HEWLETT hp PACKARD

MAINTENANCE MANUAL

30001A

CENTRAL PROCESSOR UNIT/INPUT OUTPUT PROCESSOR

(FOR HP 3000 COMPUTER SYSTEMS)

Printed-Circuit Assemblies:

30001-60001
30001-60002
30001-60003
30001-60004
30001-60005
30001-60006
30001-60007
30001-60008
30001-60009
30001-60016
30001-60021

30001-90003

HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, U.S.A.

PRINTED: MAR 1974

LIST OF EFFECTIVE PAGES

Changed pages are designated by a change date in the lower corner of the page. Original pages do not indicate a change date. Insert latest changed pages and destroy superseded pages.

PAGE NUMBER	ISSUE
Title	Original
ii and iii	Original
iv Blank	Original
v thru xii	Original
1-1 thru 1-14	Original
2-1 thru 2-17	Original
2-18 Blank	Original
3-1 thru 3-40	Original
3-41	Original
3-42 Blank	Original
3-43	Original
3-44 Blank	Original
3-45	Original
3-46 Blank	Original
3-47	Original
3-48 Blank	Original
3-49	Original
3-50 Blank	Original
3-51 thru 3-58	Original
3-59	Original
3-60 Blank	Original
3-61	Original
3-62 Blank	Original
3-63	Original
3-64 Blank	Original
3-65	Original
3-66 Blank	Original
3-67	Original
3-68 Blank	Original
3-69	Original
3-70 Blank	Original
3-71	Original
3-72 Blank	Original
3-73	Original
3-74 Blank	Original
3-75	Original
3-76 Blank	Original
3-77	Original
3-78 Blank	Original
3-79	Original
3-80 Blank	Original
3-81	Original
3-82 Blank	Original
3-83	Original
3-84 Blank	Original
3-85	Original
3-86 Blank	Original
3-87	Original
3-88 Blank	Original
3-89	Original
3-90 Blank	Original
3-91	Original
3-92 Blank	Original
3-93	Original
3-94 Blank	Original
3-95	Original
3-96 Blank	Original
3-97	Original
3-98 Blank	Original
3-99	Original
3-100 Blank	Original
3-101	Original
3-102 Blank	Original
3-103	Original
3-104 Blank	Original
3-105	Original
3-106 Blank	Original
3-107	Original
3-108 Blank	Original
3-109	Original
3-110 Blank	Original
3-111	Original
3-112 Blank	Original
3-113	Original
3-114 Blank	Original
3-115	Original
3-116 Blank	Original
3-117	Original
3-118 Blank	Original
3-119	Original
3-120 Blank	Original
3-121	Original
3-122 Blank	Original
3-123	Original
3-124 Blank	Original
3-125	Original
3-126 Blank	Original
3-127	Original
3-128 Blank	Original
3-129	Original
3-130 Blank	Original
3-131	Original
3-132 Blank	Original
3-133	Original
3-134 Blank	Original
3-135	Original
3-136 Blank	Original
3-137	Original
3-138 Blank	Original
3-139	Original
3-140 Blank	Original
3-141	Original

LIST OF EFFECTIVE PAGES (Continued)

PAGE NUMBER	ISSUE
3-142 Blank	Original
3-143	Original
3-144 Blank	Original
3-145	Original
3-146 Blank	Original
3-147	Original
3-148 Blank	Original
3-149	Original
3-150 Blank	Original
3-151	Original
3-152 Blank	Original
3-153	Original
3-154 Blank	Original
3-155	Original
3-156 Blank	Original
3-157	Original
3-158 Blank	Original
3-159	Original
3-160 Blank	Original
3-161	Original
3-162 Blank	Original
3-163	Original
3-164 Blank	Original
3-165	Original
3-166 Blank	Original
3-167	Original
3-168 Blank	Original
3-169	Original
3-170 Blank	Original
3-171	Original
3-172 Blank	Original
3-173	Original
3-174 Blank	Original
3-175	Original
3-176 Blank	Original
3-177	Original
3-178 Blank	Original
3-179	Original
3-180 Blank	Original
3-181	Original
3-182 Blank	Original
3-183	Original
3-184 Blank	Original
3-185	Original
3-186 Blank	Original
3-187	Original
3-188 Blank	Original
3-189	Original
3-190 Blank	Original
3-191	Original
3-192 Blank	Original
3-193	Original
3-194 Blank	Original
3-195	Original
3-196 Blank	Original
3-197	Original
3-198 Blank	Original
3-199	Original
3-200 Blank	Original
3-201	Original
3-202 Blank	Original
3-203	Original
3-204 Blank	Original
3-205	Original
3-206 Blank	Original
3-207	Original
3-208 Blank	Original
3-209	Original
3-210 Blank	Original
3-211	Original
3-212 Blank	Original
3-213	Original
3-214 Blank	Original
3-215	Original
3-216 Blank	Original
3-217	Original
3-218 Blank	Original
4-1 thru 4-129	Original
4-130 Blank	Original
Certification	Original
Back Cover	Original

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

PREFACE

This manual contains maintenance information for the HP 30001A Central Processor Unit and Input/Output Processor (CPU/IOP). The HP 30001A CPU/IOP is part of the HP 3000 Computer System.

The contents of this manual are organized in four sections as follows:

a. Section I contains general information of the CPU/IOP physical features and specifications.

b. Section II contains operating parameters for the CPU/IOP including a list of all machine instructions for the HP 3000 Computer System.

c. Section III contains theory of operation for the CPU/IOP.

d. Section IV contains servicing instructions including preventive maintenance information and trouble-shooting instructions.

This manual should be retained and used with related documentation for the HP 3000 Computer Systems. The related documentation should include the following publications:

a. *HP 3000 Computer System Reference Manual*, part no. 03000-90019.

b. The following detailed diagram sets in the *HP 3000 Computer System Detailed Diagrams Manual*, part no. 03000-90023.

   1) DD Set No. 200. Read-Only Memory Printed-Circuit Assembly (PCA), part no. 30001-60001.

   2) DD Set No. 201. Skip and Special Field PCA, part no. 30001-60002.

   3) DD Set No. 202. Arithmetic and Logic Unit PCA, part no. 30001-60003.

   4) DD Set No. 203. R-Bus PCA, part no. 30001-60004.

   5) DD Set No. 204. S-Bus PCA, part no. 30001-60005.

   6) DD Set No. 205. Current Instruction Register PCA, part no. 30001-60006.

   7) DD Set No. 206. Module Control Unit PCA, part no. 30001-60007.

   8) DD Set No. 207. Input/Output Processor PCA, part no. 30001-60008.

   9) DD Set No. 208. Central Data Bus Terminator PCA, part no. 30001-60009.

   10) DD Set No. 209. Input/Output Processor Bus Terminator PCA, part no. 30001-60016.

   11) DD Set No. 210. Power Bus Terminator PCA, part no. 30001-60021.

c. *HP 3000 Computer System Illustrated Parts Breakdown (IPB) Manual*, part no. 03000-90021.

d. *HP 3000 Manual of Stand-Alone Diagnostics, Stand-Alone HP 30001A CPU Diagnostic*, part no. 03000-90027.

e. *HP 3000 Computer System Diagnostic Monitor*, part no. 03000-90016.

f. *CPU Microprogram Listing Manual*, part no. 03000-90022.

g. CPU Microdiagnostic Listings, part no. 32300-90001A, 32300-90002A, 32300-90003A, 32300-90004A, and 32300-90005A.

h. HP 3000 System Support Log.

i. *HP 30350A Auxiliary Control Panel Maintenance Manual*, part no. 30350-90002 and *Operator's Guide*, part no. 30350-90001.

j. *HP 30035A Multiplexer Channel Maintenance Manual*, part no. 30035-90001.

k. *HP 30005A/30006A Memory Subsystem Maintenance Manual*, part no. 30005-90001.

l. *HP 30310A Power Supply Maintenance Manual*, part no. 30310-90003.

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

ILLUSTRATIONS (Continued)

TABLES

SECTION I GENERAL INFORMATION

1-1. INTRODUCTION.

1-2. This section describes the functional and physical features of the HP 30001A Central Processor Unit/Input Output Processor (figure 1-1). Specifications and equipment identification data are also provided. Related publications that may be required for operation of the central processor unit/input output processor are listed in the preface of this manual.

1-3. GENERAL DESCRIPTION.

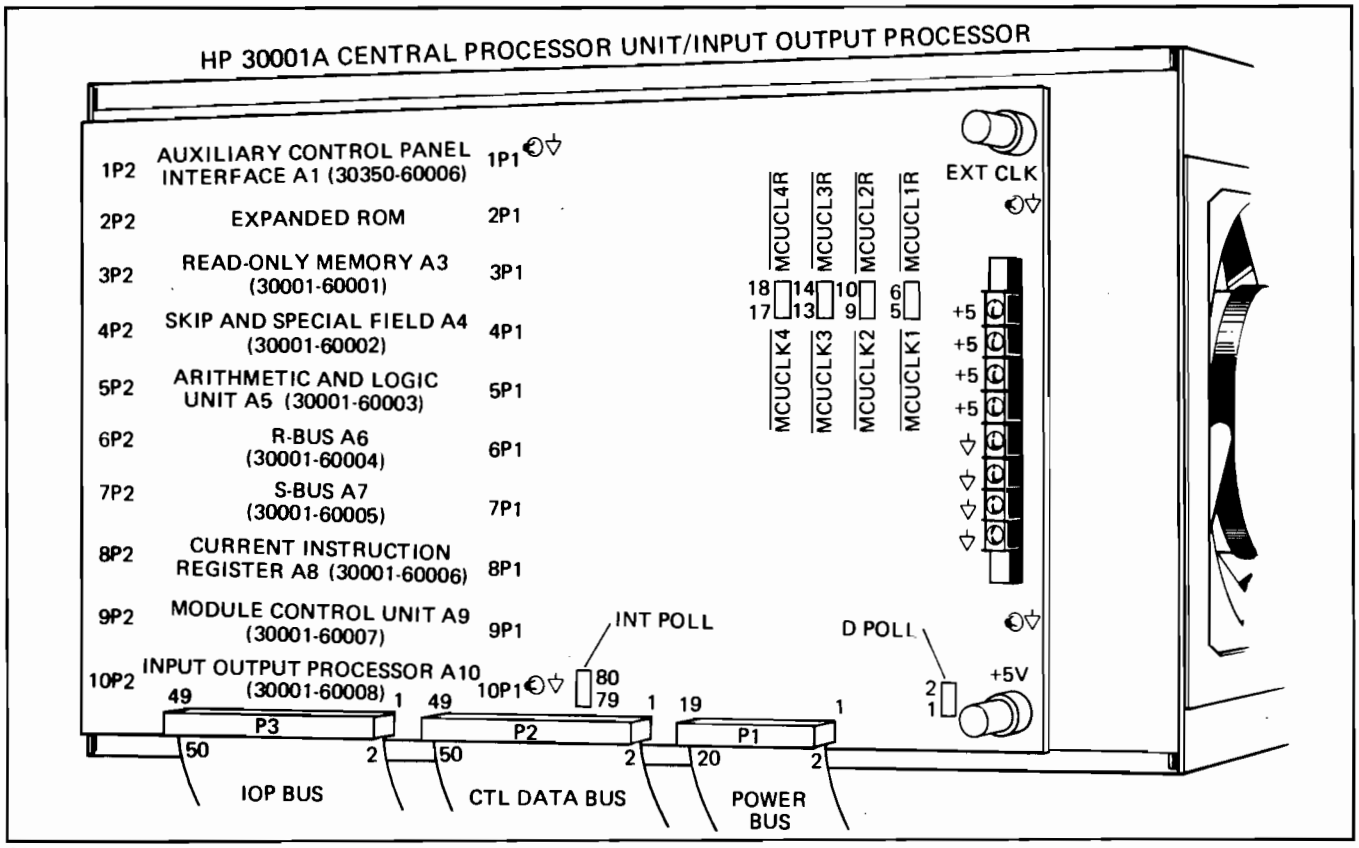
1-4. The central processor unit/input output processor is divided into three major functional sections: Central processor unit (CPU), input output processor (IOP), and module control unit (MCU). The MCU is shared by the CPU and IOP.

1-5. CENTRAL PROCESSOR UNIT.

1-6. The CPU contains most of the logic circuitry to perform the CPU/IOP functions. The major elements are a read-only memory (ROM) containing a microprogram, registers, and logic to transfer data or operands to or from memory or the IOP. Briefly, the sequence of events for the CPU is as follows: The CPU requests an instruction from memory via the MCU. When the instruction is received by the CPU, it is loaded into the next instruction register (NIR). When the current instruction is executed, the new instruction is transferred from the NIR to the current instruction register (CIR). The information contained in the new instruction causes the CPU to begin executing a microprogram which is read out of the ROM. The microprogram is decoded into a set of control signals and the contents of the CPU registers are manipulated in accordance with the microprogram. The microprogram may change the state of one or more of the registers and may initiate the transfer of operands or data to or from memory. At the conclusion of the microprogram, the desired action (such as a computation between two registers) is complete. The last step of the microprogram is to load the new NIR contents into the CIR for execution of the next instruction.

1-7. INPUT OUTPUT PROCESSOR.

1-8. The primary functions of the IOP are to provide communication between the CPU and the I/O subsystem



2184-125

Figure 1-1. HP 30001A Central Processor Unit/Input Output Processor

under control of the CPU and communication between the I/O subsystems and the memory under control of the multiplexer channel. These two functions are known as direct I/O and multiplexed I/O, respectively. In addition, the IOP receives I/O interrupts and resolves interrupt priorities. There are eight direct I/O commands which, when executed by the CPU, cause the IOP logic to perform some function. These I/O commands are as follows:

a. Set Interrupt (SIN).

b. Reset Interrupts (RIN).

c. Start I/O (SIO).

d. Set Mask (SMSK).

e. Control I/O (CIO).

f. Test I/O (TIO).

g. Write I/O (WIO).

h. Read I/O (RIO).

1-9. Operation under control of the multiplexer channel is described in detail in the *HP 30035A Multiplexer Channel Maintenance Manual*, part no. 30035-90001.

1-10. MODULE CONTROL UNIT.

1-11. The MCU consists of two nearly identical units, one for the CPU and one for the IOP. The MCU controls inter-module communication via the central data bus and establishes priority between the CPU and the IOP. Normally, the IOP has higher priority in gaining access to the bus; however, when the CPU is attempting to complete a semi-executed operation, the CPU takes higher priority.

1-12. EQUIPMENT DESCRIPTION.

1-13. The HP 30001A Central Processor Unit/Input Output Processor consists of the following components:

a. Read-only memory printed-circuit assembly A3, part no. 30001-60001 (see figure 1-2).

b. Skip and special field printed-circuit assembly A4, part no. 30001-60002 (see figure 1-3).

c. Arithmetic and logical unit printed-circuit assembly A5, part no. 30001-60003 (see figure 1-4).

d. R-bus printed-circuit assembly A6, part no. 30001-60004 (see figure 1-5).

e. S-bus printed-circuit assembly A7, part no. 30001-60005 (see figure 1-6).

f. Current instruction register printed-circuit assembly A8, part no. 30001-60006 (see figure 1-7).

g. Module control unit printed-circuit assembly A9, part no. 30001-60007 (see figure 1-8).

h. Input output processor printed-circuit assembly A10, part no. 30001-60008 (see figure 1-9).

i. Central data bus terminator printed-circuit assembly, part no. 30001-60009 (see figure 1-10).

j. Input output processor bus terminator printed-circuit assembly, part no. 30001-60016 (see figure 1-11).

k. Power bus terminator printed-circuit assembly, part no. 30001-60021 (see figure 1-12).

l. Maintenance manual, part no. 30001-90003.

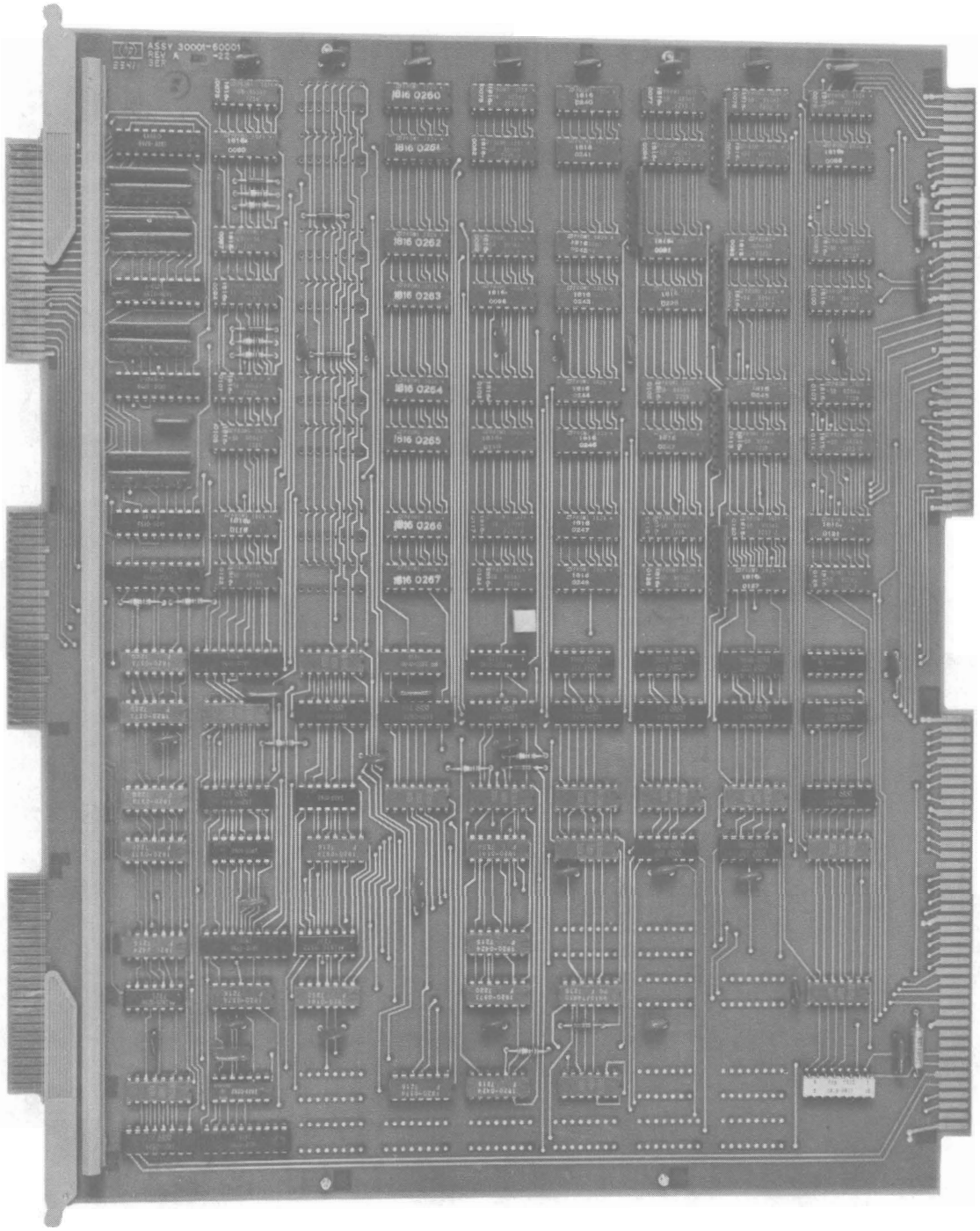
1-14. The maintenance data for the CPU/IOP consists of this maintenance manual (listed above) and detailed diagram sets DD200 through DD211, part no. 30001-90005 through 30001-90019, contained in the *HP 3000 Computer System Detailed Diagrams Manual*, part no. 03000-90023.

1-15. SPECIFICATIONS.

1-16. Specifications for the CPU/IOP are listed in table 1-1.

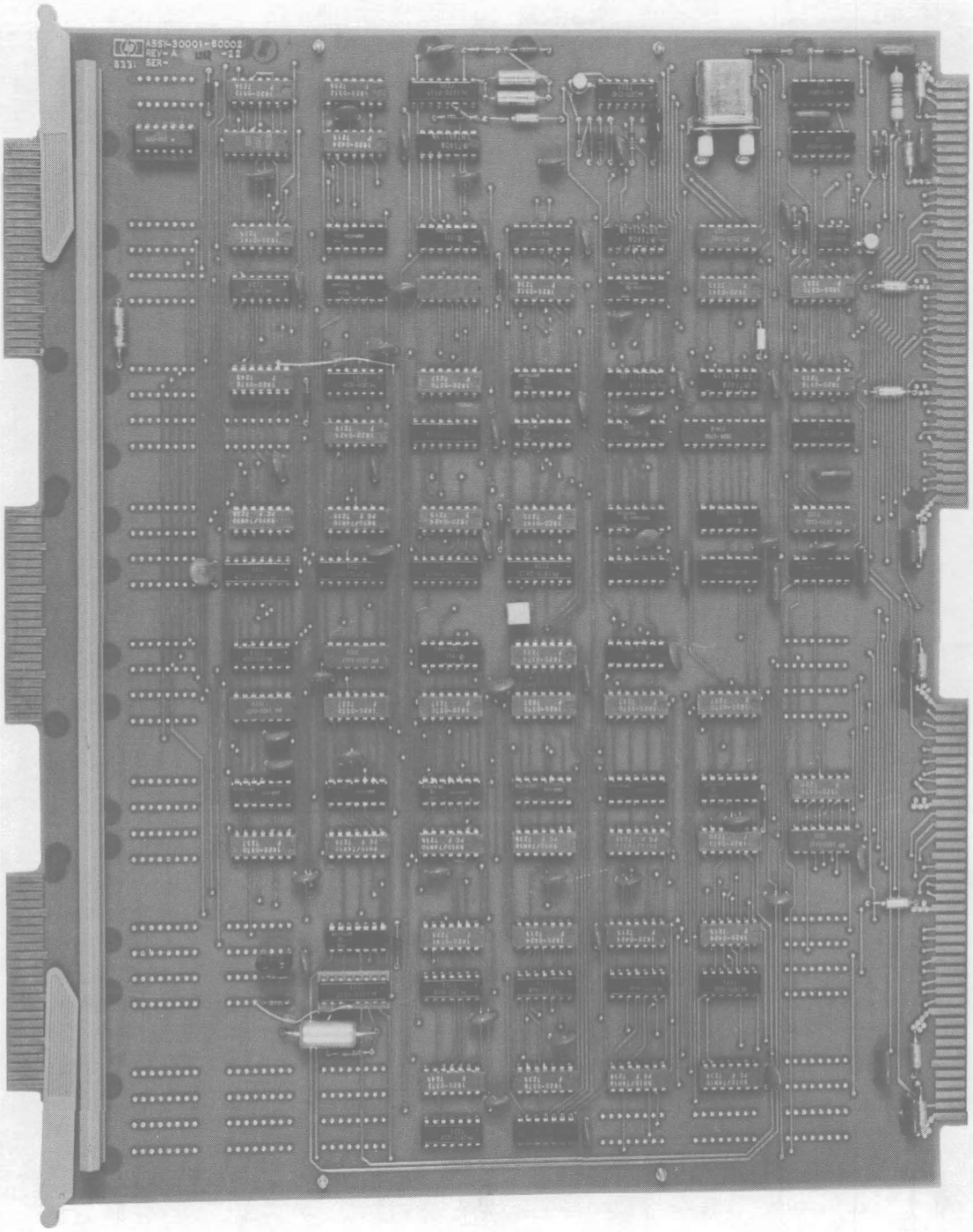
1-17. IDENTIFICATION.

1-18. Printed-circuit assembly (PCA) revisions are identified by a letter, series code, and a division code stamped on the PCA (eg, A-1130-22). The letter identifies the version of the etched trace pattern on the unloaded PCA. The series code (middle digits) refers to the electrical characteristics of the loaded PCA. The division code (last two digits) identifies the Hewlett-Packard division that manufactured the PCA.



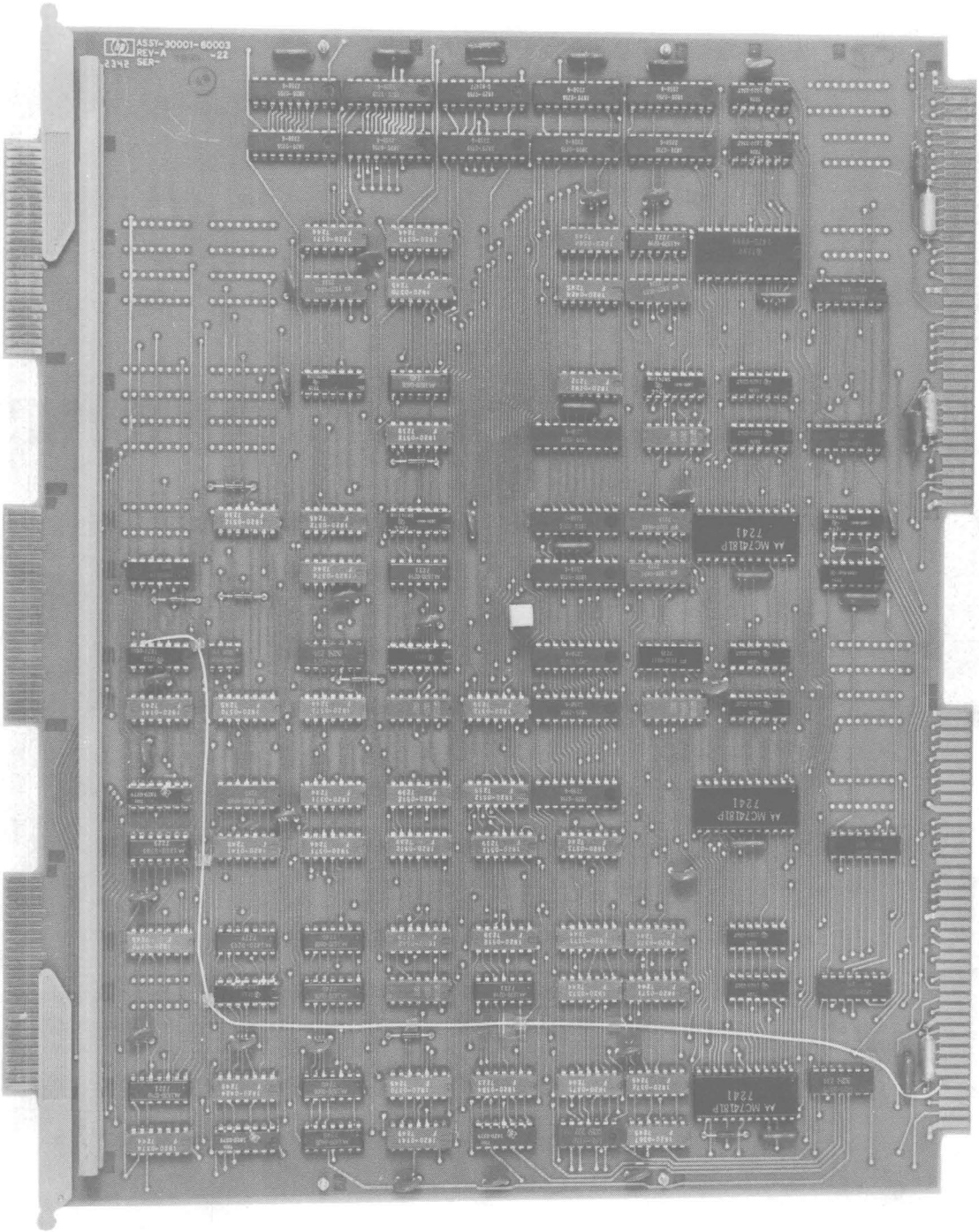
2184-126

Figure 1-2. Read-Only Memory Printed-Circuit Assembly A3, Part No. 30001-60001.



