## NCR 605 PROGRAMMING INFORMATION
## CONTENTS



| REVISION RECORD | |
|---|---|
| **Date** | **Affected Pages/Remarks** |
| May 73 | First printing as MS-785 |
| Mar. 74 | Revised and reprinted |
| Aug. 76 | Revised and reprinted as (ST-9167-06) |
| | |
| | |
| | |

ST-9167-06

# NCR 605 PROGRAMMING INFORMATION
## CONTENTS



| REVISION RECORD | |
|---|---|
| **Date** | **Affected Pages/Remarks** |
| May 73 | First printing as MS-785 |
| Mar. 74 | Revised and reprinted |
| Aug. 76 | Revised and reprinted as (ST-9167-06) |
| | |
| | |
| | |

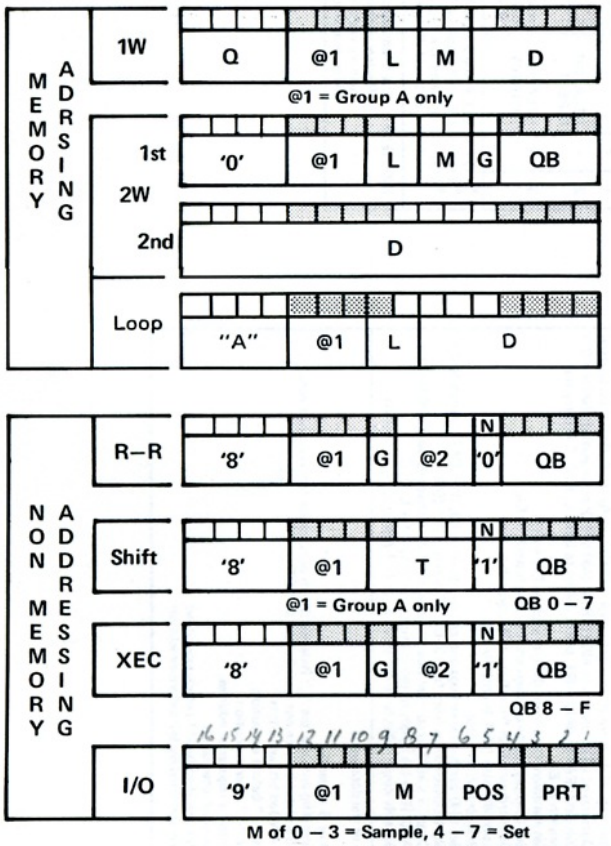
ST-9167-06

# NCR 605 PROGRAMMING INFORMATION

## PROGRAMMING INTRODUCTION

This section contains information pertaining to the programming of the 605. Included are tables for command decode and description, and procedures for clearing and filling memory by program. Refer to "Glossary & Commands Descriptions" for a more detailed description of the commands.

## 605 COMMAND FORMAT




D = Adrs. Bits or Data Word
G = Aux Reg. Group; 0 = A, 1 = B
L = Mem. Adrs. Mode Bits only
M = Mem. Adrs. & I/O Mode Bits
N = R–R or Shift/Xec. bit
POS = I/O Position 0 – 7
PRT = I/O Port 0 – 7
Q = Command Code
QB = Secondary Q
T = # of Times to Shift
@1 = Aux Register receiving result of command
@2 = 2nd Aux Register, used as an operand

## REGISTER – REGISTER COMMANDS

| Q | N | QB | Op. Code | Remarks |
|---|---|---|---|---|
| 8 | 0 | 0 | RFSD♦ | Dec. field sub. (@2) from (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if no borrow |
| | | 1 | RFAD♣ | Dec. field add (@2) to (@1) → @1; cmpr ((@1)) to 0 for L, E, G; C if carry-out |
| | | 2 | RFSB♦ | Bin. field sub. (@2) from (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if no borrow |
| | | 3 | RFAB♣ | Bin. field add (@2) to (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if carry-out |
| | | 4 | REXC | Exchange (@2) with (@1) and (@1) with (@2) |
| | | 5 | ROR■ | OR (@2) bit-by-bit to (@1) → @1; cmpr. ((@1)) to 0 for L, E, G |
| | | 6 | RXOR★ | Half-add (@2) bit-by-bit with (@1) → @1; cmpr. ((@1)) to 0 for L, E, G |
| | | 7 | RMVC | Move the binary complement of (@2) → @1 |
| | | 8 | RDSUB | Dec. sub. (@2) from (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if no borrow |
| | | 9 | RDAD | Dec. add (@2) to (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if carry-out |
| | | A | RBSUB | Bin. sub. (@2) from (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if no borrow |
| | | B | RBAD | Bin. add (@2) to (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if carry-out |
| | | C | RMV | Move (@2) → @1; cmpr. ((@1)) to 0 for L, E, G |
| | | D | RBIT | Cmpr. 1's in (@2) to = bit pos. in (@1); set B if all checked bits in (@1) = 1's |
| | | E | RAND▲ | AND (@2) bit-by-bit to (@1) → @1; cmpr. ((@1)) to 0 for L, E, G |
| 8 | 0 | F | RCMPR | Binarily cmpr. (@1) to (@2); set L, E, or G if (@1) is <, =, or > (@2) |

♦ The result has (C-1) added to it
♣ The result has (C) added to it
■ Store a 1-bit in ((@1)) when either ORed bit is a 1-bit
★ Store a 1-bit in ((@1)) only if both Exclusive-ORed bits are unequal
▲ Store a 1-bit in ((@1)) only if both ANDed bits are 1-bits

## SHIFT COMMANDS

| Q | N | QB | Op. Code | Remarks |
|---|---|---|---|---|
| 8 | 1 | 0 | SLD* | Shift (@1) and (@1+1) left 0 – 15 positions as indicated by "T" |
| | | 1 | SLC | Shift (@1) left circular 0 – 15 positions as indicated by "T" |
| | | 2 | SLO | Shift (@1) left out 0 – 15 positions as indicated by "T" |
| | | 3 | SLX | Shift (@1) left 0 – 15 positions per "T"; set C if a 1-bit shifts out |
| | | 4 | SRD* | Shift (@1) and (@1+1) right 0 – 15 positions as indicated by "T" |
| | | 5 | SRC | Shift (@1) circular 0 – 15 positions as indicated by "T" |
| | | 6 | SRO | Shift (@1) right out 0 – 15 positions as indicated by "T" |
| 8 | 1 | 7 | SRA | Shift (@1) right and spread bit 16 to the right 0 – 15 positions as per "T" |

* Since (MAR) → SRF during an error trap, a "T" of 0 recovers the error adrs. by loading (SRF) → @1+1 during an SLD or SRD. The (@1) are unchanged.

## EXECUTE COMMANDS

| Q | N | QB | Op. Code | Remarks |
|---|---|---|---|---|
| 8 | 1 | 8 | XEC0 | Execute 0 } Since Execute commands are not presently available, an Execute command causes an NXC error. |
| 8 | 1 | F | XEC7 | Execute 7 } |

## I/O COMMANDS

| Q | @1 | M | Op. Code | Remarks |
|---|---|---|---|---|
| 9 | ** | 0–3 | SMPL | Port data → @1; cmpr. transferred word to 0 to set L, E, G |
| 9 | | 4–7 | SET | (@1) → port; cmpr. transferred word to 0 to set L, E, G |

** Group B if @1 = 0 – 3; group A if @1 = 4 – 7

ST-9167-06

## MEMORY ADDRESSING COMMANDS★

| 1-Word Q♦ | @1 | 2-Word QB▲ | Op. Code | Remarks |
|---|---|---|---|---|
| – | – | 0 | WAITI | Wait for INT, COM, or ET. Wait Code = EA or D if adrs. mode is literal |
| 1 | – | 1 | ST | Store (@1) into memory at EA |
| 2 | – | 2 | STM | Store (@1) thru aux. reg. 0 into memory starting at EA |
| 3 | – | 3 | LD | Load (EA) → @1 ; compr. ((@1)) to 0 to set L, E, G |
| 4 | – | 4 | LDM | Load @1 thru aux. reg. 0 from memory starting at EA |
| 6 | 0 | 6 | JLR | (EA) or D → CR |
| \| | 1 | \| | \| | (EA) or D → A0 or G0; (EA+1) or D → CR |
| \| | 2 | \| | \| | (EA) or D → A1 or G1; (EA+1) or D → A0 or G0; (EA+2) or D → CR |
| \| | 3 | \| | \| | (EA) or D → A2 or G2; ... etc ... (EA+2) or D → A0 or G0; (EA+3) → CR |
| \| | 4 | \| | \| | (EA) or D → A3 or G3; ... etc ... (EA+3) or D → A0 or G0; (EA+4) → CR |
| \| | 5 | \| | \| | (EA) or D → A4 or G4; ... etc ... (EA+4) or D → A0 or G0; (EA+5) → CR |
| \| | 6 | \| | \| | (EA) or D → A5 or G5; ... etc ... (EA+5) or D → A0 or G0; (EA+6) → CR |
| 6 | 7 | 6 | JLR | (EA) or D → A6 or G6; ... etc ... (EA+6) or D → A0 or G0; (EA+7) → CR |
| 5 | 0 | 5 | JSR | (CR) → EA; EA+1 → CR |
| \| | 1 | \| | \| | (A0) or (G0) → EA; (CR) → EA+1; EA+2 → CR |
| \| | 2 | \| | \| | (A1) or (G1) → EA; (A0) or (G0) → EA+1; (CR) → EA+2; EA+3 → CR |
| \| | 3 | \| | \| | (A2) or (G2) → EA; ... etc ...(A0) or (G0) → EA+2; (CR) → EA+3; EA+4 →CR |
| \| | 4 | \| | \| | (A3) or (G3) → EA; ... etc ...(A0) or (G0) → EA+3; (CR) → EA+4; EA+5 →CR |
| \| | 5 | \| | \| | (A4) or (G4) → EA; ... etc ...(A0) or (G0) → EA+4; (CR) → EA+5; EA+6 →CR |
| \| | 6 | \| | \| | (A5) or (G5) → EA; ... etc ...(A0) or (G0) → EA+5; (CR) → EA+6; EA+7 →CR |
| 5 | 7 | 5 | JSR | (A6) or (G6) → EA; ... etc ...(A0) or (G0) → EA+6; (CR) → EA+7; EA+8 →CR |
| 7 | 0 | 7 | BRL | Branch to EA if LESS flag is ON |
| \| | 1 | \| | BRE | Branch to EA if EQUAL flag is ON |
| \| | 2 | \| | BRG | Branch to EA if GREATER flag is ON |
| \| | 3 | \| | BRB | Branch to EA if BIT flag is ON |
| \| | 4 | \| | BRNL | Branch to EA if LESS flag is OFF |
| \| | 5 | \| | BRNE | Branch to EA if EQUAL flag is OFF |
| \| | 6 | \| | BRNG | Branch to EA if GREATER flag is OFF |
| 7 | 7 | 7 | BRNB | Branch to EA if BIT flag is OFF |
| – | – | 8 | DSUB | Dec. sub. (EA) from (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if no borrow |
| – | – | 9 | DAD | Dec. add (EA) to (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if carry-out |
| – | – | A | BSUB | Bin. sub. (EA) from (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if no borrow |
| A | – | – | LOOP■ | Decrement (@1); branch to MD+(CR) if ((@1)) ≠ 0; no branch if ((@1)) = 0 |
| B | – | B | BAD | Bin. add (EA) to (@1) → @1; cmpr. ((@1)) to 0 for L, E, G; C if carry-out |
| C | 0 | C | BR | Branch to EA |
| \| | 1 | \| | I/OR♦ | Set hardware flags from (B6), branch to EA + (B7) |
| \| | 2 | \| | BRSF | Set (B4) per hardware flags and op. sw.; branch to EA |
| \| | 3 | \| | BRLF | Set hardware flags from flags in B4; branch to EA |
| \| | 4 | \| | BRSC | Load B5 with (CR); branch to EA |
| \| | 5 | \| | PGR | Branch to EA + (B5) |
| \| | 6 | \| | BRSB | Set (B4) per hardware flags and op. sw.; (CR) → B5, IP → OFF, branch to EA |
| C | 7 | C | PGRF | Set hardware flags from flags in B4; branch to EA + (B5) |
| D | – | D | BIT | Cmpr. 1's in (EA) to = bit pos. in (@1); set B if all checked bits in (@1) = 1's |
| E | – | E | AND* | AND (EA) bit-by-bit to (@1); cmpr. ((@1)) to 0 for L, E, G |
| F | – | F | CMPR | Binarily cmpr. (@1) to (EA); set L, E, or G if ((@1)) <, =, or > (EA) |

★ If mode is literal, D = either the EA or the data word
♦ @1 is from group A only
▲ Q = 0 if 2-word; aux. reg. group is A or B as per G
■ LOOP decrements up to 3 extra descending aux. reg. per 'L'; Relative Direct adrs. mode forms branch adrs.
♦ ETT, PFF, NXM, or MPF change if B6 status bits are not reset before an I/OR
* Store a 1-bit in ((@1)) only if both ANDed bits are 1-bits
(· ·) = the contents of or contents at "· ·" before execution
((· ·)) = the contents of or contents at "· ·" after execution
→ = indicates a value "is written into" a register or address
EA = the effective address of a data word after setup
(@1) = the contents of the aux register specified by @1 before execution
(@2) = the contents of the aux register specified by @2 before execution

