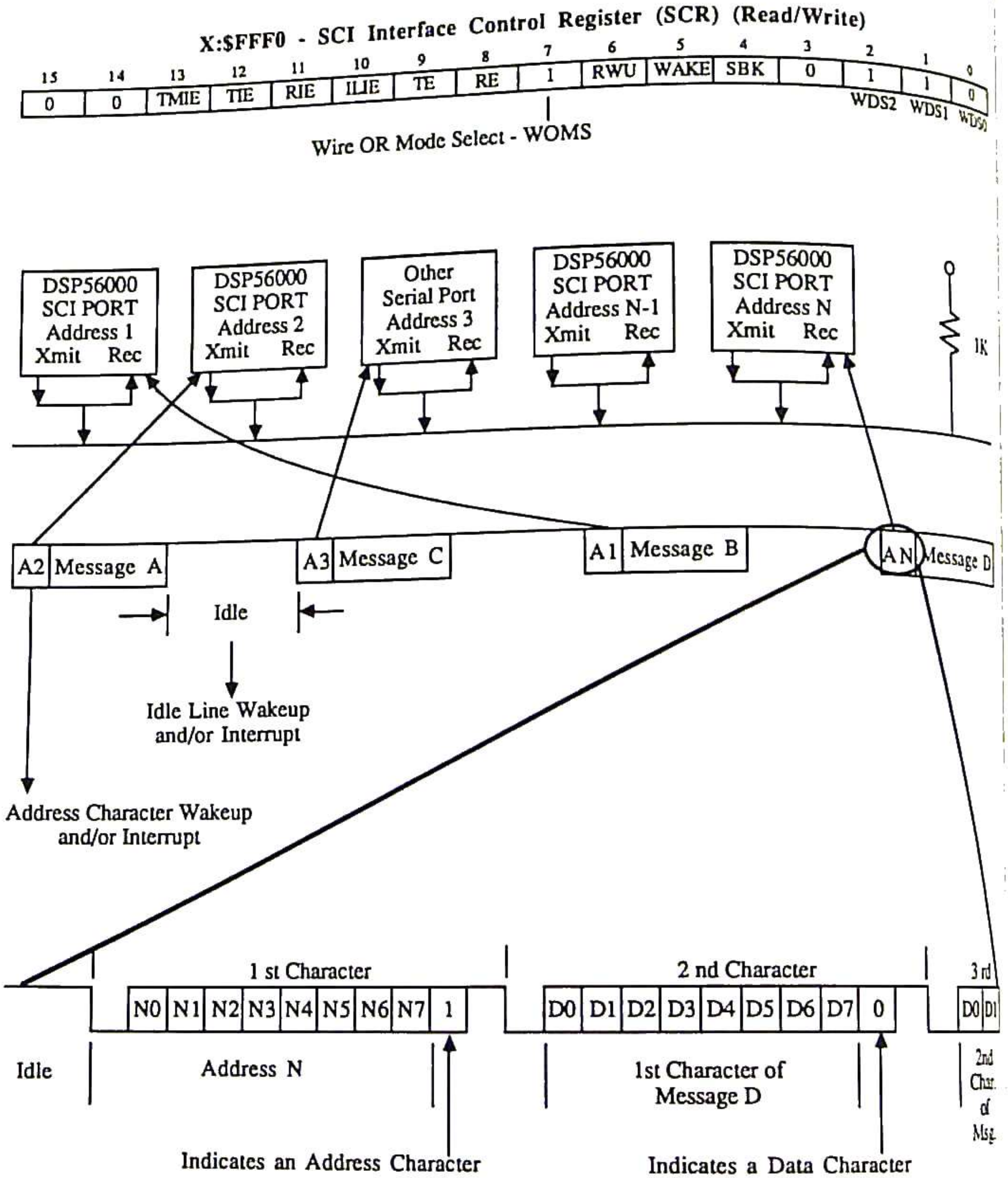
11 BIT MULTIDROP MODE



TRANSMITTING DATA AND ADDRESS CHARACTERS



WIRED-OR MODE



IDLE LINE WAKE UP

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

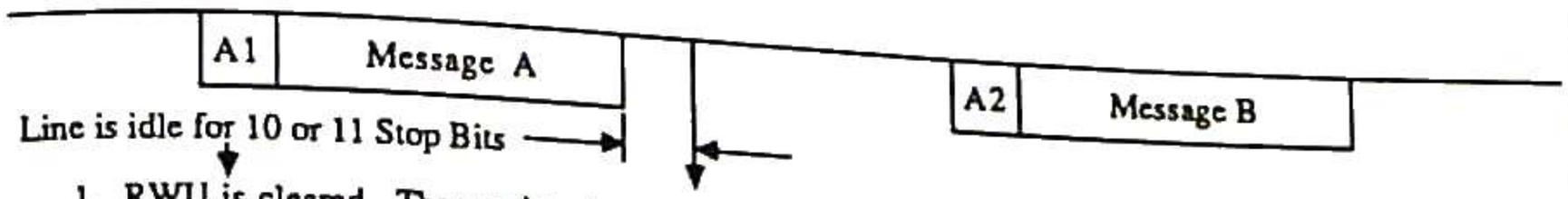
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	TMIE	TIE	1	1	TE	RE	WOMS	1	0	SBK	0	WDS2	WDS1	WDS0

RIE — Receive Interrupt Enable

ILIE — Idle Line Interrupt Enable

RWU — Receiver Wake-up Enable

WAKE — Idle Line Wake-up

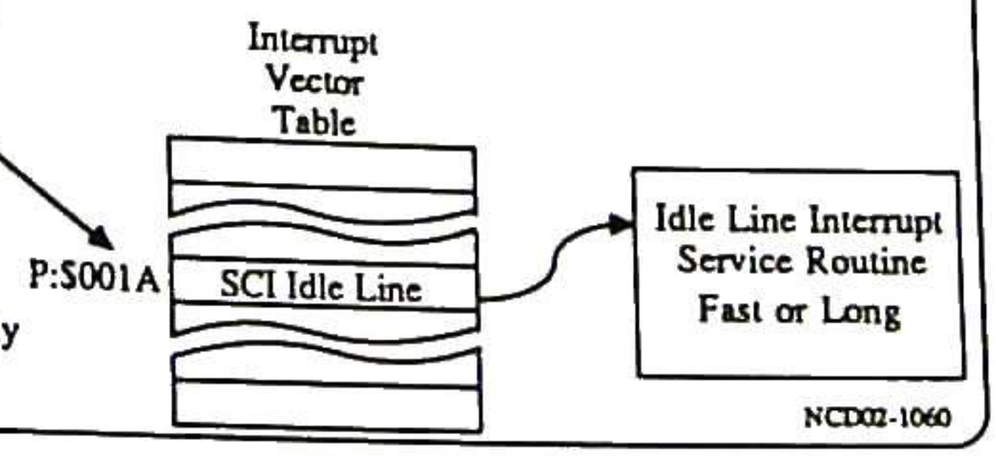


1. RWU is cleared. The receiver is enabled.
2. IDLE is set in SSR indicating the line is idle.
3. An internal flag SRIINT is generated for once each idle state, no matter how long it lasts.

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

IDLE (SRIINT)

4. If ILIE = 1 in SCR, then an SCI Idle Line Interrupt is pending.
5. When IDLE line interrupt is serviced then SRIINT is automatically cleared.



NCD02-1060

ADDRESS MODE WAKE UP

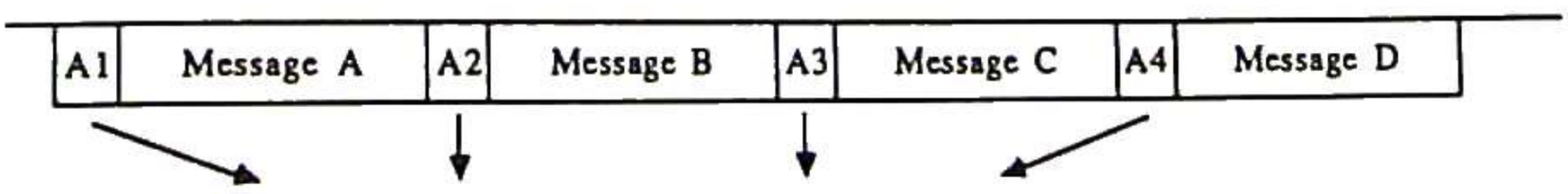
X:$FFF0 - SCI Interface Control Register (SCR) (Read/Write)

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

RIE — Receive Interrupt Enable

RWU — Receiver Wake-up Enable

WAKE — Address Wake-up



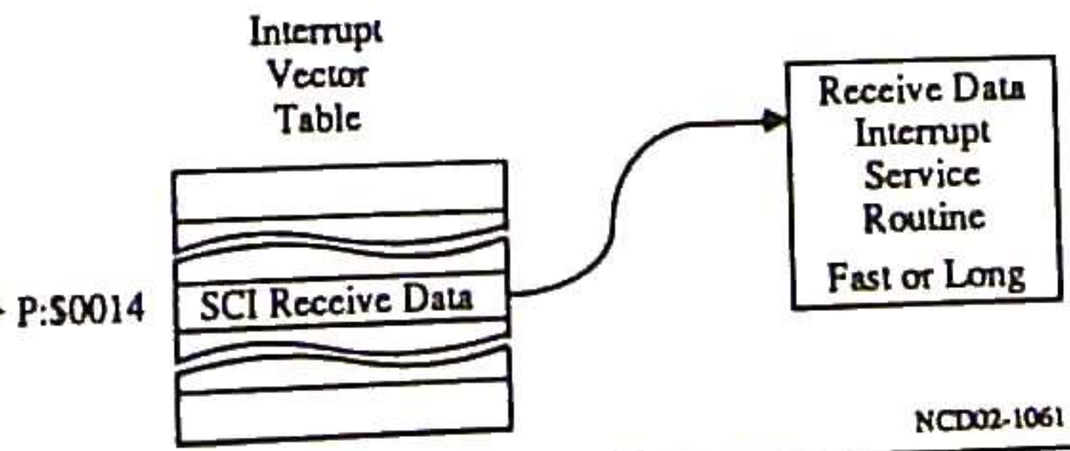
1. When Address Character is received, then R8 = 1 in SSR and RWU is cleared. The Receiver "Wakes Up".

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



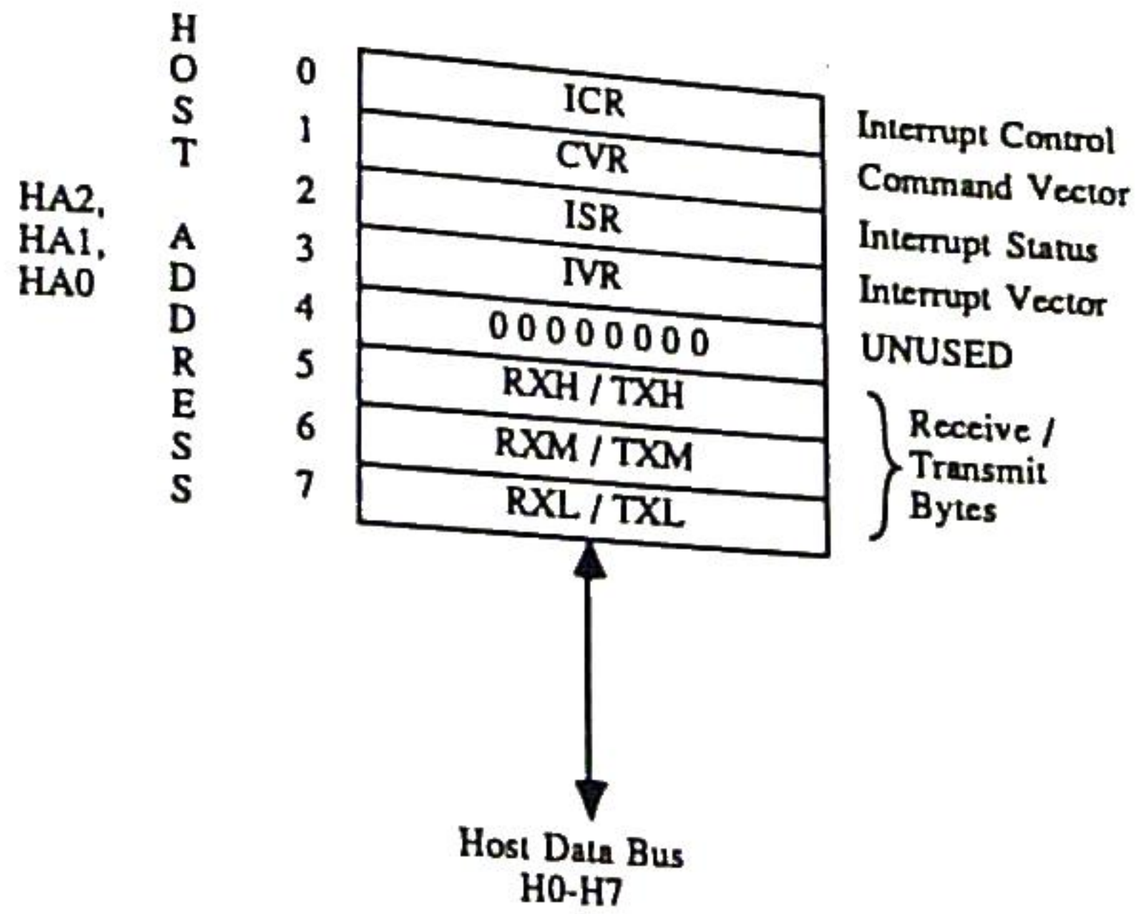
NCD02-1061

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.

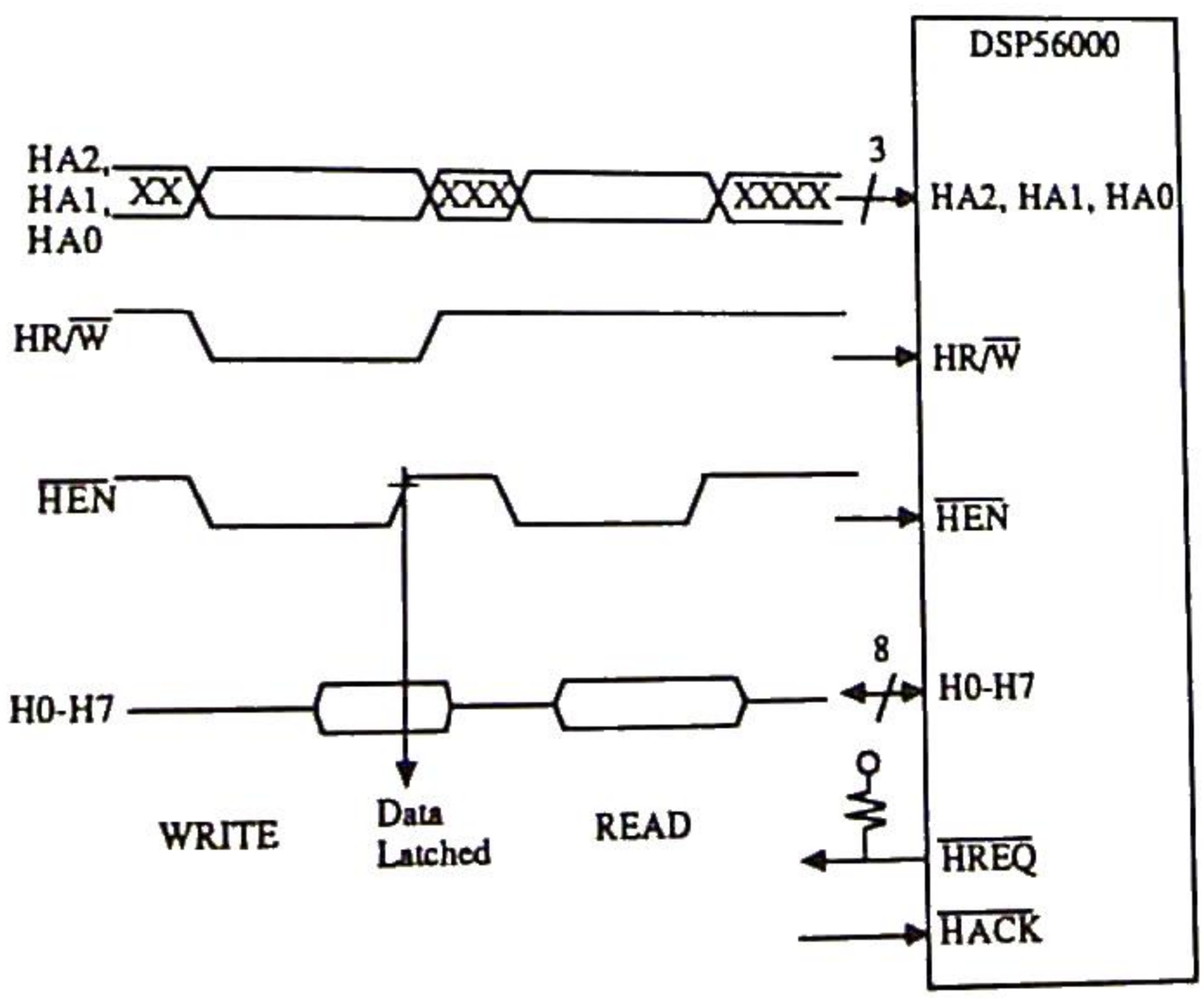
NCD02-1009-1

HOST INTERFACE REGISTER MAP



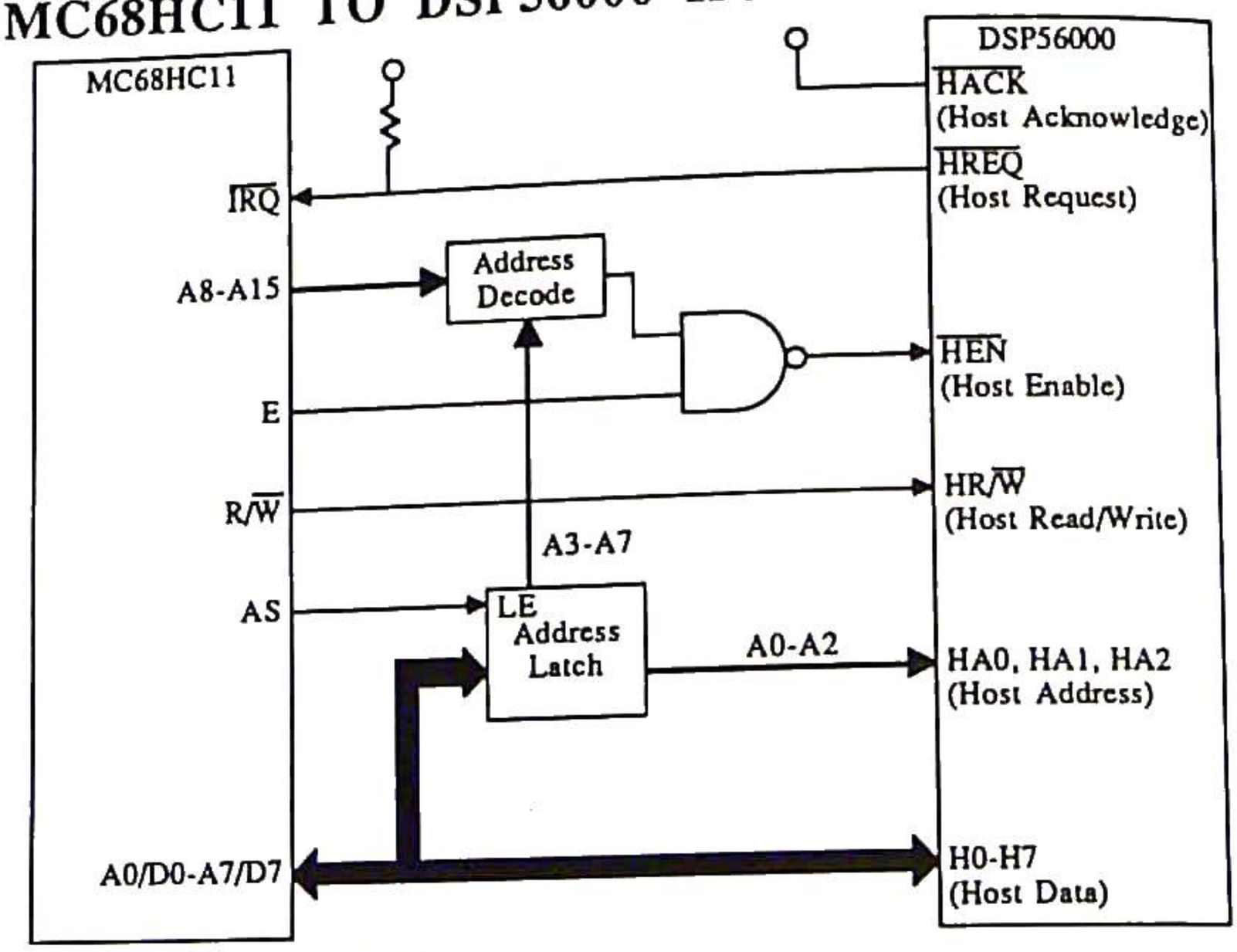
NCD02-1010

HOST TRANSFER TIMING



NCD02-1011

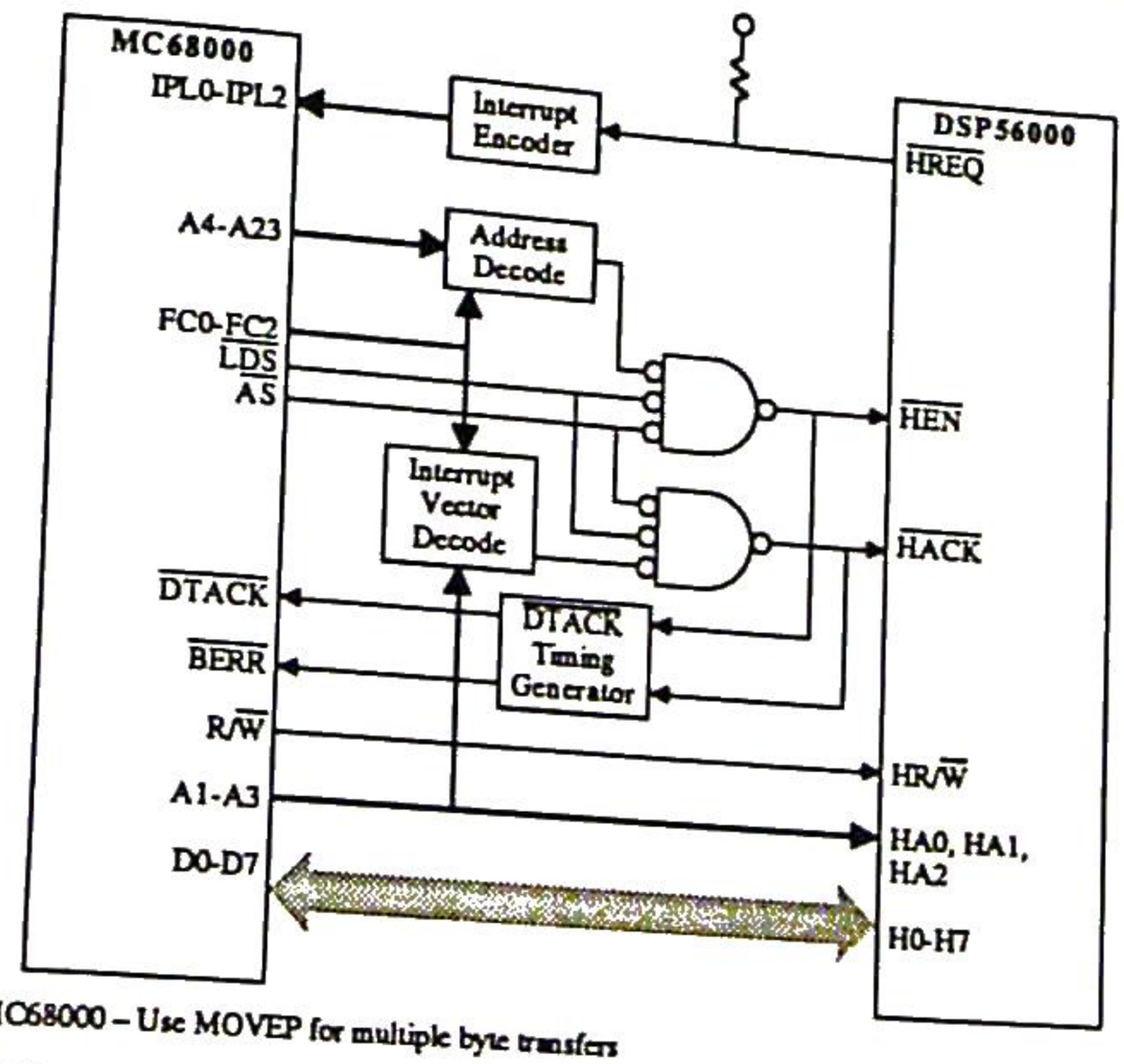
MC68HC11 TO DSP56000 HOST INTERFACE



- Use LDA and STA for 8 Bit Transfers.
- Use LDD and STD for 16 Bit Transfers.

NCD02-1012

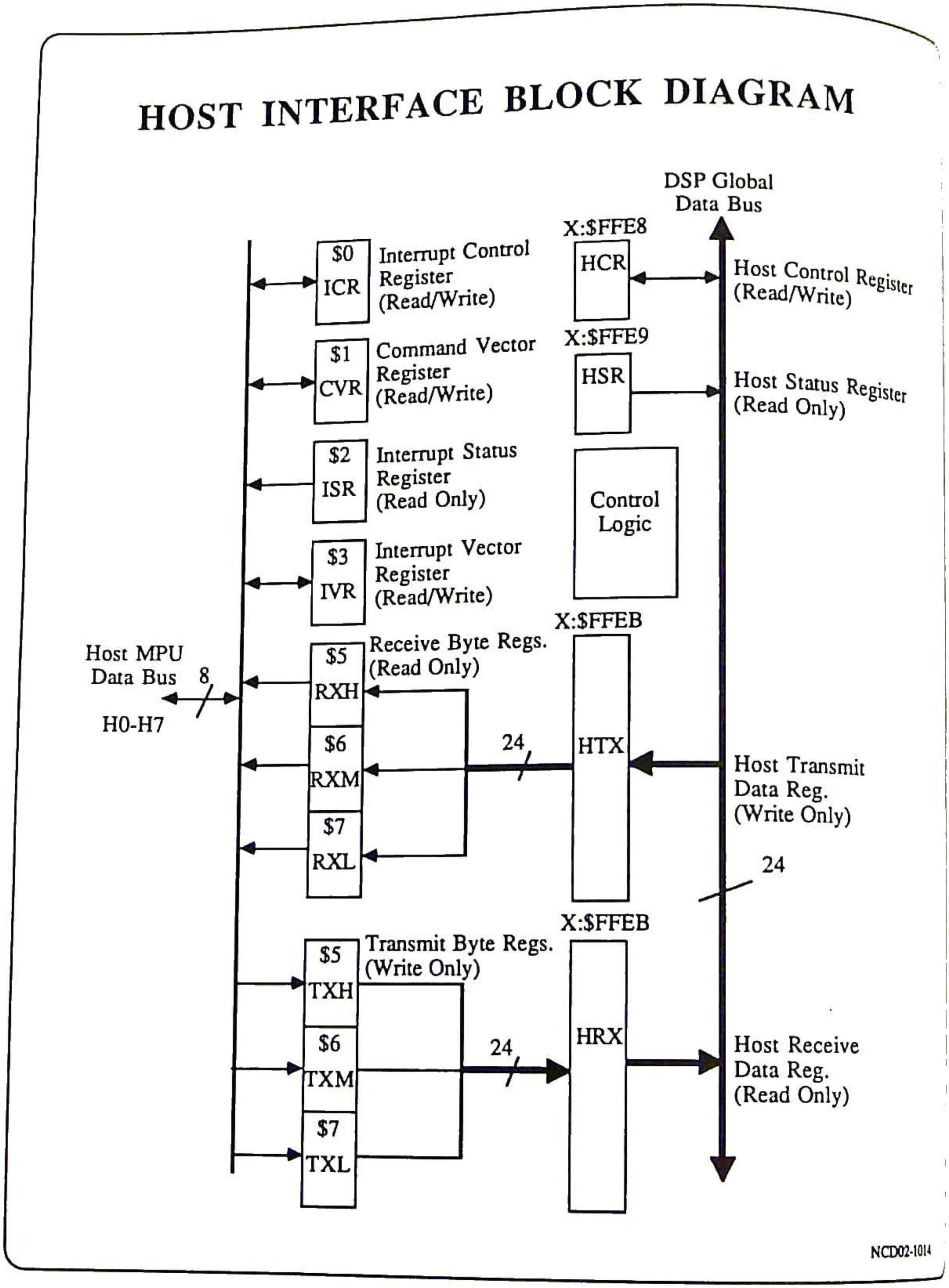
MC68000 TO DSP56000 HOST INTERFACE



- MC68000 – Use MOVEP for multiple byte transfers
- MC68020 or MC68030 – Any Memory reference will work due to dynamic bus sizing

NCD02-1013

HOST INTERFACE BLOCK DIAGRAM



HOST PROCESSOR PROGRAMMING MODEL

	7	Modes		Flags				0
$0	INIT	HM1	HM0	HF1	HF0	0	TREQ	RREQ

Interrupt Control Register (ICR)
(Read/Write)

Reset Value $00

HM1	HM0
0	0 – Interrupt Mode (DMA off)
0	1 – 24 Bit DMA Mode
1	0 – 16 Bit DMA Mode
1	1 – 8 Bit DMA Mode

	7			4 3	0
$1	HC	0	0	Host Vector	

Command Vector Register (CVR)
(Read/Write)

$12

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

Interrupt Status Register (ISR)
(Read Only)

$06

	7	4 3	0
$3	Interrupt Vector Number		

Interrupt Vector Register (IVR)
(Read/Write)

$0F

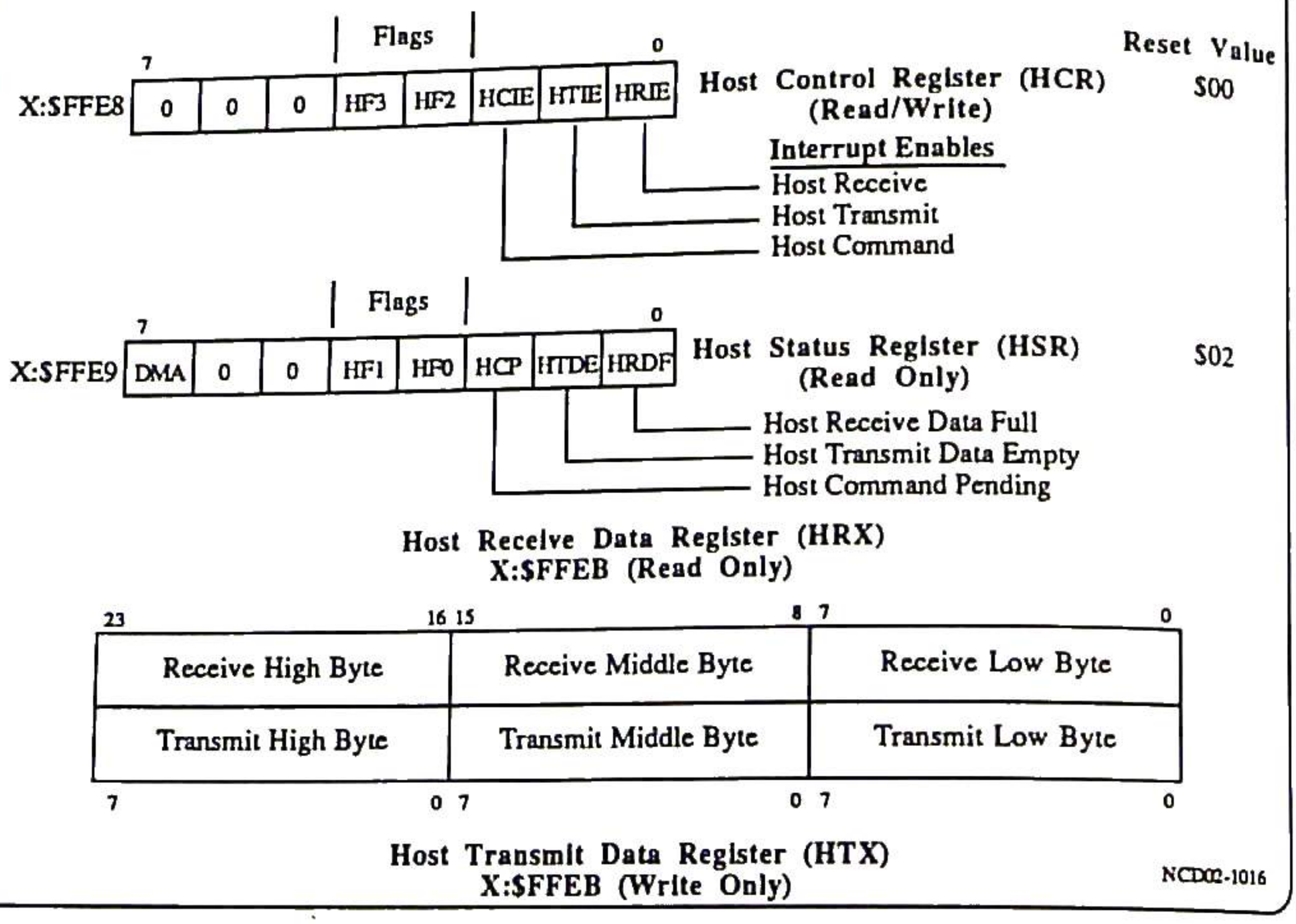
Receive Byte Registers (RXH:RXM:RXL)
(Read Only)

31 $4 24	23 $5 16	15 $6 8	7 $7 0
00000000	RXH Receive High Byte	RXM Receive Middle Byte	RXL Receive Low Byte
Not Used	TXH Transmit High Byte	TXM Transmit Middle Byte	TXL Transmit Low Byte
7 0	7 0	7 0	7 0

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

NCD02-1015

DSP56000 HOST INTERFACE PROGRAMMING MODEL

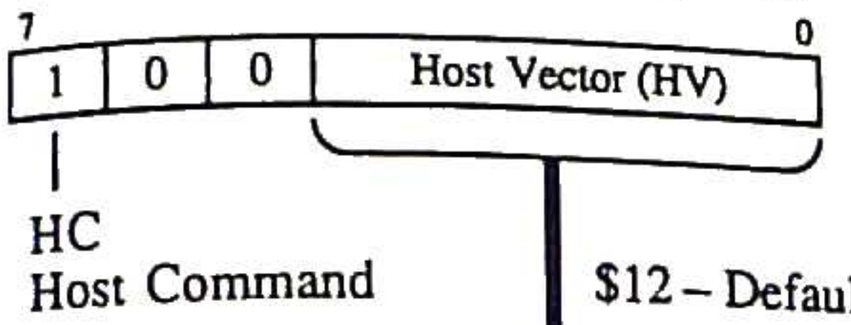


HOST COMMAND

View From Host

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

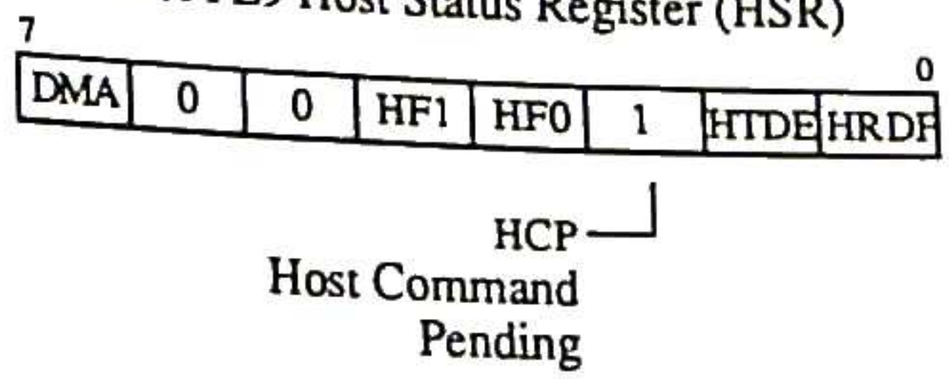
$1 Command Vector Register (CVR)



View From DSP56000

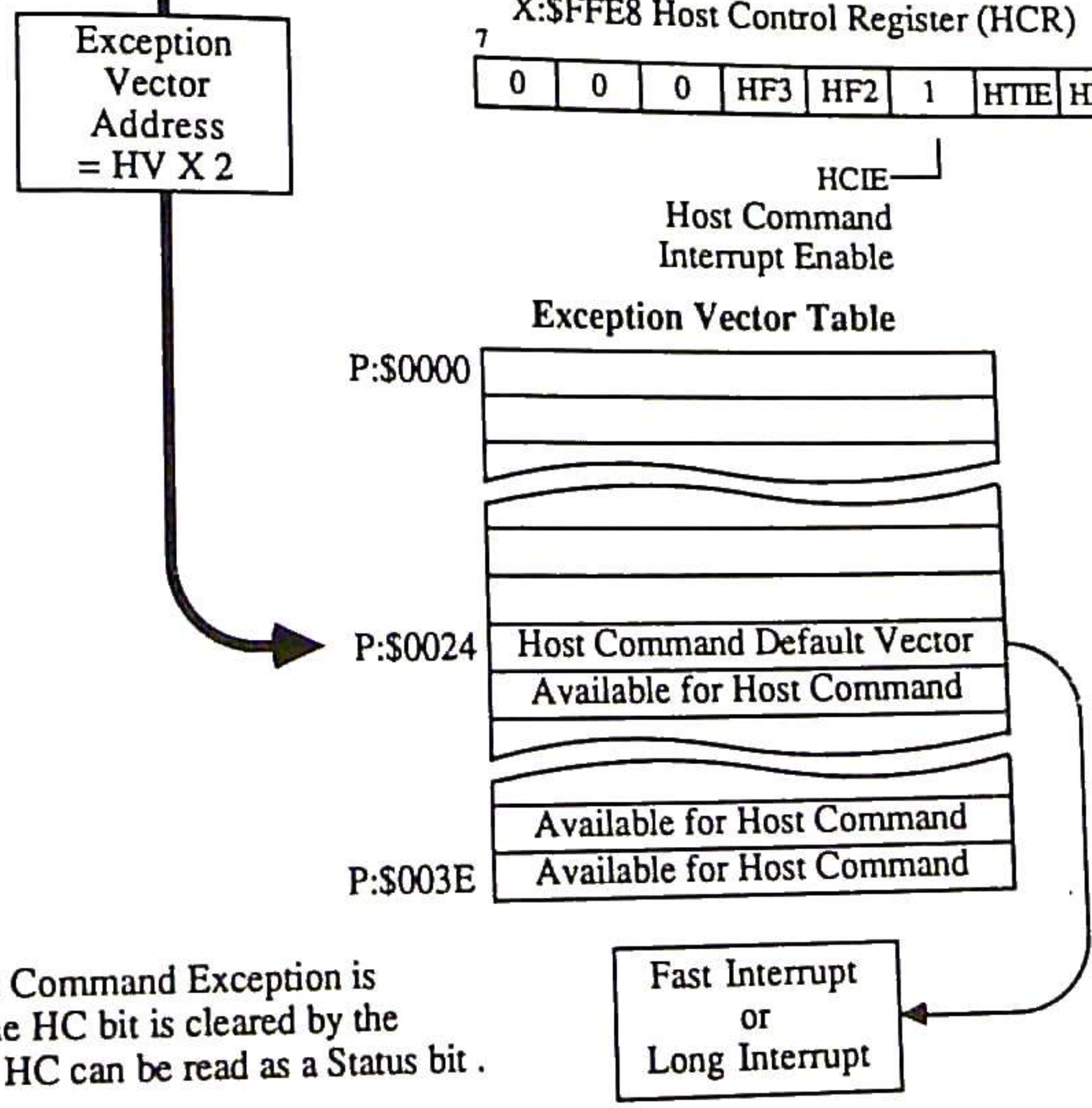
3. HCP is set until Exception Processing starts.

X:$FFE9 Host Status Register (HSR)



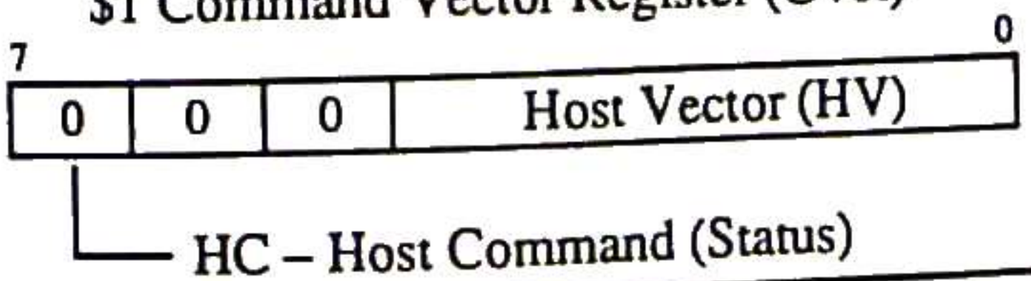
4. Host Command is Masked until HCIE = 1

X:$FFE8 Host Control Register (HCR)



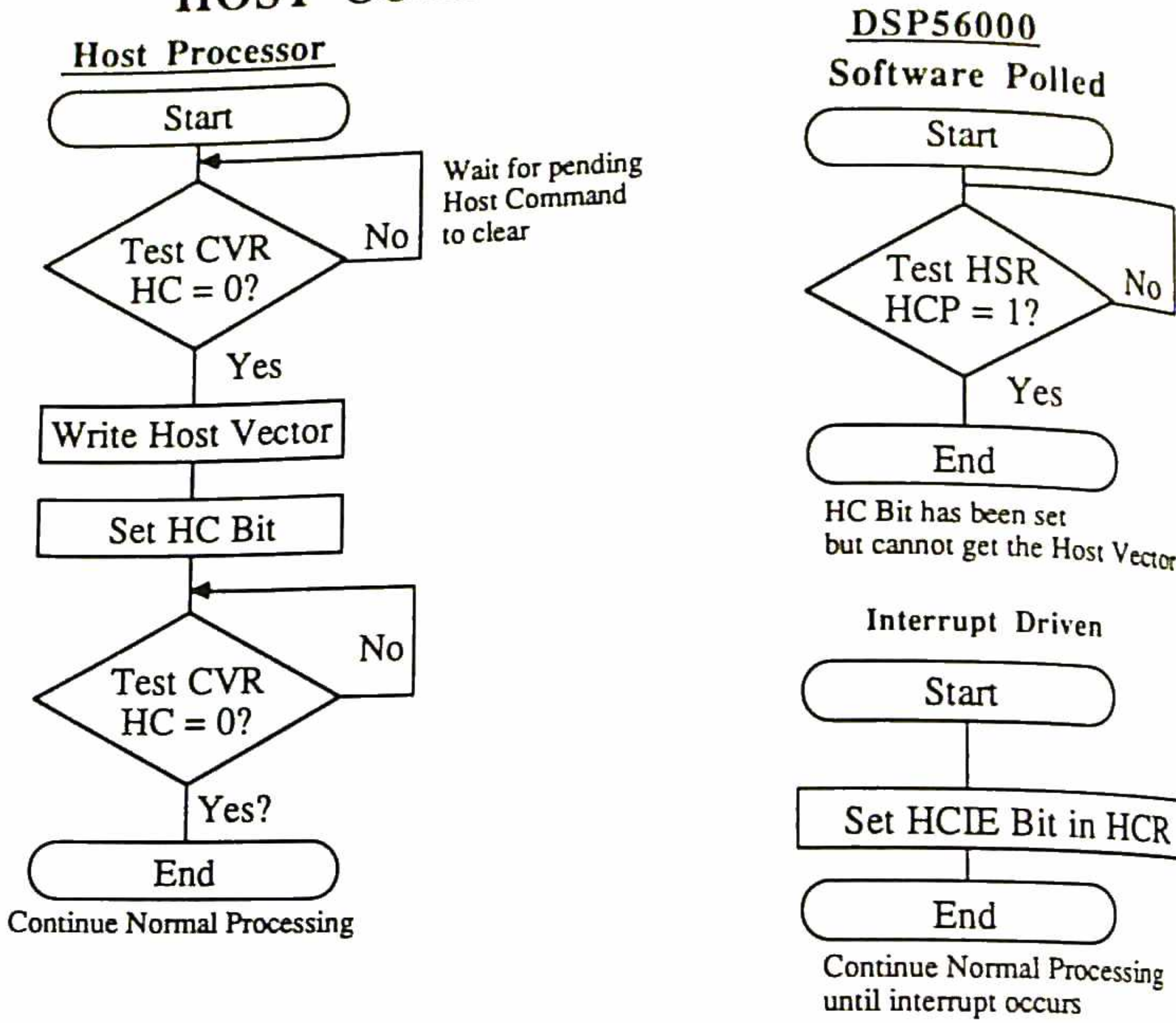
5. When the Host Command Exception is acknowledged, the HC bit is cleared by the Host Command. HC can be read as a Status bit .

$1 Command Vector Register (CVR)



NCD02-1017

HOST COMMAND PROGRAMMING



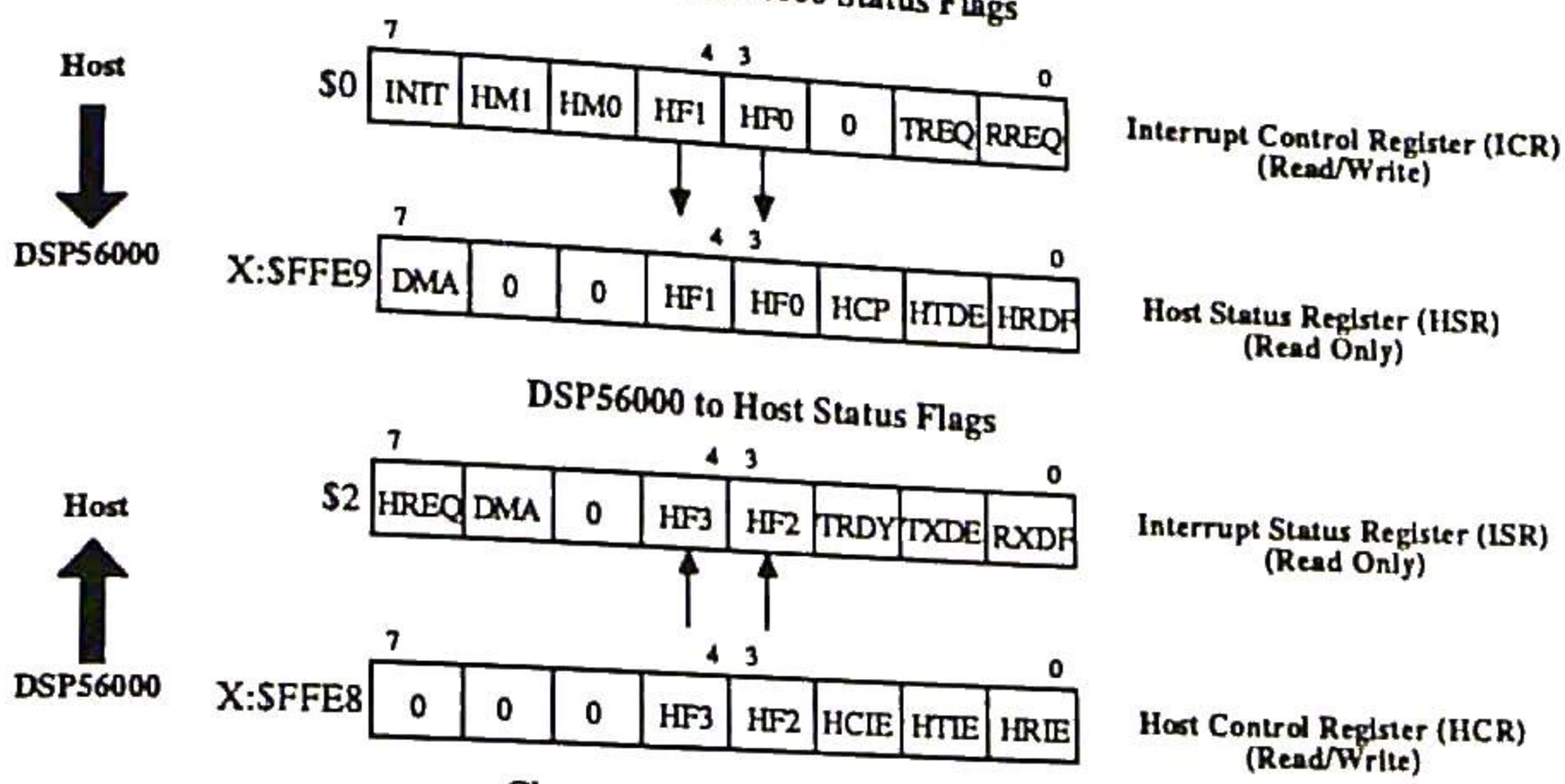
Characteristics of the Host Command

- Writing the HC bit and HV can be done simultaneously in one step
- The Host processor can clear the HC bit to cancel a Host Command at anytime before the DSP56000 exception is recognized.
- The HC bit is cleared by the DSP hardware upon exception recognition and can be read as a status bit.
- To guarantee a stable interrupt vector, write HV only when HC is clear.
- Upon RESET, HV = $12, Exception Vector P:$0024.
- The HV may be programmed to any exception vector.
- It is not recommended that HV = 0 (RESET) be used because it does not reset the DSP hardware.

NCD02-1018-1

HOST FLAGS

Host to DSP56000 Status Flags



Characteristics of Host Flags

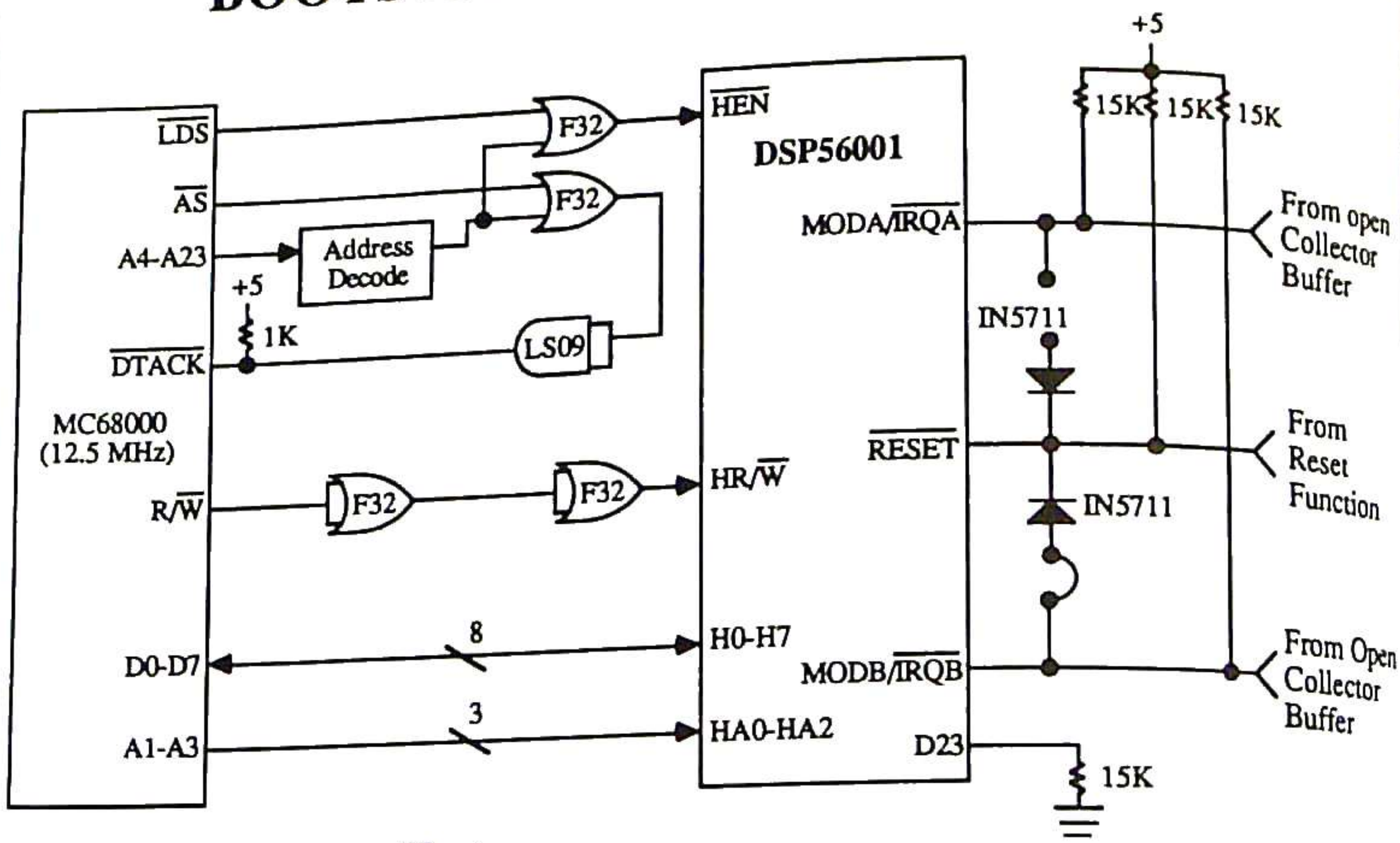
- When reading Host Flags as an encoded pair, a small probability exists that they may be read while in transition.

  There are two solutions to guarantee stability:

  1. Read the bits twice and check for consensus
  2. Assert $\overline{HEN}$ for 2T + 60nS for the Host interface logic to properly latch the bits.

NCD02-1019

BOOTSTRAP USING HOST FLAGS



Host

	7			4	3			0
$0	INIT	HM1	HM0	HF1	HF0	0	TREQ	RREQ

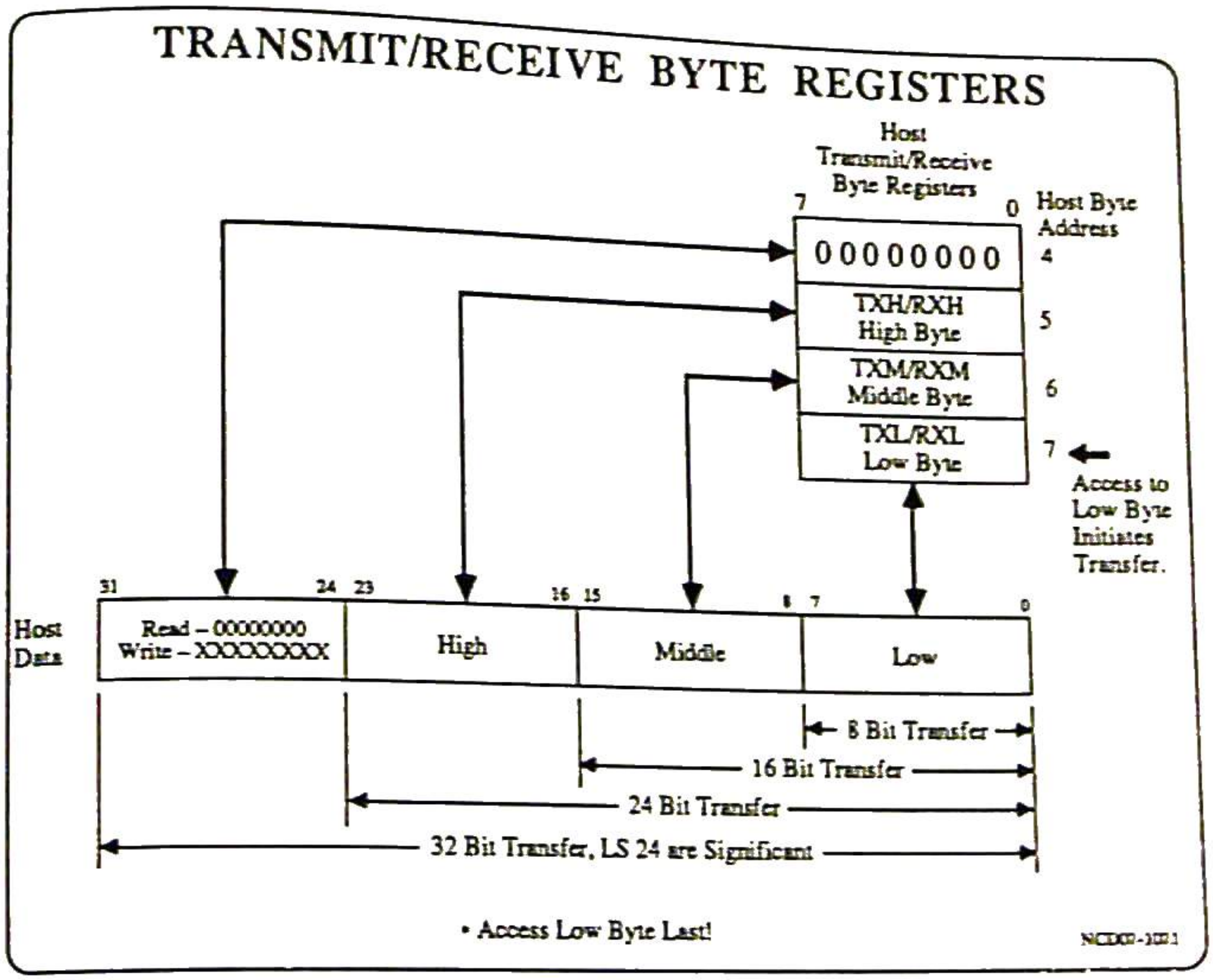
Interrupt Control Register (ICR)
(Read/Write)

Setting HFO terminates bootstrap loading and Starts Execution at location P:$0000.

Host Address Written	Contents Loaded to Internal PRAM at:
4 (Dummy)	
5	P:$0000 High Byte
6	P:$0000 Mid Byte
7	P:$0000 Low Byte
•	•
•	•
Set HF0 for early termination → •	•
4 (Dummy)	
5	P:$01FF High Byte
6	P:$01FF Mid Byte
7	P:$01FF Low Byte

• Because the DSP56000 is so fast, Host Handshaking is not required.

NCD02-1020-1


TRANSMIT/RECEIVE BYTE REGISTERS
Host Transmit/Receive Byte Registers
7
0
Host Byte Address
00000000
4
TXH/RXH High Byte
5
TXM/RXM Middle Byte
6
TXL/RXL Low Byte
7
Access to Low Byte Initiates Transfer.
31
24
23
16
15
8
7
0
Host Data
Read – 00000000
Write – XXXXXXXX
High
Middle
Low
8 Bit Transfer
16 Bit Transfer
24 Bit Transfer
32 Bit Transfer, LS 24 are Significant
• Access Low Byte Last!

BITS USED FOR HOST TO DSP56000 TRANSFER

HOST

Interrupt Status Register (ISR)
(Read Only)

	7							0
$2	HREQ	DMA	0	HF3	HF2	TRDY	TXDE	RXDF

TXDE – Transmit Data Register Empty

= 1 – Indicates the Transmit Byte Registers TXH:TXM:TXL are empty

= 0 – Cleared by writing to TXL

TXDE can be used to assert the $\overline{\text{HREQ}}$ pin.

TRDY – Transmitter Ready = TXDE • $\overline{\text{HRDF}}$

= 1 – Both the Transmit Byte Registers and the Host Receive Data Registers are empty

= 0 – One or both registers are full

Interrupt Control Register (ICR)
(Read/Write)

	7	Modes						0
$0	INIT	HM1	HM0	HF1	HF0	0	TREQ	RREQ

0	0 – Interrupt Mode (DMA off)
0	1 – 24 Bit DMA Mode
1	0 – 16 Bit DMA Mode
1	1 – 8 Bit DMA Mode

TREQ – Transmit Request Enable

Used to enable Interrupts that come from TXDE to the host via the $\overline{\text{HREQ}}$ pin.

= 1 – TXDE Interrupts pass to $\overline{\text{HREQ}}$

= 0 – TXDE Interrupts are masked

DSP56000

Host Status Register (HSR)
(Read Only)

	7							0
X:$FFE9	DMA	0	0	HF1	HF0	HCP	HTDE	HRDF

HRDF – Host Receive Data Full

= 1 – The host receive register (HRX) contains Data from the Host Processor

= 0 – HRX is Empty

DMA – Indicates the Host Processor has enabled the DMA mode

= 1 – DMA on

= 0 – Host Mode

Host Control Register (HCR)
(Read/Write)

	7							0
X:$FFE8	0	0	0	HF3	HF2	HCIE	HTIE	HRIE

HRIE – Host Receive Interrupt Enable

Enables Interrupt at P:$0020

DSP Interrupt is caused by HRDF = 1

= 1 – Interrupt P:$0020 Enabled

= 0 – Interrupt P:$0020 Disabled

DATA TRANSFER FROM HOST TO DSP56000

View from Host

1. When TXDE = 1, TDR 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

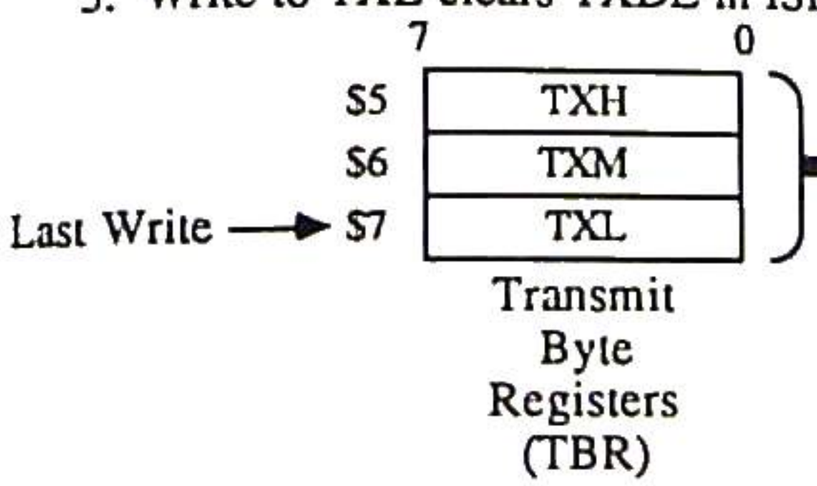
$0 Interrupt Control Register (ICR)

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

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



Transfer

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
0	0	0	HF1	HF0	HCP	HTDE	0
DMA							

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

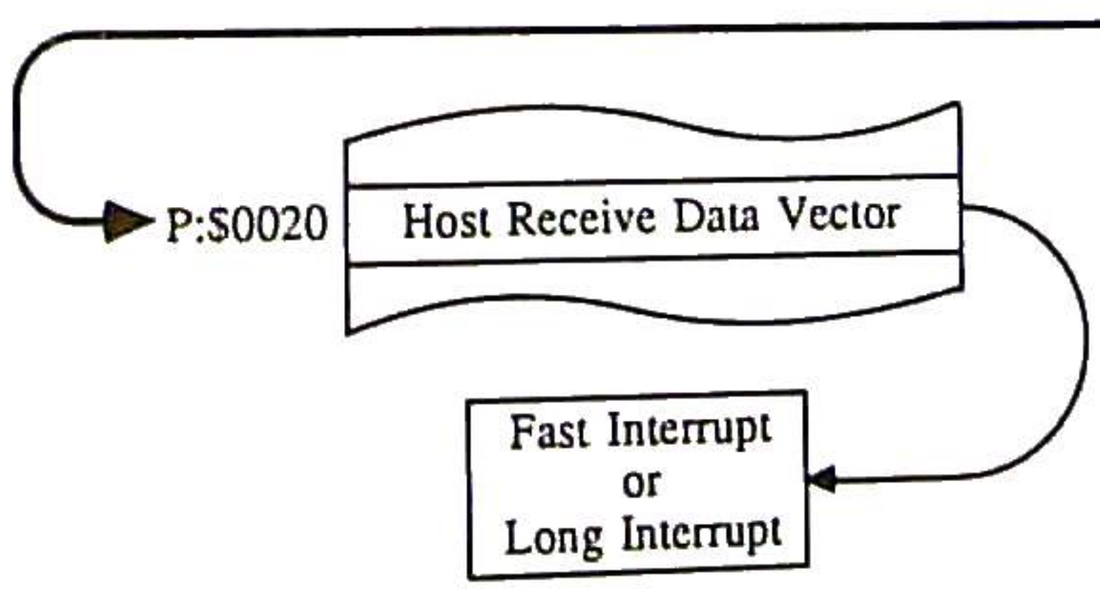
9. The transfer sets HRDF for the DSP56000 to poll

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 enabled then Exception Processing begins



NCD02-1023-1

= 0 – TXDE Interrupts are masked

BITS USED FOR DSP56000 TO HOST TRANSFER

HOST

Interrupt Status Register (ISR)
(Read Only)

	7							0
$2	HREQ	DMA	0	HF3	HF2	TRDY	TXDE	RXDF

RXDF – Receive Data Register Full

= 1 – Indicate that receive byte registers RXH:RXM:RXL contain data from the DSP.