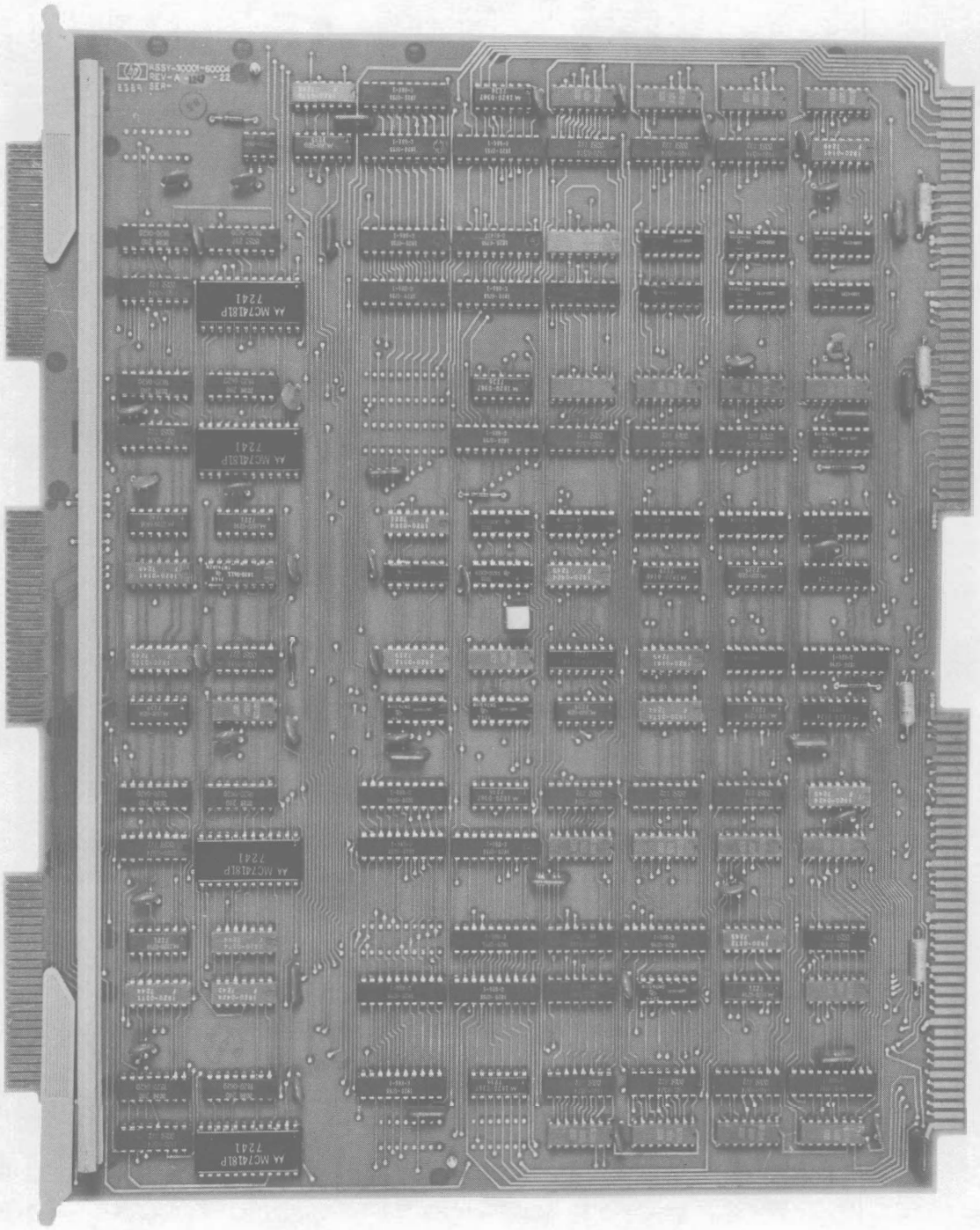
2184-127

Figure 1-3. Skip and Special Field Printed-Circuit Assembly A4, Part No. 30001-60002



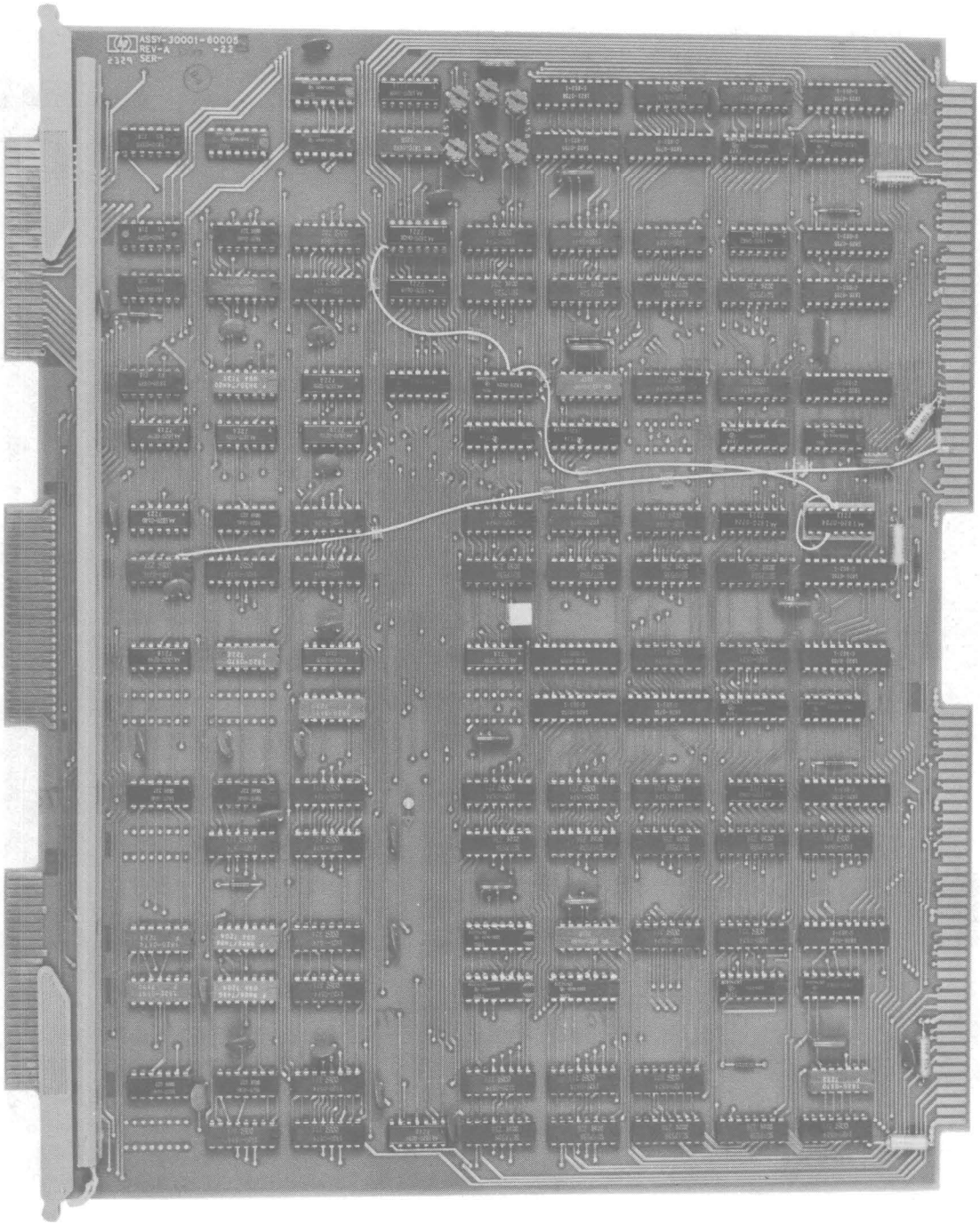
2184-128

Figure 1-4. Arithmetic and Logic Unit Printed-Circuit Assembly A5, Part No. 30001-60003



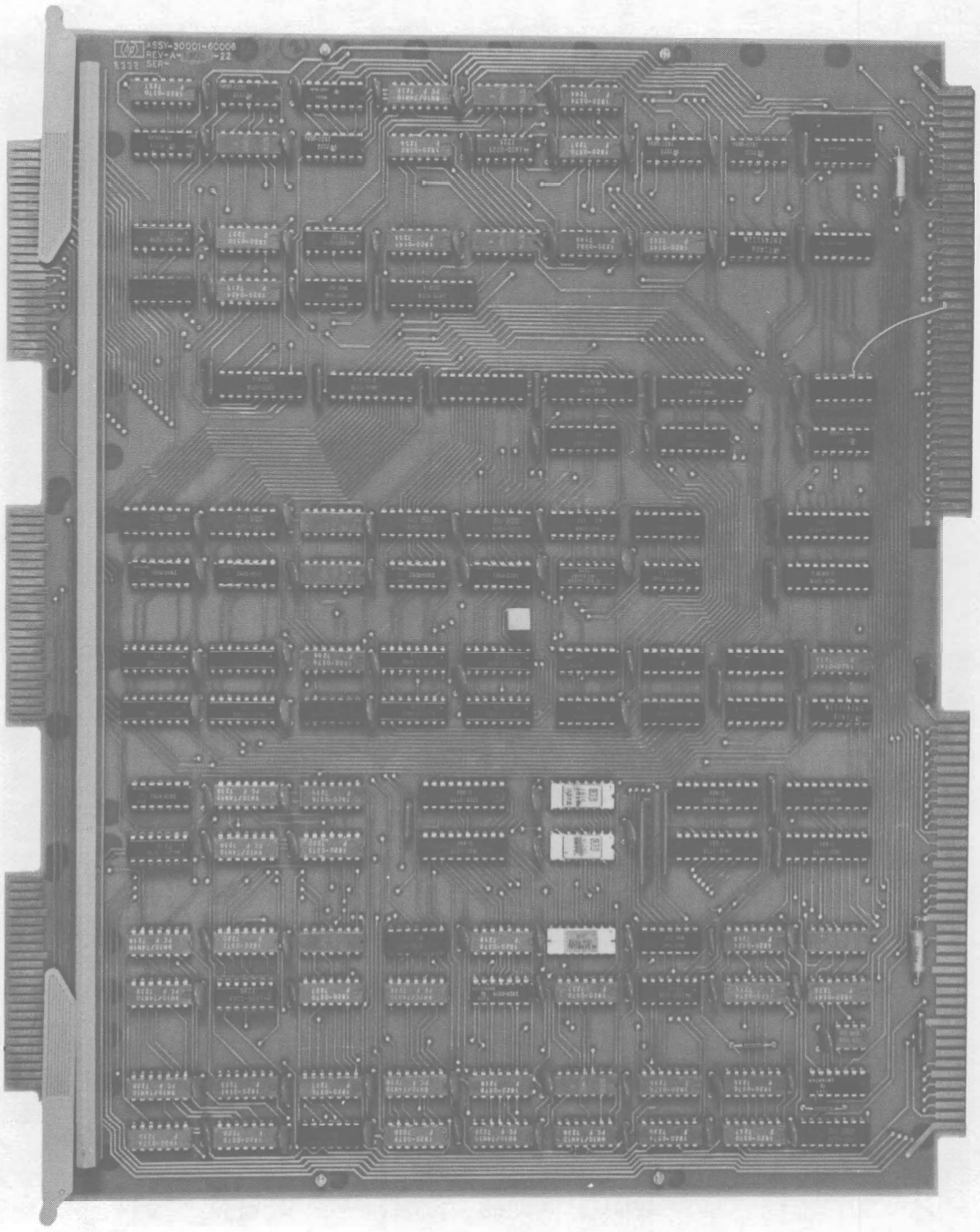
2184-129

Figure 1-5. R-Bus Printed-Circuit Assembly A6, Part No. 30001-60004



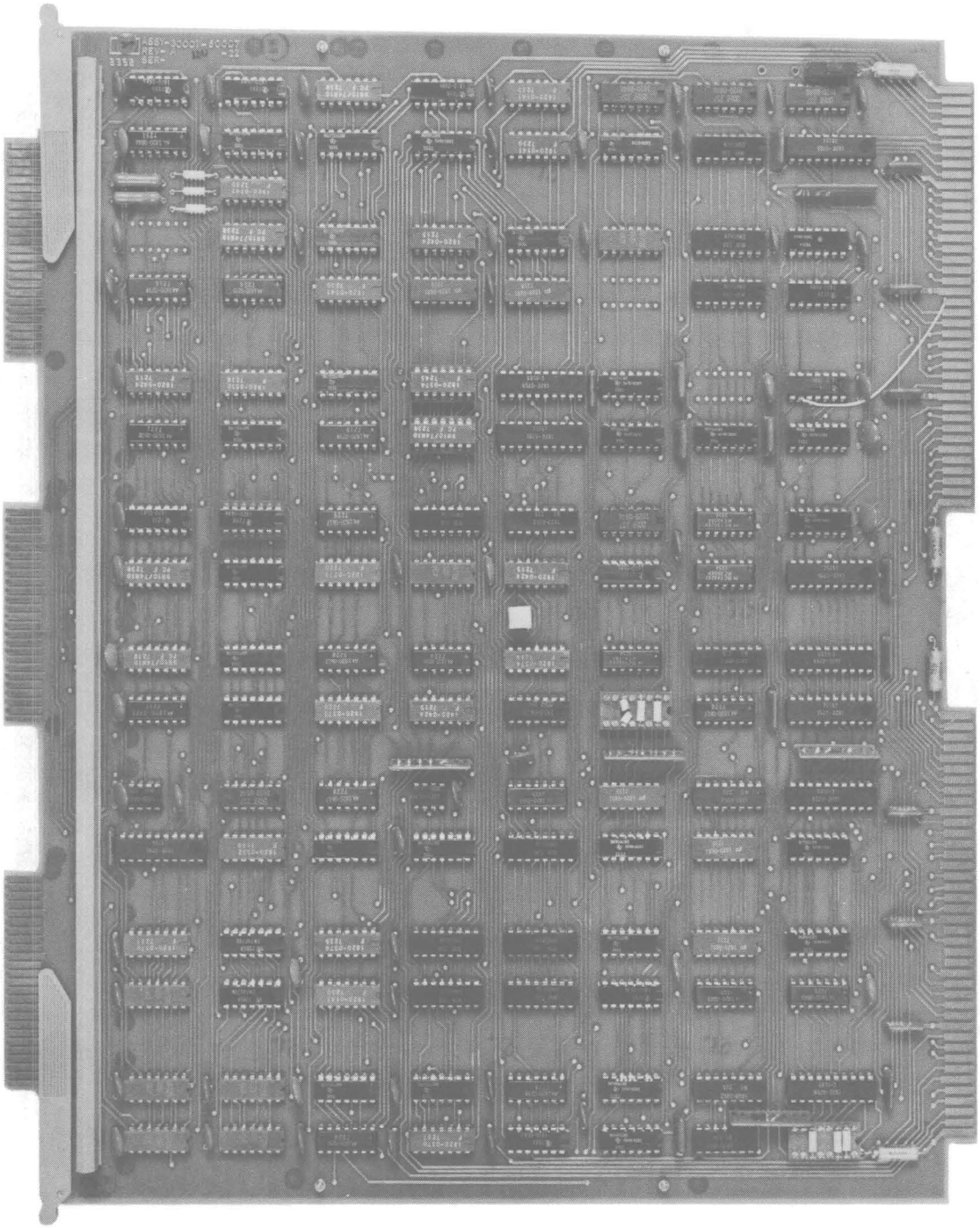
2184-130

Figure 1-6. S-Bus Printed Circuit Assembly A7, Part No. 30001-60005



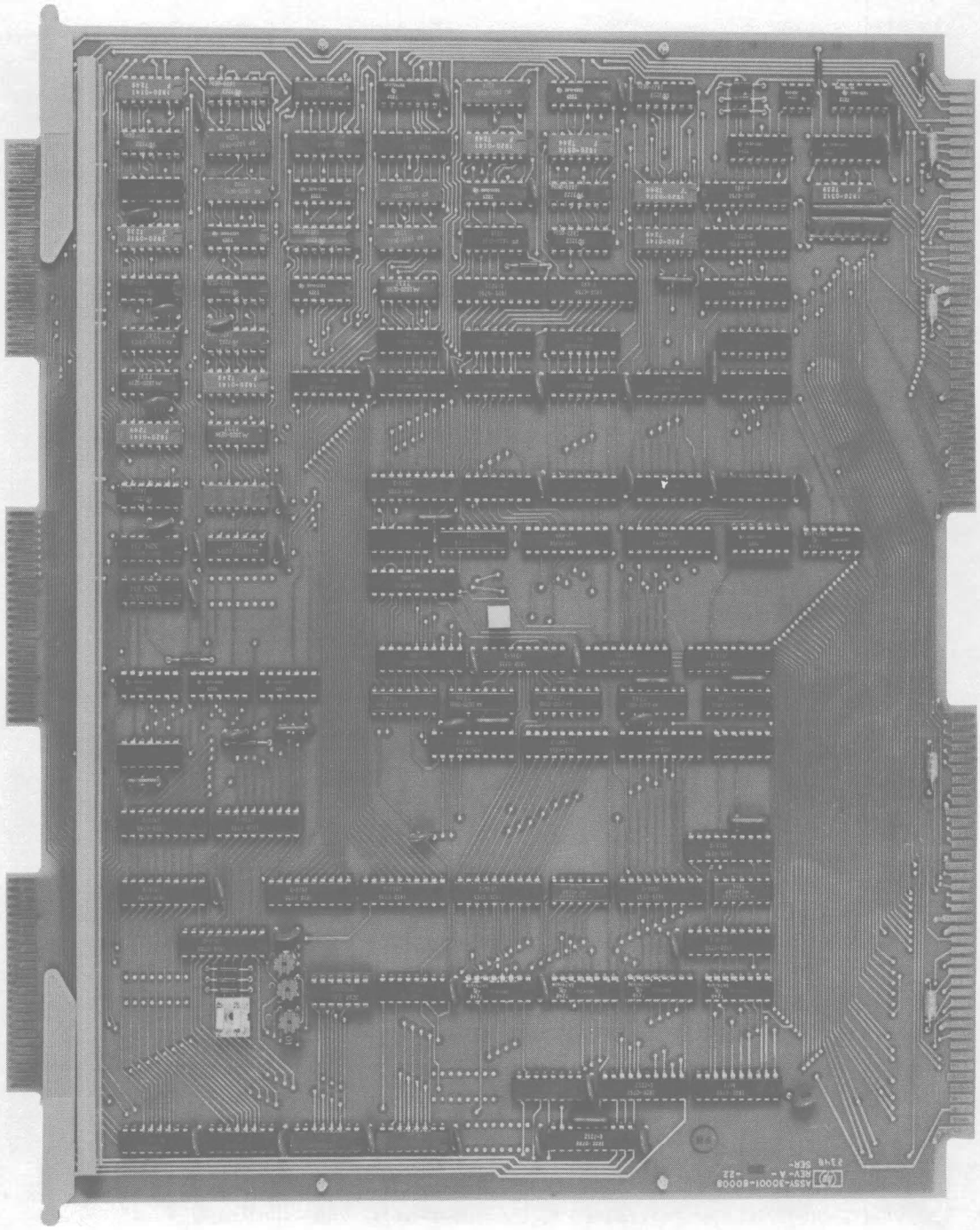
2184-131

Figure 1-7. Current Instruction Register Printed-Circuit Assembly A8, Part No. 30001-60006



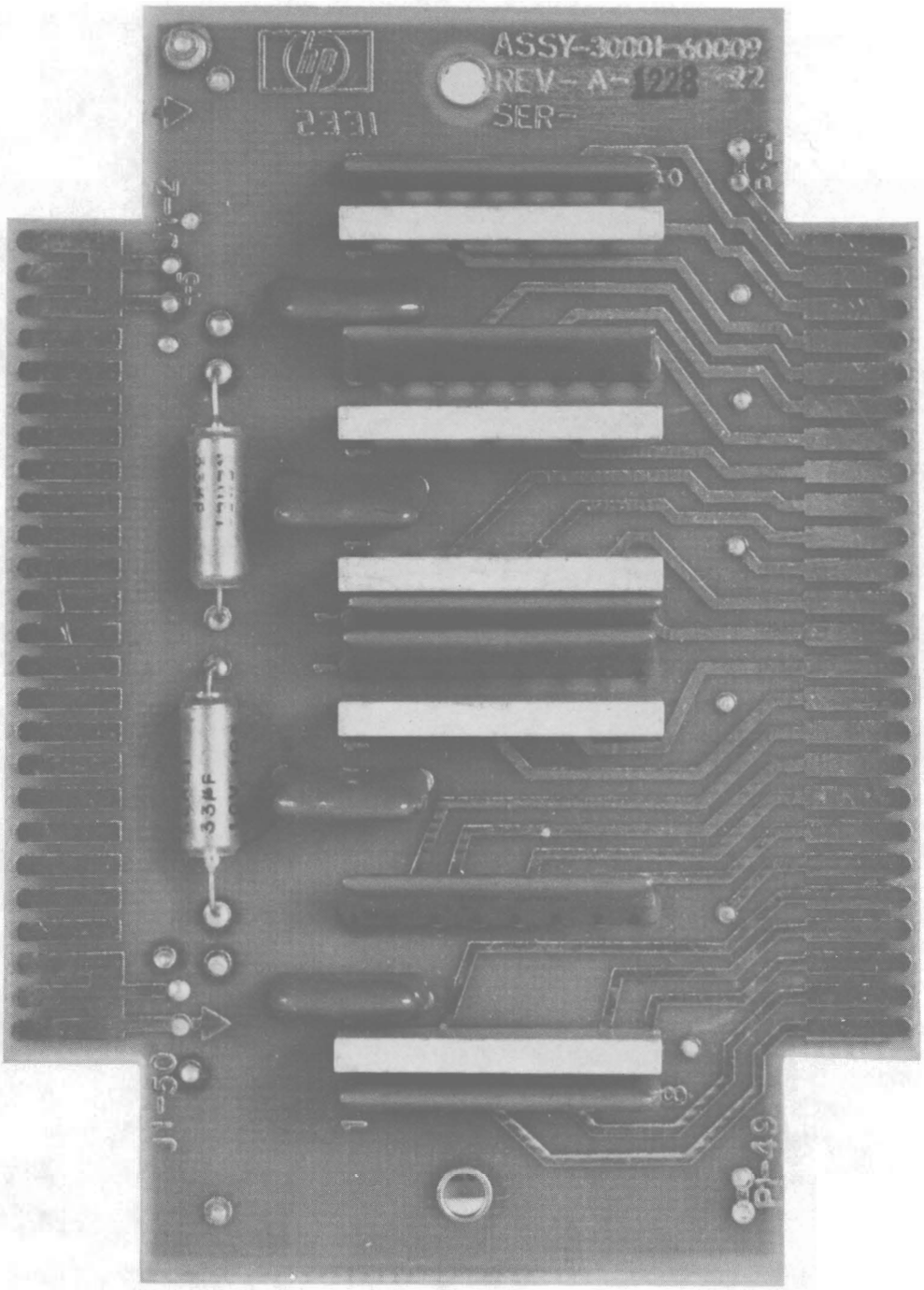
2184-132

Figure 1-8. Module Control Unit Printed-Circuit Assembly A9, Part No. 30001-60007



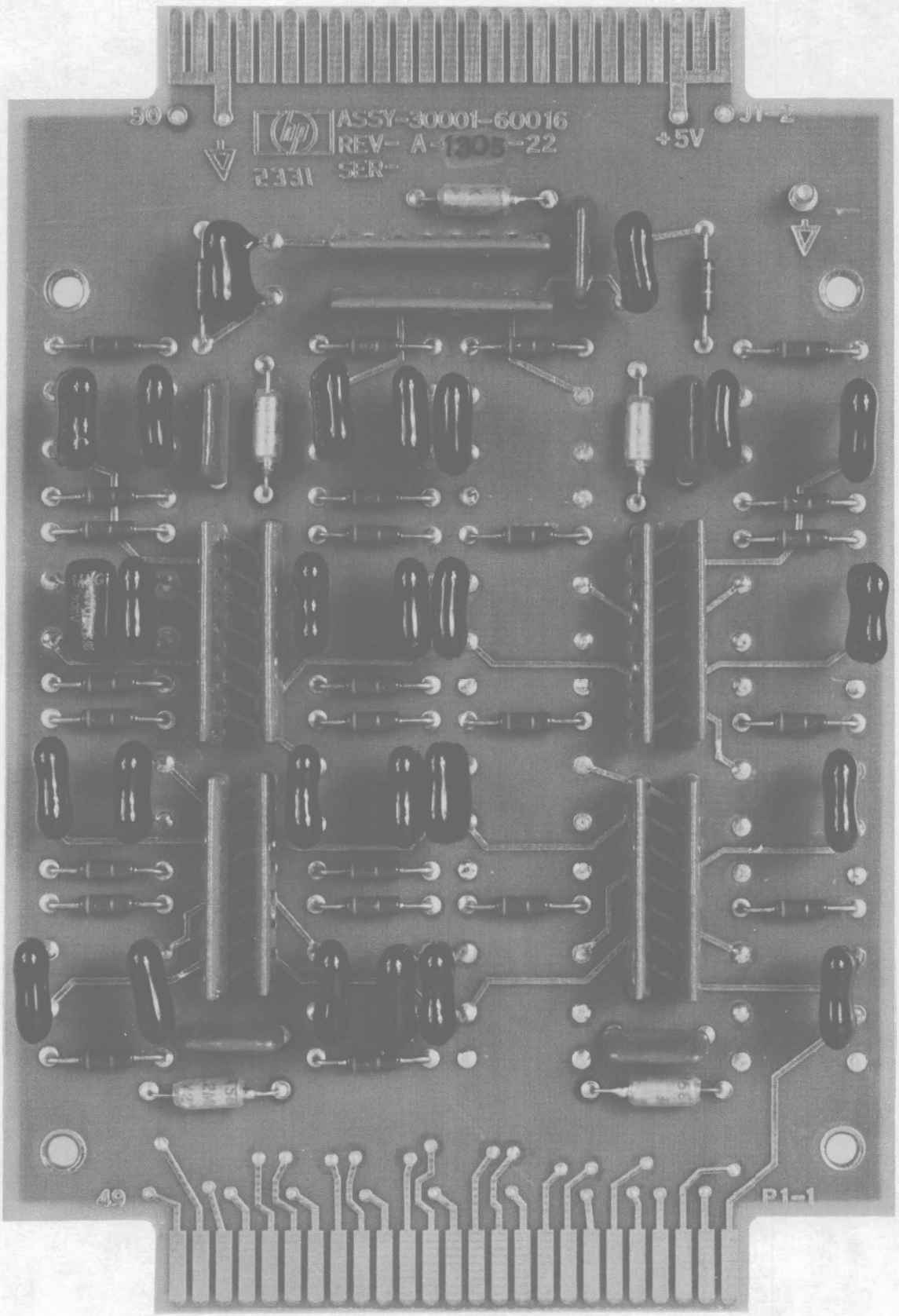
2184-133

Figure 1-9. Input Output Processor Printed-Circuit Assembly A10, Part No. 30001-60008



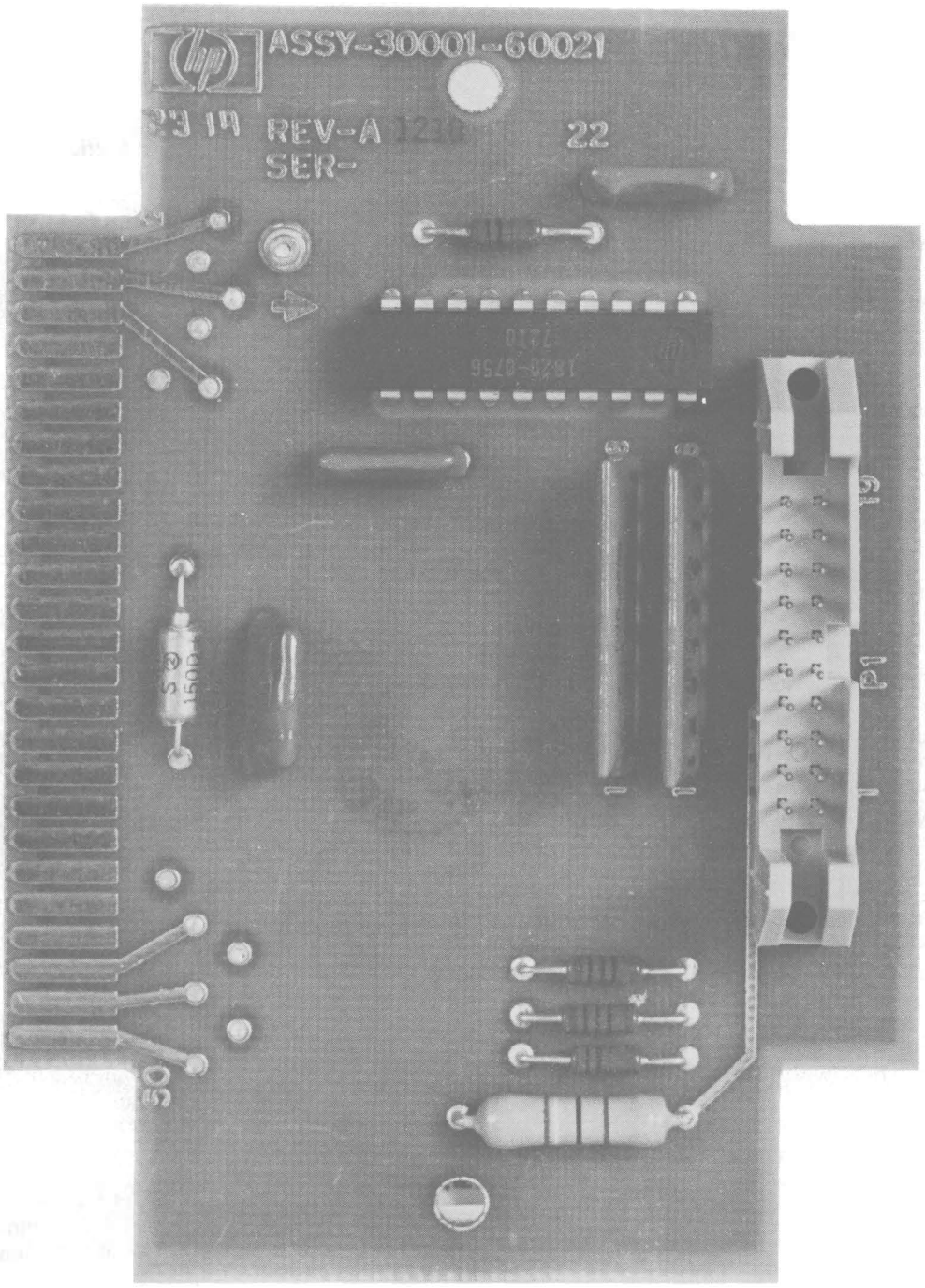
2184-134

Figure 1-10. Central Data Bus Terminator Printed-Circuit Assembly, Part No. 30001-60009



2184-135

Figure 1-11. Input Output Processor Bus Terminator Printed-Circuit Assembly, Part No. 30001-60016



2184-136

Figure 1-12. Power Bus Terminator Printed-Circuit Assembly, Part No. 30001-60021

Table 1-1. CPU/IOP Specifications

CHARACTERISTICS	SPECIFICATIONS
CURRENT REQUIRED FROM COMPUTER POWER SUPPLY	
+5-volt supply	37.4 Amperes
LOGIC LEVELS	
Logic 1 level (high):	+2.4 Vdc minimum
Logic 0 level (low):	+0.4 Vdc maximum
PHYSICAL CHARACTERISTICS	
Read-Only Memory (ROM) PCA, part no. 30001-60001	
Skip and Special Field (SSF) PCA, part no. 30001-60002	
Arithmetic and Logic Unit (ALU) PCA, part no. 30001-60003	
R-Bus PCA, part no. 30001-60004	
S-Bus PCA, part no. 30001-60005	
Current Instruction Register (CIR) PCA, part no. 30001-60006	
Module Control Unit (MCU) PCA, part no. 30001-60007	
Input Output Processor (IOP) PCA, part no. 30001-60008	
Depth:	11-1/2 in. (292.1 mm)
Width:	13-11/16 in. (347.67 mm)
Thickness (with components):	5/8 in. (15.875 mm)
Weight:	1 pound, 3 ounces
NOTE	
Physical dimensions are the same for all the previously listed printed-circuit assemblies.	
Central Data Bus Terminator PCA, part no. 30001-60009	
Depth:	2-3/4 in. (70 mm)
Width:	3-7/8 in. (98.7 mm)
Thickness (with components):	3/8 in. (9.5 mm)
Weight:	7 ounces
IOP Bus Terminator PCA, part no. 30001-60016	
Depth:	5-3/4 in. (14.6 mm)
Width:	4 in. (10.15 mm)
Thickness (with components):	1/2 in. (12.7 mm)
Weight:	7 ounces
Power Bus Terminator PCA, part no. 30001-60021	
Depth:	2-3/4 in. (70 mm)
Width:	3-7/8 in. (98.7 mm)
Thickness (with components):	3/4 in. (19 mm)
Weight:	7 ounces

SECTION II OPERATING PARAMETERS

2-1. INTRODUCTION.

2-2. This section contains the following information:

a. HP 3000 Computer System instructions.
b. Instruction word formats.
c. Consolidated microinstruction coding diagram.
d. Microinstruction word formats.

2-3. HP 3000 COMPUTER SYSTEM INSTRUCTIONS.

2-4. The HP 3000 instruction set consists of 170 unique instructions which are listed in alphabetical order in table 2-1. As the CPU executes a user program, it sequentially fetches these instructions from memory. The bit pattern of each instruction allows a ROM look-up table and decoding logic to decode a ROM address for a microprogram stored in the microprogram ROM. There is a microprogram in ROM for each of the 170 machine instructions. The ROM address, decoded by the decoding logic, is stored in a ROM address register (RAR). The RAR is used first to access the starting microprogram address and is then incremented to point to the next microinstruction. Thus, the entire microprogram for a particular machine instruction is called and executed by the CPU.