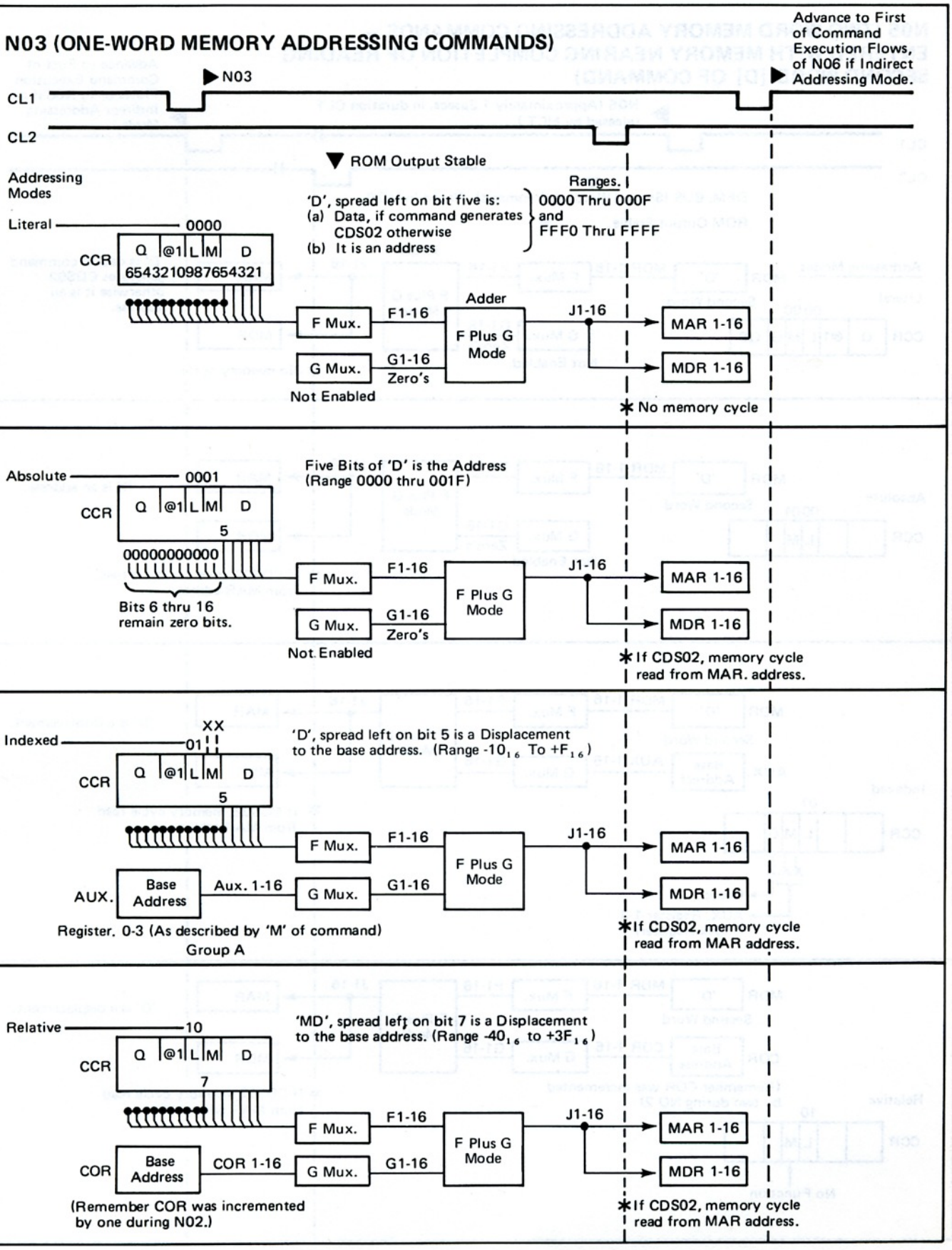
## MEMORY ADDRESSING MODES

| L* | M | MODE | EFFECTIVE ADDRESS 1-Word Command | EFFECTIVE ADDRESS 2-Word Command | 1W D VALUE (2's Comp. Form) |
|---|---|---|---|---|---|
| 0 | 0 | LITERAL▲ | Use 5 bits of D if Q = 1, 2, 5, 7, or C■; otherwise, D is the data word | Use 16 bits of D if QB = 1, 2, 5, 7, or C; otherwise, D is the data word | −10 to +F; neg. #'s have bit 5 extended |
| 0 | 1 | ABSOLUTE DIRECT | Use 5 bits of D | Use 16 bits of D | |
| 0 | 2 | ABSOLUTE INDIRECT | See INDIRECT below | See INDIRECT below | 0 to +1F, no 2's complement negative values |
| 0 | 3 | ABSOLUTE INDIR. POST INDEXED | Add (A0) to effective address for L = 0, M = 2, i.e., memory word with bit 16 OFF | Add either (A0) or (B0) as per G to effective address for L = 0, M = 2, i.e., memory word with bit 16 OFF | |
| 1 | 0 | INDEXED | Add 5 bits of D to (A0) | Add 16 bits of D to (G0) | −10 to +F, bit 5 extended, if ON, to form negative values |
| 1 | 1 | | Add 5 bits of D to (A1) | Add 16 bits of D to (G1) | |
| 1 | 2 | | Add 5 bits of D to (A2) | Add 16 bits of D to (G2) | |
| 1 | 3 | | Add 5 bits of D to (A3) | Add 16 bits of D to (G3) | |
| 2 | – | RELATIVE DIRECT | Add 7 bits of MD to (CR) | Add 16 bits of D to (CR); the 2 bits of M are not used | −40 to +3F, bit 7 extended, if ON, to form negative values |
| 3 | – | RELATIVE INDIRECT | Add 7 bits of MD to (CR) and use sum as D bits for L = 0, M = 2 | Add 16 bits of D to (CR) and use sum as D bits of L = 0, M = 2; the M bits are not used | |

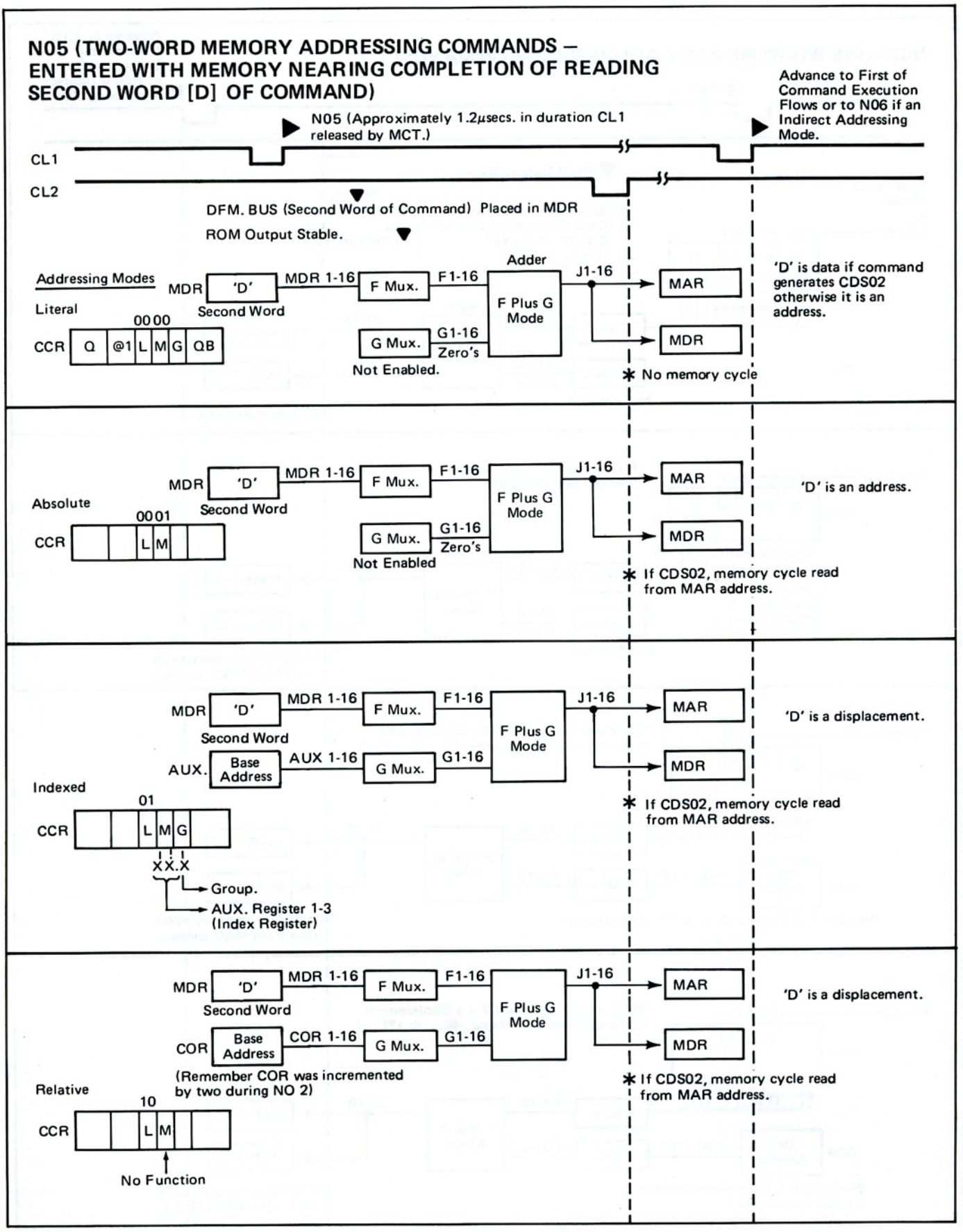
★ If Q = A, LOOP, the 2 bits of L specify up to 3 extra descending aux. registers to be decremented.
▲ If Q = 1, 2, 5, or 7, negative 1-word D values = MAE
■ When the indicated aux. reg. = 1, 5, or 7 and Q = C or Q = 0 QB = C; the effective address is the sum of D and (B7), D and (B5), or D and (B5), respectively.

INDIRECT: The D bits specify the address of a memory word. If bit 16 of this word is OFF, bits 1 – 15 = the effective address. If bit 16 is ON, bits 1 – 15 = the address of a 2nd memory word. If bit 16 of this 2nd memory word is OFF, bits 1 – 15 = the effective address. If bit 16 is ON, bits 1 – 15 = the address of a 3rd memory word, etc. If process takes more than 200 μsec., an indirect addressing error occurs.

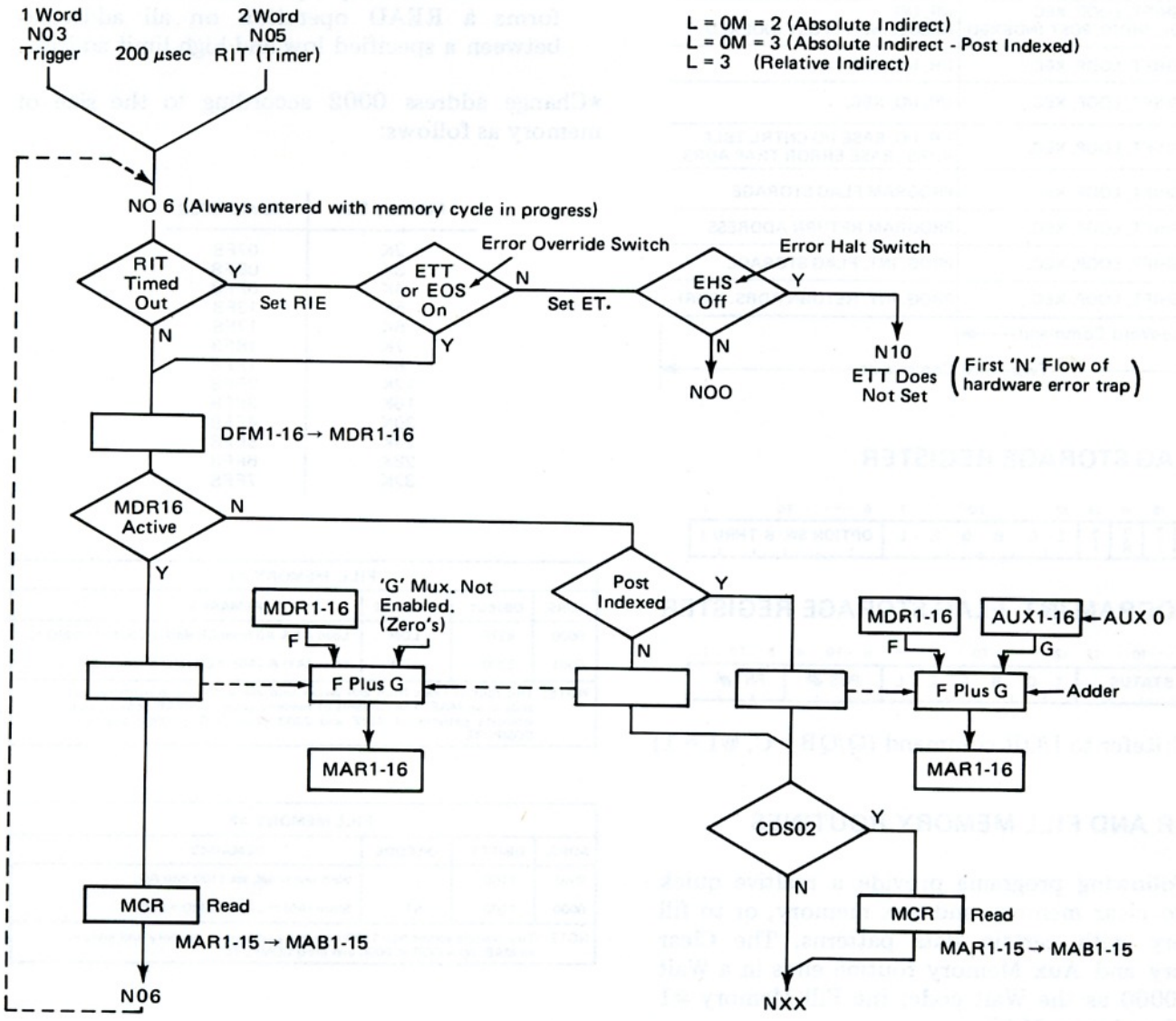
## N03 (ONE-WORD MEMORY ADDRESSING COMMANDS)



## N05 (TWO-WORD MEMORY ADDRESSING COMMANDS – ENTERED WITH MEMORY NEARING COMPLETION OF READING SECOND WORD [D] OF COMMAND)



## MEMORY ADDRESSING COMMANDS
## INDIRECT ADDRESSING MODES – N06



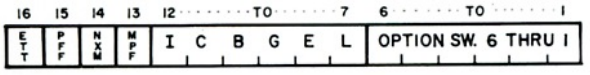
## AUXILIARY MEMORY MAP

| @ | A GROUP | B GROUP |
|---|---|---|
| 0 | I/R, SHIFT, LOOP, XEC. ABSOL. INDIR. POST INDEXED | I/R, I/O, ABSOL. INDIR. POST INDEXED |
| 1 | I/R, SHIFT, LOOP, XEC. | I/R, I/O, XEC. |
| 2 | I/R, SHIFT, LOOP, XEC. | I/R, I/O, XEC. |
| 3 | I/R, SHIFT, LOOP, XEC. | I/R, I/O, BASE I/O CNTRL TBLE. ADRS., BASE ERROR TRAP ADRS. |
| 4 | I/O, SHIFT, LOOP, XEC. | PROGRAM FLAG STORAGE |
| 5 | I/O, SHIFT, LOOP, XEC. | PROGRAM RETURN ADDRESS |
| 6 | I/O, SHIFT, LOOP, XEC. | PROG. INT. FLAG STORAGE |
| 7 | I/O, SHIFT, LOOP, XEC. | PROG. INT. RETURN ADRS. (COR) |

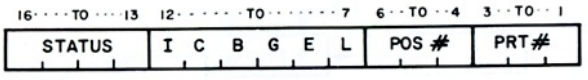
One-Word Commands

Two-Word Commands

## B4 FLAG STORAGE REGISTER




## B6 PROGRAM INT. FLAG STORAGE REGISTER




NOTE: Refer to I/OR command (Q/QB = C, @1 = 1)

## CLEAR AND FILL MEMORY ROUTINES

The following programs provide a realtive quick way to clear memory and aux memory, or to fill memory with certain data patterns. The Clear Memory and Aux Memory routine ends in a Wait with 0000 as the Wait code; the Fill Memory #1 and #2 end in an MAE.