= 0 – Cleared by reading RXL.

Interrupt Control Register (ICR)
(Read/Write)

	7	Modes						0
$0	INIT	HM1	HM0	HF1	HF0	0	TREQ	RREQ

RREQ – Receive Request Enable

used to control the $\overline{\text{HREQ}}$ pin

= 1 – Enable interrupt requests created by RXDF

= 0 – Disable interrupt requests.

DSP56000

Host Status Register (HSR)
(Read Only)

	7							0
X:$FFE9	DMA	0	0	HF1	HF0	HCP	HTDE	HRDF

HTDE – Host Transmit Data Empty

= 1 – HTX is empty and can be written by DSP

= 0 – HTX is full

Host Control Register (HCR)
(Read/Write)

	7							0
X:$FFE8	0	0	0	HF3	HF2	HCIE	HTIE	HRIE

HTIE – Host Transmit Interrupt Enable

= 1 – Enable the DSP interrupt to P:$0022

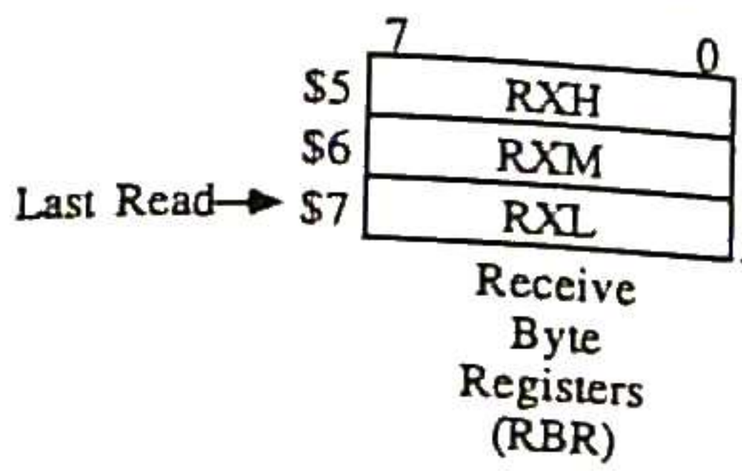
= 0 – Disable the DSP interrupt to P:$0022

DSP Interrupt is caused by HTDE = 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

$2 Interrupt Status Register (ISR)

7							0
HREQ	DMA	0	HF3	HF2	TRDY	TXDE	1

RXDF
Receiver
Data Full

$0 Interrupt Control Register (ICR)

7							0
INIT	HM1	HM0	HF1	HF0	0	TREQ	1

RREQ
Receive
Request Enable

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

X:$FFE9 Host Status Register (HSR)

7							0
DMA	0	0	HF1	HF0	HCP	1	HRDF

HTDE
Host Transmit
Data Empty

2. DSP56000 may Poll HTDE

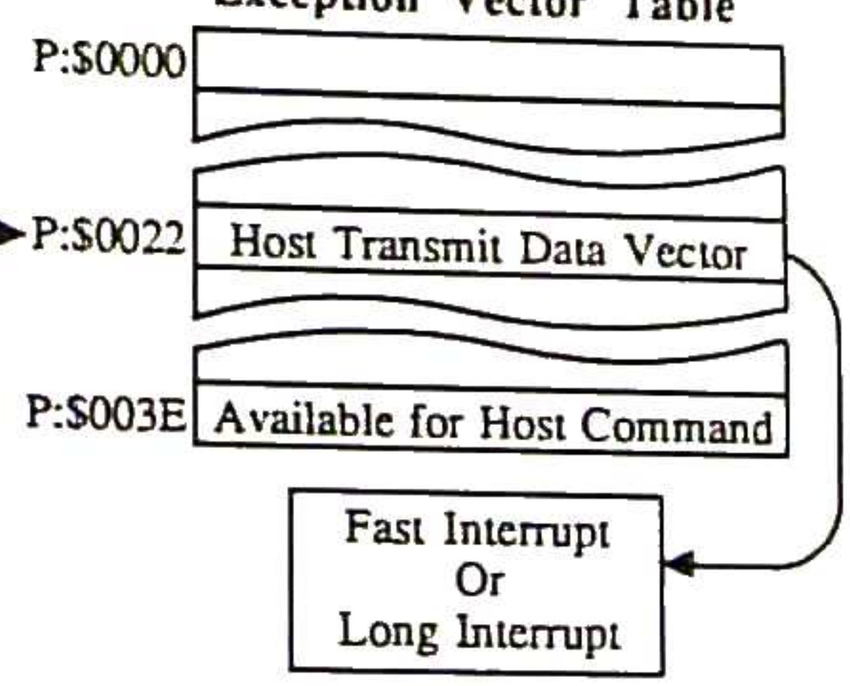
X:$FFE8 Host Control Register (HCR)

7							0
0	0	0	HF3	HF2	HCIE	1	HRIE

HTIE
Host Transmit
Interrupt Enable

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

Exception Vector Table



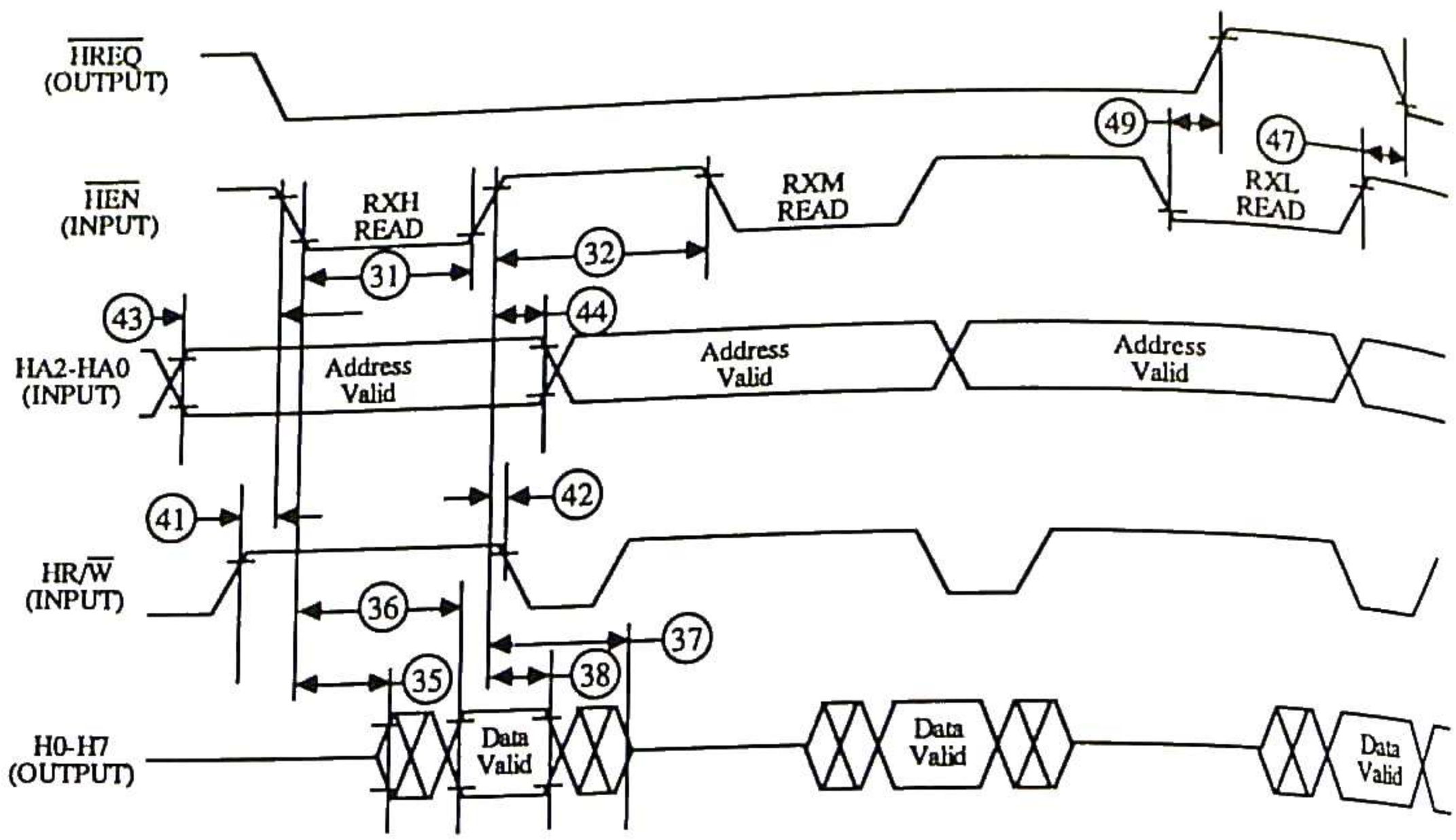
4. DSP56000 Writes Data to HTX. This clears HTDE in HSR

X:$FFEB Host Transmit Data Register (HTX)

23		0
High Byte	Middle Byte	Low Byte

NCD02-1025

HOST READ CYCLE TIMING



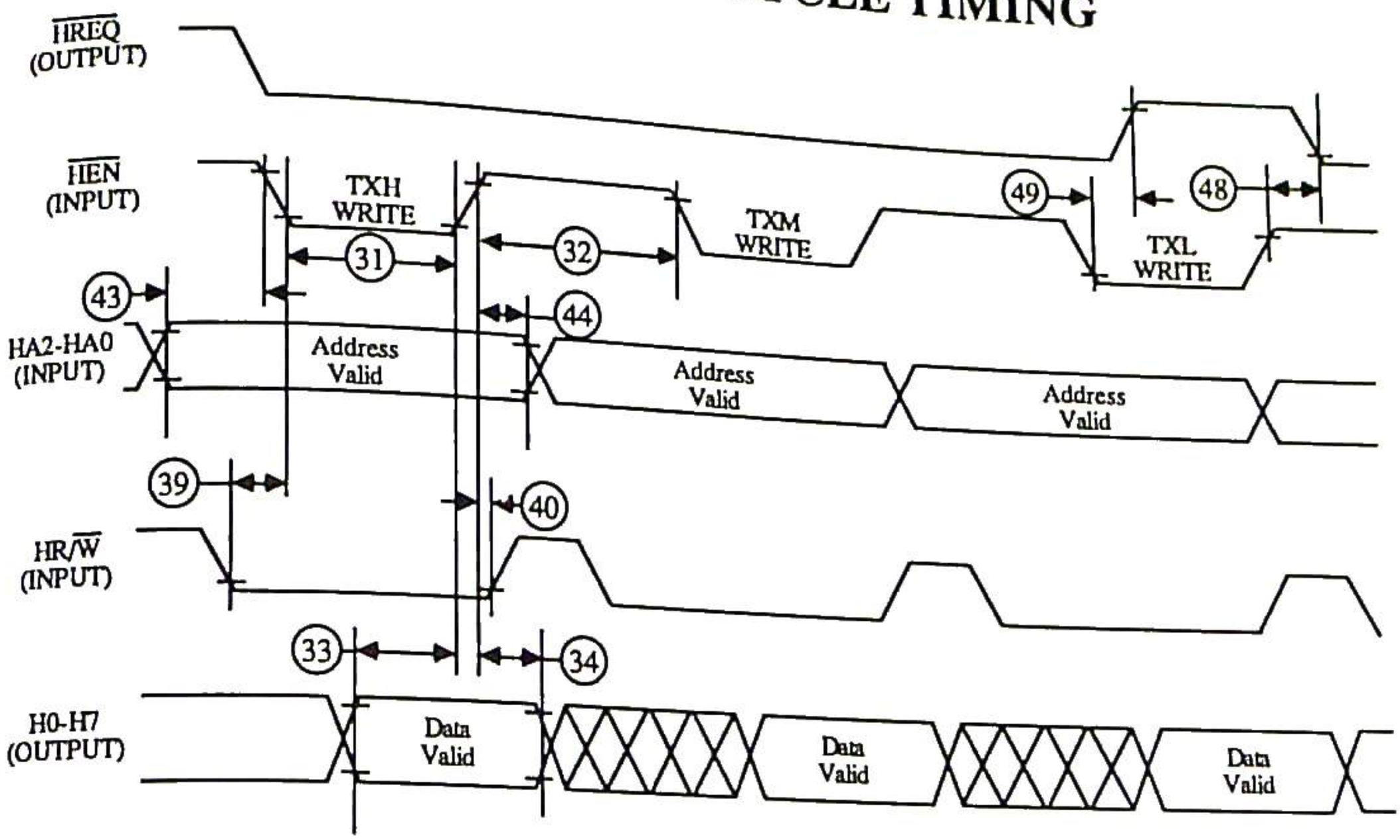
Num.	Characteristic		Min	Max	Unit
31	$\overline{HEN}/\overline{HACK}$ Assertion Width (see note 2)	Read Write	50 25	– –	ns
32	$\overline{HEN}/\overline{HACK}$ Deassertion Width (see note 2)		25	–	ns
35	$\overline{HEN}/\overline{HACK}$ Assertion to Output Data Active from High Impedance		0	–	ns
36	$\overline{HEN}/\overline{HACK}$ Assertion to Output Data Valid		–	50	ns
37	$\overline{HEN}/\overline{HACK}$ Deassertion to Output Data High Impedance		–	35	ns
38	Output Data Hold Time After $\overline{HEN}/\overline{HACK}$ Deassertion		5	–	ns
41	HR/$\overline{W}$ High Setup Time to $\overline{HEN}$ Assertion		0	–	ns
42	HR/$\overline{W}$ High Hold Time After $\overline{HEN}/\overline{HACK}$ Deassertion		5	–	ns
43	HA0-HA2 Setup Time Before $\overline{HEN}$ Assertion		0	–	ns
44	HA0-HA2 Hold Time After $\overline{HEN}$ Deassertion		5	–	ns
47	Delay from $\overline{HEN}$ Deassertion to $\overline{HREQ}$ Assertion for RXL Read (see note 3)		$t_{HSDL} + 3T + 5$	–	ns
49	Delay from $\overline{HEN}$ Assertion to $\overline{HREQ}$ Deassertion for RXL Read, TXL Write (see note 3)		5	75	ns

NOTES:

2. See HOST PORT USAGE CONSIDERATIONS.
3. $\overline{HREQ}$ is pulled up by 1 KΩ.

NCD02-1026

HOST WRITE CYCLE TIMING



Num.	Characteristic		Min	Max	Unit
31	HEN/HACK Assertion Width	Read Write	50 25	–	ns
32	HEN/HACK Deassertion Width		25	–	ns
33	Host Data Input Setup Time Before HEN/HACK Deassertion		5	–	ns
34	Host Data Input Hold Time After HEN/HACK Deassertion		5	–	ns
39	HR/W Low Setup Time Before HEN Assertion		0	–	ns
40	HR/W Low Hold Time After HEN Deassrtion		5	–	ns
43	HA0-HA2 Setup Time Before HEN Assertion		0	–	ns
44	HA0-HA2 Hold Time After HEN Deassertion		5	–	ns
48	Delay from HEN Deassertion to HREQ Assertion for TXL Write (see Note 3)		t_{HSDL} + 2T + 5	–	ns
49	Delay from HEN Assertion to HREQ Deassertion for RXL Read. TXL Write (see Note 3)		5	75	ns

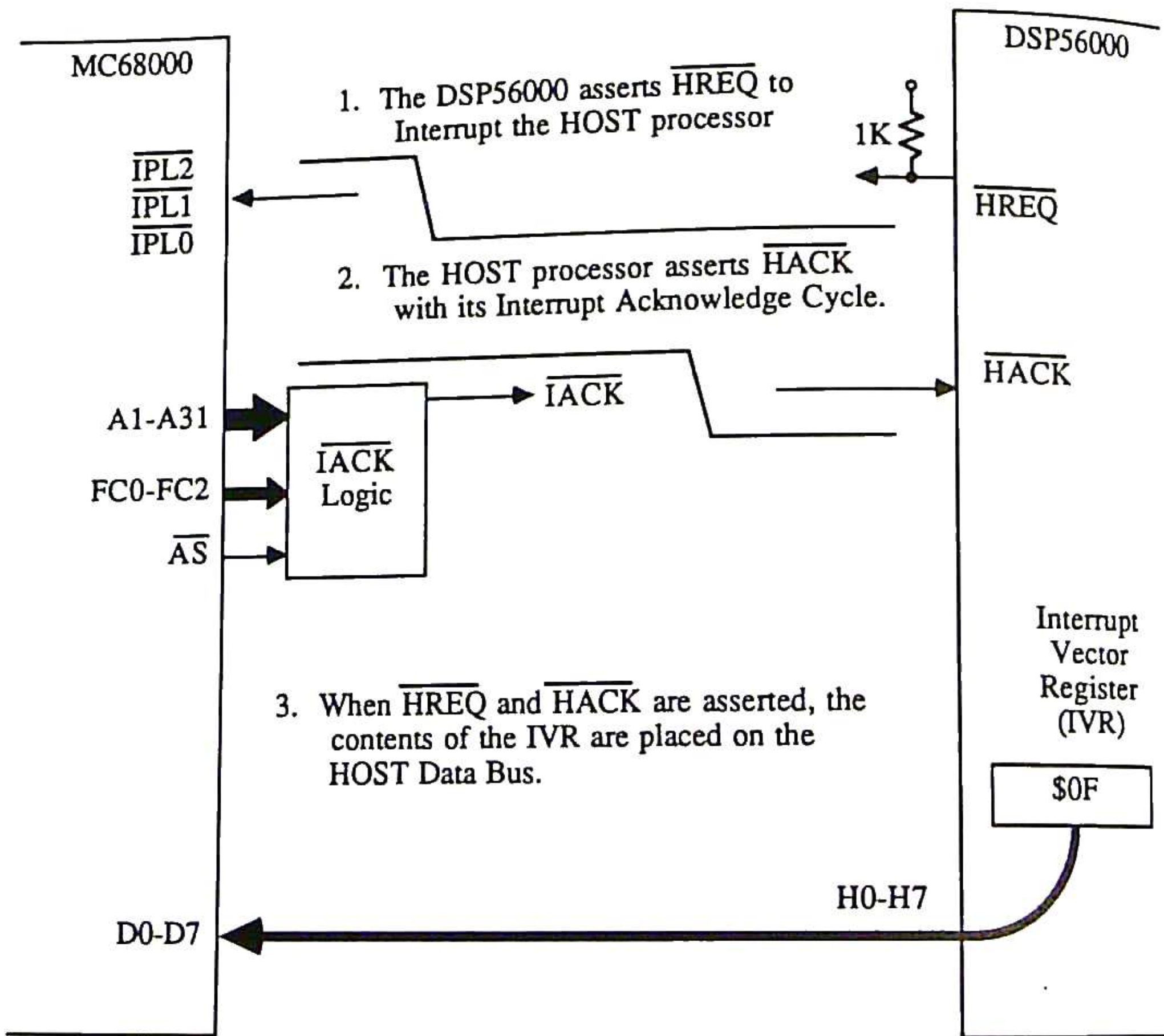
3. HREQ is pulled up by a 1kΩ.

NCD02-1027

HOST EXCEPTION

Interrupt Vector Register (IVR)
(Read/Write)

	7							0
$3	Interrupt Vector Number							



Interrupt Status Register (ISR)
(Read Only)

	7							0
$2	HREQ	DMA	0	HF3	HF2	TRDY	TXDE	RXDF

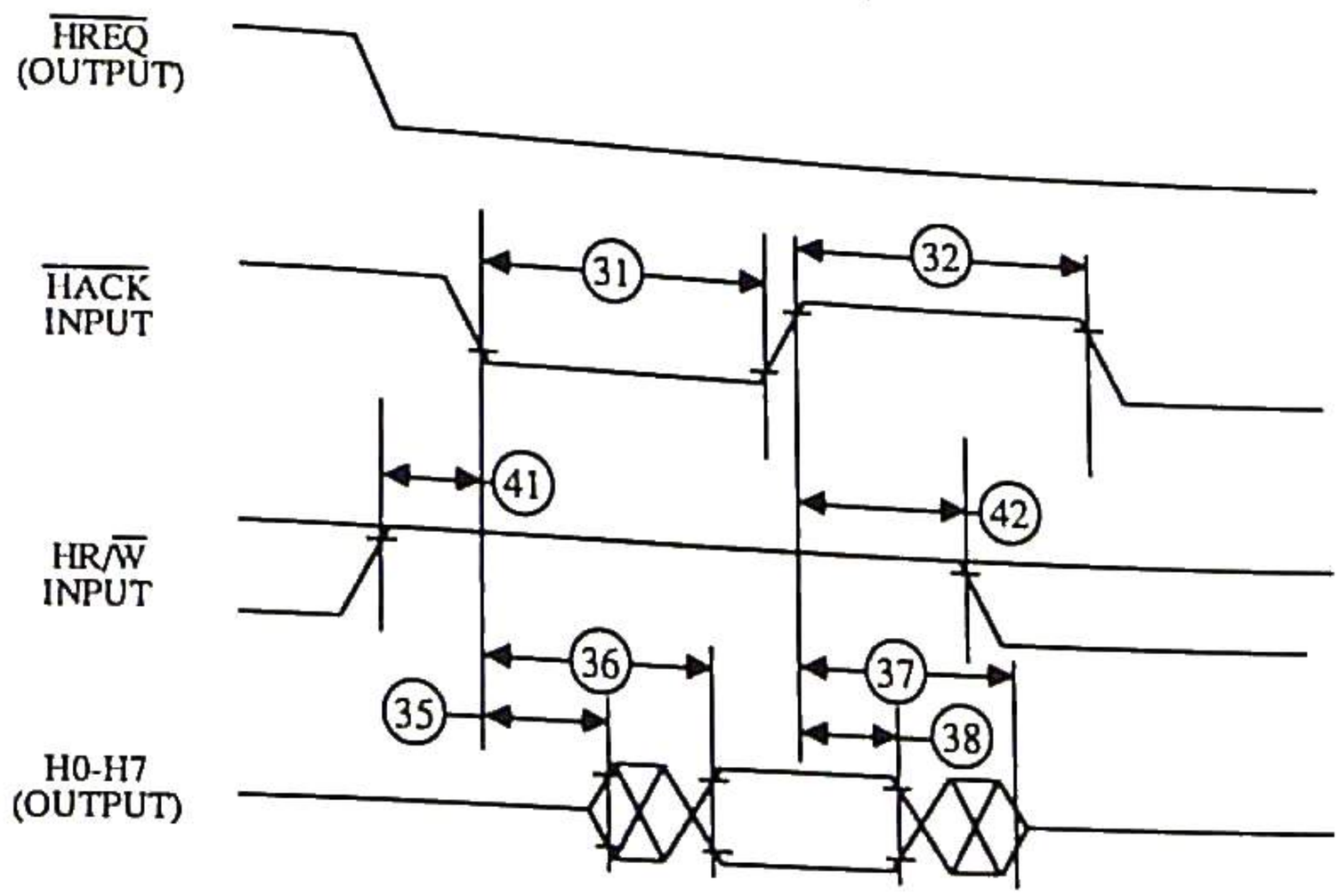
HREQ Status bit reflects the state of the HREQ pin.

= 1 – HREQ is Asserted Interrupt pending

= 0 – HREQ is Negated. No Interrupts.

NCD02-1028

HOST INTERRUPT VECTOR READ



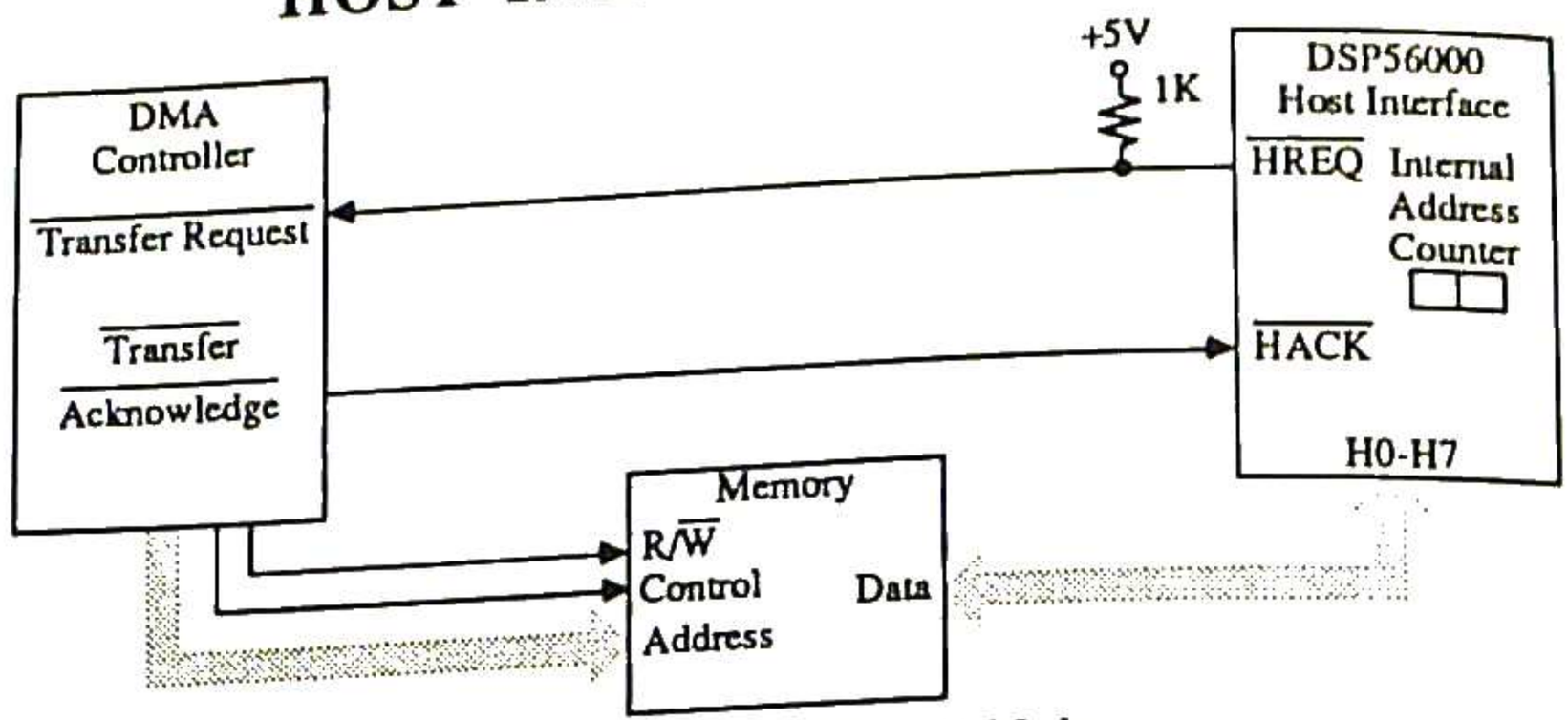
Num.	Characteristic		Min	Max	Unit
31	HEN, HACK Assertion Width	Read Write	50 25	– –	ns
32	HEN, HACK Deassertion Width		25	–	ns
35	HEN, HACK Assertion to Output Data Active from High Impedance		0	--	ns
36	HEN, HACK Assertion to Output Data Valid		–	50	ns
37	HEN, HACK Deassertion to Output Data High Impedance		–	35	ns
38	Output Data Hold Time After HEN, HACK Deassertion		5	–	ns
41	HR/W High Setup Time to HEN Assertion		0	–	ns
42	HR/W High Hold Time After HEN/HACK Deassertion		5	–	ns

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.

NCD02-1029-1

HOST INTERFACE DMA MODE

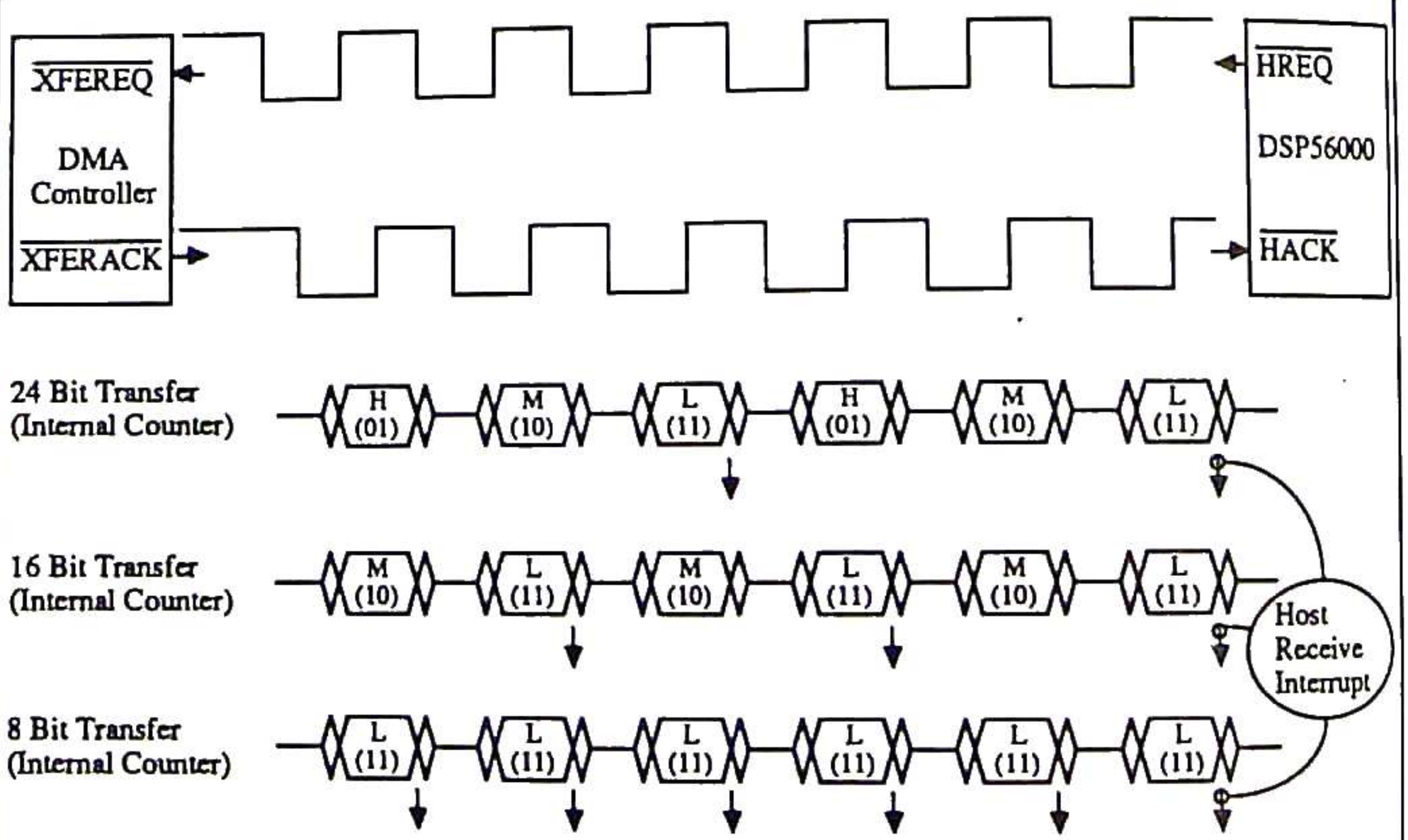


Characteristics of Host DMA Mode

- The HREQ pin is NOT available for Host Processor Interrupts
- TREQ and RREQ select the direction of DMA transfer
  - DMA to DSP56000
  - DSP56000 to DMA
  - Simultaneous Bidirectional DMA transfers are not permitted
- Host processor software polled transfers are permitted in the opposite direction of the DMA transfer
- 8, 16, or 24 bit transfers are supported.
  - 16 or 24 bit transfers reduce the DSP interrupt rate by a factor of 2 or 3 respectively.

NCD02-1030

DMA TRANSFER AND HOST INTERRUPTS

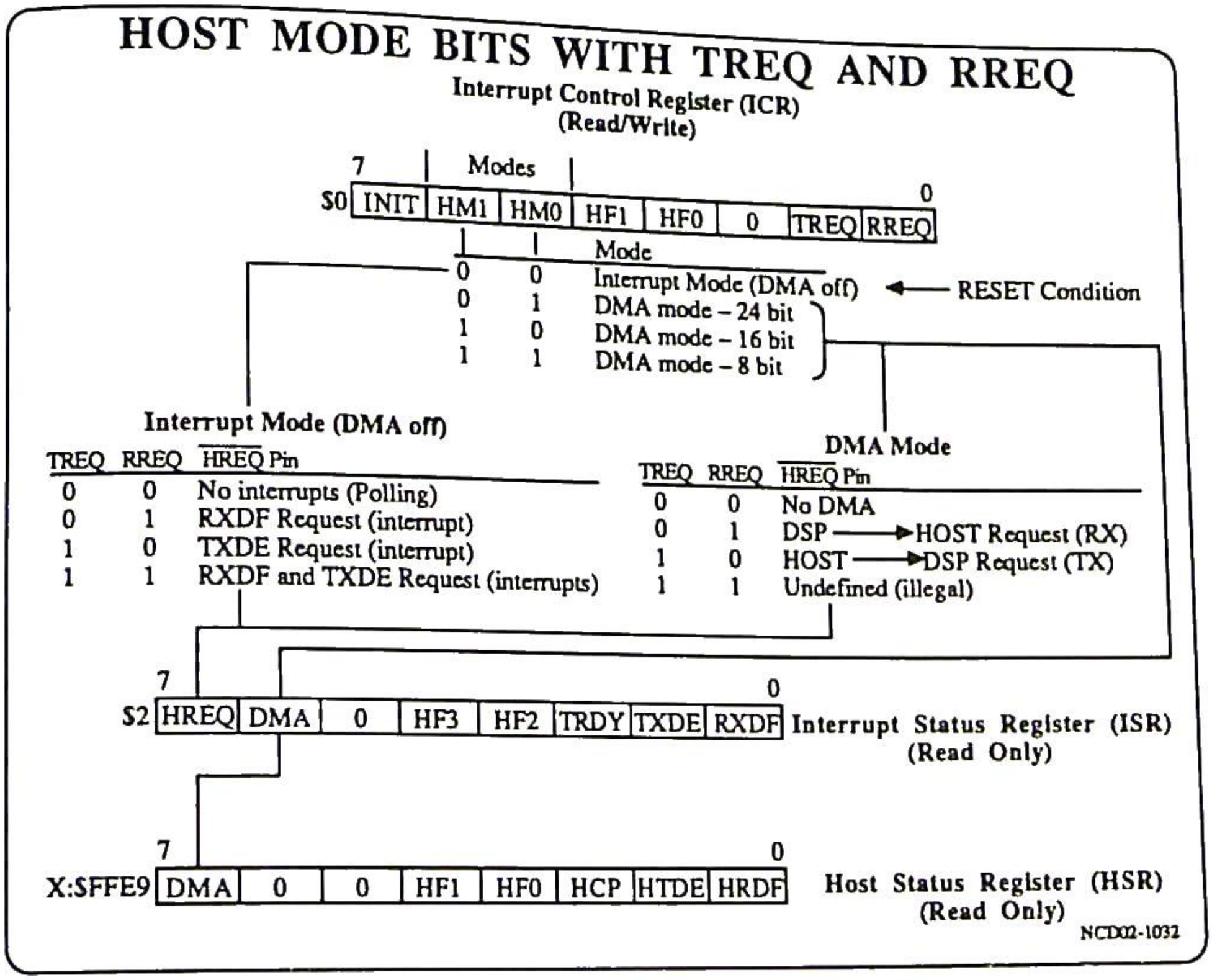


Fast Interrupt Routine

```
P:$0020  MOVE X:$FFE8, A   ; READ HRX
P:$0021  MOVE A, Y:(R7)+   ;AND PUT INTO Y MEMORY
```

NCD02-1031

HOST MODE BITS WITH TREQ AND RREQ



HOST MODE AND INITIALIZE BITS

Interrupt Control Register (ICR)
(Read/Write)

7	Modes						0
INIT	HM1	HM0	HF1	HF0	0	TREQ	RREQ

Host sets INIT bit 1 → INIT

HM1	HM0	Mode
0	0	Interrupt Mode (DMA off) ← RESET Condition
0	1	DMA mode – 24 bit
1	0	DMA mode – 16 bit
1	1	DMA mode – 8 bit

Interrupt Mode (DMA off)

TRBQ	RRBQ	INIT Execution
0	0	INIT=0; "address counter=00"
0	1	INIT=0; RXDF=0; HTDE=1; "address counter=00"
1	0	INIT=0; TXDE=1; HRDF=0; "address counter=00"
1	1	INIT=0; RXDF=0; HTDE=1; TXDE=1; HRDF=0; "address counter=00"

DMA Mode

TRBQ	RRBQ	INIT Execution
0	0	INIT=0; address counter=HM1, HM0
0	1	INIT=0; RXDF=0; HTDE=1; address counter=HM1,HM0
1	0	INIT=0; TXDE=1; HRDF=0; address counter=HM1,HM0
1	1	Undefined (Illegal)

- INIT is used by the HOST to force Initialization of the Host Interface hardware
- The Host Interface hardware automatically clears INIT when the command is complete
- INIT is cleared by DSP RESET.

NCD02-1053

HOST TO DSP56000 DMA PROCEDURE

Host Processor | **DMA Controller** | **DSP56000**

1. Program DMA Controller →
   - Start Address
   - Byte Count
   - Transfer Direction
   - Start DMA Channel

2. Initialize DSP56000 Host Interface → Write ICR →
   - Mode 24 Bit DMA
   - Host → DSP
   - Use INIT Bit to:
     - Set TXDE
     - Clear HRDF
     - Load DMA Counter

S0 Interrupt Control Register (ICR)

7							0
1	0	1	HF1	HF0	0	1	0
INIT	HM1	HM0				TREQ	RREQ

X:$FFE8 Host Control Register (HCR)

7							0
0	0	0	HF3	HF2	HCIE	HTIE	1
							HRIE

3. Tell DSP56000 →
   - Where to store Data
     i.e. Program address register R7
   - Enable Interrupt HRIE
     This can be done with a Host Command

4. Assert $\overline{HREQ}$ to start DMA Transfer

5. Host is free to perform other tasks.
   i.e. DSP to Host Transfer on a Polled Basis

6. DMA Controller performs writes.

01	TXH
10	TXM
11	TXL
01	TXH
10	TXM
11	TXL
⋮	
01	TXH
10	TXM
11	TXL

$\overline{HREQ}$

Exception Vector Table

P:$0000	
P:$0020	Host Receive Data Vector
P:$008E	Available for Host Command

Fast Interrupt or Long Interrupt

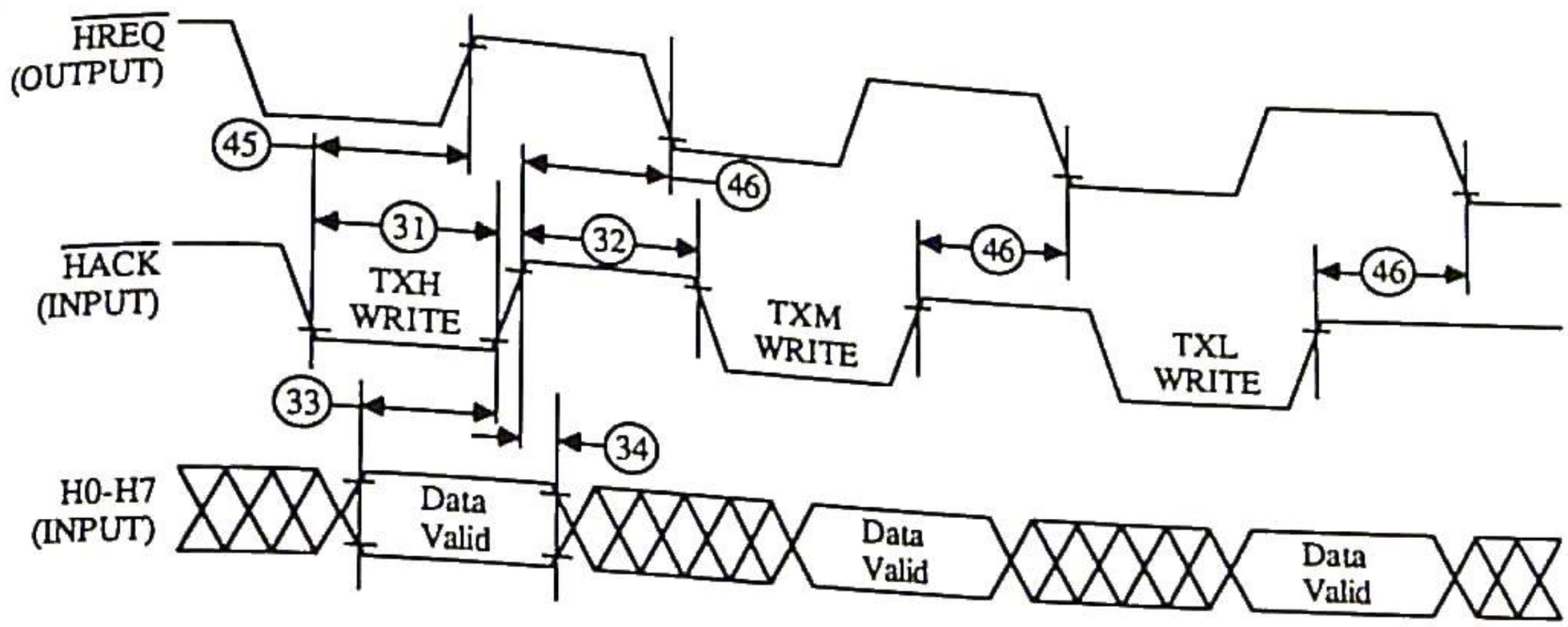
8. Terminate DMA Channel

9. Terminate DSP DMA mode by clearing HM1, HM0, and TREQ.

7. DMA Controller interrupts host when transfers are done.

NCD02-1054-1

HOST DMA WRITE CYCLE TIMING



Num.	Characteristic		Min	Max	Unit
31	$\overline{HEN}/\overline{HACK}$ Assertion Width (see note 2)	Read	50	–	ns
		Write	25	–	ns
32	$\overline{HEN}$, $\overline{HACK}$ Deassertion Width (see note 2)		25	–	ns
33	Host Data Input Setup Time Before $\overline{HEN}$, $\overline{HACK}$ Deassertion		5	–	ns
34	Host Data Input Setup Time Before $\overline{HEN}/\overline{HACK}$ Deassertion		5	–	ns
45	DMA $\overline{HACK}$ Assertion to $\overline{HREQ}$ Deassertion (see note 3)		5	60	ns
46	DMA $\overline{HACK}$ Deassertion to $\overline{HREQ}$ Assertion (see note 3)	for DMA RXL Read	t_{HSDL} + 3T + 5	–	ns
		for DMA TXL Write	t_{HSDL} + 2T + 5	–	
		for All Other Cases	5	–	

NOTES:

1. "Host synchronous delay(t_{HDSL})" is the time required for the DSP56000/56001 to sample any external asynchronous input signal, determine whether it is high or low, and synchronize it to the DSP56000/56001 internal clock.

2. See HOST PORT USAGE CONSIDERATIONS.

3. $\overline{HREQ}$ is pulled up by 1 kΩ.

NCD02-1035

DSP56000 TO HOST DMA PROCEDURE

Host Processor | **DMA Controller** | **DSP56000**

1. Program DMA Controller →
   - Start Address
   - Byte Count
   - Transfer Direction
   - Start DMA Channel

2. Initialize DSP56000 Host Interface —Write ICR→
   - Mode 24 Bit DMA
   - DSP → Host
   - Use INIT Bit to:
     - Set RXDF
     - Clear HTDE
     - Load DMA Counter

$0 Interrupt Control Register (ICR)

7							0
1	0	1	HF1	HF0	0	0	1
INIT	HM1	HM0				TREQ	RREQ

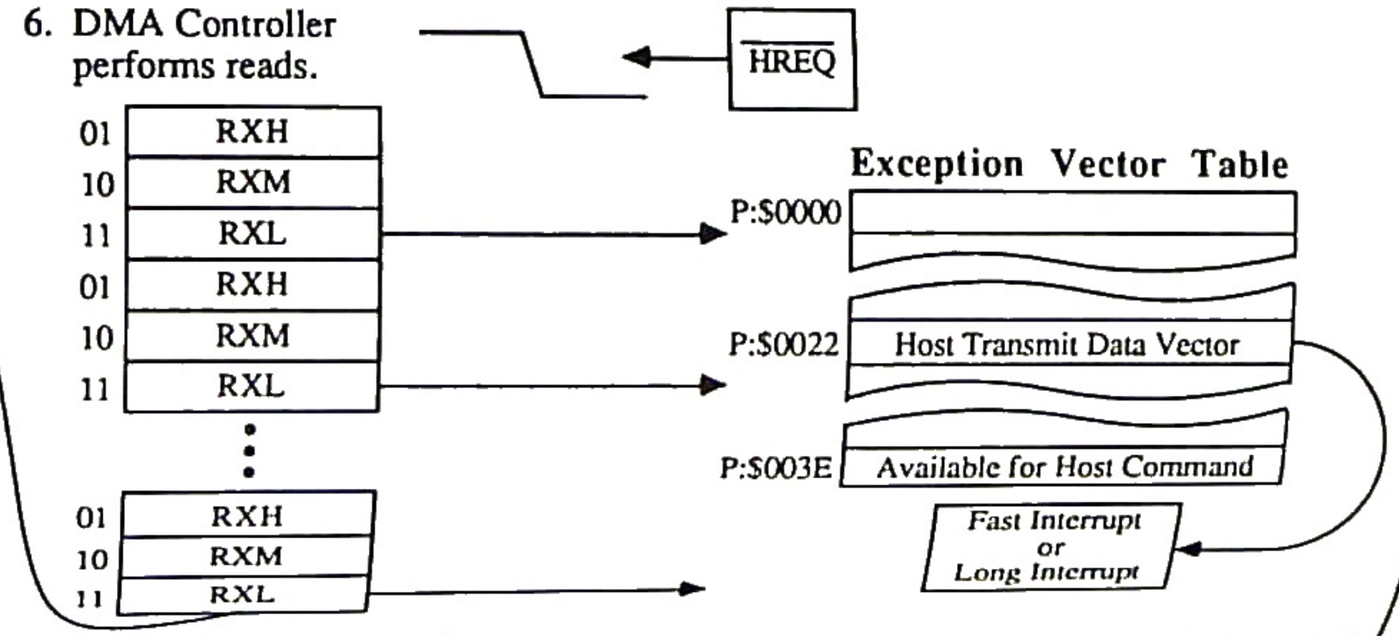
3. Tell DSP56000 →
   - Source pointer address
   - Enable HTIE

   This can be done with a Host Command

X:$FFE8 Host Control REgister (HCR)

7							0
0	0	0	HF3	HF2	HCIE	1	HRIE
						HTIE	

4. Assert $\overline{\text{HREQ}}$ to start DMA Transfer



6. DMA Controller performs reads.

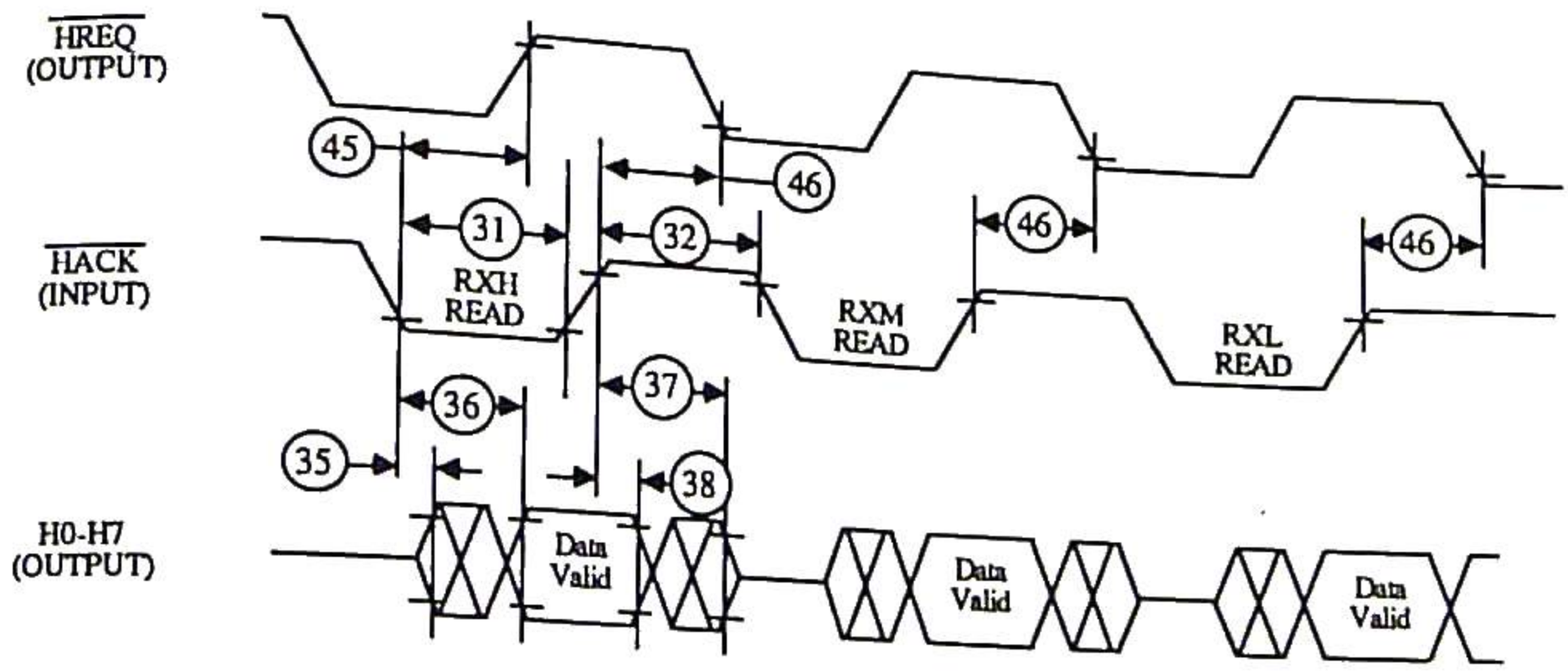
5. Host is free to perform other tasks.
   i.e. Host to DSP Transfer on a Polled Basis

8. Terminate DMA Channel

9. Terminate DSP DMA mode by clearing HM1, HM0, and TREQ.

7. DMA Controller interrupts host when transfers are done.

HOST DMA READ CYCLE TIMING



Num.	Characteristic		Min	Max	Unit
31	$\overline{HEN}$, $\overline{HACK}$ Assertion Width (see note 2)	Read Write	50 25	– –	ns
32	$\overline{HEN}$/$\overline{HACK}$ Deassertion Width (see note 2)		25	–	ns
35	$\overline{HEN}$/$\overline{HACK}$ Assertion to Output Data Active from High Impedance		0	–	ns
36	$\overline{HEN}$/$\overline{HACK}$ Assertion to Output Data Valid		–	50	ns
37	$\overline{HEN}$/$\overline{HACK}$ Deassertion to Output Data High Impedance		–	35	ns
38	Output Data Hold Time After $\overline{HEN}$/$\overline{HACK}$ Deassertion		5	–	ns
45	DMA $\overline{HACK}$ Assertion to $\overline{HREQ}$ Deassertion (see note 3)		5	60	ns
46	DMA $\overline{HACK}$ Deassertion to $\overline{HREQ}$ Assertion (see note 3)	for DMA RXL Read for DMA TXL Write for All Other Cases	t_{HSDL} + 3T + 5 t_{HSDL} + 2T + 5 5	– – –	ns

NOTES:
2. See HOST PORT USAGE CONSIDERATIONS.
3. $\overline{HREQ}$ is pulled up by 1 kΩ.

NCD02-1037

SYNCHRONOUS SERIAL INTERFACE FEATURES

- 5 Megabit per Second Serial Interface
- 6 Pin Interface
- Interface to a wide variety 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

NCD02-1062

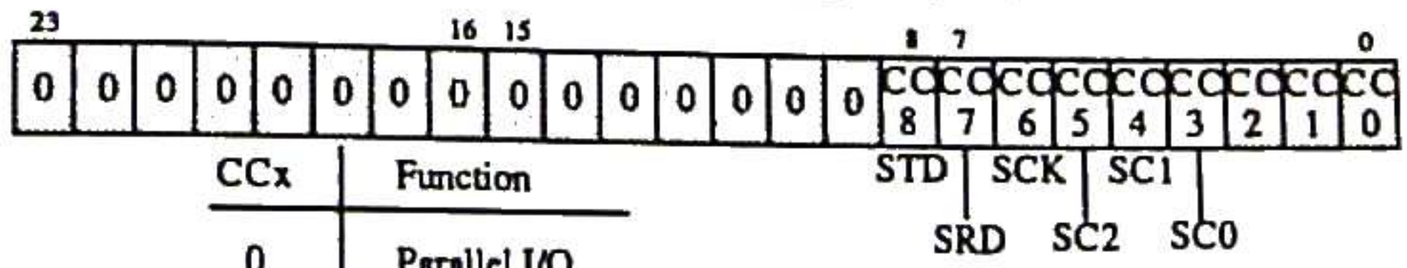
SSI OPERATING MODES

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)
Asynchronous	Continuous	Network	TDM networks
Asynchronous	Gated	On Demand	Parallel to serial and serial to parallel conversion
Synchronous	Continuous	Normal	multiple synchronous codecs
Synchronous	Gated	Normal	SPI-type devices DSP-to-MCU
Synchronous	Continuous	Network	TDM codec network TDM DSP networks
Synchronous	Gated	On Demand	DSP to SPI peripherals

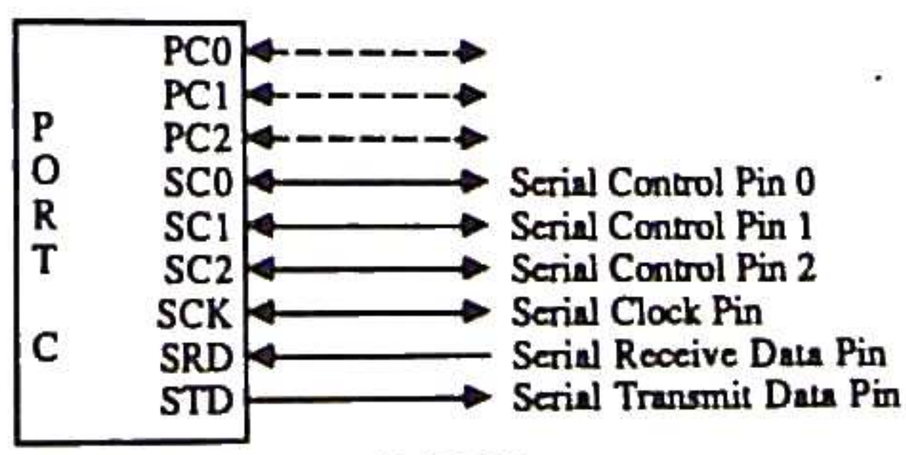
NCD02-1063-1

SSI INITIALIZATION

X:$FFE1 Port 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 at least one SSI PCC Bit as Serial Interface

- Clearing PCC8-PCC3 puts the SSI in the RESET state but does not change the control registers.
- The initialization sequence must be used when changing the following bits MOD, GCK,SYN, SCKD, SCD2, SCD1, SCD0.