2-5. The instruction words from memory are divided into thirteen format groups. The formats are as shown in figure 2-1. Only the first field is rigidly adhered to. This field, bits 0 through 3, either defines a specific instruction code in the memory address group (or the "loop control" group), or defines one of the sub-opcode groups. There are four sub-opcode groups: 1, 2, 3 and stack ops. The fields for the sub-opcodes vary: for sub-opcodes 2 and 3, bits 4:7 are used. For sub-opcode group 1 codes, bits 5:9 are used, and for stack ops the remainder of the word is used. In some cases, the sub-opcode will enable a third field, called a mini-opcode or a special opcode, in bits 8:11. The remainder of the word has several uses and is usually part of an argument field. The following paragraphs provide a brief description of the format groups. For a more detailed description, refer to the *HP 3000 Computer System Reference Manual*, Part No. 03000-90019.

2-6. STACK OP INSTRUCTIONS.

2-7. Stack op instructions are shown in table 2-2. The stack op format is defined by four "0's" in the first four bits. The remaining 12 bits are divided into two fields: stack op A and stack op B. Either or both of these fields may contain any of the 63 stack op instruction codes. Execution sequence is from left to right (A first, then B). Interrupts may occur between the execution of A and B.

2-8. SHIFT INSTRUCTIONS.

2-9. Shift instructions are shown in table 2-3. The shift instruction group uses about half of the sub-opcode 1 group of codes. Sub-opcode group 1 is defined by 0001 in the first four bits (0:3). If bit 4, the index bit is a "1," the content of the index register is added to the shift count in bits 10 through 15 to specify the number of places each data bit is shifted. Bits 5 through 9 encode the specific shift instructions.

2-10. BRANCH INSTRUCTIONS.

2-11. Branch instructions are shown in table 2-4. The branch instructions account for 11 of the sub-opcode group of codes. In the branch instruction format, bit 4 is used as an indirect bit (indirect if bit 4 = 1). Bits 5 through 9 encode the specific branch instructions. Bits 11 through 15 give a P-relative displacement (0 through 31), and bit 10 specifies whether the displacement is + or - relative to P (0 = +, 1 = -).

2-12. BIT TEST INSTRUCTIONS.

2-13. Bit test instructions are shown in table 2-5. The bit test instructions, also in sub-opcode group 1, use bits 5 through 9 to specify the instruction. Bits 10 through 15 specify a bit position in the TOS word for testing. The bit position specified is modified by the addition of the index register contents if the index bit is set (bit 4 = 1).

2-14. MOVE INSTRUCTIONS.

2-15. Move instructions are shown in table 2-6. The move group of instructions accounts for eight of the codes specified by the sub-opcode 2 code 0000. Sub-opcode group 2 is defined by 0010 in the first four bits. Bits 8, 9, and 10 of the move instruction format encode the specific instruction. Bit 11 is used for some instructions to specify whether the source of the moved data is PB relative (bit 11 = 0) or DB relative (bit 11 = 1). Bit 11 also is used in some cases as an additional code bit for specifying the instruction. Bits 12 and 13 are not used. Bits 14 and 15 are used to specify an S-decrement value to delete, if desired, the move parameters from the top of the stack.

2-16. SPECIAL INSTRUCTIONS.

2-17. Special instructions are shown in table 2-7. The special group uses four mini-opcodes. The mini-opcode group also is, like the moves, specified by the sub-opcode 2 code 0000. Bits 8 through 11, and bit 15, encode the instruction. Bits 12, 13, and 14 are not used.

2-18. IMMEDIATE INSTRUCTIONS.

2-19. Immediate instructions are shown in table 2-8. The immediate instruction group uses codes in both sub-opcode group 2 (coded 0010) and sub-opcode group 3 (coded 0011). Bits 4 through 7 encode the instruction and bits 8 through 15 are used for the immediate operand.

Table 2-1. HP 3000 Computer System Instructions

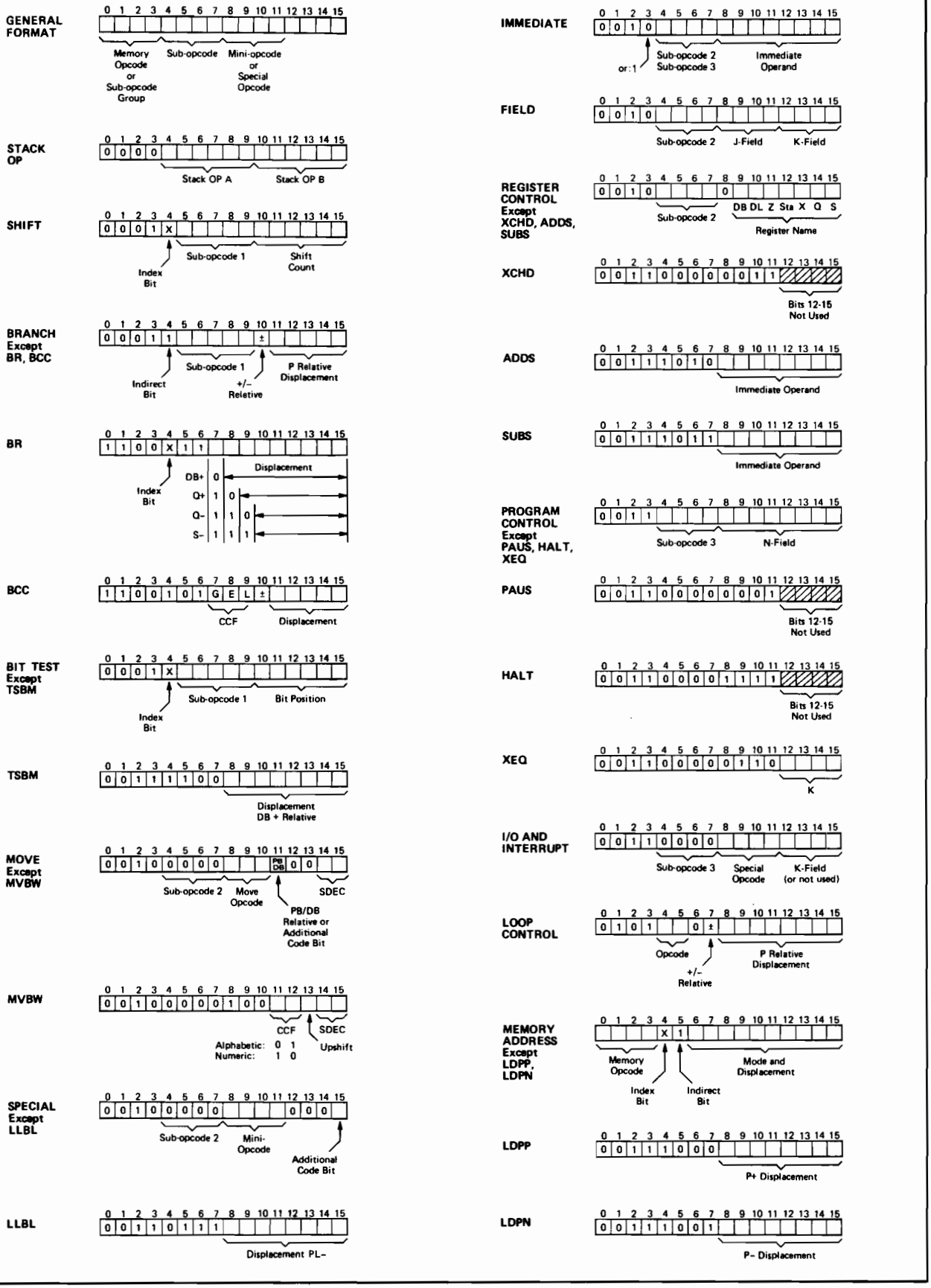
MNEMONIC	DESCRIPTION
ADAX	<Add A to X>
ADBX	<Add B to X>
ADD	<Add>
ADDI	=N <Add immediate>
ADDM	E (*,D,X,PDQS) <Add memory>
ADDS	N <Add to S>
ADXA	<Add X to A>
ADXB	<Add X to B>
ADXI	=N <Add immediate to X>
AND	<And, logical>
ANDI	=N <Logical AND immediate>
ASL	CNT <Arithmetic shift left>
ASR	CNT <Arithmetic shift right>
BCC	L (*,L,P) <Branch on Condition Code>
BCY	L (*,L,P) <Branch on carry>
BNCY	L (*,L,P) <Branch on no carry>
BNOV	L (*,L,P) <Branch on no overflow>
BOV	L (*,L,P) <Branch on overflow>
BR	E (*,D,X,P or indirect DQS) <Branch>
BRE	L (*,L,P) <Branch on TOS even>
BRO	L (*,L,P) <Branch on TOS odd>
BTST	<Test byte on TOS>
CAB	<Rotate ABC>
CIO	K <Control I/O>
CMD	K <Command>
CMP	<Compare>
CMPB	<Compare bytes>
CMPI	=N <Compare immediate>
CMPM	E (*,D,X,PDQS) <Compare memory>
CMPN	=N <Compare negative immediate>
CPRB	L (*,L,P) <Compare range and branch>
CSL	CNT <Circular shift left>
CSR	CNT <Circular shift right>
DABZ	L (*,L,P) <Decrement A, branch if zero>
DADD	<Double add>
DASL	CNT <Double arithmetic shift left>
DASR	CNT <Double arithmetic shift right>
DCMP	<Double compare>
DCSL	CNT <Double circular shift left>
DCSR	CNT <Double circular shift right>
DDEL	<Double delete>
DDUP	<Double duplicate>
DECA	<Decrement A>
DECB	<Decrement B>
DECM	E (*,D,X,DQS) <Decrement memory>
DECX	<Decrement X>
DEL	<Delete A>
DELB	<Delete B>
DFLT	<Double float>
DIV	<Divide>
DIVI	=N <Divide immediate>
DIVL	<Divide Long>
DLSL	CNT <Double logical shift left>
DLSR	CNT <Double logical shift right>
DNEG	<Double negate>
DPF	J,K <Deposit field>
DSUB	<Double subtract>
DTST	<Test double word on TOS>

Table 2-1. HP 3000 Computer System Instructions (Continued)

MNEMONIC	DESCRIPTION
DUP	<Duplicate A>
DXBZ	L (*,L,P) <Decrement X,Branch if zero>
DXCH	<Double exchange>
DZRO	<Double push zero>
EXF	J,K <Extract field>
EXIT	N <Procedure and interrupt exit>
FADD	<Floating add>
FCMP	<Floating compare>
FDIV	<Floating divide>
FIXR	<Fix and round>
FIXT	<Fix and truncate>
FLT	<Float>
FMPY	<Floating multiply>
FNEG	<Floating negate>
FSUB	<Floating subtract>
HALT	K <Halt>
IABZ	L (*,L,P) <Increment A,branch if zero>
INCA	<Increment A>
INCB	<Increment B>
INCM	E (*,D,X,DQS) <Increment memory>
INCX	<Increment index>
IXBZ	L (*,L,P) <Increment X,Branch if zero>
LADD	<Logical add>
LCMP	<Logical compare>
LDB	E (*,D,X,DQS) <Load byte>
LDD	E (*,D,X,DQS) <Load double>
LDI	=N <Load immediate>
LDIV	<Logical divide>
LDNI	=N <Load negative immediate>
LDPN	N <Load double from program, negative>
LDPP	N <Load double from program, positive>
LDX	E (*,D,X,PDQS) <Load Index>
LDXA	<Load X onto stack>
LDXB	<Load X into B>
LDXI	=N <Load X immediate>
LDXN	=N <Load X negative immediate>
LLBL	N <Load Label>
LLSH	<Linked list search>
LMPY	<Logical multiply>
LOAD	E (*,D,X,PDQS) <Load>
LRA	E (*,D,X,PDQS) <Load relative address>
LSL	CNT <Logical shift left>
LSR	CNT <Logical shift right>
LSUB	<Logical subtract>
MOVE	<Move words>
MPY	/<Multiply>
MPYI	=N <Multiply immediate>
MPYL	<Multiply Long>
MPYM	E (*,D,X,PDQS) <Multiply memory>
MTBA	E (D,P) <Modify, Test,Branch,A>
MTBX	E (D,P) <Modify, Test,Branch,X>
MVB	<Move bytes>
MVBL	<Move from DB+ to DL+>
MVBW	<Move bytes while>
MVLB	<Move from DL+ to DB+>

Table 2-1. HP 3000 Computer System Instructions (Continued)

MNEMONIC	DESCRIPTION
NEG	<Negate>
NOP	<No operation>
NOT	<One's complement>
OR	<Or, logical>
ORI	=N <Logical OR immediate>
PAUS	K <Pause>
PCAL	N <Procedure call>
PLDA	<Privileged load from absolute address>
PSHR	N <Push registers>
PSTA	<Privileged store into absolute address>
RIO	K <Read I/O>
RMSK	K <Read Mask>
RSW	<Read Switch register>
SBXI	=N <Subtract immediate from X>
SCAL	N <Subroutine Call>
SCAN	<Scan bits>
SCU	<Scan until>
SCW	<Scan while>
SED	K <Set enable/disable external interrupts>
SETR	N <Set registers>
SIN	K <Set interrupt>
SIO	K <Start I/O>
SIRF	K <Set external interrupt reference flag>
SMSK	<Set Mask>
STAX	<Store A into X>
STB	E (*,D,X,DQS) <Store byte>
STBX	<Store B into X>
STD	E((*,D,X,DQS) <Store double>
STOR	E (*,D,X,DQS) <Store>
SUB	<Subtract>
SUBI	=N <Subtract immediate>
SUBM	E (*,D,X,PDQS) <Subtract memory>
SUBS	N <Subtract from S>
SXIT	N <Subroutine exit>
TASL	CNT <Triple arithmetic shift left>
TASR	CNT <Triple arithmetic shift right>
TBA	E (D,P) <Test,branch,A>
TBC	CNT <Test bit and set condition code>
TBX	E (D,P) <Test,branch,X>
TCBC	CNT <Test and complement bit and set CC>
TEST	<Test TOS>
TIO	K <Test I/O>
TNSL	<Triple normalizing shift left>
TRBC	CNT <Test and reset bit, set condition code>
TSBC	CNT <Test,set bit,set condition code>
TSBM	N <Test and set bit in memory>
WIO	K <Write I/O>
XAX	<Exchange A and X>
XBX	<Exchange B and X>
XCH	<Exchange A and B>
XCHD	K <Exchange DB>
XEQ	K <Execute>
XOR	<Exclusive or, logical>
XORI	=N <Logical Exclusive OR immediate>
ZERO	<Push zero>
ZROB	<Zero B>
ZROX	<Zero X>



2184-137

Figure 2-1. Instruction Formats

Table 2-2. Stack Op Instructions

MNEMONIC	NAME	MNEMONIC	NAME
NOP	No Operation	XCH	Exchange A and B
DELB	Delete B	XAX	Exchange A and X
DDEL	Double Delete	CAB	Rotate A, B, C
DEL	Delete A	XBX	Interchange Second Word of Stack With the Content of the Index Register
DUP	Duplicate A		
DDUP	Double Duplicate	STBX	Store B into X
ZROX	Zero X	ADAX	Add A to X
ZERO	Push Zero	ADXA	Add X to A
DZRO	Push Double Zero	LDXB	Load X into B
ZROB	Zero B	STAX	Store A into X
INCX	Increment X	LDXA	Load X onto Stack
DECX	Decrement X	ADBX	Add B to X
INCA	Increment A	ADXB	Add X to B
DECA	Decrement A	DFLT	Double Float
INCB	Increment B	FLT	Float
DECB	Decrement B	FCMP	Floating Compare
DCMP	Double Compare	FADD	Floating Add
DADD	Double Add	FSUB	Floating Subtract
DSUB	Double Subtract	FMPY	Floating Multiply
MPYL	Multiply Long	FDIV	Floating Divide
DIVL	Divide Long	FNEG	Floating Negate
DNEG	Double Negate	FIXR	Fix and Round
CMP	Compare	FIXT	Fix and Truncate
ADD	Add	LCMP	Logical Compare
SUB	Subtract	LADD	Logical Add
MPY	Multiply	LSUB	Logical Subtract
DIV	Divide	LMPY	Logical Multiply
NEG	Negate	LDIV	Logical Divide
TEST	Test TOS	NOT	One's Complement
DTST	Test Double Word on TOS	OR	Logical OR
BTST	Test Byte on TOS	XOR	Logical Exclusive OR
DXCH	Double Exchange	AND	Logical AND

Table 2-3. Shift Instructions

MNEMONIC	NAME
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
LSL	Logical Shift Left
LSR	Logical Shift Right
CSL	Circular Shift Left
CSR	Circular Shift Right
DASL	Double Arithmetic Shift Left
DASR	Double Arithmetic Shift Right
DLSL	Double Logical Shift Left
DLSR	Double Logical Shift Right
DCSL	Double Circular Shift Left
DCSR	Double Circular Shift Right
TASL	Triple Arithmetic Shift Left
TASR	Triple Arithmetic Shift Right
TNSL	Triple Normalizing Shift Left

Table 2-4. Branch Instructions

MNEMONIC	NAME
IABZ	Increment A, Branch if Zero
IXBZ	Increment X, Branch if Zero
DXBZ	Decrement X, Branch if Zero
BCY	Branch On Carry
BNCY	Branch on No Carry
CPRB	Compare Range and Branch
DABZ	Decrement A, Branch if Zero
BOV	Branch On Overflow
BNOV	Branch On No Overflow
BRO	Branch on TOS Odd
BRE	Branch on TOS Even
BR	Branch Unconditionally
BCC	Branch on Condition Code

Table 2-5. Bit Test Instructions

MNEMONIC	NAME
SCAN	Scan Bits
TBC	Test Bit and Set Condition Code
TRBC	Test and Reset Bit, Set Condition Code
TSBC	Test and Set Bit, Set Condition Code
TCBC	Test and Complement Bit, Set Condition Code
TSBM	Test and Set Bits in Memory, Set Condition Code

Table 2-6. Move Instructions

MNEMONIC	NAME
MOVE	Move Words
MVB	Move Bytes
MVBL	Move Words from DB+ to DL+
SCW	Scan While Memory Bytes Equal Test Byte
MVLB	Move Words from DL+ to DB+
SCU	Scan Until Memory Byte Equals Test Byte or Terminal Byte
MVBW	Move Bytes While of Specified Type
CMPB	Compare Bytes

Table 2-7. Special Instructions

MNEMONIC	NAME
RSW	Read Switch Register
LLSH	Linked List Search
PLDA	Privileged Load from Absolute Address
PSTA	Privileged Store into Absolute Address
LLBL	Load Label

Table 2-8. Immediate Instructions

MNEMONIC	NAME
LDI	Load Immediate
LDXI	Load X Immediate
CMPI	Compare Immediate
ADDI	Add Immediate
SUBI	Subtract Immediate
MPYI	Multiply Immediate
DIVI	Divide Immediate
LDNI	Load Negative Immediate
LDXN	Load X Negative Immediate
CMPN	Compare Negative Immediate
ADXI	Add Immediate to X
SBXI	Subtract Immediate from X
ORI	Logical OR Immediate
XORI	Logical Exclusive OR Immediate
ANDI	Logical AND Immediate

2-20. FIELD INSTRUCTIONS.

2-21. Field instructions are shown in table 2-9. The format for deposit field and extract field instructions is specified by two of the sub-opcode 2 group of codes. Bits 4 through 7 specify the instruction and the remaining eight bits are divided into a J-field and a K-field. The J-field specifies the starting bit number and the K-field specifies the number of bits.

2-22. REGISTER CONTROL INSTRUCTIONS.

2-23. Register control instructions are shown in table 2-10. The format for the register control instructions uses bits 9 through 15 to name a register and bits 4 through 7 in sub-opcode group 2 to specify the operation.

2-24. PROGRAM CONTROL INSTRUCTIONS.

2-25. Program control instructions are shown in table 2-11. The program control instructions account for four of the sub-opcode 3 codes. Sub-opcode 3 is specified by 0011 in the first four bits. The instruction is encoded by bits 4 through 7, and the N-field in bits 8 through 15 is used either for a PL displacement (PCAL and SCAL) or to specify a number of parameters to be deleted on return from a procedure or subroutine (EXIT and SXIT).

2-26. I/O AND INTERRUPT INSTRUCTIONS.

2-27. I/O and interrupt instructions are shown in table 2-12. The I/O and interrupt instructions use 11 of the special opcodes (bits 8 through 11) defined by the sub-opcode 3 code of 0000. The K-field, bits 12 through 15, is used by some of the instructions for an S-displacement to locate a device number given in the stack.

2-28. LOOP CONTROL INSTRUCTIONS.

2-29. Loop control instructions are shown in table 2-13. The loop control instructions are defined by a special coding of bits 4, 5, and 6 for memory opcode 05 (which is otherwise defined as the STOR instruction). Bits 8 through 15 give a P-relative displacement for + (0) or - (1) relative to P.

2-30. MEMORY ADDRESS INSTRUCTIONS.

2-31. Memory address instructions are shown in table 2-14. The memory address instruction format uses bits 0, 1, 2, and 3 to encode a specific instruction. Bits 6 through 15 give both an addressing mode and a displacement. Bit 5 is used to specify indirect addressing (1), if desired. If both indirect addressing and indexing are specified, post indexing will occur.

Table 2-9. Field Instructions

MNEMONIC	NAME
EXF	Extract Field
DPF	Deposit Field

2-32. MICROINSTRUCTION CODING.

2-33. A consolidated microinstruction coding diagram listing the field codes and the octal coding required to produce the codes is shown in figure 2-2. The physical location of the fields in the 32-bit microinstruction word is shown also.

Table 2-10. Register Control Instructions

MNEMONIC	NAME
PSHR	Push Registers
SETR	Set Registers
XCHD	Exchange DB and TOS
ADDS	Add to S
SUBS	Subtract from S

Table 2-11. Program Control Instructions

MNEMONIC	NAME
PAUS	Pause
XEQ	Execute Stack Word
HALT	Computer Hardware Halts
SCAL	Subroutine Call
PCAL	Procedure Call
EXIT	Exit from Procedure
SXIT	Exit from Subroutine

Table 2-12. I/O and Interrupt Instructions

MNEMONIC	NAME
SED	Set "Enable/Disable External Interrupts"
SMSK	Set Mask
RMSK	Read Mask
SIO	Start I/O
RIO	Read I/O
WIO	Write I/O
TIO	Test I/O
CIO	Control I/O
CMD	Command (Central Data Bus)
SIRF	Set External Interrupt Reference Flag
SIN	Set Interrupt

Table 2-13. Loop Control Instructions

MNEMONIC	NAME
TBA	Test and Branch, Limit in A
MTBA	Modify Variable, Test and Branch, Limit in A
TBX	Test and Branch, Variable in X
MTBX	Modify Variable in x, Test and Branch

OCTAL CODE		S-BUS 5 BITS	STORE 5 BITS	FUNCTION 5 BITS	SKIP 5 BITS	SHIFT 3 BITS	SPECIAL 5 BITS	MCU 5 BITS	R-BUS 4 BITS
0	0	CIR	MASK	TASL	ZERO	LRZ	CCB		PL
0	1	SP1	IOA	TASR	NZRO	LLZ	CCPX		SR
0	2	PADD	IOD	ROMX	EVEN	SL1	SR0		Z
0	3		MREG	ROMN	ODD	SR1	HALT		MREG
0	4	CPX1	BUSL*	JSB	NSME	RRZ	SIFG		PADD
0	5	MOD	BSP0*	CAND	BIT 6	RLZ	SDFG		R BUS
0	6	CPX2	BUSH*	XOR	BIT 8	ROT	CTF		X
0	7	SWCH	DATA	AND	NOFL	NOP	CF3		XC
1	0	ODWN	PUSH	DVSB	CRRY		INSR		RD
1	1	IOA	PL	UBNT**	NCRY		DCSR		RC
1	2	IOD	Z	CADO	POS		INCN		RB
1	3	MASK	QUP	SUBO	NEG		INCT		RA
1	4	CTRL	SP1	JMP	F1		HBF		SP1
1	5	CTRH	SP0	BNDT**	NF1		FHB		SP0
1	6	UBUS	CTRL	CAD	F2		CLIB		UBUS
1	7	SBUS	CTRH	SUB	NF2		LBF		NOP
2	0	P	P	PNLR†	SRZ		SF2	RWAN	
2	1	Q	Q	PNLS†	SRNZ		CF2	RWAN	
2	2	DB	SM	ROMI	SR4		CF1	RWAN	
2	3	SM	DB	ROM†	SRN4		SF1	RWAN	
2	4	STA	STA	REPC†	INDR		SCRY	RWAN	
2	5	SP3	SP3	REPN†	SRL2		CCRY	RWAN	
2	6	OPND	X	IOR	NPRY		POPA	RWAN	
2	7	CC	RAR	CTSD†	SRL3		POP	RWAN	
3	0	RD	RD	MPAD†	RSB		SOV	CMD	
3	1	RC	RC	INCO†	JLUI		CLO	CRL	
3	2	RB	RB	CRS†	TEST		CCZ	NIR	
3	3	RA	RA	ADDO†	CTRM		CCL	RWN	
3	4	DL	DL	CTSS†	F3		CCG	OPND	
3	5	SP2	SP2	INC†	NEXT		CCE	RNWA	
3	6	PB	PB	RPTY†	UNC		CCA	CWA	
3	7	NOP	NOP	ADD†	NOP		NOP	RWA	

JMP, JSB — 2 CYCLE

*THESE STORE OPTIONS "NOP" THE SPECIAL FIELD FUNCTIONS AND ALLOW THE SPECIAL FIELD TO BE USED FOR MCU OPTIONS.
†THESE FUNCTIONS CAUSE AN "ADD."
**TWO-CYCLE INSTRUCTION.

FIELD LOCATIONS

0 1	2 3 4	5 6 7	8 9 10	11 12 13	14 15 16	17 18 19	20 21 22	23 24 25	26 27 28	29 30 31
S-BUS		STORE		FUNCTION	SKIP		SHIFT	SPECIAL		R-BUS
				JMP, JSB			JUMP TARGET			
				ANY ROM	⊠	ROM CONSTANT				
				REPN	COUNT		SHIFT	SPECIAL		R-BUS
		BUSL BSPO BUSH			SKIP			MCU OPTION		

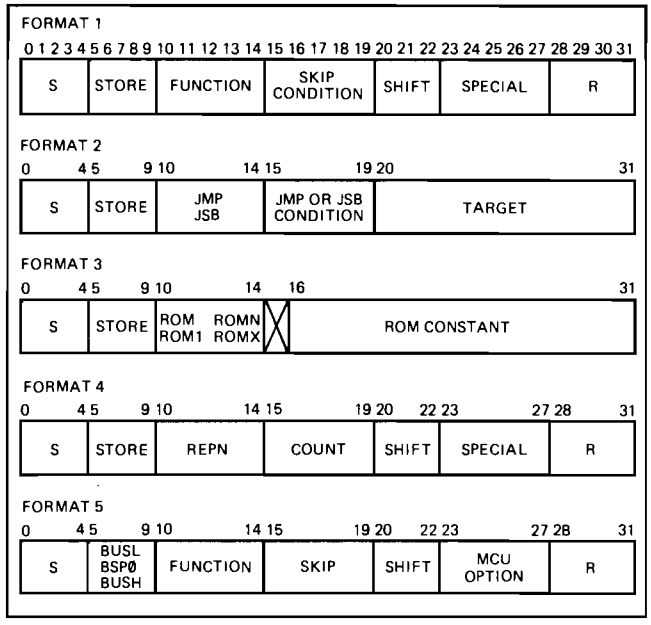
2184-91

Figure 2-2. Consolidated Microinstruction Coding Diagram

Table 2-14. Memory Address Instructions

MNEMONIC	NAME
LDPP	Load Double from Program, Positive
LDPN	Load Double from Program, Negative
LOAD	Load Word onto TOS
STOR	Store TOS into Memory
CMPM	Compare TOS with Memory
ADDM	Add Memory to TOS
SUBM	Subtract Memory from TOS
MPYM	Multiply TOS by Memory
INCM	Increment Memory
DECM	Decrement Memory
LDX	Load Index
LDB	Load Byte
LDD	Load Double
STB	Store Byte
STD	Store Double
LRA	Load Relative Address

2-34. A microinstruction word from the ROM will be in one of five formats, depending upon the code specified in the function or store fields. The formats are shown in figure 2-3.



2184-85

Figure 2-3. Microinstruction Word Formats

2-35. FORMAT 1.

2-36. An operational flow diagram for a format 1 type of microinstruction is shown in figure 2-4 and a timing diagram is shown in figure 2-5. At the start, a format 1 microinstruction (I) is contained in ROM output register rank 1 (ROR1). The ROM address register (RAR) contains the address of the next succeeding microinstruction (I+1) and ROM output register rank 2 (ROR2) contains the preceding microinstruction, or I-1. The R-bus and S-bus fields, which bypass ROR2, are executed first, allowing the R- and S-bus registers to be loaded before executing the remainder of the microinstruction. The function field is then executed and, after execution, the T-bus contains the contents of the R- and S-bus, depending upon the function code. The shift field, when decoded, determines whether the data on the T-bus will be shifted, and transfers the T-bus data to the U-bus. The skip field is examined for a skip condition and, if the condition is met, ROR1 is NOPed for one clock cycle before the store and special fields are executed. When the store and special fields are executed, the condition of RAR, ROR1, and ROR2 is as follows:

a. RAR = I + 3

b. ROR1 = I + 2

c. ROR2 = I + 1

2-37. FORMAT 2.

2-38. An operational flow diagram for microinstruction word format 2 is shown in figure 2-6 and timing diagrams are shown in figures 2-7 and 2-8. When a Jump (JMP or JSB) is specified in the function field, bits 20 through 31 of the microinstruction word are used to specify the jump address instead of being used for shift, special, or R-bus field operations. A JMP function code directs a micro-jump to the target address in the ROM if the skip field condition is met. A JSB function code directs a micro-subroutine jump to the target address in ROM if the skip field condition is met. In both cases (JMP and JSB), if the skip field code is ZERO, NZRO, EVEN, ODD, NSME BIT 6, BIT8, NOFL, CRRY, NCRY, POS, or NEG (eg, any skip condition that is data (U-bus) dependent) a one-cycle freeze is generated to allow the U-bus data to be established and checked for the jump condition.

2-39. FORMAT 3.