## CLEAR MEMORY & AUX

| ADRS. | OBJECT | OP. CODE | REMARKS |
|---|---|---|---|
| 0000 | 4E00 | LDM | Load A7 – A0 with "0's" |
| 0001 | 0003 1FF8★ | LD | Load A0 with "1FF8" or per chart ★ |
| 0003 | 1287 | ST | Store 0000 (A1) → Memory at (A0)+7 |
| 0004 | A07E | LOOP | Decrement (A0) and branch to 0003 if (A0) ≠ 0; go to 0005 when (A0) = 0 |
| 0005 | 0E14 0000 | LDM | Load B7 – B0 with "0's" |
| 0007 | 2F78 | STM | Store 0000 from (A7) – (A0) → Adrs. 0007 – 0000 |

NOTE: This routine clears all of memory and aux memory, and ends in a Wait with a 0000 Wait code. New COR at 0000 and press COMPUTE.

Refer to page 14, Pub. No. 3, for an additional memory cycle routine which performs a READ operation on all addresses between a specified low and high limit address.

★Change address 0002 according to the size of memory as follows:

| Memory Size | Adress 0002 |
|---|---|
| 2K | 07F8 |
| 3K | 0BF8 |
| 4K | 0FF8 |
| 5K | 13F8 |
| 6K | 17F8 |
| 7K | 1BF8 |
| 8K | 1FF8 |
| 12K | 2FF8 |
| 16K | 3FF8 |
| 20K | 4FF8 |
| 24K | 5FF8 |
| 28K | 6FF8 |
| 32K | 7FF8 |

| FILL MEMORY #1 | | | |
|---|---|---|---|
| ADRS. | OBJECT | OP CODE | REMARKS |
| 0000 | 437F | LDM | Load A1 & A0 from CR+MD & CR+MD+1; MD = −1 |
| 0001 | 2300 | STM | Store (A1) & (A0) → CR+MD & CR+MD+1; MD = 0 |
| NOTE: This routine reads from and writes into every address in memory, and ends in an MAE. The contents of memory when the MAE occurs equals alternate patterns of 437F and 2300. New COR at 0000 and press COMPUTE. | | | |

| FILL MEMORY #2 | | | |
|---|---|---|---|
| ADRS. | OBJECT | OP CODE | REMARKS |
| @A0 | 1100 | – | With test panel, set 1100 into A0 |
| 0000 | 1100 | ST | Store (A0) → CR+MD; MD = 0000 |
| NOTE: This routine writes hex 1100 into every address in memory and ends in an MAE. New COR at 0000 and press COMPUTE. | | | |

| Hexadecimal Equivalent | $b_{24}b_{23}b_{22}b_{21}$ | $b_{20}b_{19}b_{18}b_{17}$ | $b_{16}b_{15}b_{14}b_{13}$ | $b_{12}b_{11}b_{10}b_{9}$ | $b_{8}b_{7}b_{6}b_{5}$ | $b_{4}b_{3}b_{2}b_{1}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1,048,576 | 65,536 | 4,096 | 256 | 16 | 1 |
| 2 | 2,097,152 | 131,072 | 8,192 | 512 | 32 | 2 |
| 3 | 3,145,728 | 196,608 | 12,288 | 768 | 48 | 3 |
| 4 | 4,194,304 | 262,144 | 16,384 | 1,024 | 64 | 4 |
| 5 | 5,242,880 | 327,680 | 20,480 | 1,280 | 80 | 5 |
| 6 | 6,291,456 | 393,216 | 24,576 | 1,536 | 96 | 6 |
| 7 | 7,340,032 | 458,752 | 28,672 | 1,792 | 112 | 7 |
| 8 | 8,388,608 | 524,288 | 32,768 | 2,048 | 128 | 8 |
| 9 | 9,437,184 | 589,824 | 36,864 | 2,304 | 144 | 9 |
| A | 10,485,760 | 655,360 | 40,960 | 2,560 | 160 | 10 |
| B | 11,534,336 | 720,896 | 45,056 | 2,816 | 176 | 11 |
| C | 12,582,912 | 786,432 | 49,152 | 3,072 | 192 | 12 |
| D | 13,631,488 | 851,968 | 53,248 | 3,328 | 208 | 13 |
| E | 14,680,064 | 917,504 | 57,344 | 3,584 | 224 | 14 |
| F | 15,728,640 | 983,040 | 61,440 | 3,840 | 240 | 15 |

# NCR 605 PROCESSOR
## CONTENTS



ST-9167-06

# NCR 605 PROCESSOR

## INTRODUCTION

This section contains the routines, charts, and timing diagrams associated with the 605 processor, and timing diagrams for the memory interface.

| BINARY WORD VALUES | DECIMAL WORD VALUES |
|---|---|
| 0111111111111111 = +32,767 | 0111 1001 1001 1001 = +7,999 |
| 0111111111111110 = +32,766 | 0111 1001 1001 1000 = +7,998 |
| 0111111111111101 = +32,765 | 0111 1001 1001 0111 = +7,997 |
| 0000001111101000 = + 1,000 | 0001 0000 0000 0000 = +1,000 |
| 0000000000000010 = + 2 | 0000 0000 0000 0010 = + 2 |
| 0000000000000001 = + 1 | 0000 0000 0000 0001 = + 1 |
| 0000000000000000 = 0 | 0000 0000 0000 0000 = 0 |
| 1111111111111111 = − 1 | 1001 1001 1001 1001 = − 1 |
| 1111111111111110 = − 2 | 1001 1001 1001 1000 = − 2 |
| 1111110000011000 = − 1,000 | 1001 0000 0000 0000 = −1,000 |
| 1000000000000010 = −32,766 | 1000 0000 0000 0010 = −1,998 |
| 1000000000000001 = −32,767 | 1000 0000 0000 0001 = −1,999 |
| 1000000000000000 = −32,768 | 1000 0000 0000 0000 = −2,000 |

## BINARY FIELD

0000000000000110/0001101010000001 = +400,001
(0006/1A81)
0000000000000000/1000000000000001 = + 32,769
(0000/8001)
1111111111111111/0111111111111111 = − 32,769
(FFFF/7FFF)
1111111111111001/1110010101111111 = −400,001
(FFF9/E57F)

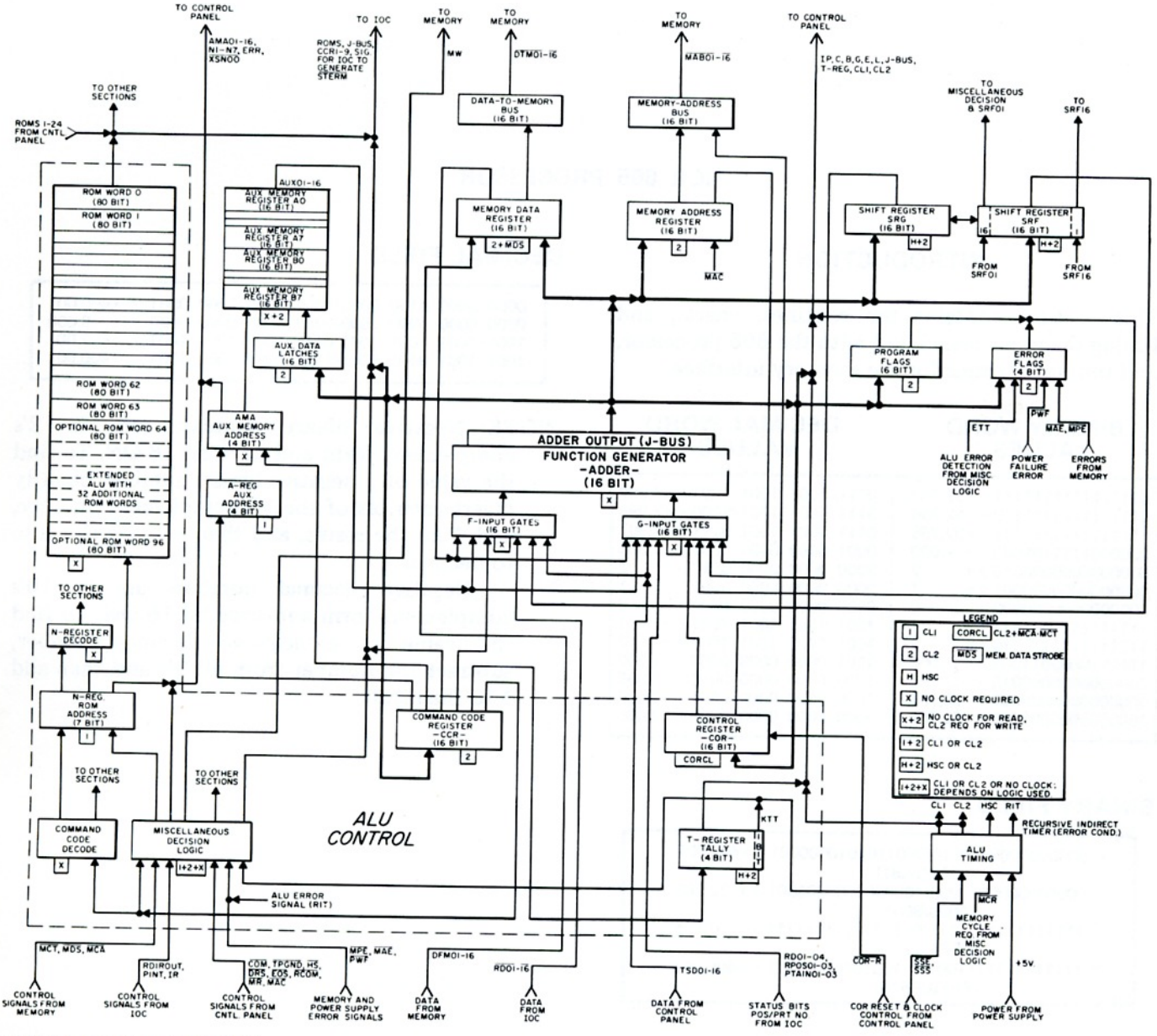
## DECIMAL FIELD

0000 0000 0100 0000/0000 0000 0000 0001 = +400,001
0000 0000 0000 0000/1000 0000 0000 0000 = + 8,000
1001 1001 1001 1001/0111 1001 1001 1001 = − 2,001
1001 1001 0101 1001/1001 1001 1001 1001 = −400,001

NOTE: Negative **binary** numbers are in 2's complement form and have bit 16 ON. To find the value of a negative binary number, binarily invert each bit of the 2's complement number, add 1 to the result, and then convert the sum to decimal.

Negative **decimal** numbers are in 10's complement form and have bit 16 ON. To find the value of a negative decimal number, subtract the number from all 9's and then add 1 to the result.

ST-9167-06




**ALU BLOCK DIAGRAM**

## CDS/CDB DECODE FOR ROM 31

| COMMAND TYPE | IN BLK N-- | INPUTS ✳ | INPUTS Q | ACTION | OUTPUTS CDS 02 | OUTPUTS CDS 01 | OUTPUTS CDB 06 | OUTPUTS CDB 05 | OUTPUTS CDB 04 | OUTPUTS CDB 03 | OUTPUTS CDB 02 | OUTPUTS CDB 01 | EXIT TO N-- |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TWO-WORD | N2 | $\overline{A4}$ | $\overline{A3}$ $\overline{A2}$ $\overline{A1}$ $\overline{A0}$ | GO TO N05 | L | H | L | L | L | H | L | H | N5 |
| ONE-WORD | N2 or N3 or N6 | $\overline{A4}$ | $\overline{A3}$ $\overline{A2}$ $\overline{A1}$ $\overline{A0}$ | ST | L | L | L | H | L | L | L | H | N21 |
| | | $\overline{A4}$ | $\overline{A3}$ $\overline{A2}$ $\overline{A1}$ A0 | STM | L | L | H | L | H | L | L | L | N50 |
| | | $\overline{A4}$ | $\overline{A3}$ $\overline{A2}$ A1 $\overline{A0}$ | LD | H | L | L | H | L | L | H | L | N22 |
| | | $\overline{A4}$ | $\overline{A3}$ $\overline{A2}$ A1 A0 | LDM | H | L | H | H | L | L | H | L | N62 |
| | | $\overline{A4}$ | $\overline{A3}$ A2 $\overline{A1}$ $\overline{A0}$ | JSR | L | L | H | L | L | H | H | H | N47 |
| | | $\overline{A4}$ | $\overline{A3}$ A2 $\overline{A1}$ A0 | JLR | H | L | H | L | H | L | L | H | N51 |
| | | $\overline{A4}$ | $\overline{A3}$ A2 A1 $\overline{A0}$ | BR (Con) | L | L | L | H | L | H | H | L | N26 |
| | | $\overline{A4}$ | A3 $\overline{A2}$ $\overline{A1}$ $\overline{A0}$ | GO TO N07 | L | H | L | L | L | H | H | H | N07 |
| | N2 | $\overline{A4}$ | A3 $\overline{A2}$ $\overline{A1}$ A0 | I O | L | H | H | H | H | H | H | L | N76 |
| | | $\overline{A4}$ | A3 $\overline{A2}$ A1 $\overline{A0}$ | LOOP | L | H | H | H | L | L | L | L | N60 |
| | N2 N3 or N6 | $\overline{A4}$ | A3 A2 $\overline{A1}$ $\overline{A0}$ | BAD | H | L | L | L | H | L | H | H | N11 |
| | | $\overline{A4}$ | A3 A2 $\overline{A1}$ A0 | BR (Sn) | L | L | L | H | L | H | H | H | N27 |
| | | $\overline{A4}$ | A3 A2 A1 $\overline{A0}$ | BIT | H | L | L | H | L | H | L | L | N24 |
| | | $\overline{A4}$ | A3 A2 A1 A0 | AND | H | L | L | L | H | L | H | H | N13 |
| | | $\overline{A4}$ | A3 A2 A1 A0 | CMPR | H | L | L | L | H | H | H | L | N16 |
| TWO-WORD | N5 or N6 | ✳ | QB | | | | | | | | | | |
| | | A4 | $\overline{A3}$ $\overline{A2}$ $\overline{A1}$ $\overline{A0}$ | WAITI | L | L | L | L | L | L | L | L | N00 |
| | | A4 | $\overline{A3}$ $\overline{A2}$ $\overline{A1}$ A0 | ST | L | L | L | H | L | L | L | H | N21 |
| | | A4 | $\overline{A3}$ $\overline{A2}$ A1 $\overline{A0}$ | STM | L | L | H | L | H | L | L | L | N50 |
| | | A4 | $\overline{A3}$ $\overline{A2}$ A1 A0 | LD | H | L | L | H | L | L | H | L | N22 |
| | | A4 | $\overline{A3}$ A2 $\overline{A1}$ $\overline{A0}$ | LDM | H | L | H | H | L | L | H | L | N62 |
| | | A4 | $\overline{A3}$ A2 $\overline{A1}$ A0 | JSR | L | L | H | L | L | H | H | H | N47 |
| | | A4 | $\overline{A3}$ A2 A1 $\overline{A0}$ | JLR | H | L | H | L | H | L | L | H | N51 |
| | | A4 | $\overline{A3}$ A2 A1 A0 | BR (Con) | L | L | L | H | L | H | H | L | N26 |
| | | A4 | A3 $\overline{A2}$ $\overline{A1}$ $\overline{A0}$ | DSUB | H | L | L | H | H | L | L | H | N35 |
| | | A4 | A3 $\overline{A2}$ $\overline{A1}$ A0 | DAD | H | L | L | H | H | H | H | L | N36 |
| | | A4 | A3 $\overline{A2}$ A1 $\overline{A0}$ | BSUB • | H | L | L | L | H | L | L | H | N12 |
| | | A4 | A3 $\overline{A2}$ A1 A0 | BAD | H | L | L | L | H | L | L | H | N11 |
| | | A4 | A3 A2 $\overline{A1}$ $\overline{A0}$ | BR (Sn) | L | L | L | H | L | H | H | H | N27 |
| | | A4 | A3 A2 $\overline{A1}$ A0 | BIT | H | L | L | H | L | H | L | L | N24 |
| | | A4 | A3 A2 A1 $\overline{A0}$ | AND | H | L | L | L | H | L | H | H | N13 |
| | | A4 | A3 A2 A1 A0 | CMPR | H | L | L | L | H | H | H | L | N16 |