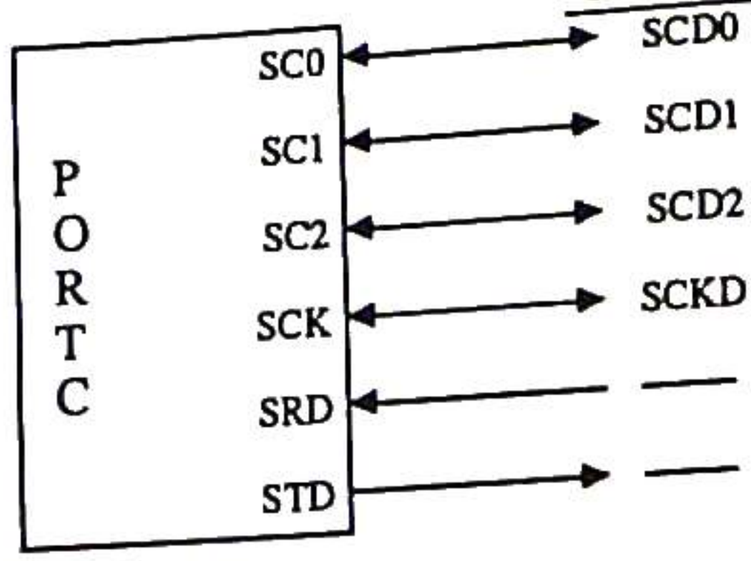
NCD02-1064-1

SERIAL CONTROL DIRECTION BITS

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

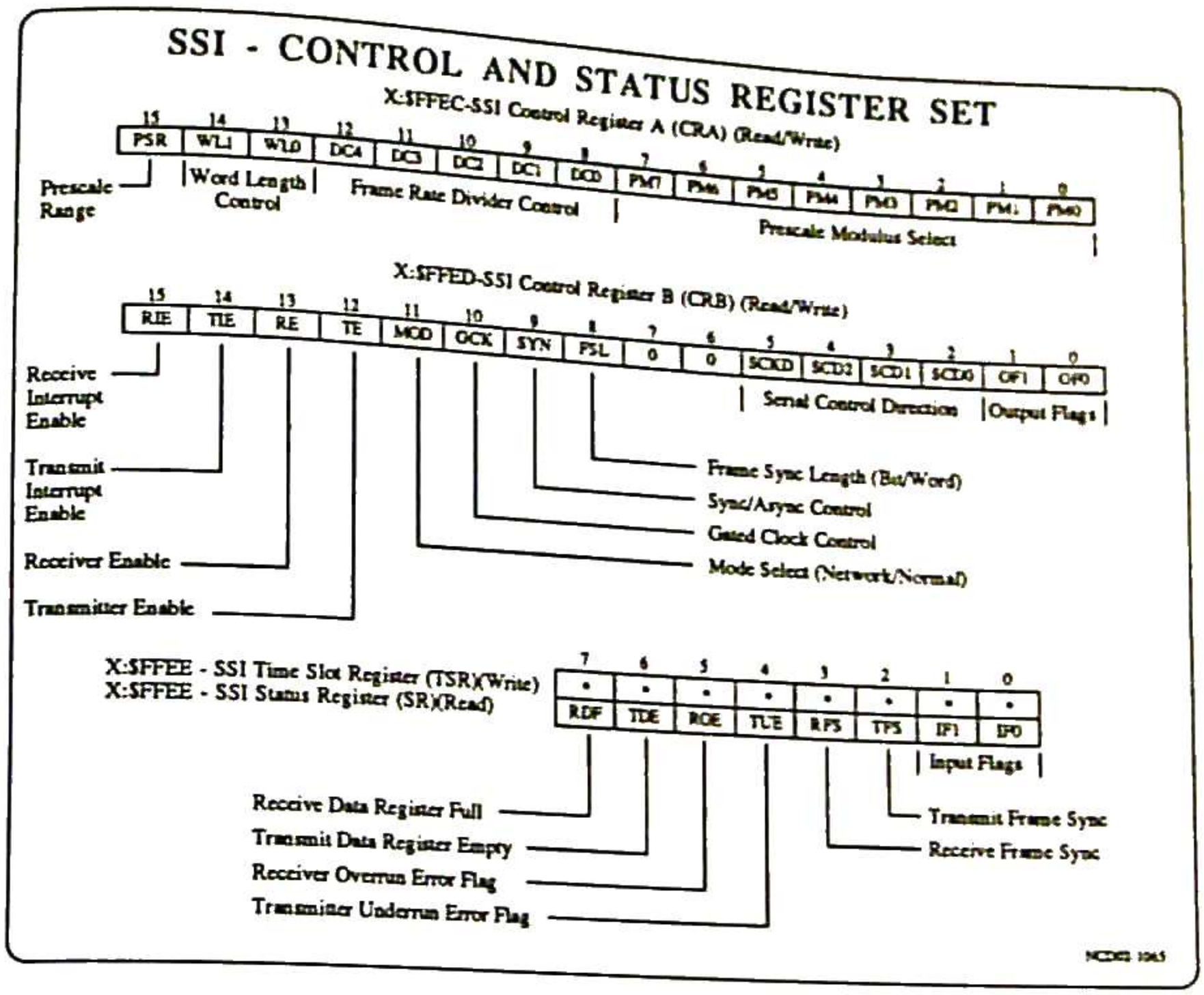
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RIE	TIE	RE	TE	MOD	GCK	SYN	FSL	0	0	SCKD	SCD2	SCD1	SCD0	OF1	OF0

1 = Output
0 = Input

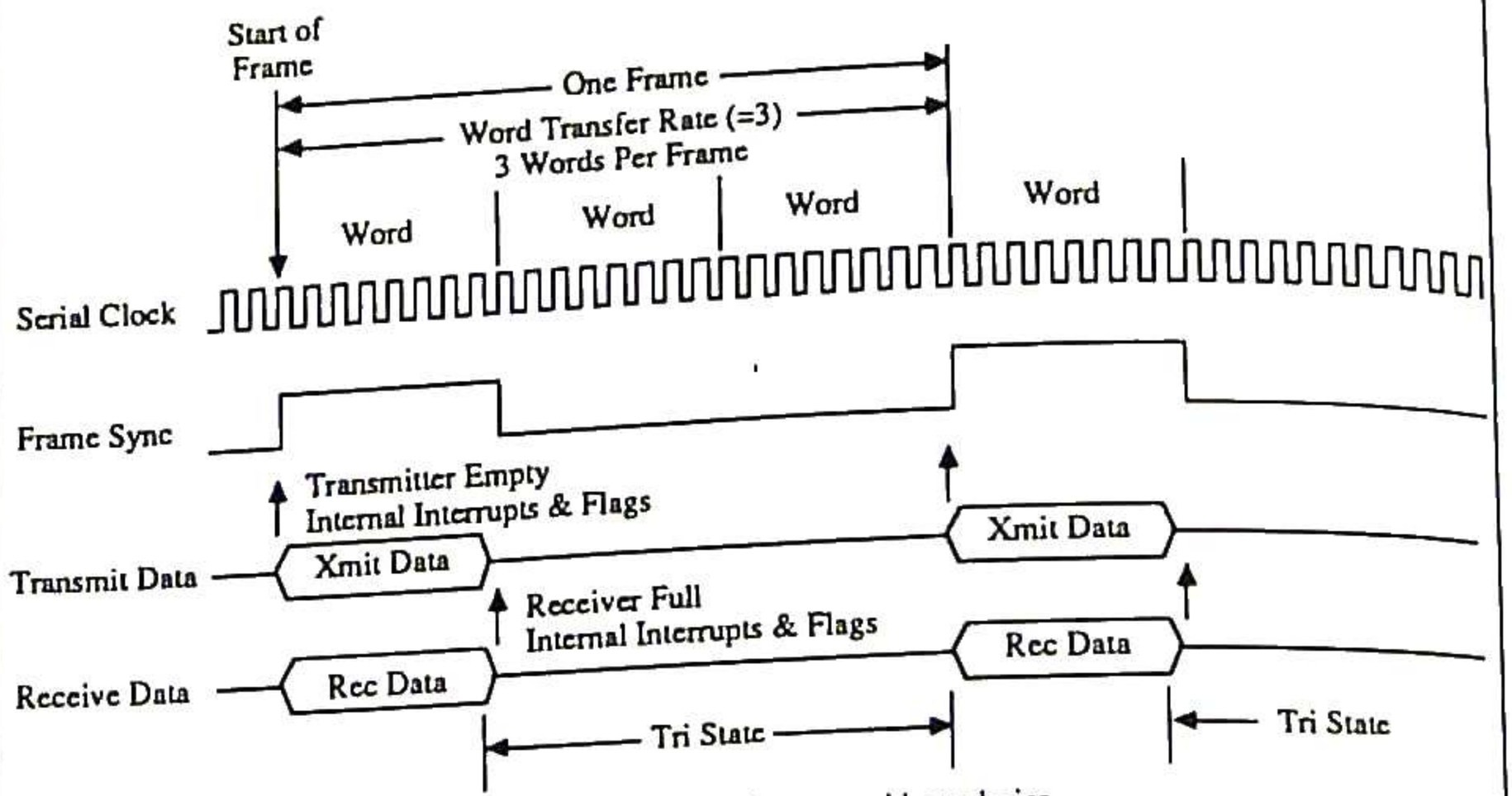


Direction Controlled by	Basic Function
SCD0	Receive Clock/Flag 0
SCD1	Receive Frame Sync/Flag 1
SCD2	Transmit Frame Sync /Tx and Rx Frame Sync
SCKD	Transmit Clock/Tx and Rx Clock
—	SSI Receive Data
—	SSI Transmit Data

NCD02-1067


SSI - CONTROL AND STATUS REGISTER SET
X:$FFEC-SSI Control Register A (CRA) (Read/Write)
PSR
WL1
WL0
DC4
DC3
DC2
DC1
DC0
PM7
PM6
PM5
PM4
PM3
PM2
PM1
PM0
Prescale Range
Word Length Control
Frame Rate Divider Control
Prescale Modulus Select
X:$FFED-SSI Control Register B (CRB) (Read/Write)
RIE
TIE
RE
TE
MOD
GCK
SYN
FSL
0
0
SCKD
SCD2
SCD1
SCD0
OF1
OF0
Receive Interrupt Enable
Transmit Interrupt Enable
Receiver Enable
Transmitter Enable
Serial Control Direction
Output Flags
Frame Sync Length (Bit/Word)
Sync/Async Control
Gated Clock Control
Mode Select (Network/Normal)
X:$FFEE - SSI Time Slot Register (TSR)(Write)
X:$FFEE - SSI Status Register (SR)(Read)
RDF
TDE
ROE
TUE
RFS
TFS
IF1
IF0
Input Flags
Receive Data Register Full
Transmit Data Register Empty
Receiver Overrun Error Flag
Transmitter Underrun Error Flag
Transmit Frame Sync
Receive Frame Sync
SSI TRANSMIT AND RECEIVE REGISTER SET
Serial Receive Data Register (RX)
X:$FFEF (Read Only)
Receive High Byte
Receive Middle Byte
Receive Low Byte
SRD
Serial Receive Shift Register
Serial Transmit Data Register (TX)
X:$FFEF (Write Only)
Transmit High Byte
Transmit Middle Byte
Transmit Low Byte
STD
Serial Transmit Shift Register
MSB
LSB
8-Bit Data
12-Bit Data
16-Bit Data
24-Bit Data
Least Significant Zero Fill
• Data is sent MSB first.
• Compatible with fractional format.

SYNCHRONOUS COMMUNICATION



- The SSI must transmit and receive simultaneously on word boundaries.
- The Transmit/Receive word transfer rate or "Duty Cycle" is set by programming DC4-DC0
- If the transmitter is not loaded before the next word transfer time then Transmitter Underrun occurs.
- If the receiver is not read before the next word transfer time then Receiver Overrun occurs.

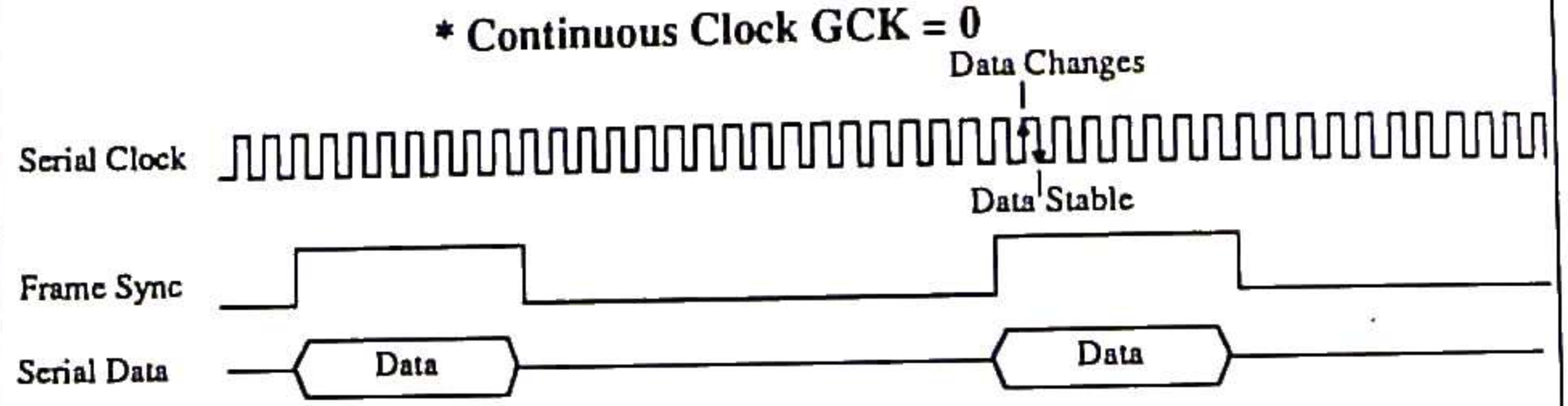
NCD02-1068-1

CLOCK MODES

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

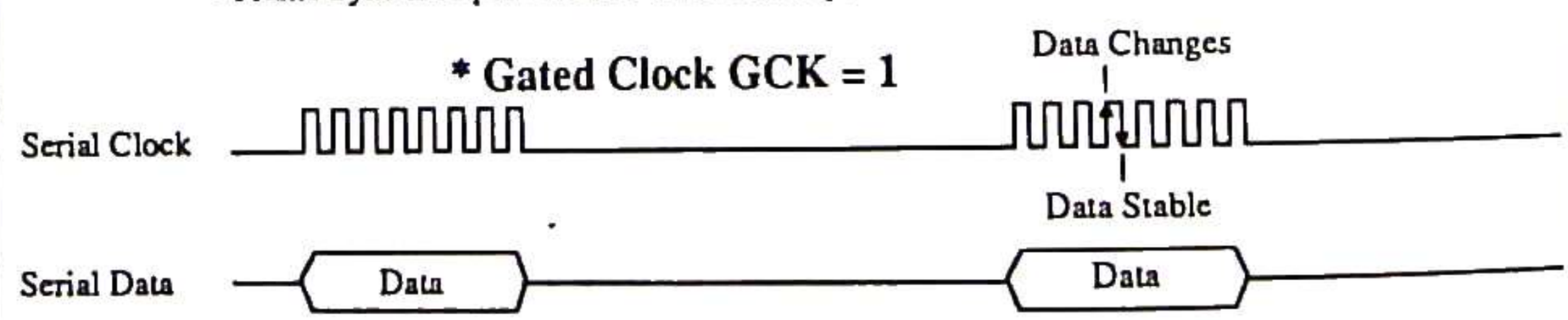
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RIE	TIE	RE	TE	MOD	GCK *	SYN	FSL	0	0	SCKD	SCD2	SCD1	SCD0	OF1	OF0

* Continuous Clock GCK = 0



- Frame Sync is required to tell when data is present.

* Gated Clock GCK = 1



- Word Synchronization is inherent in the Serial Clock signal.
- Frame Sync generation is optional.

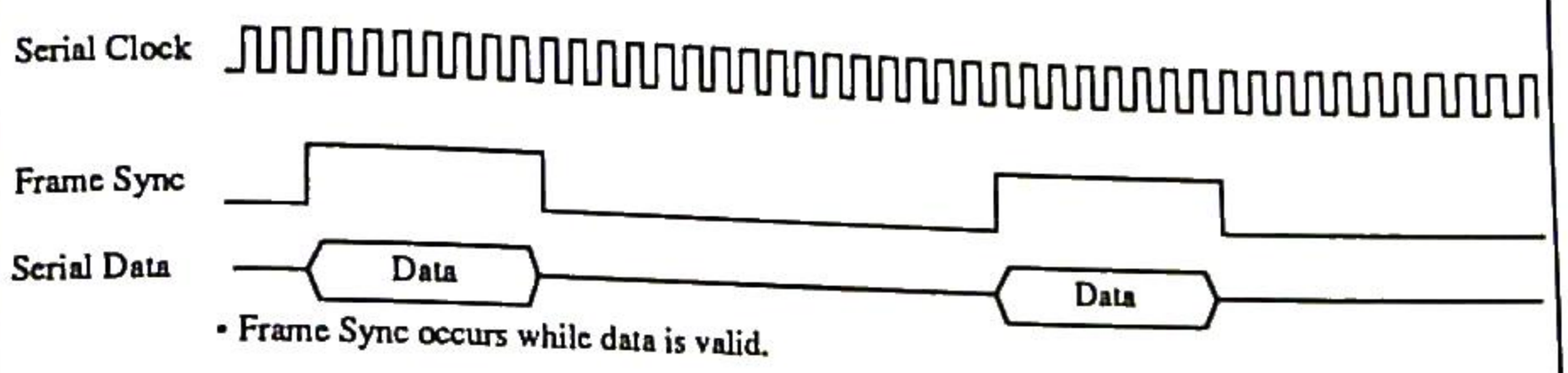
NCD02-1069

FRAME SYNC LENGTH

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

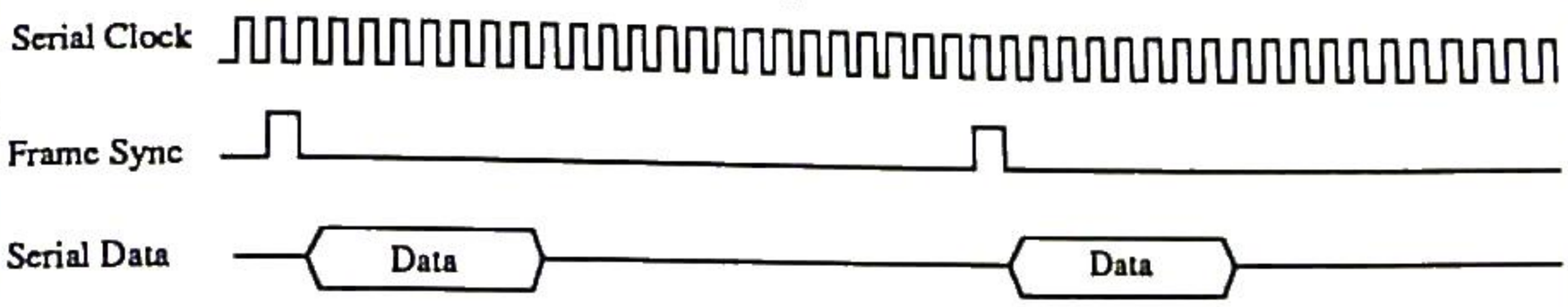
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RIE	TIE	RE	TE	MOD	GCK	SYN	FSL	0	0	SCKD	SCD2	SCD1	SCD0	OF1	OF0
							*								

*** Word Length FSL = 0**



• Frame Sync occurs while data is valid.

*** One Bit FSL = 1**



• Frame Sync occurs for one bit time preceding the data.

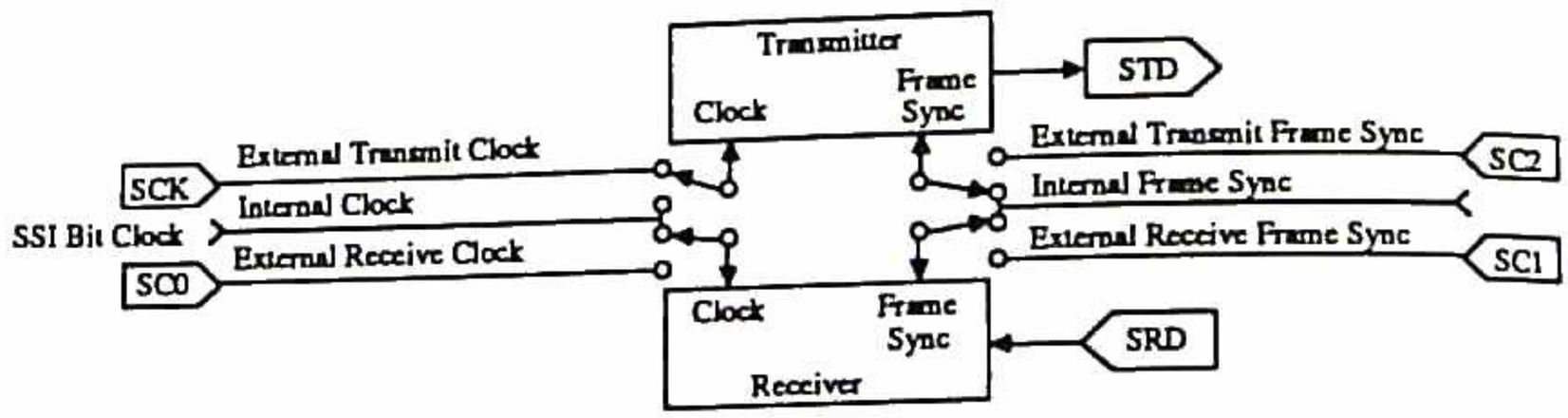
NCD02-1070

SYN BIT

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

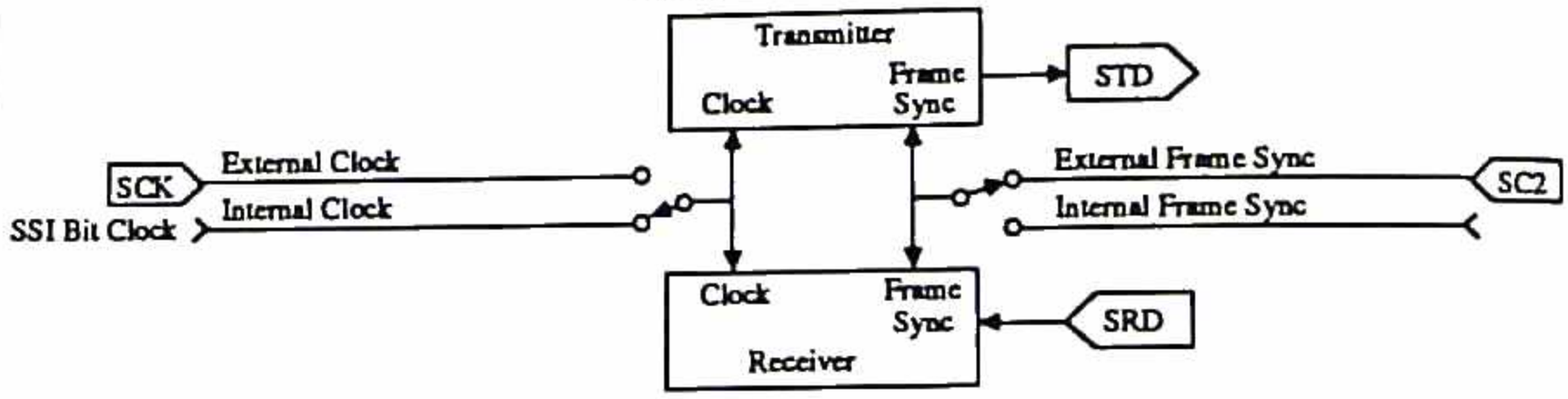
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RIE	TIE	RE	TE	MOD	OCK	SYN	PSL	0	0	SCKD	SCD2	SCD1	SCD0	OF1	OF0

• Asynchronous SYN = 0



• Transmitter and Receiver may have different clocks and frame syncs.

• Synchronous SYN = 1



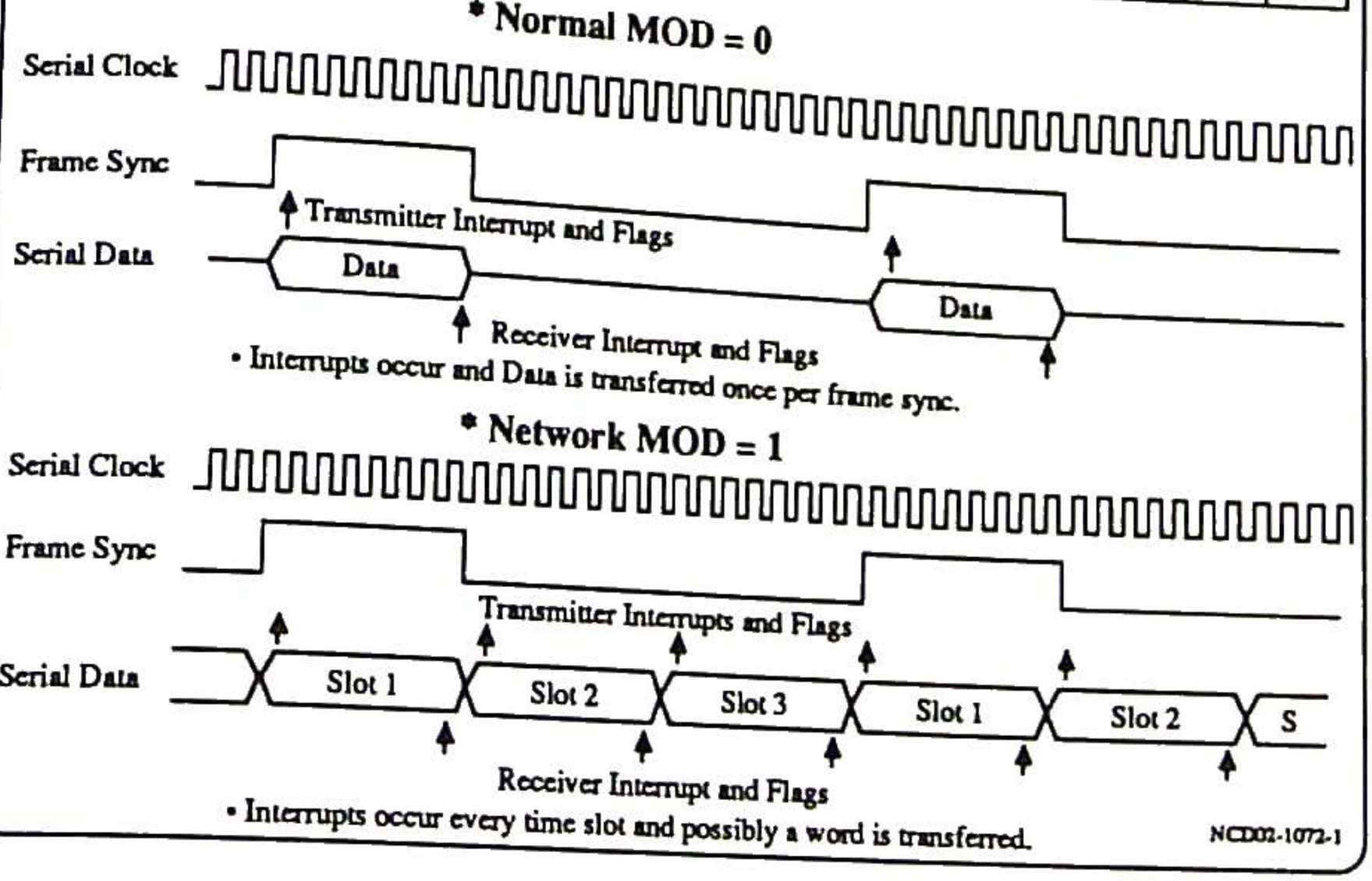
• Transmitter and Receiver have the same clock and frame syncs.

NCD02-1071

MODE BIT

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

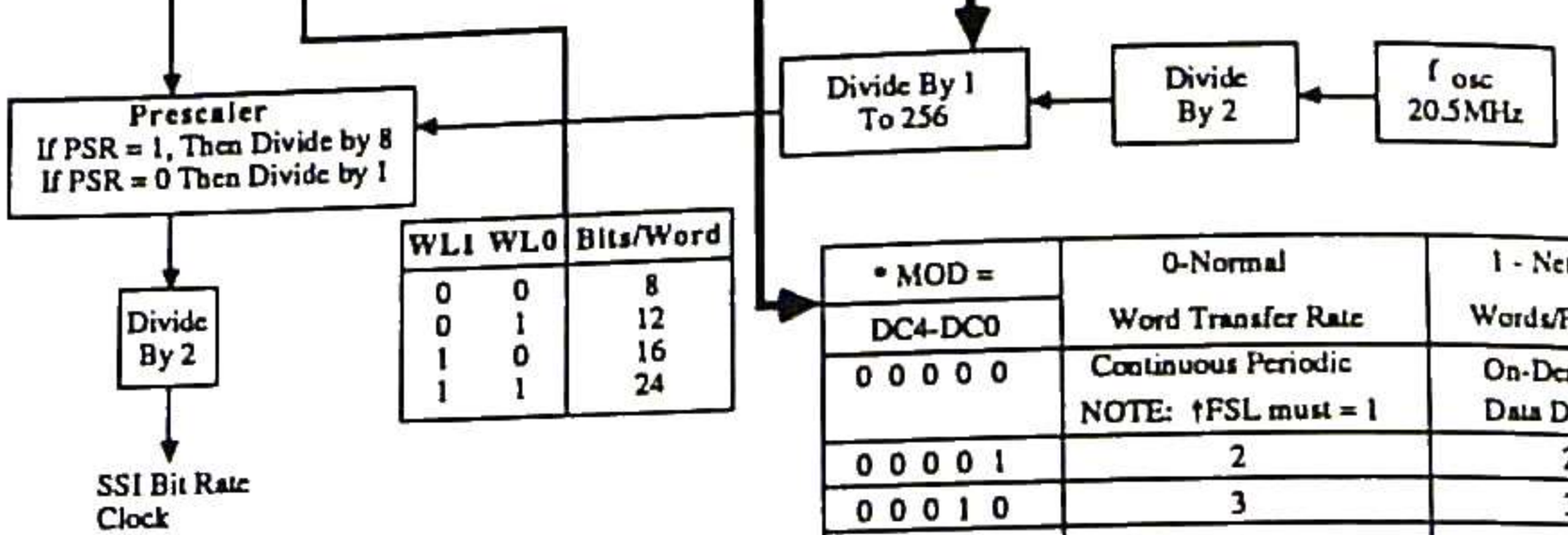
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RIE	TIE	RE	TE	MOD *	GCK	SYN	FSL	0	0	SCKD	SCD2	SCD1	SCD0	OF1	OF0



SSI - CLOCK

X:SFFEC-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	PM6	PM3	PM2	PM1	PM0



WL1	WL0	Bits/Word
0	0	8
0	1	12
1	0	16
1	1	24

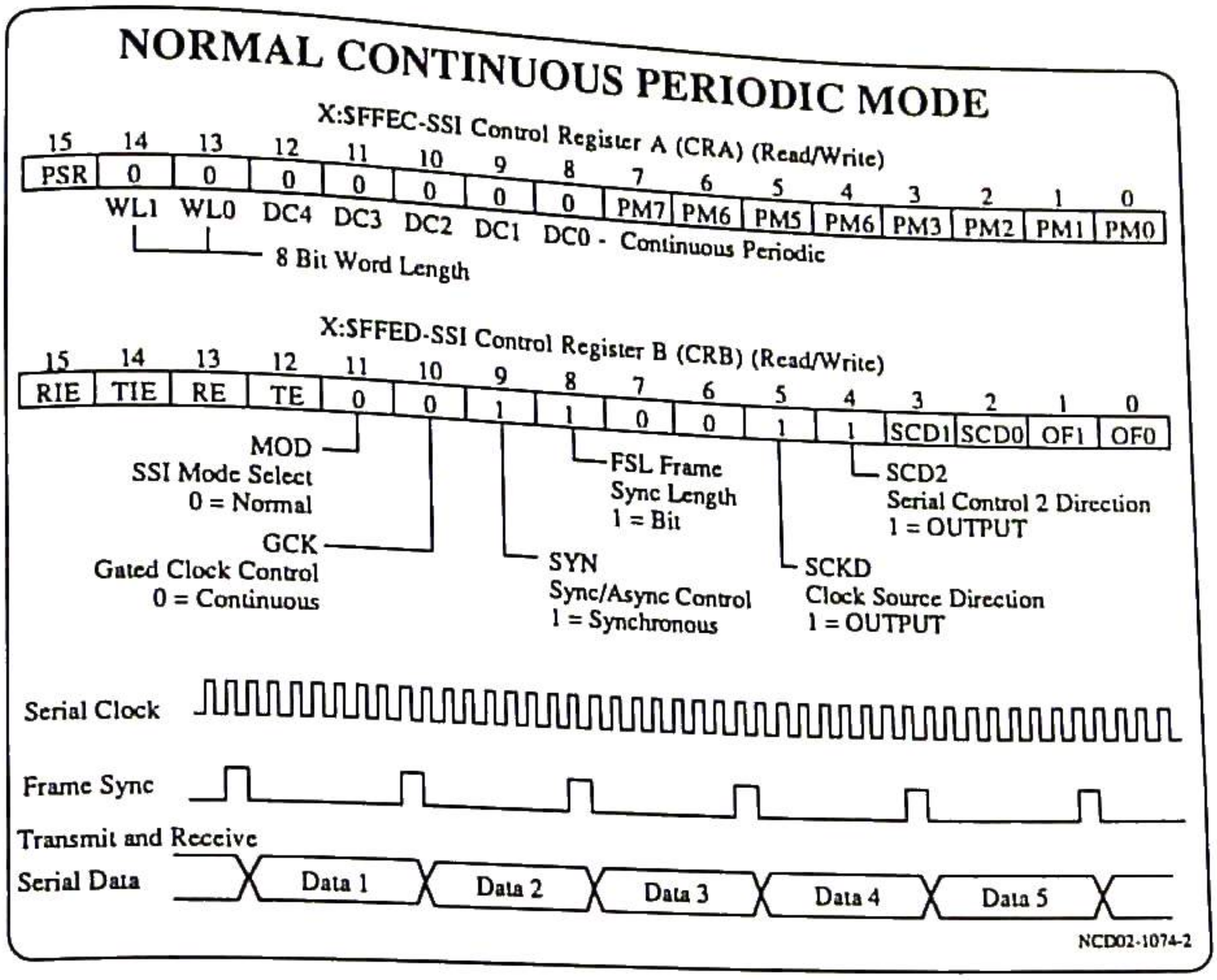
* MOD = DC4-DC0	0-Normal Word Transfer Rate	1 - Network Words/Frame
0 0 0 0 0	Continuous Periodic NOTE: †FSL must = 1	On-Demand Data Driven
0 0 0 0 1	2	2
0 0 0 1 0	3	3
0 0 0 1 1	4	4
• • •	• • •	• • •
1 1 1 1 1	32	32

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

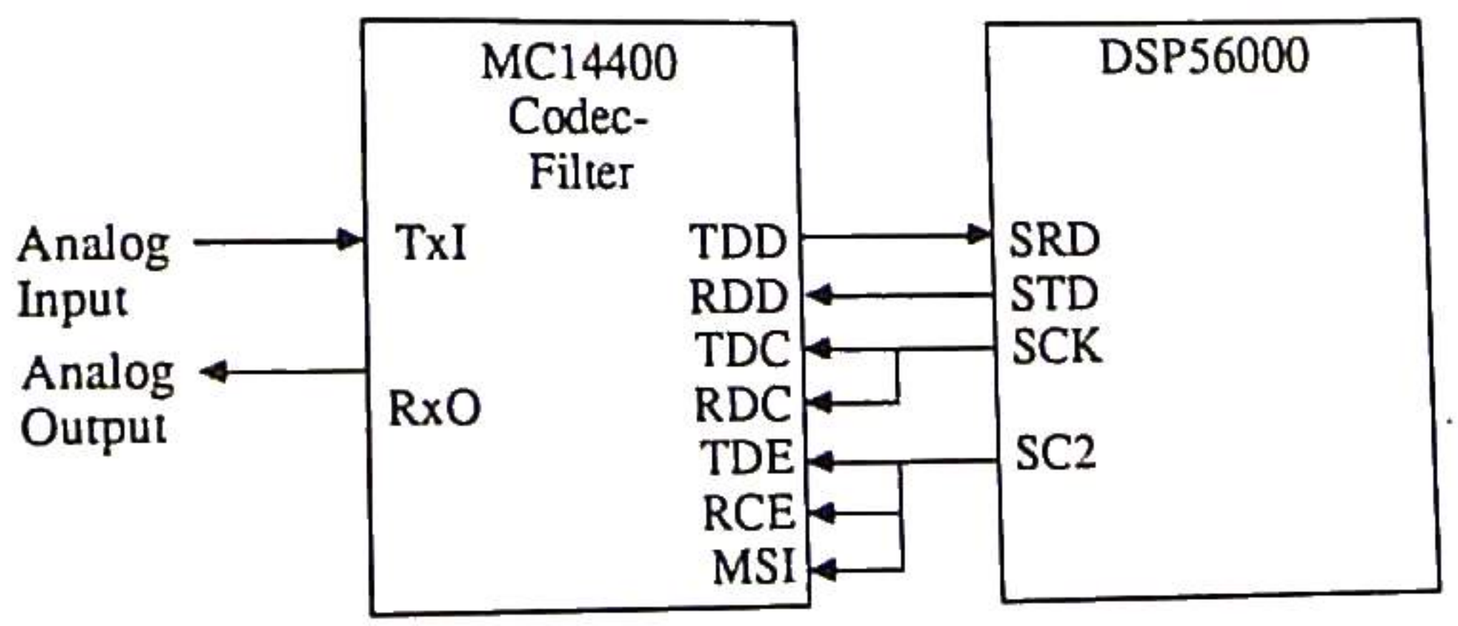
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RIE	TIE	RE	TE	MOD	GCK	SYN	FSL	0	0	SCKD	SCD2	SCD1	SCD0	OF1	OF0
				*			†								

NCD02-1073-1

NORMAL CONTINUOUS PERIODIC MODE



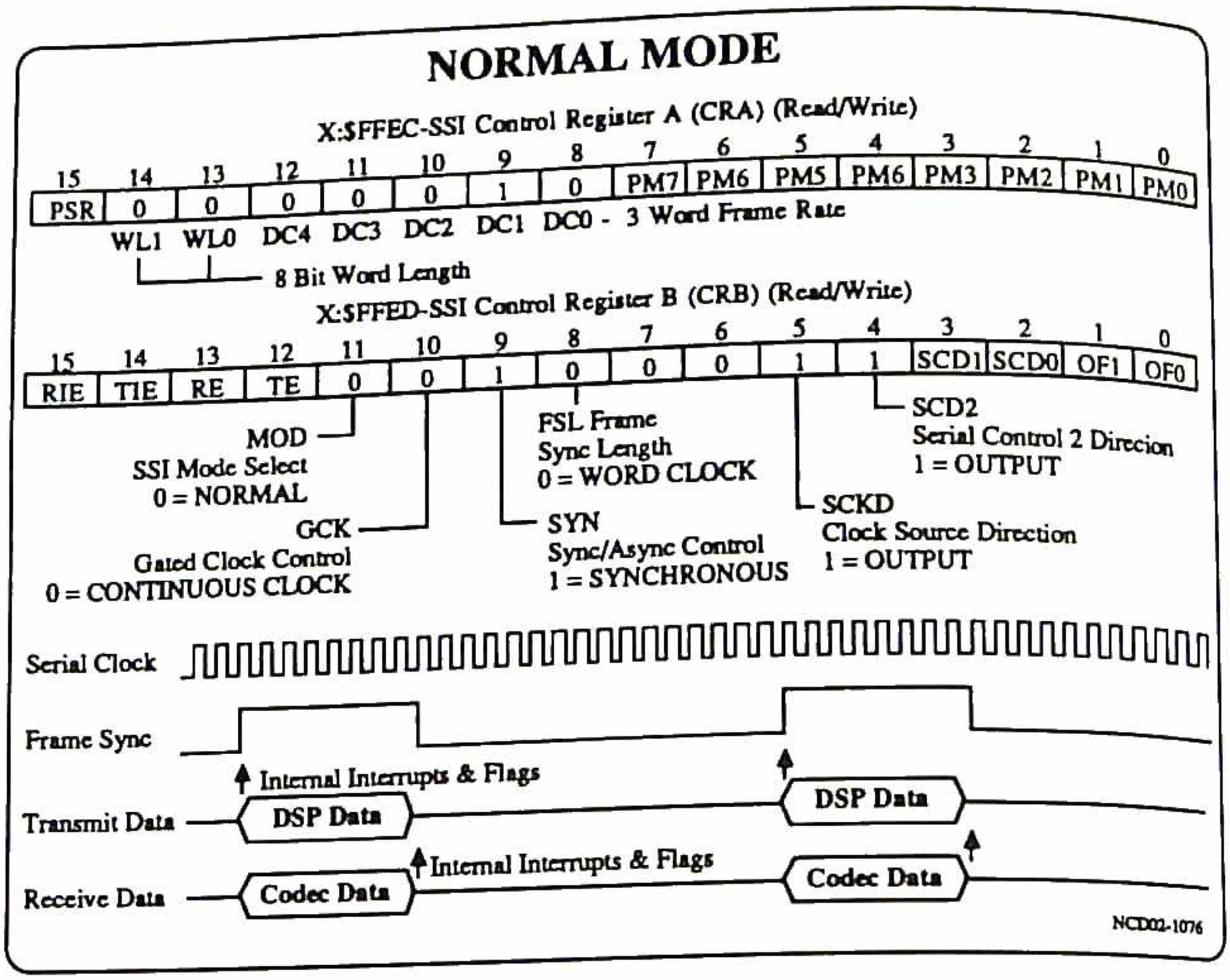
NORMAL MODE EXAMPLE



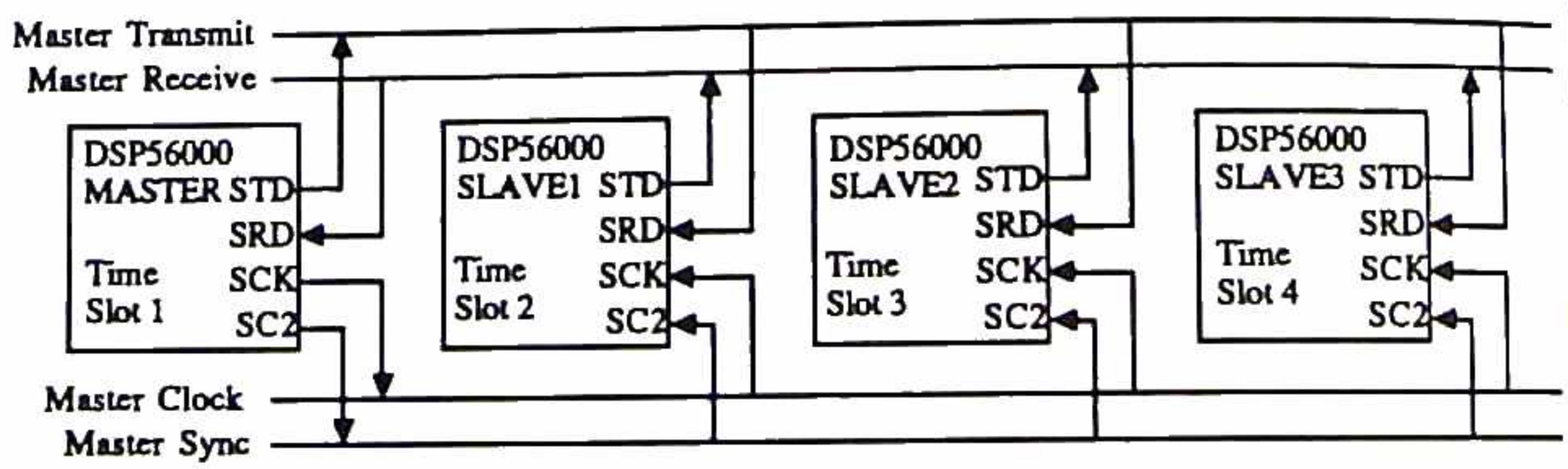
- No Glue Required!!
- Common CODEC Bit Rates 2.048 MHz, 1.544 MHz, 1.536MHz

NCD02-1075-1

NORMAL MODE



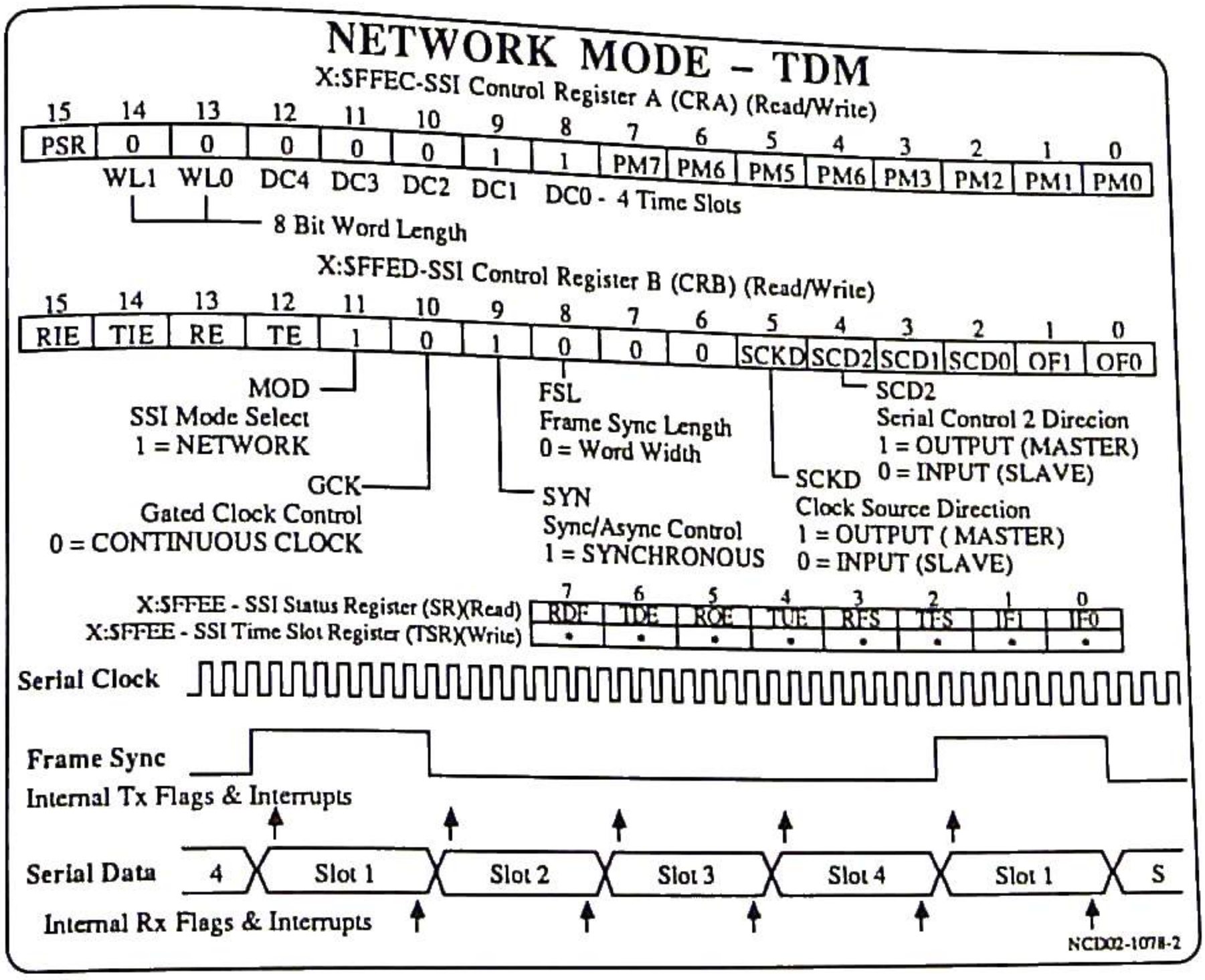
SSI TIME DIVISION MULTIPLEXED MASTER-SLAVE NETWORKING EXAMPLE



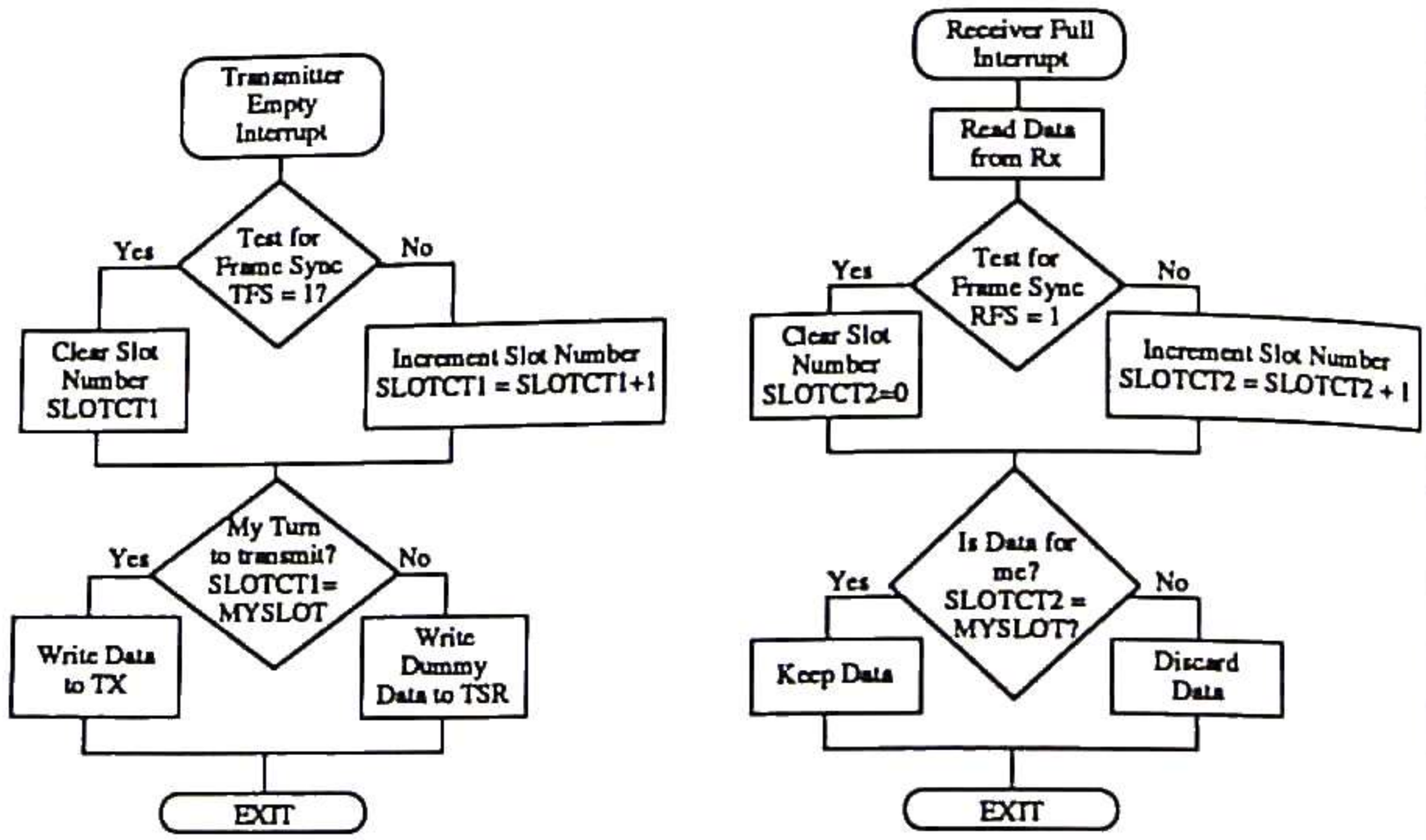
- 2 to 32 Time Slots per Serial Frame for Time Division Multiplexed (TDM) I/O
- Data may be transferred Every Time Slot
- Compatible with Bell and CCITT PCM Protocols
- Time Slot Assignment (TSA) is totally under Software Control
  - Devices can Transmit on Multiple Slots
  - Devices can Receive Multiple Time Slots
  - TSA can be Changed Dynamically
- Overrun and Underrun Error Detection

NCD02-1077-1

NETWORK MODE – TDM



TDM NETWORK SOFTWARE FLOWCHART

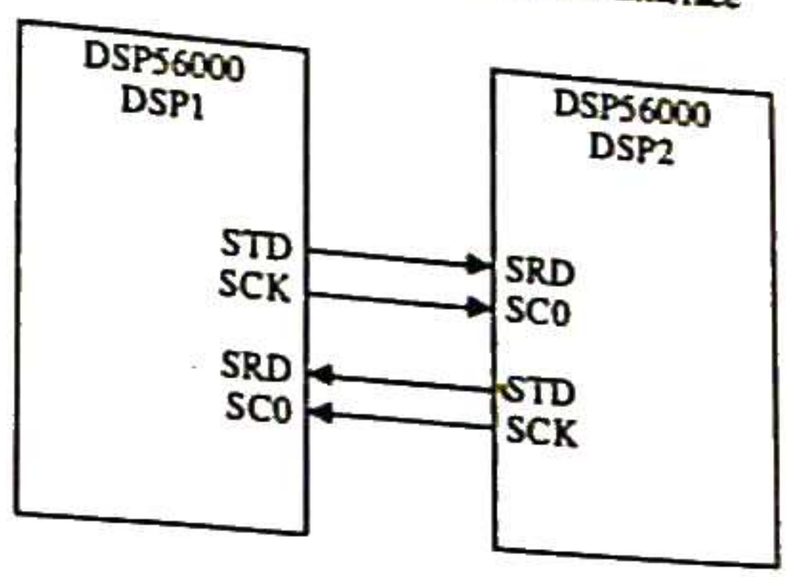


- The time slot is tracked in software. The example uses variables SLOTCT1 and SLOTCT2.
- The DSP keeps its own time slot in MYSLOT.
- TFS – Transmit Frame Sync Bit indicates if it is the start of a frame.
- If it is not time to transmit, Transmitter underrun is avoided by a dummy write to the slot register.

NCD02-1079

ON DEMAND EXAMPLE

Simple DSP56000 to DSP56000 Interface



Characteristics of On-Demand Mode

- Transmitter Underruns (TUE) are impossible and disabled.
- Non periodic and no time slots are defined.
- Frame Sync and Flags are generated when data is transferred.
- SPI compatible.

NCD02-1080

NETWORK MODE – ON DEMAND, DATA DRIVEN

X:SFFEC-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

WL1, WL0 — 24 Bit Word Length

X:SFFED-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	1	0	FSL	0	0	1	SCD2	SCD1	0	OF1	OF0

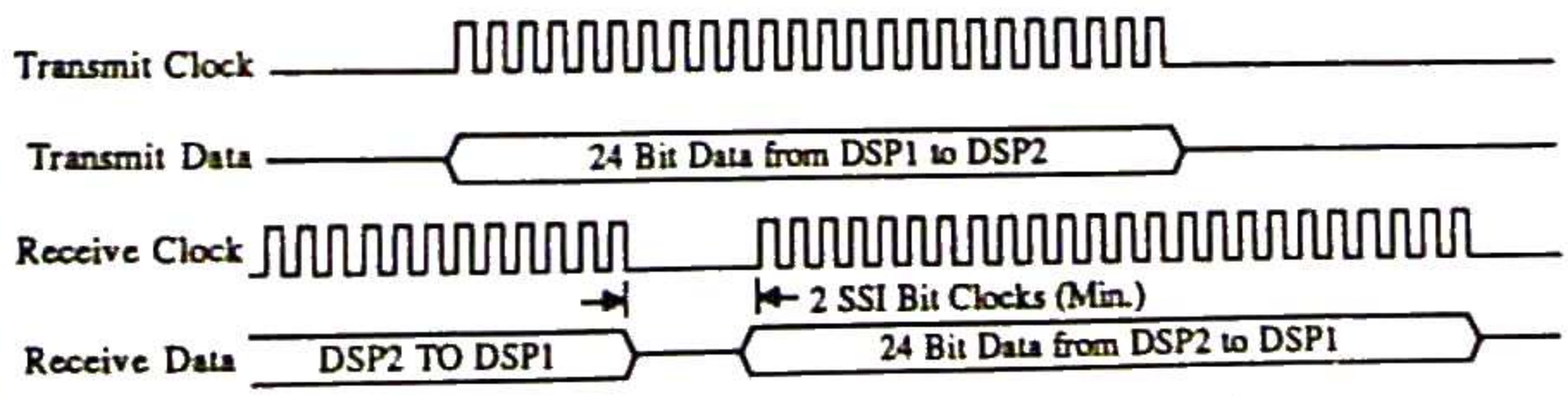
MOD — SSI Mode Select 1=NETWORK

GCK — Gated Clock Control 1=GATED CLOCK

SYN — Sync/Async Control 0=ASYNCHRONOUS

SCKD — Clock Source Direction 1=OUTPUT

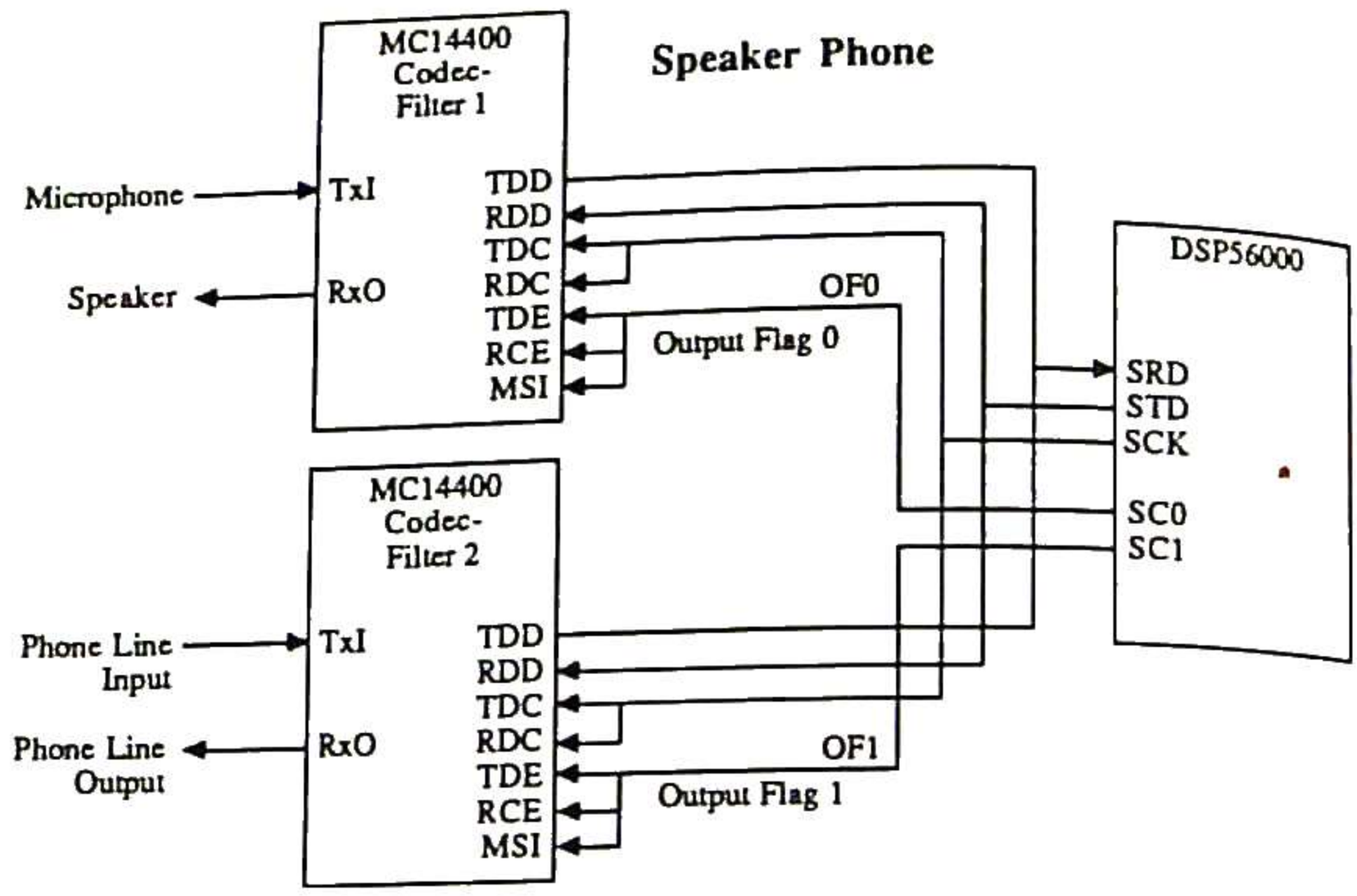
SCD0 — Serial Control 2 Direction 0=INPUT



- Two SSI Bit Clock times are automatically inserted between each data word. This guarantees Frame Sync will be low between every data word transmitted.

NCD02-1081

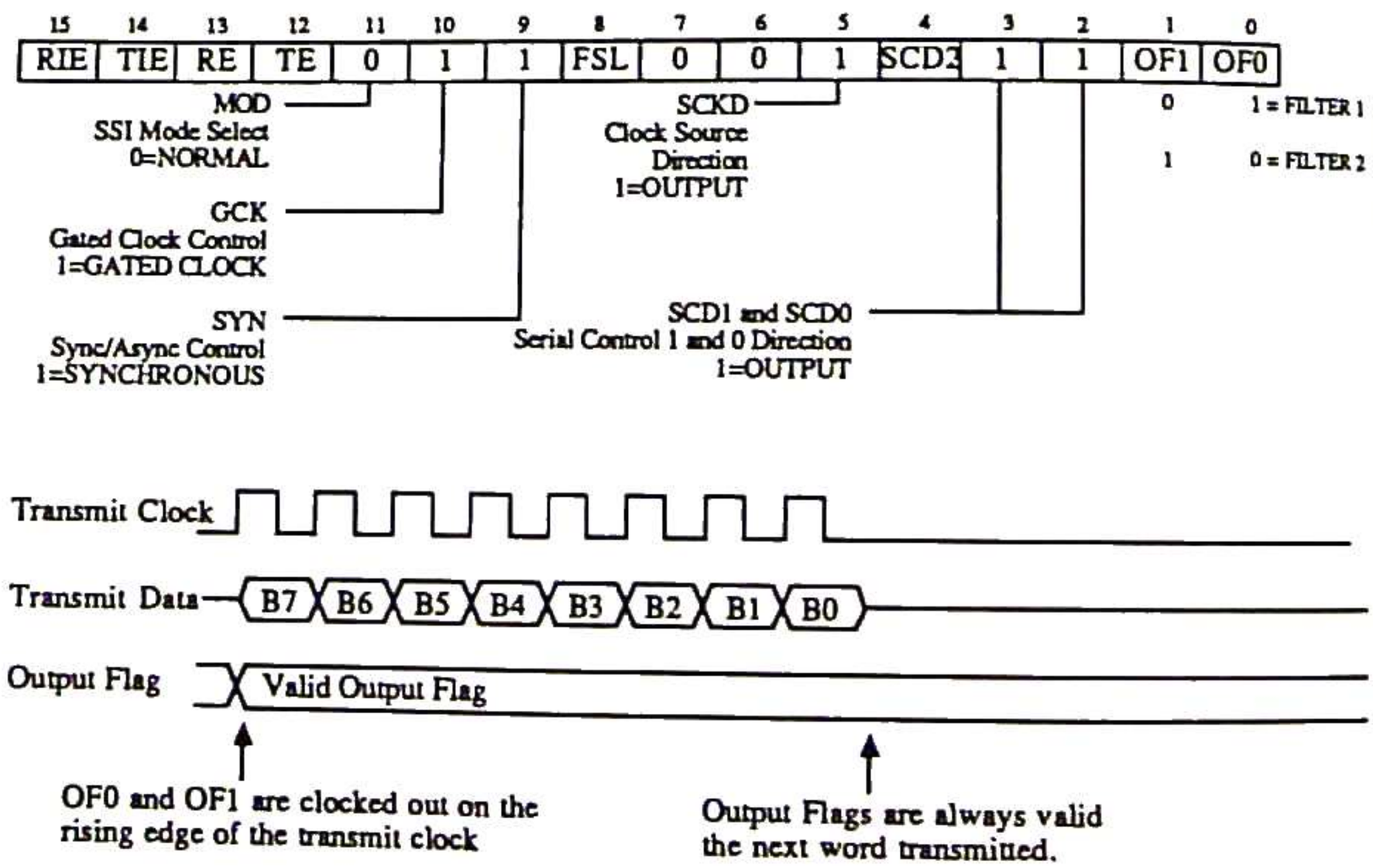
OUTPUT FLAG EXAMPLE



• SC0 and SC1 are Ouput Flag 0 and 1 used to software select either Filter 1 or 2.

NCD02-1082

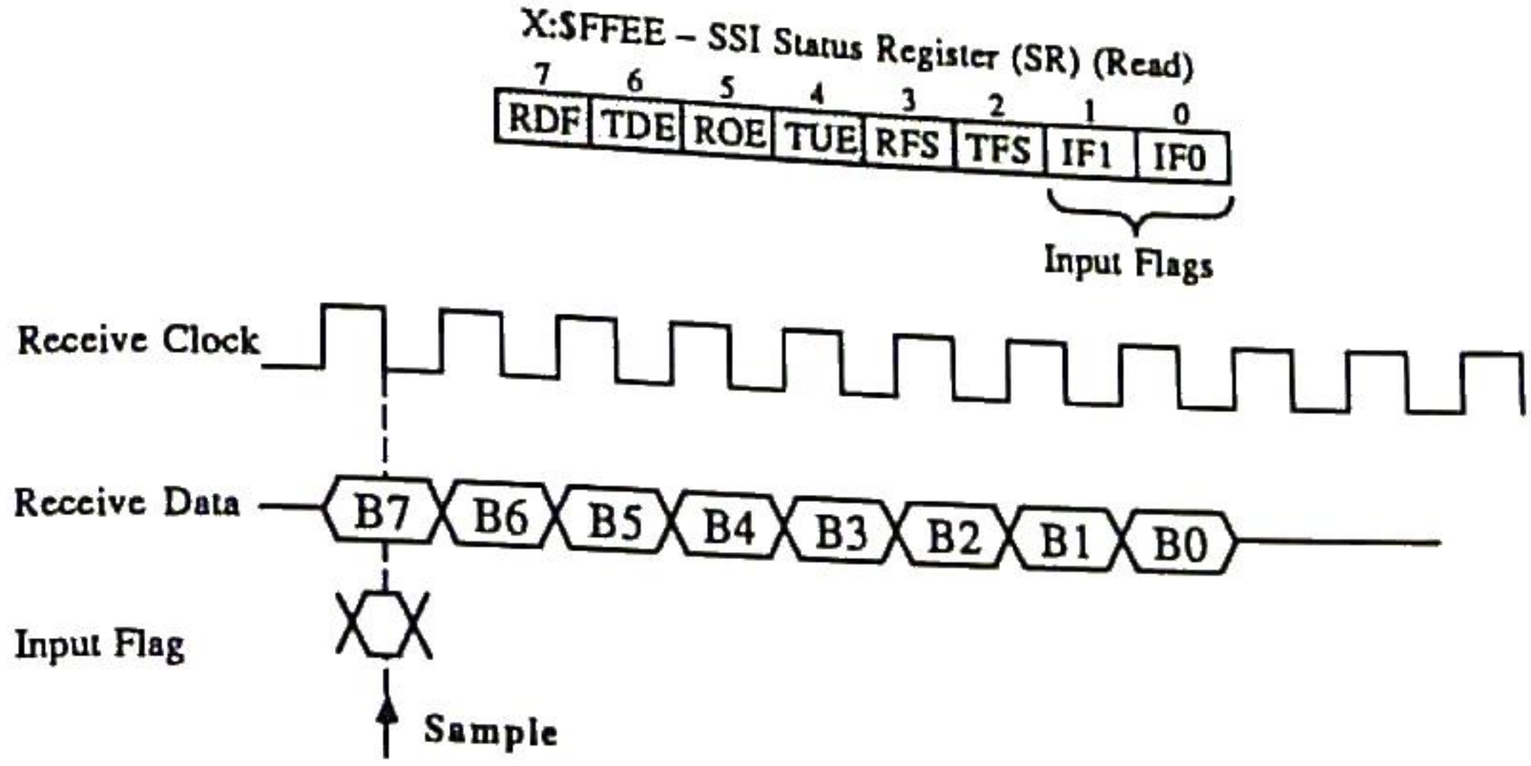
OUTPUT FLAGS



• Applications include Serial Device Chip Selects, Implementing Multidrop Protocols, Generating Bell PCM Signaling Frame Syncs, or Outputting Status Information.