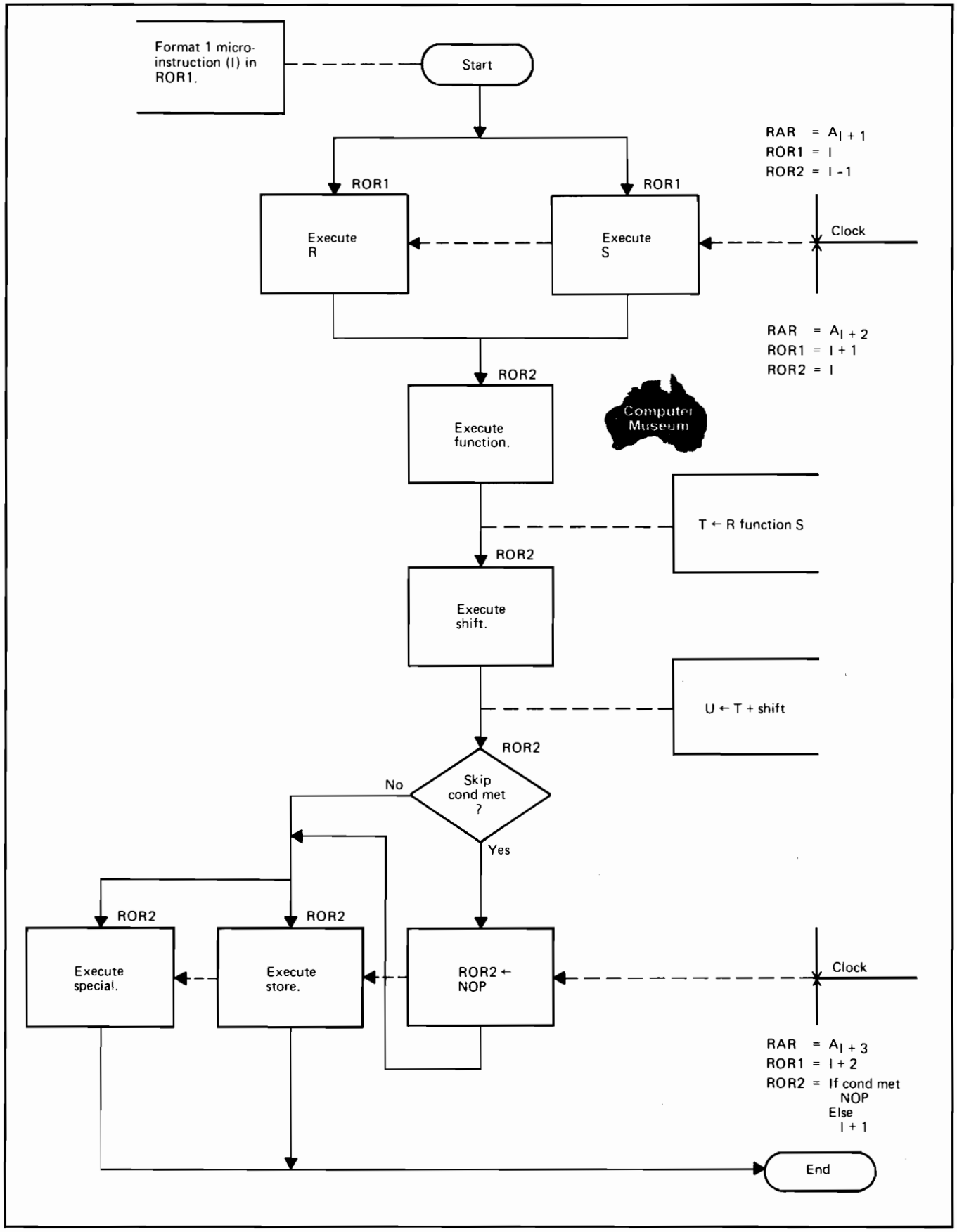
2-40. An operational flow diagram for a format 3 microinstruction word is shown in figure 2-9. When a ROM, ROMI, ROMN, or ROMX function code is specified in a microinstruction, the ROR1 function decoder (see simplified diagram, figure 3-22, sheet 13) disables the skip, shift, special, and R-bus fields and these bits (16:31) are loaded directly from ROR1 into the R-bus register. When the microinstruction is loaded into ROR2, the ALU function decoder (which decodes the output of the ROR2 function field) generates the following ALUS (0:3) codes, depending upon the type of ROM function:

ROM: ALUS (0:3) = 1001 (ADD)

ROMI: ALUS (0:3) = 1000 (IOR)

ROMN: ALUS (0:3) = 1101 (AND)

ROMX: ALUS (0:3) = 0110 (XOR)



2184-86

Figure 2-4. Microinstruction Word Format 1 Flow Diagram

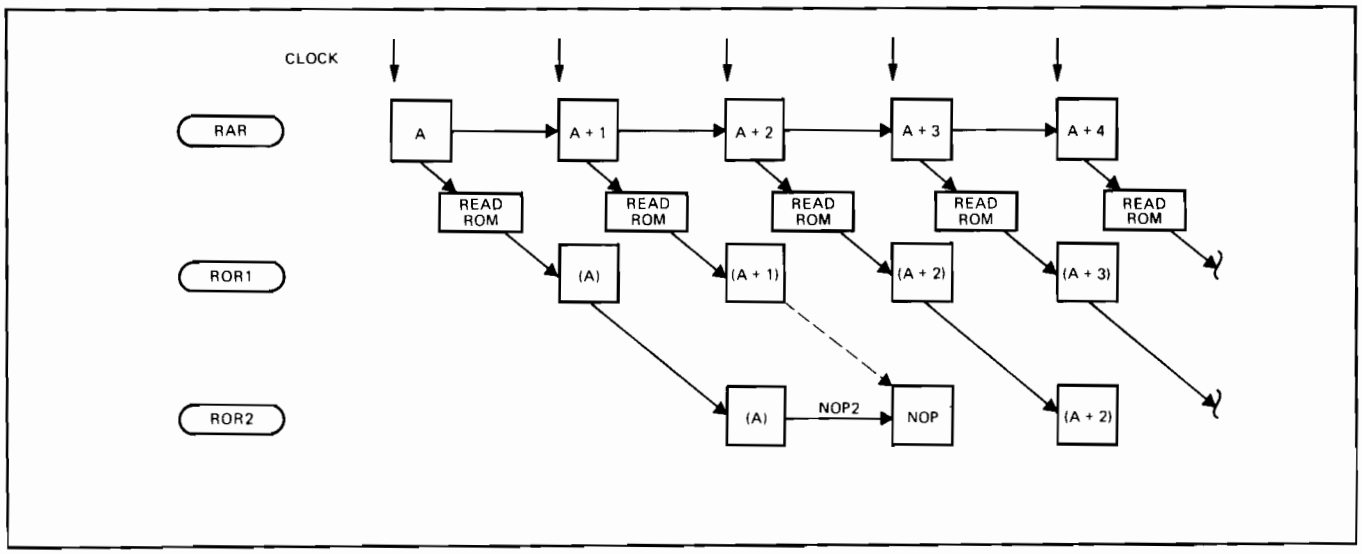
2-41. The above functions, ADD, IOR, AND, X and XOR, are then executed and the U-bus contains the T-bus content, which is the content of the R-bus register added, "anded," or "ored", to the content of the S-bus register. See table 3-3 for a description of operation for the ROM, ROMI, ROMN, and ROMX codes. When the store field code is executed, the content of the U-bus is stored in one of the CPU registers.

2-42. FORMAT 4.

2-43. A format 4 operational flow diagram is shown in figure 2-10. In microinstruction word format 4, the REPN function field code causes the next microinstruction to be executed repeatedly. The count for the number of repeats to be performed is contained in the skip field (bits 15:19). These bits are loaded into a repeat counter register. To utilize the counter, the microinstruction word contains an INCTR (Increment Counter) code in the special field and a CTRM (Counter Maximum) code in the skip field.

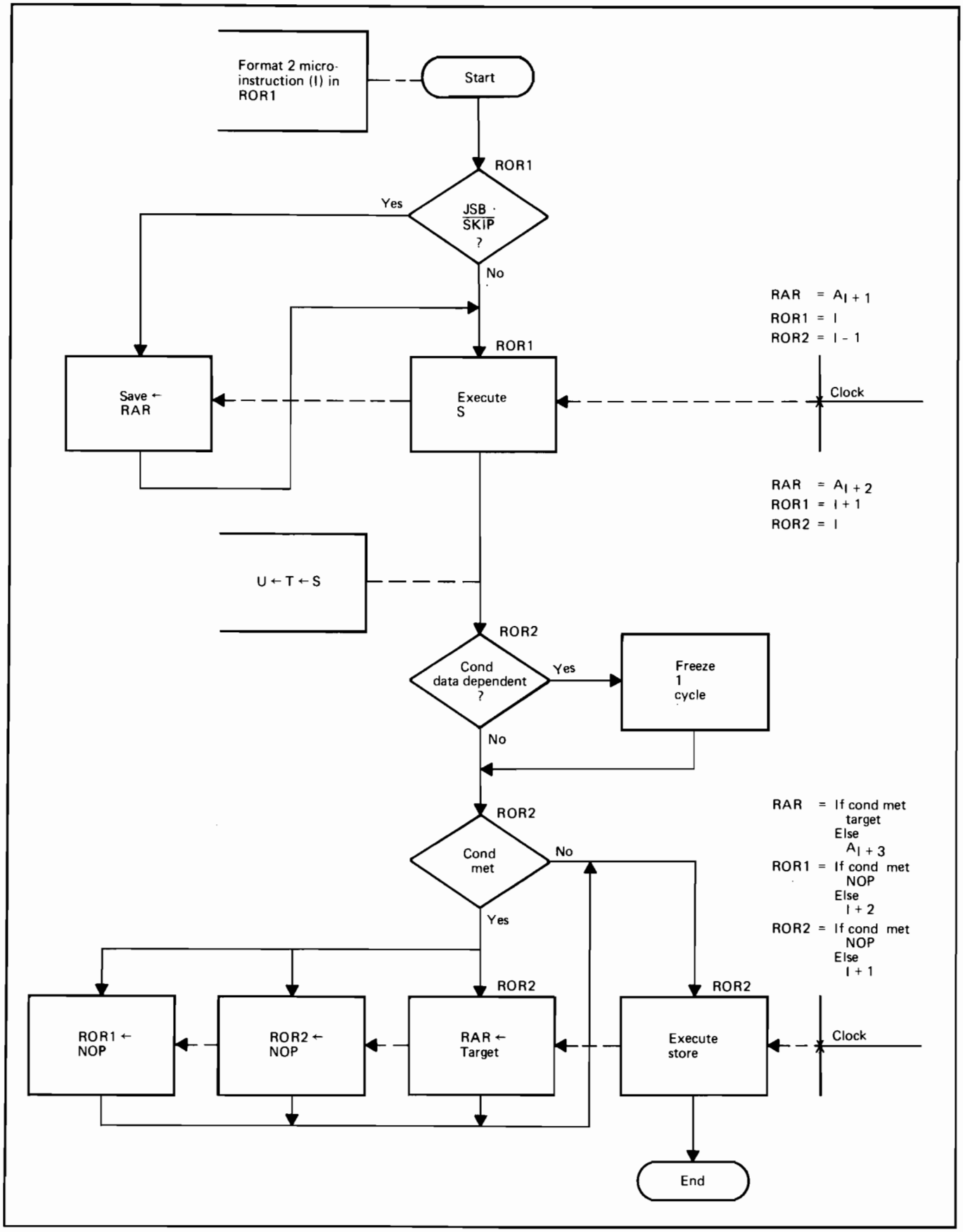
2-44. FORMAT 5.

2-45. A format 5 microinstruction word is used to specify an MCU option operation. An operational flow diagram for format 5 is shown in figure 2-11. When a BUSL, BSPO, or BUSH store field code is decoded, the special field decoder is disabled and the special field bits (bits 23:27) are used by the MCU operation decoder to specify an MCU option. The BUSL and BSPO store field codes are used to initiate MCU memory operations. The store field code BUSH allows use of the central data bus to transfer data from the CPU output register (COR) to the operand, next instruction, or command registers.



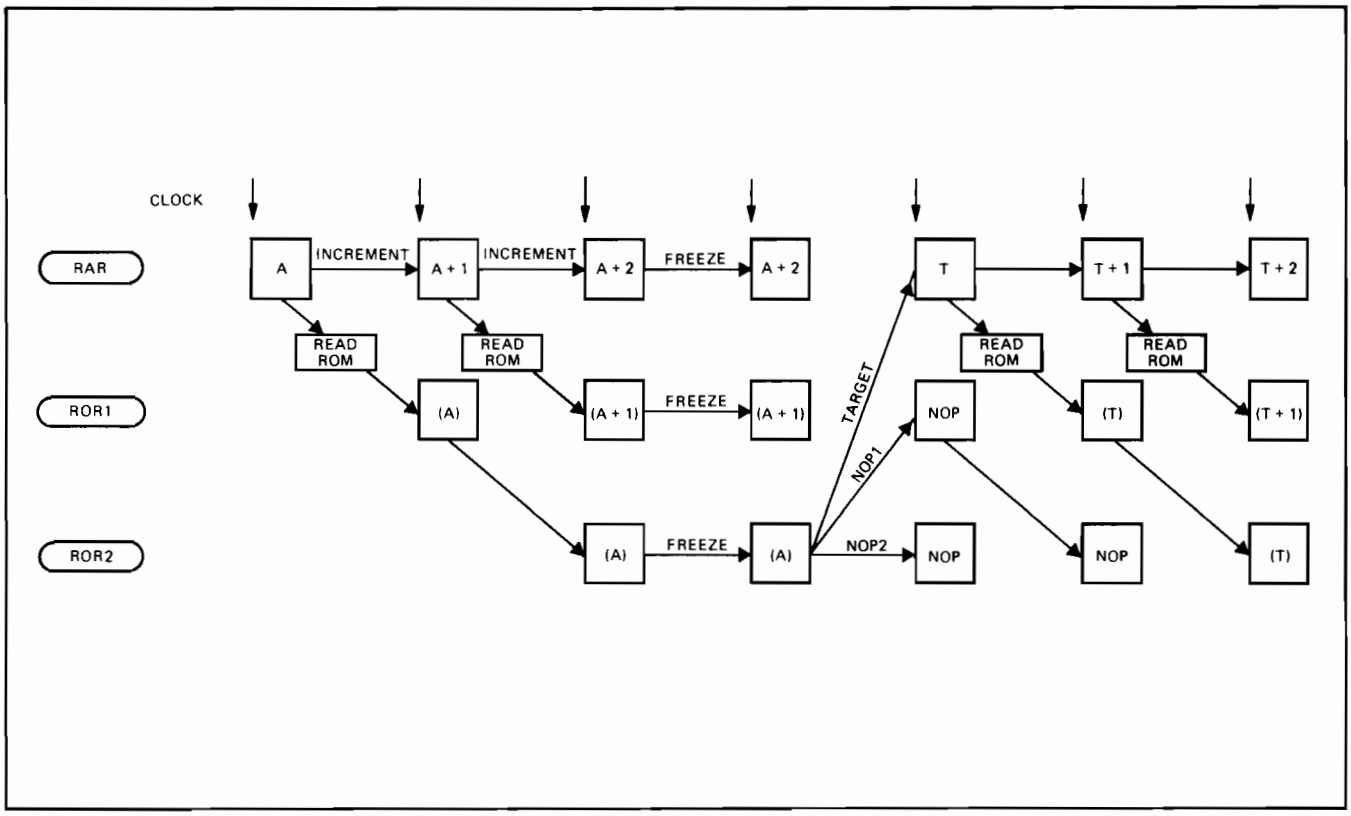
2184-94

Figure 2-5. Execution of Microinstruction Containing Skip Condition (Condition Met)



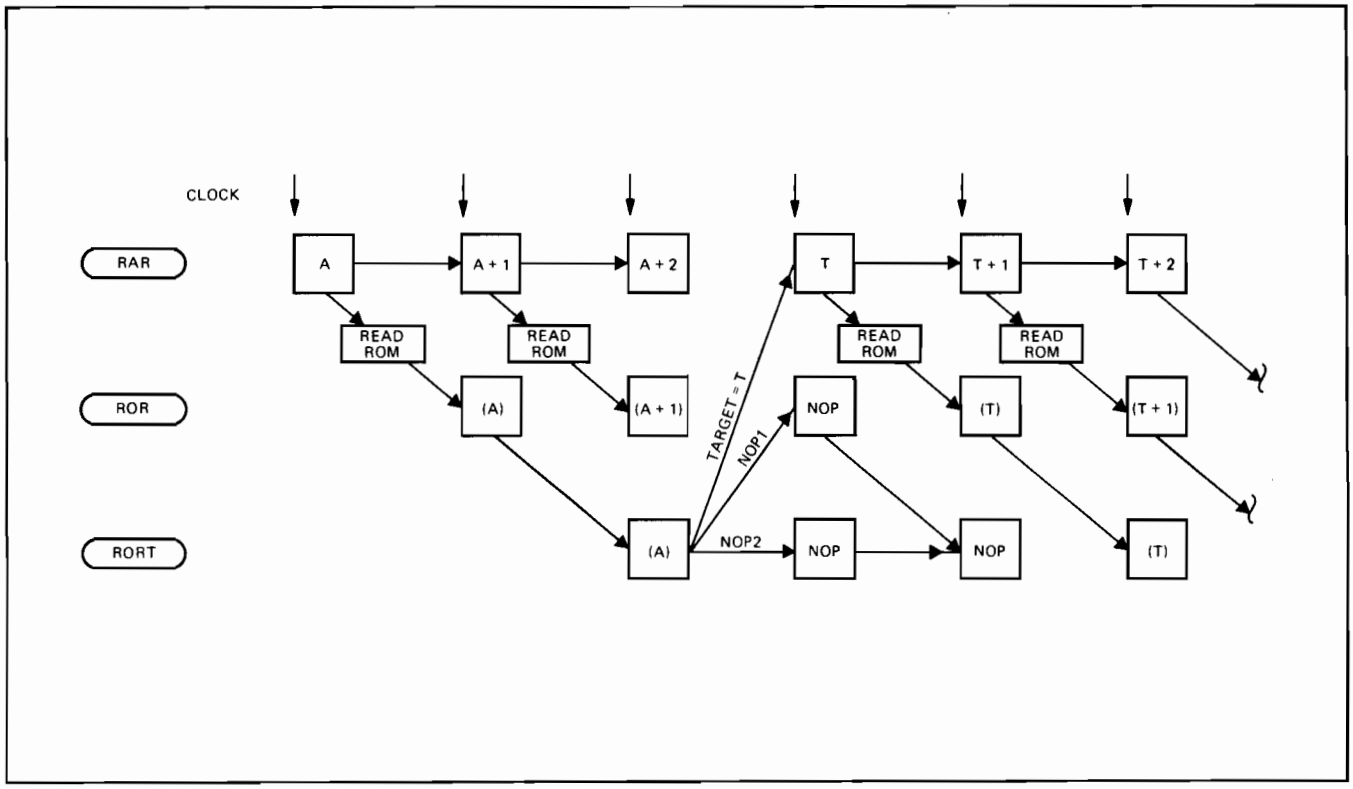
2184-87

Figure 2-6. Microinstruction Word Format 2 Flow Diagram



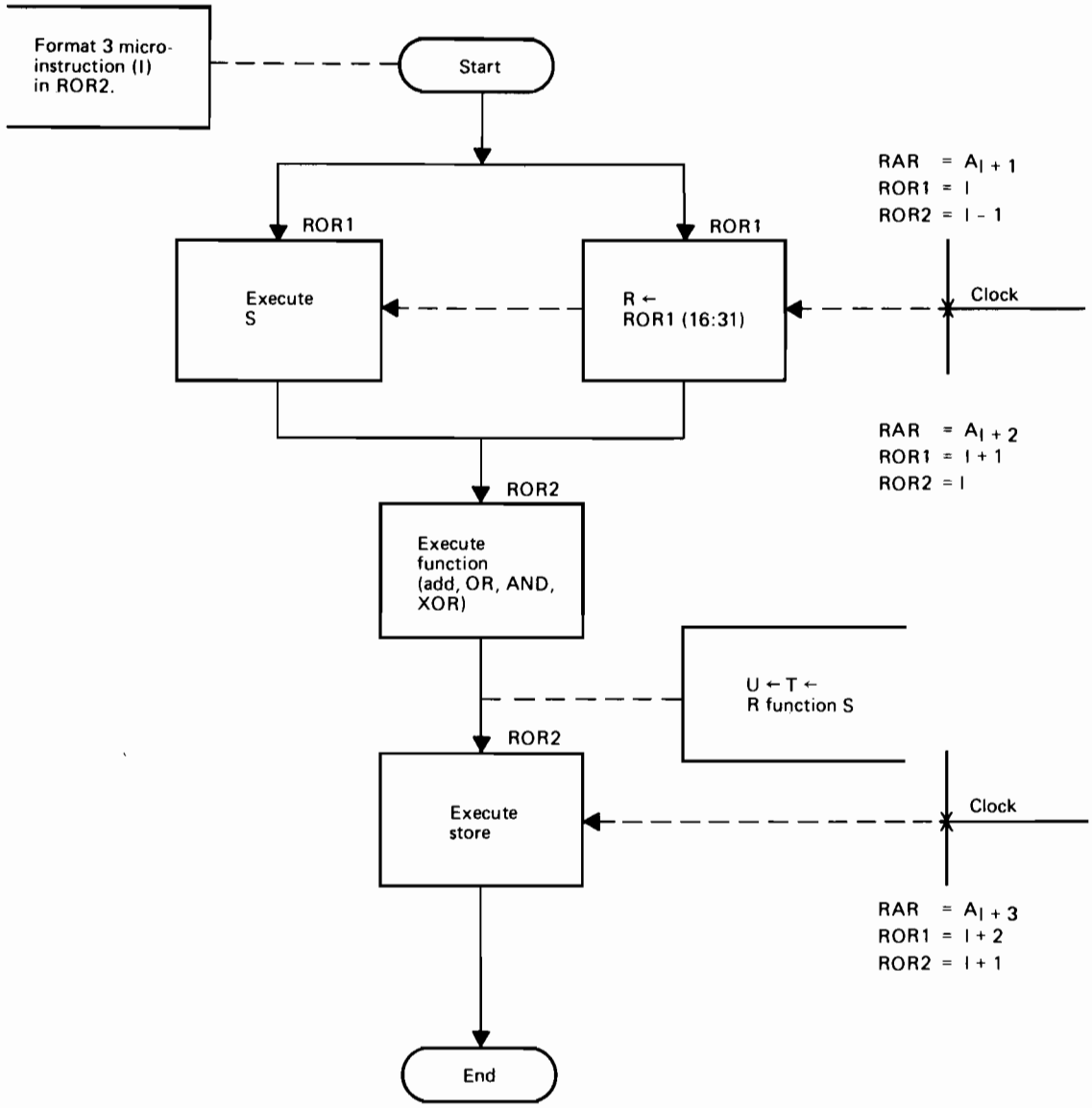
2184-93

Figure 2-7. Execution of Microinstruction Containing Data Dependent JMP or JSB (Condition Met) to Target Address T



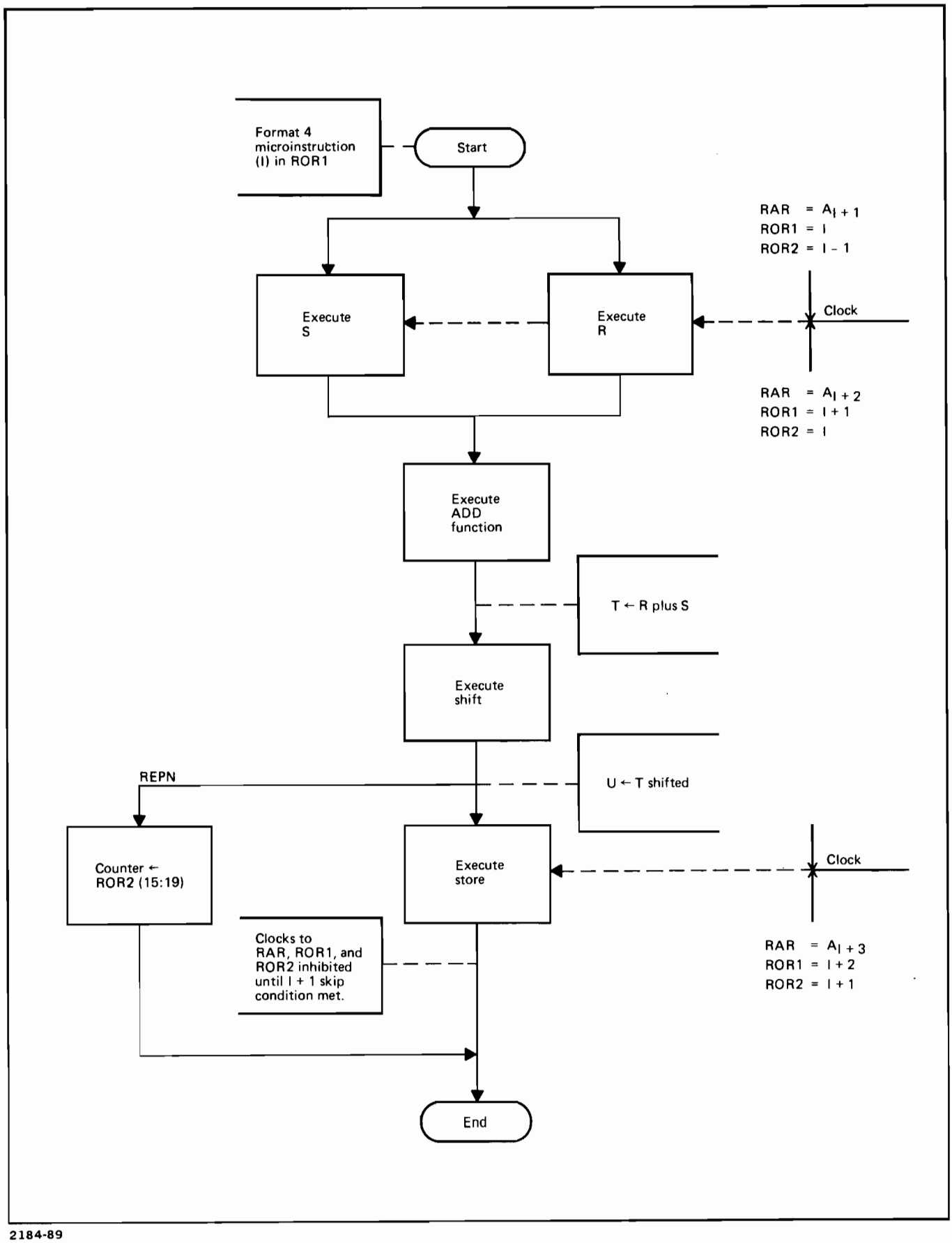
2184-95

Figure 2-8. Execution of Microinstruction Containing JMP or JSB (Condition Met)



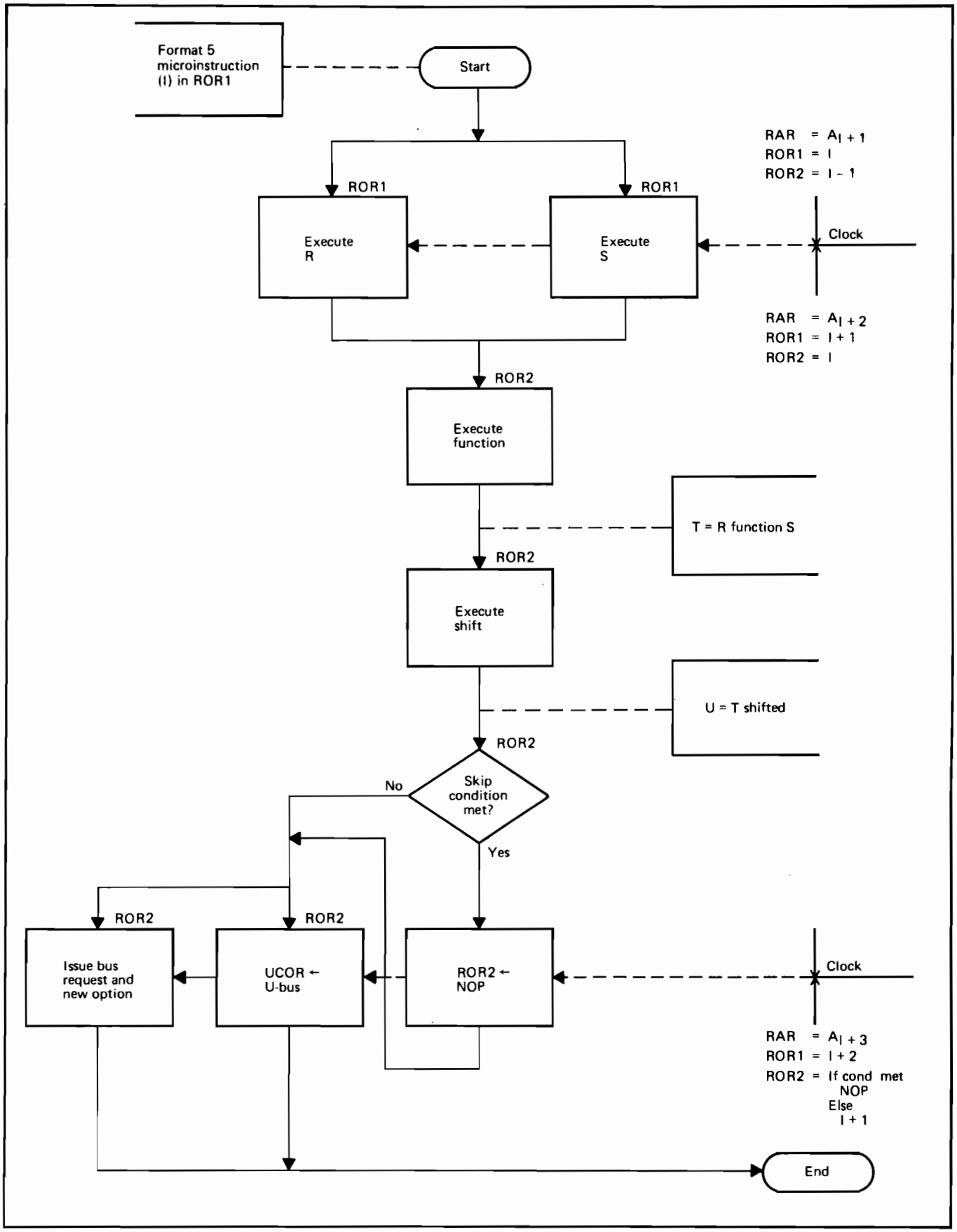
2184-88

Figure 2-9. Microinstruction Word Format 3 Flow Diagram



2184-89

Figure 2-10. Microinstruction Word Format 4 Flow Diagram



2184-90

Figure 2-11. Microinstruction Word Format 5 Flow Diagram

SECTION III THEORY OF OPERATION

3-1. INTRODUCTION.

3-2. This section contains a system-level description of operation for the HP 3000 Computer System and a block-level and functional-level description of operation for the CPU/IOP. The system-level description provides a brief description of the CPU/IOP operation in relation to other components of the HP 3000 Computer System. The block-level theory of operation divides the CPU/IOP into circuit groups and the operation of each circuit group is described, referencing a block diagram of the CPU/IOP and simplified and detailed logic diagrams of the circuit groups. The functional-level theory of operation describes the operation of the CPU/IOP during its normal operation (address/data transfers, etc). Operation of both direct and programmed (multiplexed) modes is described, using operational flow diagrams to supplement the text. All signals are referred to by their mnemonic. Refer to table 4-1 for a listing of all signal names and mnemonics.

3-3. SYSTEM-LEVEL DESCRIPTION.