L = Low Voltage Level
H = High Voltage Level
• A4 = Q of 0 = $\overline{N02}$

Gates CDB01 – 06 at N1 – N6 if High
Causes Mem. Cycle During Set-Up if High = $\overline{MDR16}$

## CDS/CDB DECODE FOR ROM 32

| IN N7 AND | INPUTS N | INPUTS QB | ACTION | OUTPUTS CDS 02 | OUTPUTS CDS 01 | OUTPUTS CDB 06 | OUTPUTS CDB 05 | OUTPUTS CDB 04 | OUTPUTS CDB 03 | OUTPUTS CDB 02 | OUTPUTS CDB 01 | EXIT TO N-- |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REGISTER TO REGISTER | $\overline{A4}$ | $\overline{A3}$ $\overline{A2}$ $\overline{A1}$ $\overline{A0}$ | RFSD | L | H | L | H | H | H | L | H | N35 |
| | $\overline{A4}$ | $\overline{A3}$ $\overline{A2}$ $\overline{A1}$ A0 | RFAD | L | H | L | H | H | H | H | L | N36 |
| | $\overline{A4}$ | $\overline{A3}$ $\overline{A2}$ A1 $\overline{A0}$ | RFSB | L | H | L | L | H | L | H | L | N12 |
| | $\overline{A4}$ | $\overline{A3}$ $\overline{A2}$ A1 A0 | RFAB | L | H | L | L | H | L | L | H | N11 |
| | $\overline{A4}$ | $\overline{A3}$ A2 $\overline{A1}$ $\overline{A0}$ | REXC | L | H | H | L | H | H | L | H | N55 |
| | $\overline{A4}$ | $\overline{A3}$ A2 $\overline{A1}$ A0 | ROR | L | H | L | L | H | H | L | L | N14 |
| | $\overline{A4}$ | $\overline{A3}$ A2 A1 $\overline{A0}$ | RXOR | L | H | L | L | H | H | L | H | N15 |
| | $\overline{A4}$ | $\overline{A3}$ A2 A1 A0 | RMVC | L | H | L | H | L | L | L | L | N20 |
| | $\overline{A4}$ | A3 $\overline{A2}$ $\overline{A1}$ $\overline{A0}$ | RDSUB | L | H | L | H | H | H | L | H | N35 |
| | $\overline{A4}$ | A3 $\overline{A2}$ $\overline{A1}$ A0 | RDAD | L | H | L | H | H | H | H | L | N36 |
| | $\overline{A4}$ | A3 $\overline{A2}$ A1 $\overline{A0}$ | RBSUB | L | H | L | L | H | L | H | L | N12 |
| | $\overline{A4}$ | A3 $\overline{A2}$ A1 A0 | RBAD | L | H | L | L | H | L | L | H | N11 |
| | $\overline{A4}$ | A3 A2 $\overline{A1}$ $\overline{A0}$ | RMV | L | H | L | L | H | H | H | H | N17 |
| | $\overline{A4}$ | A3 A2 $\overline{A1}$ A0 | RBIT | L | H | L | H | L | H | L | L | N24 |
| | $\overline{A4}$ | A3 A2 A1 $\overline{A0}$ | RAND | L | H | L | H | L | L | H | H | N13 |
| | $\overline{A4}$ | A3 A2 A1 A0 | RCMPR | L | H | L | L | H | H | H | L | N16 |
| SHIFT | A4 | $\overline{A3}$ $\overline{A2}$ $\overline{A1}$ $\overline{A0}$ | SLD | L | H | H | L | L | L | H | H | N42 |
| | A4 | $\overline{A3}$ $\overline{A2}$ $\overline{A1}$ A0 | SLC | L | H | H | L | L | L | H | H | N43 |
| | A4 | $\overline{A3}$ $\overline{A2}$ A1 $\overline{A0}$ | SLO | L | H | H | L | L | L | H | H | N43 |
| | A4 | $\overline{A3}$ $\overline{A2}$ A1 A0 | SLX | L | H | H | L | L | L | H | H | N43 |
| | A4 | $\overline{A3}$ A2 $\overline{A1}$ $\overline{A0}$ | SRD | L | H | H | L | L | L | H | H | N42 |
| | A4 | $\overline{A3}$ A2 $\overline{A1}$ A0 | SRC | L | H | H | L | L | L | H | H | N43 |
| | A4 | $\overline{A3}$ A2 A1 $\overline{A0}$ | SRO | L | H | H | L | L | L | H | H | N43 |
| | A4 | $\overline{A3}$ A2 A1 A0 | SRA | L | H | H | L | L | L | H | H | N43 |
| EXECUTE | A4 | A3 $\overline{A2}$ $\overline{A1}$ $\overline{A0}$ | | L | H | H | H | H | H | H | H | N77 |
| | A4 | A3 $\overline{A2}$ $\overline{A1}$ A0 | ERROR | L | H | H | H | H | H | H | H | N77 |
| | A4 | A3 $\overline{A2}$ A1 $\overline{A0}$ | TRAP IF | L | H | H | H | H | H | H | H | N77 |
| | A4 | A3 $\overline{A2}$ A1 A0 | EXECUTE | L | H | H | H | H | H | H | H | N77 |
| | A4 | A3 A2 $\overline{A1}$ $\overline{A0}$ | OPTION | L | H | H | H | H | H | H | H | N77 |
| | A4 | A3 A2 $\overline{A1}$ A0 | IS NOT | L | H | H | H | H | H | H | H | N77 |
| | A4 | A3 A2 A1 $\overline{A0}$ | IN 605 | L | H | H | H | H | H | H | H | N77 |
| | A4 | A3 A2 A1 A0 | | L | H | H | H | H | H | H | H | N77 |

L = Low Voltage Level
H = High Voltage Level

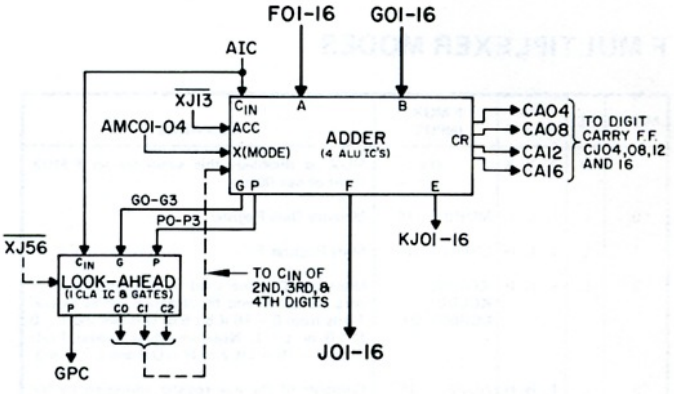
Gates CDB01 – 06 at N1 – N6 if High
Causes Mem. Cycle During Set-Up if High = $\overline{MDR16}$

## FUNCTION GENERATOR MODES

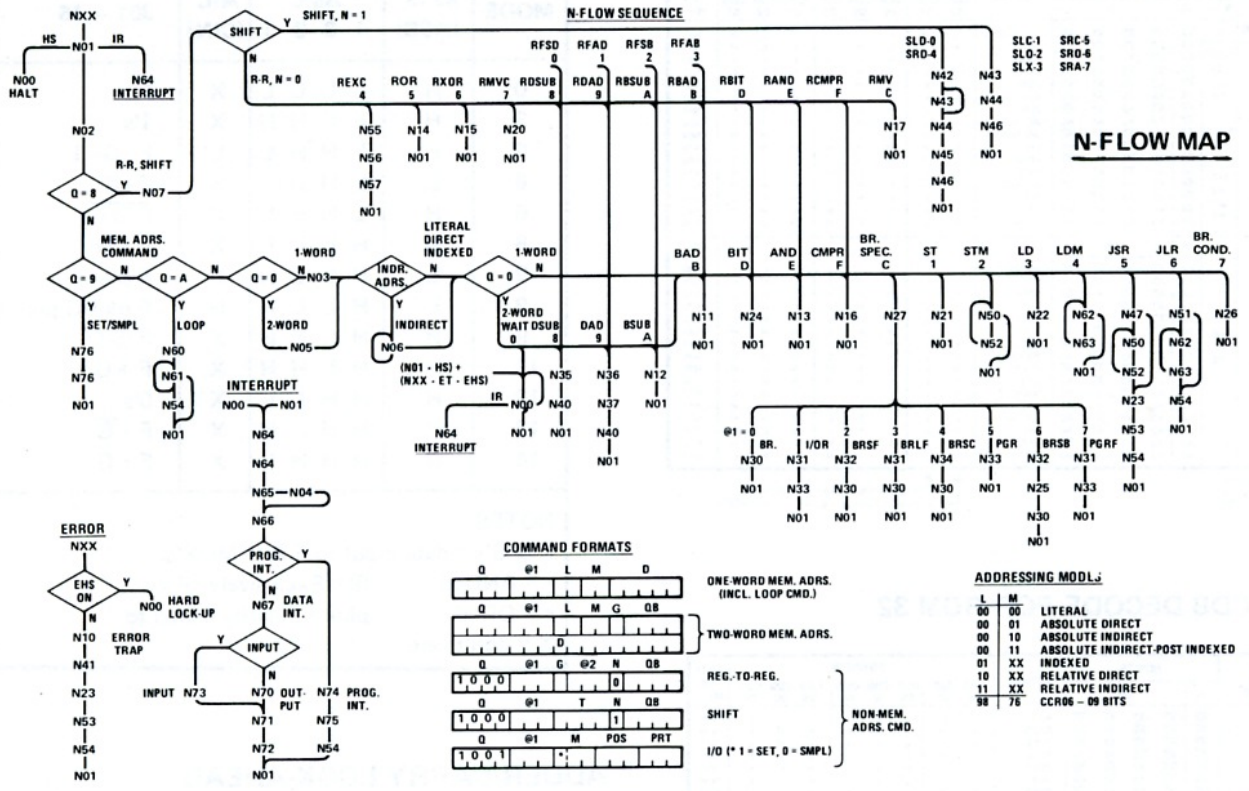
| MODE | $\overline{XJ13}$ (ACC) | AMC 4 3 2 1 | AIC (CIN) | J01 – 16 |
|---|---|---|---|---|
| 0 | H | L L L L | X | $\overline{F}$ |
| 3 | H | L L H H | X | 1's |
| 6 | L | L H H L | L | F–G–1 |
| 6 | L | L H H L | H | F–G |
| 6 | H | L H H L | X | $\overline{F \oplus G}$ |
| 8 | H | H L L L | X | $\overline{F} \cdot G$ |
| 9 | L | H L L H | L | F plus G |
| 9 | L | H L L H | H | F plus G plus 1 |
| 9 | H | H L L H | X | $F \oplus G$ |
| 11 | H | H L H H | X | F + G |
| 12 | H | H H L L | X | 0's |
| 13 | H | H H L H | X | $F \cdot \overline{G}$ |
| 14 | H | H H H L | X | F • G |

NOTES:
F & G's = data input to F & G MUX's
• = ANDed  $\oplus$ = Exclusively-ORed
+ = ORed  plus = binarily added to
X = don't care

## ADDER/CARRY LOOK-AHEAD



## N-FLOW MAP




## F MULTIPLEXER MODES

| MODE | FE | FA 3 2 1 | F-MUX INPUT | REMARKS |
|---|---|---|---|---|
| – | H | X X X | 0's | MUX is disabled; this simulates an F-MUX input of hex 0000. |
| 0 | L | L L L | MDR01 – 16 | Memory Data Register. |
| 1 | L | L L H | SRF01 – 16 | Shift Register F. |
| 2 | L | L H L | KCCR02, KCCR01, CCR05 – 01 | One-word D value used as a displacement or adrs. as per L and M. Negative values extend 1-bits from 6 – 16 if bit 5 of D is ON and L = 0 M = 0 or L = 1. Negative values extend 1-bits from 8 – 16 if bit 7 of M is ON and L = 2 or 3. |
| 3 | L | L H H | AUX01 – 16 | Contents of the aux register addressed by the AMA lines. |
| 4 | L | H L L | GND, FLOAT, XS-N04 | Input is hex 6666 for first dec. add step (N36), or hex FFFE (–2) during interrupt flow so two can be subtracted from COR if interrupt occurred during a Wait. |
| 5 | L | H L H | $\overline{\text{RD01}}$ – $\overline{16}$ | Receive Data input lines. |
| 6 | L | H H L | AUX16 – 08, RPOS03 – 01, PTAIN03 – 01, XS-N41 | The value required to compute the following:<br>• The adrs. of the NC-TA pair for ADT.<br>• The adrs. of TA for prog. int.<br>• The error trap adrs. |
| 7 | L | H H H | TSD01 – 16 | The state of the DATA ENTER switches on the test panel (DSD01 – 16), or the contents of the micro-diagnostic CRA register (CRA01 – 16). |