NCD02-1083

INPUT FLAGS



- Input Flag Pin is sampled and latched Synchronous to the first bit sampled in the Receive Data Word.
- Input Flag Pin is Don't Care Until Next Receive Data Word.
- Useful for detecting Bell PCM Signaling Frame Sync, in implementing Multidrop Protocols, or Passing Status Information.

NCD02-1084

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

NCD02-1085

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.

NCD02-1086

DSP56000 RELOCATABLE ASSEMBLER AND LINKER

Objectives:

- Describe the assembler syntax, expressions, and directives.
- Assemble/Link/Load a Program.

NCD02-0800-1

Table of Contents

Assembler Command Line 9-5
Development Process 9-6
Source Statement Formats 9-7
Constants 9-8
Symbols 9-9
Operators 9-10
Operator Precedence 9-11
Memory Space Attributes 9-11
Functions (1 of 3) 9-12
Functions (2 of 3) 9-13
Functions (3 of 3) 9-14
Assembler Significant Characters 9-15
Load Time vs Run Time 9-16
Initialize Memory Space and Location Counters 9-17
Assembly Control Directives 9-18
Data Definitions 9-19
Listing Control 9-20
Assembler Options 9-21
Symbol Definitions 9-22
Object File Control 9-23
Macros 9-24
Duplicate Directives 9-25
Linker Command Line 9-26
Assembler/Linker Example 9-27
FIRT.ASM 9-28
FIRFLT.ASM 9-29
FIRT.LNK (1 of 3) 9-30
FIRT.LNK (2 of 3) 9-31
FIRT.LNK (3 of 3) 9-32
COEFICNT.ASM 9-33
COEFICNT.LNK 9-34
Link Map (1 OF 2) 9-34
Link Map (2 OF 2) 9-35
Download to ADS 9-35

Table of Contents

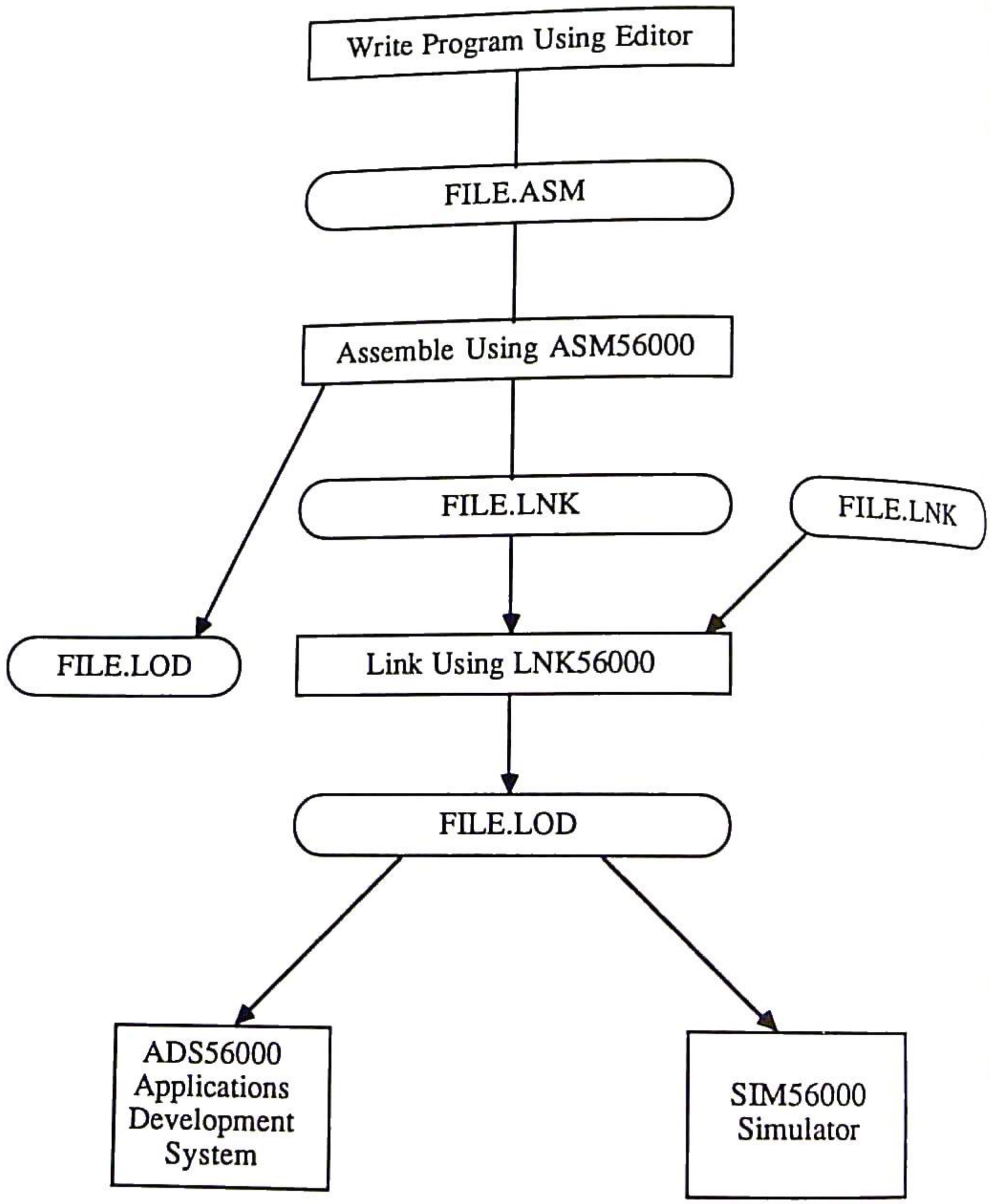
Assembler Command Line 9-5
Development Process 9-6
Source Statement Formats 9-7
Constants 9-8
Symbols 9-9
Operators 9-10
Operator Precedence 9-11
Memory Space Attributes 9-11
Functions (1 of 3) 9-12
Functions (2 of 3) 9-13
Functions (3 of 3) 9-14
Assembler Significant Characters 9-15
Load Time vs Run Time 9-16
Initialize Memory Space and Location Counters 9-17
Assembly Control Directives 9-18
Data Definitions 9-19
Listing Control 9-20
Assembler Options 9-21
Symbol Definitions 9-22
Object File Control 9-23
Macros 9-24
Duplicate Directives 9-25
Linker Command Line 9-26
Assembler/Linker Example 9-27
FIRT.ASM 9-28
FIRFLT.ASM 9-29
FIRT.LNK (1 of 3) 9-30
FIRT.LNK (2 of 3) 9-31
FIRT.LNK (3 of 3) 9-32
COEFICNT.ASM 9-33
COEFICNT.LNK 9-34
Link Map (1 OF 2) 9-34
Link Map (2 OF 2) 9-35
Download to ADS 9-35

ASSEMBLER COMMAND LINE

```
Usage:  ASM56000 [-A] [-B[<lodfil>]] [-D<symbol> <string>] [-I<ipath>] [-L[<lstfil>]]
                 [-M<mpath>] [-O<opt>[,<opt>...]] <srcfil>...

Where:
        -A       - Create and absolute load module                          Example
        -B       - Create and output object module                          -A
        lodfil   - Optional load file name for assembler output             -BOBJECT.LNK
        -D       - Define a symbol
        symbol   - User-defined symbol                                      -DPOINTS '256'
        string   - String associated with symbol
        -I       - Search a directory path for an include file
        ipath    - Include file directory path                              -I\FILTER\IIR\
        -L       - Specify an output listing be generated
        lstfil   - Optional list file name                                  -LFILTER.LST
        -M       - Search a directory path for a macro library
        mpath    - Macro library directory path                             -MFFTLIB\
        -O       - Command line assembler option directives
        opt      - Assembler option                                         -OS,CRE
        srcfil   - assembler source file(s)                                  SOURCE.ASM

Example:

        ASM56000 -BOBJECT -DPOINTS '256' -I\FILTER\IIR\ -L -MFFTLIB\ -OCRE,S SOURCE
```

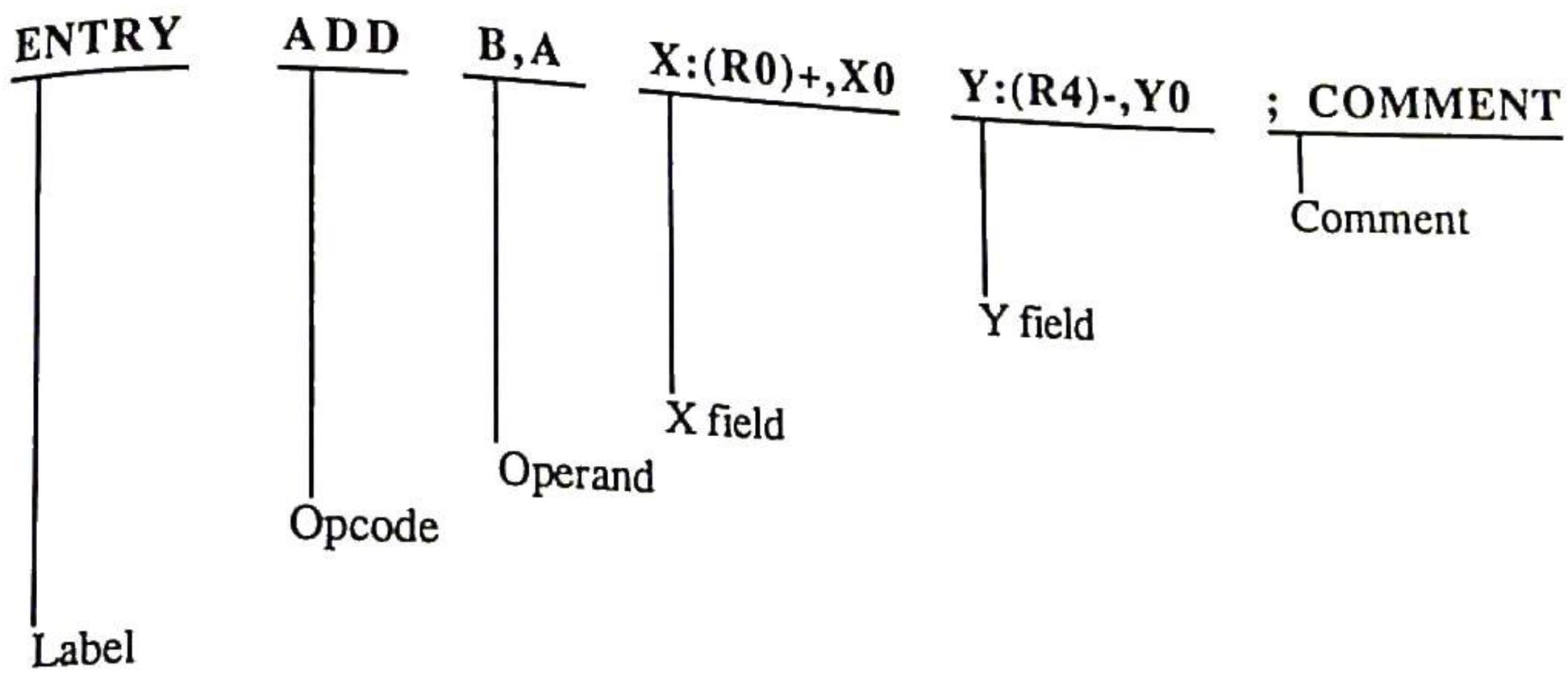
NCD02-0801

DEVELOPMENT PROCESS



NCD02-0802

SOURCE STATEMENT FORMATS



Label

<space> or <tab> – No Label

Underscore '_' for the first character is a local label

Alphabetic – first character is a symbol

Comment

; – Preceding a non literal string

; ; – A comment that is not reproduced in the listing

Continuation

- Source statements may be continued on multiple lines up to 255 characters by putting \ as the last character of the line to be continued.

SOURCE STATEMENT\
CONTINUATION OF SOURCE STATEMENT

NCD02-0804

CONSTANTS

Binary	Hexadecimal	Decimal	String
%1011010001	$1AB56C	Integer 12345 Floating Point 6E10 0.6 2.7e2	'ABC' 'DEF'

NCD02-003

SYMBOLS

- One to 255 characters long.
- First character must be alphabetic A-Z or a-z.
- Remaining characters can be A-Z, a-z, 0-9 or _.
- User-defined symbols are case sensitive.

Valid Symbols

loop_1
EXIT
a_b_C_19

Invalid Symbols

1_oop
AB•CD
HELLO.U

Reserved Symbols

X	A	PC	R0	N0	M0
X0	A0	MR	R1	N1	M1
X1	A1	CCR	R2	N2	M2
Y	A2	SR	R3	N3	M3
Y0	B	LC	R4	N4	M4
Y1	B0	LA	R5	N5	M5
	B1	SS	R6	N6	M6
	B2	SSH	R7	N7	M7
	AB	SSL			
	BA	OMR			
	A10				
	B10				

- Symbols are either a 32 bit integer or 32 bit floating point value.

NCD02-0805-1

OPERATORS

Unary

– Minus
~ Negate, integer only
! Logical negate, integer only

Arithmetic

+ Addition
- Subtraction
* Multiplication
/ Division
% Mod

Bitwise

& AND, integer only
| Inclusive OR, integer only
^ Exclusive OR, integer only

Shift

>> Shift right, integer only
<< Shift left, integer only

Relational

< Less than
> Greater than
== Equal
<= Less than or equal
>= Greater than or equal
! = Not equal

Logical

&& Logical AND
|| Logical OR

NCD02-0806-1

OPERATOR PRECEDENCE

1. Parenthetical expression (innermost first)
2. Unary minus, unary negate, unary logical negation
3. Multiplication, division, mod
4. Addition, subtraction
5. Bitwise AND, OR, EOR
6. Shift
7. Relational Operators
8. Logical AND, OR

- Operators of the same precedence are evaluated left to right.
- These operators cannot be applied to floating point operands.
  - Unary negate
  - Unary logical negate
  - Bitwise operators
  - Shift operators

NCD02-0807

MEMORY SPACE ATTRIBUTES

- Every Symbol carries one of the attributes:
  X, Y, L, P, or None
- The result of an expression will always have a memory space attribute.
- The attribute of an expression with two or more operands or operators is determined by this chart.

Left Operand Memory Space Attribute

Right Operand Memory Space Attribute \ Left Operand Memory Space Attribute	X	Y	L	P	None
X	X	*	X	*	X
Y	*	Y	Y	*	Y
L	X	Y	L	*	L
P	*	*	*	P	P
None	X	Y	L	P	N

* – This operation is illegal and will result in an error.

NCD02-0808

FUNCTIONS (1 OF 3)

	Memory Space Attribute
@ACS(<expression>) - ARC Cosine Result = 0 to pi, <expression> = between -1 and 1	None
@ASN(<expression>) - ARC Sine Result = pi/2 to pi/2, <expression> = between -1 and 1	None
@AT2(<expr1>,<expr2>) - ARC Tangent of <expr1>/<expr2> Result = -pi to pi	None
@ATN (<expression>) - ARC Tangent Result = -pi/2 to pi/2	None
@COS(<expression>) - Cosine	None
@CVF(<expression>) - Convert to floating point value	Same as <expression>
@CVI(<expression>) - Convert to integer value	Same as <expression>
@CVS({X | Y | L | P | N},<expression>) - Convert memory space attribute	Attribute Specified
@DEF(<symbol>) - Returns an Integer 1 (Memory space attribute N) if <symbol> has been defined with the DEFINE directive. Returns 0 otherwise.	N/A

NCD02-0809

FUNCTIONS (2 OF 3)

	Memory Space Attribute
@EXP(<expression>) - Returns the integer 1 if the evaluation of <expression> would NOT result in errors. Returns 0 otherwise.	N/A
@FRC(<expression>) - Obtains the binary fractional representation by the transformation @CVI((@POW(2,23)*<expression>)+0.5)	None
@INT(<expression>) - Returns 1 if <expression> has an integer result. Returns 0 otherwise.	N/A
@LCV({L | R}) - If L, return the value and memory space attribute of the load location counter. If R, return the value and memory space attribute of the runtime location counter	
@L10(<expression>) - Base 10 Log <expression> = greater than zero	None
@LOG(<expression>) - Natural Log <expression> = greater than zero	None
@MAC(<symbol>) - Returns an Integer 1 (Memory space attribute N) if <symbol> has been defined as a macro name. Returns 0 otherwise.	N/A
@MSP(<expression>) - Return the memory space attribute as an integer: 0 = None 1 = X Space 2 = Y Space 3 = L Space 4 = P Space	X | Y | L | P | N

NCD02-0810

FUNCTIONS (3 OF 3)

	Memory Space Attribute
@POW(<expr1>,<expr2>) - <expr1> raised to the power of <expr2>.	None
@SCP(<string1>,<string2>) - Returns the integer 1 if the two strings compare. Returns 0 otherwise.	N/A
@SIN(<expression>) - Sine	None
@SQT(expression) - Square root	None
@TAN(<expression>) - Tangent	None
@XPN(<expression>) - Exponential function. e to the power of <expression>	None

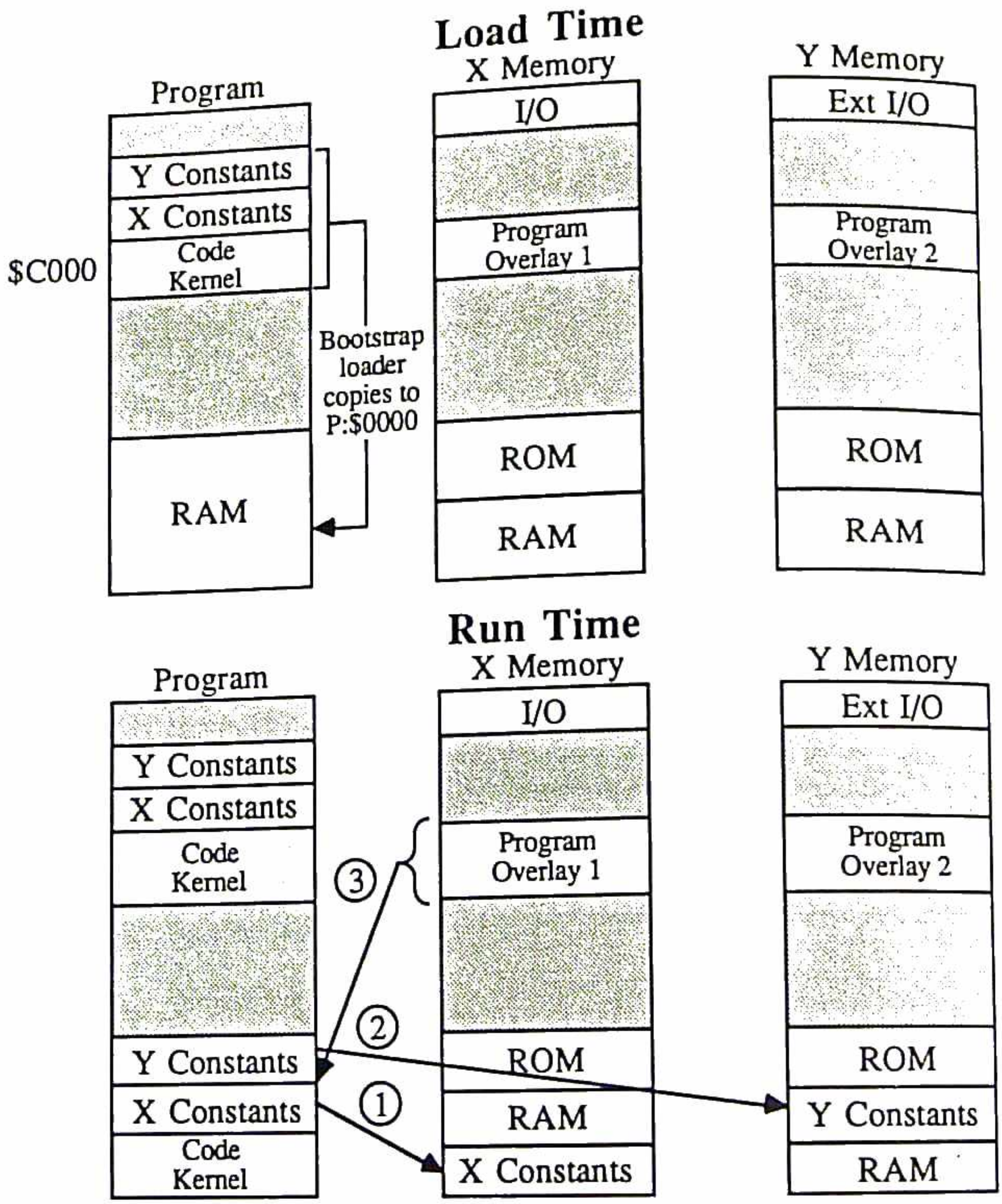
NCD02-0811

ASSEMBLER SIGNIFICANT CHARACTERS

;	–	Comment delimiter
::	–	Unreported comment delimiter
\	–	Line continuation character or Macro dummy argument concatenation operator
?	–	Macro value substitution operator
"	–	Macro string delimiter
@	–	Function delimiter
•	–	Location counter substitution
++	–	String concatenation operator
<<	–	I/O short addressing mode force operator
>	–	Long addressing mode force operator
<	–	Short addressing mode force operator
#	–	Immediate addressing mode
#>	–	Immediate long addressing mode force operator
#<	–	Immediate short addressing mode force operator

NC002-0812

LOAD TIME VS RUNTIME



At runtime the Code Kernel can then copy

1. X Constants to X Memory
2. Y Constants to Y Memory
3. Program Overlay 1 to replace the X and Y Constants that were in Program Memory

NCD02-0813

INITIALIZE MEMORY SPACE AND LOCATION COUNTERS

ORG <rms>[<rlc>]:[<exp1>][,<lms>[<llc>]:[<exp2>]] [<comment>]

<rms>
Which memory space (X, Y, L, or P) will be used as the Runtime Memory Space.

<rlc>
Which runtime counter H, L, or default (if neither H or L is specified), that is associated with the <rms> will be used as the Runtime Location Counter.

<exp1>
Initial value to assign to the runtime counter used as the <ric>. If <exp1> is not specified, then the last value the counter had will be used.

<lms>
Which memory space (X, Y, L, or P) will be the Load Memory Space. If this is not specified, then the <lms> will be the same as the <rms>.

<llc>
Which load counter, H, L, or default (if neither H or L is specified), that is associated with the <lms> will be used as the Load Location Counter.

<exp2>
Initial value to assign to the load counter used as the <llc>. If <exp2> is not specified, then the last value the counter had will be used.

- Each memory space has six counters
  - High load
  - Low load
  - Default load
  - High runtime
  - Low runtime
  - Default runtime

NCD02-0814

ASSEMBLY CONTROL DIRECTIVES

– COMMENT <delimiter> - Start comment lines

```
COMMENT   * Start of comment
            another comment line
          *
```

– DEFINE <symbol> < string> [comment] - Define substitution string

```
DEFINE ARRAYSIZ  '10*SAMPLSIZ'
```

– END [<expression>][<comment>] - End of Source Program

```
END START ; Starting execution address
```

– FAIL [<string>] [<comment>] – Programmer generated error

```
FAIL 'divide by zero' ; Report error in listing
```

– INCLUDE <string> [<comment>] – Include secondary file

```
INCLUDE 'MYPROG\IOEQU' ; MS-DOS file path
```

– RADIX < expression> [<comment>] – Change the input radix for comments

```
RADIX '16; Set input radix to Hex
```

– RDIRECT <direc> [,<direc>,...,<direc>] [<comment>] – Remove directive or mnemonic from table

```
RDIRECT RADIX, MAC
```

– UNDEF [<symbol>] [<comment>] – Undefine DEFINE symbol

```
UNDEF ARRAYSIZ ; Undefines the substitution string
```

– WARN [<string>] [<comment>] – Programmer generated warning

```
WARN 'Does not compute'
```

NCD02-0815

DATA DEFINITIONS

Block Storage of Constant

[<label>] BSC <expression> [,<expression] [<comment>]

UNUSED BSC $2FFF – @LCV(R), $FFFFFF ;Fill up unused EPROM

Define Constant

[<label>] DC <arg> [,<arg>,...,<arg>] [<comment>]

TABLE DC 1234, 555, $1212, 'ABCDEF' ;A table of data

Define Storage

[<label>] DS <expression> [<comment>]

S_BUF DS 12 ;12 word sample buffer

Define Modulo Storage

[<label>] DSM <expression> [<comment>]

ORG X:$100

M_BUF DSM 24 ;Circular buffer MOD24

Define Reverse Carry Storage

[<label>] DSR <expression> [<comment>]

ORG X:$100

R_BUF DSR 8 ;Reverse Carry Buffer for 16 point FFT

NCD02-0816

LISTING CONTROL

List the Assembly

LIST [<comment>]

Stop Assembly Listing

NOLIST [<comment>]

Top of Page/Size Page

PAGE [<page_width>,[<page_length>,<blank_top>,<blank_bottom>,<blank_left>]]

Argument	Default	Min	Max
page_width	80	1	255
page_length	66	10	255
blank_top	0	0	*
blank_bottom	0	0	*
blank_left	0	0	†

* blank_top + blank_bottom <=page_length -10

† blank_left < page_width

Send Control String to Printer

PRCTL <exp><string>, ... ,<exp><string> [<comment>]

Arguments are concatenated and sent to listing file

Initialize Program Title

TITLE [<String>] [<Comment>]

Initialize Program Sub-Title

STITLE [<String>] [<Comment>]

NCD02-0817

ASSEMBLER OPTIONS

OPT <option>[,<option>,...,<option>] [<comment>]

CC	Enable cycle counts and clear total cycle count.
NOCC	(default, reset) Disable cycle counts. Does not clear total cycle count.
CEX	Print DC expansions.
NOCEX	(default, reset) Do not print DC expansions.
CL	(default, reset) Print the conditional assembly directives
NOCL	Do not print the conditional assembly directives.
CM	(default, reset) Preserve comment lines of macros when they are defined.
NOCM	Do not preserve comment lines of macros when they are defined.
CONTC	Re-enable cycle counts. Does not clear total cycle counts.
CRE	Print a cross reference table at the end of the source listing.
FC	Fold trailing comments.
NOFC	(default, reset) Inhibit folded comments.
IL	Inhibit source listing.
LOC	Include local labels in the symbol table and cross-reference listing.
MC	(default, reset) Print macro calls.
NOMC	Do not print macro calls.
MD	(default, reset) Print macro definitions.
NOMD	Do not print macro definitions.
MEX	Print macro expansions.
NOMEX	(default, reset) Do not print macro expansions.
MU	Include a memory utilization report in the source listing.
S	Print symbol table at the end of the source listing.
SO	Write symbol information to object file.
U	Print the unassembled lines skipped due to failure to satisfy the condition of a conditional assembly directive.

NCD02-0818

SYMBOL DEFINITION

Start and End Section

SECTION <symbol> [<comment>]

•
•
•

<Section source statements>

•
•
•

ENDSEC [<comment>]

- All labels within a section have <symbol> associated with them.
- Labels outside of a section are global.

Equate a Symbol

<label> EQU [{X: | Y: | L: | P:}] <expression> [<comment>]

A_D_PORT EQU X:$FFC0 ;Analog to Digital Converter

Set a Symbol to a value

< label > SET <expression> [<comment>]

COUNT SET 0 ; Initialize count

External Section Symbol Definition

XDEF <symbol> [,<symbol>,...,<symbol>] [<comment>]

External Section Symbol Reference

XREF <symbol> [,<symbol>,...,<symbol>] [<comment>]

```
SECTION  FILTER
XDEF     LOOP_A; Entry point for FILTER Section
XREF     A_To_D : External call to A/D Driver
 •
 •
 •
END SEC
```

NCD02-0819

OBJECT FILE CONTROL

Comment Object File

COBJ <string> [<comment>]

COBJ 'Start of filter coefficients'

Object Code Identification Record

[<label>] IDENT <expression 1>, <expression 2> [<comment>]

FFILTER IDENT 1, 2 ; FIR FILTER VERSION 1.2

Write Symbol Information to Object File

SYMOBJ <symbol>[,<symbol>,<symbol>] [<comment>]

SYMOBJ XSTART, HIRTN, ERRPROC

NCD02-0820

MACROS

Macro Definition

```
<label> MACRO [<dumarg>[,<dumarg>,...,<dumarg>]] [<comment>]
        ⋮
        <macro definition statements>
        ⋮
        ENDM [<comments>]
```

```
N_R_MUL     MACRO     N,ARRA,ARRB,RESULT                    header
;
;THIS MACRO IMPLEMENTS N REAL MULTIPLES
;result(I) = a(I) * b(I) I=1...N
;where
;       N          = number of multiplications
;     ARRA         = base address of array a(I)
;     ARRB         = base address of array b(I)
;    RESULT        = base address of array result(I)
;
;
            MOVE     #ARRA,R0
            MOVE     #ARRB,R4
            MOVE     #RESULT,R1
            MOVE                        X:(R0)+,X0     Y:(R4)+,Y1
            DO       _END,#N
            MPYR     X0,Y1,A            X:(R0)+,X0     Y:(R4)+,Y1
            MOVE                        A,X:(R1)+
_END
            ENDM                                       terminator
```

Macro Call

```
N_R_MUL  LA + 5, LB + 5, LC, LD
```

Purge a Macro

```
PMACRO <symbol>[,<symbol>, ,<symbol>] [<comment>]
PMACRO N_R_MUL ;Reclaim macro table space
```

Macro Library

```
MACLIB <pathname> [<comment>]
MACLIB '\MACROS\MATH\'          ;IBMPC
```

NCD02-0821-1

DUPLICATE DIRECTIVES

Duplicate a Sequence of Source Lines

```
[<label>]  DUP <expression> [<comment>]
           .
           .
           .
           ENDM
```

Example:

The sequence of source input statements,

```
COUNT SET 3
      DUP COUNT ;ASR BY COUNT
      ASR A
      ENDM
```

would generate the following in the source listing:

```
COUNT SET 3
      DUP COUNT ;ASR BY COUNT
      ASR A
      ASR A
      ASR A
      ENDM
```

Duplicate Sequence with Arguments

```
[<label>]    DUPA    <dummy>,<arg>[,<arg>,...,<arg>] [<comment>]
             .
             .
             .
             ENDM
```

Example:

If the input source file contained the following statements,

```
DUPA    VALUE, 12,32,34
DC      VALUE
ENDM
```

then the assembled source listing would show

```
DUPA    VALUE,12,32,34
DC      12
DC      32
DC      34
ENDM
```

Duplicate with Characters

```
[<label>]    DUPC    <dummy>,<string> [<comment>]
             .
             .
             .
             ENDM
```

Example:

If the input source file contained the following statements,

```
DUPC    VALUE, '123'
DC      VALUE
ENDM
```

then the assembled source listing would show

```
DUPC    VALUE, '123'
DC      1
DC      2
DC      3
ENDM
```

NCD02-0822

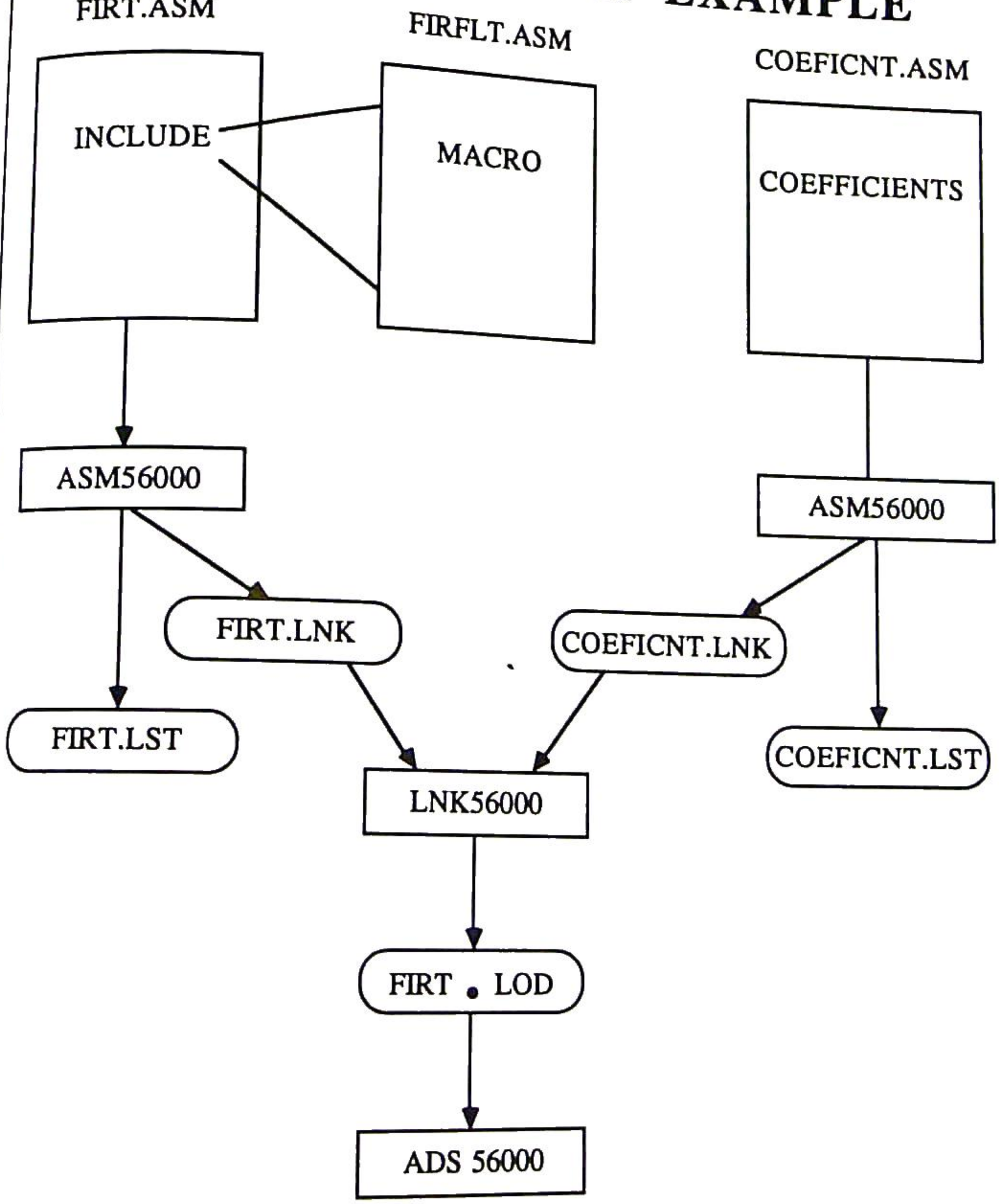
LINKER COMMAND LINE

```
Usage:  LNK56000 [-B[<lodfil>]] [-L<library>] [-M[<mapfil>]]
                 [-O<mem>[<ctr>][<map>]:<origin>] [-R[<memfil>]] [<lnkfil>...
```

where:		Example
-B	- Create an output object module	-BOBJECT.LOD
lodfil	- Optional load file name	
-L	- Search a library for modules	-LIO.LIB
library	- Library file name	
-M	- Create a map file	-MFILTER.MAP
mapfil	- Optional load map file name	
-O	- Specify module origin	
mem	- memory space (X, Y, L, P)	
ctr	- location counter (L-low, H-high, none-default)	
map	- memory mapping (I-internal, E-external, B -bootstrap)	-OPLI:40
origin	- memory space origin	
-R	- Read a file for map locations	-RLINK.MEM
memfil	- optional memory map file name	
lnkfil	- link input file name	

NCD02-0823-1

ASSEMBLER/LINKER EXAMPLE



NCD02-0824-1

FIRT.ASM

```
title   'Finite Impulse Response Filter'
        comment +
  This program originally available on the Motorola DSP bulletin board.
  It is provided under a DISCLAIMER OF WARRANTY available from
  Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx., 78735.

  Last Update 16 Jul 87   Version 1.0
                +

;
;       FIR test program
;
        opt     cex,mex,cre,cc,mu
        page    132,66,0,10
        section filter
        include 'firflt'

datin   equ     $ffff           ;location in Y memory of input file
datout  equ     $fffe           ;location in Y memory of output file
npts    equ     20              ;number of points to process
ntaps   equ     5               ;number of taps in filter

        org     x:0             ;filter states
states  dsm     ntaps

        org     y:              ;external coefficients
        xref    coef

        org     p:$100
start
        move    #states,r0      ;point to filter states
        move    #ntaps-1,m0     ;mod(ntaps)
        move    #coef,r4        ;point to filter coefficients
        move    #ntaps-1,m4     ;mod(ntaps)

        do      #npts,_endp     ;one loop for each tap

        movep   y:datin,x0      ;get sample

        fir     ntaps           ;do fir macro

        movep   a,y:datout      ;output sample
_endp
        endsec
        end
```

NCD02-0825

FIRFLT. ASM

```
;
fir      macro   ntaps
firmac   ident   1,0
;
;        FIR filter macro
;
         clr     a     x0,x:(r0)+  y:(r4)+,y0      ;save first state
         rep     #ntaps-1
         mac     x0,y0,a  x:(r0)+,x0  y:(r4)+,y0
         macr    x0,y0,a  (r0)-
         endm
```

Invoking the Assembler

```
ASM56000 -B -Lprn firt
```

NCD02-0826

FIRT.LST (1 OF 3)

```
Motorola DSP56000 Macro Cross Assembler  Version 2.01  88-04-25  10:14:21  firt.asm  Page 1
Finite Impulse Response Filter

                        comment *
                        This program originally available on the Motorola DSP bulletin board.
                        It is provided under a DISCLAIMER OF WARRANTY available from
                        Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx., 78735.

                        Last Update 16 Jul 87   Version 1.0
                        *

                    ;
                    ;       FIR test program
                    ;
                                    opt     cex,mex,cre,cc,mu
                                    page    132,66,0,10
                                    section filter
                                    include 'firflt'

                            ;
                            fir     macro   ntaps
                            firmac  ident   1.0
                            ;
                            ;       FIR filter macro
                            ;
                                    clr     a           x0,x:(r0)+  y:(r4)+,y0  ;save first state
                                    rep     #ntaps-1
                                    mac     x0,y0,a     x:(r0)+,x0  y:(r4)+,y0
                                    macr    x0,y0,a     (r0)-
                                    endm

                            datin   equ     $ffff                               ;location in Y memory of input file
        0000FFFF            datout  equ     $fffe                               ;location in Y memory of output file
        00000FFFE           npts    equ     20                                  ;number of points to process
        00000014            ntaps   equ     5                                   ;number of taps in filter
        00000005

                                    org     x:0
        X:0000              states  dsm     ntaps                               ;filter states
        X:0000

                                    org     y:
        Y:0000                      xref    coef                                ;external coefficients

                                    org     p:$100
        P:0100                  start
                                    move            #states,r0                  ;point to filter states
        P:0100 300000  [2 -     2]  move            #ntaps-1,m0                 ;mod(ntaps)
        P:0101 0504A0  [2 -     4]  move            #coef,r4                    ;point to filter coefficients
        P:0102 64F400  [4 -     8]  move
               000000
        P:0104 0504A4  [2 -    10]  move            #ntaps-1,m4                 ;mod(ntaps)

        P:0105 061480  [6 -    16]  do              #npts,_endp                 ;one loop for each tap
               00010C

        P:0107 09443F  [4 -    20]  movep           y:datin,x0                  ;get sample

                                    fir     ntaps                               ;do fir macro
                                firmac  ident   1.0
```

FIRT. LST (2 OF 3)

```
Motorola DSP56000 Macro Cross Assembler  Version 2.01  88-04-25  10:14:21  firt.asm  Page 2
Finite Impulse Response Filter

52      *                                       ;
53      *                                       ;       FIR filter macro
54      *                                       ;
55      *    P:0108 F01813  [2 -      22]              clr     a           x0,x:(r0)+  y:(r4)+,y0  ;save first state
56      *    P:0109 0604A0  [4 -      26]              rep     #ntaps-1
57      *    P:010A F098D2  [2 -      28]              mac     x0,y0,a     x:(r0)+,x0  y:(r4)+,y0
58      *    P:010B 2050D3  [2 -      30]              macr    x0,y0,a     (r0)-
59
60           P:010C 09CE3E  [4 -      34]              movep                           a,y:datout              ;output sample
61                                               _endp
62                                                       endsec
63                                                       end
0     Errors
0     Warnings


Motorola DSP56000 Macro Cross Assembler  Version 2.01  88-04-25  10:14:21  firt.asm  Page 3
Finite Impulse Response Filter

Macros:
Name                Definition        Section
                       Line
fir...............17                   filter

Relocatable Sections:
Name
filter

Symbols:
Name                Type     Value          Section          Attributes
datin............int      0000FFFF       filter
datout...........int      0000FFFE       filter
npts.............int      00000014       filter
ntaps............int      00000005       filter
start............int    P:00000100       filter
states...........int    X:00000000       filter

Symbol cross-reference listing:
Name                Line number (* is definition)
datin............    28*     48
datout...........    29*     60
npts.............    30*     46
ntaps............    31*     34      42      44      56
start............    40*
states...........    34*     41
```

FIRT.LST (3 OF 3)

```
Motorola DSP56000 Macro Cross Assembler  Version 2.01  88-04-25  10:14:21  firt.asm  Page 4
Finite Impulse Response Filter

                        Memory Utilization Report

X Memory

Start    End     Length   Type     Label          Section        Overlay Address
0000     0004         5   MOD      states         filter
0005     FFFF     65531   UNUSED

Y Memory

Start    End     Length   Type     Label          Section        Overlay Address
0000     FFFF     65536   UNUSED

L Memory

Start    End     Length   Type     Label          Section        Overlay Address
0000     FFFF     65536   UNUSED

P Memory

Start    End     Length   Type     Label          Section        Overlay Address
0000     00FF       256   UNUSED
0100     010C        13   CODE                    filter
010D     FFFF     65267   UNUSED
```

NCD02-0829-1

COEFICNT. ASM

```
; File of coefficients to be assembled separately and then linked to the FIRT
; filter file.

coefile ident    0,4
        title    'table of coefficients'
        opt      cex,cre,mex
        page     132,66,0,10

        section ymem
        org      y:0
        xdef     coef
coef    dupa     numbers,.1,.3,-.1,.2,-.5
        dc       numbers
        endm
        endsec
        end
```

Invoking the assembler

```
ASM56000  -B -Lprn coeficnt
```

NCD02-0830

COEFICNT. LNK

```
Motorola DSP56000 Macro Cross Assembler  Version 2.01  88-04-25  11:45:01  coeficnt.asm  Page 1

                        ; file of coefficients to be assembled separately and then linked to the FIRT
1                       ; filter file.
2
3                       coefile   ident   0,4
4                                 opt     cex,crs,mex
6                                 page    132,66,0,10
7
8                                 section ymem
9                                 org     y:0
10        Y:0000                  xdef    coef
11                      coef      dupa    numbers,.1,.3,-.1,.2,-.5
12                                dc      numbers
13   m                            endm
14   m                            dc      .1
15   d+   Y:0000 0CCCCD           dc      .3
16   d+   Y:0001 266666           dc      -.1
17   d+   Y:0002 F33333           dc      .2
18   d+   Y:0003 19999A           dc      -.5
19   d+   Y:0004 C00000           endsec
20                                end
21
0    Errors
0    Warnings

Motorola DSP56000 Macro Cross Assembler  Version 2.01  88-04-25  11:45:01  coeficnt.asm  Page 2
table of coefficients

Relocatable Sections:

Name

ymem

Symbols:

Name              Type    Value        Section        Attributes

coef.............int    Y:00000000   ymem           GLOBAL

Symbol cross-reference listing:

Name              Line number (* is definition)
coef.............   11      12*
```

Invoking the Linker

```
LNK56000 -B -M firt coeficnt
```

NCD02 001

LINK MAP (1 OF 2)

```
Motorola DSP56000 Cross Linker  Version 1.00  88-04-25  12:00:10  firt.map  Page 1
                          Section Link Map by Address
X Memory (default)
Start    End     Length    Section
  No sections

X Memory (low)
Start    End     Length    Section
  No sections

X Memory (high)
Start    End     Length    Section
  No sections

Y Memory (default)
Start    End     Length    Section
0000     0004        5     ymem              Abs

Y Memory (low)
Start    End     Length    Section
  No sections

Y Memory (high)
Start    End     Length    Section
  No sections

L Memory (default)
Start    End     Length    Section
  No sections

L Memory (low)
Start    End     Length    Section
  No sections

L Memory (high)
Motorola DSP56000 Cross Linker  Version 1.00  88-04-25  12:00:10  firt.map  Page 2
Start    End     Length    Section
  No sections

P Memory (default)
Start    End     Length    Section
0100     010C       13     filter            Abs

P Memory (low)
Start    End     Length    Section
  No sections

P Memory (high)
Start    End     Length    Section
  No sections

                          Section Link Map by Name
Section                 Memory       Start    End      Length
GLOBAL                  None
filter            Abs   P default    0100     010C       13
ymem              Abs   Y default    0000     0004        5
```

NCD02 001

LINK MAP (2 OF 2)

```
Motorola DSP56000 Cross Linker  Version 1.00  88-04-25  12:00:10  firt.map  Page 3

                    Symbol Listing by Name

Name                Type    Value        Section          Attributes
coef.............int     Y:00000000   ymem             GLOBAL
datin............int       0000FFFF   filter
datout...........int       0000FFFE   filter
npts.............int       00000014   filter
ntaps............int       00000005   filter
start............int     P:00000100   filter
states...........int     X:00000000   filter
                                      filter

                    Symbol Listing by Value

Value           Name
00000000        coef
00000000        states
00000005        ntaps
00000014        npts
00000100        start
0000FFFE        datout
0000FFFF        datin


          Starting the Applications Development System

     ADS56000
```

NCD02-0833

DOWNLOAD TO ADS

```
load firt
Loading from file: C:\NCD02\ASSEMB.LER\firt.lod


disassemble p:$100#11
ADM#0 P:$0100 300000          = MOVE #$0,R0
ADM#0 P:$0101 0504A0          = MOVE #$4,M0
ADM#0 P:$0102 64F400 000000 = MOVE #>$0,R4
ADM#0 P:$0104 0504A4          = MOVE #$4,M4
ADM#0 P:$0105 061480 00010C = DO #<$14,$10D
ADM#0 P:$0107 09443F          = MOVEP Y:<<$FFFF,X0
ADM#0 P:$0108 F01813          = CLR A X0,X:(R0)+ Y:(R4)+,Y0
ADM#0 P:$0109 0604A0          = REP #<$4
ADM#0 P:$010A F098D2          = MAC Y0,X0,A X:(R0)+,X0 Y:(R4)+,Y0
ADM#0 P:$010B 2050D3          = MACR Y0,X0,A (R0)-
ADM#0 P:$010C 09CE3E          = MOVEP A,Y:<<$FFFE


display y:0..4
Y:$0000 0CCCCD 266666 F33333 19999A C00000
ADM#0 P:$0100 300000          = MOVE #$0,R0
```

NCD02-0834

DIGITAL SIGNAL PROCESSING ALGORITHMS

Objectives:

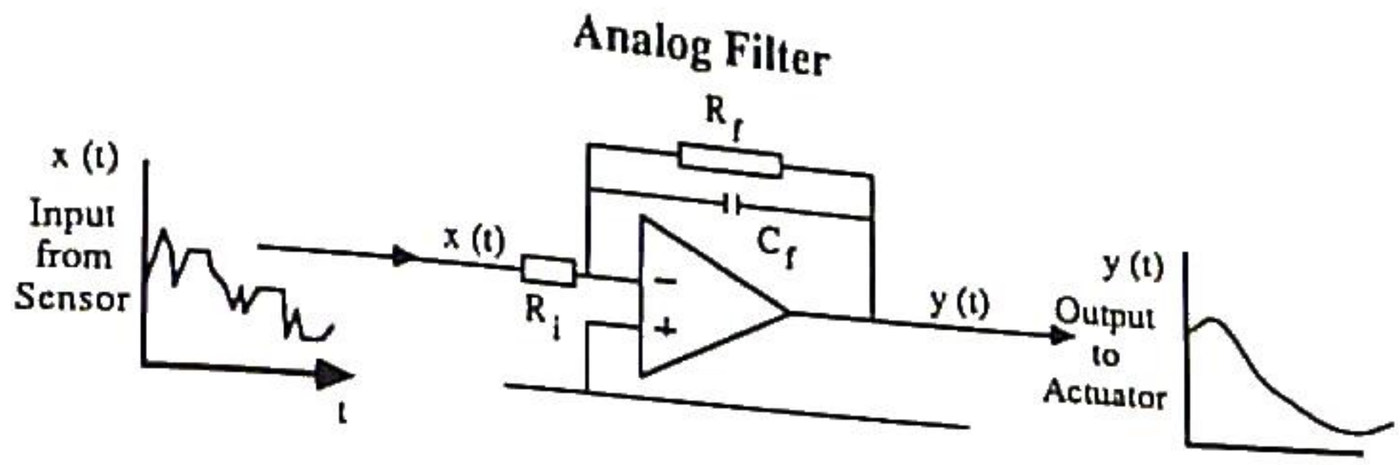
- Describe the function of a General Purpose Digital Signal Processing System.
- Describe the basic concepts of a Finite Impulse Filter.
- Describe the basic concepts of an Infinite Impulse Response Filter.
- Describe the basic concepts of an Adaptive FIR Filter.

NCD02-0900-2

Table of Contents

Analog Signal Processing 10-5
Analog Filter Characteristics 10-5
Digital Signal Processing 10-6
Digital Filter Characteristics 10-6
Analog Signal Sampling 10-7
Aliasing in the Time Domain 10-8
Aliasing in the Frequency Domain 10-8
Bandlimiting Antialiasing Filter 10-9
Analog Output Reconstruction Filter 10-10
Basic Digital Signal Processing Equation 10-11
Digital Signal Processing System 10-12
Filter Design (Steps) 10-13
Digital Filtering 10-14
Digital Filter Types 10-14
Direct Form Finite Impulse Response Filter 10-15
FIR Filter Realization 10-15
Four Tap FIR Filter Example 10-16
Four Tap FIR Filter Realization 10-16
Four Tap FIR Filter Flow Chart 10-17
Four Tap Fir Filter Program 10-17
Infinite Impulse Response All Pole Filter 10-18
Five State All Pole IIR Filter Example 10-18
Five State All Pole IIR Filter Program 10-19
Infinite Impulse Response Filter 10-20
IIR Filter Direct Canonic Form 10-21
Fifth Order IIR Filter Program 10-23
Selection of Filter Coefficients 10-24
Adaptive Filtering 10-25
Adaptive Filter Model 10-25
Adptive Filter Applications 10-26
Noise Canceller 10-26
System Identification 10-27
Adaptive Filter Lattice Diagram 10-27
Adaptive Filter Equation 10-28
Adaptive Filter Program 10-28

ANALOG SIGNAL PROCESSING

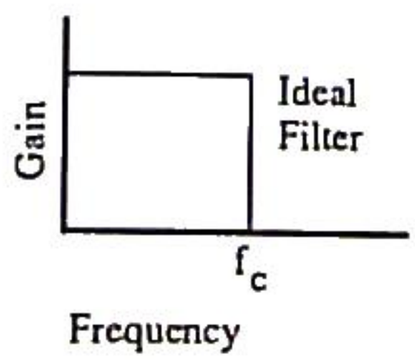


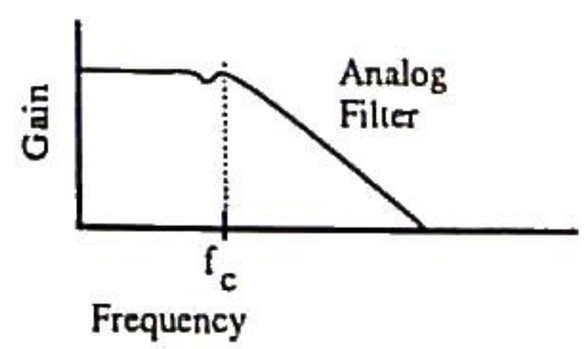
$$\frac{y(j\omega)}{x(j\omega)} = -\frac{R_f}{R_f}\left[\frac{1}{1 + j\omega R_f C_f}\right]$$

NCD02-0201-1

ANALOG FILTER CHARACTERISTICS

Frequency Characteristics



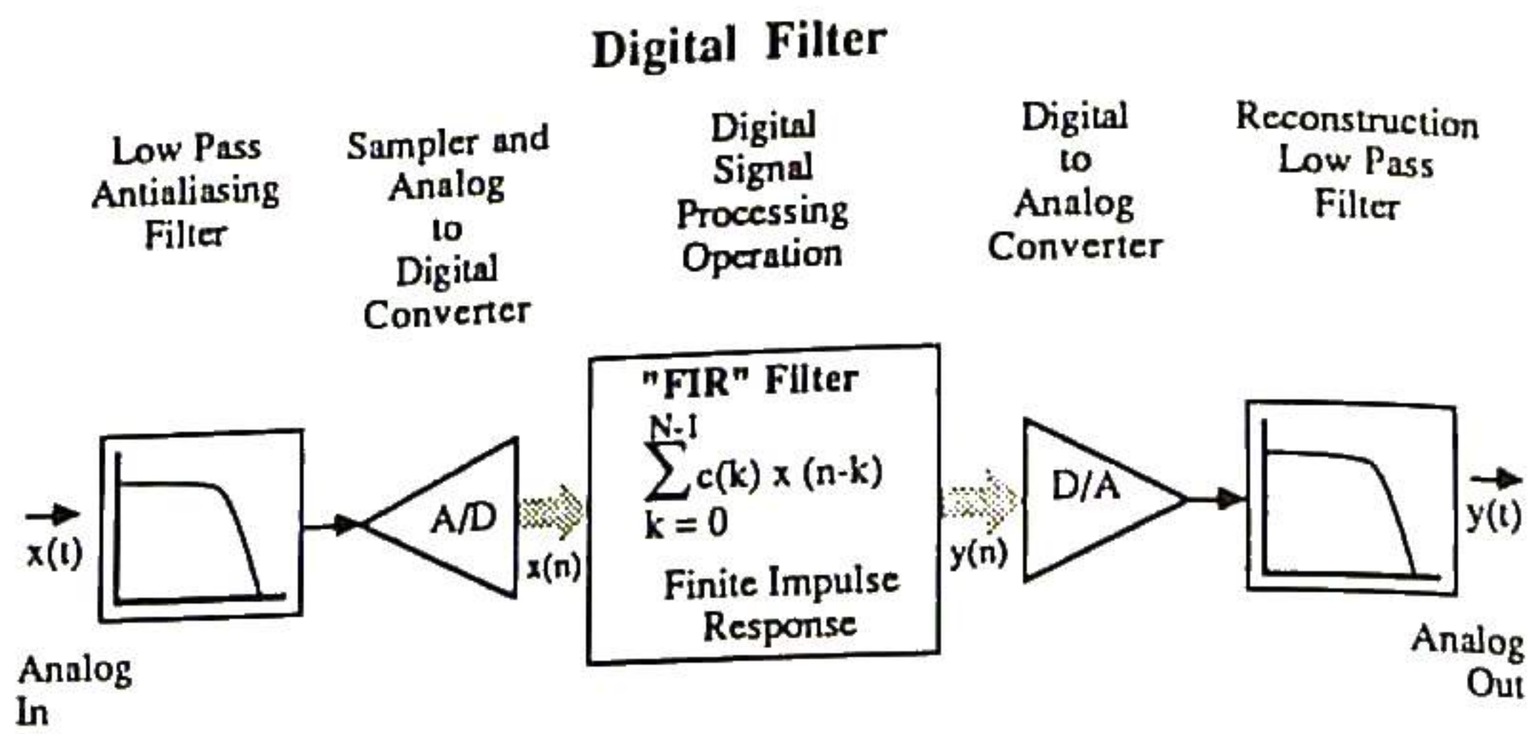


Drawbacks:

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

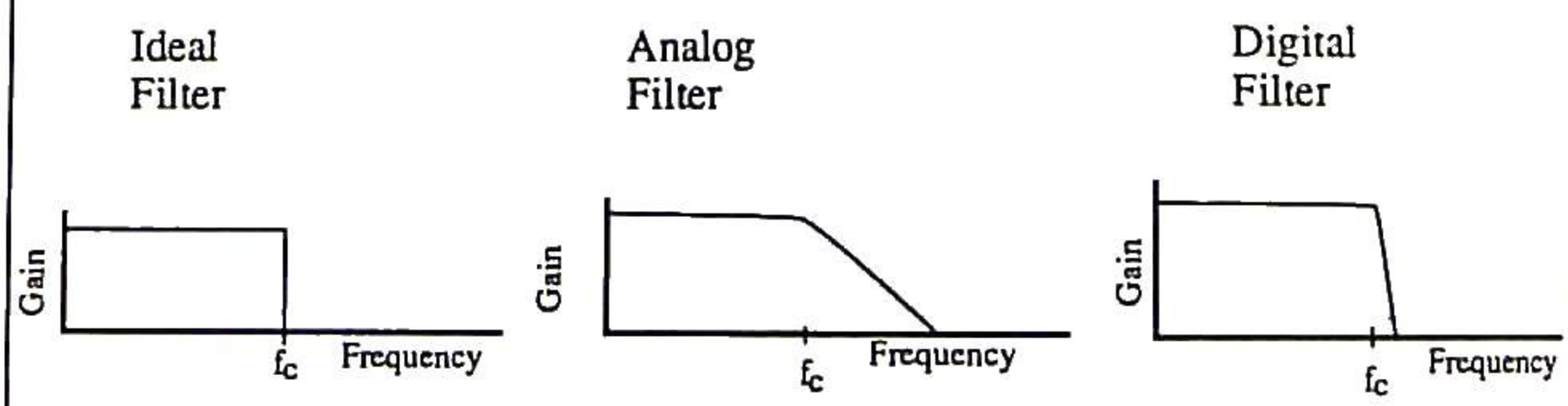
NCD02-0202

DIGITAL SIGNAL PROCESSING



NCD02-0203-1

DIGITAL FILTER CHARACTERISTICS



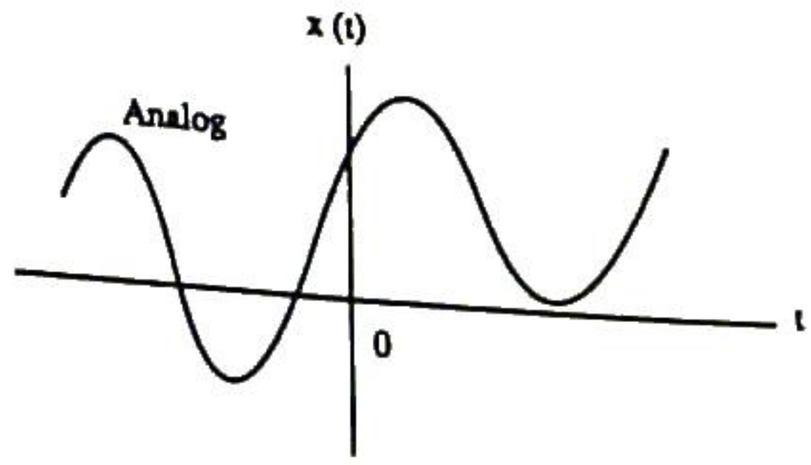
Advantages:

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

NCD02-0204

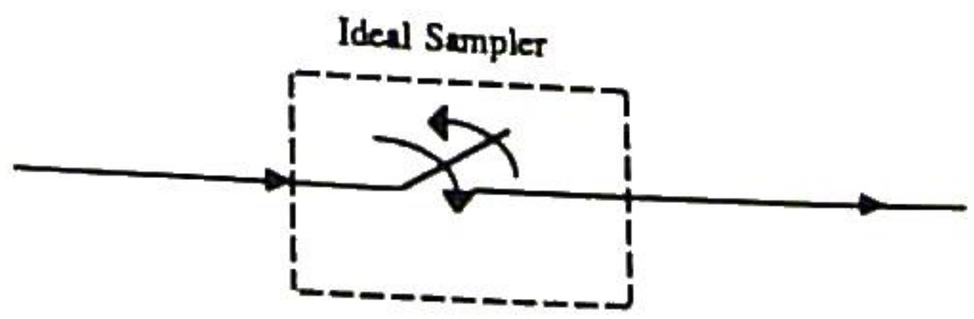
ANALOG SIGNAL SAMPLING

Continuous Time
and
Amplitude

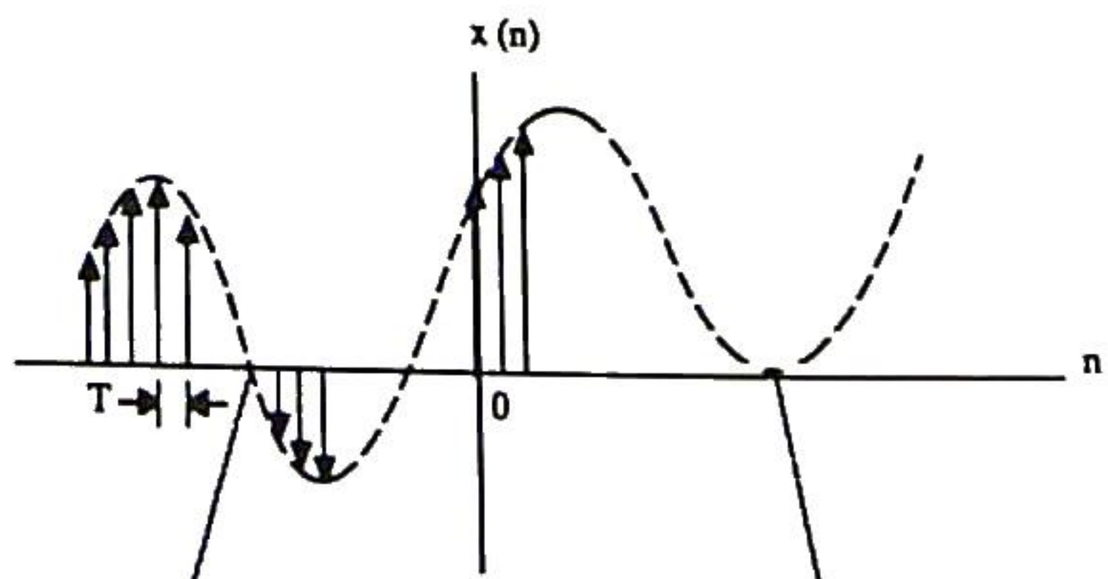


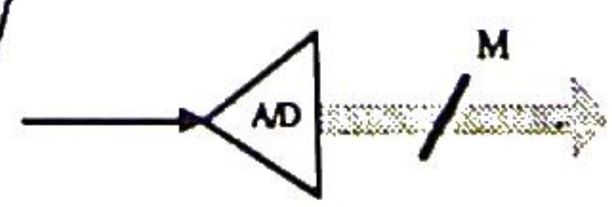
f_s = Sampling Frequency

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

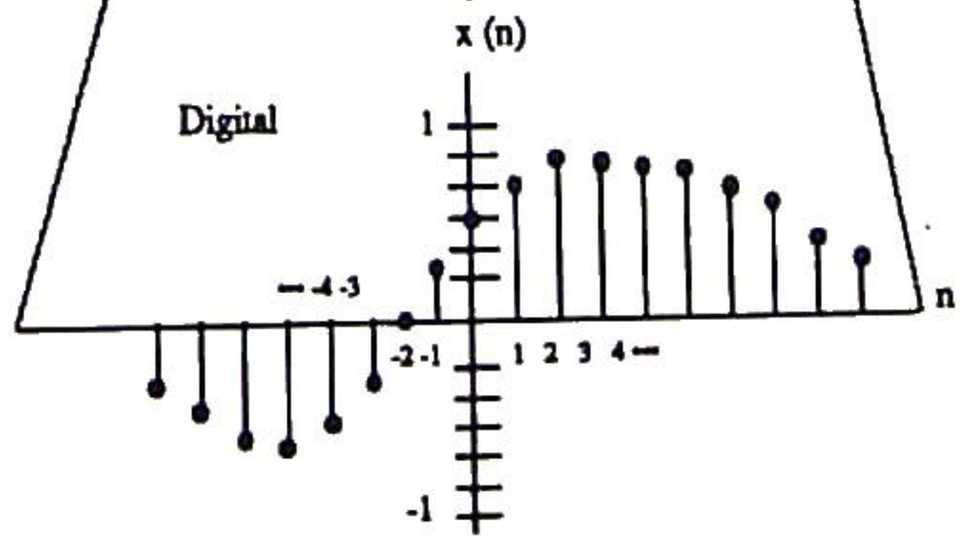


Discrete Time
Continuous
Amplitude



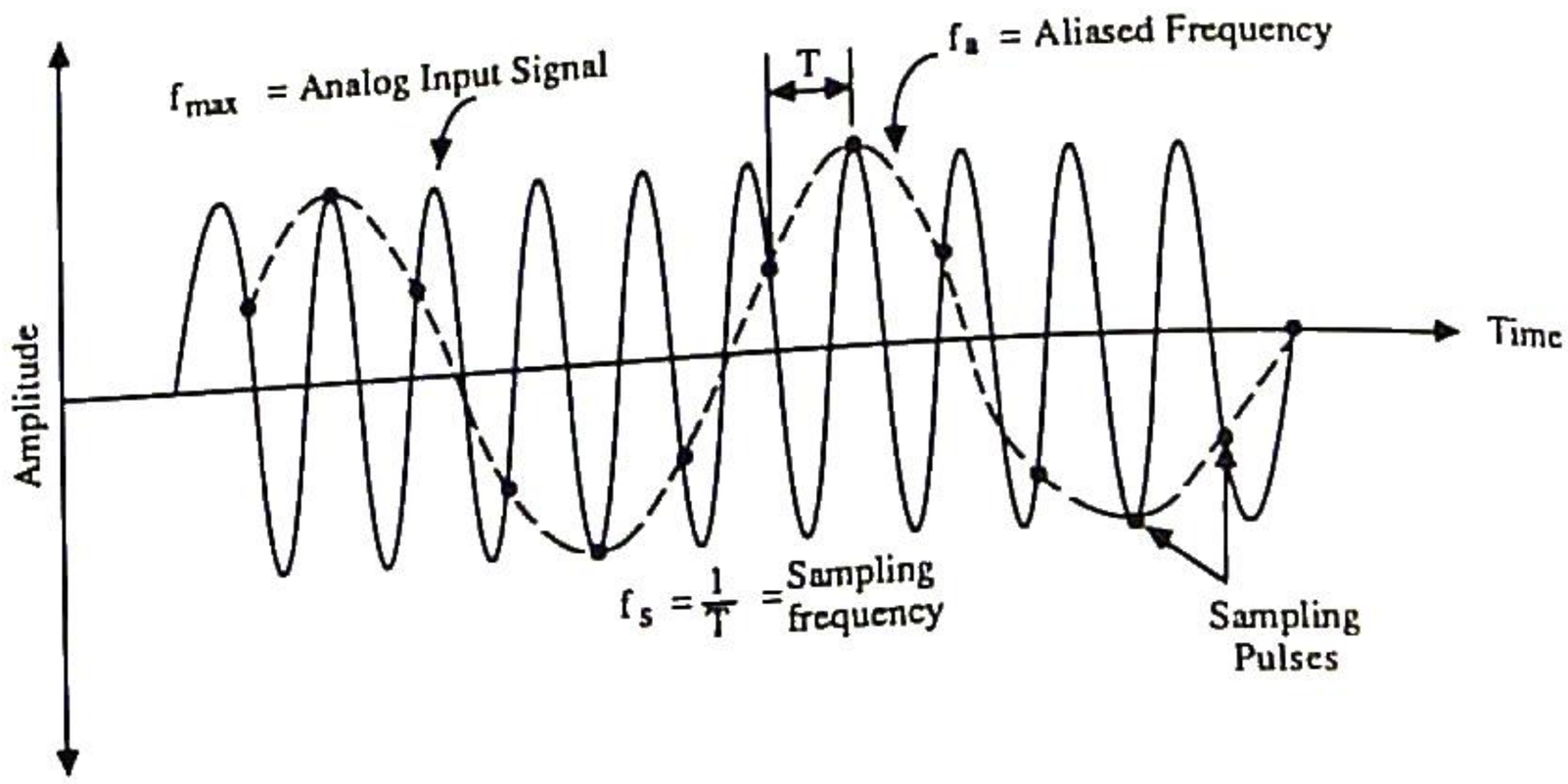


Discrete Time
and
Amplitude



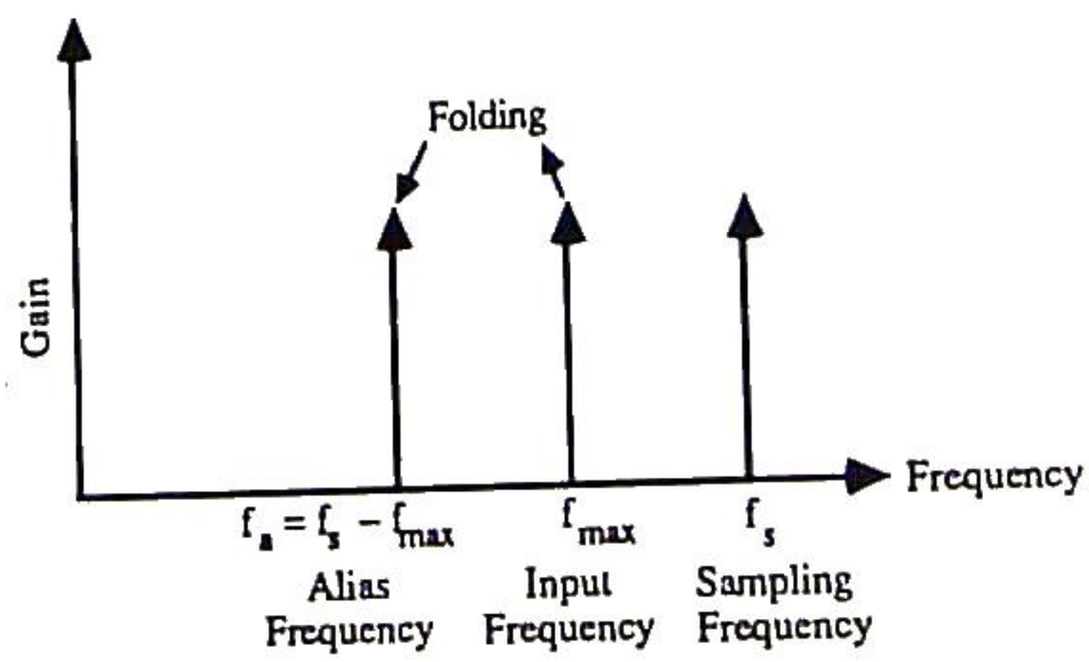
NCD02-0205

ALIASING IN THE TIME DOMAIN



NCD02-0206

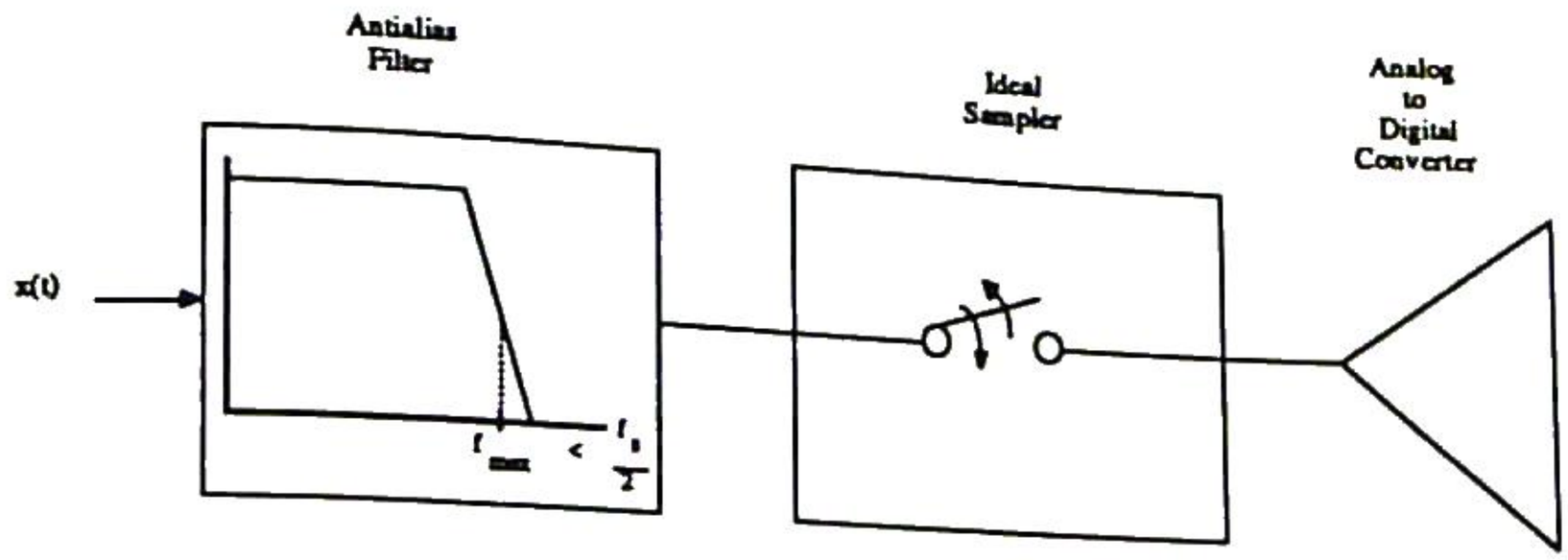
ALIASING IN THE FREQUENCY DOMAIN



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

NCD02-0207-1

BANDLIMITING ANTIALIASING FILTER

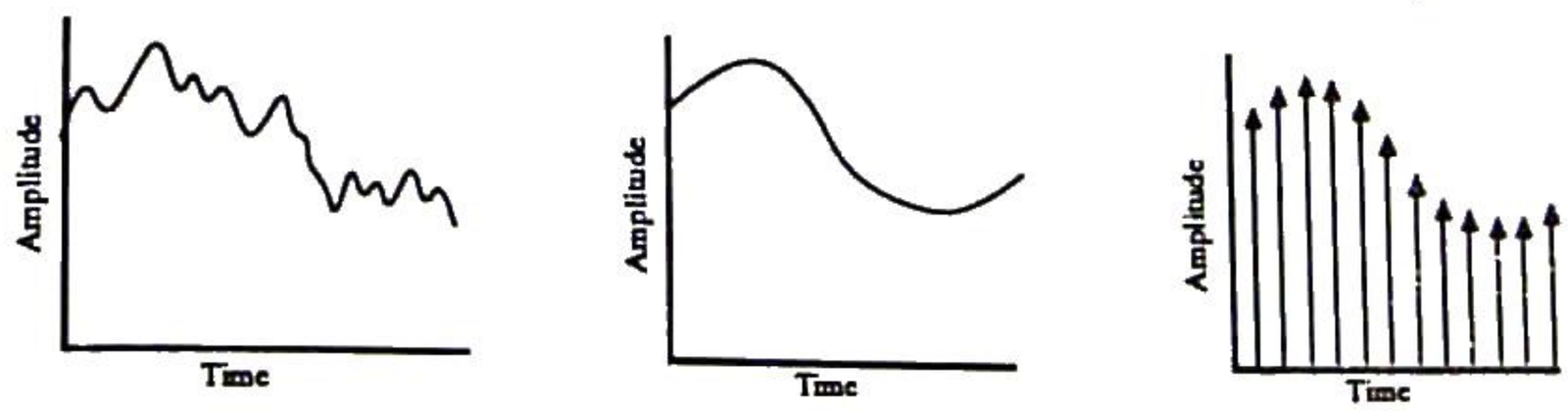


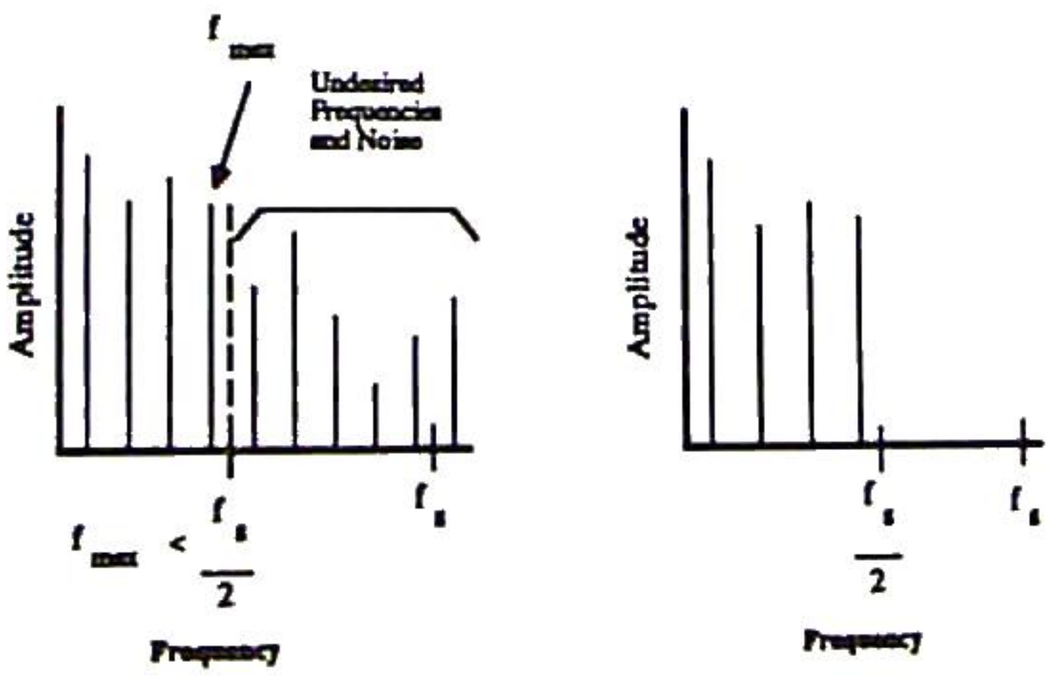
Analog Input with undesired frequencies and noise

Band Limited Analog Input

x(n)

Sampled Analog input

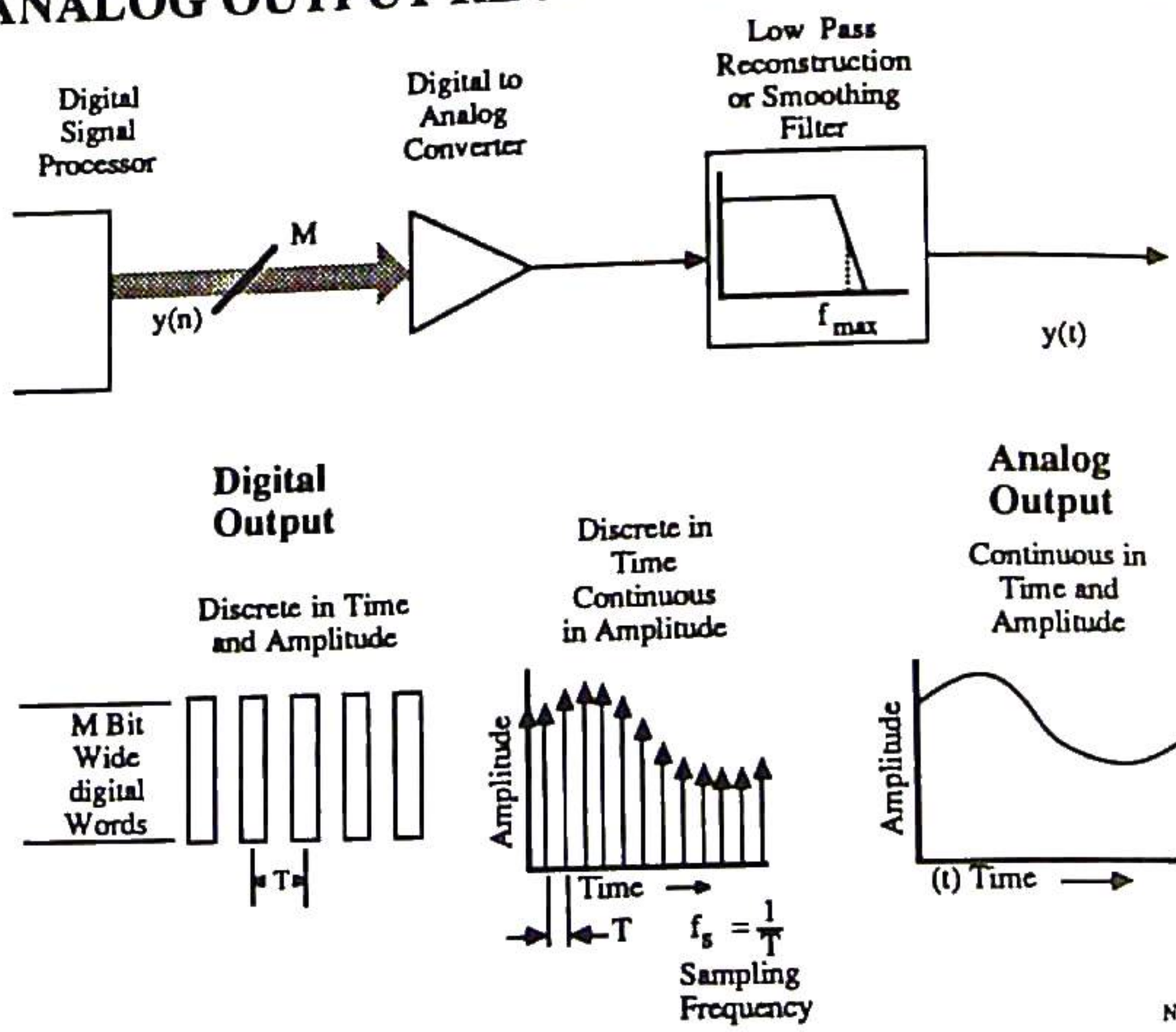




- Solution to aliasing problem:

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

ANALOG OUTPUT RECONSTRUCTION FILTER



NCD02-0209

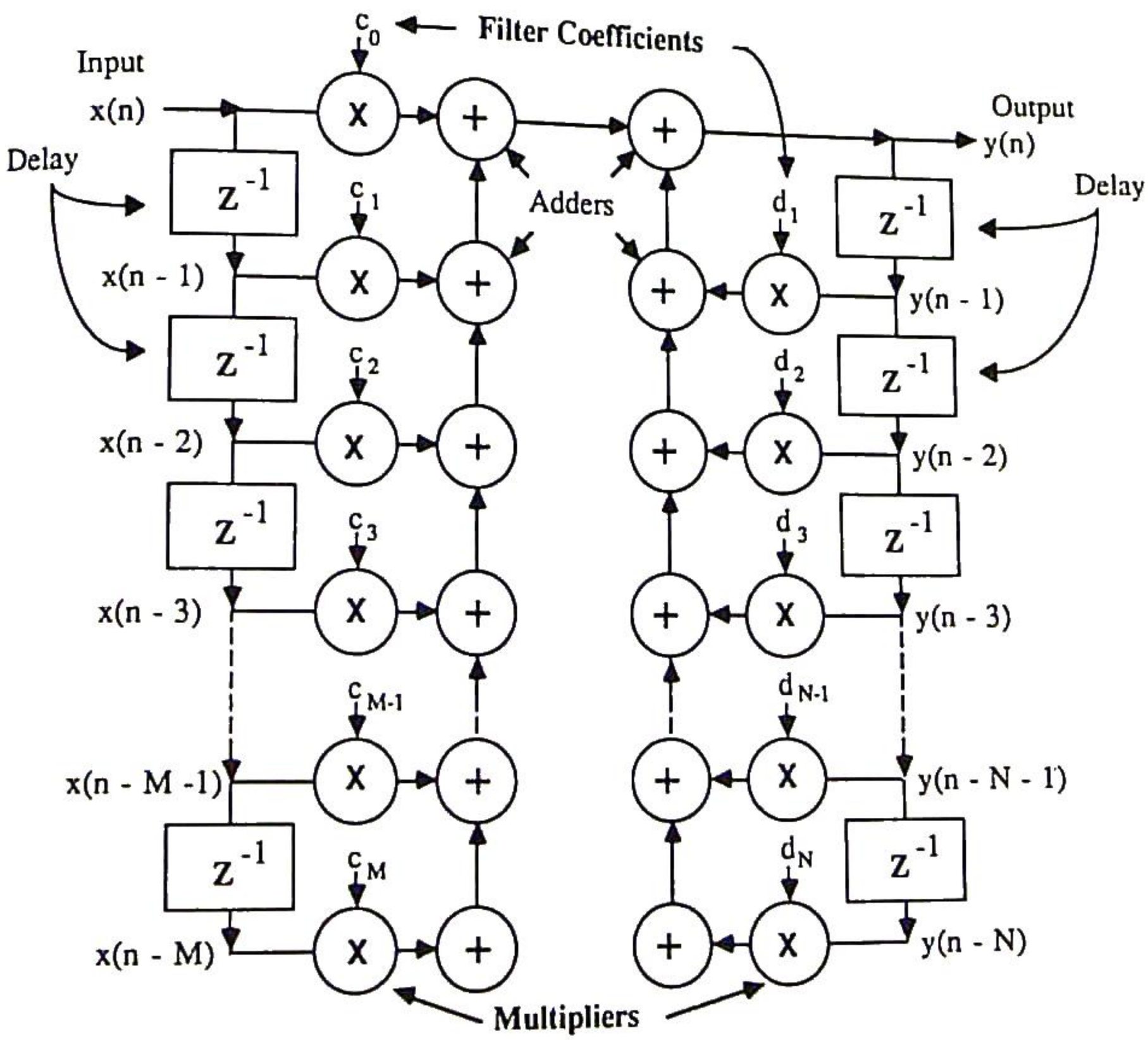
BASIC DIGITAL SIGNAL PROCESSING EQUATION

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

Recursive Filter
Infinite Impulse Response (IIR)

Non Recursive Filter
Finite Impulse Response (FIR)
Zeros

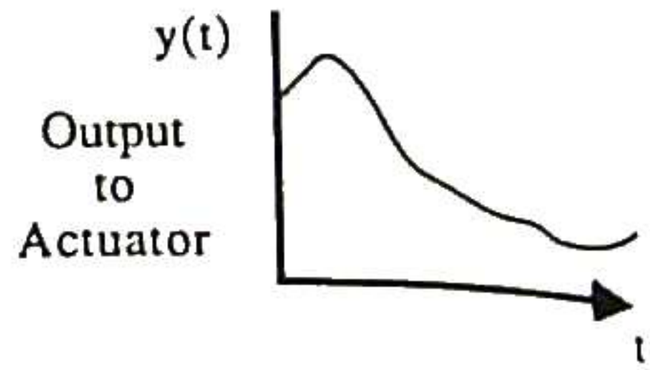
Poles



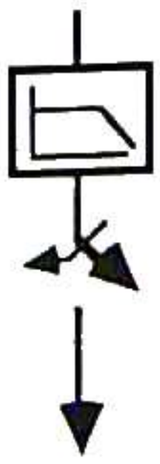
NCD02-0210

DIGITAL SIGNAL PROCESSING SYSTEM
FINITE IMPULSE RESPONSE FILTER

Continuous in time and amplitude



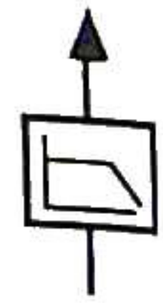
Bandlimiting (Antialias) Filter

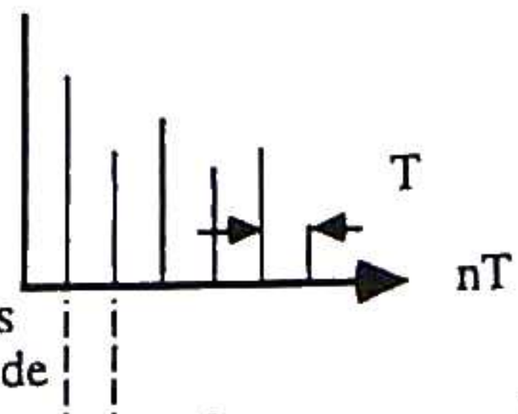


Sample

f_s = Sampling Frequency $= \frac{1}{\text{Sampling Period, T}}$

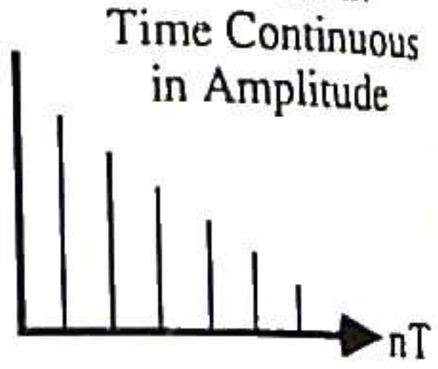
Low Pass (Reconstruction) Filter





Discrete in Time Continuous in Amplitude

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



Discrete in Time Continuous in Amplitude

A/D

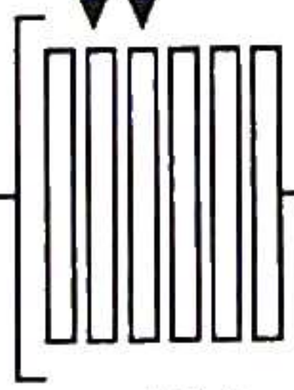
D/A

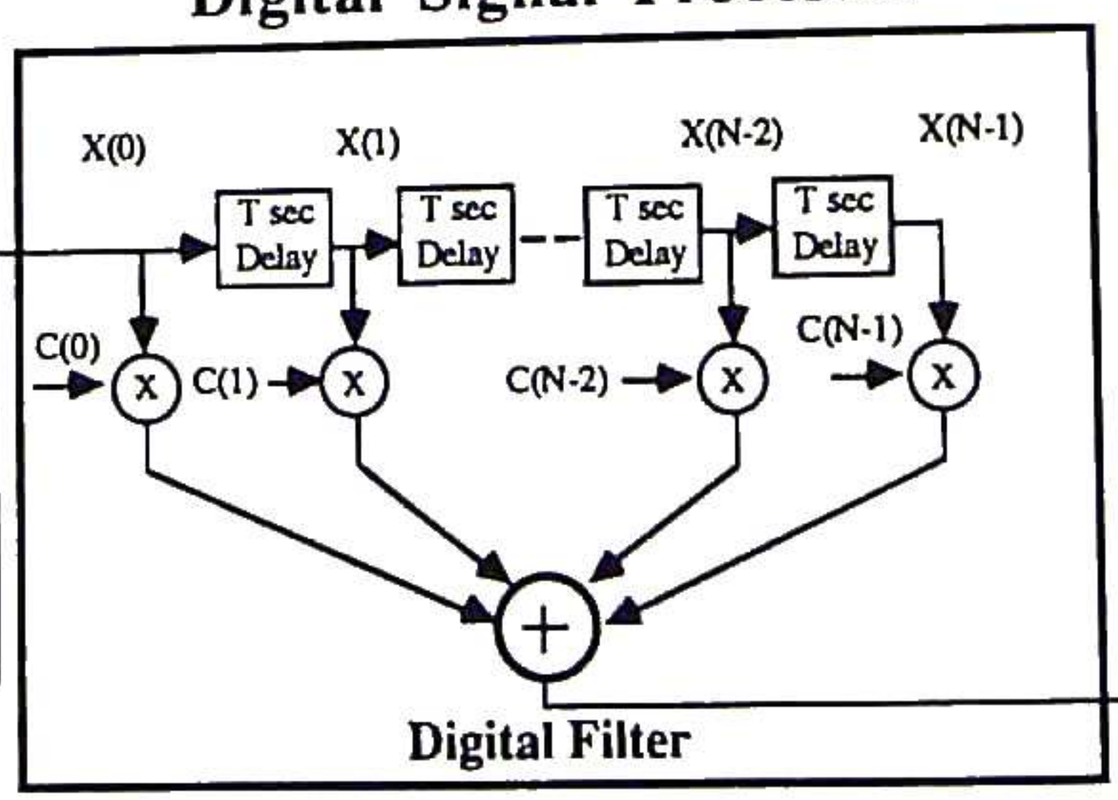
Discrete in Time and Amp.

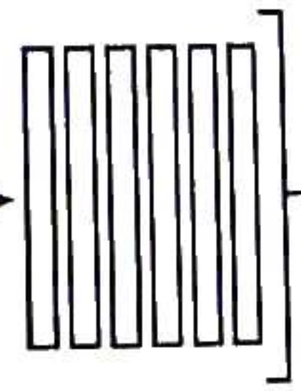
Discrete in Time and Amplitude

12-16 Bit Words

X(n)







12-16 Bit Words

Y(n)

X(k) = Data from A/D stored in RAM
C(k) = Coefficients (determine filter characteristics) stored in RAM or ROM

NCD02-0211-1

FILTER DESIGN (STEPS)

↓

Start the Design

↓

Spec:

Spec the Digital filter
e.g., Low pass filter with a
Cutoff area fc & Pass band
Ripple of 1db or less.

↓

Approximation Problem:

Obtain an Input-Output Characterization
of the filter
[(such as H(2)] that satisfies the spec.

↓

Realization Problem:

Obtain a Realization that defines the internal structure of the filter that has a transfer function H(2). The realization is chosen to optimize the criteria associated with the actual computation.

↓

Implementation:

Determine H/W to Implement the realization so that the complexity is minimized & the data throughput is optimized.

NCD02-0924

DIGITAL FILTERING

ADVANTAGES

- Sharper cutoff
- Better stability over time
- Deterministic
- Adaptive filtering
- Easy to implement

NCD02-0901

DIGITAL FILTER TYPES

- Two Kinds

1. Non - recursive Filter - Finite Inpulse Response (FIR)

$$y(n) = \sum_{k=0}^{N-1} c(k)x(n-k)$$

2. Recursive Filter - Infinite Impulse Response (IIR)

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

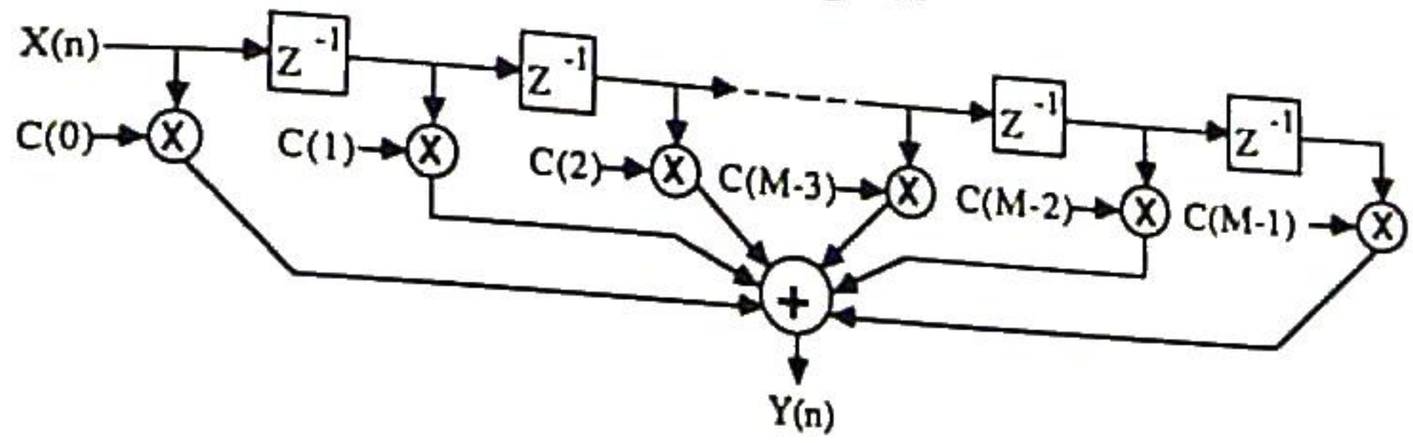
NCD02-0902-1

DIRECT FORM FINITE IMPULSE RESPONSE FILTER

Equation

$$Y(n) = \sum_{K=0}^{M-1} C(K)\, X(n-K)$$

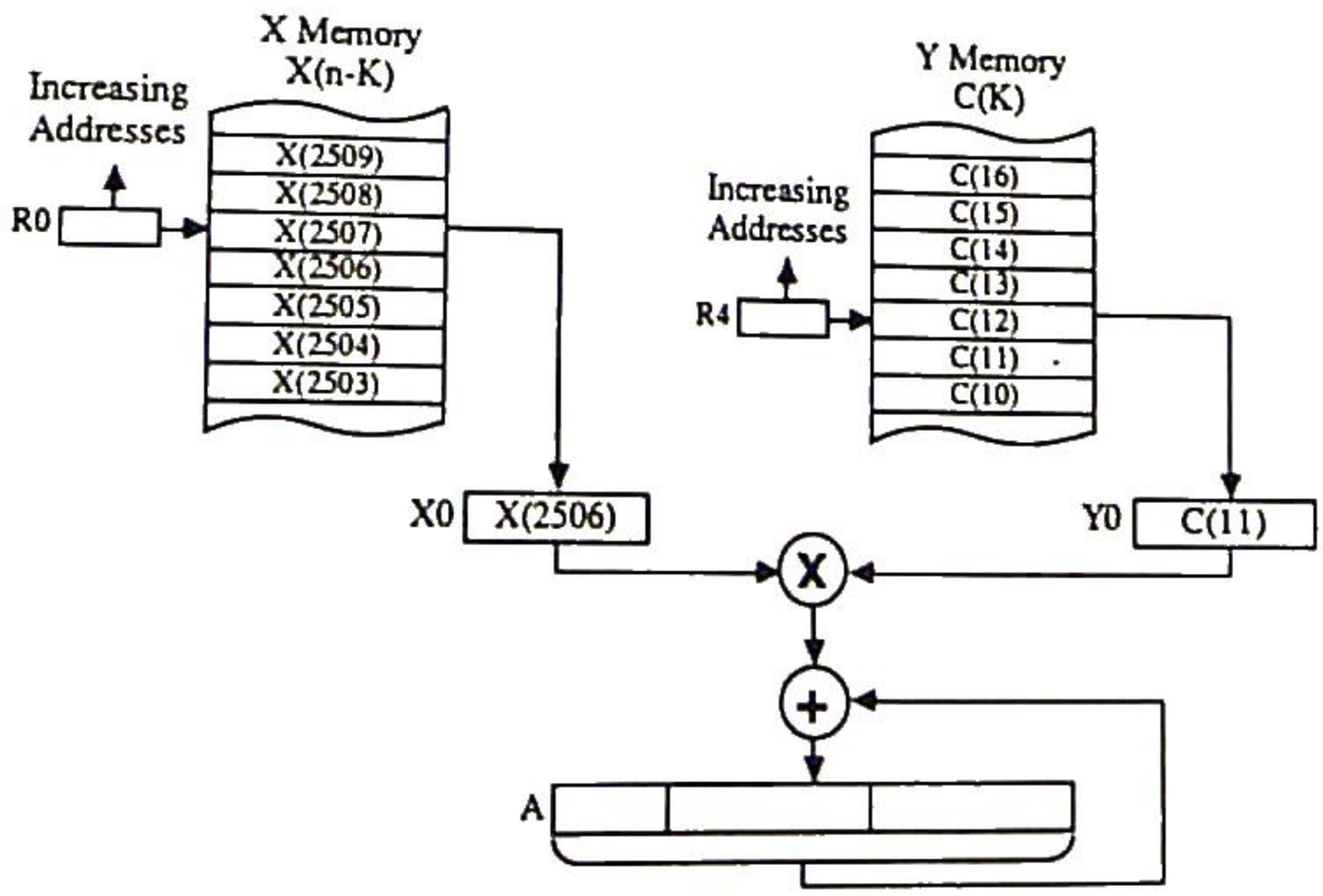
Lattice Diagram



NCD02-0903-1

FIR FILTER REALIZATION

DSP56000 Hardware Implementation



DSP56000 Software Implementation

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

NCD02-0904

FOUR TAP FIR FILTER EXAMPLE

Equation

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

$$Y(n) = C(0)\ X(n) + C(1)\ X(n-1) + C(2)\ X(n-2) + C(3)\ X(n-3)$$

with z transform:

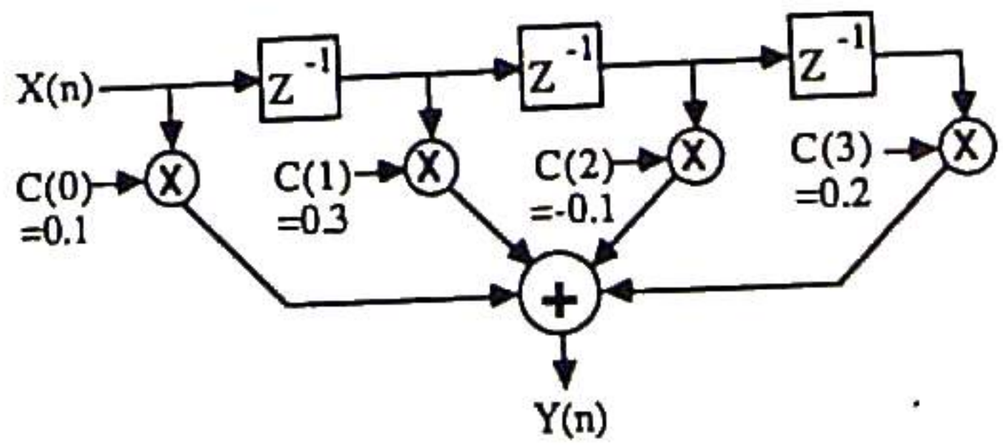
$$\frac{Y(z)}{X(z)} = C(0) + C(1)z^{-1} + C(2)z^{-2} + C(3)z^{-3}$$

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

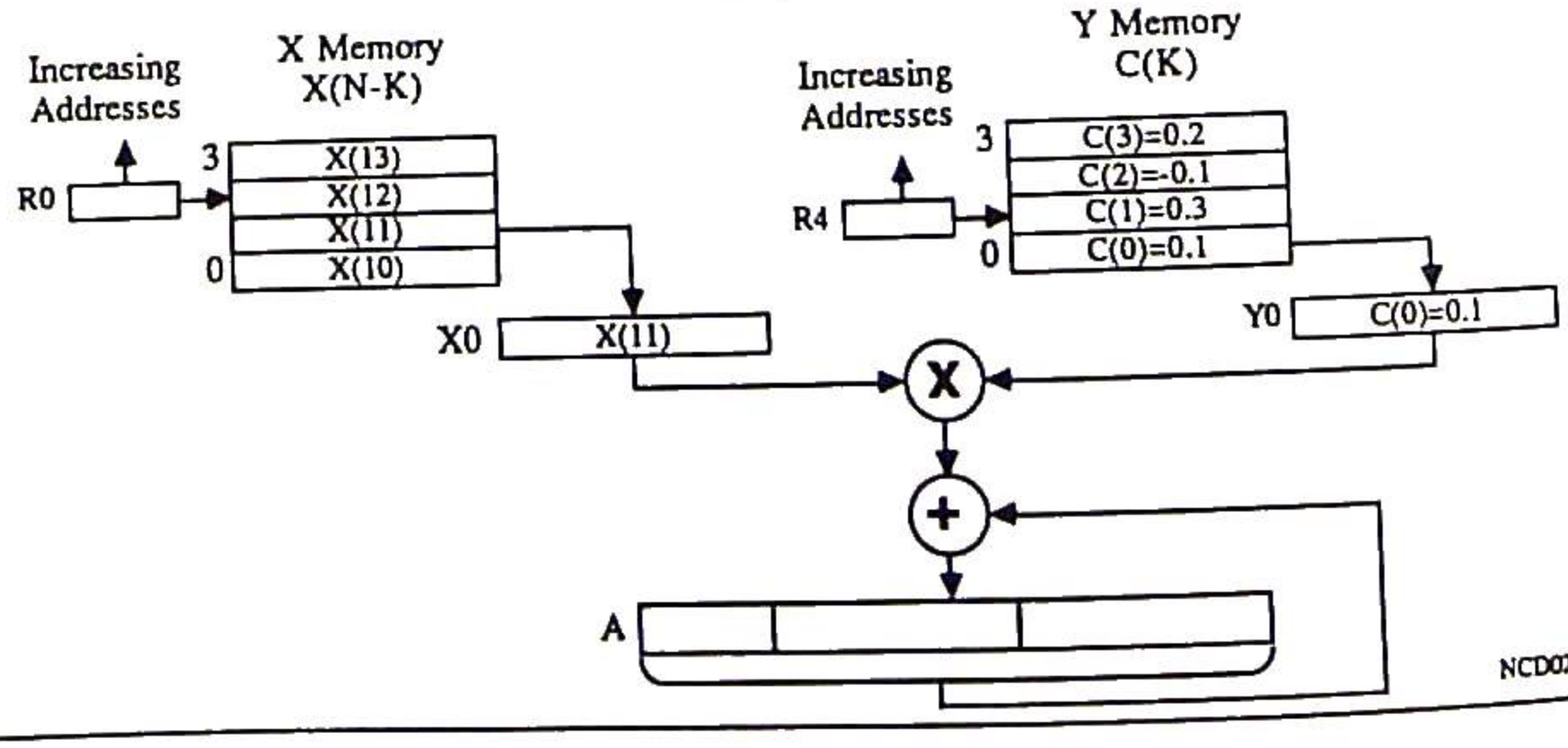
NCD02-0905-1

FOUR TAP FIR FILTER REALIZATION

Lattice Diagram

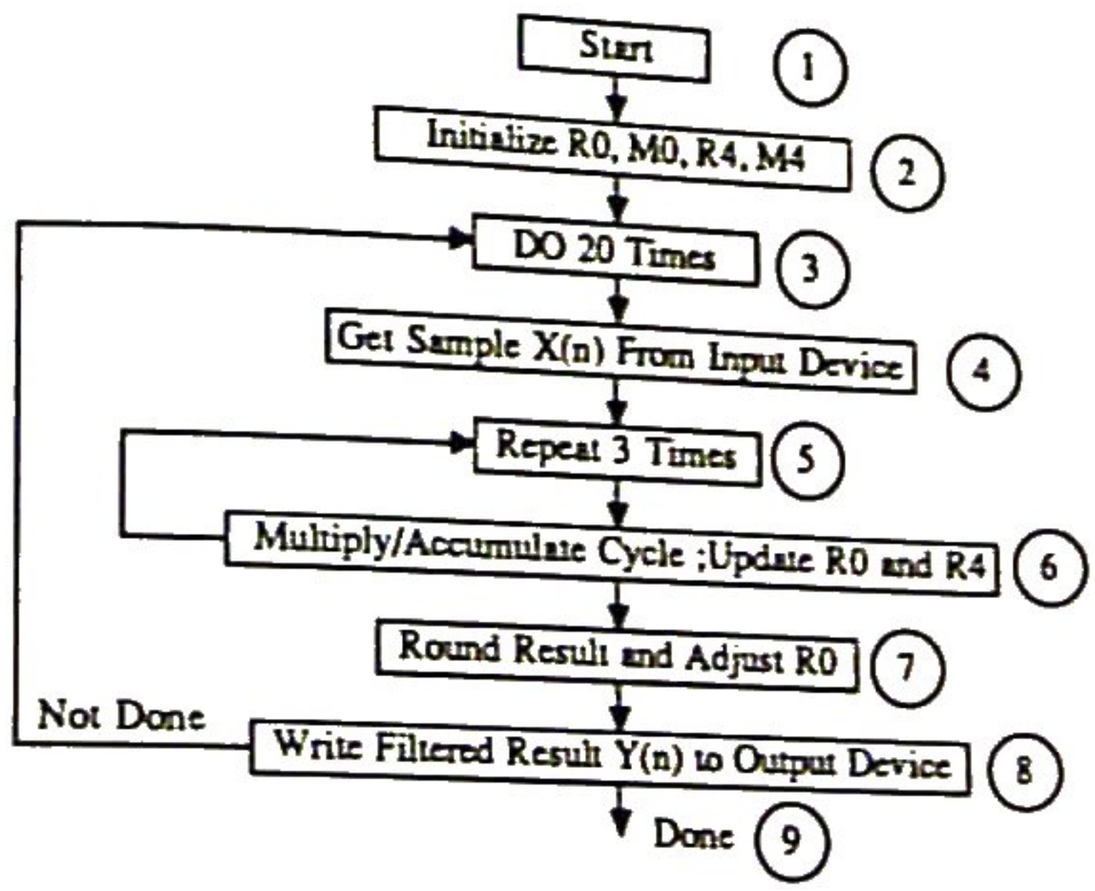


DSP56000 Implementation



NCD02-0906

FOUR TAP FIR FILTER FLOW CHART



NCD02-0907

FOUR TAP FIR FILTER PROGRAM

```
datin     equ     $ffff                                  ;addr. in Y memory of input device
datout    equ     $fffe                                  ;addr. 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                                    ;start address in X memory
states    dsm     ntaps                                  ;define storage for filter states

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

          org     p:$100                                 ;start address in program memory
start  ①                                                 ;initialize parameters
       ②  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)

       ③  do      #npts,_endp                            ;one pass for each sample point

       ④  movep                 y:datin,x0              ;get sample from input device

;
;         Begin four tap FIR filter loop
;
          clr     a             x0,x:(r0)+   y:(r4)+,y0  ;save first state
       ⑤  rep     #ntaps-1                               ;repeat for ntaps-1
       ⑥  mac     x0,y0,a       x:(r0)+,x0   y:(r4)+,y0  ;multiply/accumulate,get nxt data/coef
       ⑦  macr    x0,y0,a       (r0)-                    ;do last tap and round result

       ⑧  movep                 a,y:datout               ;output sample
_endp
       ⑨  end
```

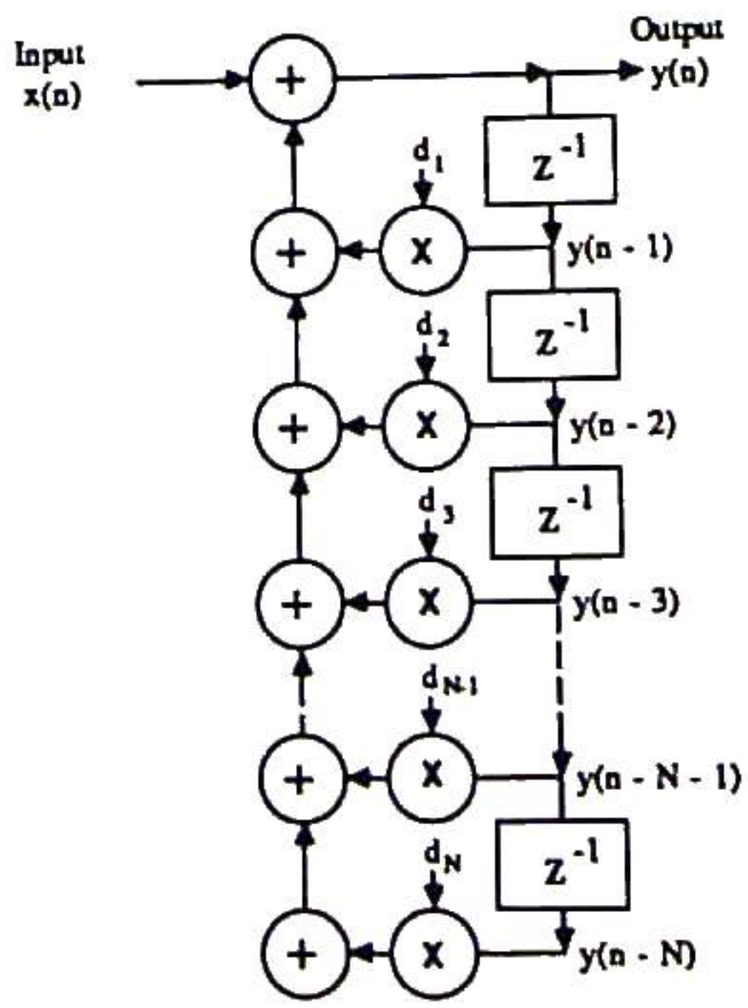
NCD02-0908

INFINITE IMPULSE RESPONSE ALL POLE FILTER

Equation

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

Lattice Diagram



NCD02-0909

FIVE STATE ALL POLE IIR FILTER EXAMPLE

Equation

$$y(n) = x(n) + \sum_{j=1}^{5} D(j)\, Y(n-j)$$

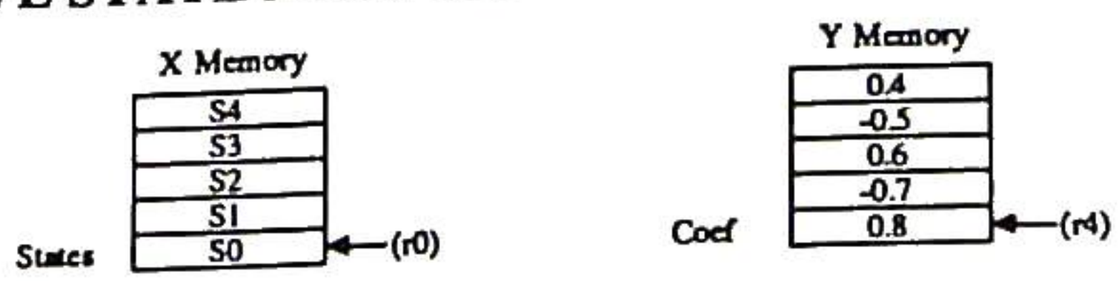
$$Y(n) = x(n) + D(1)Y(n-1) + D(2)Y(n-2) + D(3)Y(n-3) + D(4)Y(n-4) + D(5)Y(n-5)$$

Where:

X(n) = input sample at time nT
Y(n) = output sample at time nT
T = sampling period
D(1) = 0.8
D(2) = 0.7
D(3) = 0.6
D(4) = 0.5
D(5) = 0.4

NCD02-0910

FIVE STATE ALL POLE IIR FILTER PROGRAM

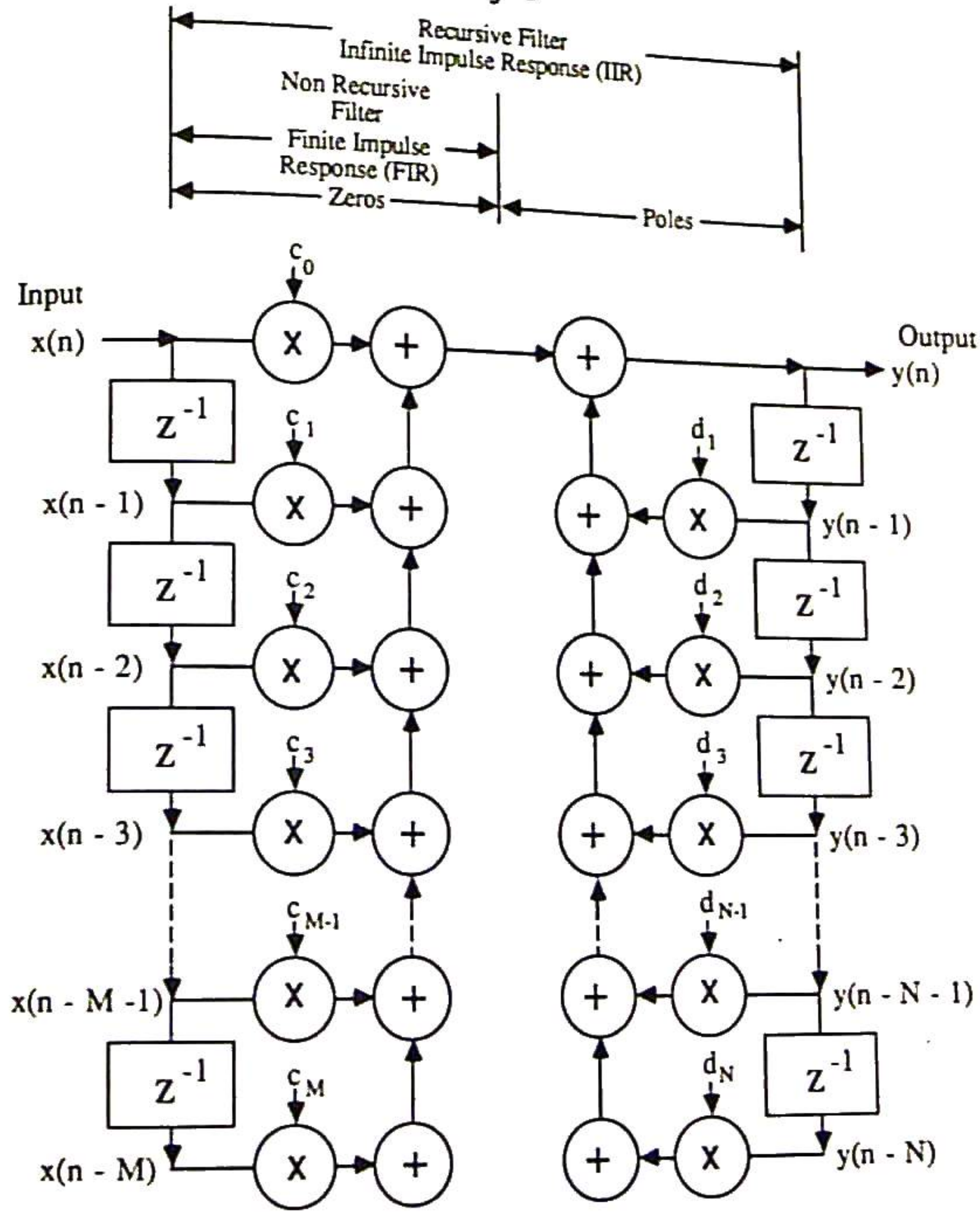



```
datin     equ     $ffff                          ;location in Y memory of input file
datout    equ     $fffe                          ;location in Y memory of output file
npts      equ     20                             ;number of points to process
ncoef     equ     5                              ;number of coefficients

          org     x:0                            ;filter states
states    dsm     ncoef

          org     y:0                            ;coefficients
coef      dc      .8,-.7,.6,-.5,.4

          org     p:$100
start
          move              #states,r0           ;point to filter states
          move              #ncoef-1,m0          ;mod(ncoef)
          move              #coef,r4             ;point to filter coefficients
          move              #ncoef-1,m4          ;mod(ncoef)
sample
          movep             y:datin,a            ;get sample and put in a
;
;         Implements IIR filter with only poles
;
          move              x:(r0)+,x0  y:(r4)+,y0  ;first state, first coef
          rep       #ncoef-1                     ;do macs
          mac       x0,y0,a   x:(r0)+,x0  y:(r4)+,y0
          macr      x0,y0,a   (r0)-                ;rnd, cnt down thru table
          move                a,x:(r0)             ;save output as first state
                                                    for next pass
;
          movep               a,y:datout           ;output sample
          jmp       sample
          end
```

NCD02-0911

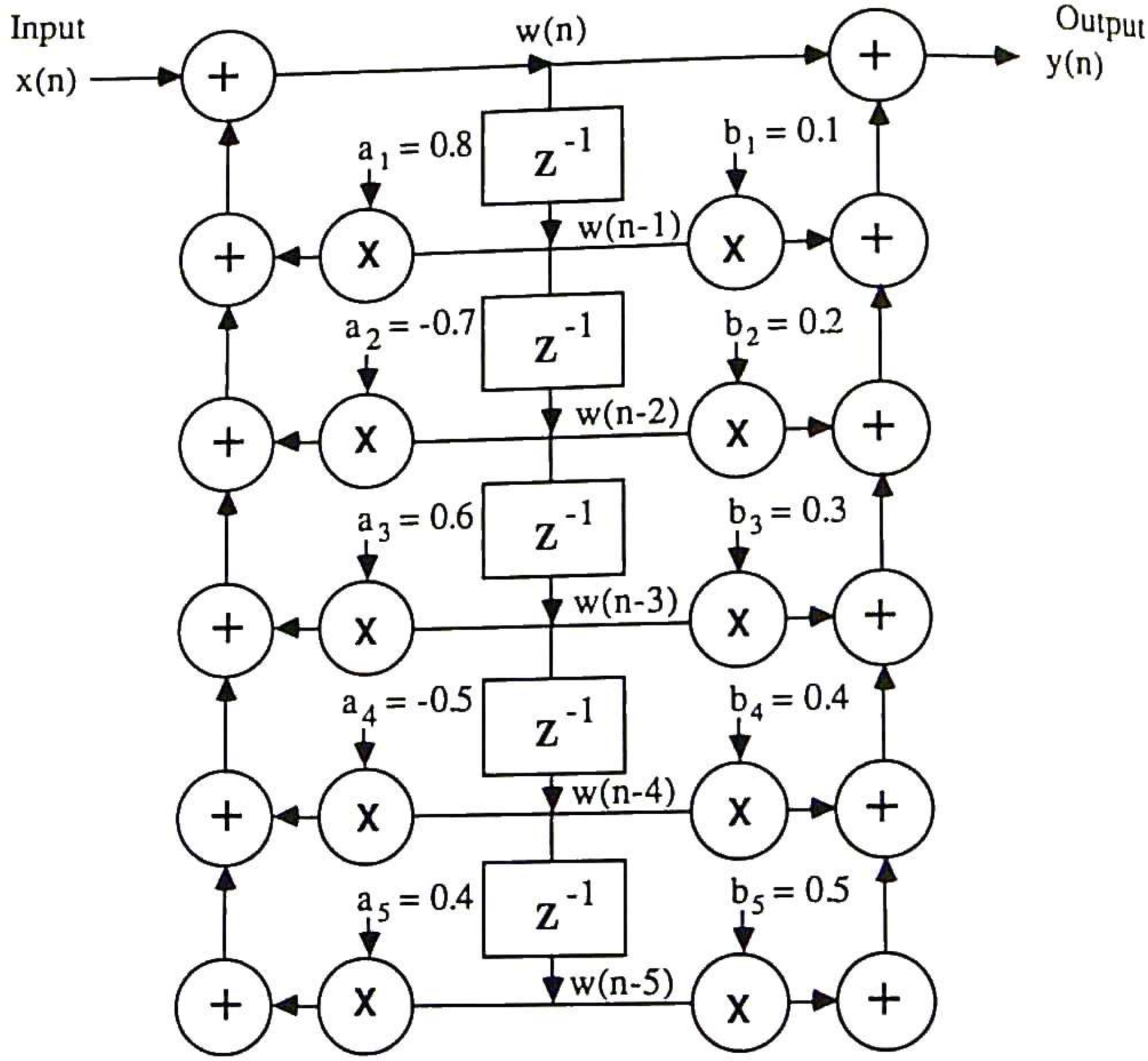
INFINITE IMPULSE RESPONSE FILTER

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

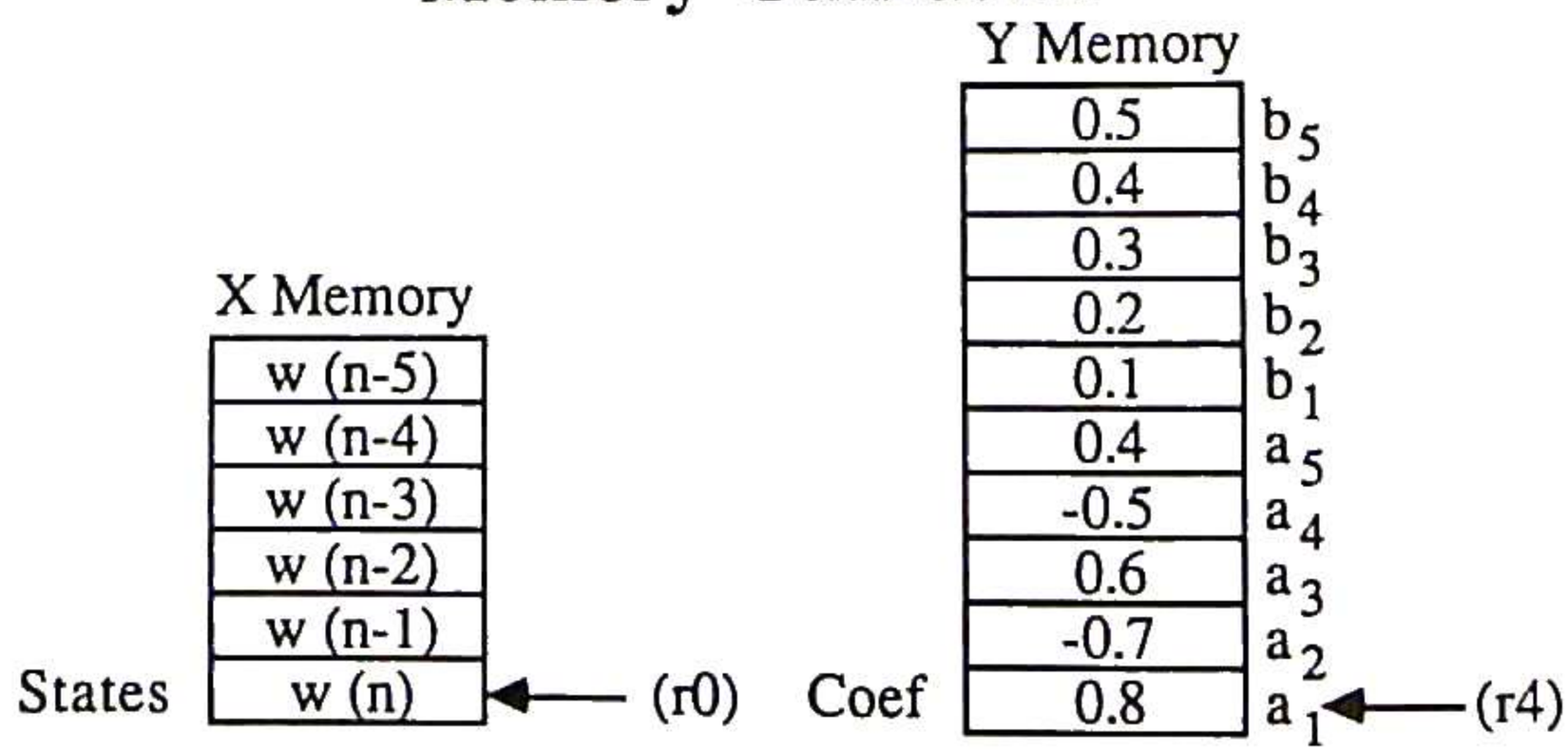


NCD02-0912-1

IIR FILTER DIRECT CANONIC FORM



Memory Utilization



NCD02-0913-1

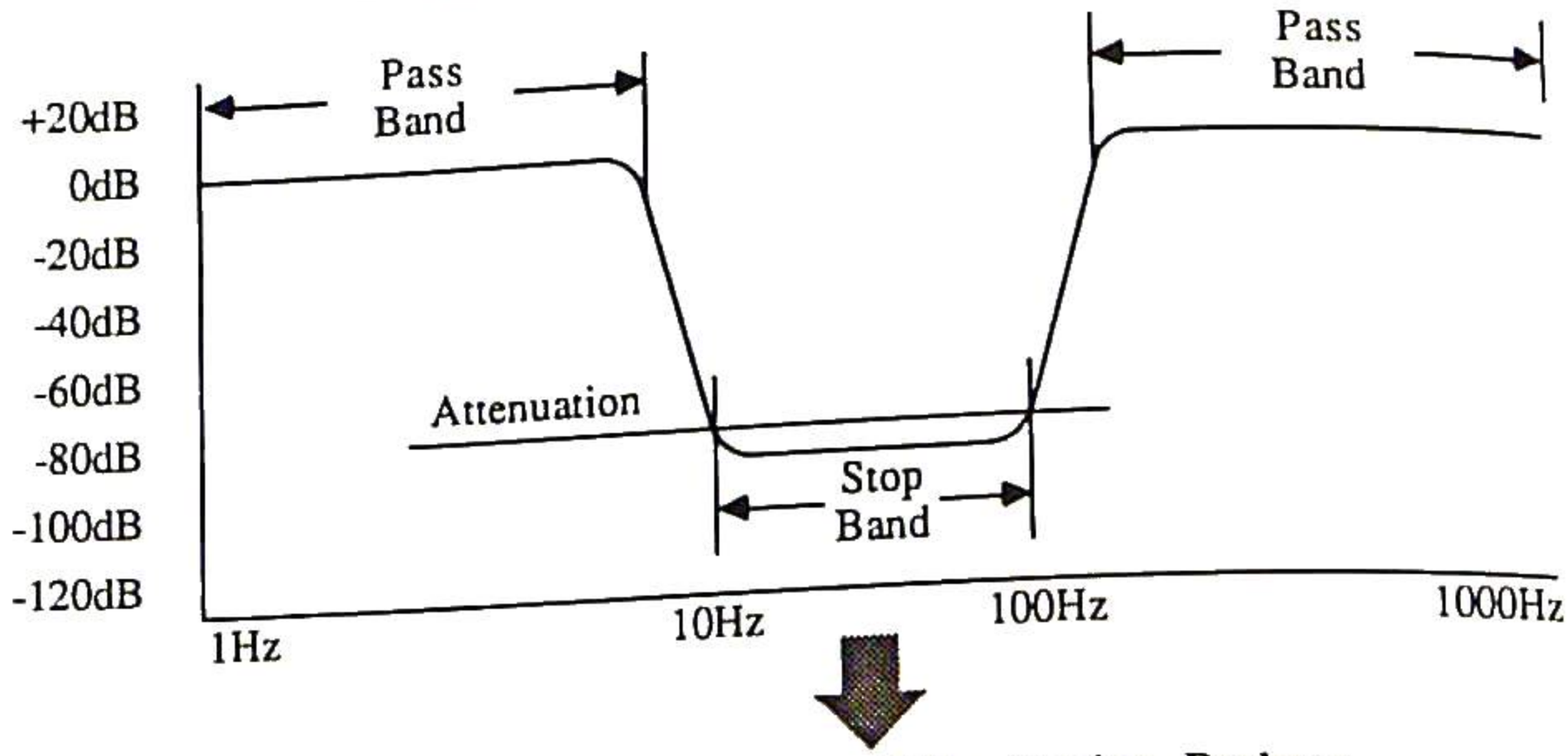
FIFTH ORDER IIR FILTER PROGRAM

```
datin     equ     $ffff
datout    equ     $fffe
npts      equ     20                                  ;location in Y memory of input
nstates   equ     5                                   ;location in Y memory of output
                                                      ;number of points to process
          org     x:0                                 ;number of states
states    dsm     nstates+1

          org     y:0                                 ;filter states
coef      dc      .8,-.7,.6,-.5,.4
          dc      .1,.2,.3,.4,.5                      ;(r4)-> a1,a2,a3,a4,a5
                                                      ;(r4)-> b1,b2,b3,b4,b5
          org     p:$100
start     move                #states,r0
          move                #nstates,m0             ;point to filter states
          move                #coef,r4                ;mod (nstates+1)
          move                #2*nstates-1,m4         ;point to filter coefficients
                                                      ;mod (2*nstates)
          do      #npts,_endp
          movep               y:datin,a
;                                                     ;put sample in accumulator a
;         Implements direct form iir with zeros and poles
;
          move                x:(r0)+,x0  y:(r4)+,y0  ;first state, get a1
          rep     #nstates-1                          ;do iir part
          mac     x0,y0,a     x:(r0)+,x0  y:(r4)+,y0
          macr    x0,y0,a
          move                a,x:(r0)+   y:(r4)+,y0  ;save new s1, get b1
          move                x:(r0)+,x0              ;get old s1
          rep     #nstates-1                          ;do fir part
          mac     x0,y0,a     x:(r0)+,x0  y:(r4)+,y0
          macr    x0,y0,a

          movep   a,y:datout                          ;output sample
          _endp
          end     start
```

NCD02-0914-1

SELECTION OF FILTER COEFFICIENTS

Determine Filter Characteristics



Key Parameters Into Filter Design Package

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

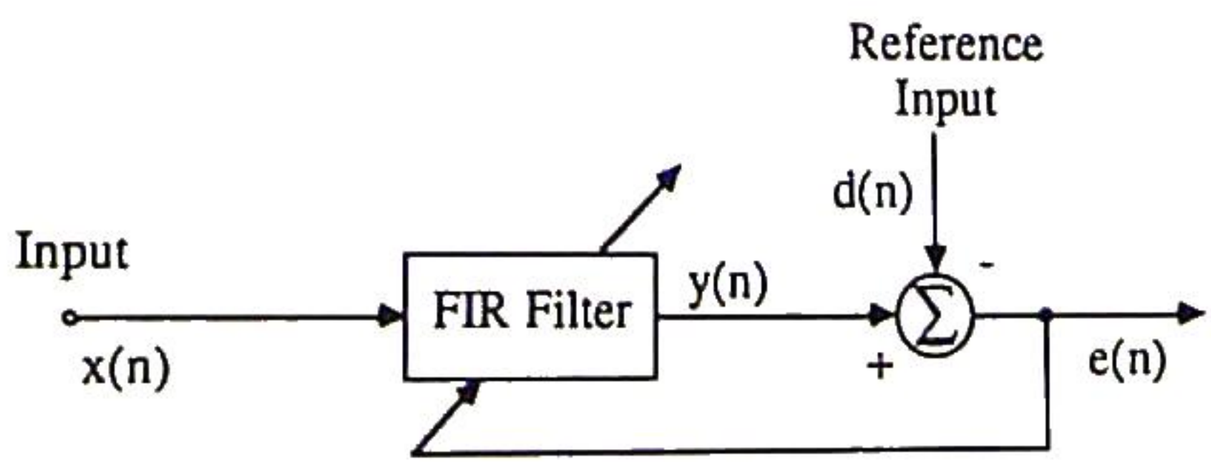
NCD02-0915

ADAPTIVE FILTERING

- What is an adaptive filter?
- A filter that automatically adjusts its parameters (adapts) in accordance with a specific optimality criterion.