3-4. A block diagram of the HP 3000 Computer System is shown in figure 3-1. The diagram does not show a complete system but is a typical system containing a multiplexer channel and I/O interface and device (I/O subsystem). Direct I/O devices are not shown on the diagram, nor are additional multiplexer channels and their associated I/O subsystems. The following components, therefore, comprise the typical system depicted:

a. Memory subsystem.

b. Central processor unit (CPU).

c. Input/output processor (IOP).

d. Multiplexer channel.

e. I/O subsystem (interface PCA and I/O device), such as line printer or tape reader.

3-5. The memory subsystem contains the I/O drivers that are executed by the CPU, I/O programs that are transferred by the IOP to the multiplexer channel, and the device reference tables (DRT). The I/O drivers contain I/O instructions such as Read I/O (RIO), Write I/O (WIO), Start I/O (SIO), Test I/O (TIO), and Control I/O (CIO). The DRT begins in memory location octal 14 and contains a maximum of 253 four-word entries. Each table entry corresponds to a unique I/O device number. The first word of each entry contains the address of the next I/O program instruction for the device.

3-6. The central processor unit is divided into three major sections: central processor unit (CPU), I/O processor (IOP), and module control unit (MCU). The MCU is shared by the CPU and IOP. When the CPU executes an I/O instruction (TIO, CIO, RIO, WIO, or SIO), direct commands are sent through the IOP and IOP bus to the addressed I/O subsystem. When the I/O subsystem accepts a direct command, it returns an acknowledge signal and executes the command.

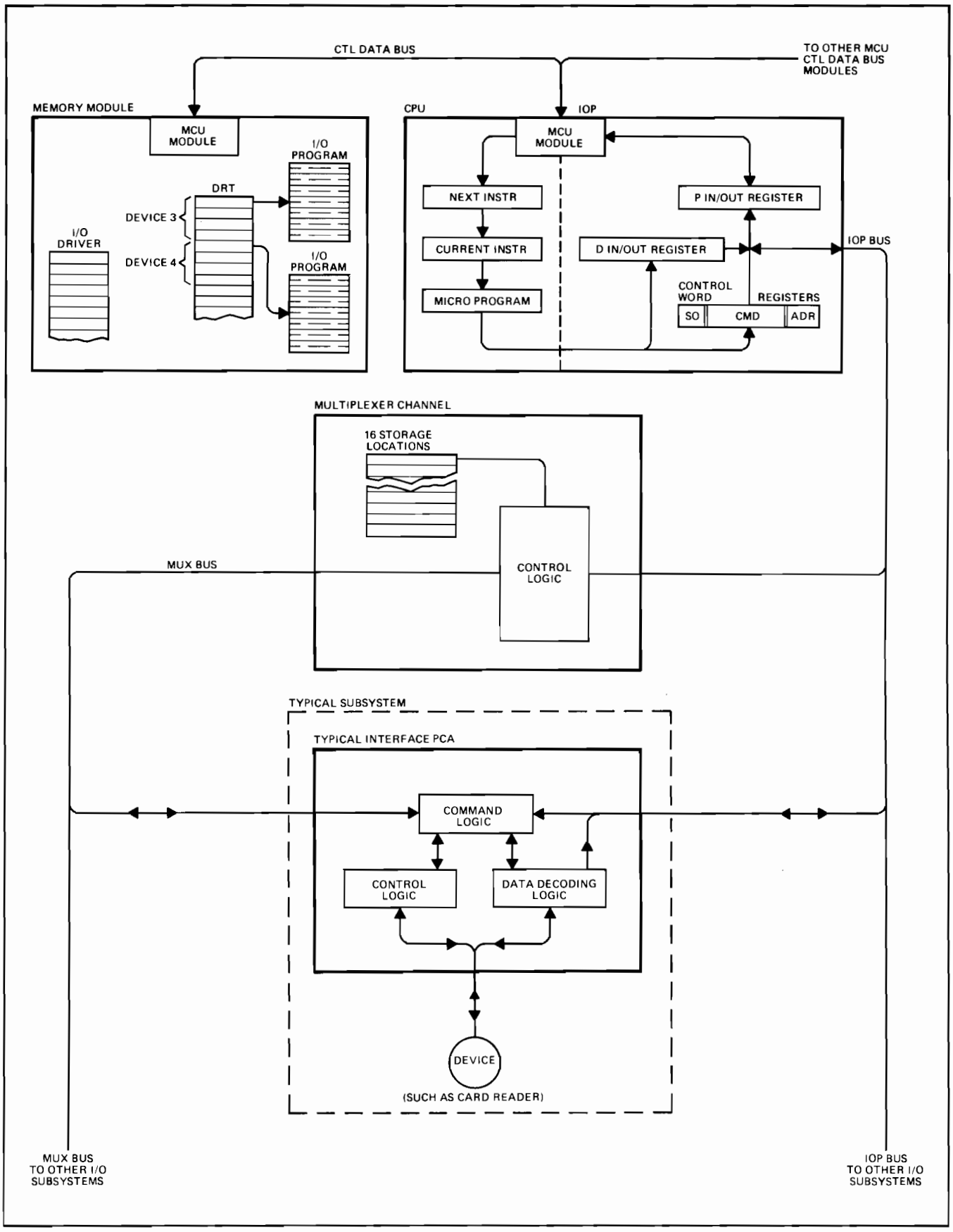
3-7. If the CPU executes an SIO instruction, the IOP, in conjunction with the multiplexer channel, assumes control of the I/O subsystem and the CPU is free to continue processing other functions. The IOP transfers an I/O program, one instruction double word at a time, from the memory to the multiplexer channel. The multiplexer channel then controls the operation of the I/O subsystem.

3-8. BLOCK-LEVEL DESCRIPTION.

3-9. The block-level description divides the CPU/IOP (and MCU) into circuit groups and provides a description of operation for each circuit. Each description references a simplified logic diagram and a detailed logic diagram of the circuit. In addition, a drawing grid number is listed, showing the location of the circuit on the detailed diagrams (set nos. DD200 through DD209).

Note: Simplified diagrams are contained at the back of this section (figure 3-22). Figure 3-22 consists of 35 sheets and the references throughout the text are by sheet number.

3-10. Operation of the CPU/IOP is controlled by the software set of 170 instructions (see table 2-1) and the microprogram. The CPU requests an instruction from memory via the MCU. When the instruction is received over the central data bus it is loaded into the next instruction register (NIR), and from there to the current instruction register (CIR). The look-up table (LUT) address decoder and the LUT ROM produce a microprogram starting address from the instruction received from memory. The microinstruction is read out of the microprogram ROM address specified by the instruction, and decoded into a set of control codes which cause the CPU/IOP to perform certain functions. The following paragraphs describe the operation of the CPU/IOP circuits, starting with the next instruction register. A block diagram is shown in figure 3-2.



2189-11

Figure 3-1. HP 3000 Computer System

3-11. NEXT INSTRUCTION REGISTER.

3-12. The next instruction register (NIR) is a 16-bit register, located on CIR PCA A8 and shown on simplified diagram sheet 9 and detailed diagram DD205, location B3. The NIR provides storage for the instruction to be executed immediately following the current instruction. This allows an instruction to be fetched concurrently with the execution of the preceding instruction. The NIR is loaded by an NIP signal from the MCU operation decoder. The NIP signal is generated as a result of a skip field code NEXT as described in the skip field code definitions (see table 3-5) or MCU options NIR, RWAN, and RWN as described in the MCU option field code definitions (see table 3-8).

3-13. CURRENT INSTRUCTION REGISTER.

3-14. The current instruction register (CIR) is a 16-bit register, located on CIR PCA A8 and shown on simplified diagram sheet 9 and detailed diagram DD203, location B4. The CIR contains the instruction currently being executed by the CPU. The CIR is loaded from the NIR by an NIRTOCIR signal from the next logic. The NIRTOCIR signal is generated as a result of a skip field code NEXT as described in the skip field code definitions (see table 3-5) or the cycle after a special field code CCPX (see table 3-7) with U-bus bit 0 a logic 1.

3-15. ROM MAPPER.

3-16. The CPU logic consisting of the LUT address decoder, LUT ROM, ROM address register (RAR), RAR increment logic, save register, and microprogram ROM comprise an instruction decoder, or ROM mapper. The procedure for entry into the microprogram ROM is as follows:

a. The current instruction register provides the 16-bit instruction fetched from memory to the LUT address decoder.

b. The LUT address decoder uses the instruction to produce an eight-bit LUT address.

c. The LUT address provides an entry into the LUT ROM. The LUT ROM reads out 12 bits, consisting of the W-bit and an 11-bit address which specifies the location in the microprogram ROM for the first microinstruction to be executed as called for by the current instruction. The W-bit is loaded from LUT ROM(0).

d. The 12-bit microprogram ROM address is loaded into the ROM address register (RAR) and from RAR to the microprogram ROM and to the increment RAR logic.

e. The RAR is incremented every 175 nanoseconds, producing a new address for the next microinstruction, until the current instruction is executed completely.

3-17. LUT ADDRESS DECODER. The LUT address decoder is loaded on CIR PCA A8 and shown on simplified diagram sheet 18 and detailed diagram DD205, location A15. The LUT address decoder consists of gating logic which decodes the 16-bit instruction word from the current instruction register into an entry address for the LUT ROM.

3-18. LUT ROM. The LUT ROM, located on CIR PCA A8, is shown on simplified diagram sheet 18 and detailed diagram DD205, location B16. The LUT ROM contains 256 (400 octal) 12-bit locations, which provide starting addresses for the microprogram ROM and "jump to" addresses during microprogram ROM jump conditions.

3-19. Data bit 0 in the look-up table ROM is the W-bit (bits 1 through 11 of the LUT contain the starting address of the microprogram for the instruction to be executed). When a new instruction is to be executed, the W-bit is read out of the LUT and stored in the W-bit register on PCA A8. The W-bit has different meanings for different instructions (the bit has a fixed, known value for every instruction).

a. For STACKOPS (CIR (0:3) = 00_8), the W-bit has no meaning, it is set equal to logic 1 merely for convenience.

b. For SUBOP 1 (CIR (0:3) = 01_8) instructions:

(1) The W-bit is set to a logic 1 for instructions regarding P-relative addresses (branches). In this case, CIR (10) is treated as a sign bit for the P-relative displacement in CIR (11:15). This bit controls the function of the pre-added (add or subtract) such that a positive number (p+) (= CIR (11:15)) or a negative number (p–) (= 2^{16} - CIR (11:15)) may be obtained from it. This number may be used in the microprogram by using PADD in the ROR S-bus field.

(2) The W-bit is set equal to logic 0 for shift-type instructions. In this case, the pre-adder output is CIR (10:15) (a 6-bit shift count), with zeros in all other bit positions.

c. For SUBOP 2 (CIR (0:3) = 02_8) instructions, the W-bit controls the function of the pre-adder. In all cases, the input to the pre-adder is CIR (8:15).

(1) When the W-bit is logic 0 the pre-adder is set to the ADD function. Since the second input to the pre-adder is logic 0 (no indexing), the output is -CIR (8:15) (= 2^{16} - CIR (8:15)), a negative number.

d. For SUBOP 3 (CIR (0:3) = 03_8) instructions there are two cases:

(1) For SPECOP 00 (CIR (0:3) = 03_8), the W-bit is set to logic 0. This forces the pre-adder to the ADD function. In addition, CIR (12:15) only is applied to the pre-adder input. The output, therefore, is the K-field, CIR (12:15).

(2) For SPECOP 01 through 17 (CIR (4:7) = 01_8 - 17_8), the W-bit causes the same action as in SUBOP 2, explained in paragraph 3-19.c.

e. SUBOP 04_8 through 17_8 (CIR (0:3) = 04_8 - 17_8) instructions, in general, reference an operand in memory. The operations necessary to obtain the effective address of this operand are common to most of the instructions, and therefore one microprogram is used for this calculation. When one of these instructions is to be executed, it maps (through the LUT) to this microprogram to obtain the operand address. When this is done, the instruction then jumps to the microprogram that executes the specific instruction called for and the W-bit now becomes effective. The W-bit is set to logic 1. When the foregoing address calculation routine has been completed, a micro-operation (JLUI) in the ROM skip field is executed. If the instruction does not specify indirect addressing (or if one level of indirect addressing has been completed), the execution of JLUI forces a microprogram jump to an address contained in the LUT. Since the contents of CIR have not changed, the LUT would normally still be pointing to the address of the foregoing address calculation routine, and an infinite loop would result. The W-bit, however, now modifies the LUT entry address to a different (but related) address. This LUT address contains the microprogram address of the desired instruction to be executed.

3-20. ROM ADDRESS REGISTER AND INCREMENT LOGIC. The ROM address register (RAR) is a 12-bit register located on ROM PCA A3. The RAR is shown on simplified diagram sheet 18 and detailed diagram DD200, location B6. The RAR is loaded with the 12-bit output of the LUT ROM and, after loading, is automatically incremented every 175 nanoseconds by the increment RAR logic (see DD200, location B4) until the end of the microprogram for the instruction is reached. The RAR is normally loaded from the V-bus unless a repeat is specified, at which time the content of RAR does not change until the repeat is terminated.

3-21. When a Jump to Subroutine (JSB) is decoded by the function field decoder (see table 3-3), a JSB1 signal is generated and the contents of the RAR are loaded into the save register until a Return from Subroutine (RSB) is decoded by the skip field decoder. The RSB signal loads the contents of the save register back into the RAR.

3-22. In addition to the 12-bit output from the LUT ROM, the RAR can be loaded from the ROM output register rank 2 (ROR2) by a JMPGATE signal generated in response to a function field code Jump (JMP) or Jump to Subroutine (JSB), by the interrupt logic due to an interrupt or power failure, or from the U-bus in response to a SWLDRAR signal from the maintenance panel. When RAR is loaded from the U-bus, the condition of the maintenance panel RAR load register switches is stored in RAR.

3-23. SAVE REGISTER. The save register is a 12-bit register located on ROM PCA A3 and shown on simplified diagram sheet 18 and detailed diagram DD200, location B2. When a function field code JSB (Jump to Subroutine) is detected, the JSB1 signal is generated and the content of the RAR is loaded into the save register, and remains there until a skip field code RSB (Return from Subroutine) is detected, at which time the save register content is transferred back to the RAR. The CPU then continues executing the microprogram with the microinstruction following the JSB. The save register is inhibited from loading its content back into the RAR when RAR is being loaded from the U-bus by the SWLDRAR signal.

3-24. MICROPROGRAM ROM. The microprogram ROM is located on ROM PCA A3 and shown on detailed diagram DD200, sheet 2. The ROM accepts the 12-bit address from the RAR and outputs a 32-bit microinstruction word from that address to the ROM output registers. The ROM contains 2048 (4000 octal), 32-bit microinstruction words. Each instruction generally calls several microinstructions from the ROM. For example, instructions which affect the top-of-stack (TOS) will first call a microprogram routine to check that there are enough filled or vacant top-of-stack registers to carry out the operation; then, after one or more memory transfers to adjust the stack, remaining microinstructions called by the instruction will begin. Updated addresses for succeeding microinstructions called by the instruction are furnished to the ROM every 175 nanoseconds by the RAR.

3-25. ROM OUTPUT REGISTER RANK 1.

3-26. The ROM output register rank 1 (ROR1) is a 32-bit register located on ROM PCA A3 (bits 0:9 and 15:31) and ALU PCA A5 (bits 10:14). ROR1 is shown on simplified diagram sheets 13 and 17 and detailed diagrams DD200 (location F2) and DD202 (location C23). ROR1 receives the 32-bit microinstruction word from the microprogram ROM. The R-bus and S-bus fields are connected directly from ROR1 to the R-bus and S-bus decoders with the remainder of the fields (with a few exceptions) connected to ROR2 and then to the decoders. This method allows the R- and S-bus registers to be loaded before executing the rest of the microinstruction.

3-27. When a function field code Repeat (REPC or REPN) is detected, the ROR1 input function and the RAR increment function are disabled. The RAR then contains the address of the microinstruction following the one to be repeated and ROR1 contains the microinstruction to be repeated. The next cycle loads ROR2 and executes the microinstruction to be repeated. The REPEAT signal, inhibiting ROR1 input, is generated by the logical "or" of the decoded REPC/REPN function field codes, and the Repeat flip-flop (located on SSF PCA A4). The Repeat flip-flop is set by a RPTFCN signal generated by the function field decoder. As long as the Repeat flip-flop is set, the microinstruction contained in ROR1 is repeated; when the flip-flop is cleared by a SKIP signal from the skip field decoder, the ROR1 input function is enabled and the next microinstruction is loaded.

3-28. ROM OUTPUT REGISTER RANK 2.

3-29. The ROM output register rank 2 (ROR2) is a 23-bit register. The bits are numbered 5 through 27. Bits 10 through 14 contain the function field code and are located on ALU PCA A5 and shown on simplified diagram sheet 13 and detailed diagram DD202, location C24. The remaining bits contain the store, skip, shift, and special field codes and are located on ROM PCA A3 and shown on simplified diagram sheet 17 and detailed diagram DD200, location G5. The output of ROR2 is connected to the field decoders.

3-30. MICROINSTRUCTION FIELD DECODERS.

3-31. The 32-bit microinstruction word from the ROM output registers is divided into seven fields, each field containing from three to five bits. Each field, when decoded, produces a set of microcode signals which control the operation of the CPU.

3-32. S-BUS FIELD DECODER. The S-bus field decoder is located on S-bus PCA A7 and is shown on simplified diagram sheets 2 and 3 and detailed diagram DD204. The S-bus field decoder (bits 0:4) selects one of 31 registers (or sets of lines) to be loaded into the S-bus register. S-bus field code definitions are shown in table 3-1.

3-33. STORE FIELD DECODER. The R-bus registers store decoder is located on R-bus PCA A6 and shown on simplified diagram sheet 4 and detailed diagram DD203. The S-bus registers store decoder is located on S-bus PCA A7 and shown on simplified diagram sheet 5 and detailed diagram DD204. The store field (bits 5:9) selects one of 22 registers in which to store the U-bus data. In addition, the store field code PUSH moves all stack elements down one location and loads the U-bus word onto the top of stack. Store field code definitions are shown in table 3-2.

3-34. FUNCTION FIELD DECODER. The function field decoder is located on ALU PCA A5 and shown on simplified diagram sheet 13 and detailed diagram DD202. The function field (bits 10:14), when decoded, specifies the function to be performed by the arithmetic logic unit (ALU) on the two operands in the R- and S-bus registers. Function field code definitions are shown in table 3-3, and signals generated as a result of function field codes are shown in table 3-4.

3-35. SKIP FIELD DECODER. The skip field decoder is located on SSF PCA A4 and shown on simplified diagram sheet 23 and detailed diagram DD201. The skip field (bits 15:19) determines what condition will be tested for a possible skip. If the condition is met (e.g., U-bus positive/negative, odd/even, zero/non-zero, overflow set, etc), ROM output register 2 (ROR2) will execute a no-operation (NOP), effectively skipping one microinstruction word. The skip field also specifies the condition under which a JUMP or JUMP to Subroutine (JMP or JSB) will be executed if coded in the microinstruction. Other signals, such as NEXT, which calls the next instruction from memory, are also decoded from the skip field. Skip field code definitions are shown in table 3-5.

3-36. SHIFT FIELD DECODER. The shift field decoder is located on ALU PCA A5 and shown on simplified diagram sheet 15 and detailed diagram DD202. The shift field (bits 20:22) specifies how the T-bus data will be shifted (right one, left one, straight through, etc). In addition, the shift field generates the scratch pad 1 register and scratch pad 3 register shift signals (SP1SHIFT and SP3SHIFT). In the case of certain shifts, the shift field determines what data is shifted into SP1 or SP3 by generating the SP1IN and SP3IN signals. The shift field code definitions are shown in table 3-6.

3-37. SPECIAL FIELD DECODER. The special field decoder is located on SSF PCA A4 and shown on simplified diagram sheet 22 and detailed diagram DD201. The special field (bits 23:27) has varied uses such as generating the memory operation code signals and the POP signal, which moves the stack elements up one location such that the second element (S minus 1) becomes the top of stack. The special field code definitions are shown in table 3-7.

3-38. MCU OPTION FIELD DECODER. The MCU option field decoder is located on MCU PCA A9 and shown on simplified diagram sheet 32 and detailed diagram DD206, location B25. The MCU option field uses the same bits (bits 23:27) as the special field. (The special field is disabled and the MCU option field is enabled when executing a store field code BSP0, BUSH, or BUSL.) MCU option field codes initiate transfers to or from memory and transfers from the CPU output register (UCOR) to the operand, next instruction, or command registers via the central data bus. MCU option field code definitions are shown in table 3-8.

3-39. R-BUS FIELD DECODER. The R-bus field decoder is located on R-bus PCA A6 and shown on simplified diagram sheet 1 and detailed diagram DD203. The R-bus field (bits 28:31) selects one of 15 registers (or the U-bus) for loading into the R-bus register. R-bus field code definitions are shown in table 3-9.

3-40. NAME REGISTER.

3-41. The name register is a two-bit register located on R-bus PCA A6 and shown on simplified diagram sheet 11 and detailed diagram DD203, location C18. The name register functions as a map for access to the TOS registers. When the name register is used to address TOS registers directly onto the TNAME lines, the relationship is as follows:

TNAME =	00	01	10	11
RA =	TR0	TR1	TR2	TR3
RB =	TR1	TR2	TR3	TR0
RC =	TR2	TR3	TR0	TR1
RD =	TR3	TR0	TR1	TR2

Table 3-1. S-Bus Field Code Definitions

LABEL AND NAME	FIELD CODE	DESCRIPTION
(blank)	11111	The S-bus register is loaded with all zeros.
CC (Condition Code)	10111	The CC S-bus field code is used to retrieve the condition code (CC) portion of the status word for use with certain conditional branch instructions. When executed, bits 6 and 7 of the status word are loaded into bits 8 and 9 of the S-bus register and if both of these bits are zeros, S-bus register bit 7 becomes a one. All other S-bus register bits become zeros.
CIR (Current Instruction Register)	00000	The 16-bit output of the current instruction register (CIR) is loaded into the S-bus register.
CPX1	00100	The interrupt status register (CPX1), active only when the CPU is in the RUN mode, is loaded into the S-bus register.
CPX2	00110	The interrupt status register (CPX2), active only when the CPU is in the HALT mode, is loaded into the S-bus register.
CTRH (Counter High)	01101	The 6-bit content of the counter (CNTR) register is loaded into bits 4 thru 9 of the S-bus register. All other S-bus register bits become zeros.
CTRL (Counter Low)	01100	The 6-bit content of the counter (CNTR) register is loaded into bits 10 thru 15 of the S-bus register. All other S-bus register bits become zeros.
DB (Data Base)	10101	The 16-bit content of the data base (DB) register is loaded into the S-bus register.
DL (Data Limit)	11100	The 16-bit content of the data limit (DL) register is loaded into the S-bus register.
IOA (I/O Address)	01001	The 8-bit content of the interrupt device number (IDN) register is loaded into bits 8 thru 15 of the S-bus register. Bits 0 thru 7 of the S-bus register become zeros.
IOD (I/O Data)	01010	The 16-bit content of the direct input data (DID/MUXMA) register in the IOP is loaded into the S-bus register.
MASK	01011	The 16-bit content of the mask register in the IOP is loaded into the S-bus register.
MOD (Module Number)	00101	The MOD S-bus field code provides the CPU with two pieces of information required during interrupt processing. When executed, the 4-bit content of the interrupt module number (IMN) register is loaded into bits 4 thru 7 of the S-bus register. Also, if the CPU is CPU1, bit 13 of the S-bus register becomes a 1. If CPU2, bit 12 of the S-bus register becomes a 1. These bits are used to fetch the correct Q1 and ZI entries in the code segment table. All other bits of the S-bus register become zeros.
OPND (Operand)	10110	The 16-bit content of the operand (OPND) register is loaded into the S-bus register. An attempt to execute an OPND while an MCU operand directed operation is in progress results in a CPU freeze until the MCU operation is complete.
P (Program Counter)	10000	The 16-bit content of the program counter (P) register is loaded into the S-bus register.

Table 3-1. S-Bus Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
PADD (Pre-Adder)	00010	The 16-bit output of the pre-adder (PADD) is loaded into the S-bus register.
PB (Program Base)	11110	The 16-bit content of the program base (PB) register is loaded into the S-bus register.
Q (Stack Marker Pointer)	10001	The 16-bit content of the stack marker pointer (Q) register is loaded into the S-bus register.
QDWN (Stack Marker Pointer Down)	01000	The QDWN S-bus field code is used to put the content of the lowest TOS register in the S-bus register. During execution, TNAME becomes the sum of NAME and SR(1:2) (SR must not be equal to zero) and the S-bus register is loaded as follows: If TNAME = 00 then S-BUS := TR3S If TNAME = 01 then S-BUS := TR0S If TNAME = 10 then S-BUS := TR1S If TNAME = 11 then S-BUS := TR2S To preserve stack integrity, a DCSR (Decrement SR) code must be in the special field. Due to the pipeline effect, a TOS reference in the store field of the preceding microinstruction also uses the above described TNAME.
RA	11011	The RA S-bus field code is used to read the content of the first TOS register (location S). SR must be greater than zero.* During execution, TNAME becomes NAME and the S-bus register is loaded as follows: If TNAME = 00 then S-BUS := TR0S If TNAME = 01 then S-BUS := TR1S If TNAME = 10 then S-BUS := TR2S If TNAME = 11 then S-BUS := TR3S
RB	11010	The RB S-bus field code is used to read the content of the second TOS register (location S-1). SR must be greater than 1.* During execution, TNAME becomes NAME and the S-bus register is loaded as follows: If TNAME = 00 then S-BUS := TR1S If TNAME = 01 then S-BUS := TR2S If TNAME = 10 then S-BUS := TR3S If TNAME = 11 then S-BUS := TR0S
RC	11001	The RC S-bus field code is used to read the content of the third TOS register (location S-2). SR must be greater than 2.* During execution, TNAME becomes NAME and the S-bus register is loaded as follows: If TNAME = 00 then S-BUS := TR2S If TNAME = 01 then S-BUS := TR3S If TNAME = 10 then S-BUS := TR0S If TNAME = 11 then S-BUS := TR1S

*True only if RA:RD are being used as part of the stack. RA:RD often are used by the microprogram as scratch pad registers when not used otherwise.

Table 3-1. S-Bus Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
RD	11000	The RD S-bus field code is used to read the content of the fourth TOS register (location S-3). SR must be equal to 4.* During execution, TNAME becomes NAME and the S-bus register is loaded as follows: If TNAME = 00 then S-BUS := TR3S If TNAME = 01 then S-BUS := TR0S If TNAME = 10 then S-BUS := TR1S If TNAME = 11 then S-BUS := TR2S
SBUS	01111	The SBUS code causes the S-bus register content to remain unchanged.
SM (Stack Memory)	10011	The 16-bit content of the stack memory (SM) register is loaded into the S-bus register.
SP1 (Scratch Pad 1)	00001	The 16-bit content of the scratch pad 1 (SP1) register is loaded into the S-bus register.
SP2 (Scratch Pad 2)	11101	The 16-bit content of the scratch pad 2 (SP2) register is loaded into the S-bus register.
SP3 (Scratch Pad 3)	10101	The 16-bit content of the scratch pad 3 (SP3) register is loaded into the S-bus register.
STA (Status)	10100	The 16-bit status word is loaded into the S-bus register.
SWCH	00111	The 16-bit content of the switch register is loaded into the S-bus register.
UBUS	01110	The 16-bit U-bus data word is loaded into the S-bus register. The U-bus data is established by the preceding microinstruction.

*True only if RA:RD are being used as part of the stack. RA:RD often are used by the microprogram as scratch pad registers when not used otherwise.

Table 3-2. Store Field Code Definitions

LABEL AND NAME	FIELD CODE	DESCRIPTION
(blank)	11111	The U-bus word is not stored.
BSP0 (Bus to Scratch Pad 0)	00101	The store field code BSP0 is used to initiate an MCU memory operation. The U-bus data is stored in the CPU output register (COR) and in scratch pad 0 (SP0). A bus low request is issued along with a module TO code decoded from U-bus bits 0, 1, and 2. The special field decoder is disabled and ROR2 bits 23 thru 27 provide an MCU option which specifies the type of operation to be performed (RWA, RWAN, RWN, RNWA, CWA, or CRL).
BUSH (Bus High)	00110	The store field code BUSH allows use of the central data bus to transfer data from the CPU output register (COR) to the operand, next instruction, or command (CMDMOP) register. The U-bus data is stored in the CPU output register and a bus high request is issued. The special field decoder is disabled and MCU option decoder enabled. MCU options allowed are OPND, NIR, and CRL.
BUSL (Bus Low)	00100	The store field code BUSL is used to initiate an MCU memory operation. The U-bus data is stored in the CPU output register (COR) and a bus low request is issued along with a module TO code derived from U-bus bits 0, 1, and 2. The special field decoder is disabled and ROR2 bits 23 thru 27 provide an MCU option which specifies the type of operation to be performed. MCU options allowed are RWA, RWAN, RWN, RNWA, CWA, and CRL.
CTRH (Counter High)	01111	The store field code CTRH stores U-bus bits 4 thru 9 in the counter (CNTR) register bits 0 thru 5.
CTRL (Counter Low)	01110	The store field code CTRL stores U-bus bits 10 thru 15 in the counter (CNTR) register bits 0 thru 5.
DATA	00111	The store field code DATA is used to transfer data via the central data bus after a clear-write memory operation has been initialized. Only the memory module initialized by a low bus request (BSP0 or BUSL) and a CWA MCU option will accept this data transfer. A bus high request is issued by the DATA code. The MCU option decoder is not enabled by DATA and the special field is decoded and executed in the usual manner.
DB (Data Base)	10011	The store field code DB stores the 16-bit U-bus word in the data base (DB) register.
DL (Data Limit)	11100	The store field code DL stores the 16-bit U-bus word in the data limit (DL) register.
IOA (I/O Address)	00001	The store field code IOA stores bits 0 and 5 through 15 of the word currently on the S-bus in the I/O processor control (IOPC) register. Due to the pipeline affect, this word is established by the S-bus field of the microinstruction following the one containing the IOA code. The store field code IOA also disables CPU flag 1 and substitutes I/O flag 1 in its place. This condition remains until the special field code CF1 restores the use of flag 1 to the CPU.