## G MULTIPLEXER MODES

| MODE | GE | GA 3 2 1 | G-MUX INPUT | REMARKS |
|---|---|---|---|---|
| – | H | X X X | 0's | MUX is disabled; this simulates a G-MUX input of hex 0000. |
| 0 | L | L L L | MAR01 – 16 | Memory Address Register |
| 1 | L | L L H | SRG01 – 16 | Shift Register G. |
| 2 | L | L H L | AUX01 – 16 | Contents of the aux register addressed by the AMA lines. |
| 3 | L | L H H | COR01 – 16 | Control Register. |
| 4 | L | H L L | GND, $\overline{\text{CJ04}}$, $\overline{\text{CJ08}}$, $\overline{\text{CJ12}}$, $\overline{\text{CJ16}}$ | Input is four hex digits when entering N40; each digit is either hex A or 0 depending on the digit carries in N35 or N37. A digit is hex A if $\overline{\text{CJ04}}$, $\overline{08}$, $\overline{12}$, or $\overline{16}$ is reset (the adder operation did not result in a decimal digit, since a carry-out did not occur). A digit is hex 0 if $\overline{\text{CJ04}}$, $\overline{08}$, $\overline{12}$, or $\overline{16}$ is set. |
| 5 | L | H L H | RD04 – 01, I, C, B, G, E, L, RPOS03 – 01, PTAIN03 – 01 | Interrupt status lines (16 – 13), flags (12 – 7), position number (6 – 4), and port number (3 – 1) stored in B6 during program interrupt. |
| 6 | L | H H L | MDR01 – 16 | Memory Data Register. |
| 7 | L | H H H | ETT, PFF, NXM, MPF, I, C, B, G, E, L, TSD06 – 01 | Error flags (16 – 13), program flags (12 – 7), and option switch values (6 – 1) stored in B4 during BRSF and BRSB commands. |

## 605 INTERRUPT FLOW

| BLOCK | REMARKS | IF | GO TO |
|---|---|---|---|
| NXX | Receive int. signal(s) from port(s): data int. = $\overline{\text{RDINT\#}}$ and RPINT#; prog. int. = $\overline{\text{RPINT\#}}$. Port 0 has highest priority. | $\overline{\text{N00}}$ & $\overline{\text{N01}}$ | N00 or N01 at end of CMD or INT |
| N00<br>N01 | Set KA to stay in N64 twice. If N00, set IRH. Term PINT goes High if IP flag is set and prog. int. | – | N64 |
| N64 | Output $\overline{\text{SSEL\#}}$ to port with priority. Receive POS#, PRT#, data direction RDIROUT. Also, receive data on $\overline{\text{RD01}}$ – $\overline{\text{T6}}$ if input or status bits on $\overline{\text{RD01}}$ – $\overline{\text{04}}$ if prog. int. At CL2, NC → MAR and SRF; reset KA at ending CL1. | KA | N64 |
| N64 (2nd) | Read NC from memory → MDR. | $\overline{\text{KA}}$ | N65 |
| N65 | Transfer NC (MDR) → SRG. If NC is FFFF, i.e., −1, set KB with GPC at CL1 to output terminate, $\overline{\text{STERM}}$, in N70 or N73. | IRH<br>$\overline{\text{IRH}}$ | N04<br>N66 |
| N04 | Subtract 2 from (COR) → COR; actually −2 or FFFE is added to (COR). | – | N66 |
| N66 | Add 1 to adrs. of NC (MAR) to form adrs. of TA → MAR. Hold KB if KB; read TA from memory → MDR. | $\overline{\text{PINT}}$<br>PINT | N67<br>N74 |
| | **DATA INTERRUPT** | | |
| N67 | Add NC (SRG) to TA (MDR) → MAR to form pick-up or put-away adrs. If carry-out, $\overline{\text{CA16}}$ → Low, NC is negative; if no carry-out, $\overline{\text{CA16}}$ is High, NC is 0 or a pos. #. Hold KB if KB; set KB at CL2 if $\overline{\text{CA16}}$ is High, to output $\overline{\text{STERM}}$ in N70 or N73. If RDIROUT is High, read output word from memory → MDR. If $\overline{\text{CA16}}$ is High, set KA at CL1 to prevent a memory write cycle in N73. | RDIROUT<br>$\overline{\text{RDIROUT}}$ | N70<br>N73 |
| N70 | Force $\overline{\text{STERM}}$ Low if KB; send output word (MDR) to port on $\overline{\text{SD01}}$ – $\overline{\text{T6}}$. Hold KB at CL2 if KB, to keep $\overline{\text{STERM}}$ Low; output strobe, $\overline{\text{SSTB}}$, and reset KA at CL1. Port removes signals to 605. | – | N71 |
| N71 | Force $\overline{\text{STERM}}$ High and remove output data from $\overline{\text{SD01}}$ – $\overline{\text{T6}}$ lines. Add 1 to NC (SRG) → MDR; reset KB at CL2. | – | N72 |
| N72 | Transfer adrs. of NC (SRF) → MAR; write updated NC into memory. | – | N01 |
| N73 | Force $\overline{\text{STERM}}$ Low if KB; transfer input word from $\overline{\text{RD01}}$ – $\overline{\text{T6}}$ to MDR. Hold KB at CL2 if KB, to keep $\overline{\text{STERM}}$ Low; write input word (MDR) to put-away adrs. if KA is Low. Output strobe, $\overline{\text{SSTB}}$, and reset KA at CL1; port removes signals to 605. | – | N71 |
| | **PROGRAM INTERRUPT** | | |
| N74 | Transfer status bits, $\overline{\text{RD01}}$ – $\overline{\text{04}}$, program flags, ICBGEL, and position# / port#, RPOS03 – 01 / RPAIN03 – 01, to B6 at CL2; reset KB at CL2. Output strobe, $\overline{\text{SSTB}}$, at CL1; port removes signals to 605. | – | N75 |
| N75 | Transfer (COR) → B7; reset IP flag. | – | N54 |
| N54 | Transfer TA word (MDR) → COR to trap to prog. int. routine. | – | N01 |
| | **AT END OF INTERRUPT** | | |
| N01 | Reset IRH if set; force $\overline{\text{SSEL\#}}$ High; reset PINT. | INT<br>$\overline{\text{INT}}$ | N64<br>N02<br>etc. |
| NOTE | Refer below to calculate adrs. of prog. int. routine. | | |

## CALCULATING PROGRAM INT. ADDRESS

M.S. 9 bits of B3/POS#/PRT#/1 = TA address
(TA address) = Prog. Int. Address
(B7) = COR

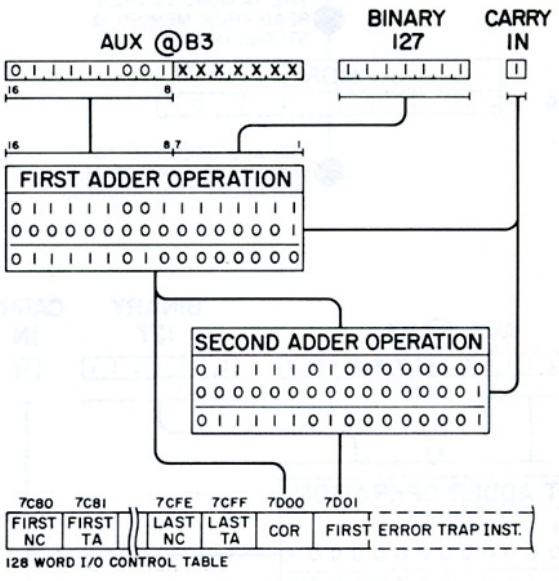
## 605 ERROR TRAP

| BLOCK | REMARKS | IF | GO TO |
|---|---|---|---|
| NXX | ET latch sets, if ERROR OVERRIDE switch is not ON, as soon as an IAE, PWF, MAE, or MPE error is detected. Enable bit 4 of N-reg. Inhibit KN01 – 03 logic, data or prog. int., aux. reg. write, loading of MAR, and setting of ETT if IAE. Output ERR, output ERR02 if IAE or MAE. | $\overline{\text{EHS}}$<br>EHS | N10<br>N00 |
| N10 | Reset ET latch, set ETT if not IAE, inhibit recognition of any other errors by ET latch, transfer (MAR) → SRF to save error adrs. | – | 41 |
| N41 | Form COR storage adrs. by adding 1 to maximum TANC table adrs. → MAR. MAX. TANC adrs. = bits 16 – 8 of B3 and bits 7 – 1 at all 1-bits. | – | N23 |
| N23 | Transfer (COR) → MDR. Write present COR (MDR) into memory at first adrs. above MAX TANC adrs. (MAR). | – | N53 |
| N53 | Add 1 to (MAR) → MDR to form error trap adrs. | – | N54 |
| N54 | Transfer error trap adrs. (MDR) → COR. | – | N01 |
| NOTE | Refer below to calculate error trap adrs. | | |

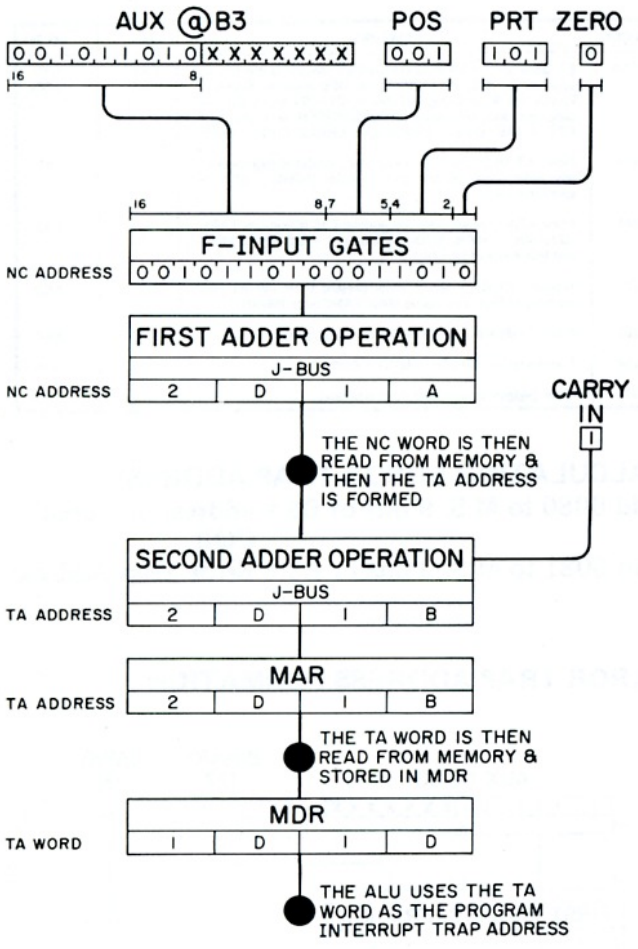
## CALCULATING ERROR TRAP ADDRESS

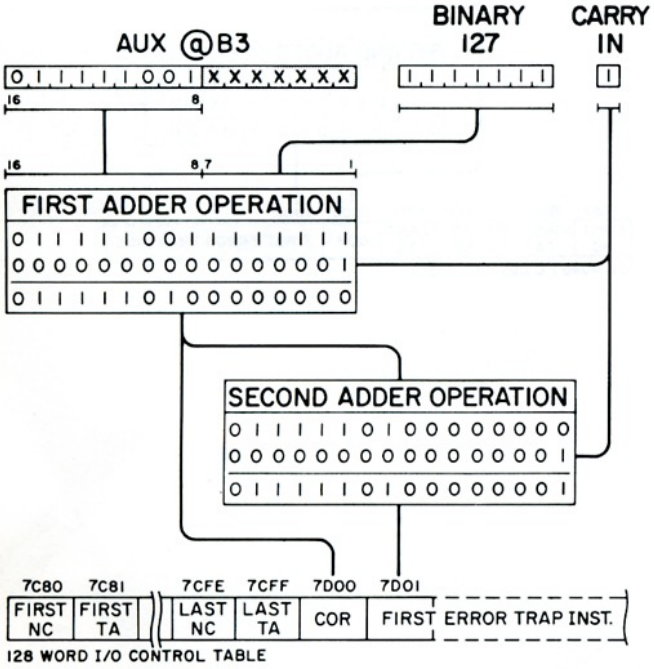
Add 0080 to M.S. 9 bits of B3 = address of stored COR
Add 0081 to M.S. 9 bits of B3 = Error Trap Address

## ERROR TRAP ADDRESS FORMATION



## PROGRAM INT. ADDRESS FORMATION




## 128 WORD I/O CONTROL TABLE

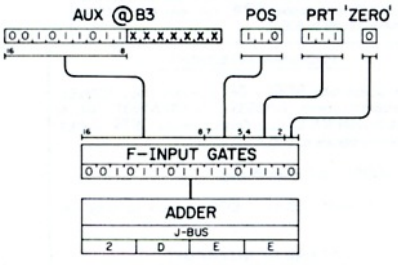
| POS | PRT | POS | PRT | POS | PRT | POS | PRT | POS | PRT | POS | PRT | POS | PRT | POS | PRT | POS | PRT | POS | PRT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 7 | 1 | 0 | 1 | 7 | 2 | 0 | 2 | 1 | 2 | 2 | 7 | 6 | 7 | 7 |
| NC | TA | NC | TA | NC | TA | NC | TA | NC | TA | NC | TA | NC | TA | NC | TA | NC | TA | NC | TA |