NCD02-0916

ADAPTIVE FILTER MODEL



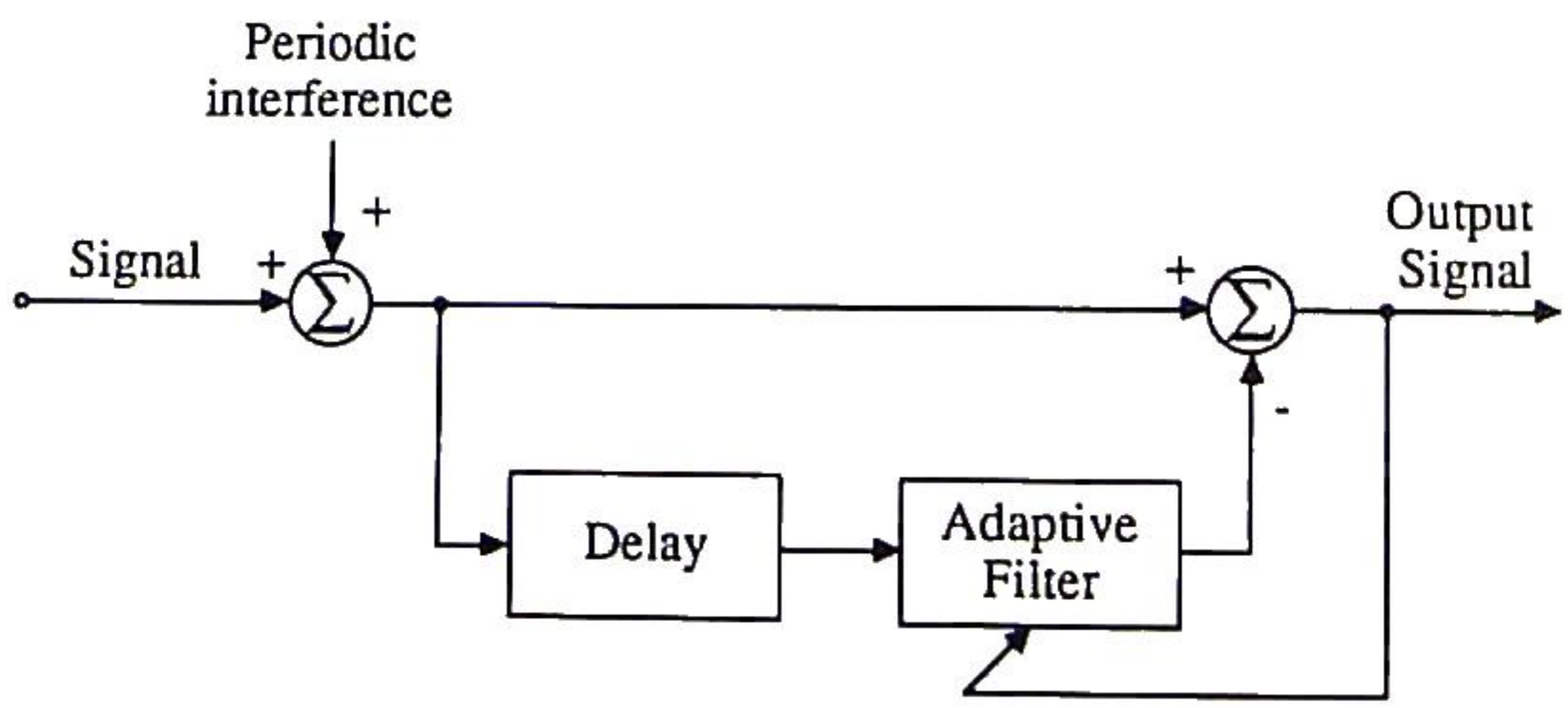
NCD02-0917-1

ADAPTIVE FILTER APPLICATIONS

- Echo Cancellers
- Adaptive Line Equalizers
- Noise Cancelling
- Prediction
- System Identification

NCD02-0918

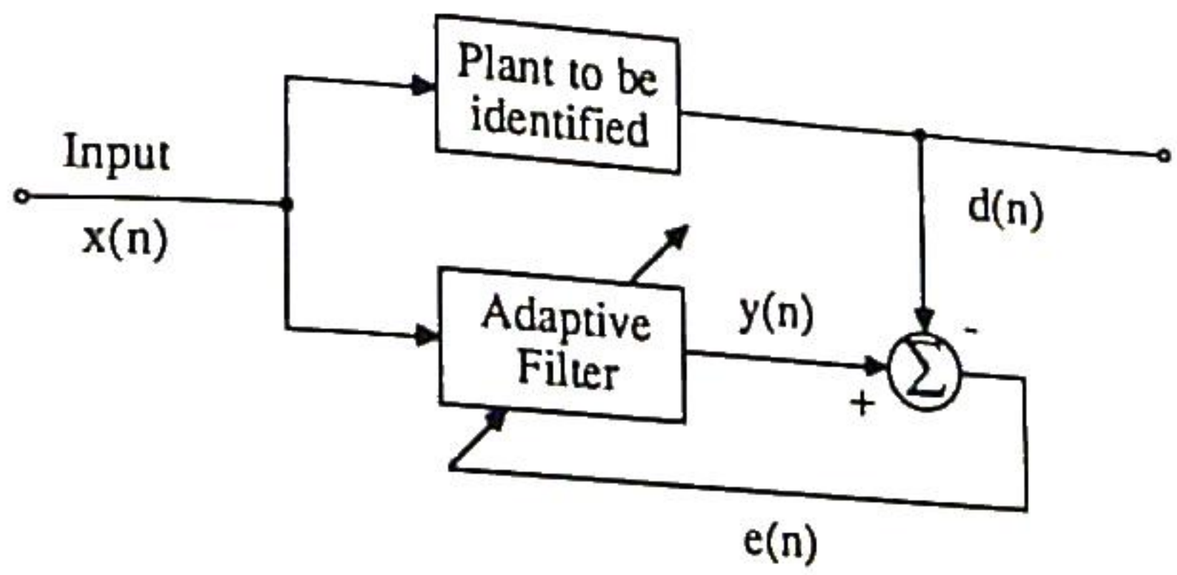
NOISE CANCELLER



NCD02-0919

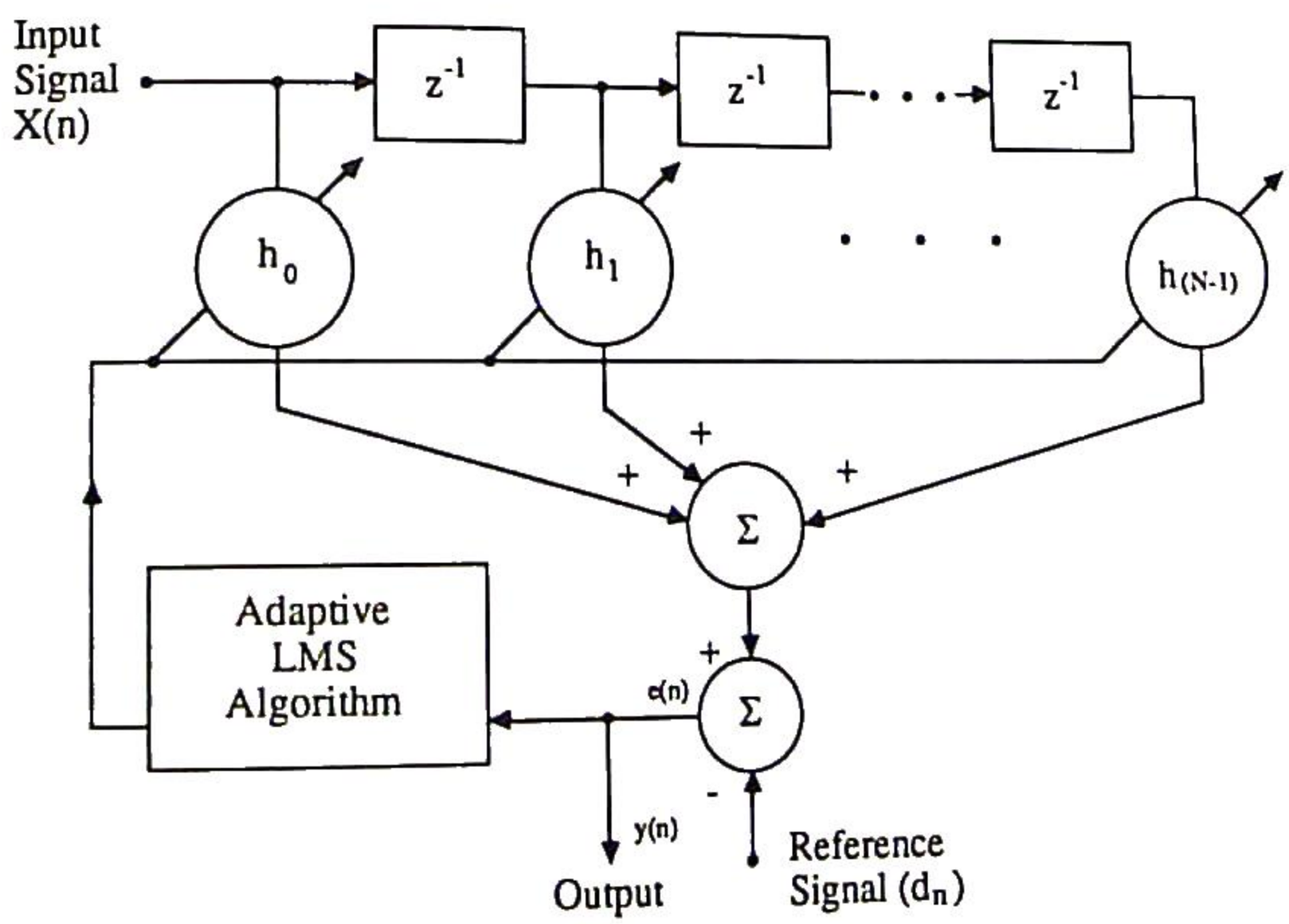
SYSTEM IDENTIFICATION

- Adaptive Control



NCD02-0920-1

ADAPTIVE FILTER LATTICE DIAGRAM



- Coefficient update $h_i(new) = h_i(old) + Ke(n-i)$

NCD02-0921-1

ADAPTIVE FILTER EQUATION

- Error Term Calculation

  Error = Desired Response - Actual Response

  $$E(n) = d(n) - y(n) = d(n) - \sum_{i=0}^{N-1} h(i)\, x(n-i)$$

- Coefficient Update

  $$h(i)_{new} = h(i)_{old} + Ke\, x(n-i)$$

NCD02-0922

ADAPTIVE FIR FILTER PROGRAM

```
datin     equ     $ffff                                  ;location in Y memory of input
datout    equ     $fffe                                  ;location in Y memory of output
refin     equ     $fffd                                  ;reference input from Y memory
ntaps     equ     4                                      ;number of taps in filter
          org     x:0
states    dsm     ntaps                                  ;filter states
gain      dc      $006000                                ;K factor
          org     y:0
coef      dsm     ntaps                                  ;coefficients
          org     p:$100
start     move                #states,r0                 ;point to filter states
          move                #ntaps-1,m0                ;mod(ntaps)
          move                #coef,r4                   ;point to filter coefficients
          move                #ntaps-1,m4                ;mod(ntaps)
          move                #-2,n0                     ;used for pointer adjustment
sample    movep               y:datin,x0                 ;get sample
          movep               y:refin,b                  ;get reference input
;       direct form adaptive FIR filter
          clr     a           x0,x:(r0)+  y:(r4)+,y0     ;first do FIR filter
          rep     #ntaps-1
          mac     x0,y0,a     x:(r0)+,x0  y:(r4)+,y0
          macr    x0,y0,a
          movep               a,y:datout                 ;output sample
;       update coefficients with LMS algorhthm
          sub     a,b                                    ;error signal d(n)-FIR in b
          move                x:gain,x1   b,y1           ;gain in x1, error in y1
          mpyr    x1,y1,a     x:(r0)+,x0  y:(r4)+,y0     ;K*e in a, get x(n-i) & h(i)old
          tfr     y0,a        a,x1                       ;put h(i)old in a, put K*e in x1
          do      #ntaps,aend                            ;update each coefficient
          macr    x0,x1,a     x:(r0)+,x0  y:(r4),y0      ;h(i)new, get x(n-i+1) & h(i+1)old
          tfr     y0,a                    a,y:(r4)+      ;put h(i+1)old in a,store h(i) new
aend      move                x:(r0)+n0,x0 y:(r4)-,y0    ;dummy move for pointer adjustment
          jmp     sample
          end
```

NCD02-0923

DSP56200
CASCADABLE ADAPTIVE FINITE IMPULSE RESPONSE DIGITAL FILTER (CAFIR)

Objectives:

- Describe DSP56200 Host Transfers.
- Describe the dual FIR mode or the DSP56200.
- Describe the adaptive FIR mode of the DSP56200.

NCD02-1100-1

Table of Contents

DSP56200 Features
Pinout 11-5
Internal Architecture 11-6
Single FIR Mode 11-7
MC68000 to DSP56200 Host Interface 11-8
DSP56200 Register Model (1 of 3) 11-8
DSP56200 Register Model (2 of 3) 11-9
DSP56200 Register Model (3 of 3) 11-9
DSP56200 Operation 11-10
Host Interface Transfers 11-11
Host Interface Transfer Examples 11-12
Dual FIR Mode 11-12
Dual FIR Register Model 11-13
Dual FIR Operation 11-14
Adaptive FIR Filter Mode 11-15
Adaptive FIR System Example 11-16
Coefficient Update 11-17
Adaptive Filter Register Model 11-18
Leakage 11-19
DC Tap 11-19
Cascading 11-20
Cascade FIR Filter 11-20
Cascading DSP56200's 11-21
Speakerphone 11-21
Adaptive Echo Canceller 11-22
DSP56200 Performance 11-23

NCD02-1199

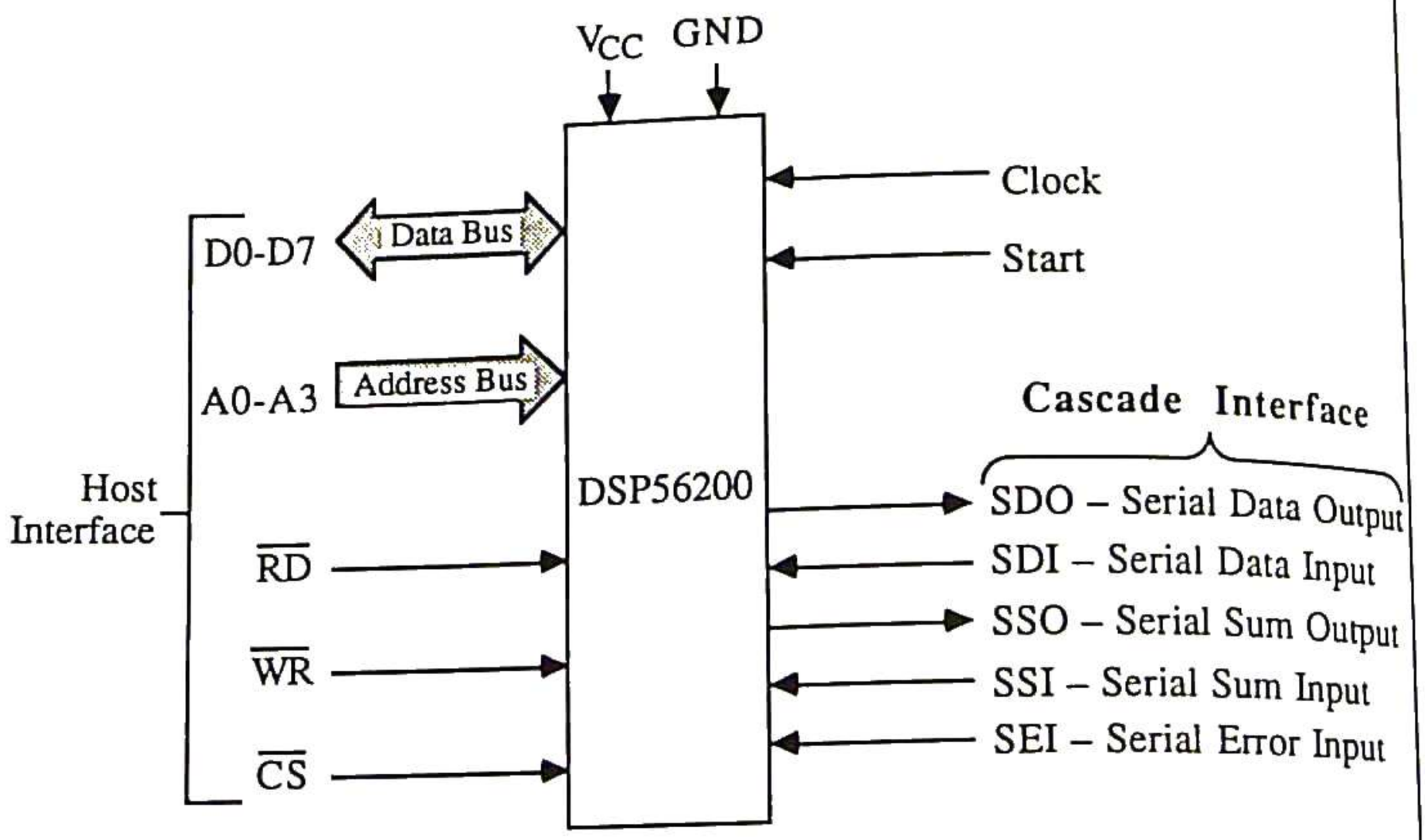
DSP56200 FEATURES

- Three Operational Modes
  - Single Channel FIR Filter
  - Dual Channel FIR Filter
  - Single Channel Adaptive FIR Filter with DC Tap and Leakage Control
- High Performance Architecture Optimized for Digital Filtering
  - 24 X 16 Multiplication with 40-Bit Accumulation
  - 97.5 Nanosecond Cycle Time
  - Static 256 x 24-Bit Coefficient RAM
  - Static 256 x 16-Bit Data RAM
  - Low-Power Standby Mode
  - Three Execution Units Operate in Parallel
  - Multiple Internal Buses
- Cascadable
  - Longer Tap-Length Filters
  - Higher Sampling Rates
- High Data Throughput Rates
  - 227 kHz FIR Filter, 32 Taps (Single Chip)
  - 37 kHz FIR Filter, 256 Taps (Single Chip)
  - 115 kHz Adaptive Filter, 256 Taps (8 Cascaded Chips)
  - Many Other Configurations Possible
- Simple Memory-Mapped Interface to Popular Hosts
  - Microprocessors and Microcomputers
  - General Purpose Digital Signal Processors

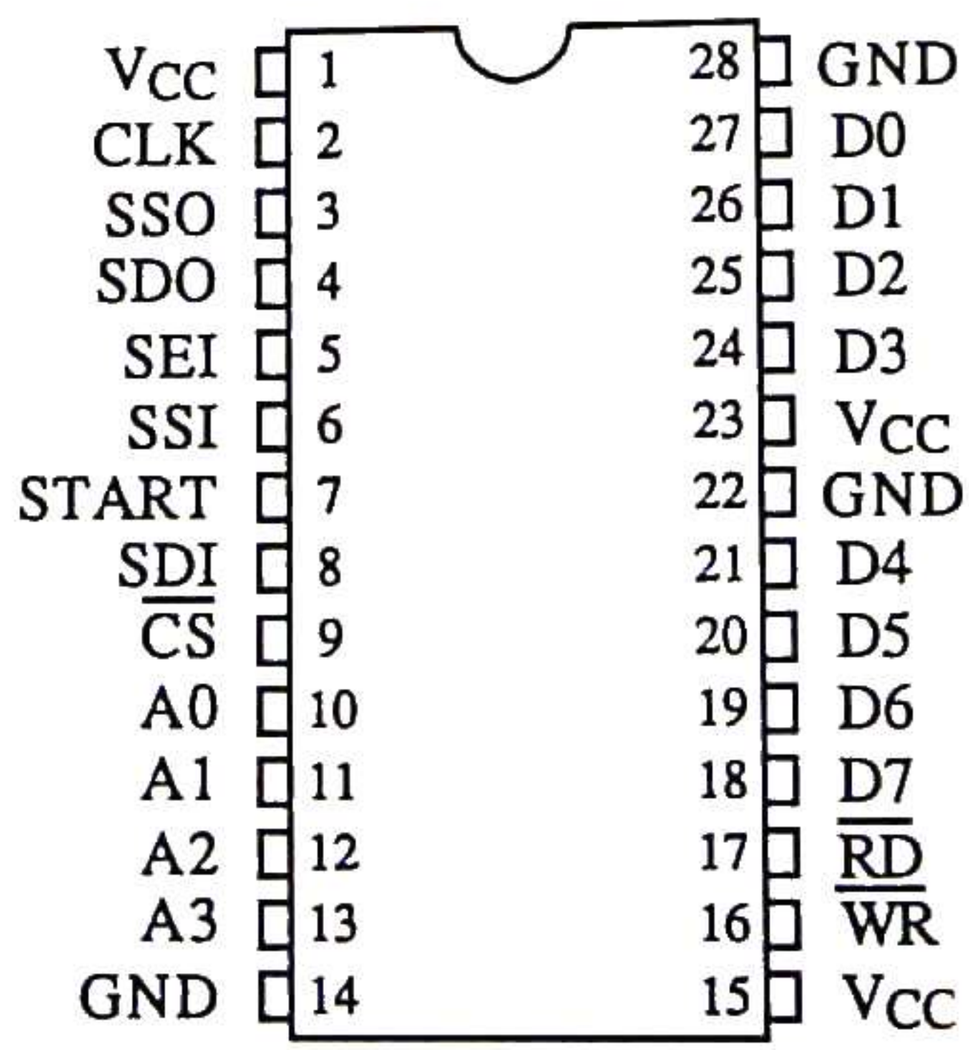
NCD02-1101

PINOUT

Functional Pinout

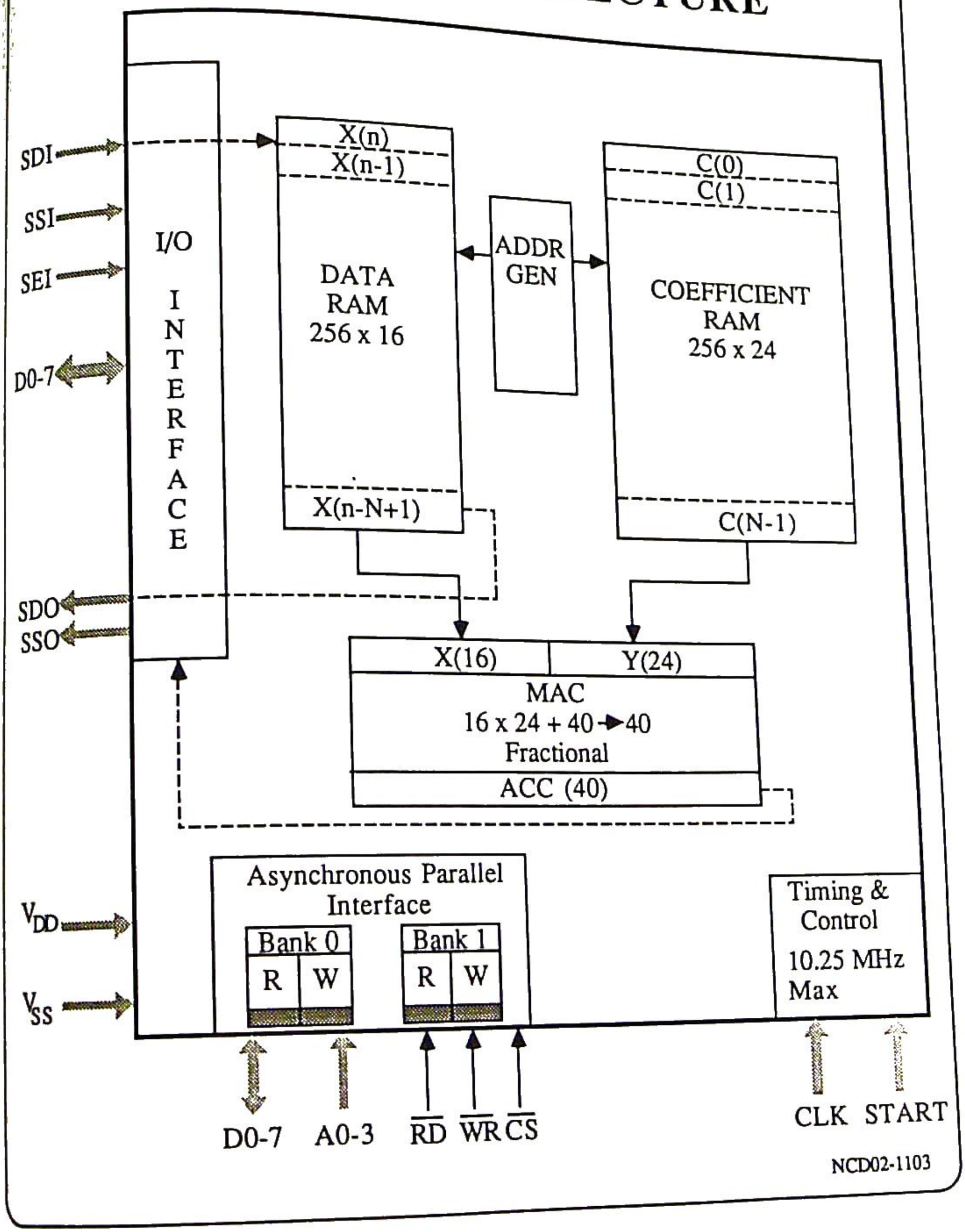


Pin Assignment

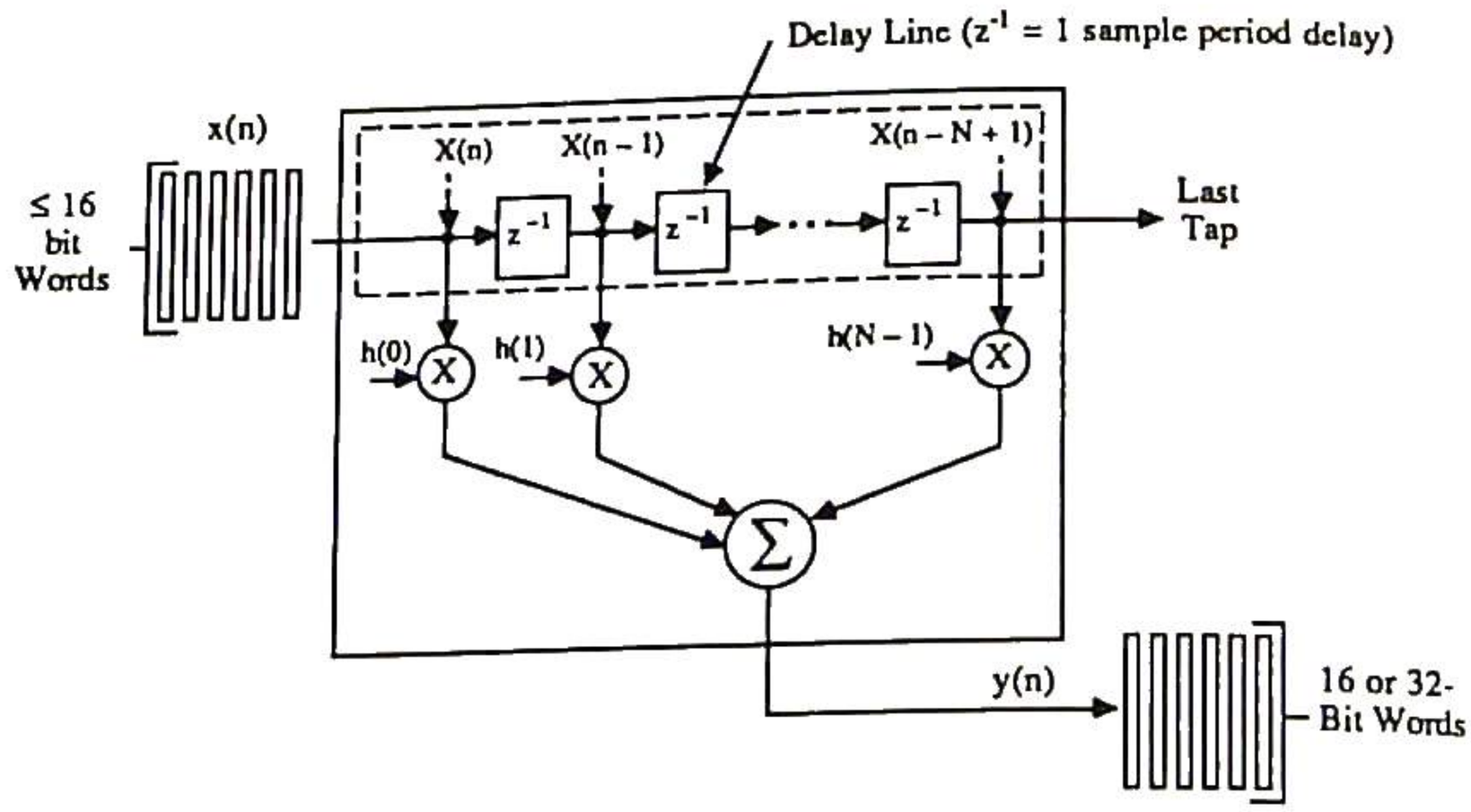


NCD02-1102

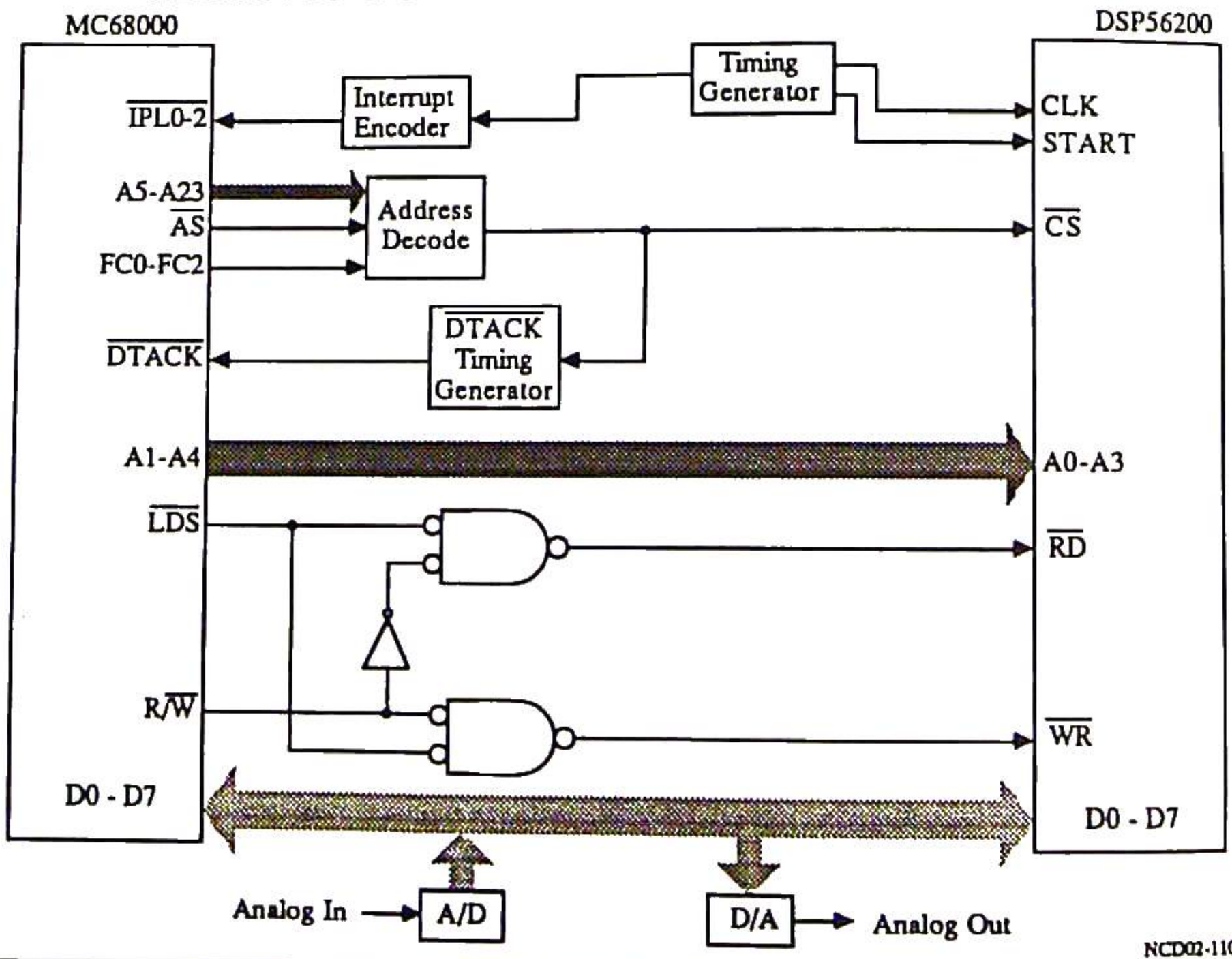
INTERNAL ARCHITECTURE



SINGLE FIR MODE

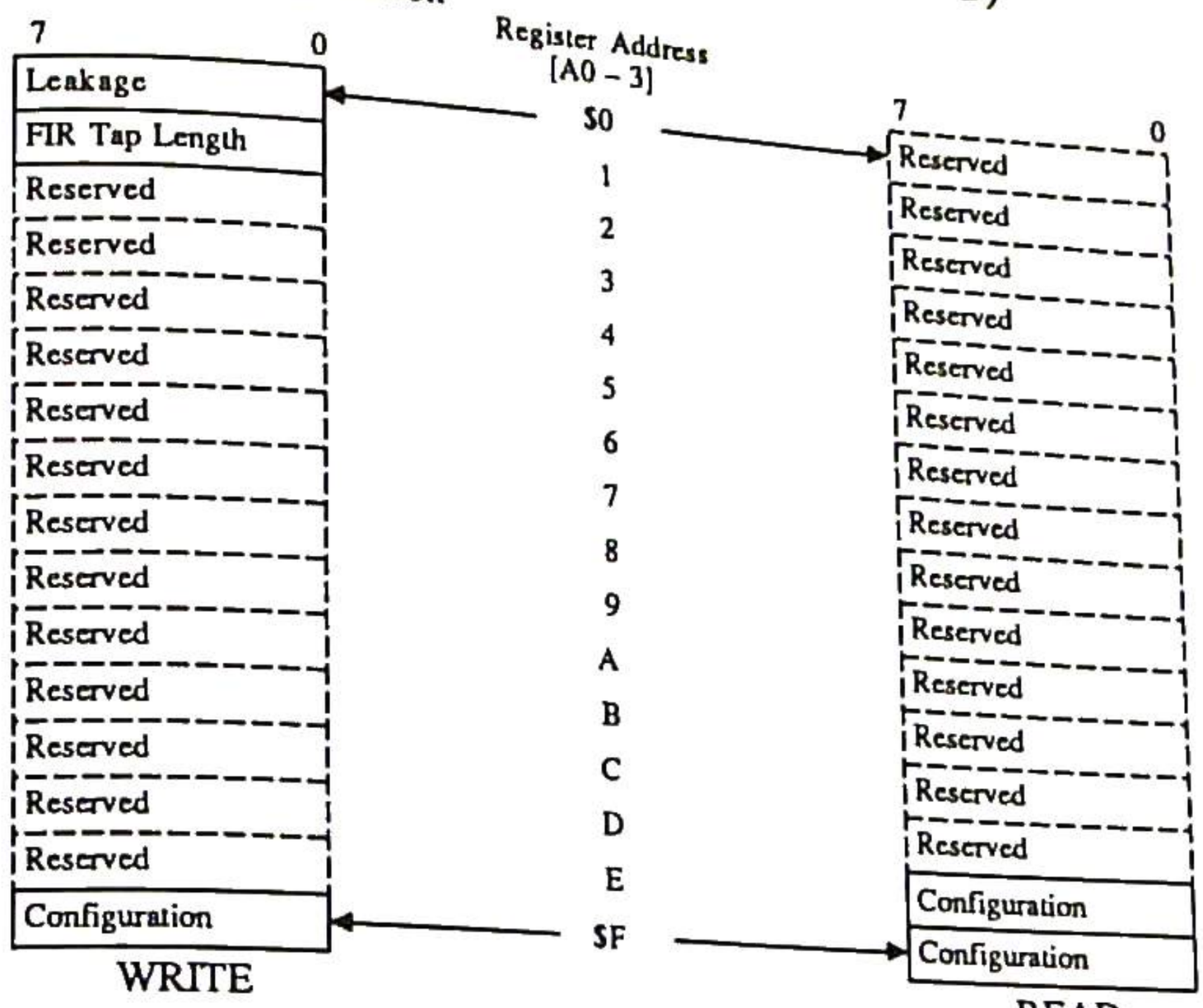


MC68000 TO DSP56200 HOST INTERFACE



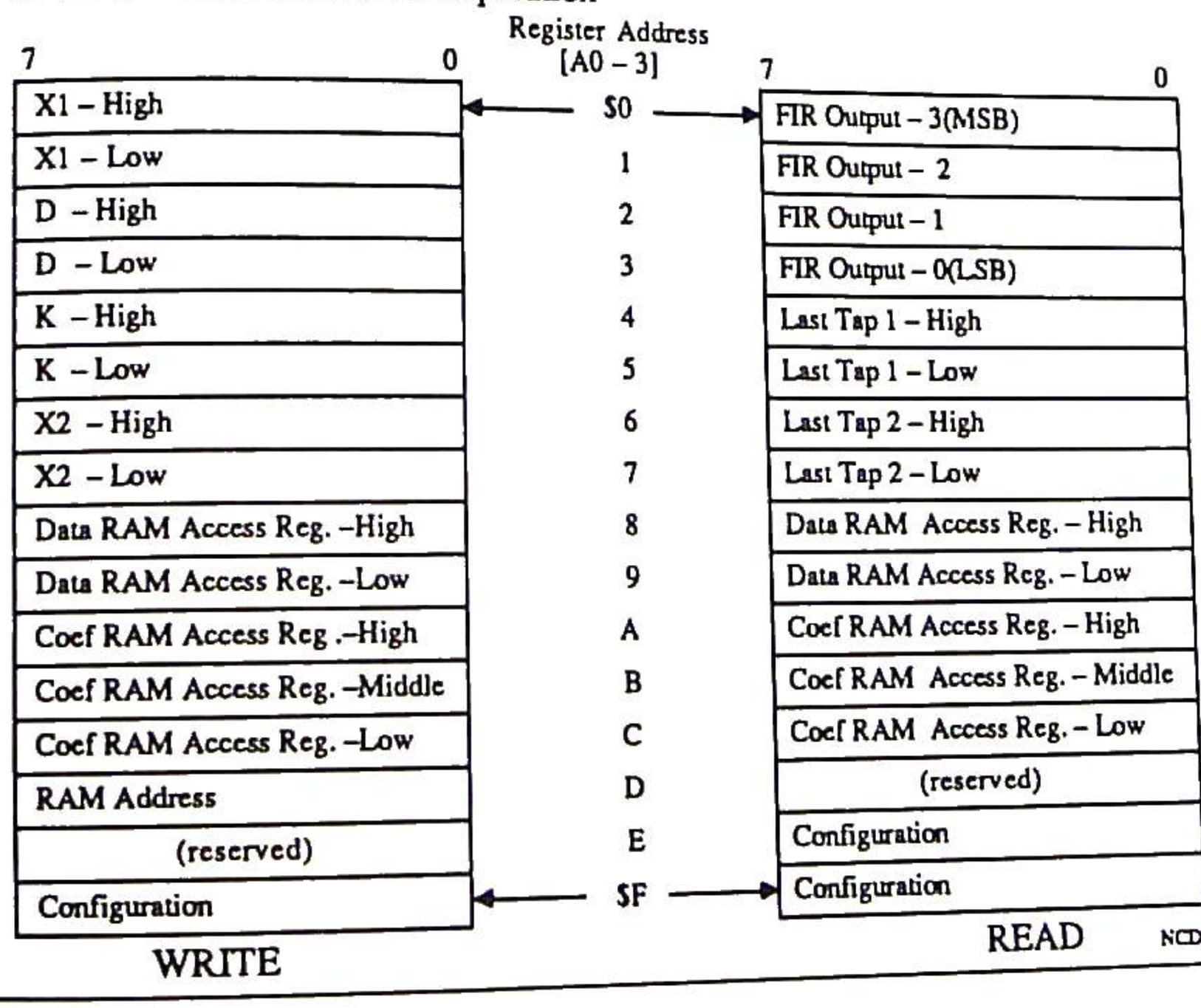
DSP56200 REGISTER MODEL (1 of 3)

- Bank 1 – Used for Initialization



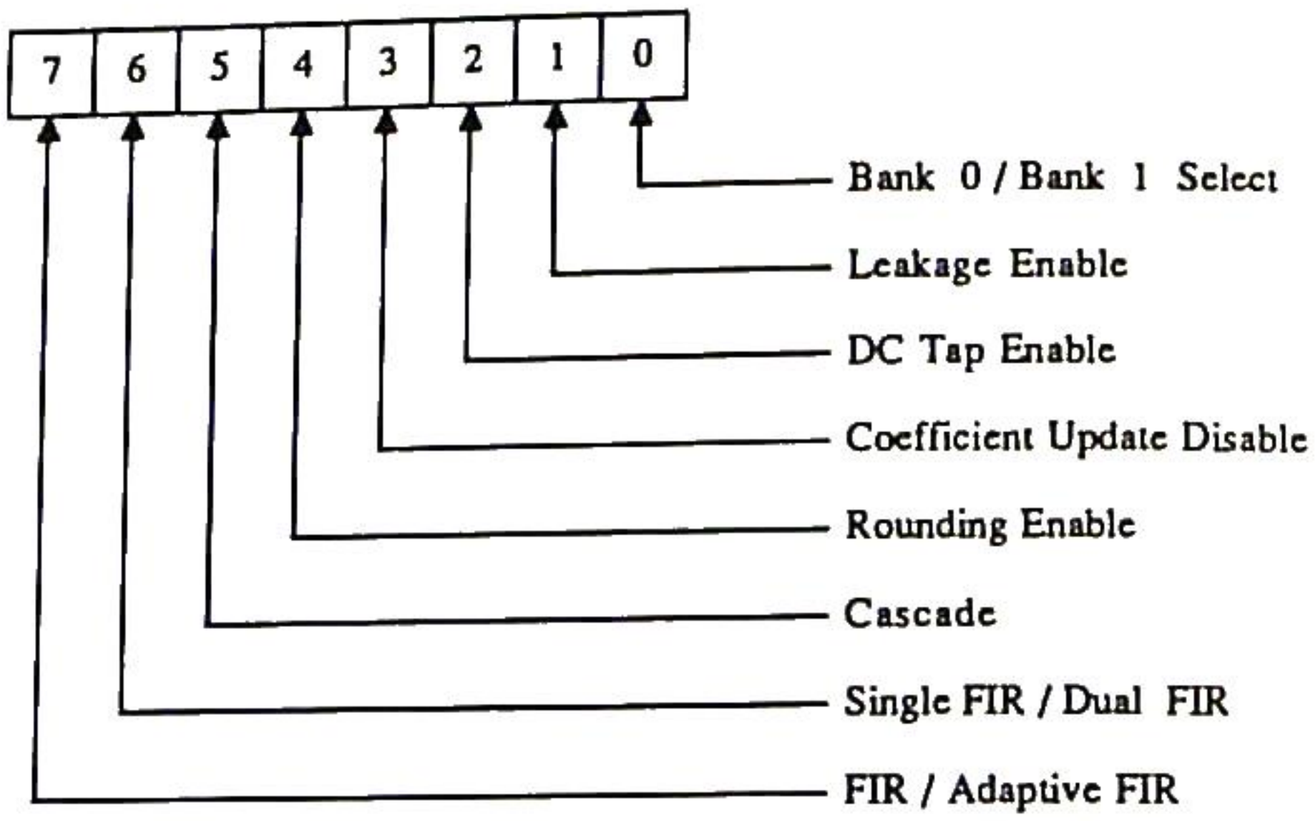
DSP56200 REGISTER MODEL (2 of 3)

- Bank 0 – Used for Normal Operation



DSP56200 REGISTER MODEL (3 of 3)

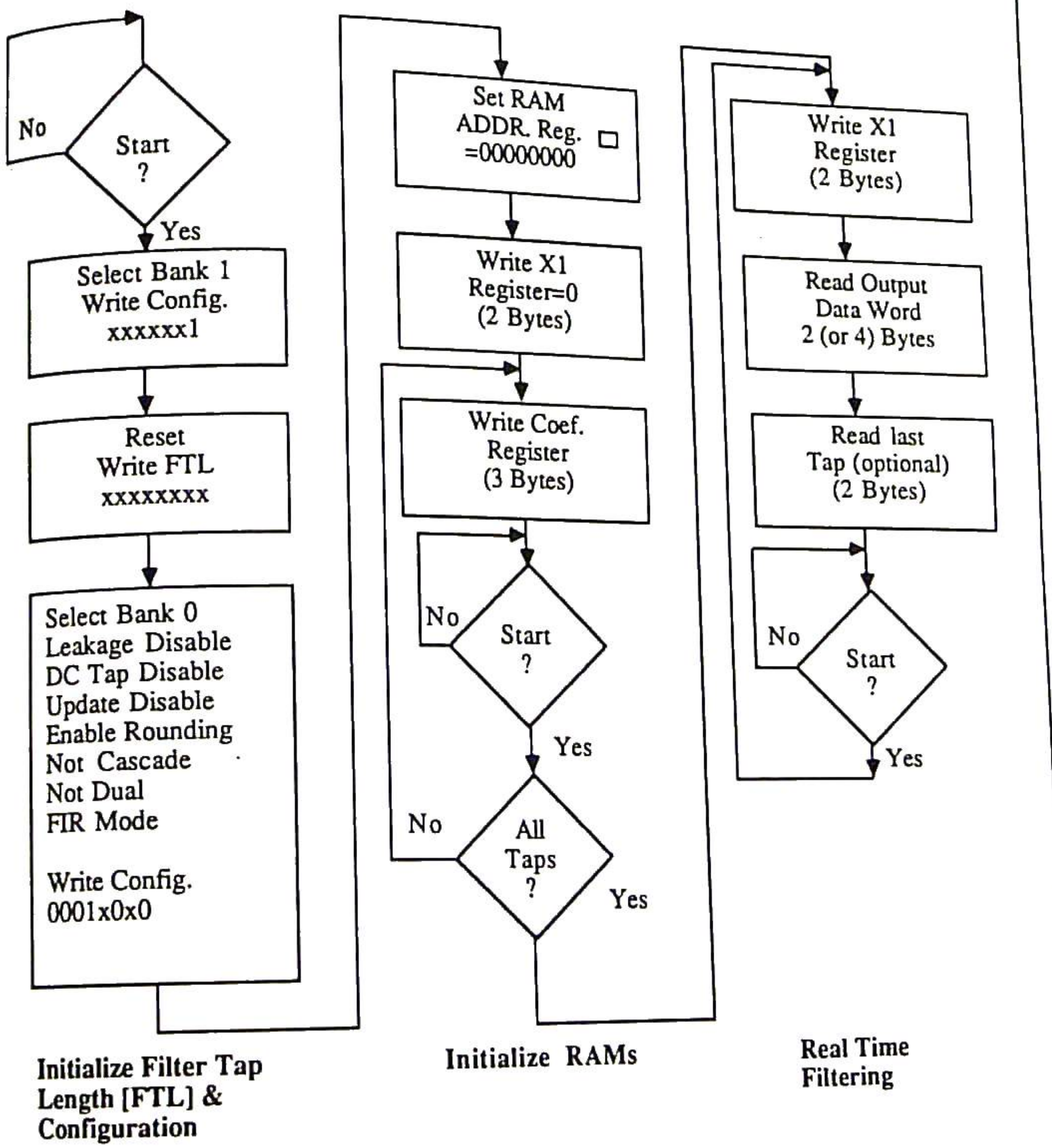
- Configuration Register – Used for System Control



NCD02-1107

DSP56200 OPERATION

- Flow Diagram - Standalone Single FIR

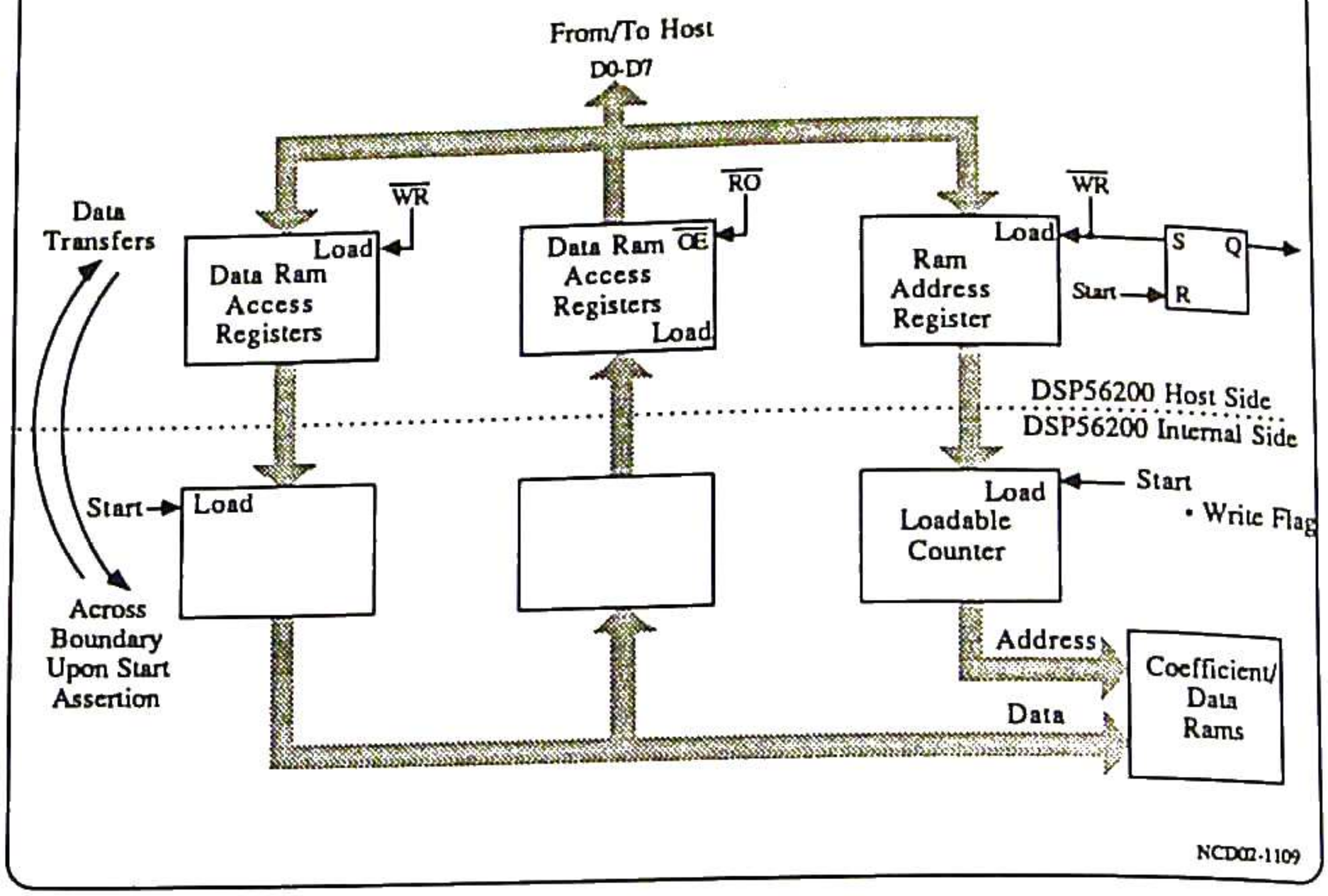


Note: The parallel interface is double buffered. A START pulse is used to transfer the data from the byte wide input buffers to the internal registers with one exception. Bit 0 of the configuration register is transferred immediately to change the bank selection.

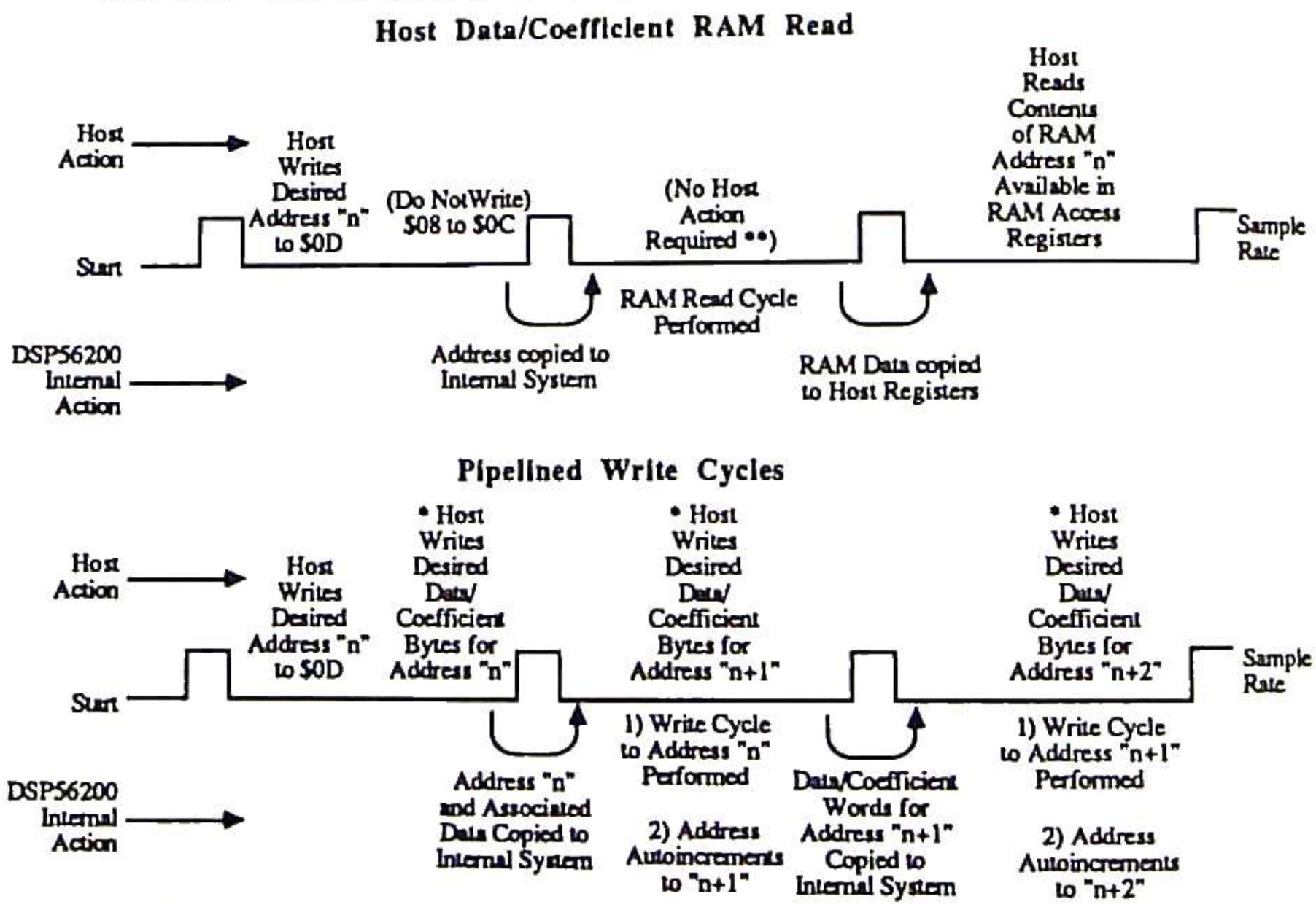
NCD02-1108

□ DSP56200 autoincrements this register.