Table 3-2. Store Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
IOD (I/O Data)	00010	The store field code IOD stores the 16-bit word currently on the S-bus in the direct output data (DOD) register. Due to the pipeline effect, this word is established by the S-bus field of the microinstruction following the one containing the IOD code. The store field code IOD also disables CPU flag 1 and substitutes I/O flag 1 in its place. This condition remains until the special field code CF1 restores the use of flag 1 to the CPU.
MASK	00000	The store field code MASK stores the 16-bit word currently on the S-bus in the mask register. Due to the pipeline effect, this word is established by the S-bus field of the microinstruction following the one containing the MASK code. The store field code MASK also disables CPU flag 1 and substitutes I/O flag 1 in its place. This condition remains until the special field code CF3 clears the Direct Data and Direct Active flip-flops (setting DDG and DAG to zero), and the special field code CF1 restores the use of flag 1 to the CPU.
MREG (Memory Register)	00011	The store field code MREG is used to store data in an address that lies in a TOS register (ie, S $\geqslant$ E $>$ SM where S = SR + SM). Prior to executing MREG, the value E minus S is placed in the SP1 register. During execution, TNAME becomes the sum of NAME and SP1(14:15) and the TOS registers are loaded as follows: If TNAME = 00 then TR0 := U-BUS If TNAME = 01 then TR1 := U-BUS If TNAME = 10 then TR2 := U-BUS If TNAME = 11 then TR3 := U-BUS Due to the pipeline effect, a TOS register referenced in the R- or S-bus field of the following microinstruction assumes the above described TNAME.
P (Program Count)	10000	The store field code P stores the 16-bit U-bus word in the program counter (P) register.
PB (Program Base)	11110	The store field code PB stores the 16-bit U-bus word in the program base (PB) register.
PL (Program Limit)	01001	The store field code PL stores the 16-bit U-bus word in the program limit (PL) register.
PUSH	01000	The store field code PUSH effectively moves all stack elements down one location and loads the U-bus word on the top of stack. To maintain stack integrity, SR must be less than four. When PUSH is executed, TNAME becomes NAME and the TOS registers are loaded as follows: If TNAME = 00 then TR3 := U-BUS If TNAME = 01 then TR0 := U-BUS If TNAME = 10 then TR1 := U-BUS If TNAME = 11 then TR2 := U-BUS To complete the operation, NAME is decremented and SR is incremented.

Table 3-2. Store Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
Q (Stack Marker Pointer)	10001	The store field code Q stores the 16-bit U-bus word in the stack marker pointer (Q) register.
QUP (Stack Marker Pointer Up)	10110	The store field code QUP effectively inserts the U-bus word into the stack at location SM plus one. For example, if stack locations S and S minus one are in the TOS registers and location S minus two is the first stack element in memory (location SM), execution of QUP places the U-bus word in a TOS register at stack location S minus two. The first stack element in memory (location SM) becomes S minus three. To maintain stack integrity, the SR register must be incremented (special field code INSR) indicating the addition of a TOS register element. SR must be less than four to properly execute QUP. When the store field code QUP is executed, TNAME becomes the sum of NAME and SR and the TOS registers are loaded as follows: If TNAME = 00 then TR0 := U-BUS If TNAME = 01 then TR1 := U-BUS If TNAME = 10 then TR2 := U-BUS If TNAME = 11 then TR3 := U-BUS Due to the pipeline effect, a TOS register referenced in the R- or S-bus fields of the following microinstruction assumes the above described TNAME.
RA	11011	The store field code RA stores the U-bus word in the first TOS register (location S). SR must be greater than zero.* During execution of RA, TNAME becomes NAME and the TOS registers are loaded as follows: If TNAME = 00 then TR0 := U-BUS If TNAME = 01 then TR1 := U-BUS If TNAME = 10 then TR2 := U-BUS If TNAME = 11 then TR3 := U-BUS
RAR (ROM Address Register)	10111	The store field code RAR stores bits 4 thru 15 of the U-bus in ROM address register bits 0 thru 11. The intent of this code is to force the processor to a new microprogram address specified by the U-bus word. Execution of the RAR code requires three microcycles. The first loads the ROM address register and the next two are NOPs allowing the ROM output registers (ROR1 and ROR2) to be loaded with the new microinstruction.
RB	11010	The store field code RB stores the U-bus word in the second TOS register (location S minus one). SR must be greater than one.* During execution of RB, TNAME becomes NAME and the TOS registers are loaded as follows: If TNAME = 00 then TR1 := U-BUS If TNAME = 01 then TR2 := U-BUS If TNAME = 10 then TR3 := U-BUS If TNAME = 11 then TR0 := U-BUS

*True only if RA:RD are being used as part of the stack, RA:RD often are used by the microprogram as scratch pad registers when not used otherwise.

Table 3-2. Store Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
RC	11001	The store field code RC stores the U-bus word in the third TOS register (location S minus two). SR must be greater than two.* During execution of RC, TNAME becomes NAME and the TOS registers are loaded as follows: If TNAME = 00 then TR2 := U-BUS If TNAME = 01 then TR3 := U-BUS If TNAME = 10 then TR0 := U-BUS If TNAME = 11 then TR1 := U-BUS
RD	11000	The store field code RD stores the U-bus word in the fourth TOS register (location S minus three). SR must be equal to four.* During execution of RD, TNAME becomes NAME and the TOS registers are loaded as follows: If TNAME = 00 then TR3 := U-BUS If TNAME = 01 then TR0 := U-BUS If TNAME = 10 then TR1 := U-BUS If TNAME = 11 then TR2 := U-BUS
SM (Stack Memory Pointer)	10010	The store field code SM stores the U-bus word in the stack memory (SM) register.
SP0 (Scratch Pad 0)	10101	The store field code SP0 stores the U-bus word in the scratch pad 0 (SP0) register.
SP1 (Scratch Pad 1)	01100	The store field code SP1 stores the U-bus word in the scratch pad 1 (SP1) register. A special two-bit register (SP1X) that duplicates SP1 bits 14 and 15 for use with the namer circuits is also loaded with U-bus bits 14 and 15 by the store field code SP1.
SP2 (Scratch Pad 2)	11101	The store field code SP2 stores the U-bus word in the scratch pad 2 (SP2) register.
SP3 (Scratch Pad 3)	10101	The store field code SP3 stores the U-bus word in the scratch pad 3 (SP3) register.
STA (Status)	10100	The store field code STA stores the U-bus word in the status register.
X (Index)	10110	The store field code X stores the U-bus word in the index (X) register.
Z (Stack Limit Pointer)	01010	The store field code Z stores the U-bus word in the stack limit pointer (Z) register.

*True only if RA:RD are being used as part of the stack, RA:RD often are used by the microprogram as scratch pad registers when not used otherwise.

Table 3-3. Function Field Code Definitions

LABEL AND NAME	FIELD CODE	DESCRIPTION
ADD	11111	The content of the R-bus register is added to the content of the S-bus register and the result is placed on the T-bus.
ADDO (Add-Enable Overflow)	11011	The content of the R-bus register is added to the content of the S-bus register and the result is placed on the T-bus. Carry and overflow are enabled and the condition code is set to CCA on the U-bus. This instruction takes two cycles if executed on same line as NEXT.
AND	00111	The content of the R-bus register is logically "anded" with the content of the S-bus register and the result is placed on the T-bus.
BNDT (Bounds Test)	01101	The function field code BNDT is used to perform a bounds test of an address. Execution of this code results in the content of the R-bus register minus the content of the S-bus register being placed on the T-bus. The R- and S-bus fields are coded so that this result is a negative number (CARRY = 0) if a bounds violation occurs. If the CPU is not operating in the priviledged mode (STATUS(0) = 0), and a bounds violation occurs, a microjump to ROM address 0002 is executed. If no violation has occurred (CARRY = 1) or the CPU is operating in the priviledged mode (STATUS(0) = 1), the next microinstruction will be executed in the usual manner. This is a two-cycle operation (causes a one-cycle JMPFRZ).
CAD (Complement and Add)	01110	The content of the R-bus register is added to the one's complement of the content of the S-bus register and the result is placed on the T-bus. If the S-bus register contains all zeros, CAD results in the R-bus register content minus 1 on the T-bus.
CADO (Complement and Add-Enable Overflow)	01010	The content of the R-bus register is added to the one's complement of the content of the S-bus register and the result is placed on the T-bus. Carry and overflow are enabled and the condition code is set to CCA. This instruction takes two cycles if executed on same line as NEXT.
CAND (Complement-And)	00101	The R-bus register content is logically "anded" with the complement of the S-bus register content and the result is placed on the T-bus.
CRS (Circular Shift)	11010	The R-bus register content is added to the S-bus register content and the result is placed on the T-bus. The T-bus is then circular shifted one place right or left as specified in the shift field (SR1 or SL1) and placed on the U-bus.
CTSD (Controlled Shift Double)	10111	The function field code CTSD adds the contents of the R-bus register and the S-bus register, puts the result on the T-bus, and performs a double word shift of the T-bus and a scratch pad left or right as specified by the shift field code (SR1 or SL1). The type of shift is determined by the content of the current instruction register (CIR) as follows: If CIR(7) = 1 then circular shift If CIR(7:8) = 01 then logical shift If CIR(7:8) = 00 then arithmetic shift The most significant word is on the T-bus. If a left shift is specified, scratch pad 1 (SP1) contains the least significant word. If a right shift is specified, scratch pad 3 (SP3) contains the least significant word. Regardless of the direction of the shift, both SP1 and SP3 are shifted left and right respectively.

Table 3-3. Function Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
CTSS (Controlled T-bus Shift Single)	11100	The R-bus register content is added to the S-bus register content and the result is placed on the T-bus. The T-bus is then shifted left or right as specified by the shift field code (SR1 or SL1). The type of shift is determined by the content of the current instruction register as follows: If CIR(7) = 1 then circular shift If CIR(7:8) = 01 then logical shift If CIR(7:8) = 00 then arithmetic shift

DVSB (Divide-Subtract)	01000	The function field code DVSB performs the subtract, shift, and test necessary to execute a divide algorithm. The R- and S-bus fields of the microinstruction are coded so that initially the 16-bit divisor is in the S-bus register and the most significant 16-bits of the dividend are in the R-bus register. The least significant 16-bits of the dividend are in the SP1 register. Both divisor and dividend must be positive numbers upon execution of the DVSB code and Flag 2 (F2) must be 0 (cleared). An SL1 code in the shift field of the microinstruction directs the left shift of the T-bus. The following algorithm is then executed 17 times to perform the complete divide.
		
		TBUS := RBUS - SBUS;
		UBUS(0:14) := TBUS(1:15);
		If ALU carry or F2=1 then
		BEGIN
		RREG(0:14) := UBUS(0:14);
		RREG(15) := SP1(0);
		SP1(0:14) := SP1(1:15);
		SP1(15) := 1
		F2 := TBUS(0);
		END
		else
		BEGIN
		RREG(0:14) := RREG(1:15);
		RREG(15) := SP1(0);
		SP1(0:14) := SP1(1:15);
		SP1(15) := 0
		F2 := RREG(0);
		END
		
		After 17 executions of the above algorithm, the 16-bit quotent is contained in the SP1 register and the remainder times 2 is contained in the R-bus register. When the remainder is unloaded from the R-bus register, it is shifted right one place (divided by 2).

INC (Incremented Add)	11101	The R-bus register content is added to the S-bus register content plus 1. This results in an increment of either register (R- or S-bus) content placed on the T-bus if the other register contains all zeros.

Table 3-3. Function Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
INCO (Incremented Add-Enable Overflow)	11001	The R-bus register content is added to the S-bus register content plus 1, the result placed on the T-bus, and the carry and overflow logic is enabled. The condition code is set to CCA on the U-bus data. This instruction takes two cycles if executed on same line as NEXT.
IOR (Inclusive OR)	10110	The content of the R-bus register is logically inclusively "ored" with the content of the S-bus register and the result is placed on the T-bus.
JMP (Jump)	01100	The JMP function field code directs a micro-jump to the ROM address (jump target) specified by bits 20 thru 31 of the ROM output register if the skip field condition is met (a condition must be specified). The R-bus, shift, and special field decoders are disabled and the U-bus and T-bus become the S-bus register content. If the skip field code is ZERO, NZRO, EVEN, ODD, NSME, BIT6, BIT8, NOFL, CRRY, NCRY, POS, or NEG, a one-cycle freeze is generated to allow the U-bus data to be established and checked for the condition. If the condition is met, the ROM address register (RAR) is loaded with the jump target and both ranks of the ROM output register (ROR1 and ROR2) are NOPed. Two NOP cycles allow the pipeline to fill and the microinstruction at the jump target address is then executed. If the condition is not met the next microinstruction is executed in the usual manner.
JSB (Jump to Subroutine)	00100	The JSB function field code directs a micro-subroutine jump to the ROM address specified by bits 20 thru 31 of the ROM output register if the skip field code condition is met. A JSB is first detected in rank one of the ROM output register 1 (ROR1) at which time the ROM address register (RAR) content is loaded into the save register if the next cycle is not going to have NOP2 on (i.e. SKIP = 0). If the condition is met and the JSB is executed, the save register content is used as a return address at the subroutine end (see function field code RSB). During execution of the JSB, the R-bus, shift, and special field decoders are disabled and the T-bus and U-bus become the S-bus register content. If the skip field code is ZERO, NZRO, EVEN, ODD, NSME, BIT6, BIT8, NOFL, CRRY, NCRY, POS or NEG, a one-cycle freeze is generated to allow the U-bus data to be established and checked for the condition. If the condition is met, the RAR is loaded with the subroutine address and both ranks of the ROR are NOPed. Two NOP cycles allow the pipeline to fill and the microinstruction at the subroutine address is then executed. If the condition is not met, the next micro-instruction is executed in the usual manner.

Table 3-3. Function Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION

MPAD (Multiply-Add)	11000	The function field code MPAD performs the add, shift, and test necessary to execute a multiply algorithm. The R-bus field of the microinstruction is coded so that initially the 16-bit multiplicand is in the R-bus register. The S-bus field code is UBUS which is initially all zeros. Scratch pad 3 contains the 16-bit multiplier. Both multiplier and multiplicand must be positive numbers upon execution of the MPAD code. An SR1 code in the shift field directs the right shift of the T-bus. The following algorithm is executed 16 times to perform a complete multiply.
		
		T-BUS := R-REG plus S-REG;
		U-BUS(1:15) := T-BUS(0:14);
		U-BUS(0) := ALUcarry;
		If SP3(15) = 1 then
		BEGIN
		S-REG := U-BUS;
		SP3(1:15) := SP3(0:14);
		SP3(0) := T-BUS(15);
		END
		else
		BEGIN
		S-REG(1:15) := S-REG(0:14);
		SP3(1:15) := SP3(0:14);
		SP3(0) := S-REG(15);
		END
		
		After 16-executions of the above algorithm, the result is a 32-bit word with the most significant 16-bits in the S-bus register and the least significant 16-bits in scratch pad 3.

PNLR (Panel Read)	10000	The PNLR function field code allows the auxiliary control panel to select and display a CPU register. This code appears in the microprogram during execution of HALT and PAUSE routines. When PNLR is executed, the ROM output register (ROR1) R-bus and S-bus fields are disabled. The maintenance panel interface supplies these field codes which put the content of the selected register in the associated (R- or S-bus) register. The T-bus and U-bus become the R-bus register content plus the S-bus register content (one of which will be zeros). The auxiliary control panel completes this operation by displaying the U-bus as the selected register.
PNLS (Panel Store)	10001	The PNLS function field code allows the auxiliary control panel to load a CPU register with the content of one of its switch registers. This code is part of the halt mode interrupt micro-routine for servicing a maintenance panel interrupt. When PNLS is executed, the ROM output register (ROR2) store field is disabled and the maintenance panel interface card supplies the store field code respective of the selected CPU register. A SWCH S-bus field code causes the S-bus register to be loaded with the content of the selected auxiliary control panel switch register. The T-bus and U-bus become the R-bus register content (zeros) plus the S-bus register content and at the end of the cycle, the selected register is loaded with the U-bus data.

Table 3-3. Function Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
REPC (Repeat Until Condition)	10100	The REPC function field code causes the next microinstruction to be executed repeatedly until the skip field condition of that microinstruction is met. The skip field of the microinstruction containing the REPC code is not used. During execution the T-bus becomes the R-bus register content plus the S-bus register content. The REPC code is decoded from ROR2 and at that time disables the RAR increment function and the ROR1 load function and sets the Repeat FF. The RAR then contains the address of the microinstruction following the one to be repeated and ROR1 contains the microinstruction to be repeated. The next cycle loads ROR2 and executes the microinstruction to be repeated. As long as the Repeat FF remains set, the content of ROR1 and ROR2 does not change and is executed each cycle. When the skip field condition is met, the Repeat FF is cleared, the pipeline is filled correctly, and the next microinstruction is fetched in the usual manner.
REPN (Repeat N Times)	10101	The REPN function code operates in the same manner as the REPC code described for the preceding label. The difference is that REPN loads a repeat counter register with the content of the microinstruction skip field. Bits 1 thru 5 of the counter become ROR2 bits 5 thru 19; bit 0 of the counter becomes a 1. The counter content is then the two's complement of the number of repeats to be performed. To utilize the counter, the repeated microinstruction contains a special field code INCTR (Increment Counter) and a skip field condition CTRM (Counter Maximum).
ROM	10010	The function field code ROM loads the R-bus register with a 16-bit constant obtained from the microinstruction. The ROM code is decoded from ROR1, loading the R-bus register with bits 16 thru 31 of ROR1. The T-bus then becomes the R-bus register content plus the S-bus register content. The R-bus, shift, special, and skip field decoders are disabled by the ROM code.
ROMI (ROM Inclusive)	10010	The function field code ROMI loads the R-bus register with a 16-bit constant obtained from the microinstruction. The ROMI code is decoded from ROR1, loading the R-bus register with ROR1 bits 16 thru 31. The T-bus then becomes the R-bus register content inclusive "ored" with the S-bus register content. The R-bus, shift, special, and skip field decoders are disabled by the ROMI code.
ROMN (ROM And)	00011	The function field code ROMN loads the R-bus register with a 16-bit constant obtained from the microinstruction. The ROMN code is decoded from ROR1, loading the R-bus register with ROR1 bits 16 thru 31. The T-bus becomes the R-bus register content logically "anded" with the S-bus register content. The R-bus, shift, special, and skip field decoders are disabled by the ROMN code.
ROMX (ROM Exclusive)	00010	The function field code ROMX loads the R-bus register with a 16-bit constant obtained from the microinstruction. The ROMX code is decoded from ROR1, loading the R-bus register with ROR1 bits 16 thru 31. The T-bus becomes the R-bus register content exclusive "ored" with the S-bus register content. The R-bus, special, shift, and skip field decoders are disabled by the ROMX code.
RPTY (Reverse Parity)	11110	The function field code RPTY reverses the parity generated by the CPU for central data bus bound words. This function is used for diagnostic purposes only.

Table 3-3. Function Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
SUB (Subtract)	01111	The content of the S-bus register is subtracted from the content of the R-bus register and the result is placed on the T-bus.
SUBO (Subtract-Enable Overflow)	01011	The content of the S-bus register is subtracted from the content of the R-bus register and the result is placed on the T-bus. Carry and overflow are enabled and condition code CCA is set on the U-bus data word. This instruction takes two cycles if executed on same line as NEXT.
TASL (Triple Arithmetic Shift Left)	00000	The function field code TASL causes a three word arithmetic left shift of the contents of the S-bus, SP1, and the R-bus registers containing the most, middle and least significant words respectively. The sign bit is saved, bit 1 of the most significant word is lost, and bit 15 of the least significant word becomes zero. The S-bus register content is read to the T-bus and shifted onto the U-bus by an SL1 code in the shift field. The other two words are shifted in their respective registers.
TASR (Triple Arithmetic Shift Right)	00001	The function field code TASR causes a three word arithmetic right shift of the contents of the R-bus, SP3, and S-bus registers containing the most, middle, and least significant words respectively. The sign bit remains bit 0 of the most significant word and is copied into bit 1 of the most significant word. Bit 15 of the least significant word is lost. The R-bus register content is read onto the T-bus and shifted onto the U-bus by an SR1 code in the shift field. The other two words are shifted in their respective registers.
UBNT (Unconditional Bounds Test)	01001	The function field code UBNT is used to perform an unconditional bounds test of an address. Execution of this code results in the content of the R-bus register minus the content of the S-bus register being placed on the T-bus. The R-bus and S-bus field are coded so that this result is a negative number (CARRY = 0) if a bounds violation occurs. The response to a bounds violation is a micro-jump to ROM address 0002. If no violation occurs (CARRY = 1), the next microinstruction is executed in the usual manner. This is a two-cycle operation (causes a one-cycle JMPFRZ).
XOR (Exclusive OR)	00110	The content of the R-bus register is exclusive "ored" with the content of the S-bus register and the result is placed on the T-bus.

Table 3-4. Function Field Code Signals

FUNCTION FIELD LABEL	FIELD CODE	SIGNALS GENERATED	FUNCTION FIELD LABEL	FIELD CODE	SIGNALS GENERATED
ADD	11111	ALUS(0:3) = 1001 ALUMODE = 0 $\overline{\text{ALUFC}}$ = 1	INCO	11001	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 0 OFCENB = 1
ADDO	11011	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 1 OFCENB = 1	IOR	10110	ALUS(0:3) = 1000 ALUMODE = 0 ALUFC = 1 S3:S0 = 0001
AND	00111	ALUS(0:3) = 1101 ALUMODE = 1	JMP	01100	ALUS(0:3) = 0101 ALUMODE = 1 ALUFC = 1 JMPJSB2 = 1 $\overline{\text{SKIPNOP}}$ = 0 RFINH = 1
BNDT	01101	ALUS(0:3) = 0110 ALUMODE = 0 $\overline{\text{ALUFC}}$ = 0 $\overline{\text{BNDT}}$ = 0			
CAD	01110	ALUS(0:3) = 0110 ALUMODE = 0 ALUFC = 1	JSB	00100	ALUS(0:3) = 0101 ALUMODE = 1 ALUFC = 1 JMPJSB2 = 1 JSB1 = 1 $\overline{\text{SKIPNOP}}$ = 0 RFINH = 1
CADO	01010	ALUS(0:3) = 0110 ALUMODE = 0 ALUFC = 1 OFCENB = 1			
CAND	00101	ALUS(0:3) = 1110 ALUMODE = 1 S3:S0 = 0111	MPAD	11000	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 1 SP3SHIFT = 1
CRS	11010	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 1 CRS = 1	PNLR	10000	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 1 $\overline{\text{PNLREAD}}$ = 1 RSSEL = 1
CTSD	10111	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 1 SP1SHIFT = 1 SP3SHIFT = 1	PNLS	10001	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 1 PANLSTOR = 1
DVSB	01000	ALUS(0:3) = 0110 ALUMODE = 0 ALUFC = 1 DVSB = 1 SP1SHIFT = 1	REPC	10100	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 1 RPTFCN = 1 REPEAT = 1
INC	11101	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 0			

Table 3-4. Function Field Code Signals (Continued)

FUNCTION FIELD LABEL	FIELD CODE	SIGNALS GENERATED	FUNCTION FIELD LABEL	FIELD CODE	SIGNALS GENERATED
REPN	10101	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 1 RPTFCN = 1 REPEAT = 1 $\overline{\text{REPN}} = 0$	RPTY	11110	PRTYMODE = 1
			SUB	01111	ALUS(0:3) = 0110 ALUMODE = 0 ALUFC = 0 Two's Complement Subtracts
ROM	10010	ALUS(0:3) = 1001 ALUMODE = 0 ALUFC = 1 ROMFCN1 = 1 RFINH = 1 $\overline{\text{ROMFCNT}} = 0$ $\overline{\text{SKIPNOP}} = 0$	SUBO	01011	ALUS(0:3) = 0110 ALUMODE = 0 ALUFC = 0 OFCENB = 1
ROMI	10010	ALUS(0:3) = 1000 ALUMODE = 0 ALUFC = 1 ROMFCN1 = 1 RFINH = 1 $\overline{\text{ROMFCNT}} = 0$ $\overline{\text{SKIPNOP}} = 0$	TASL	00000	ALUS(0:3) = 0101 ALUMODE = 1 SP1SHIFT = 1
			TASR	00001	ALUS(0:3) = 1111 ALUMODE = 1 SP3SHIFT = 1
ROMN	00011	ALUS(0:3) = 1101 ALUMODE = 1 ROMFCN1 = 1 RFINH = 1 $\overline{\text{ROMFCNT}} = 0$ $\overline{\text{SKIPNOP}} = 0$	UBNT	01001	ALUS(0:3) = 0110 $\overline{\text{ALUMODE}} = 0$ $\overline{\text{ALUFC}} = 0$ $\overline{\text{UBNT}} = 0$
ROMX	00010	ALUS(0:3) = 0110 ALUMODE = 1 ROMFCN1 = 1 RFINH = 1 $\overline{\text{ROMFCNT}} = 0$ $\overline{\text{SKIPNOP}} = 0$	XOR	00110	ALUS(0:3) = 0110 ALUMODE = 1

Table 3-5. Skip Field Code Definitions

LABEL AND NAME	FIELD CODE	DESCRIPTION
(blank)	11111	No skip field operation.
BIT6	00101	The skip field code BIT6 sets the NOP2 FF if bit 6 of the U-bus word is a logic 1. Causes conditional jump and JSB to be two-cycle instructions.
BIT8	00110	The skip field code BIT8 sets the NOP2 FF if bit 8 of the U-bus word is a logic 1. Causes conditional jump and JSB to be two-cycle instructions.
CRRY (Carry)	01000	The skip field code CRRY sets the NOP2 FF if the ALU carry out is a logic 1. Causes conditional jump and JSB to be two-cycle instructions.
CTRM (Counter Max)	11011	The skip field code CTRM sets the NOP2 FF if the counter contains all ones.
EVEN	00010	The skip field code EVEN sets the NOP2 FF if the U-bus word is an even number (U-bus bit 15 is a logic 0). Causes conditional jump and JSB to be two-cycle instructions.
F1 (Flag 1)	01100	The skip field code F1 sets the NOP2 FF if Flag 1 FF is set.
F2 (Flag 2)	01110	The skip field code F2 sets the NOP2 FF is Flag 2 FF is set.
F3 (Flag 3)	11100	The skip field code F3 sets the NOP2 FF if Flag 3 FF is set.
INDR (Indirect)	10100	The skip field code INDR sets the NOP2 FF if the Indirect Bit FF is set and the indirect signal is a logic 1.
JLUI	11001	The skip field code JLUI causes a microjump to the ROM address specified by the LUT (look up table) providing the indirect condition (skip field INDR code) is not met. If the indirect condition is met the microjump is not executed.
NCRY	01001	The skip field code NCRY sets the NOP2 FF if the carry out from the ALU is zero. Causes conditional jump and JSB to be two-cycle instructions.
NEG	01011	The skip field code NEG sets the NOP2 FF if the U-bus word is a negative number (U-bus bit 0 is a logic 1). Causes conditional jump and JSB to be two-cycle instructions.

Table 3-5. Skip Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
NEXT	11101	The skip field code NEXT is a complex operation that causes the hardware to fetch and decode the next user instruction. If stackop A has just been executed and stackop B is not a NOP, and no interrupts are pending, the hardware executes the stackop B portion of the current instruction. For any other condition without interrupts pending, a four cycle operation occurs as follows: Cycle 1. The current instruction register (CIR) is loaded with the content of the next instruction register (NIR). The CPU output register (PCOR) is loaded with the P-register content and sent to the MCU along with a command to read-write memory and return the results to the NIR. If Flag 1 or Flag 2 FF's have been set by some previous operation, they are cleared at this time. CLIB FF is cleared and ROR1 is NOPed. Cycle 2. The new CIR content is decoded via the look up table (LUT) and the LUT output is loaded into the ROM address register (RAR). ROR1 and ROR2 are NOPed. Cycle 3. At the end of this cycle, ROR1 is loaded with the micro-instruction at the address specified by the RAR content. This is the first microinstruction for execution of the new user instruction. The pre-adder output (relative address portion of the CIR plus index) is loaded into the R-bus register and the content of the base register (P, DB, Q, or S) is loaded into the S-bus register depending on whether the instruction contained in CIR is a MEM REF instruction or not. If the base is the P-register, it is important to note that the P-register is pointing at the instruction being fetched from memory and not the current instruction. The resultant effective address will therefore be one greater than the desired address. The CPU must decrement this address in some following microinstruction. At the end of cycle 3, the P-register is incremented by one to point to the next instruction in memory. ROR1 and ROR2 are NOPed. Cycle 4. The U-bus becomes the sum of the R-bus register content plus the S-bus register content. This is the effective address of the operand or, in the case of P-relative addressing, the effective address plus 1. Also, at the end of this cycle ROR2 is loaded with the content of ROR1, the R-bus and S-bus fields of ROR1 are executed and ROR2 is NOPed. The pipeline is now full and execution of the new user instruction is in progress. When the memory operation started by cycle 1 is complete, the NIR contains the next sequential user instruction.
NF1 (Not Flag 1)	01101	The skip field code NF1 sets the NOP2 FF if Flag 1 FF is cleared.
NF2 (Not Flag 2)	01111	The skip field code NF2 sets the NOP2 FF is Flag 2 FF is cleared.
NOFL (Not Overflow)	00111	The skip field code NOFL sets the NOP2 FF if the ALU overflow bit is not a logic 1. Causes conditional jump and JSB to be two-cycle instructions.
NPRV (Not Privileged)	10110	The skip field code NPRV sets the NOP2 FF if the privileged mode bit (status word bit 0) is zero.

Table 3-5. Skip Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
NSME (Not Same)	00100	The skip field code NSME sets the NOP2 FF if all bits of the T-bus are not the same. Causes conditional jump and JSB to be two-cycle instructions.
NZRO (Not Zero)	00001	The skip field code NZRO sets the NOP2 FF if the T-bus word is not equal to zero. Causes conditional jump and JSB to be two-cycle instructions.
ODD	00110	The skip field code ODD sets the NOP2 FF if the U-bus word is an odd number (U-bus bit 15 is a logic 1). Causes conditional jump and JSB to be two-cycle instructions.
POS (Positive)	01010	The skip field code POS sets the NOP2 FF if the U-bus word is a positive number (U-bus bit 0 is a logic 0). Causes conditional jump and JSB to be two-cycle instructions.
RSB (Return from Subroutine)	11000	The skip field code RSB causes a microjump to the ROM address contained in the save register. ROR1 and ROR2 are NOPed allowing two cycles to fill the pipeline with the target address microinstructions.
SR4 (SR=4)	10010	The skip field code SR4 sets the NOP2 FF if the SR register content is equal to four.
SRL2 (SR<2)	10101	The skip field code SRL2 sets the NOP2 FF if the SR register content is less than two.
SRL3 (SR<3)	10111	The skip field code SRL3 sets the NOP2 FF if the SR register content is less than three.
SRN4 (SR Not 4)	10011	The skip field code SRN4 sets the NOP2 FF if the SR register content is not four.
SRNZ (SR Not Zero)	10001	The skip field code SRNZ sets the NOP2 FF if the SR register content is not zero.
SRZ (SR Zero)	1000	The skip field code SRZ sets the NOP2 FF if the SR register content is equal to zero.
TEST	11010	The skip field code TEST sets the NOP2 FF if any enabled interrupt is pending.
UNC (Unconditional)	11110	The skip field code UNC sets the NOP2 FF unconditionally.
ZERO	00000	The skip field code ZERO sets the NOP2 FF if the T-bus word is equal to zero.