## TANC TABLE ADDRESSES, TWO L.S. HEX DIGITS

| POS | PRT | @B3-b8 0 | @B3-b8 1 | POS | PRT | @B3-b8 0 | @B3-b8 1 | POS | PRT | @B3-b8 0 | @B3-b8 1 | POS | PRT | @B3-b8 0 | @B3-b8 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 00 | 80 | 2 | 0 | 20 | A0 | 4 | 0 | 40 | C0 | 6 | 0 | 60 | E0 |
| 0 | 1 | 02 | 82 | 2 | 1 | 22 | A2 | 4 | 1 | 42 | C2 | 6 | 1 | 62 | E2 |
| 0 | 2 | 04 | 84 | 2 | 2 | 24 | A4 | 4 | 2 | 44 | C4 | 6 | 2 | 64 | E4 |
| 0 | 3 | 06 | 86 | 2 | 3 | 26 | A6 | 4 | 3 | 46 | C6 | 6 | 3 | 66 | E6 |
| 0 | 4 | 08 | 88 | 2 | 4 | 28 | A8 | 4 | 4 | 48 | C8 | 6 | 4 | 68 | E8 |
| 0 | 5 | 0A | 8A | 2 | 5 | 2A | AA | 4 | 5 | 4A | CA | 6 | 5 | 6A | EA |
| 0 | 6 | 0C | 8C | 2 | 6 | 2C | AC | 4 | 6 | 4C | CC | 6 | 6 | 6C | EC |
| 0 | 7 | 0E | 8E | 2 | 7 | 2E | AE | 4 | 7 | 4E | CE | 6 | 7 | 6E | EE |
| 1 | 0 | 10 | 90 | 3 | 0 | 30 | B0 | 5 | 0 | 50 | D0 | 7 | 0 | 70 | F0 |
| 1 | 1 | 12 | 92 | 3 | 1 | 32 | B2 | 5 | 1 | 52 | D2 | 7 | 1 | 72 | F2 |
| 1 | 2 | 14 | 94 | 3 | 2 | 34 | B4 | 5 | 2 | 54 | D4 | 7 | 2 | 74 | F4 |
| 1 | 3 | 16 | 96 | 3 | 3 | 36 | B6 | 5 | 3 | 56 | D6 | 7 | 3 | 76 | F6 |
| 1 | 4 | 18 | 98 | 3 | 4 | 38 | B8 | 5 | 4 | 58 | D8 | 7 | 4 | 78 | F8 |
| 1 | 5 | 1A | 9A | 3 | 5 | 3A | BA | 5 | 5 | 5A | DA | 7 | 5 | 7A | FA |
| 1 | 6 | 1C | 9C | 3 | 6 | 3C | BC | 5 | 6 | 5C | DC | 7 | 6 | 7C | FC |
| 1 | 7 | 1E | 9E | 3 | 7 | 3E | BE | 5 | 7 | 5E | DE | 7 | 7 | 7E | FE |

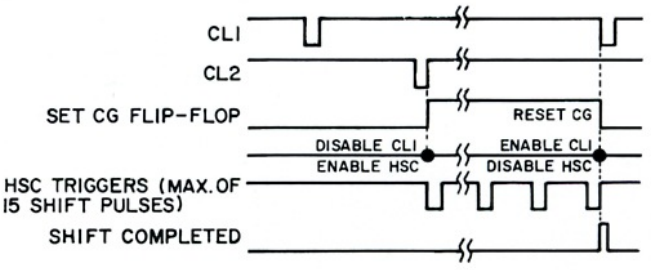
| AUX @B3 | POS# | PRT# | '0' |
|---|---|---|---|
| @ @ @ @ | @ @ @ @ | @ S S S | T T T 0 |

## TANC TABLE ADDRESS FORMATION

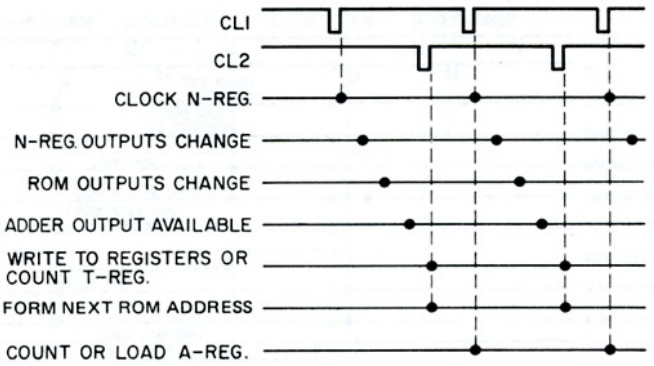



NOTE: Transfer data if NC < 0, or if NC plus 1 = 0. Terminate if NC plus 1 = 0, or if NC > 0. PORT 0 has highest priority.