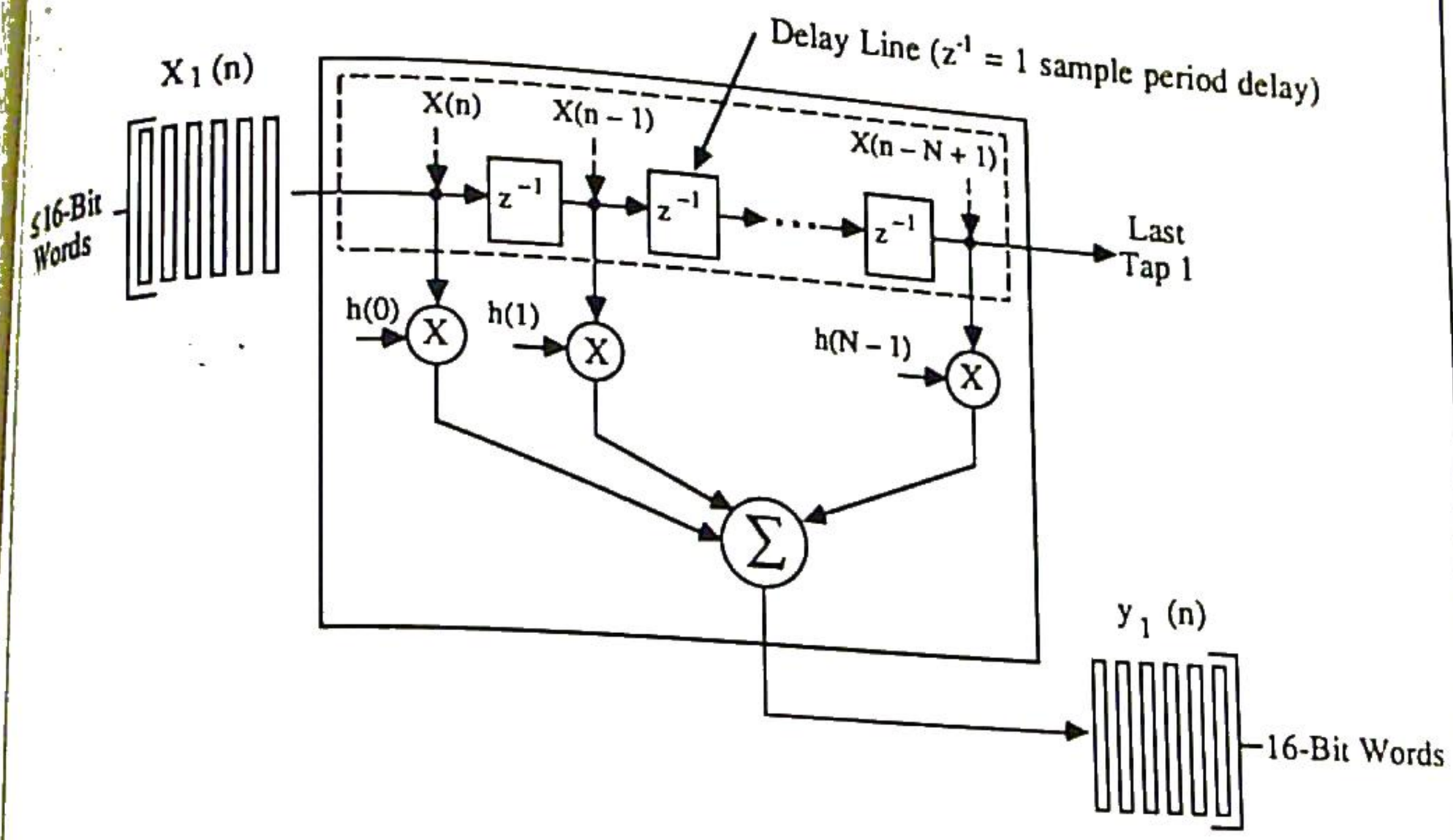
HOST INTERFACE TRANSFERS

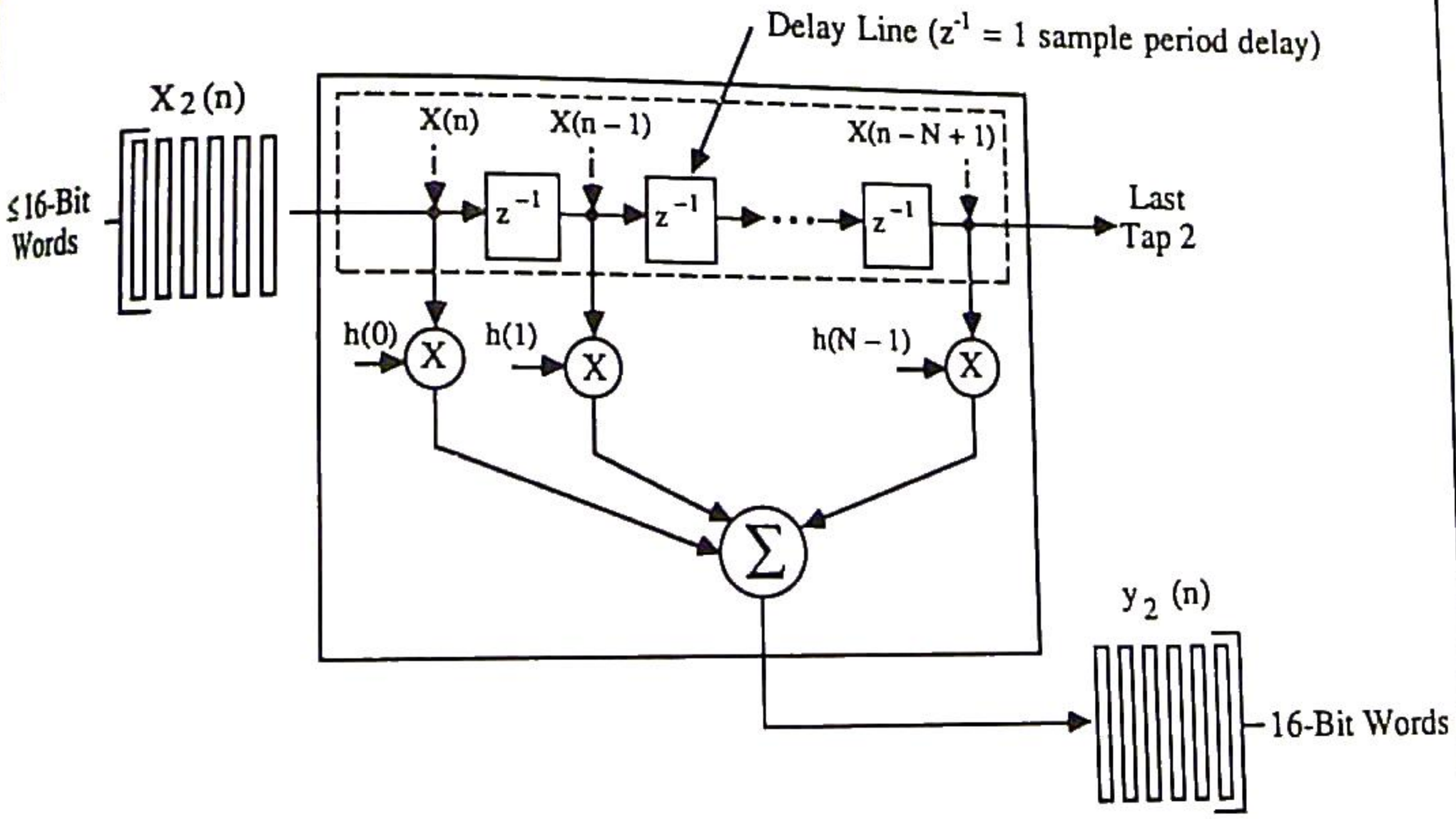


HOST INTERFACE TRANSFER EXAMPLES



DUAL FIR MODE

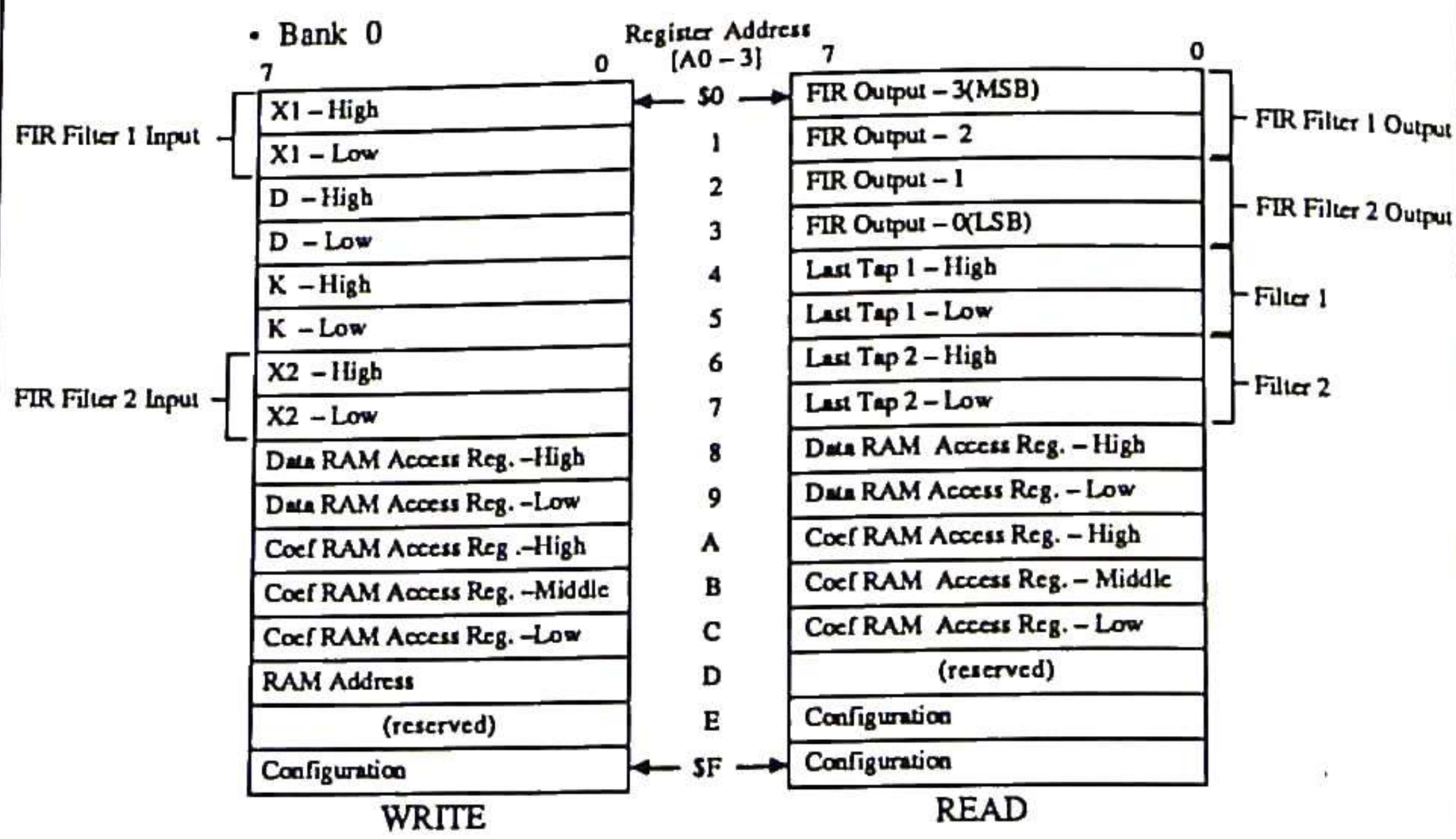




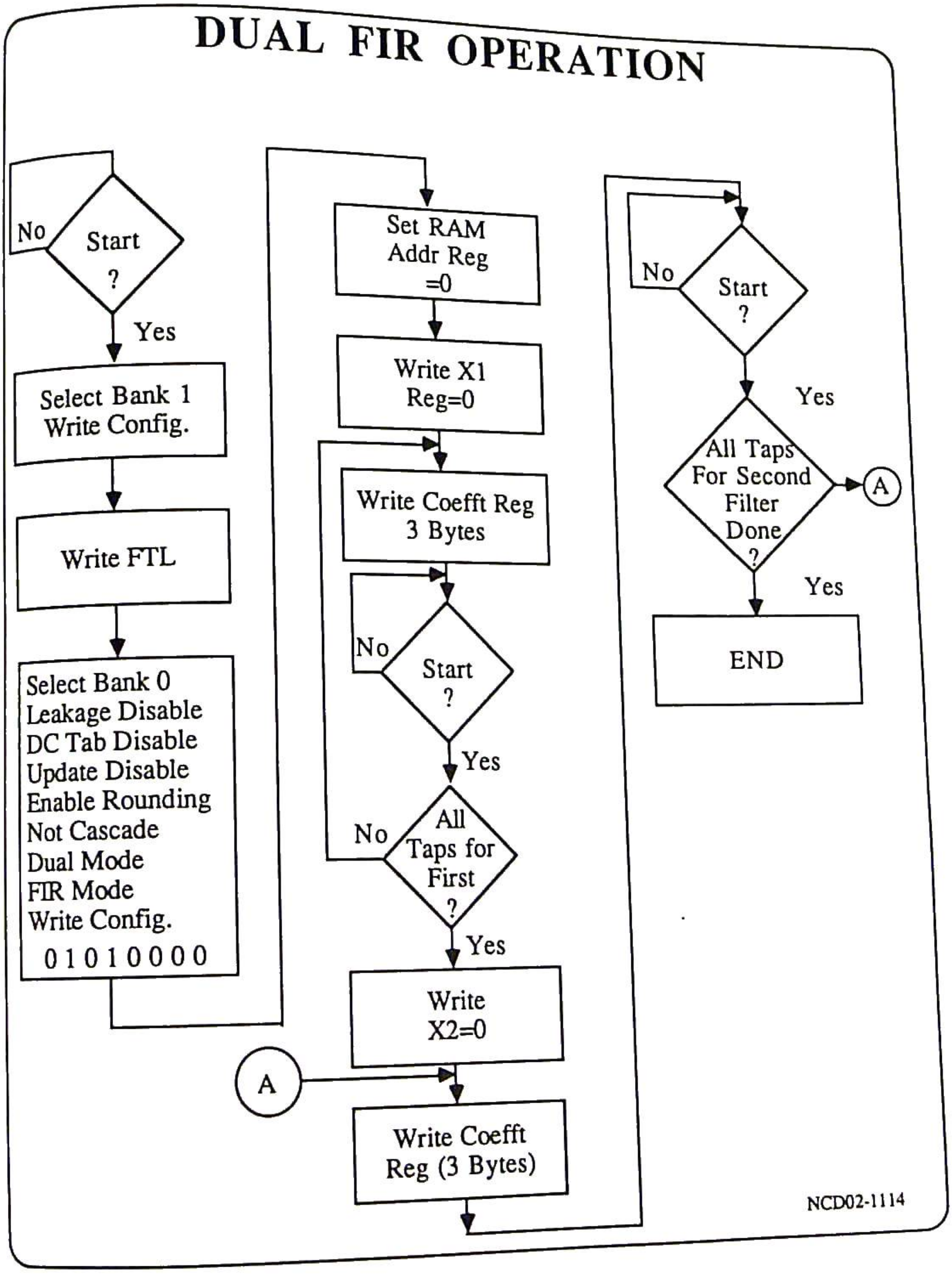
- Filter Lengths Must be the Same.

NCD02-1112

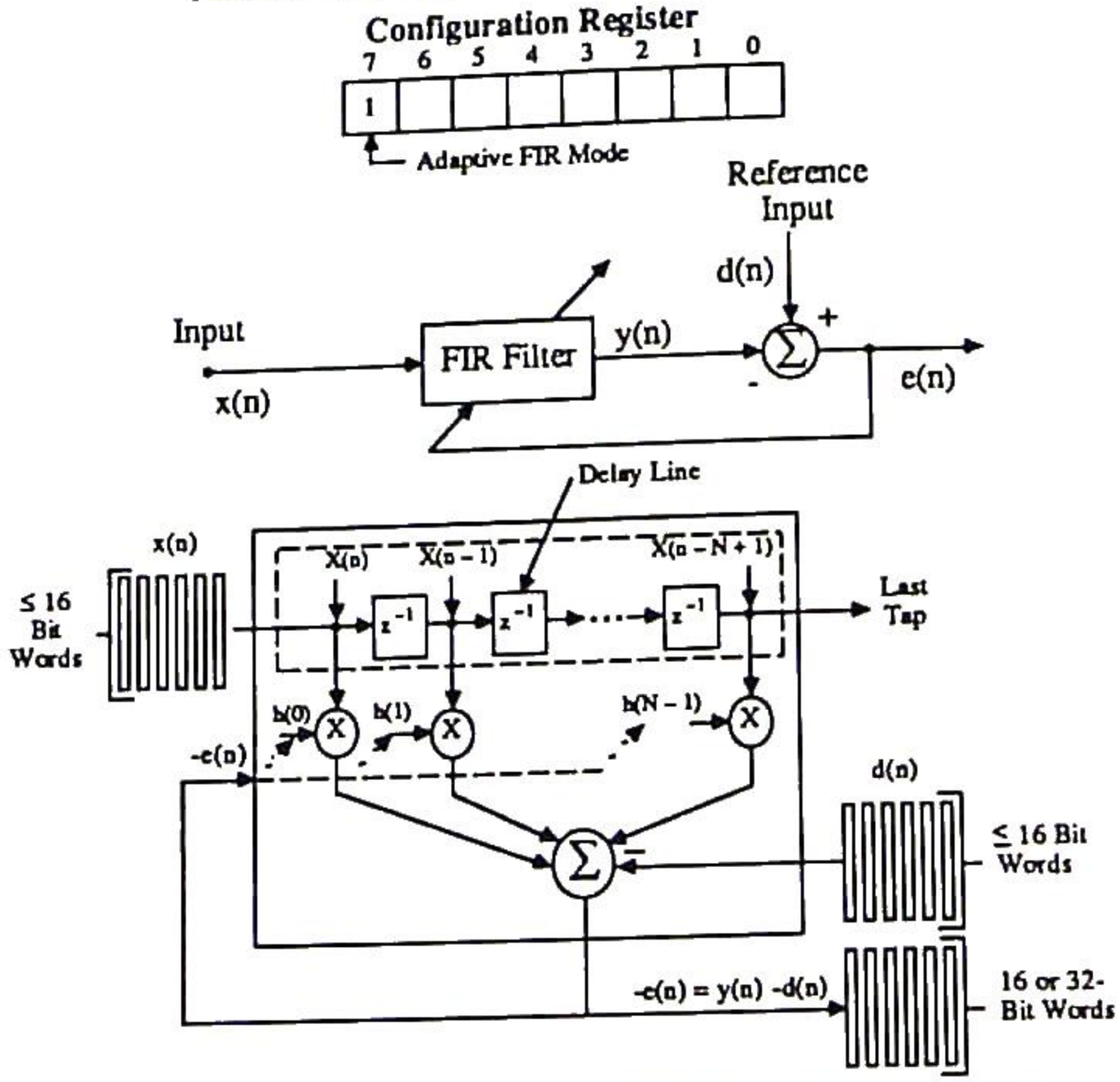
DUAL FIR REGISTER MODEL



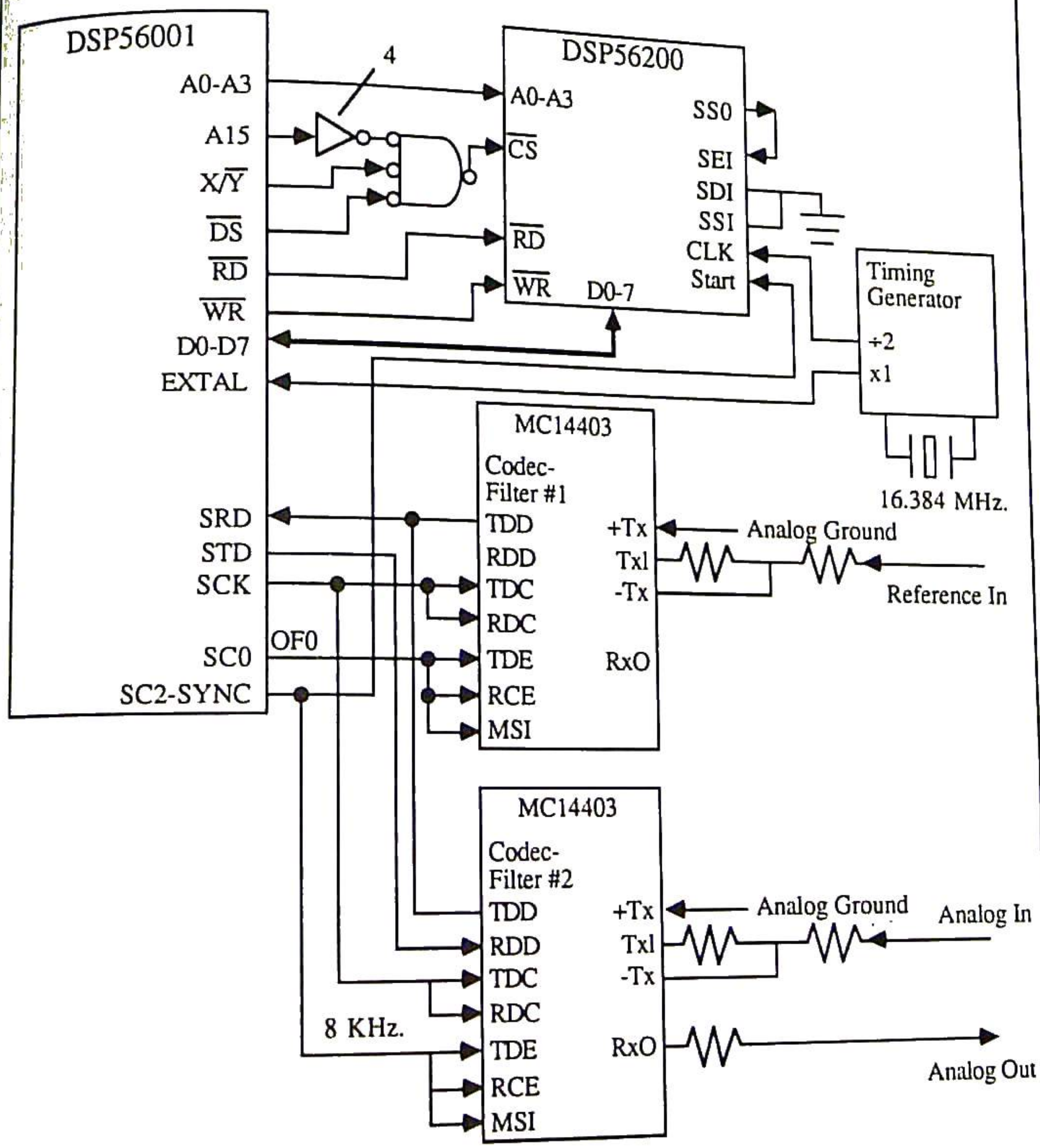
DUAL FIR OPERATION



ADAPTIVE FIR FILTER MODE

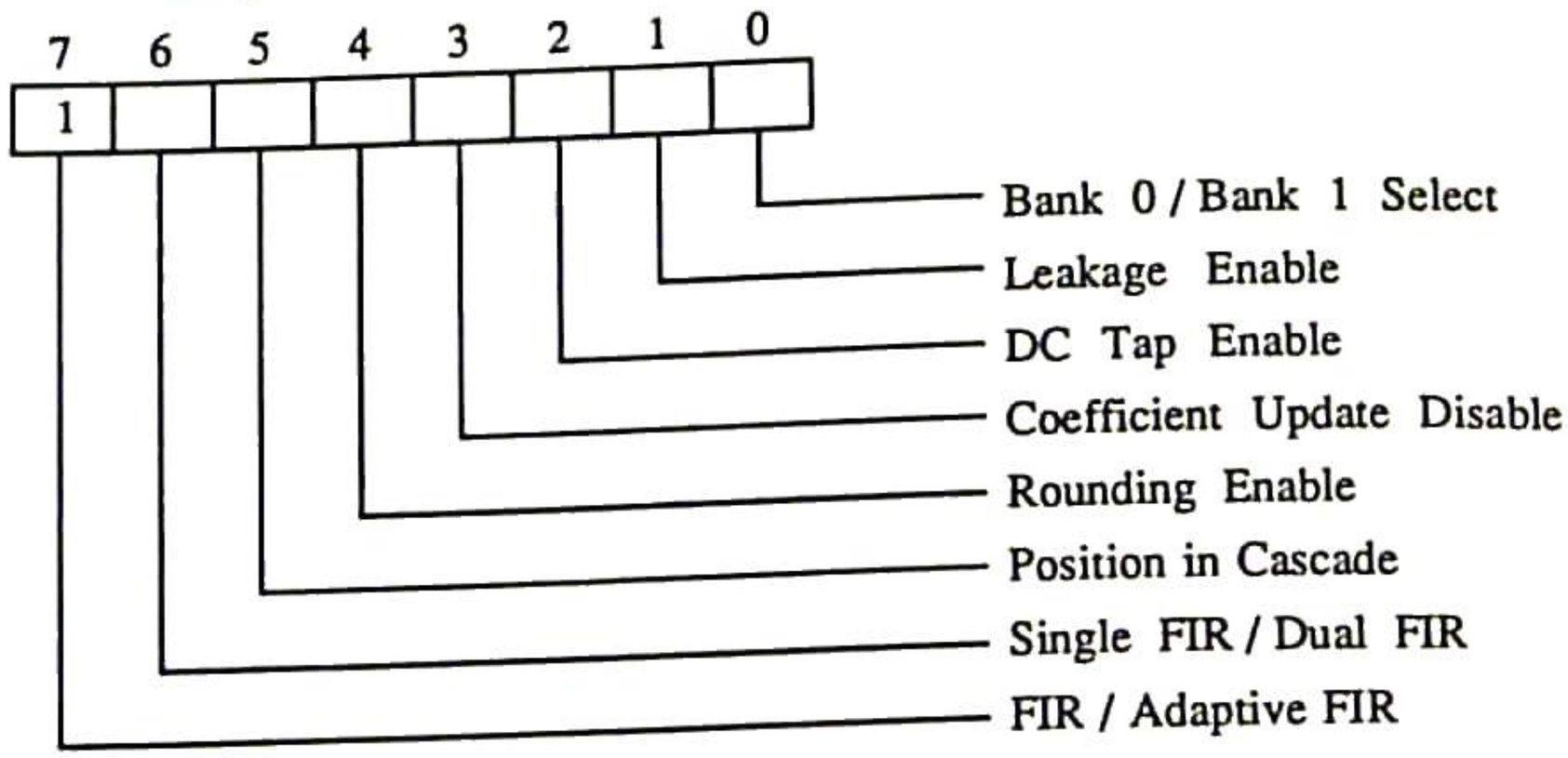


ADAPTIVE FIR SYSTEM EXAMPLE

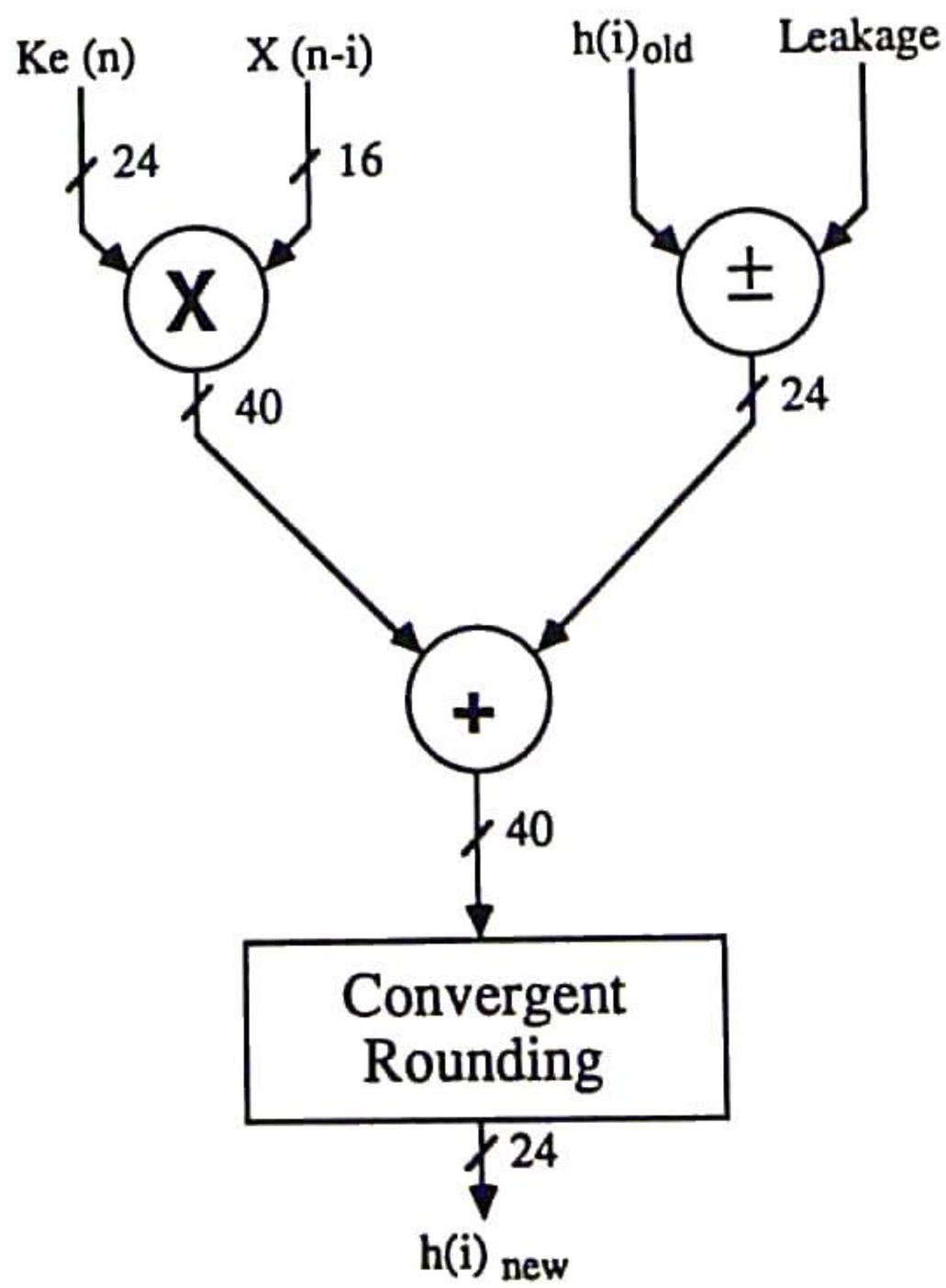


COEFFICIENT UPDATE

Configuration Register



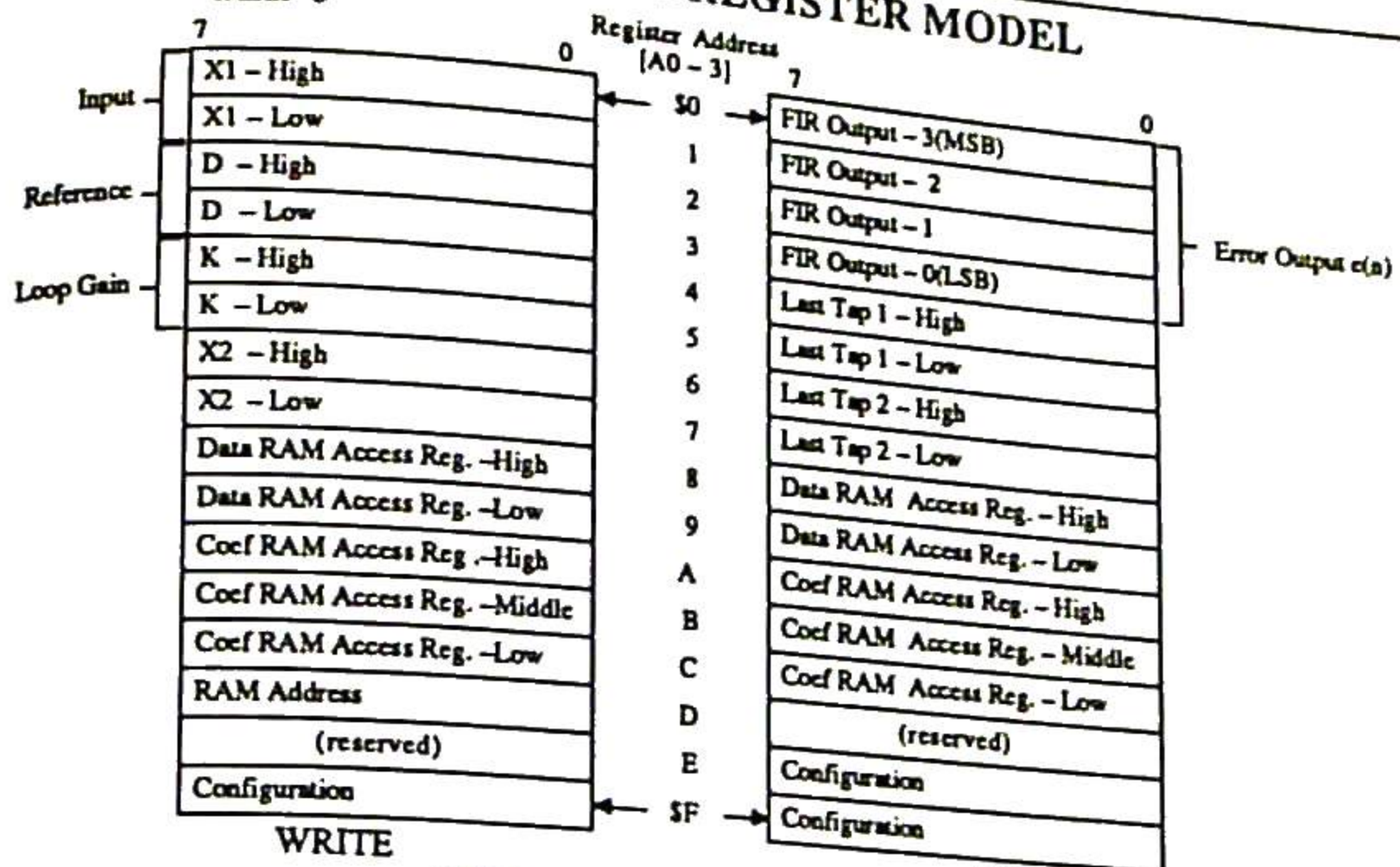
$$h(i)_{new} = h(i)_{old} + Ke \times (n-i) \pm Leakage \qquad e(n) = d(n) - \sum_{i=0}^{N-1} h(i) \times (n-i)$$



NCD02-1117

ADAPTIVE FILTER REGISTER MODEL

• Bank 0

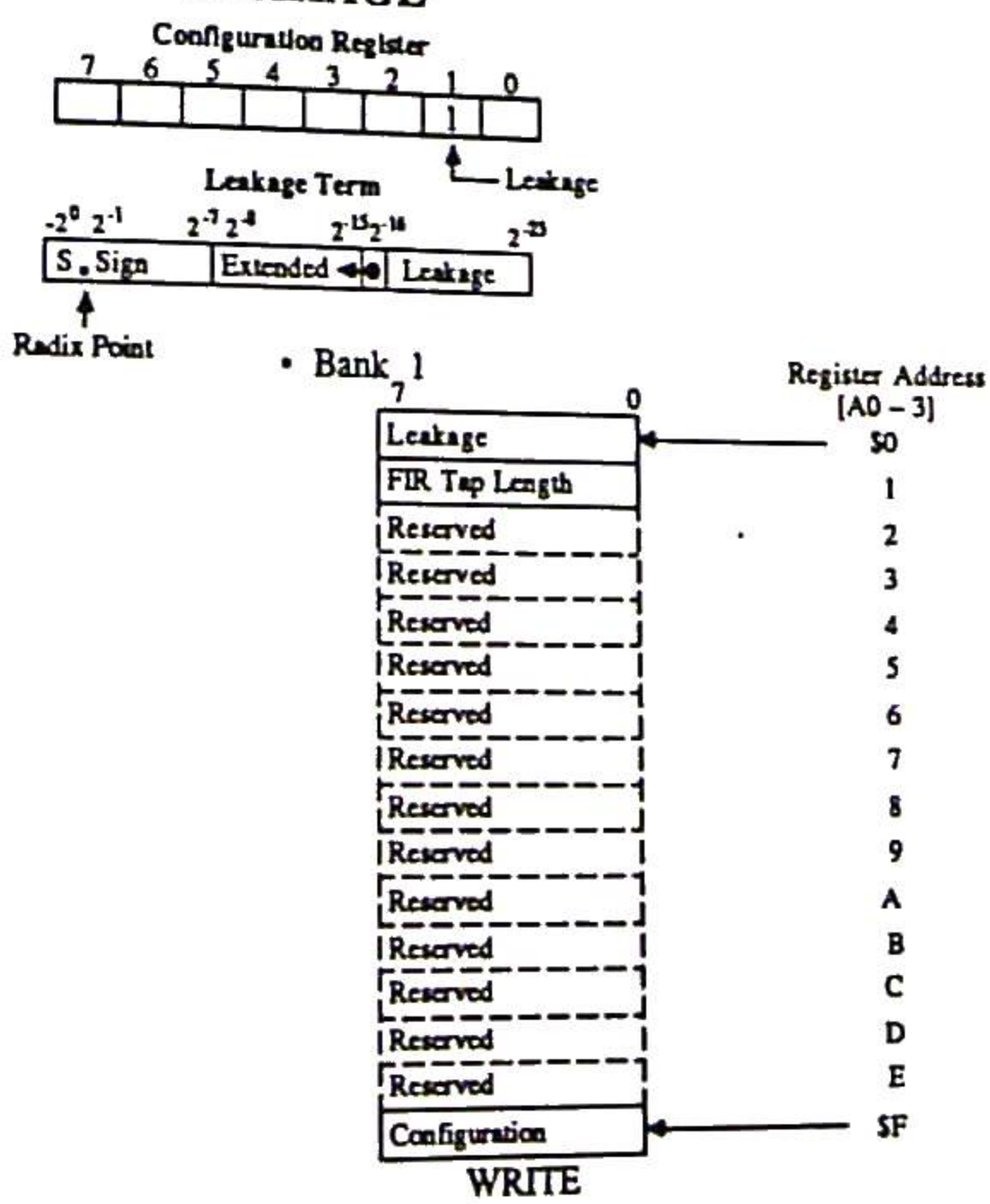


FIR Tap Length Used	* Maximum Recommended K
32 (20 Hex)	6000 Hex
64 (40 Hex)	3000 Hex
128 (80 Hex)	1000 Hex
192 (C0 Hex)	A00 Hex
256 (FF Hex)	900 Hex

* M. M. Sondhi, D. A. Berkely, "Silencing Echoes on the Telephone Network", Proc. IEEE, Vol. 68, No. 8, pp. 948-963, Aug. 1980.

NCD02-1118

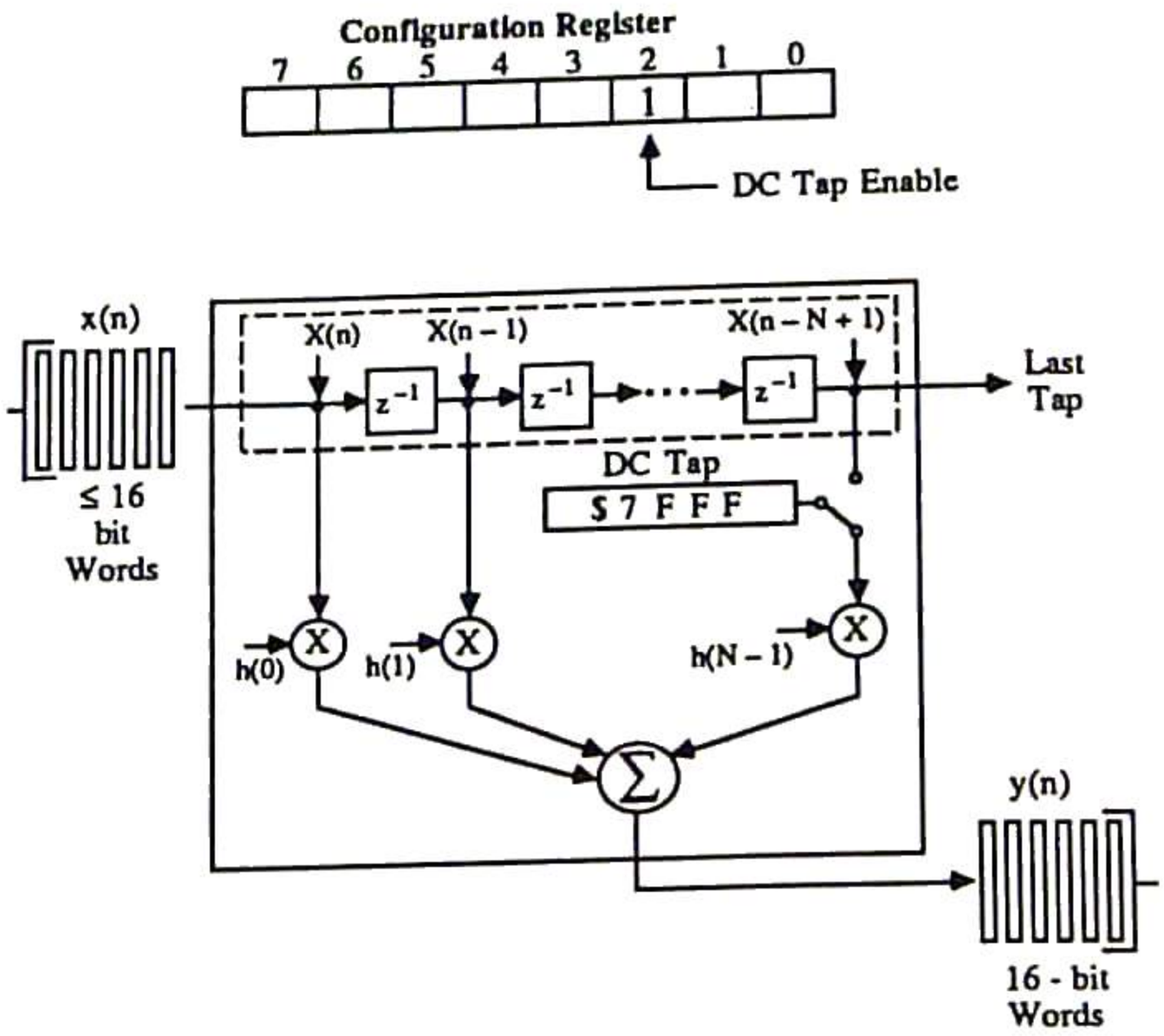
LEAKAGE



• Used to slowly push a coefficient toward zero.

NCD02-1119

DC TAP

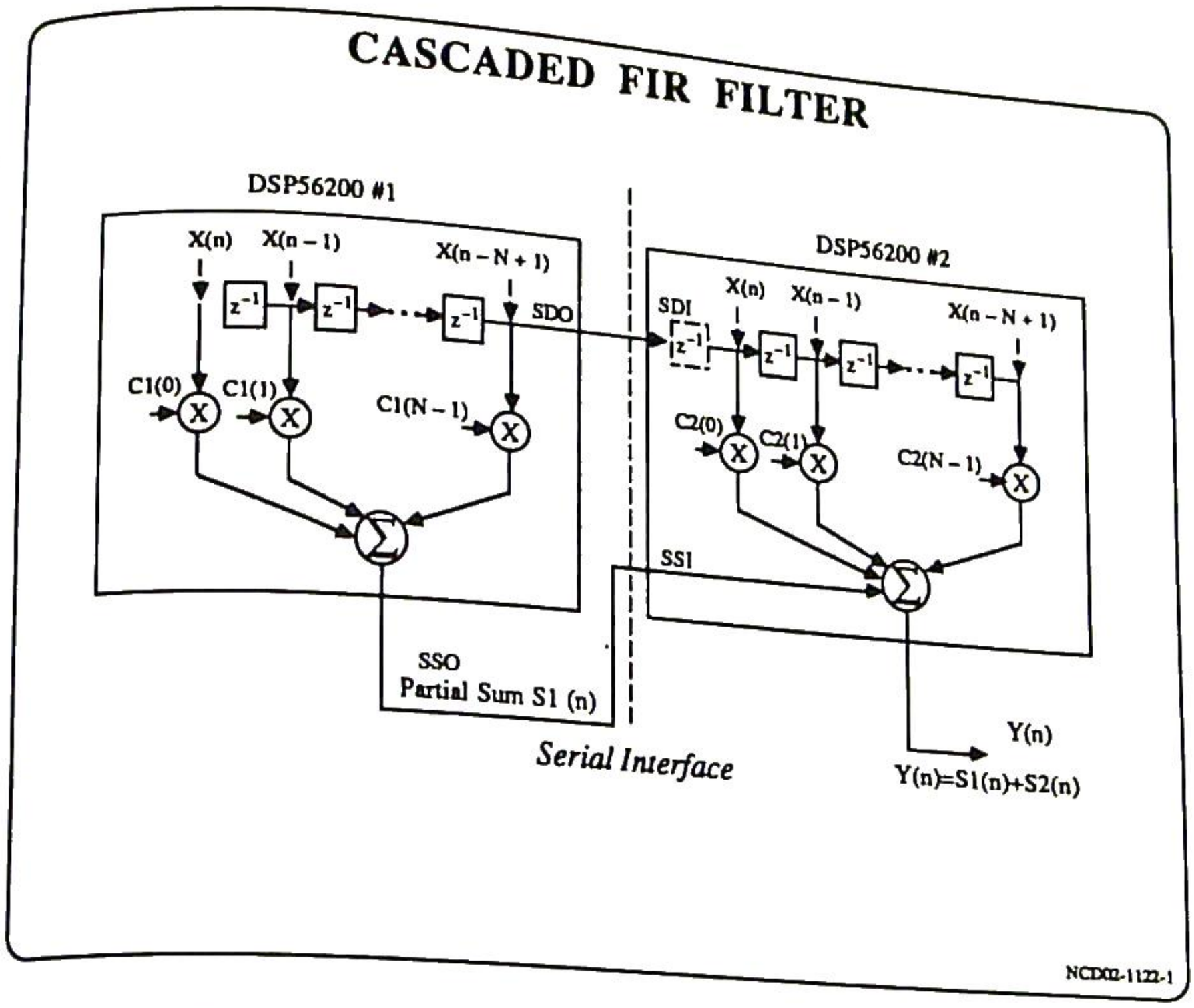


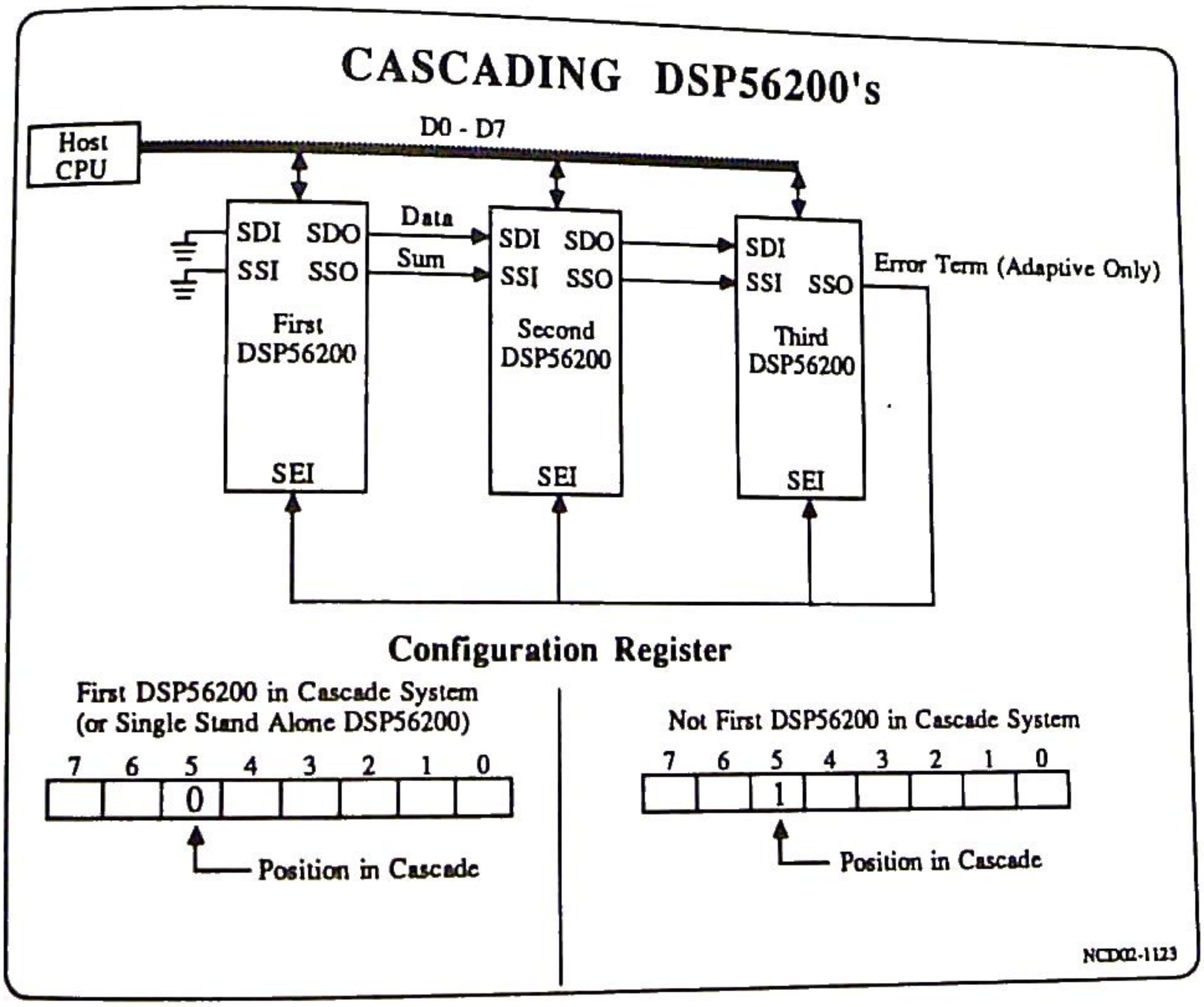
NCD02-1120

CASCADING

- Not Permitted in Dual FIR Mode
- Advantages
  1. Allows longer-tap filters
  2. Provides 3-way tradeoff:
     - # Taps/chip
     - # Chips used
     - Maximum sampling rate

NCD02-1121

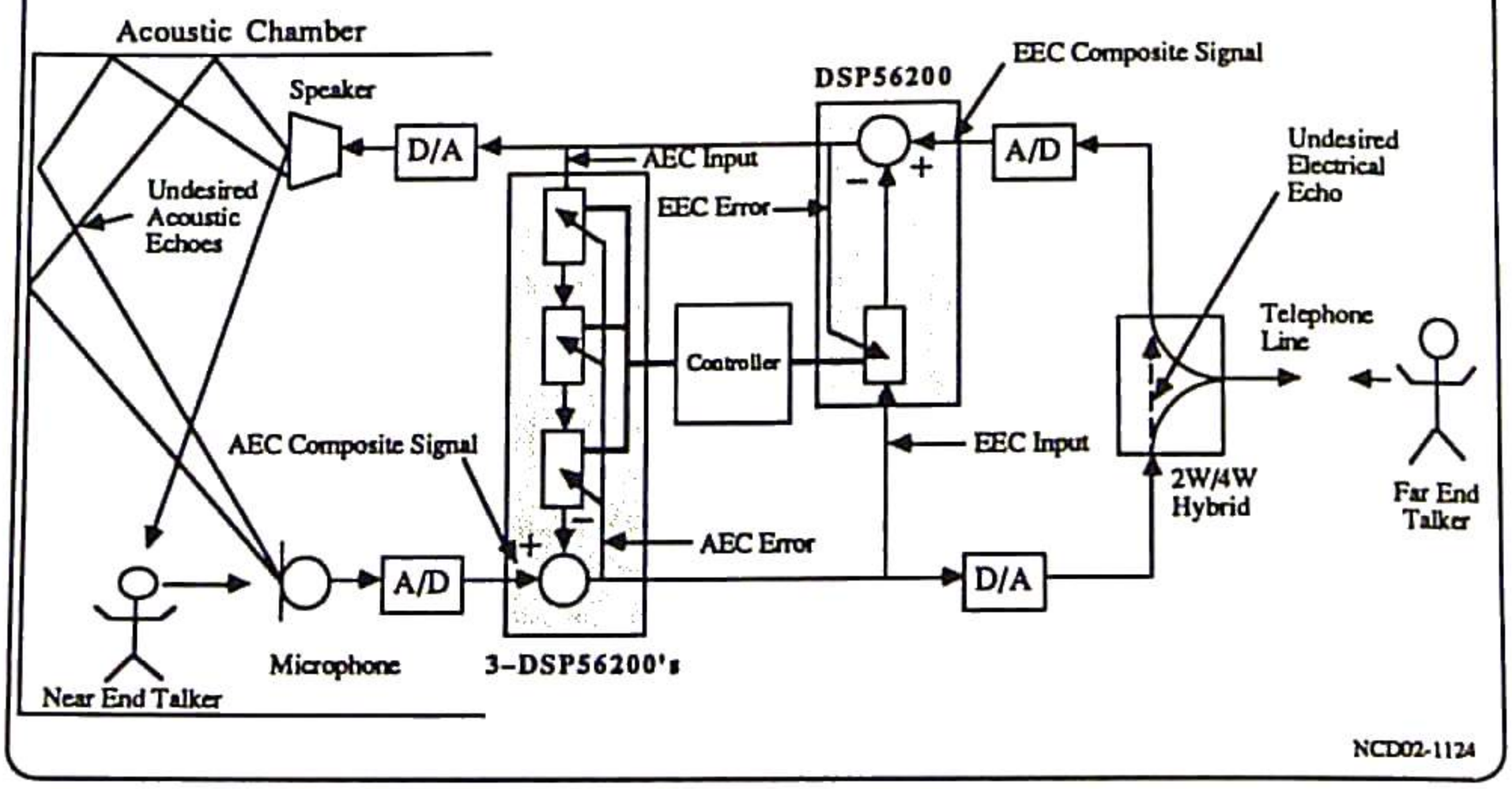

CASCADED FIR FILTER
DSP56200 #1
X(n)
X(n – 1)
X(n – N + 1)
SDO
C1(0)
C1(1)
C1(N – 1)
SSO
Partial Sum S1 (n)
DSP56200 #2
SDI
X(n)
X(n – 1)
X(n – N + 1)
C2(0)
C2(1)
C2(N – 1)
SSI
Serial Interface
Y(n)
Y(n)=S1(n)+S2(n)
NCD02-1122-1

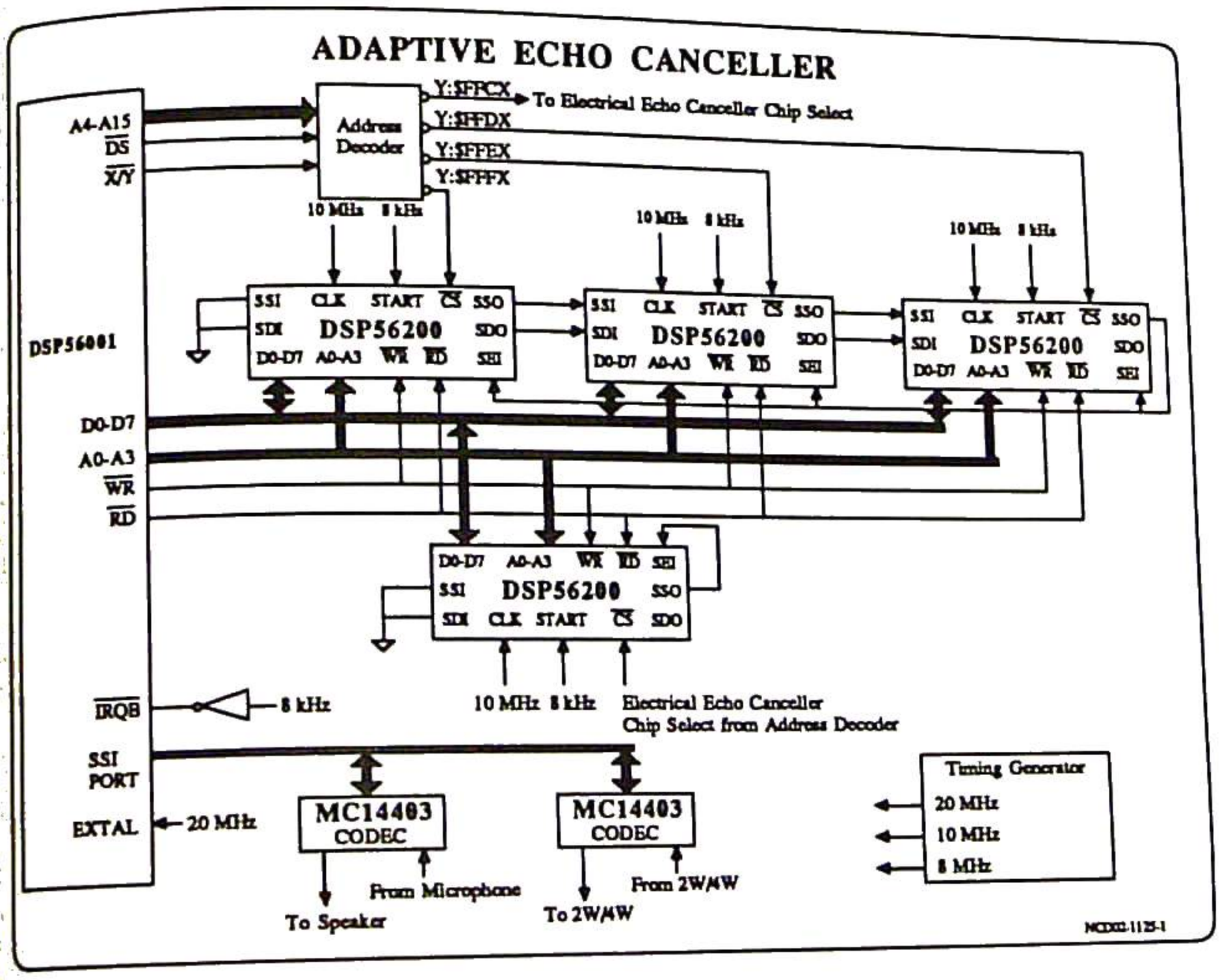

CASCADING DSP56200's
Host CPU
D0 - D7
SDI SDO
SSI SSO
First DSP56200
SEI
Data
Sum
Second DSP56200
Third DSP56200
Error Term (Adaptive Only)
Configuration Register
First DSP56200 in Cascade System
(or Single Stand Alone DSP56200)
7 6 5 4 3 2 1 0
0
Position in Cascade
Not First DSP56200 in Cascade System
1
NCD02-1123

SPEAKER PHONE

Echo Cancelling

- Adaptive FIR Digital Filter




ADAPTIVE ECHO CANCELLER
A4-A15
DS
X/Y
Address Decoder
Y:$FFCX
To Electrical Echo Canceller Chip Select
Y:$FFDX
Y:$FFEX
Y:$FFFX
10 MHz 8 kHz
DSP56001
SSI CLK START CS SSO
SDI DSP56200 SDO
D0-D7 A0-A3 WR RD SEI
D0-D7
A0-A3
WR
RD
IRQB
8 kHz
10 MHz 8 kHz
Electrical Echo Canceller
Chip Select from Address Decoder
SSI
PORT
EXTAL
20 MHz
MC14403
CODEC
From Microphone
To Speaker
From 2W/4W
To 2W/4W
Timing Generator
20 MHz
10 MHz
8 MHz

DSP56200 PERFORMANCE

* Maximum Sampling Frequency [KHz]

MODE	TOTAL NUMBER of TAPS					
	32	64	128	256	512	1024
FIR Filter * Standalone * Four(4) in Cascade	227	132	71 222	37 132	71	37
Adaptive FIR * Standalone * Four (4) in Cascade	123	69	37 120	19 69	37	19
Dual FIR Filter * Standalone	122	68	36			

NCD02-1126-1

FILTER DESIGN AND ANALYSIS SYSTEM
VERSION 1.3

CUSTOMER SERVICE

FOR TELEPHONE SUPPORT ON THIS PRODUCT, PLEASE CALL
(714) 548-3257 (0800-1700 PACIFIC STANDARD TIME)