Table 3-6. Shift Field Code Definitions

LABEL AND NAME	FIELD CODE	DESCRIPTION
(blank)	111	The T-bus word is placed directly on the U-bus.
LLZ (Left to Left and Zero)	001	The shift field code LLZ places the left byte of the T-bus in the left byte of the U-bus and zeros in the right byte of the U-bus.
LRZ (Left to Right and Zero)	000	The shift field code LRZ places the left byte of the T-bus in the right byte of the U-bus and zeros in the left byte of the U-bus.
RLZ (Right to Left and Zero)	101	The shift field code RLZ places the right byte of the T-bus in the left byte of the U-bus and zeros in the right byte of the U-bus.
ROT (Rotate)	110	The shift field code ROT places the left byte of the T-bus in the right byte of the U-bus and the right byte of the T-bus in the left byte of the U-bus.
RRZ (Rotate to Right and Zero)	100	The shift field code RRZ places the right byte of the T-bus in the right byte of the U-bus and zeros in the left byte of the U-bus.
SL1 (Shift Left 1)	010	The shift field code SL1 shifts the T-bus one place left onto the U-bus. Refer to the function field code descriptions for the action taken when used with function field codes CRS, CTSD, CTSS, DVSB, and TASL.
SR1 (Shift Right 1)	011	The shift field code SR1 shifts the T-bus logically one place right onto the U-bus. Refer to the function field code descriptions for the action taken when used with function field codes CRS, CTSD, CTSS, MPAD, and TASR.

Table 3-7. Special Field Code Definitions

LABEL AND NAME	FIELD CODE	DESCRIPTION
(blank)	11111	No special field operation.
CCA (Condition Code A)	11110	The special field code CCA sets the condition code bits to CCL (01) if the T-bus word is less than zero (T(0) = 1), CCE (10) if the T-bus word is equal to zero (Signal T = 0 is true), or CCG (00) if the T-bus word is greater than zero (T(0) = 0 and signal T = 0 is false).
CCB (Condition Code B)	00000	The special field code CCB sets the condition code to CCL (01) if bits 8 thru 15 of the SP1 register form a special ASCII character, CCE (10) if an alphabetic ASCII character, or CCG (00) if a numeric ASCII character.
CCE (Condition Code E)	11101	The special field code CCE sets the condition code bits to CCE (10).

Table 3-7. Special Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
CCG (Condition Code G)	11100	The special field code CCG sets the condition code bits to CCG (00).
CCL (Condition Code L)	11011	The special field code CCL sets the condition code bits to CCL (01).
CCPX (Clear CPX1)	00001	The special field code CCPX is used to control the interrupt status registers (CPX1 and CPX2) of the CPU. Executing a CCPX code with the respective true U-bus bits results in the following: U-Bus bit 0 Force NIR to CIR (for diagnostics). 1 Clear System Parity Error FF. 2 Clear Address Parity Error FF. 3 Clear Data Parity Error FF. 4 Clear CPU Timer FF. 5 Clear Bounds Violation FF. 6 Clear Illegal Address FF. 7 Clear Module Interrupt FF. 8 Clear External Interrupt FF. 9 Clear maintenance panel Interrupt FF's. 10 Turn off all interrupts. (Note: The only way to re-enable the interrupts is to press the CPU RESET switch or if a power failure occurs.) 11 Clear interrupt stack flag. 12 Clear dispatcher flag. 13 Set panel error light. 14 Freeze processor. (For diagnostics). 15 Clear maintenance panel Interrupt FF's.
CCRY (Clear Carry)	10101	The special field code CCRY clears the ALU Carry FF.
CCZ (Condition Code Zero)	11010	The special field code CCZ sets the condition code bits to CCE (10) if the T-bus word is equal to zero (signal T = 0 true) or CCG (00) if the T-bus word is not equal to zero (signal T = 0 false).
CF1 (Clear Flag 1)	10010	The special field code CF1 clears CPU Flag 1 FF.
CF2 (Clear Flag 2)	10001	The special field code CF2 clears CPU Flag 2 FF.
CF3 (Clear Flag 3)	00111	The special field code CF3 clears CPU Flag 3 FF.
CLIB (Clear Indirect Bit)	01110	The special field code CLIB clears the Indirect Bit FF. The Indirect Bit FF remains cleared until a skip field code NEXT or JLUI is executed or the CIR is loaded with the NIR content by the special field code CCPX.

Table 3-7. Special Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
CLO (Clear Overflow)	11001	The special field code CLO clears the status word overflow bit.
CTF (Set Carry to Flag 1)	00110	The special field code CTF stores the ALU carry bit in the Flag 1 FF.
DCSR (Decrement SR)	01001	The special field code DCSR decrements the content of the SR register by a count of one.
FHB (Flag to High Bit)	01101	The special field code FHB transfers the content of the Flag 1 FF to bit 0 of the U-bus.
HALT	00011	The special field code HALT clears CPX2 bit 0 which enables maintenance panel control of the CPU.
HBF (High Bit to Flag 1)	01100	The special field code HBF transfers the content of U-bus bit 0 to the Flag 1 FF.
INCN (Increment Name)	01010	The special field code INCN increments the content of the name register by a count of one.
INCT (Increment Counter)	01011	The special field code INCT increments the content of the counter register by a count of one.
INSR (Increment SR)	01000	The special field code INSR increments the content of the SR register by a count of one.
LBF (Low Bit to Flag 2)	01111	The special field code LBF transfers the content of U-bus bit 15 to the Flag 2 FF.
POP	10111	The special field code POP moves the stack elements up one location such that the second element of the stack (S minus one) becomes the top element (S), etc. The previous top of stack element is lost. When executed, this is accomplished by decrementing the SR register and incrementing the name register.
POPA (Pop setting CCA)	10110	The special field code POPA functions the same as special field code POP with the addition that the condition code is set to CCL (01) if the T-bus word is less than zero, CCE (10) if the T-bus word is equal to zero, or CCG (00) if the T-bus word is greater than zero.

Table 3-7. Special Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
SCRY (Set Carry Bit)	10100	The special field code SCRY sets the Carry FF.
SDFG (Set Dispatcher Flag)	00101	The special field code SDFG sets the dispatcher flag (bit 12 of interrupt status register CPX2).
SF1 (Set Flag 1)	10011	The special field code SF1 sets CPU Flag 1 FF.
SF2 (Set Flag 2)	10000	The special field code SF2 sets CPU Flag 2 FF.
SIFG (Set Interrupt Stack Flag)	00100	The special field code SIFG sets the interrupt flag (bit 11 of interrupt status register CPX2).
SOV (Set Overflow)	11000	The special field code SOV sets the status word overflow bit.
SR0 (Set SR to Zero)	00010	The special field code SR0 clears the SR register. No other operation referencing the SR register is allowed during this microcycle (note that pipeline effects must be allowed for).

Table 3-8. MCU Option Field Code Definitions

LABEL AND NAME	FIELD CODE	DESCRIPTION
CMD (Command)	11000	The MCU option CMD gates the TO code and MOP code from the CMDTO and CMDMOP registers onto the central data bus. This option is used with a store field code BSP0 or BUSL.
CRL (Control)	11001	The MCU option CRL stores bits 10, 11 and 13 thru 15 of the CPU output register (COR) in the CMDTO and CMDMOP registers. This option is used with a store field code BUSH.
CWA (Clear-Write Address)	11110	The MCU option CWA initiates a bus request for a memory clear-write cycle and tells the memory that the word on the central data bus is a memory address and that a data word will be sent on the next transmission. The selected memory module will be busy for all modules until that data word is received via the store field code DATA. The MCU option CWA is issued with a store field code BSP0 or BUSL.
NIR (Next Instruction Register)	11010	The MCU option NIR transfers data from the central data bus to the next instruction register. This option is used with a store field code BUSH.

Table 3-8. MCU Option Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
RNWA (Read-No-Write Address)	11101	The MCU option RNWA initiates a bus request for memory and tells the selected memory module that the word on the central data bus is an address. The content of the address is to be sent back to the operand register and the write portion of the memory cycle is to be omitted leaving all ones in that memory location. This option is used with a store field code BSP0 or BUSL.
RWA (Read-Write Address)	11111	The MCU option RWA initiates a bus request for memory and tells the selected memory module that the word on the central data bus is a memory address. The content of the address is to be sent back to the operand register and restored in the memory location. This option is used with a store field code BSP0 or BUSL.
RWAN (Read-Write Address NIR)	10xxx	The MCU option RWAN initiates a bus request for memory and tells the selected memory module that the word on the central data bus is a memory address. The content of the address is to be sent back to the operand register and the next instruction register and restored in the memory location. This option is used with a store field code BSP0 or BUSL.
RWN (Read-Write NIR)	11011	The MCU option RWN initiates a bus request for memory and tells the selected memory module that the word on the bus is a memory address. The content of the address is to be sent back to the next instruction register and restored in the memory location. This option is used with a store field code BSP0 or BUSL.

Table 3-9. R-Bus Field Code Definitions

LABEL AND NAME	FIELD CODE	DESCRIPTION
(blank)	1111	The R-bus register is loaded with all zeros.
MREG	0011	The MREG R-bus field code is used to fetch a memory element that happens to lie in a TOS register (i.e., E is greater than SM). Prior to executing MREG, the value S minus E must be placed in the SP1 register. During execution of MREG, TNAME becomes the sum of NAME and SP1(14:15) and the R-bus register is loaded as follows: If TNAME = 00 then R-BUS := TR0R If TNAME = 01 then R-BUS := TR1R If TNAME = 10 then R-BUS := TR2R If TNAME = 11 then R-BUS := TR3R Due to the pipeline affect, a TOS register referenced in the store field of the preceding microinstruction assumes the above TNAME.

Table 3-9. R-Bus Field Code Definitions

LABEL AND NAME	FIELD CODE	DESCRIPTION
PADD (Pre-Adder)	0100	The 16-bit output of the pre-adder is loaded into the R-bus register.
PL (Program Limit)	0000	The 16-bit content of the program limit (PL) register is loaded into the R-bus register.
RA	1011	The RA R-bus field code is used to read the content of the first TOS register (location S). SR must be greater than 0. During execution, TNAME becomes NAME and the R-bus register is loaded as follows: If TNAME = 00 then R-BUS := TR0R If TNAME = 01 then R-BUS := TR1R If TNAME = 10 then R-BUS := TR2R If TNAME = 11 then R-BUS := TR3R
RB	1010	The RB R-bus field code is used to read the second TOS register (location S-1). SR must be greater than 1. During execution, TNAME becomes NAME and the R-bus register is loaded as follows: If TNAME = 00 then R-BUS := TR1R If TNAME = 01 then R-BUS := TR2R If TNAME = 10 then R-BUS := TR3R If TNAME = 11 then R-BUS := TR0R
RBUS	0101	The RBUS R-bus field code causes the R-bus register to remain unchanged.
RC	1001	The RC R-bus field code is used to read the third TOS register (location S2). SR must be greater than 2. During execution, TNAME becomes NAME and the R-bus register is loaded as follows: If TNAME = 00 then R-BUS := TR2R If TNAME = 01 then R-BUS := TR3R If TNAME = 10 then R-BUS := TR0R If TNAME = 11 then R-BUS := TR1R
RD	1000	The RD R-bus field code is used to read the fourth TOS register (location S-3). SR must be equal to 4. During execution, TNAME becomes NAME and the R-bus register is loaded as follows: If TNAME = 00 then R-BUS := TR3R If TNAME = 01 then R-BUS := TR0R If TNAME = 10 then R-BUS := TR1R If TNAME = 11 then R-BUS := TR2R
SP0 (Scratch Pad 0)	1101	The 16-bit content of the scratch pad 0 (SP0) register is loaded into the R-bus register.
SP1 (Scratch Pad 1)	1100	The 16-bit content of the scratch pad 1 (SP1) register is loaded into the R-bus register.

Table 3-9. R-Bus Field Code Definitions (Continued)

LABEL AND NAME	FIELD CODE	DESCRIPTION
SR (Stack Register)	0001	The 3-bit content of the stack (SR) register is loaded into the R-bus register bits 0 thru 2. R-bus register bits 3 thru 15 become zeros.
UBUS	1110	The 16-bit U-bus data word is loaded into the R-bus register. The U-bus data is established by the preceding microinstruction.
X (Index)	0110	The 16-bit content of the index (X) register is loaded into the R-bus register.
XC (X Conditional)	0111	The XC R-bus field code is used with indexed memory addressing. If the index bit of the current instruction (CIR bit 4) is zero, the R-bus register is loaded with zeros, otherwise the R-bus register is loaded with the 16-bit content of the X-register.
Z (Stack Limit)	0010	The 16-bit content of the stack limit (Z) register is loaded into the R-bus register.

3-42. The foregoing does not imply that all TOS registers are a legitimate part of the stack. The stack register (SR) counter is also considered in referencing a TOS register. If, for example, SR = 2 (010), then the TOS registers RA and RB are stack locations S and S-1 but RC and RD are invalid. Stack locations S-2 and S-3 are in memory. The content of the name register is affected by the store field code PUSH and special field codes INCN and POP as described in the field code definitions.

3-43. The adder (located on R-bus PCA A6 and shown on simplified diagram sheet 11 and detailed diagram DD203, location D16) and namer (located on R-bus PCA A6 and shown on simplified diagram sheet 11 and detailed diagram DD203, location D17) combine the outputs of the name, stack, and scratch pad 1 registers and generate the TNAME 0:1 signals for the TOS mappers.

3-44. TOP-OF-STACK MAPPERS.

3-45. The top-of-stack (TOS) mappers are located on R-bus PCA A6 and S-bus PCA A7 and are shown on simplified diagram sheets 1, 2 and 4 and detailed diagrams DD203, locations D8 and F17, and DD204, location G47. The TOS mappers use the TNAME code to control access to the TOS registers. This code from the namer signifies to the mappers which of the TOS registers (TR0R:TR3R and TR0S:TR3S) is RA, RB, etc. The outputs from the mappers are combined with the outputs from the R-bus, S-bus and store decoders to load data into and out of the TOS registers.

3-46. TOP-OF-STACK REGISTERS.

3-47. The top-of-stack registers consist of eight 16-bit registers designated TR0R:TR3R and TR0S:TR3S. The R-bus TOS registers are located on R-bus PCA A6 and shown on simplified diagram sheet 6 and detailed diagram DD203, locations B42 (TR0R), B43 (TR1R), B44 (TR2R), and B46 (TR3R). The S-bus TOS registers are located on S-bus PCA A7 and are shown on simplified diagram sheet 7 and detailed diagram DD204, locations B45 (TR0S), B46 (TR1S), B47 (TR2S), and B49 (TR3S). The two groups of registers always contain the same data, ie, TR0R = TR0S, TR1R = TR1S, etc. The registers contain up to four of the top elements of the current data stack. The TOS registers are read by R-bus field codes RA, RB, RC, RD, and MREG and S-bus field codes RA, RB, RC, RD, and QDWN as described in the field code definitions. The TOS registers are loaded by store field codes RA, RB, RC, RD, PUSH, and QUP as described in the field code definitions.

3-48. INDEX REGISTER.

3-49. The index (X) register is a 16-bit register located on R-bus PCA A6 and shown on simplified diagram sheet 6 and detailed diagram DD203, location C39. The index register contains the index word to be used by memory reference instructions if indexing is specified. Certain other instructions (not memory reference) use the X-register for parameters or addresses. The X-register is read by R-bus field codes X and XC and loaded by store field code X as described in the field code definitions.

3-50. STACK LIMIT REGISTER.

3-51. The stack limit (Z) register is a 16-bit register located on R-bus PCA A6 and shown on simplified diagram sheet 6 and detailed diagram DD203, location B47. The Z-register contains an absolute address which points to the top memory location available to the current data stack. There are locations (28 words) in the stack above the stack limit; however, these are reserved for stack markers in the event of an interrupt. The Z-register is read by R-bus field code Z and loaded by store field code Z as described in the field code definitions.

3-52. PROGRAM LIMIT REGISTER.

3-53. The program limit (PL) register is a 16-bit register located on R-bus PCA A6 and shown on simplified diagram sheet 6 and detailed diagram DD203, location B48. The PL register contains the absolute address of the upper location of the current program segment. The PL register is read by an R-bus field code PL and loaded by a store field code PL as described in the field code definitions.

3-54. SCRATCH PAD 0 REGISTER.

3-55. The scratch pad 0 (SP0) register is a 16-bit register located on R-bus PCA A6 and shown on simplified diagram sheet 6 and detailed diagram DD203, location B49. The SP0 register is used by the CPU for storage of partial results during various CPU routines and as an address for memory transfers. SP0 also is used to display the memory address on the auxiliary control panel (ACP). On ACP, SP0 is named MEM ADDR. The SP0 register is read by R-bus field code SP0 and loaded by store field codes SP0 and BSP0 as described in the field code definitions.

3-56. SCRATCH PAD 1 REGISTER.

3-57. The scratch pad 1 (SP1) register is a 16-bit register located on R-bus PCA A6 and shown on simplified diagram sheet 6 and detailed diagram DD203, location B25. The SP1 register is used by the CPU to store partial results during various microprogram routines. SP1 also is used to display memory contents on the auxiliary control panel (ACP). SP1 is named MEM DATA on ACP. The SP1 register can be shifted left and provides serial data input to bit 15 and output from bit 0. A two-bit register which duplicates SP1 bits 14 and 15 (cannot be shifted) is referred to as SP1X and is used by the namer circuits during execution of R-bus or store field MREG codes. The SP1 register is read by R-bus field code SP1, loaded by store field code SP1, and shifted by function field codes CTSD, DVSB, and TASL as described in the field code definitions. In addition, SP1 can be read onto the S-bus by S-bus field code SP1 (code is not the same as R-bus code SP1). SP1 bits 8:15 also are used to set CCB by special field code CCB.

3-58. SCRATCH PAD 1X REGISTER.

3-59. The scratch pad 1X (SP1X) register is a two-bit register located on R-bus PCA A6 and shown on simplified diagram sheet 11 and detailed diagram DD203, location F23. SP1X is loaded by store field code SP1 and duplicates bits 14 and 15 of the SP1 register; however, the SP1X register cannot be shifted. The content of the SP1X register is used during execution of the R-bus field code MREG and store field code MREG as described in the field code definitions.

3-60. STACK REGISTER.

3-61. The stack register (SR) counter is a three-bit register located on R-bus PCA A6 and shown on simplified diagram sheet 11 and detailed diagram DD203, location C15. The SR counter provides the number of TOS registers that are currently in use. The SR counter works in conjunction with the name register to locate and access any of the top four elements of the data stack. The SR counter is read by R-bus field code SR and modified by store field code PUSH and special field codes INSR, DCSR, POPA, SR0, and POP, as described in the field code definitions.

3-62. PROGRAM BASE REGISTER.

3-63. The program base (PB) register is a 16-bit register located on S-bus PCA A7 and shown on simplified diagram sheet 7 and detailed diagram DD204, location B39. The PB register contains the absolute address of the bottom location of the current program segment. The PB register is read by S-bus field code PB and loaded by store field code PB as described in the field code definitions.

3-64. DATA LIMIT REGISTER.

3-65. The data limit (DL) register is a 16-bit register located on S-bus PCA A7 and shown on simplified diagram sheet 7 and detailed diagram DD204, location B35. The DL register contains the absolute address of the bottom useable location in the current data stack. The DL register is read by S-bus field code DL and loaded by store field code DL as described in the field code definitions.

3-66. STACK MEMORY REGISTER.

3-67. The stack memory (SM) register is a 16-bit register located on S-bus PCA A7 and shown on simplified diagram sheet 7 and detailed diagram DD204, location B34. The SM register contains the absolute address of the top element of the data stack in memory. Depending upon the number of TOS registers in use (reflected by the contents of the SR register), this address can be from zero to four locations below the actual top of stack. The SM register is read by S-bus field code SM and loaded by store field code SM as described in the field code definitions.

3-68. DATA BASE REGISTER.

3-69. The data base (DB) register is a 16-bit register located on S-bus PCA A7 and shown on simplified diagram sheet 7 and detailed diagram DD204, location B33. The DB register is one of the stack limit registers. The DB register contains the absolute address of the first location of directly addressable storage in the current data stack. The DB register is read by S-bus field code DB and loaded by store field code DB as described in the field code definitions.

3-70. Q-REGISTER.

3-71. The Q-register is a 16-bit stack marker register located on S-bus PCA A7 and shown on simplified diagram sheet 7 and detailed diagram DD204, location B32. The Q-register contains the absolute address of the current stack marker being used within the data stack. The Q-register is read by S-bus field code Q and loaded by store field code Q as described in the field code definitions.

3-72. SCRATCH PAD 2 REGISTER.

3-73. The scratch pad 2 (SP2) register is a 16-bit register located on S-bus PCA A7 and shown on simplified diagram sheet 7 and detailed diagram DD204, location B37. The SP2 register is used by the CPU to store partial results during various microprogram routines. The SP2 register is read by S-bus field code SP2 and loaded by store field code SP2 as described in the field code definitions.

3-74. SCRATCH PAD 3 REGISTER.

3-75. The scratch pad 3 (SP3) register is a 16-bit register located on S-bus PCA A7 and shown on simplified diagram sheet 7 and detailed diagram DD204, location C9. The SP3 register is used by the CPU to store partial results during various microprogram routines. The SP3 register can be shifted right and provides serial data input to bit 0 and output from bit 15. The SP3 register is read by S-bus field code SP3, loaded by store field code SP3, and shifted by function field codes CTSD, MPAD, and TASR as described in the field code definitions.

3-76. PROGRAM COUNTER REGISTER.

3-77. The program counter (P) register is a 16-bit register located on S-bus register PCA A7 and shown on simplified diagram sheet 9 and detailed diagram DD204, location C7. The P-register contains the absolute address of the next program instruction to be fetched from memory. During execution of a skip field code NEXT, bits 0, 1, 2, 14, and 15 are encoded to provide a memory module number and the entire content of the P-register is loaded into the PCOR register to be sent via the central data bus to the selected memory module. The P-register is incremented by an INCP signal which is generated by the next logic. The P-register is read by S-bus field code P and loaded by store field code P as described in the field code definitions.

3-78. COUNTER REGISTER.

3-79. The counter (CNTR) register is a six-bit register located on S-bus PCA A7 and shown on simplified diagram sheet 8 and detailed diagram DD204, location E25. The CNTR register is used as a repeat counter by the CPU. The two's complement of the desired count is loaded into the CNTR register; the CNTR register is then incremented for each repeated execution until it contains all ones as indicated by a Counter Maximum (CTRM) code from the skip field. The CNTR register is affected or referenced by S-bus field codes CTRI and CTRH, function field code REPN, store field codes CTRL and CTRH, special field code INCT, and skip field code CTRM as described in the field code definitions.

3-80. In addition to the above function, the CNTR register saves the content of the SR register when the CPU is put in the halt mode. This is necessary because the halt microroutine pushes all data stack elements into memory thus forcing the SR register content to zero. The CNTR register can then be displayed to show what the content of the SR register was just prior to the halt.

3-81. STATUS REGISTER.

3-82. The status register is a 16-bit register. The first eight bits (0:7) are located on SSF PCA A4 and are reset by CPU RESET. The second eight bits (8:15), located on S-bus PCA A7, are not affected by CPU RESET. The status register is shown on simplified diagram sheet 8 and detailed diagrams DD201 and DD204. The status register indicates the current status of the CPU hardware. The function and the location of the source for each bit is as follows:

a. Bit 0. Privileged mode bit (DD201F26).
1 = Privileged mode.
0 = User mode.

b. Bit 1. External interrupts (DD201F26). Includes module interrupts and console interrupts (if in DISPATCH).
1 = Enable.
0 = Disable.

c. Bit 2. User traps (DD201F26).
1 = Enable.
0 = Disable.

d. Bit 3. Stack op B (DD201G25).
1 = Pending.
0 = Not pending.

e. Bit 4. Overflow bit (DD201D9).

f. Bit 5. Carry bit (DD201C8)

g. Bits 6:7. Condition code. (DD201E8 and F8).

00 = CCG
01 = CCL
10 = CCE

11 = Note that on this implementation of the CPU, a "11" stored in bits 6:7 acts as though the condition code is both CCE and CCL. This should not be used and is presented for information only.

h. Bits 8:15. Current executing code segment number (DD204D38).

3-83. The status register is read by S-bus field code STA and loaded by store field code STA as described in the field code definitions. Status bits also are affected by function field codes CADO, SUBO, INCO, ADDO; and special field codes CCB, SCRY, CCRY, POPA, SOV, CLO, CCZ, CCL, CCG, CCE, and CCA.

3-84. PRE-ADDER.

3-85. The pre-adder, located on R-bus PCA A6 and shown on simplified diagram sheet 12 and detailed diagrams DD203 (sheet 4) and DD205 (location B37), is used to gain a speed increase for instructions which use or perform computations on bits in the CIR. For example, when executing indexed memory reference instructions (and not indirect), the proper displacement field of the CIR is preadded to the contents of the X-register. Thus the final absolute address can be computed in only one cycle by adding the output of the pre-adder to the contents of the base register (PB, DB, Q or Z).

3-86. R-BUS REGISTER.

3-87. The R-bus register is a 16-bit register located on ALU PCA A5 and shown on simplified diagram sheet 14 and detailed diagram DD202, location B44. The R-bus register provides buffer storage between the R-bus and the CPU arithmetic and logic function. The R-bus register can be shifted left one bit position (see function field code TASL). The R-bus register is loaded from the R-bus. (Refer to R-bus field code definitions.)

3-88. S-BUS REGISTER.

3-89. The S-bus register is a 16-bit register located on ALU PCA A5 and shown on simplified diagram sheet 14 and detailed diagram DD202, location B26. The S-bus register provides buffer storage between the S-bus and the CPU arithmetic and logic function. The S-bus register can be shifted right one bit position (refer to function field code TASR). The S-bus register is loaded from the S-bus. (Refer to S-bus field code definitions.)

3-90. ALU FUNCTION GENERATOR.

3-91. The arithmetic and logic unit (ALU) function generator, located on ALU PCA A5 and shown on simplified diagram sheet 14 and detailed diagram DD202, sheet 1, combines the R- and S-bus data in a manner specified by a four-bit function code (ALUS 0:3) from the ALU function decoder. Functions that are generated are divided into two modes, or groups: arithmetic functions and logic functions. An ALU mode signal (ALUMODE) selects the function mode, or group, and the four-bit ALUS code selects the function within the group.

3-92. FLAG 1 REGISTER.

3-93. Flag 1 is a one-bit register located on SSF PCA A4 and shown on simplified diagram sheet 19 and detailed diagram DD201, location G8. The flag 1 register is used as a CPU flag and for temporary storage of the ALU carry bit and U-bus sign bit during certain arithmetic routines. During execution of direct I/O operations, the output of the flag 1 register is inhibited and replaced by an I/O flag which indicates that a direct data or command transfer is currently taking place on the IOP bus. The flag 1 register is set by special field code SF1 and cleared by special field code CF1 and skip field code NEXT as described in the field code definitions.

3-94. FLAG 2 REGISTER.

3-95. Flag 2 is a one-bit register located on SSF PCA A4 and shown on simplified diagram sheet 19 and detailed diagram DD201, location F18. The flag 2 register is used as a CPU flag, as temporary storage for bit 15 of the U-bus, and as temporary storage for the most significant bit of the dividend during execution of the function field code DVSB. Flag 2 is set by special field code SF2 and cleared by special field code CF2 and skip field code NEXT as described in the field code definitions.

3-96. T-BUS SHIFTER.

3-97. The T-bus shifter is located on ALU PCA A5 and shown on simplified diagram sheet 16 and detailed diagram DD202, sheet 2. The 16-bit output of the ALU function generator (the combined R- and S-bus data) is applied to the T-bus shifter. All shifts and rotates of the T-bus (left shift, right shift, right-left swap, etc) are executed as directed by the shift field decoder. The output of the T-bus shifter is placed on the U-bus to be stored in one of the U-bus registers.

3-98. MAPPERS.

3-99. The mappers, located on S-bus PCA A7 and shown on detailed diagram DD204, sheet 2, examine the three most significant bits of each address word from the P-register or the U-bus and convert these bits into the TO code of the memory module. Switches are used to configure the mappers appropriately for the quantity and sizes of memory modules existent in the computer system.

3-100. U-CPU OUTPUT REGISTER.

3-101. The U-CPU output register (UCOR) is a 16-bit register located on S-bus PCA A7 and shown on simplified diagram sheet 9 and detailed diagram DD204, location A2. The UCOR register functions as a buffer for memory bound data and operand addresses transferred between the U-bus and the central data bus. The UCOR register is loaded with the U-bus word by store field codes BUSL, BSPO, BUSH, and DATA as described in the field code definitions.

3-102. P-CPU OUTPUT REGISTER.

3-103. The P-CPU output register (PCOR) is a 16-bit register located on S-bus PCA A7 and shown on simplified diagram sheet 9 and detailed diagram DD204, location A4. The PCOR register functions as a memory address buffer between the P-register and the central data bus. A skip field code NEXT loads the PCOR register with the contents of the P-register.

3-104. P-REGISTER OR U-BUS MEMORY OPERATION REGISTER.

3-105. The P-register or U-bus memory operation (PUMOP) register is a two-bit register located on MCU PCA A9 and shown on simplified diagram sheet 34 and detailed diagram DD206, location F28. The PUMOP register is loaded with the memory operation code (Read-Restore, Clear-Write, or Read-No Write) specified by the MCU operation decoder when executing a low bus request (store field codes BUSL or BSPO). In the event of a skip field code NEXT, the PUMOP register is loaded with a Read-Restore memory operation code. The content of the PUMOP register is then put on the central data bus MOP lines to control the operation of the selected memory module.

3-106. COMMAND MEMORY OPERATION REGISTER.

3-107. The command memory operation (CMDMOP) register is a two-bit register located on MCU PCA A9 and shown on simplified diagram sheet 34 and detailed diagram DD206, location F29. The CMDMOP register contains a memory operation code to be transferred to a selected memory module via the central data bus during execution of MCU option field code CMD. The CMDMOP register is loaded with bits 10 and 11 of the UCOR register during execution of MCU option field code CRL as described in the field code definitions.

3-108. COMMAND-TO REGISTER.

3-109. The command-to (CMDTO) register is a three-bit register located on MCU PCA A9 and shown on simplified diagram sheet 35 and detailed diagram DD206, location F27. The CMDTO register contains the module number of the memory module selected to receive the content of the CMDMOP register during execution of MCU option field code CMD. The CMDTO register is loaded with bits 13, 14, and 15 of the UCOR register during execution of MCU option field code CRL as described in the field code definitions.

3-110. P-TO REGISTER.

3-111. The P-to (PTO) register is a three-bit register located on MCU PCA A9 and shown on simplified diagram sheet 35 and detailed diagram DD206, location F26. The PTO register is loaded with the module number of the memory module containing the instruction to be fetched during execution of a special field code NEXT. The content of the PTO register is used to check the module READY lines to ensure that the selected memory module is not busy. The PTO register content is then put on the central data bus TO lines to direct the memory operation to the selected memory module.

3-112. U-TO REGISTER.

3-113. The U-to (UTO) register is a three-bit register located on MCU PCA A9 and shown on simplified diagram sheet 35 and detailed diagram DD206, location F24. The UTO register is loaded with the number of the memory module selected during a low bus request (store field codes BUSL and BSPO). The content of the UTO register is compared with the module READY lines to ensure that the selected memory module is not busy. The UTO register content is then put on the central data bus TO lines to direct the operation to the selected memory module.