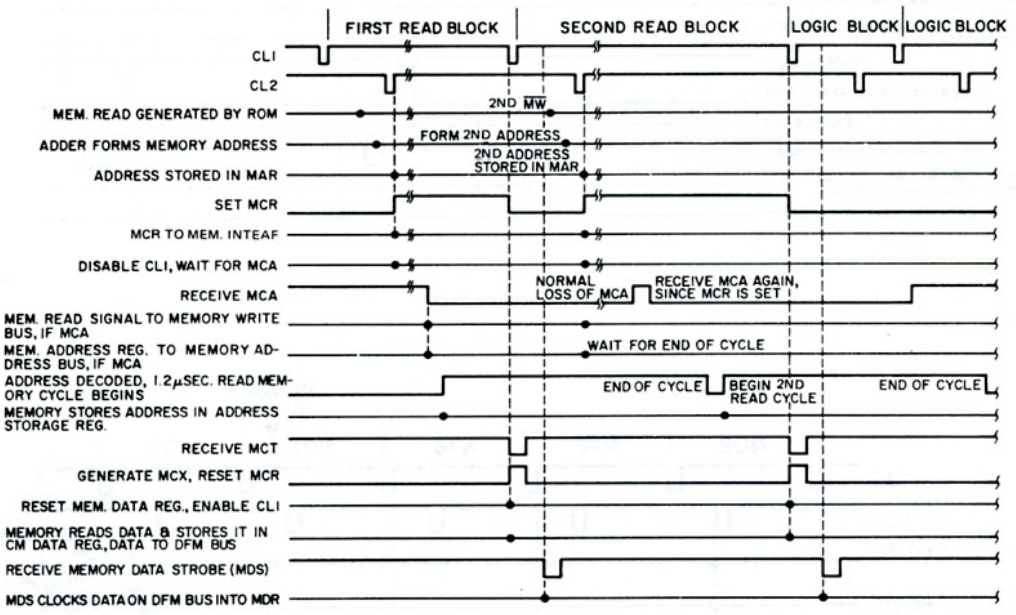
## SHIFT BLOCK TIMING



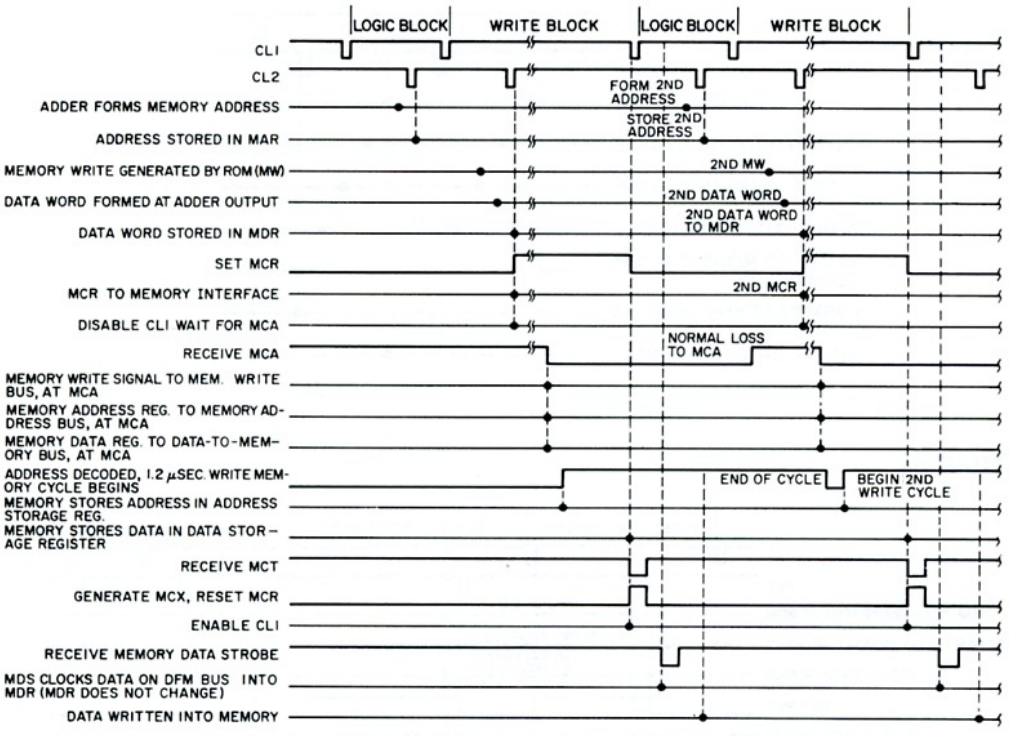
## LOGIC BLOCK TIMING



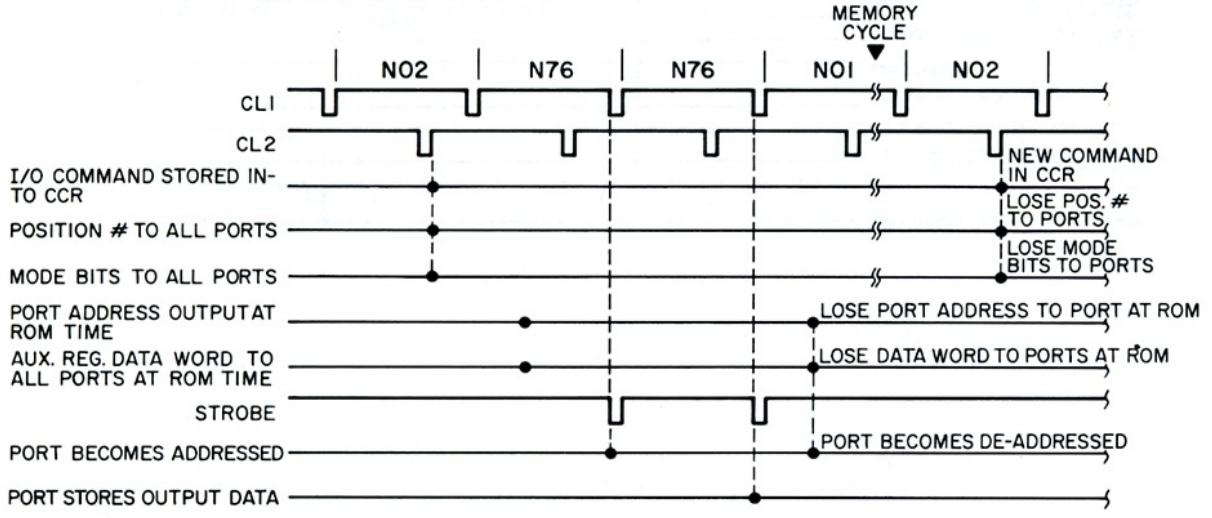

## READ BLOCK TIMING



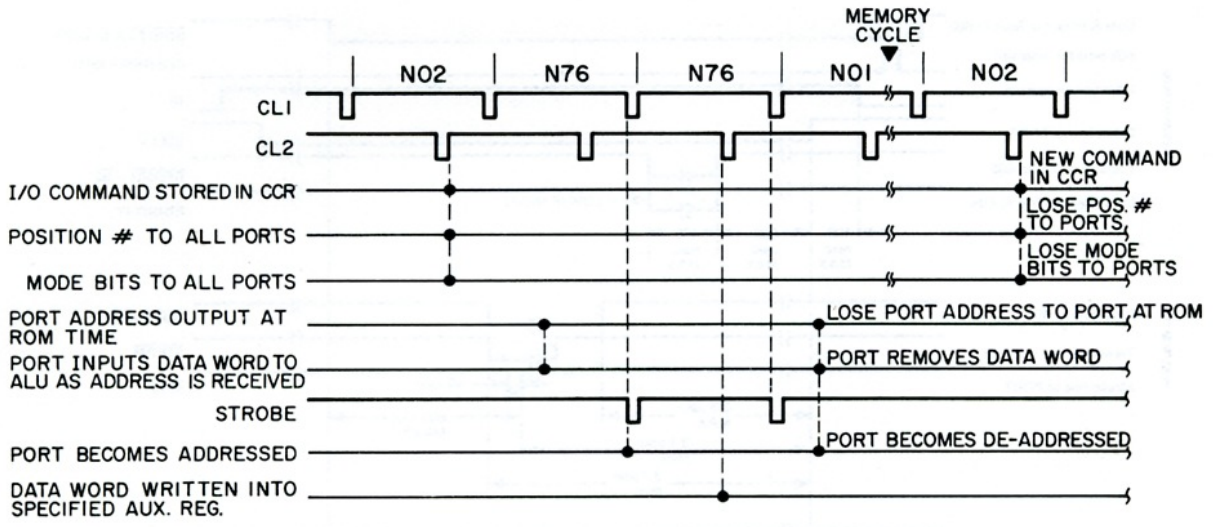
## WRITE BLOCK TIMING



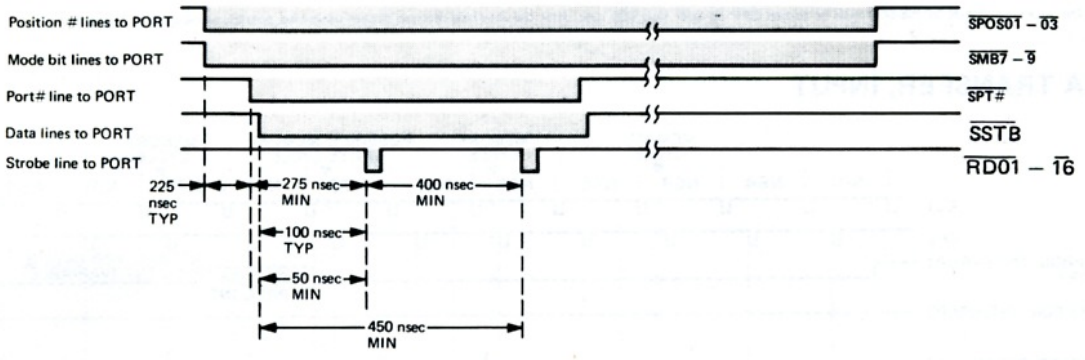

## SET TIMING



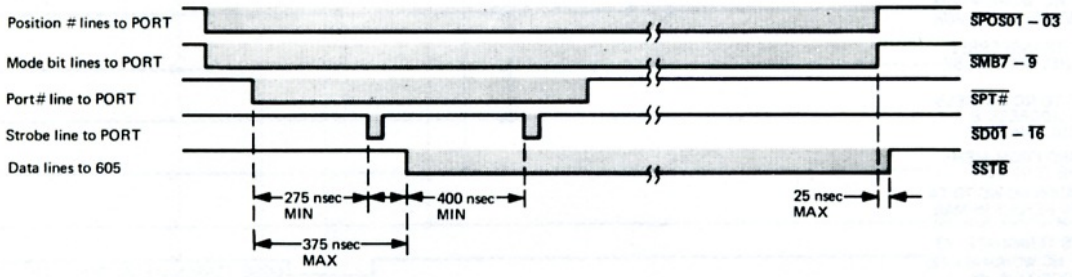
## SAMPLE TIMING



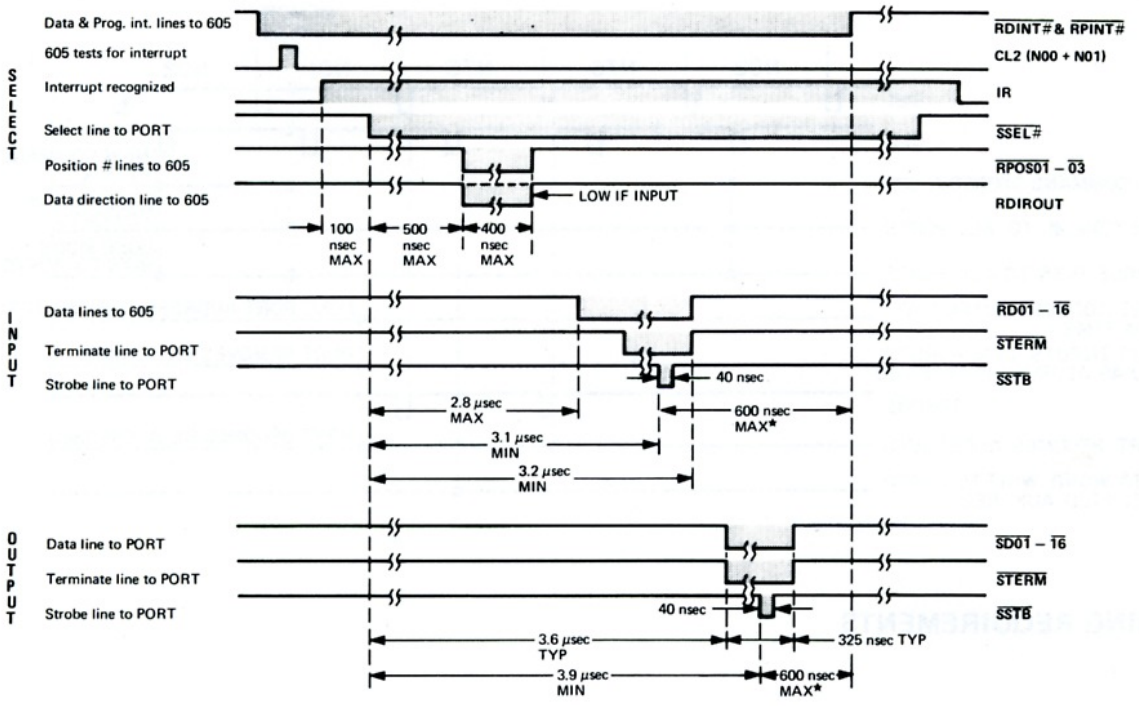

## SET TIMING REQUIREMENTS




## SAMPLE TIMING REQUIREMENTS

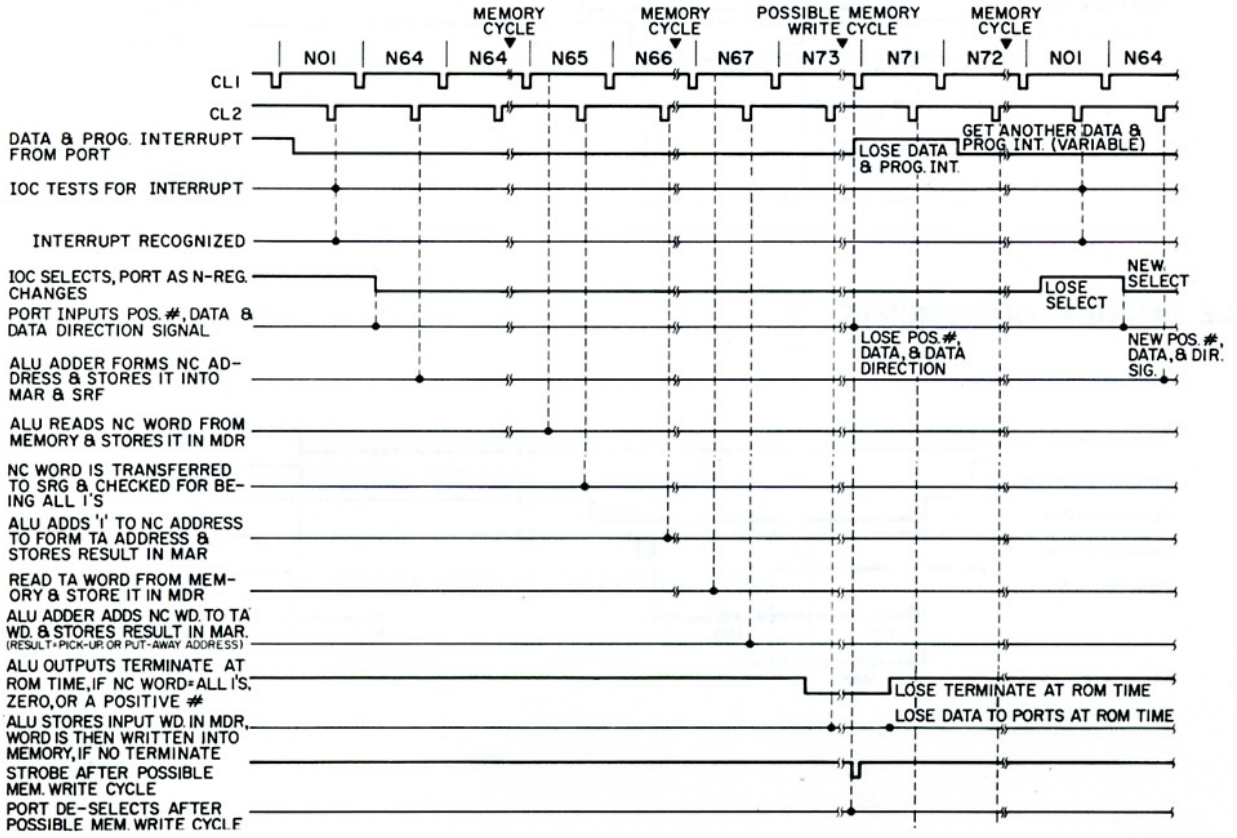


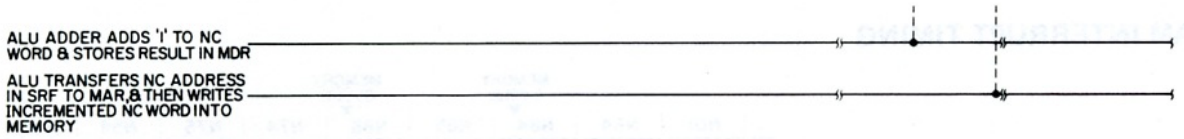
## ADT TIMING REQUIREMENTS



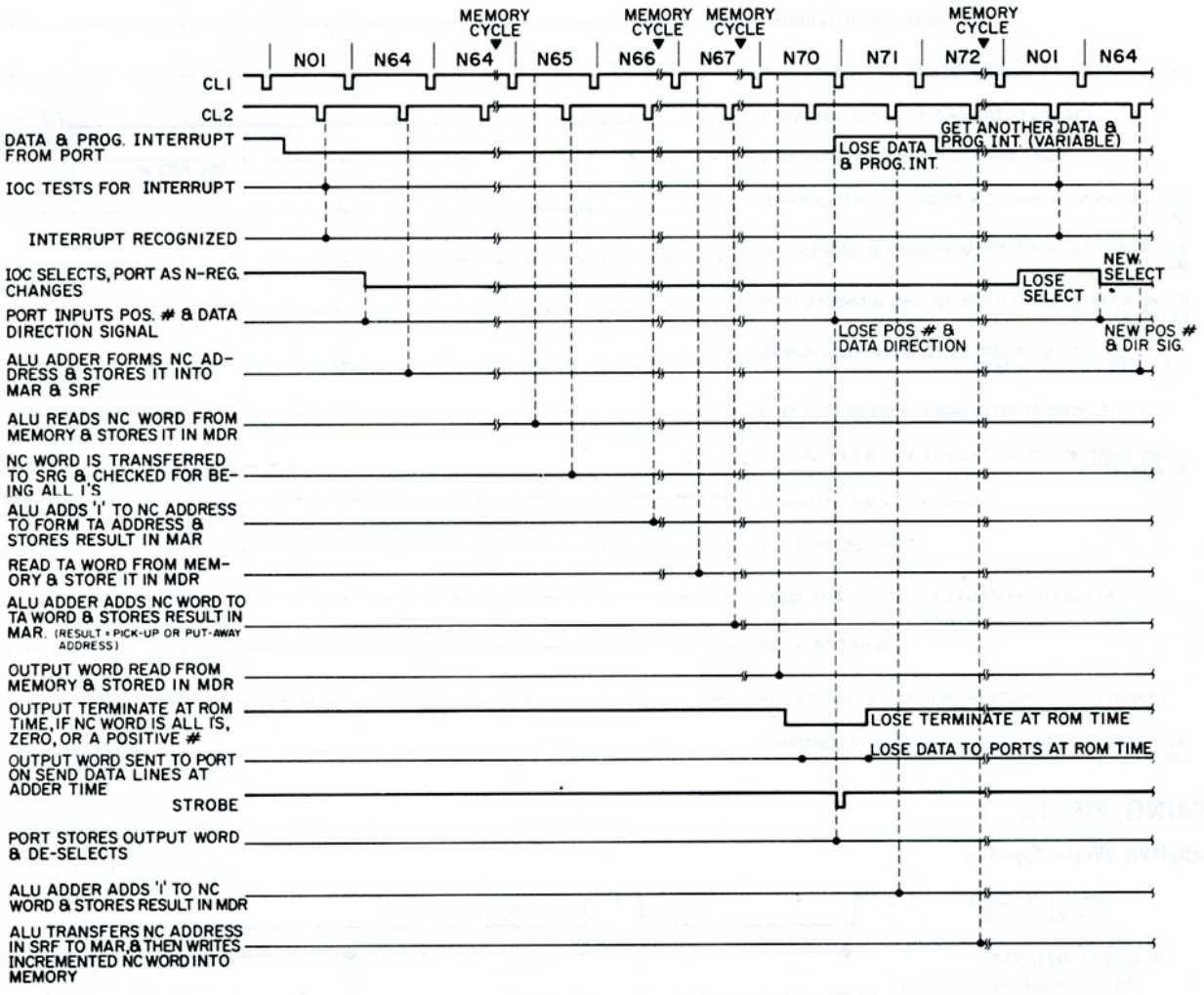

★ The interrupt signals to the 605 must be turned OFF no later than 600 nsec. after the interface receives strobe; otherwise, possible re-recognition may occur.