Momentum Data Systems
1666 Newport Boulevard #115
Costa Mesa, CA 92627

© COPYRIGHT MOMENTUM DATA SYSTEMS

1988 - ALL RIGHTS RESERVED

NCD02-1200

Table of Contents

Getting Started
Select Filter Algorithm 12-5
Select Filter Type 12-5
Select Filter Characteristics 12-6
Select Filter Window and Size 12-6
Select Desired Output and Format 12-7
Coefficient Quantization 12-7
Output Files 12-8
Filter Computation 12-8
Magnitude Versus Frequency 12-9
Log Magnitude Versus Frequency 12-10
Impulse Response Versus Time 12-11
COEFF.FLT 12-12
Code Generator 12-13
Code Generation Sequence 12-14
COEFF.ASM Listing (1 of 3) 12-15
COEFF.ASM Listing (2 of 3) 12-16
COEFF.ASM Listing (3 of 3) 12-17
Floating Point Magnitude Versus Frequency 12-18
Floating Point Impulse Response 12-19
24 Bit Magnituce Versus Frequency 12-20
24 Bit Impulse Response 12-21
16 Bit Magnitude Versus Frequency 12-22
16 Bit Impulse Response 12-23
..... 12-24

GETTING STARTED

```
= CD      \FDAS
= FDAS
```

Startup Screen

FILTER DESIGN AND ANALYSIS SYSTEM

VERSION 1.3

MOMENTUM DATA SYSTEMS

HIT F10 KEY TO CONTINUE

Copyright 1987

NCD02-1201

SELECT FILTER ALGORITHM

Filter Design and Analysis Sytem
Main Menu

```
IIR Filter Design . . . . . . . . . . . . . . . . . . . . . . . 1
FIR Filter Design with Windows . . . . . . . . . . . . . . . . . 2
Equiripple Filter Design (Parks-McClellan). . . . . . . . . . . 3
Analysis of Digital System . . . . . . . . . . . . . . . . . . . 4
Read Filter Specification File . . . . . . . . . . . . . . . . . 5

Exit to DOS                                                     X


Enter Action Desired        2
```

AFTER ENTERING ACTION DESIRED, HIT F10 KEY TO START THE SYSTEM

NCD02-1202

SELECT FILTER TYPE

Finite Impulse Response Filter Design Menu
(With Windows)

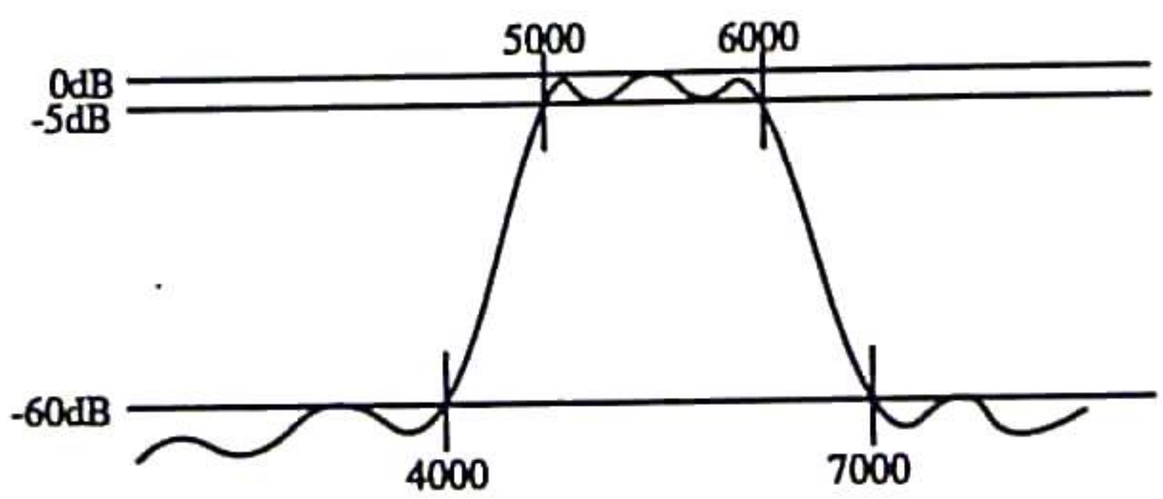
Filter Type 3
- 1 - Lowpass
- 2 - Highpass
- 3 - Bandpass
- 4 - Bandstop
- 5 - Differentiator

Frequency Mode H
- H - Frequency is in Hertz
- R - Frequency is in Radians/sec

↵ :accept field
F1/F10:accept screen Esc:main menu ↑:previous field ↓:next field

NCD02-1203

SELECT FILTER CHARACTERISTICS



FIR FILTER DESIGN

BANDPASS FILTER

Sampling Frequency	21333	
Passband Frequencies	5000	6000
Stopband Frequencies	4000	7000
Passband Ripple (dB) (attenuation)	5	
Stopband Ripple (dB) (attenuation	60	

↵ :accept field
F1/F10:accept screen Esc:previous screen ↑:previous field ↓:next field

NCD02-1204

SELECT FILTER WINDOW AND SIZE

```
Estimated Number of Taps for Rectangular Window is . . . . . 19
Estimated Number of Taps for Triangular Window is . . . . . 89
Estimated Number of Taps for Hanning Window is . . . . . 69
Estimated Number of Taps for Hamming Windiow is . . . . . 73
Estimated Number of Taps for Blackman Window is . . . . . 121
Estimated Number of Taps for Kaiser Window is . . . . . 79

        Select Desired window Function . . . . . 6
           1 - Rectangular
           2 - Triangular
           3 - Hanning
           4 - Hamming
           5 - Blackman
           6 - Kaiser

        Enter Number of Taps Desired . . . . . .79

↵ :accept field
F1/F10:accept screen   Esc:previous screen   ↑:previous field   ↓:next field
```

NCD02-1205

SELECT DESIRED OUTPUT AND FORMAT

```
                    Select Desired Output (Plots and Reports)

                              CRT PRINTER
Magnitude . . . . . . . . X                    options are selected
Log10 Magnitude . . . . X                      by entering an X
Impulse Response  . . . X

Save Filter Specifications . . . . . . . . . . . . . . X
Quantize Filter Coefficients . . . . . . . . . . . . . X
Create Filter Coefficient File . . . . . . . . . . . . X
Create Graphics Interchange File . . . . . . . . . . .
Detailed Magnitude and Log10 Magnitude Reports . . . X
        (appended to file FDAS.OUT)
Grid Lines on Plots  . . . . . . . . . . . . . . . . X


Frequency Range:      from        3000   to        8000
Magnitude Range:      from  0.000        to   1.000
Log Magnitude Range:  from -100.00       to     0.00

↵ :accept field
F1/F10:accept screen      ↑:previous field      ↓:next field
```

NCD02-1206

COEFFICIENT QUANTIZATION

Select Quantization Options

Number of Significant Bits (8 to 32) 24

Quantization Type 2
- 1 - Floating Point Quantization (All Coefficients have Same Number of Significant Bits)
- 2 - Fixed Point Fractional Quantization

Realization Type 1
- 1 - Second Order Sections (Transpose of Direct Form II - also known as Direct Form I)
- 2 - Second Order Sections (Direct Form II)
- 3 - Floating Point Implementation

IIR FILTER ONLY

↵ :accept field
F1/F10:accept screen ↑:previous field ↓:next field

NCD02-1207

OUTPUT FILES

Enter File Names for Coefficient File and Graphics Interchange Files

Filter Specification File . . . . . SPEC.SPC

Filter Coefficient File . . . . . COEFF.FLT

Graphics Interchange Files

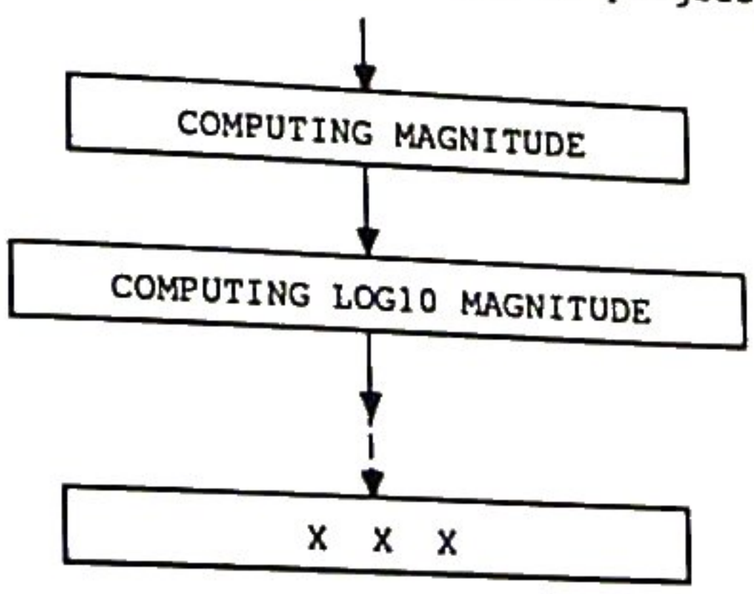
Magnitude . . . . . MAGN.GMF
Log Magnitude . . . . . LMAG.GMF
Phase . . . . . PHAS.GMF
Poles and Zeros . . . . . PANZ.GMF
Impulse Response . . . . . IMPL.GMF
Group Delay . . . . . GDEL.GMF

↵ :accept field
F1/F10:accept screen ↑:previous field ↓:next field

NCD02-1208

FILTER COMPUTATION

Please wait - calculations in progress

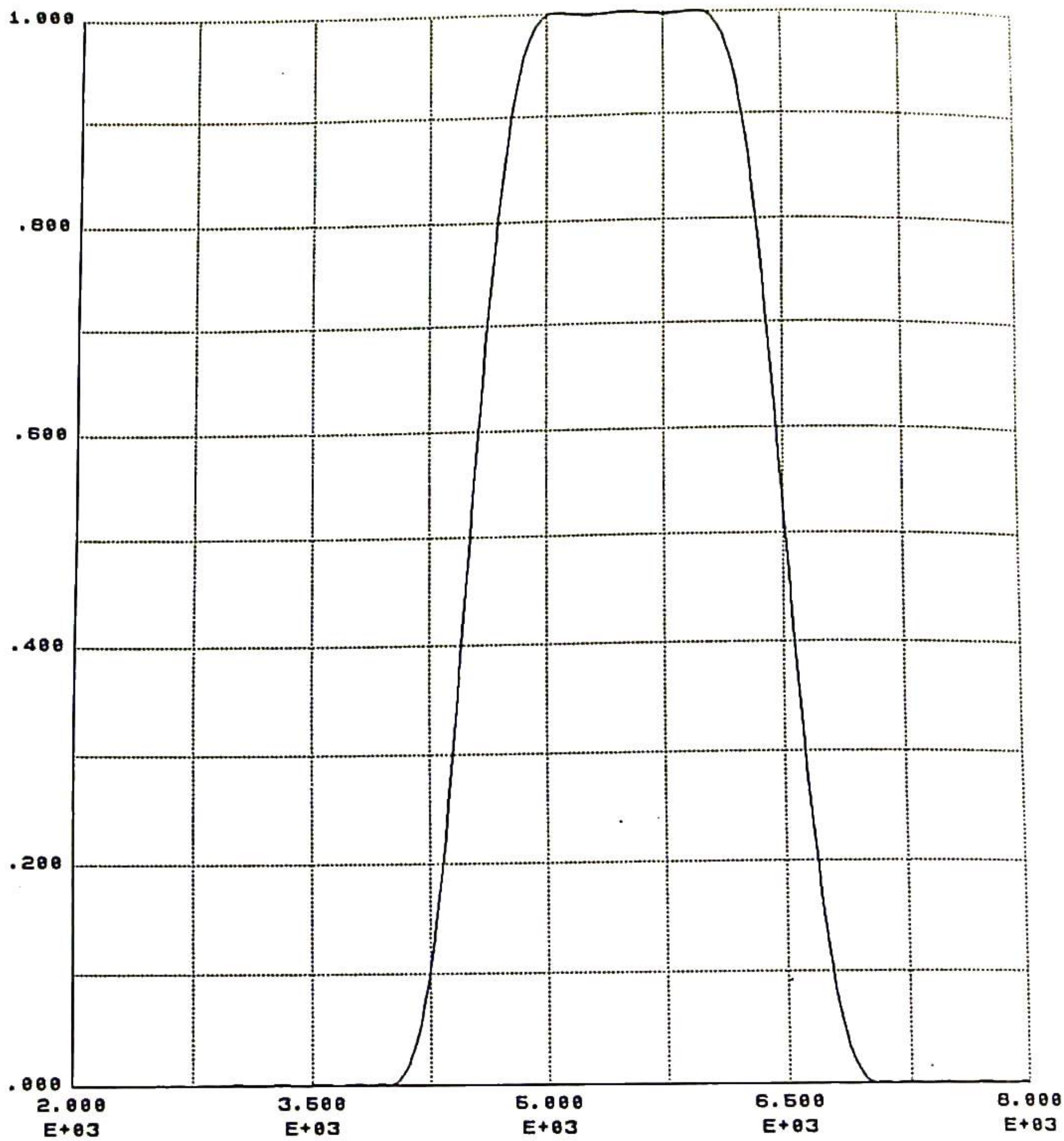


Calculation time is seconds to tens of minutes
Time to calculate filter depends on:

- Filter Complexity
- Math chip

NCD02-1209

MAGNITUDE VERSUS FREQUENCY

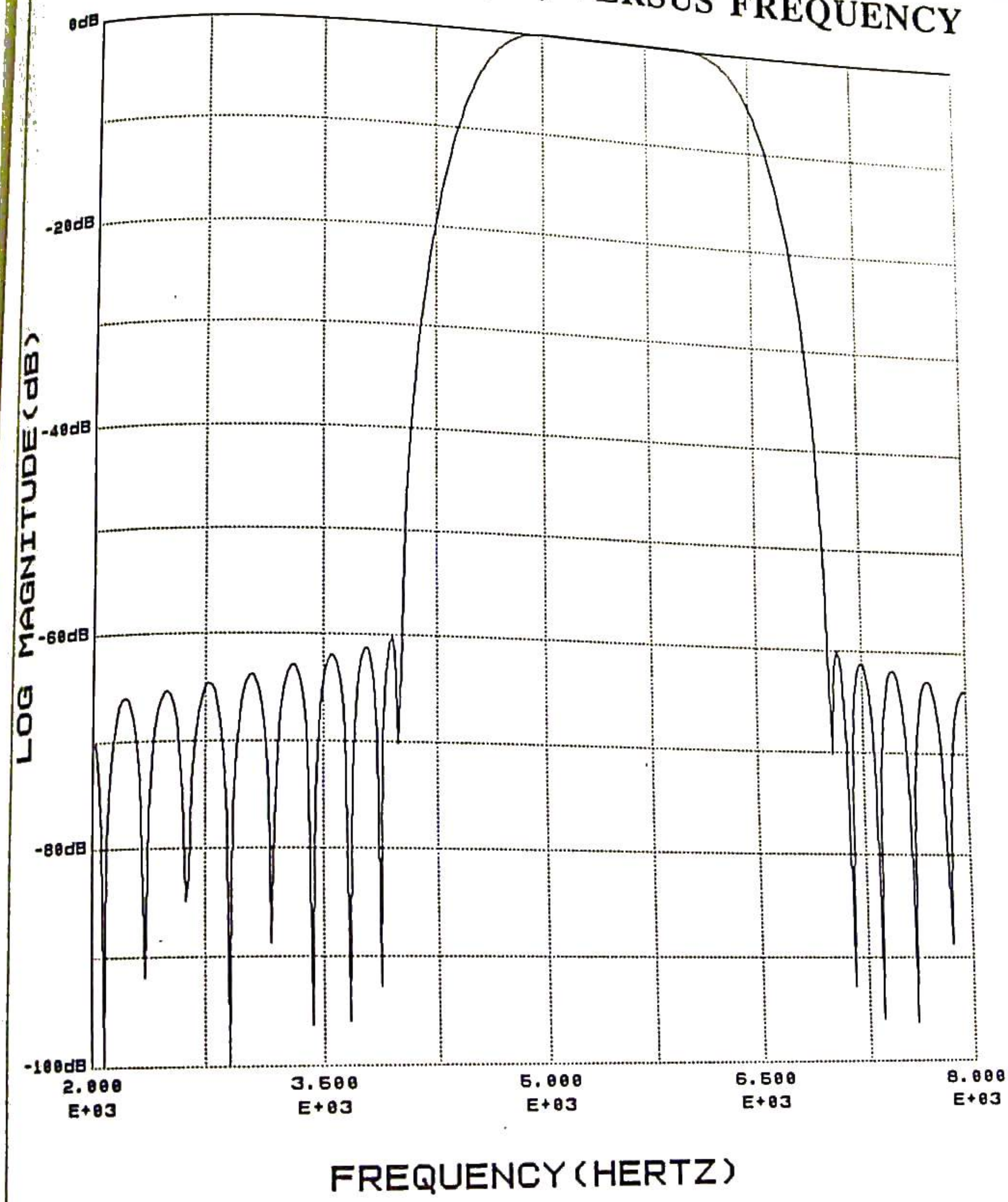


PASSBAND RIPPLE IN -dB	-6.0000	
STOPBAND RIPPLE IN -dB	-60.0000	
PASSBAND CUTOFF FREQUENCIES	5000.00	6000.00
STOPBAND CUTOFF FREQUENCIES	4000.00	7000.00
SAMPLING FREQUENCY	22000.0	
ALL FREQUENCIES IN HERTZ		
QUANTIZATION 48 BITS - FLOATING POINT		

BAND PASS FILTER
FIR (WINDOW) DESIGN
KAISER WINDOW
NUMBER OF TAPS: 79

FILTER DESIGN & ANALYSIS SYSTEM
MOMENTUM DATA SYSTEMS, INC.

LOG MAGNITUDE (dB) VERSUS FREQUENCY



PASSBAND RIPPLE IN -dB	-5.0000	
STOPBAND RIPPLE IN -dB	-60.0000	
PASSBAND CUTOFF FREQUENCIES	5000.00	6000.00
STOPBAND CUTOFF FREQUENCIES	4000.00	7000.00
SAMPLING FREQUENCY	22000.0	

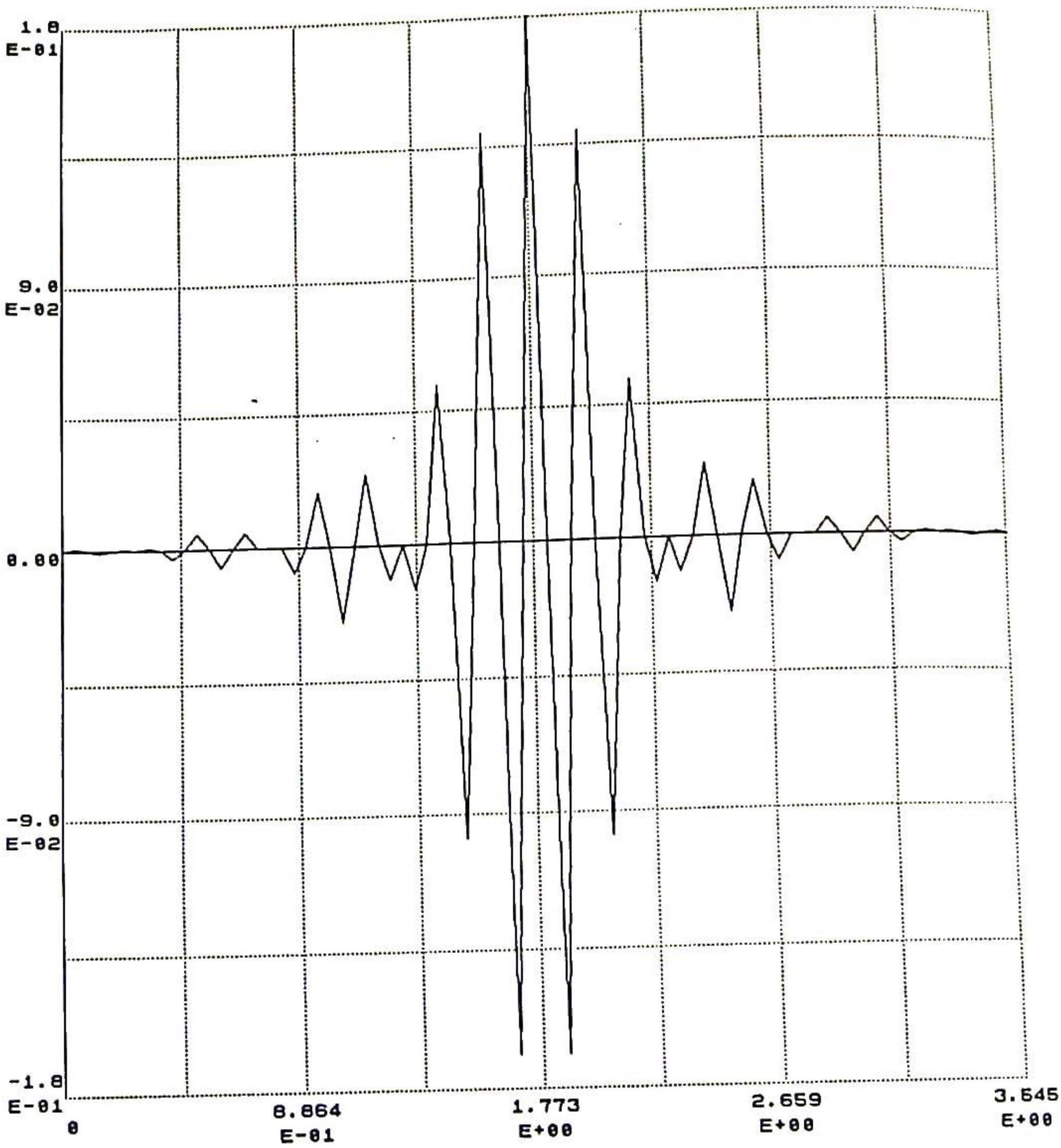
ALL FREQUENCIES IN HERTZ
QUANTIZATION 48 BITS - FLOATING POINT

BAND PASS FILTER
FIR (WINDOW) DESIGN
KAISER WINDOW
NUMBER OF TAPS: 79

FILTER DESIGN & ANALYSIS SYSTEM
MOMENTUM DATA SYSTEMS, INC.

NCD02 - 1211

IMPULSE RESPONSE VERSUS TIME



PASSBAND RIPPLE IN -dB	-5.0000	
STOPBAND RIPPLE IN -dB	-60.0000	
PASSBAND CUTOFF FREQUENCIES	5000.00	6000.00
STOPBAND CUTOFF FREQUENCIES	4000.00	7000.00
SAMPLING FREQUENCY	22000.0	
ALL FREQUENCIES IN HERTZ		
QUANTIZATION 48 BITS - FLOATING POINT		

BAND PASS FILTER
FIR (WINDOW) DESIGN
KAISER WINDOW
NUMBER OF TAPS: 79

FILTER DESIGN & ANALYSIS SYSTEM
MOMENTUM DATA SYSTEMS, INC.

NCD02 - 1212

COEFF.FLT

```
FILTER COEFFICIENT FILE
      FIR DESIGN
            79
         004F
           24
         0018
            0
         0000
```

Fractional		or	Floating Point	
0	00000000		.0000000000000000D+00	0000000000000000
4936	00001348		-.1521742343902588D-01	BF8F2A5000000000
0	00000000		.0000000000000000D+00	0000000000000000
-6834	FFFFE54E		.5884170532226562D-03	3F43480000000000
0	00000000		.0000000000000000D+00	0000000000000000
4133	00001025		-.8146762847900391D-03	BF4AB20000000000
0	00000000		.0000000000000000D+00	0000000000000000
6258	00001872		.4926919937133789D-03	3F40250000000000
0	00000000		.0000000000000000D+00	0000000000000000
-24227	FFFFA15D		.7460117340087891D-03	3F48720000000000
0	00000000		.0000000000000000D+00	0000000000000000
44182	0000AC96		-.2888083457946777D-02	BF67A8C000000000
0	00000000		.0000000000000000D+00	0000000000000000
-54999	FFFF2929		.5266904830932617D-02	3F7592C000000000
0	00000000		.0000000000000000D+00	0000000000000000
43333	0000A945		-.6556391716003418D-02	BF7ADAE000000000
0	00000000		.0000000000000000D+00	0000000000000000
0	00000000		.5165696144104004D-02	3F7528A000000000
0	00000000		.0000000000000000D+00	0000000000000000
-72663	FFFEE429		.0000000000000000D+00	0000000000000000
0	00000000		.0000000000000000D+00	0000000000000000
155861	000260D5		-.8662104606628418D-02	BF81BD7000000000
0	00000000		.0000000000000000D+00	0000000000000000
-215096	FFFCB7C8		.1858007907867432D-01	3F9306A800000000
0	00000000		.0000000000000000D+00	0000000000000000
208147	00032D13		-.2564144134521484D-01	BF9A41C000000000
0	00000000		.0000000000000000D+00	0000000000000000
-98855	FFFE7DD9		.2481305599212646D-01	3F99689800000000
0	00000000		.0000000000000000D+00	0000000000000000
-127653	FFFE0D5B		-.1178443431854248D-01	BF88227000000000

⋮ ⋮

NCD02-1213

CODE GENERATOR

=MGEN

FILTER DESIGN AND ANALYSIS SYSTEM

CODE GENERATOR

MOMENTUM DATA SYSTEMS

HIT F10 KEY TO CONTINUE

Copyright 1988

NCD02-1214

CODE GENERATION SEQUENCE

Screen 1

Filter Design and Analysis System
Motorola DSP56000 Generator Menu

Select Filter Coefficient File 1
Exit to DOS X
Enter Action Desired 1

After Entering Action Desired, Hit F10 Key To Start System

Screen 2

List of Coefficient Files Available
Enter an X next to the File to be Loaded

COEFF1.FLT FIR1.FLT COEFF.FLT x

↵ :accept field
F1/F10:accept screeen Esc:main menu ↑:previous field ↓:next field

Screen 3

Assembly language code for implementation of the filter
has been generated

This code is in file COEFF.ASM

HIT F1/F10 TO RETURN TO MAIN MENU

NCD02-1215

COEFF.ASM LISTING (1 OF 3)

```
Motorola DSP56000 Macro Cross Assembler  Version 2.01  88-05-02  11:33:09  coeff.asm  Page 1

                        COEFF     ident   1,0
                                  include 'head4.asm'
                                  page    132,66,3,3
                                  opt     cex,mex
                        ;
                        ; This program implements an FIR filter in direct form
                        ;
          0000FFFF      datin     equ     $ffff                           ;location in Y memory of input file
          0000FFFF      datout    equ     $ffff                           ;location in Y memory of output file
          0000000E      xx        equ     @cvi(20480/(64*22.0))
          0000FFF0      m_scr     equ     $fff0                           ; sci control register
          0000FFF1      m_ssr     equ     $fff1                           ; sci status register
          0000FFF2      m_sccr    equ     $fff2                           ; sci clock control register
          0000FFE1      m_pcc     equ     $ffe1                           ; port c control register
          0000FFFF      m_ipr     equ     $ffff                           ; interrupt priority register
          0000004F      ntaps     equ     79                              ;number of coefficients
                                  include 'direct1.asm '
                        direct1   macro   ntaps
m                       ;
m                       ;   FIR Finite Impulse Response - Direct form filter
m                       ;
m                       ;    x0     -    input sample
m                       ;    a      -   output sample
m                       ;    ntaps  -    number of filter coefficients i.e. the filter order + 1
m                       ;
m                                 clr     a         x0,x:(r0)+  y:(r4)+,y0  ;save input sample
m                                 rep     #ntaps-1                          ;loop ntaps-1 times
m                                 mac     x0,y0,a   x:(r0)+,x0  y:(r4)+,y0  ;compute b(i)*x(n-i)
m                                 macr    x0,y0,a   (r0)-                   ;round result,reset r0
m                                 endm
          X:0000                  org     x:0
          X:0000        states    dsm     ntaps                             ;x(n-i)  saved input values
          Y:0000                  org     y:0
                        coef
d         Y:0000 000000           dc      0                                 ;b(  0)
d         Y:0001 00134B           dc      4936                              ;b(  1)
d         Y:0002 000000           dc      0                                 ;b(  2)
d         Y:0003 FFE54E           dc      -6834                             ;b(  3)
d         Y:0004 000000           dc      0                                 ;b(  4)
d         Y:0005 001025           dc      4133                              ;b(  5)
d         Y:0006 000000           dc      0                                 ;b(  6)
d         Y:0007 001872           dc      6258                              ;b(  7)
d         Y:0008 000000           dc      0                                 ;b(  8)
d         Y:0009 FFA15D           dc      -24227                            ;b(  9)
d         Y:000A 000000           dc      0                                 ;b( 10)
d         Y:000B 00AC96           dc      44182                             ;b( 11)
d         Y:000C 000000           dc      0                                 ;b( 12)
d         Y:000D FF2929           dc      -54999                            ;b( 13)
d         Y:000E 000000           dc      0                                 ;b( 14)
d         Y:000F 00A945           dc      43333                             ;b( 15)
d         Y:0010 000000           dc      0                                 ;b( 16)
d         Y:0011 000000           dc      0                                 ;b( 17)
d         Y:0012 000000           dc      0                                 ;b( 18)
d         Y:0013 FEE429           dc      -72663                            ;b( 19)
d         Y:0014 000000           dc      0                                 ;b( 20)
d         Y:0015 0260D5           dc      155861                            ;b( 21)
```

NCD02-1216

COEFF.ASM LISTING (2 OF 3)


```
57   d   Y:0016 000000   dc   0          ;b( 22)
58   d   Y:0017 FCB7C8   dc   -215096    ;b( 23)
59   d   Y:0018 000000   dc   0          ;b( 24)
60   d   Y:0019 032D13   dc   208147     ;b( 25)
61   d   Y:001A 000000   dc   0          ;b( 26)
62   d   Y:001B FE7DD9   dc   -98855     ;b( 27)
63   d   Y:001C 000000   dc   0          ;b( 28)
64   d   Y:001D FE0D5B   dc   -127653    ;b( 29)
65   d   Y:001E 000000   dc   0          ;b( 30)
66   d   Y:001F 06EE20   dc   454176     ;b( 31)
67   d   Y:0020 000000   dc   0          ;b( 32)
68   d   Y:0021 F35523   dc   -830173    ;b( 33)
69   d   Y:0022 000000   dc   0          ;b( 34)
70   d   Y:0023 120952   dc   1182034    ;b( 35)
71   d   Y:0024 000000   dc   0          ;b( 36)
72   d   Y:0025 EA2438   dc   -1432520   ;b( 37)
73   d   Y:0026 000000   dc   0          ;b( 38)
74   d   Y:0027 173E33   dc   1523251    ;b( 39)
75   d   Y:0028 000000   dc   0          ;b( 40)
76   d   Y:0029 EA2438   dc   -1432520   ;b( 41)
77   d   Y:002A 000000   dc   0          ;b( 42)
78   d   Y:002B 120952   dc   1182034    ;b( 43)
79   d   Y:002C 000000   dc   0          ;b( 44)
80   d   Y:002D F35523   dc   -830173    ;b( 45)
81   d   Y:002E 000000   dc   0          ;b( 46)
82   d   Y:002F 06EE20   dc   454176     ;b( 47)
83   d   Y:0030 000000   dc   0          ;b( 48)
84   d   Y:0031 FE0D5B   dc   -127653    ;b( 49)
85   d   Y:0032 000000   dc   0          ;b( 50)
86   d   Y:0033 FE7DD9   dc   -98855     ;b( 51)
87   d   Y:0034 000000   dc   0          ;b( 52)
88   d   Y:0035 032D13   dc   208147     ;b( 53)
89   d   Y:0036 000000   dc   0          ;b( 54)
90   d   Y:0037 FCB7C8   dc   -215096    ;b( 55)
91   d   Y:0038 000000   dc   0          ;b( 56)
92   d   Y:0039 0260D5   dc   155861     ;b( 57)
93   d   Y:003A 000000   dc   0          ;b( 58)
94   d   Y:003B FEE429   dc   -72663     ;b( 59)
95   d   Y:003C 000000   dc   0          ;b( 60)
96   d   Y:003D 000000   dc   0          ;b( 61)
97   d   Y:003E 000000   dc   0          ;b( 62)
98   d   Y:003F 00A945   dc   43333      ;b( 63)
99   d   Y:0040 000000   dc   0          ;b( 64)
100  d   Y:0041 FF2929   dc   -54999     ;b( 65)
101  d   Y:0042 000000   dc   0          ;b( 66)
102  d   Y:0043 00AC96   dc   44182      ;b( 67)
103  d   Y:0044 000000   dc   0          ;b( 68)
104  d   Y:0045 FFA15D   dc   -24227     ;b( 69)
105  d   Y:0046 000000   dc   0          ;b( 70)
106  d   Y:0047 001872   dc   6258       ;b( 71)
107  d   Y:0048 000000   dc   0          ;b( 72)
108  d   Y:0049 001025   dc   4133       ;b( 73)
109  d   Y:004A 000000   dc   0          ;b( 74)
110  d   Y:004B FFE54E   dc   -6834      ;b( 75)
111  d   Y:004C 000000   dc   0          ;b( 76)
112  d   Y:004D 001348   dc   4936       ;b( 77)
```

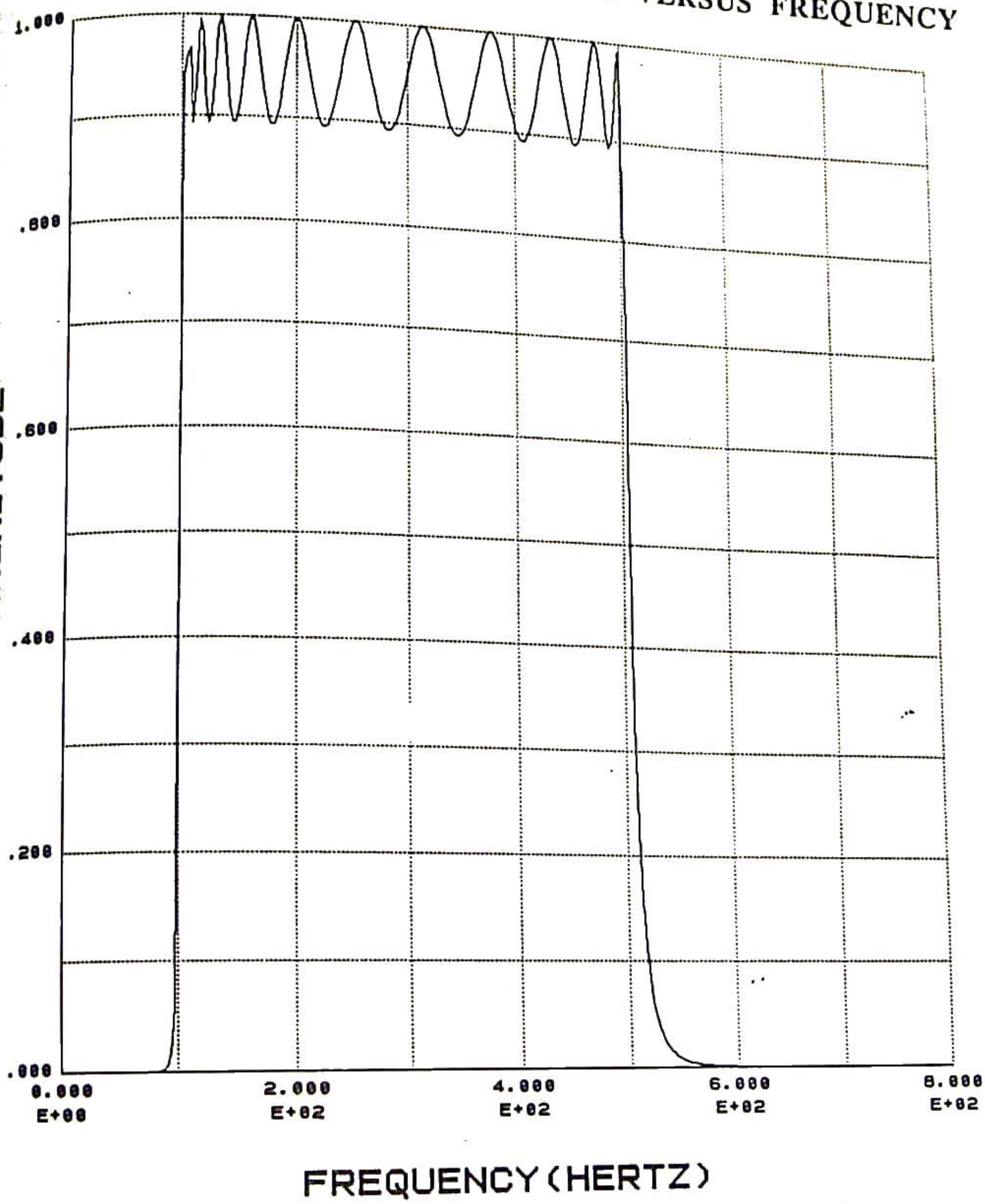
NCD02-1217

COEFF.ASM LISTING (3 OF 3)


```
113  d    Y:004E 000000            dc      0                                  ;b(  78)
114                                include 'body4.asm'
115       00000040       start     equ     $40                                ;origin for user program
116
117       P:0000                   org     p:$0                               ;origin for reset vector
118       P:0000 0C0040            jmp     start                              ;jump to 'start' on system reset
119
120       P:001C                   org     p:$1c                              ;origin for timer interrupt vector
121       P:001C 0BF080            jsr     filter                             ;jump to 'filter' on timer interrupt
                 000052
122
123       P:0040                   org     p:start                            ;origin for user program
124
125       P:0040 0003F8            ori     #3,mr                              ;disable all interrupts
126       P:0041 08F4B2            movep               #(xx-1),x:m_sccr       ;cd=xx-1 for divide by xx
                 00000D
127       P:0043 08F4A1            movep               #$7,x:m_pcc            ;set cc(2;0) to turn on timer
                 000007
128       P:0045 08F4B0            movep               #$2000,x:m_scr         ;enable timer interrupts
                 002000
129       P:0047 08F4BF            movep               #$c000,x:m_ipr         ;set interrupt priority for sci
                 00C000
130
131       P:0049 300000            move                #states,r0             ;point to filter states
132       P:004A 054EA0            move                #ntaps-1,m0            ;circular buffer modulo(ntaps)
133       P:004B 340000            move                #coef,r4               ;point to filter coefficients
134       P:004C 054EA4            move                #ntaps-1,m4            ;circular buffer modulo(ntaps)
135
136       P:004D 200013            clr     a                                  ;initialize internal state storage
137       P:004E 064FA0            rep     #ntaps                             ;*  set memory to zero
138       P:004F 565800            move                a,x:(r0)+              ;*
139
140       P:0050 00FCB8            andi    #$fc,mr                            ;enable timer interrupt
141       P:0051 0C0051            jmp     *                                  ;wait for interrupt
142
143                         filter
144       P:0052 09443F            movep               y:datin,x0             ;get sample
145
146                                direct1 ntaps                              ;do direct form FIR filter
147  +                      ;
148  +                      ;   FIR Finite Impulse Response - Direct form filter
149  +                      ;
150  +                      ;    x0     -    input sample
151  +                      ;    a      -   output sample
152  +                      ;    ntaps  -    number of filter coefficients i.e. the filter order + 1
153  +                      ;
154  +    P:0053 F01813            clr     a          x0,x:(r0)+  y:(r4)+,y0  ;save input sample
155  +    P:0054 064EA0            rep     #ntaps-1                           ;loop ntaps-1 times
156  +    P:0055 F098D2            mac     x0,y0,a    x:(r0)+,x0  y:(r4)+,y0  ;compute b(i)*x(n-i)
157  +    P:0056 2050D3            macr    x0,y0,a    (r0)-                   ;round result,reset r0
158
159       P:0057 09CE3F            movep               a,y:datout             ;output sample
160       P:0058 000004            rti
161                                end
0    Errors
0    Warnings
```

NCD02-1218

FLOATING - POINT MAGNITUDE VERSUS FREQUENCY



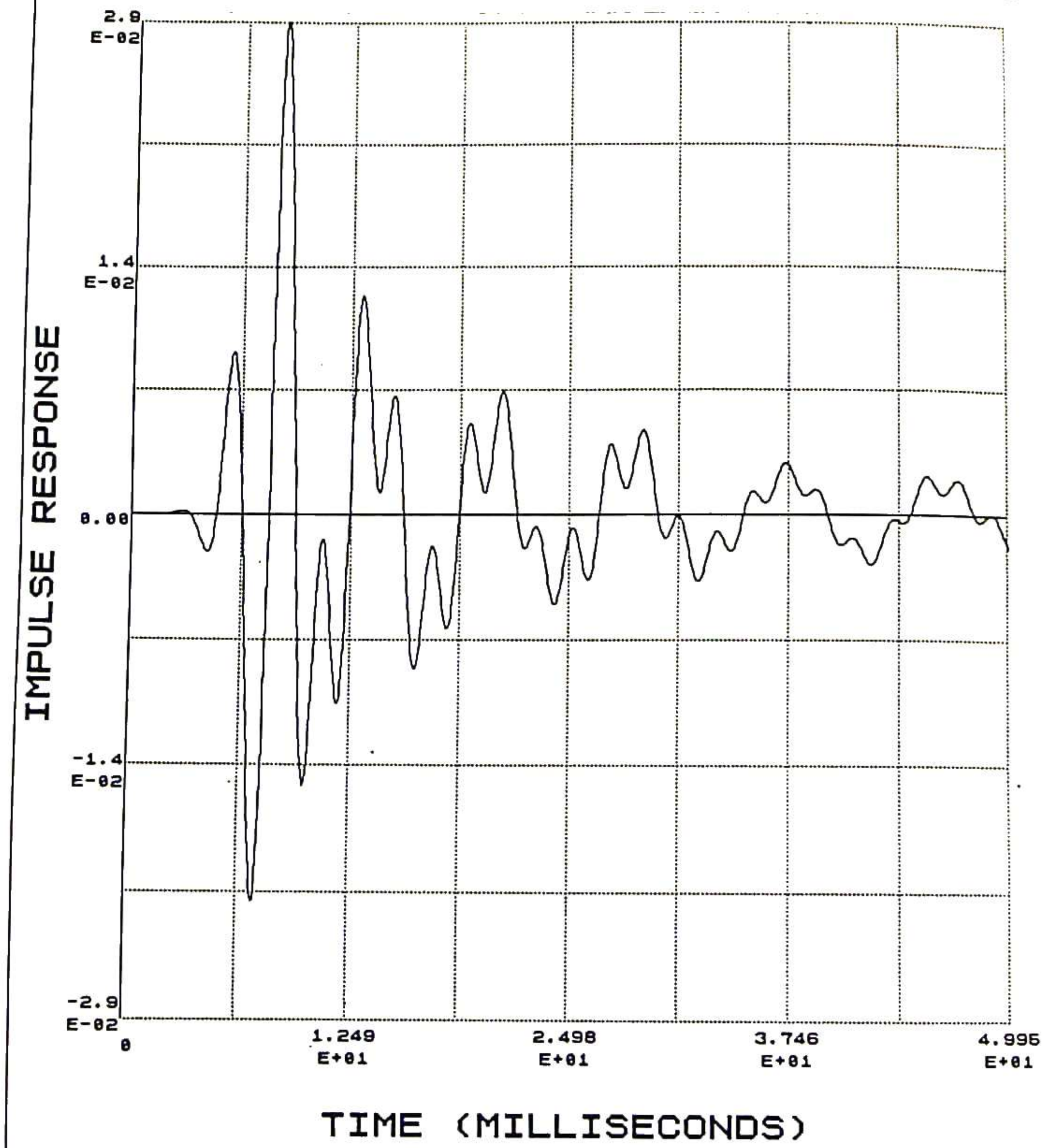
PASSBAND RIPPLE IN -dB	-1.0000	
STOPBAND RIPPLE IN -dB	-60.0000	
PASSBAND CUTOFF FREQUENCIES	100.000	500.000
STOPBAND CUTOFF FREQUENCIES	50.0000	600.000
SAMPLING FREQUENCY	20000.0	
ALL FREQUENCIES IN HERTZ		
QUANTIZATION 48 BITS - FLOATING POINT		

BAND PASS FILTER
IIR DESIGN - BILINEAR TRANS.
TSCHEBYSCHEFF
FILTER ORDER: 24

FILTER DESIGN & ANALYSIS SYSTEM
MOMENTUM DATA SYSTEMS, INC.

NCD02 - 1219-1

FLOATING - POINT IMPULSE RESPONSE VERSUS TIME



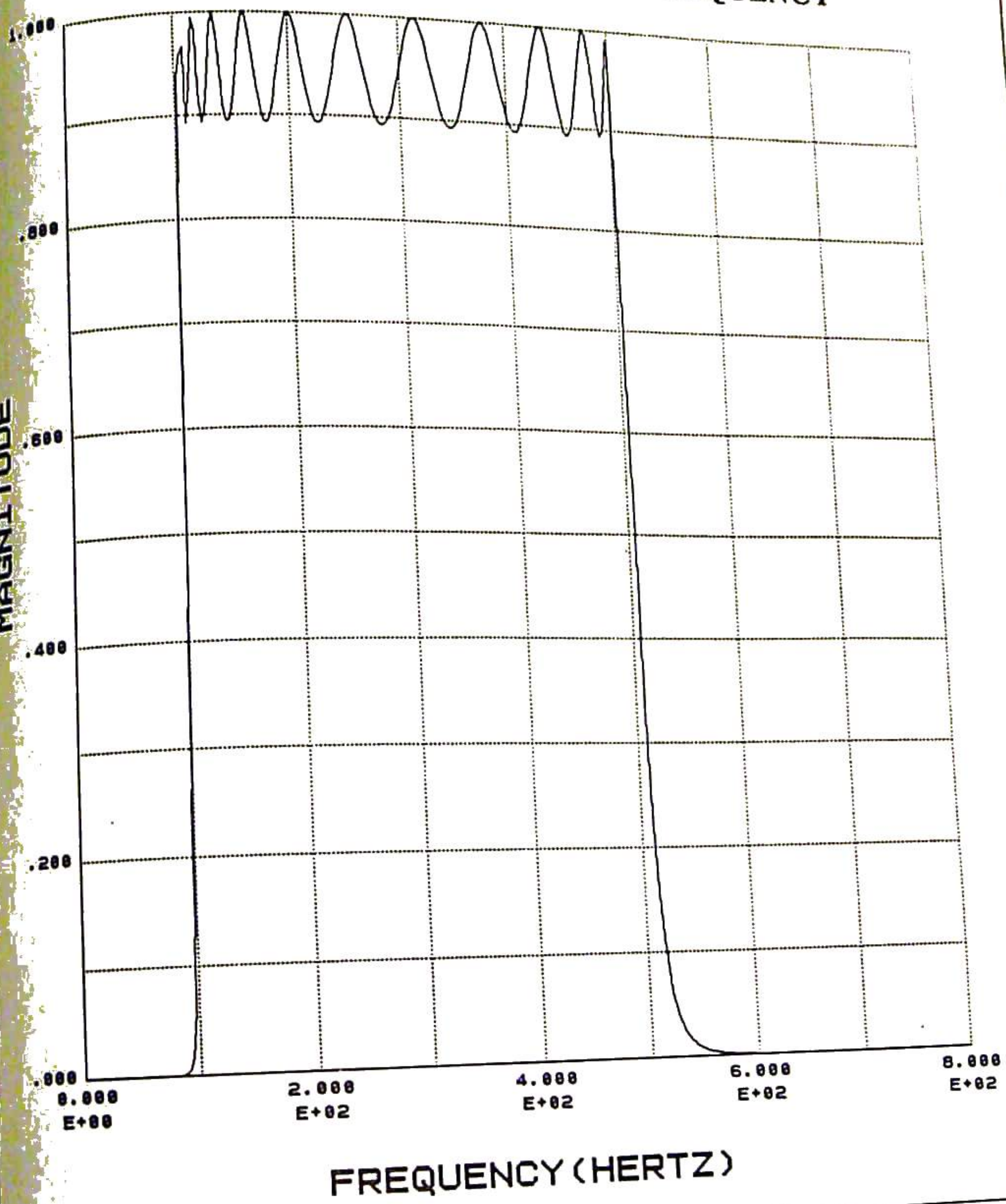
PASSBAND RIPPLE IN -dB	-1.0000	
STOPBAND RIPPLE IN -dB	-60.0000	
PASSBAND CUTOFF FREQUENCIES	100.000	500.000
STOPBAND CUTOFF FREQUENCIES	50.0000	600.000
SAMPLING FREQUENCY	20000.0	
ALL FREQUENCIES IN HERTZ		
QUANTIZATION 48 BITS - FLOATING POINT		

BAND PASS FILTER
IIR DESIGN - BILINEAR TRANS.
TSCHEBYSCHEFF
FILTER ORDER: 24

FILTER DESIGN & ANALYSIS SYSTEM
MOMENTUM DATA SYSTEMS, INC.

NCD02 - 1220-1

24-BIT MAGNITUDE VERSUS FREQUENCY



PASSBAND RIPPLE IN -dB	-1.0000	
STOPBAND RIPPLE IN -dB	-60.0000	
PASSBAND CUTOFF FREQUENCIES	100.000	500.000
STOPBAND CUTOFF FREQUENCIES	50.0000	600.000
SAMPLING FREQUENCY	20000.0	

ALL FREQUENCIES IN HERTZ

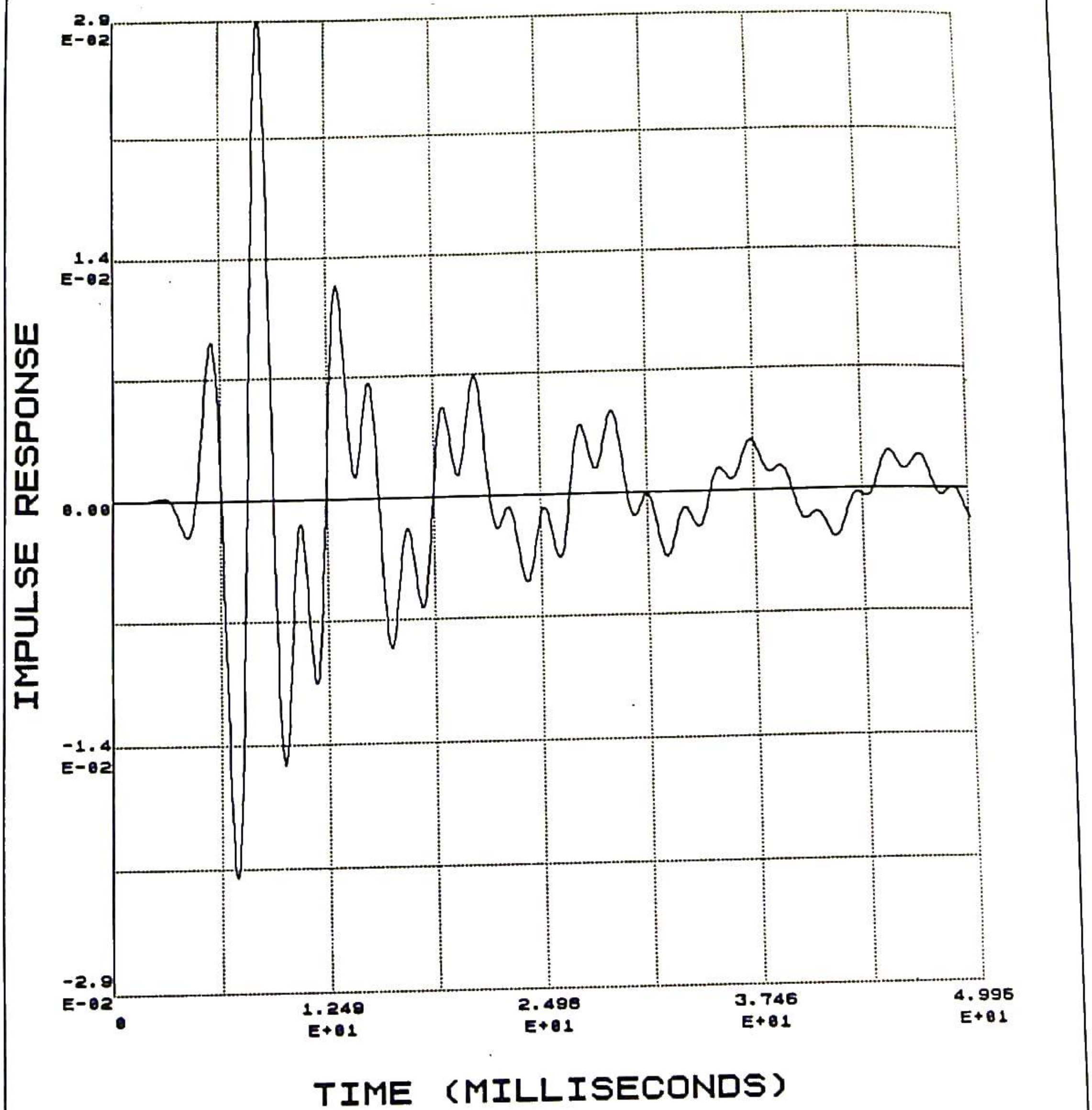
QUANTIZATION 24 BITS FIXED POINT FRACTIONAL

BAND PASS FILTER
IIR DESIGN - BILINEAR TRANS.
TSCHEBYSCHEFF
FILTER ORDER: 24

FILTER DESIGN & ANALYSIS SYSTEM
MOMENTUM DATA SYSTEMS, INC.

NCD02 - 1221-1

IMPULSE RESPONSE VERSUS TIME



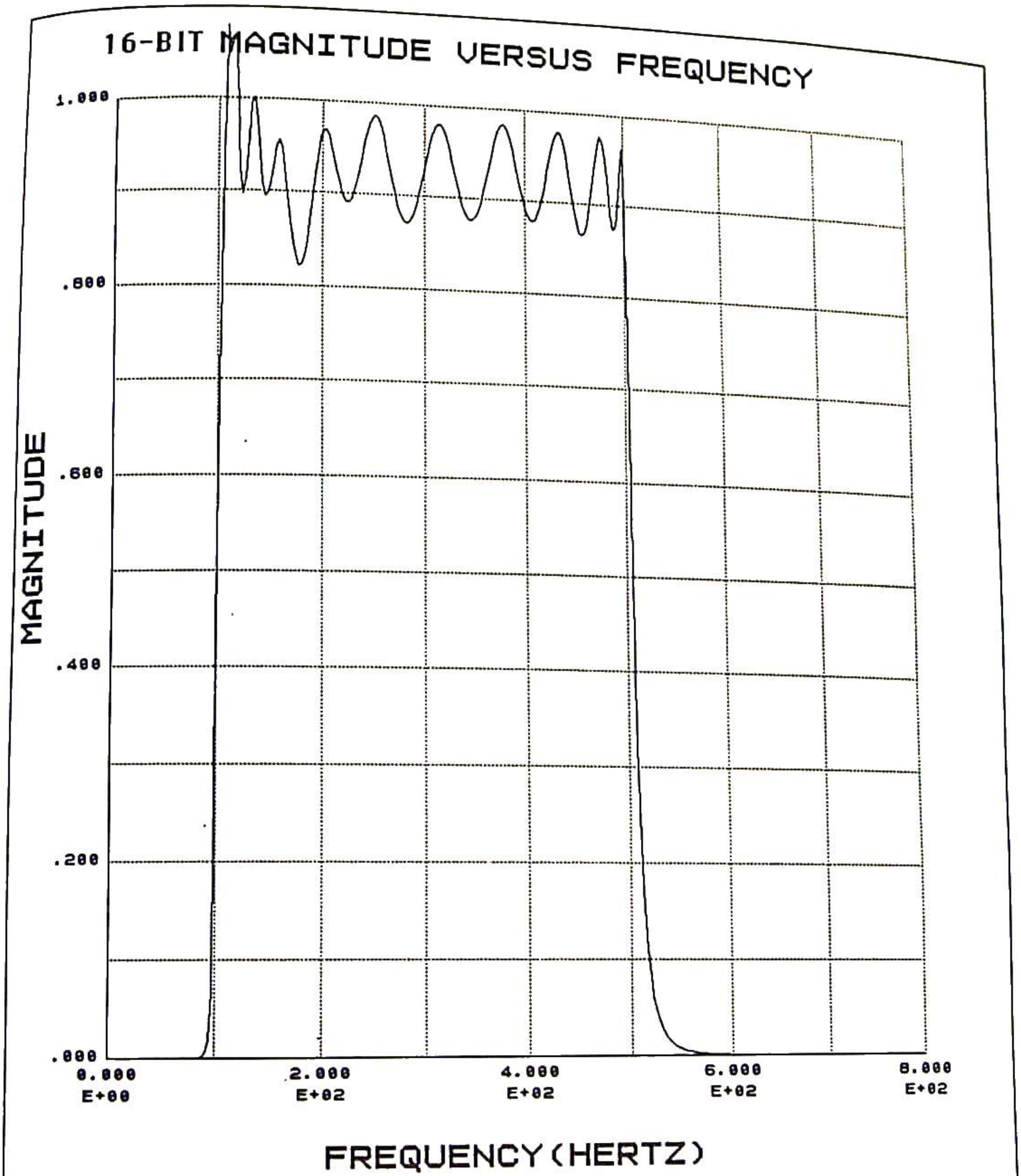
PASSBAND RIPPLE IN -dB	-1.0000	
STOPBAND RIPPLE IN -dB	-60.0000	
PASSBAND CUTOFF FREQUENCIES	100.000	500.000
STOPBAND CUTOFF FREQUENCIES	50.0000	600.000
SAMPLING FREQUENCY	20000.0	

ALL FREQUENCIES IN HERTZ
QUANTIZATION 24 BITS FIXED POINT FRACTIONAL

BAND PASS FILTER
IIR DESIGN - BILINEAR TRANS.
TSCHEBYSCHEFF
FILTER ORDER: 24

FILTER DESIGN & ANALYSIS SYSTEM
MOMENTUM DATA SYSTEMS, INC.

NCD02- 1222-1



PASSBAND RIPPLE IN -dB	-1.0000	
STOPBAND RIPPLE IN -dB	-60.0000	
PASSBAND CUTOFF FREQUENCIES	100.000	500.000
STOPBAND CUTOFF FREQUENCIES	50.0000	600.000
SAMPLING FREQUENCY	20000.0	

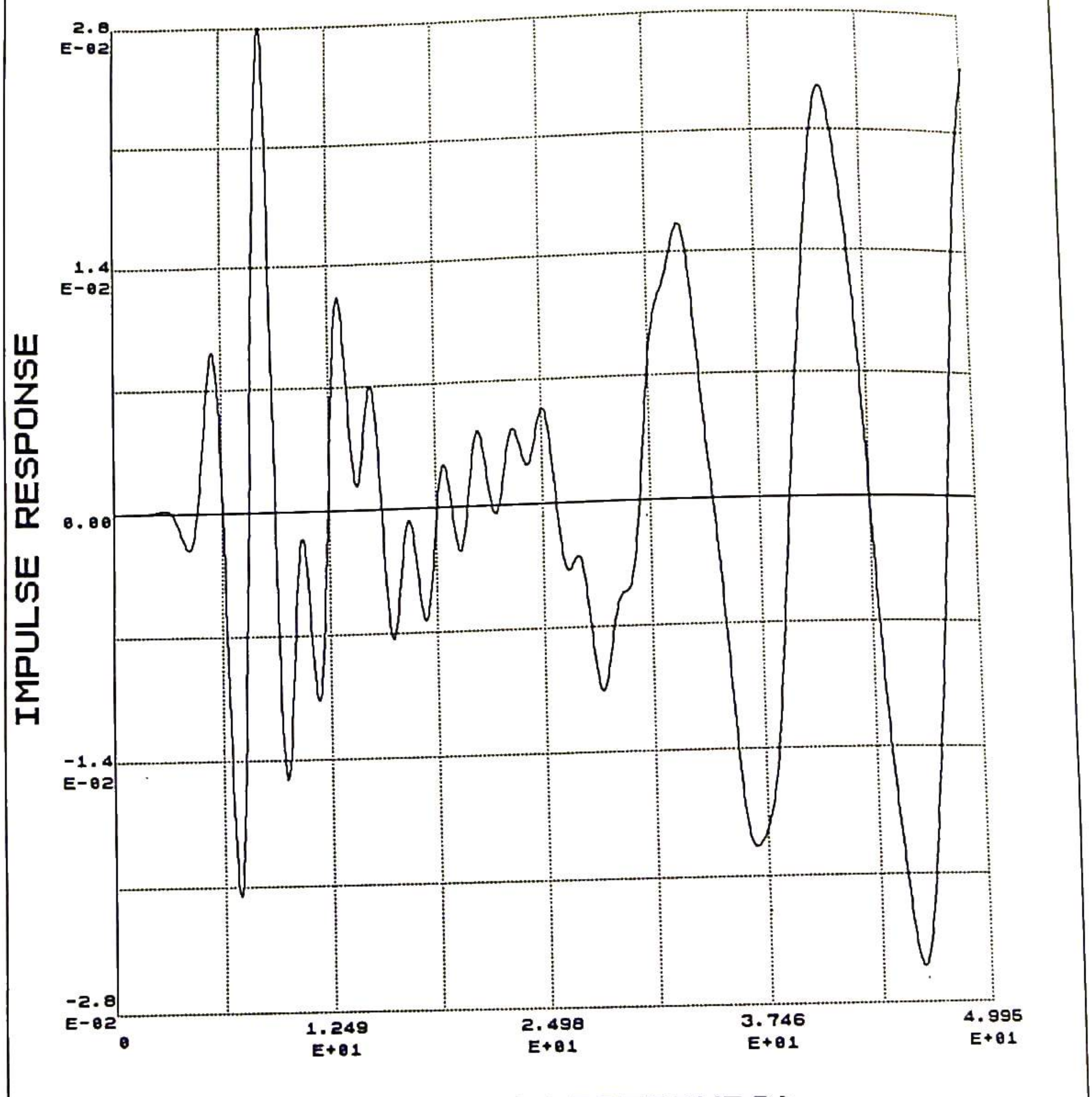
ALL FREQUENCIES IN HERTZ
QUANTIZATION 16 BITS FIXED POINT FRACTIONAL

BAND PASS FILTER
IIR DESIGN - BILINEAR TRANS.
TSCHEBYSCHEFF
FILTER ORDER: 24

FILTER DESIGN & ANALYSIS SYSTEM
MOMENTUM DATA SYSTEMS, INC.

NCD02 - 1223

IMPULSE RESPONSE VERSUS TIME



PASSBAND RIPPLE IN -dB	-1.0000	
STOPBAND RIPPLE IN -dB	-60.0000	
PASSBAND CUTOFF FREQUENCIES	100.000	500.000
STOPBAND CUTOFF FREQUENCIES	50.0000	600.000
SAMPLING FREQUENCY	20000.0	

ALL FREQUENCIES IN HERTZ
QUANTIZATION 16 BITS FIXED POINT FRACTIONAL

BAND PASS FILTER
IIR DESIGN - BILINEAR TRANS.
TSCHEBYSCHEFF
FILTER ORDER: 24

FILTER DESIGN & ANALYSIS SYSTEM
MOMENTUM DATA SYSTEMS, INC.

NCD02 - 1224