## AUTO DATA TRANSFER, INPUT

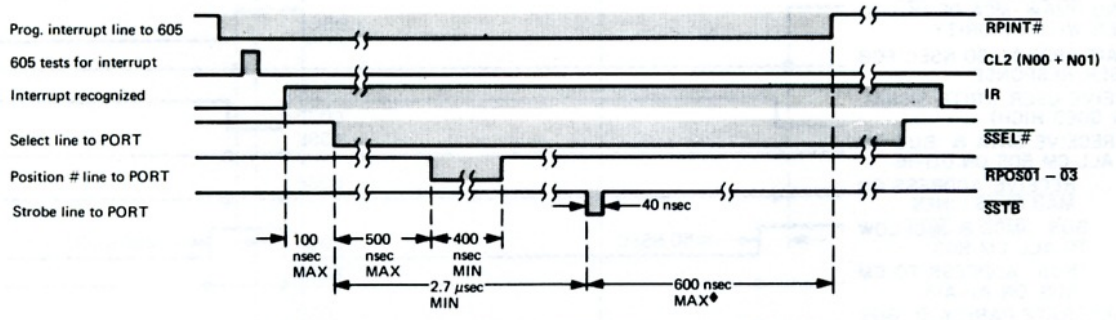





## AUTO DATA TRANSFER, OUTPUT

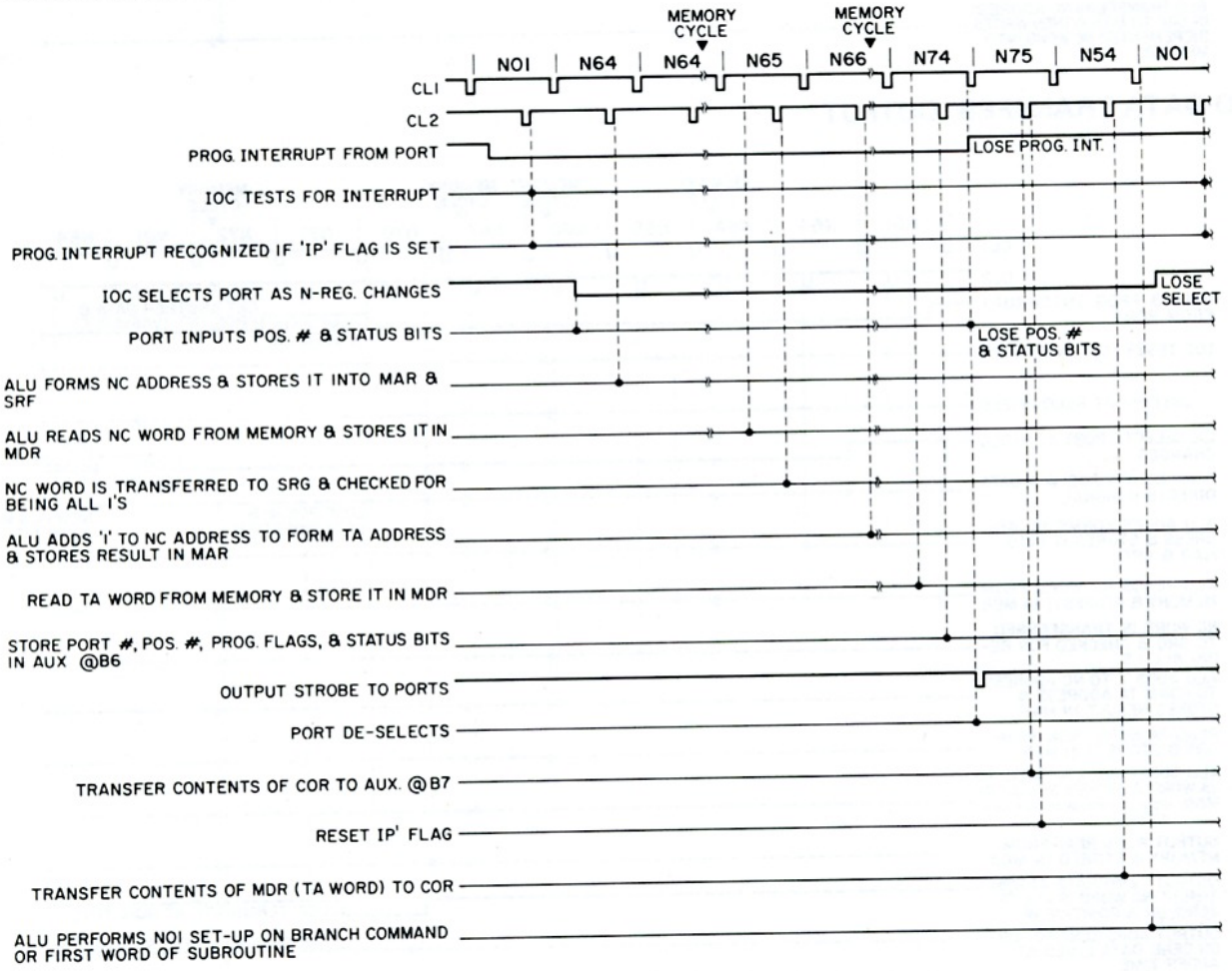



## PROGRAM INTERRUPT TIMING REQUIREMENTS



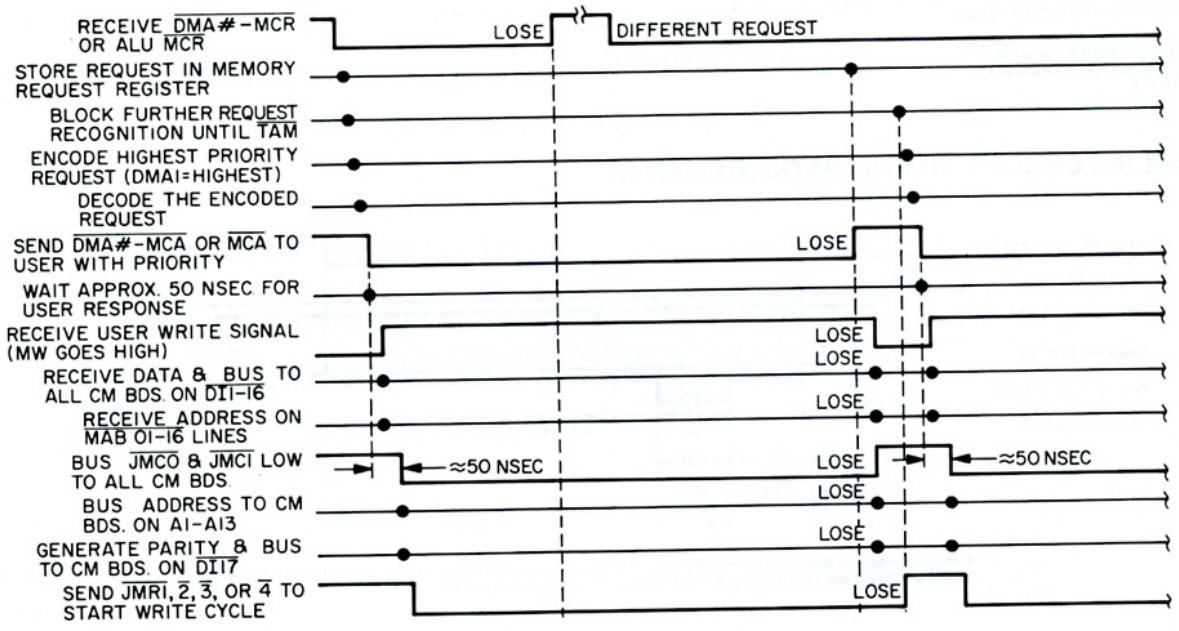

♦ Program interrupt should be turned OFF when interface receives select signal; however, it may be left ON until the time specified if design so dictates. See note under ADT Timing Requirements.

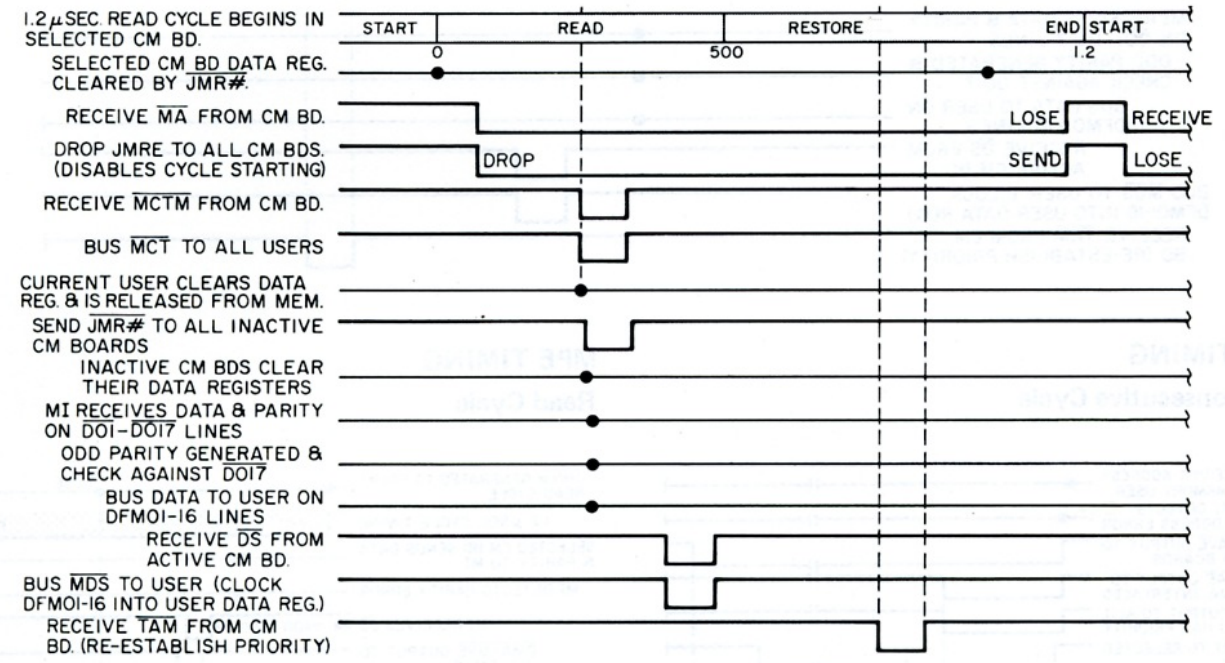
## PROGRAM INTERRUPT TIMING




## MI TIMING, WRITE

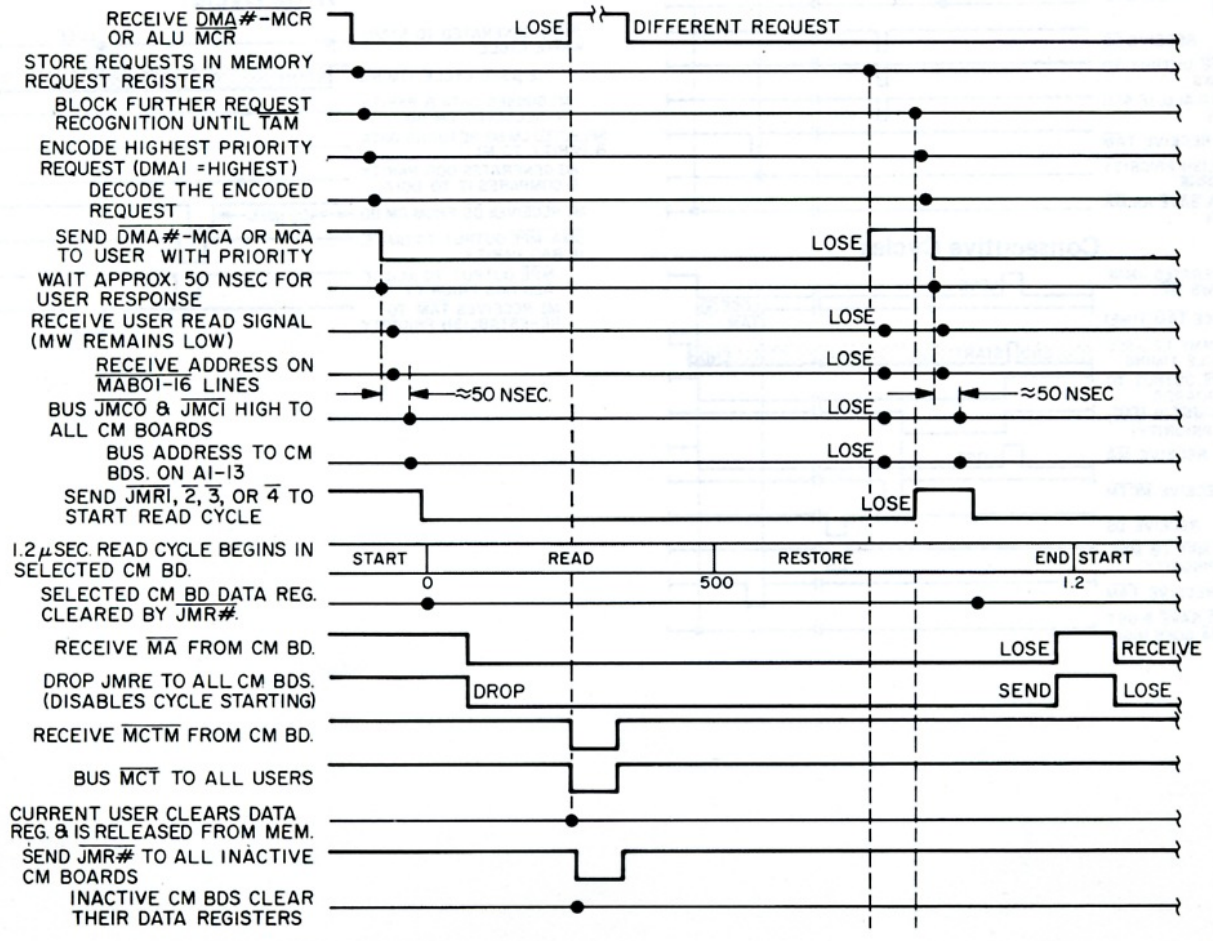
### Consecutive Write Cycles

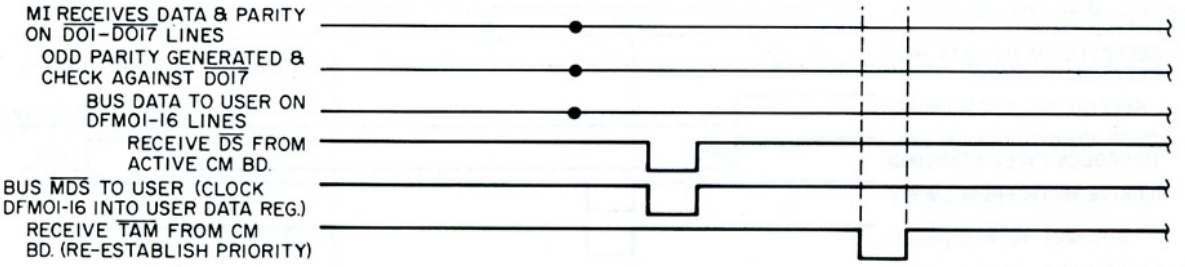





## MI TIMING, READ

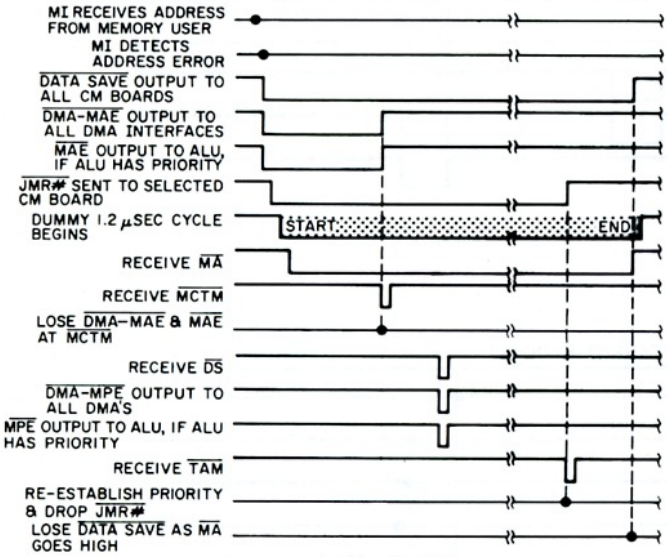
### Consecutive Read Cycles



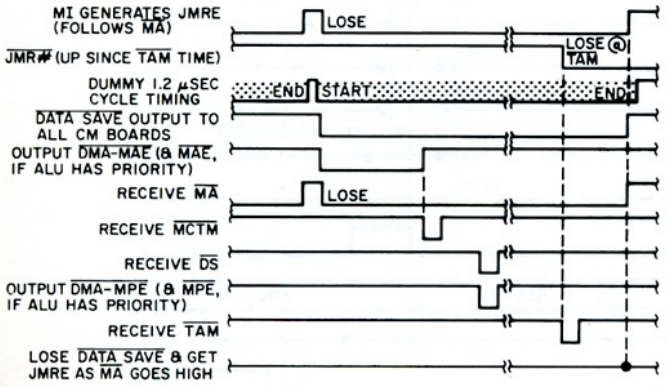



## MAE TIMING

### Non-Consecutive Cycle

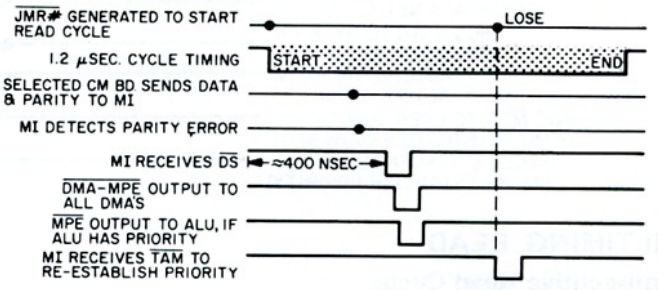



### Consecutive Cycles




## MPE TIMING

### Read Cycle




### Write Cycle