3-114. P-TO-NEXT INSTRUCTION REGISTER.

3-115. The P-to-next instruction PTO(NIR) register is a three-bit register located on MCU PCA A9 and shown on simplified diagram sheet 35 and detailed diagram DD206, location F23. The PTO(NIR) register is loaded with the number of the memory module containing the instruction being fetched during execution of a skip field code NEXT. The content of the PTO(NIR) register is compared with the central data bus FROM lines. When the FROM lines match the PTO(NIR) register content and the TO lines contain the CPU module number, the word on the MCU data lines (part of the central data bus) is assumed to be the fetched instruction and is directed to the next instruction register.

3-116. U-TO-NEXT INSTRUCTION REGISTER.

3-117. The U-to-next instruction register UTO(NIR) is a three-bit register located on MCU PCA A9 and shown on simplified diagram sheet 35 and detailed diagram DD206, location F22. The UTO(NIR) register is loaded with the module number of the selected memory module during execution of a low bus request (store field codes BUSL and BSPO) in combination with an MCU option field code which specifies that memory data be returned to the next instruction register (MCU options RWAN and RWN). The content of the UTO(NIR) register is compared with the central data bus FROM lines. When the FROM lines match the UTO(NIR) register content and the TO CPU number, the word on the MCU data lines (part of the central data bus) is assumed to be the data requested from memory and is directed to the next instruction register.

3-118. U-TO OPERAND REGISTER.

3-119. The U-to operand UTO(OPND) register is a three-bit register located on MCU PCA A9 and shown on simplified diagram sheet 35 and detailed diagram DD206, location F25. The UTO(OPND) register is loaded with the module number of the selected memory module during execution of a low bus request (store field codes BUSL and BSPO) in combination with an MCU option field code that specifies that memory data be returned to the operand register (MCU option field codes RNWA, RWA, and RWAN). The content of the UTO(OPND) register is compared with the central data bus FROM lines. When the FROM lines match the UTO(OPND) register content and the TO CPU number, the word on the MCU data lines (part of the central data bus) is assumed to be the data requested from memory and is directed to the operand register.

3-120. TO-FROM COMPARATORS.

3-121. The TO-FROM comparators are located on MCU PCA A9 and shown on simplified diagram sheet 35 and detailed diagram DD206, locations G24 and G26. The comparators compare the TO lines from the UTO(OPND), UTO(NIR), or PTO(NIR) (depending upon the type of operation being executed) registers with the central data bus FROM lines. When the TO and FROM lines match, the ROPND and RNIR signals are generated and sent to the MCU operation decoder, causing the operand or next instruction register to be loaded with the contents of the central data bus MCU data lines.

3-122. READY DECODER AND COMPARATOR.

3-123. The ready decoder and comparator are located on MCU PCA A9 and shown on simplified diagram sheet 34 and detailed diagram DD206, locations D4 and D5. The ready decoder multiplexes the three-bit output of the UTO, PTO, or CMDTO register, depending upon the type of operation being executed, and generates a module READY number. The comparator compares this READY number with the READY lines from the central data bus to ensure that the selected module is ready to receive.

3-124. INTERRUPT MODULE NUMBER REGISTER.

3-125. The interrupt module number (IMN) register is a four bit register located on MCU PCA A9 and shown on detailed diagram DD206, location B16. The central data bus FROM code is stored in the IMN register. When the CPU is ready to send data to a memory module, the FROM code is read out of the IMN register and becomes the TO code of the destination module.

3-126. CPU REQUEST/SELECT LOGIC.

3-127. The CPU request/select logic is located on MCU PCA A9 and shown on simplified diagram sheet 34 and detailed diagram DD206, sheet 1. The CPU request/select logic controls the operation of the UCOR and PCOR and CMDMOP and PUMOP registers. A CPU Select (CPUSEL) flip-flop, located on the CPU request/select logic, generates the CPUSEL signal when set, gating the contents of the UCOR or PCOR registers and the CMDMOP or PUMOP registers to the central data bus. The CPU Select flip-flop is set for one cycle by the output of the ready comparator. The setting of this flip-flop is inhibited whenever a MCUERR or an ILLADR signal is asserted.

3-128. INTERRUPT DEVICE NUMBER REGISTER.

3-129. The interrupt device number (IDN) register is an eight-bit register located on IOP PCA A10 and shown on simplified diagram sheet 30 and detailed diagram DD207, location E18. When an interrupt is acknowledged, the IOP interrupt control logic generates an IENB signal, loading the device number of the interrupting subsystem from the IOP bus into the IDN register. The CPU reads the content of the IDN register by S-bus field code IOA as described in the field code definitions.

3-130. INPUT/OUTPUT PROCESSOR CONTROL REGISTER.

3-131. The I/O processor control (IOPC) register is a 12-bit register located on IOP PCA A10 and shown on simplified diagram sheet 30 and detailed diagram DD207, location B16. The IOPC register is loaded by a STOIOA signal, which is generated by the store decoder as a result of store field code IOA as described in the field code definitions. Bit 0 of the IOPC register, when a logic 1, causes a SO signal to be asserted. Bits 1:3 (I/O command) and bits 4:11 (device address) are then sent via the command and device number lines of the IOP bus out to the selected device controller by a DAG signal from the IOP direct control logic.

3-132. DIRECT OUTPUT DATA REGISTER.

3-133. The direct output data (DOD) register is a 16-bit register located on IOP PCA A10 and shown on simplified diagram sheet 30 and detailed diagram DD207, location C2. During execution of direct I/O commands, the DOD register serves as a buffer for data transferred from the CPU to an I/O subsystem. The DOD register is loaded by a STOIOD signal, which is generated by the store field decoder as a result of store field code IOD. The 16-bit content of the DOD register is gated onto the IOP bus by a DDG signal generated by the IOP direct control logic.

3-134. IOP DIRECT CONTROL LOGIC.

3-135. The IOP direct control logic is located on IOP PCA A10 and shown on simplified diagram sheet 28 and detailed diagram DD207, sheet 4. When the CPU/IOP is operating in the direct mode, the IOP direct control logic monitors the status of the Data Poll, Service Out, and Service In signals and controls the operation of the IOPC and DOD registers. The Direct Data flip-flop, when set, generates the DDG signal which gates the contents of the DOD register onto the IOP bus. The Direct Address flip-flop, when set, generates the DAG signal which gates the contents of the IOPC register onto the IOP bus. When Data Poll is issued by the IOP, the DDG and DAG signals are inhibited until the requesting subsystem responds by asserting SI.

3-136. FLAG 3 REGISTER.

3-137. Flag 3 is a one-bit register located on IOP PCA A10 and shown on simplified diagram sheet 28 and detailed diagram DD207, location B44. Flag 3 is used to indicate the completion of a direct I/O operation. Flag 3 is set by the presence of SI, SO, or MSK RTRN, when IOFLG1 is set (generated when either the DDG or DAG signal goes high). Flag 3 is cleared by a CLFLAG 3 signal, generated by the special field decoder as a result of special field code CF3.

3-138. IOP SERVICE OUT LOGIC.

3-139. The IOP service out logic, located on IOP PCA A10 and shown on simplified diagram sheet 27 and detailed diagram DD207, sheet 5, contains the Service Out flip-flop and generates the SO signal for transmittal to the I/O subsystem.

3-140. IOP MULTIPLEXER CONTROL LOGIC.

3-141. The IOP multiplexer control logic is located on IOP PCA A10 and shown on simplified diagram sheet 26 and detailed diagram DD207, sheet 5. During multiplexed I/O operation, the IOP multiplexer control logic monitors the status of the input command lines and the HSREQ and SI signals from the multiplexer channels, and controls the input/output gates to the multiplexed input data, multiplexed output data, and direct input data/multiplexed memory address registers. In addition, the IOP multiplexer control logic issues the Data Poll signal to determine multiplexer channel priority.

3-142. MULTIPLEXED INPUT DATA REGISTER.

3-143. The multiplexed input data (MUXID) register is a 16-bit register located on IOP PCA A10 and shown on simplified diagram sheet 30 and detailed diagram DD207, location B23. During multiplexed I/O operation, the MUXID register acts as a buffer for data transferred from the IOP bus to the central data bus. The MUXID register has a left shift capability which is used to increment the DRT entry by two before it is returned to memory. The MUXID register is loaded by a LDENB signal from the Low Request Initializing (LOREQ INIT) flip-flop in the IOP multiplexer control logic. The shift is enabled by a CE signal from the Count Enable (CE) flip-flop in the IOP multiplexer control. The 16-bit content of the MUXID register is gated onto the central data bus by an IOHSEL signal from the I/O High Select (IOHSEL) flip-flop in the MCU I/O request/select logic.

3-144. MULTIPLEXED OUTPUT DATA REGISTER.

3-145. The multiplexed output data (MUXOD) register is a 16-bit register located on IOP PCA A10 and shown on simplified diagram sheet 30 and detailed diagram DD207, location D4. During multiplexed I/O operation, the MUXOD acts as a buffer for data transferred from the central data bus to the IOP bus. During a DRT fetch, the content of the MUXOD register is transferred to the MUXID register to be incremented by two (left shift) and used to replace the original DRT entry in memory. The MUXOD register input is enabled by an IOINP signal from the I/O In Process (IOINP) flip-flop in the MCU I/O request/select logic. The 16-bit output from the MUXOD register is gated to the IOP bus by a DRTG signal from the DRT Gate flip-flop in the IOP multiplexer control logic, or, during a DRT fetch, the output of the MUXOD register is gated to the MUXID register by a STODRT signal from the DRT 2 flip-flop in the IOP multiplexer control logic.

3-146. DIRECT INPUT DATA/MULTIPLEXED MEMORY ADDRESS REGISTER.

3-147. The direct input data/multiplexed memory address (DID/MUXMA) register is a 16-bit register located on IOP PCA A10 and shown on simplified diagram sheet 30 and detailed diagram DD207, location B25. During execution of direct I/O commands, the DID/MUXMA register serves as a buffer for data transferred from an I/O subsystem to the CPU. During multiplexed I/O operation, the DID/MUXMA register is the buffer for memory address words from the IOP bus to the central data bus. The store function of the DID/MUXMA register is enabled by an AENB signal from the IOP multiplexer control logic. Data from the IOP bus is gated into the DID/MUXMA register by the IODE signal from the IOP multiplexer control logic. During direct I/O operation, the 16-bit output from the DID/MUXMA register is gated to the S-bus by a RDIOD signal from the S-bus decoder. In the multiplexed I/O mode of operation, the output of the DID/MUXMA register is gated to the central data bus by an IOLOSEL signal from the I/O Low Select (IOLSEL) flip-flop in the MCU I/O request/select logic.

3-148. OPERAND REGISTER.

3-149. The operand (OPND) register is a 16-bit register located on CIR PCA A8 and shown on simplified diagram sheet 9 and detailed diagram DD205, location B36. The operand register provides storage for data read from memory by the CPU. The operand register is loaded by an OPINP signal from the Operand In Process (OPINP) flip-flop in the MCU operation decoder as a result of MCU options OPND, RNWA, RWA, and RWAN. The operand register is read by an RDOPND signal from the S-bus decoder as a result of an S-bus field code OPND as described in the field code definitions.

3-150. MCU I/O REQUEST/SELECT LOGIC.

3-151. The MCU I/O request/select logic, located on MCU PCA A9 and shown on simplified diagram sheet 31 and detailed diagram DD206, sheet 4, controls the operation of the I/O registers during address/data transfers between the CPU/IOP and memory. The MCU I/O request/select logic generates the IOINP signal, which loads the data on the central data bus into the MUXOD register; the IOLSEL signal, which gates the contents of the DID/MUXMA register to the central data bus; and the IOHSEL signal, which gates the contents of the MUXID register to the central data bus.

3-152. MCU ERROR LOGIC.

3-153. The MCU error logic, located on MCU PCA A9 and shown on detailed diagram DD206, sheet 2, detects a SYSPE or MCUDPE returned from memory and sends a MODINT and error signals to the CPU. An MCUDPE signal received from memory sets the MCU Address Parity Error (MCUD ADDR PE) flip-flop in the error logic, which sets bit 2 (address parity error) to the system interrupt register (CPX1) to logic 1. The SYSPE signal from memory clears the MCU System Parity Error (MCUSYS PE) flip-flop, setting bit 1 (system parity error) to the CPX1 register to a logic 1. CPX1 bit 3 (data parity error) is set to logic 1 when a DATAPE signal is received from the DATA PE flip-flop on CIR PCA A8. The Module Interrupt flip-flop in the MCU error logic is set when a parity error is generated by the MCU parity generator (DD206A14) and when the TO-FROM module numbers do not match. The MODINT sets bit 7 (module interrupt) of the interrupt register (CPX1) to a logic 1.

3-154. IOP INTERRUPT CONTROL AND ERROR LOGIC.

3-155. The IOP interrupt control and error logic is located on IOP PCA A10 and shown on simplified diagram sheet 29 and detailed diagram DD207, sheets 2 and 4. The interrupt control and error logic receives the INTREQ signal from the interrupt requesting I/O subsystem, issues the INTPOLL signal to determine the highest-priority request and, when INTACK is received, generates the IENB signal which loads the interrupting subsystem's address into the IDN register. When the Interrupt Acknowledge flip-flop is set by the INTACK signal, it in turn sets the External Interrupt flip-flop, which sets bit 8 (external interrupt) to the interrupt register (CPX1) to a logic 1.

3-156. The SYSPE and MCUDPE signals also are monitored by the IOP interrupt control and error logic during the I/O to memory transfer operations. Either the SYSPE or MCUDPE signal will set the Address Parity Error (APE) flip-flop which, when set, sends an IOAPE signal to the front panel indicators and a XERR signal to the transferring I/O subsystem, and the service out logic, causing the assertion of the SO signal.

3-157. CPX1 REGISTER.

3-158. The CPX1 register, shown on simplified diagram sheet 10, provides 16 bits that are used to monitor the system run mode interrupt status. When a run mode interrupt is experienced, the CPU reads the CPX1 register and checks the content for the cause of the interrupt. The S-bus field code CPX1 reads the CPX1 register and the special field code CCPX affects the CPX1 register as described in the field code definitions. The following is a list of the CPX1 register contents with the significance of each bit and the location of the output gate for each bit.

Bit 0.	Integer overflow	CIR PCA A8 (DD205E28)
Bit 1.	System parity error	MCU PCA A9 (DD206C19)
Bit 2.	Address parity error	MCU PCA A9 (DD206B19)
Bit 3.	Data parity error	MCU PCA A9 (DD206B19)
Bit 4.	CPU timer	MCU PCA A9 (DD206E19)
Bit 5.	Bounds violation	CIR PCA A8 (DD205E28)
Bit 6.	Illegal address	MCU PCA A9 (DD206F19)
Bit 7.	Module interrupt	CIR PCA A8 (DD205E28)
Bit 8.	External interrupt	CIR PCA A8 (DD205E28)
Bit 9.	Console interrupt	CIR PCA A8 (DD205E28)
Bit 10.	Power fail interrupt	CIR PCA A8 (DD205E28)
Bit 11.	0	CIR PCA A8 (DD205E28)
Bit 12.	0	CIR PCA A8 (DD205E28)
Bit 13.	0	MCU PCA A9 (DD206B39)
Bit 14.	0	MCU PCA A9 (DD206B39)
Bit 15.	0	MCU PCA A9 (DD206B39)

3-159. CPX2 REGISTER.

3-160. The CPX2 register, shown on simplified diagram sheet 10, provides 16 bits that are used to monitor the system halt mode interrupt status. When a halt mode interrupt is experienced, the CPU reads the CPX2 register and checks its content for the cause of the interrupt. The S-bus field code CPX2 reads the CPX2 register and the special field code CCPX affects the CPX2 register content as described in the field code definitions. The following is a list of the CPX2 register content with the significance of each bit and the location of the output gate for each bit.

Bit 0.	Run FF	CIR PCA A8 (DD205F23)
Bit 1.	Cold load	CIR PCA A8 (DD205F23)
Bit 2.	Single instruction	CIR PCA A8 (DD205E23)
Bit 3.	Load register	CIR PCA A8 (DD205F23)
Bit 4.	Display memory	CIR PCA A8 (DD205F23)
Bit 5.	Load memory	CIR PCA A8 (DD205F23)
Bit 6.	Execute switch	CIR PCA A8 (DD205E23)
Bit 7.	Increment memory address	CIR PCA A8 (DD205F23)

Bit 8.	Decrement memory address	CIR PCA A8 (DD205F23)
Bit 9.	Inhibit auto-restart	CIR PCA A8 (DD205F23)
Bit 10.	I/O timer	ALU PCA A5 (DD202H47)
Bit 11.	Interrupt stack flag	ALU PCA A5 (DD202G47)
Bit 12.	Dispatcher flag	ALU PCA A5 (DD202G47)
Bit 13.	0	ALU PCA A5 (DD202G47)
Bit 14.	0	CIR PCA A8 (DD205F23)
Bit 15.	System dump	CIR PCA A8 (DD205F23)

3-161. MASK REGISTER.

3-162. The mask register is a 16-bit register located on IOP PCA A10 and shown on simplified diagram sheet 30 and detailed diagram DD207, location B14. The mask register is a copy of all the individual flip-flops on all subsystem controllers. The Mask bit for any subsystem controller, when set to logic 0, holds off any interrupts from the controller. The held off interrupts are not lost and the controller is able to interrupt when the Mask is removed.

3-163. The mask register is loaded by a STMSK signal from the store field decoder and read by a RDMSK signal from the S-bus decoder.

3-164. NOP1, NOP2 LOGIC.

3-165. The NOP1, NOP2 logic is located on SSF PCA A4 and shown on simplified diagram sheet 24 and detailed diagram DD201, sheet 3. The NOP logic monitors the output of the skip field decoder and NO-OPS the ROR1 and ROR2 registers during skip conditions.

3-166. NEXT LOGIC.

3-167. The next logic is located on SSF PCA A4 and shown on simplified diagram sheet 25 and detailed diagram DD201, sheet 3. When a NEXT microinstruction is decoded by the skip field decoder, a NEXT signal is generated by the next logic and sent to the MCU operation decoder to initiate a next instruction fetch from memory.

3-168. MCU OPERATION DECODER.

3-169. The MCU operation decoder, located on MCU PCA A9 and shown on simplified diagram sheet 32 and detailed diagram DD206, sheet 3, decodes the MCU option code from ROR2 bits 23:27 and controls central data bus transmissions. The NIP and OPINP signals, both of which are generated by the MCU operation decoder, load the contents of the central data bus into the NIR and OPND registers, respectively, during next instruction and operand fetches from memory.

3-170. FREEZE LOGIC.

3-171. The freeze logic is located on MCU PCA A9 and shown on simplified diagram sheet 33 and detailed diagram DD206, location G13. If a fetch operation is in process or LREQ or HREQ flip-flops are set, the freeze logic inhibits the CPU clock (inhibiting further CPU operations), and prevents the LREQ and HREQ flip-flops in the CPU request/select logic and the Operand In Process (OPINP) and the Next In Process (NIP) flip-flops in the MCU control logic from being set until the fetch is completed. In addition, ROR1 is inhibited from loading the ROM output until the fetch is completed.

3-172. OVERFLOW FLIP-FLOP.

3-173. The Overflow flip-flop, located on SSF PCA A4 and shown on simplified diagram sheet 21 and detailed diagram DD201, location D8, controls the overflow bit (bit 4) of the status word. The Overflow flip-flop stores the state of the Overflow signal from the ALU, when OFCENB signal is true. The Overflow flip-flop is set by special field code SOV (Set Overflow) and cleared by the special field code CLO (Clear Overflow).

3-174. CARRY FLIP-FLOP.

3-175. The Carry flip-flop, located on SSF PCA A4 and shown on simplified diagram sheet 21 and detailed diagram DD201, location C8, controls the carry bit (bit 5) of the status word. The Carry flip-flop stores the state of the Carry signal from the ALU, when OFCENB signal is true. The Carry flip-flop is set by special field code SCRY (Set Carry) and cleared by the special field code CCRY (Clear Carry).

3-176. CONDITION CODE LOGIC.

3-177. The condition code logic, located on SSF PCA A4 and shown on simplified diagram sheet 20 and detailed diagram DD201, location E6, controls the condition code. Bits 6 and 7 of the status word are used for the condition code. Although several instructions make use of the condition code, the condition code typically indicates the state of an operand (or a comparison result with two operands). The operand may be a word, byte, double word, or triple word, and may be located on the top of the stack, in the index register, or in a specified memory location. Three codings are used: 00, 01, and 10. (The 11 combination is not used.) Except for special interpretations there are three basic patterns for interpreting these codes. The three patterns are shown in table 3-10.

Table 3-10. Condition Codes.

CCA sets CC = CCG (00) if operand > 0
= CCL (01) if operand < 0
= CCE (10) if operand = 0

CCB sets CC = CCG (00) if numerical (octal 060-071)
= CCL (01) if special char (all others)
= CCE (10) if alphabetic (upper 101 - 132
lower 141 -172)

CCC sets CC = CCG (00) if operands 1 > 2
= CCL (01) if operands 1 < 2
= CCE (10) if operands 1 = 2

3-178. The most common condition code pattern is pattern A, designated as condition code A, or CCA. The special field code CCA sets condition code pattern A. The condition code bits are set to CCG (00) if the T-bus word is greater than zero (T(0)=0 and signal T = 0 is false), CCL (01) if the T-bus word is less than zero (T(0)=1), and CCE (10) if the T-bus word is equal to zero (signal T = 0 is true). Since this usage of the condition code is so common, the three codes 00, 01, and 10 are named to reflect the meanings. Thus 00 is CCG (greater), 01 is CCL (less), and 10 is CCE (equal).

3-179. Condition code pattern B is used with byte oriented instructions and is controlled by special field code CCB. The special field code CCB sets the condition code bits to CCG (00) if bits 8:15 of the SP1 register form a numeric ASCII character, CCL (01) if a special ASCII character, and CCE (10) if an alphabetic ASCII character.

3-180. Condition code pattern C is used with comparison instructions and is controlled by special field codes CCG, CCL, and CCE. Special field code CCG sets the condition code bits to CCG (00) if operand 1 is greater than operand 2, special field code CCL sets the bits to CCL (01) if operand 1 is less than operand 2, and special field code CCE sets the bits to CCE (10) if operand 1 equals operand 2.

3-181. FUNCTIONAL-LEVEL DESCRIPTION.

3-182. The following paragraphs contain a functional-level description of operation for the CPU/IOP. The text is supported by operational flow diagrams. In addition, the block diagram (figure 3-2), simplified logic diagrams (figure 3-22) and detailed logic diagrams (set nos DD200 through DD207) should be referenced. For ease in using these diagrams, the location of all circuit elements (flip-flops, registers, etc) referenced on the flow diagrams is shown. The simplified or detailed diagram is listed above the box which references the circuit element.

3-183. The CPU, with the IOP and MCU, is connected to the memory modules via the central data bus. All transmissions to or from the CPU/IOP and memory are routed through the MCU. Access to the central data bus is granted when two conditions are met. First, the CPU/IOP requests service from the MCU. The destination memory module signifies it is ready by pulling its READY line low. There is one READY line for each memory module. The second condition that must be met is that there must not be any higher priority module seeking access to the bus. Bus priority is a function of data transfer urgency. Memory modules have higher priority than the CPU or IOP. When a module is ready for access to the bus, it pulls its ENABLE line low, blocking lower priority modules. The CPU and IOP share the MCU. Although the CPU and IOP both have access to the central data bus, the IOP has higher priority.

3-184. Data to or from the CPU/IOP and I/O subsystems is carried over the IOP bus. Input/output operations are divided into three categories: direct I/O, programmed I/O and interrupt processing. Programmed I/O operations have priority over other modes of operation.

3-185. Direct I/O operations take place as a result of the execution of an I/O instruction by the CPU. Each direct I/O operation either exchanges a word of information between the TOS register in the CPU and the I/O subsystem, or causes a control function to be executed. During the execution of direct I/O instructions the CPU performs the basic control functions such as assembling the I/O command, checking the status of the I/O subsystem, and exchanging a word of information between the TOS register and the I/O subsystem via the IOP and the IOP bus.

3-186. Programmed I/O operations transfer blocks of data between the I/O subsystem and memory, via the IOP and bypassing the CPU. This type of operation begins when the CPU decodes an SIO instruction and issues an SIO command to a particular I/O subsystem. The I/O subsystem, in conjunction with the multiplexer channel and the IOP, then executes the I/O program without further CPU control.

3-187. The interrupt mode of operation is based on a priority system. An interrupt poll determines priority for each I/O subsystem, and priority is established by the logical proximity of the I/O subsystem to the IOP. The 16-bit mask register stores information that masks off groups of interrupts. Each I/O subsystem is assigned to a particular mask group and is masked off interrupt priority with that group.

3-188. Polling is also used to determine priority for each multiplexer channel during programmed I/O operation. A data poll is sent out to all multiplexer channels and, again, priority is established based on the logical proximity of each multiplexer channel to the IOP. The data poll is separate from the interrupt poll, however, so that data priority for any I/O subsystem can be different from its interrupt priority.

3-189. As the CPU executes a program, it sequentially fetches instructions from memory. Each instruction is loaded into the next instruction register (NIR), from there to the current instruction register (CIR), and then to the look-up table (LUT) address decoder. The LUT address decoder and LUT ROM provide a starting address for the microprogram ROM. Each of the 170 machine instructions references a specific microprogram in the microprogram ROM. The microprogram ROM starting address is stored in the ROM address register (RAR) and is used to read out a microprogram from ROM to the ROM output register rank 1 (ROR1). The RAR is incremented after each ROM address is furnished to the microprogram ROM, thus the address in RAR always points to the next microinstruction in ROM for the program being executed. The software program is always executed in the following sequence: next instruction fetch, operand fetch or store, and execution. Memory address computation for a next instruction fetch or operand fetch or store is performed in the CPU. A normal fetch or store sequence, then, consists of determining the memory address where the information is to be read from or stored into, sending this address to memory with the appropriate memory operation code (Read-Restore or Clear-Write), and, finally, transferring the data between memory and the CPU.

3-190. NEXT INSTRUCTION FETCH.

3-191. A timing diagram for a next instruction fetch is shown in figure 3-3 and an operational flow diagram is shown in figure 3-4. Briefly, a NEXT field code is decoded by the skip field decoder, an address is sent to memory, and the next instruction is read out of this memory address and transferred to the CPU.

3-192. The skip field code NEXT sets ROR1 bits 15:19 to 11101 and sets the NXT1 signal to a logic 1. The NXT1 signal sets the Next 2 (NXT2) flip-flop (The RPTN1 and ROMFCN1 signals are both high) and the NXT2 signal goes low, setting the NXTDCD signal to logic 1. The P-register, which contains the absolute address of the next instruction to be fetched from memory, is loaded into the PCOR register.

3-193. The Flag 1 and Flag 2 flip-flops are cleared. If the Next Delayed (NXTDLD) flip-flop is set, the CPU clock is frozen for one cycle, allowing the previous operation to finish. This flip-flop is set when NEXT and DATA occur in the same line of microcode, and cleared at the same time that NEXT sets the LOREQ flip-flop.

3-194. P-register bits 0, 1, 2, 14, and 15, which had been loaded into the mapper, are used to generate the TO code of the memory module. This code is loaded into the PTO and PTO(NIR) registers by the LOADPTO(NIR) signal.

3-195. If the Next In Process (NIP), Low Request (LREQ) or High Request (HREQ) flip-flops are set, and if the INTRP signal is high, the FREEZE signal is set to "0" and the CPU is forced to wait until the NIP, LREQ, and HREQ flip-flops are cleared before proceeding with the current operation. If no interrupts are pending and if the Next Fetch Inhibit flip-flop is not set, the NIRTOCIR level goes to "1" and the NIR content is loaded into the CIR. ROR1 is NOP'ed and the NIR to CIR Delay flip-flop is set.

3-196. The skip field code NEXT forces the MCU option NIR and the MCU operation decoder generates a Read-Restore (RR) memory opcode (MOP). This code is loaded into the PUMOP register. The NIR flip-flop is set at this point, enabling the NIR register store function. The NXT=1 flip-flop is set, setting NXT=1 to "0" and LUTGATE to "1", and loading the LUT ROM into RAR. On the next clock cycle, the NXT=2 flip-flop is set and the NXT=2 and INCP levels go high, incrementing the P-register by 1, pointing to the next instruction in memory.

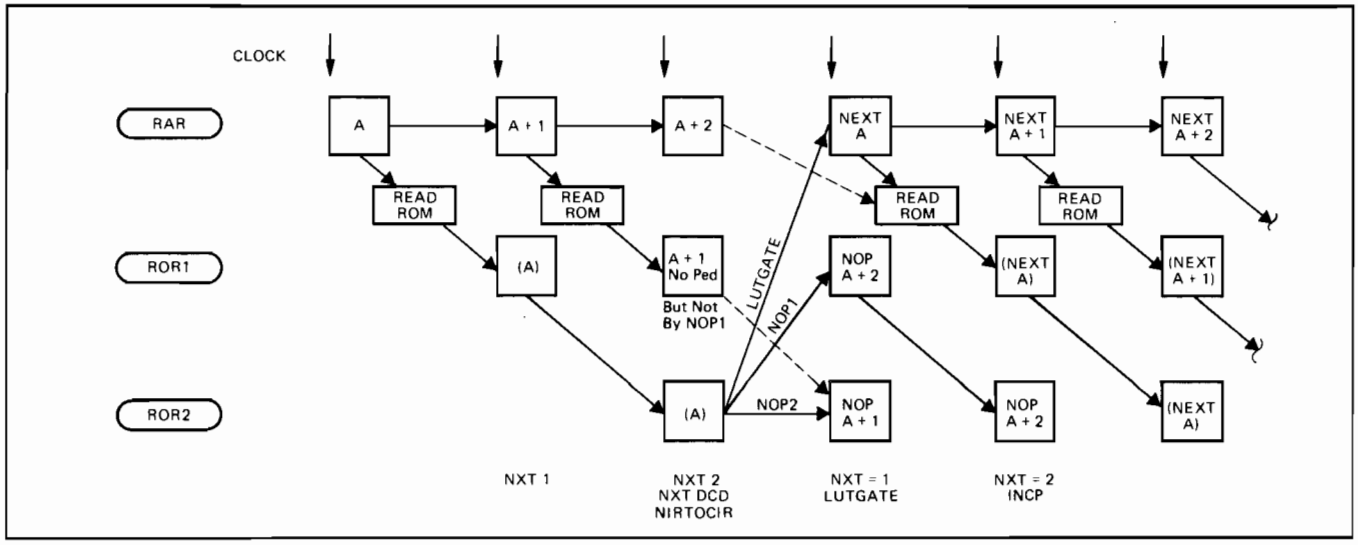


Figure 3-3. Execution of Microinstruction Containing Next Skip Field Code

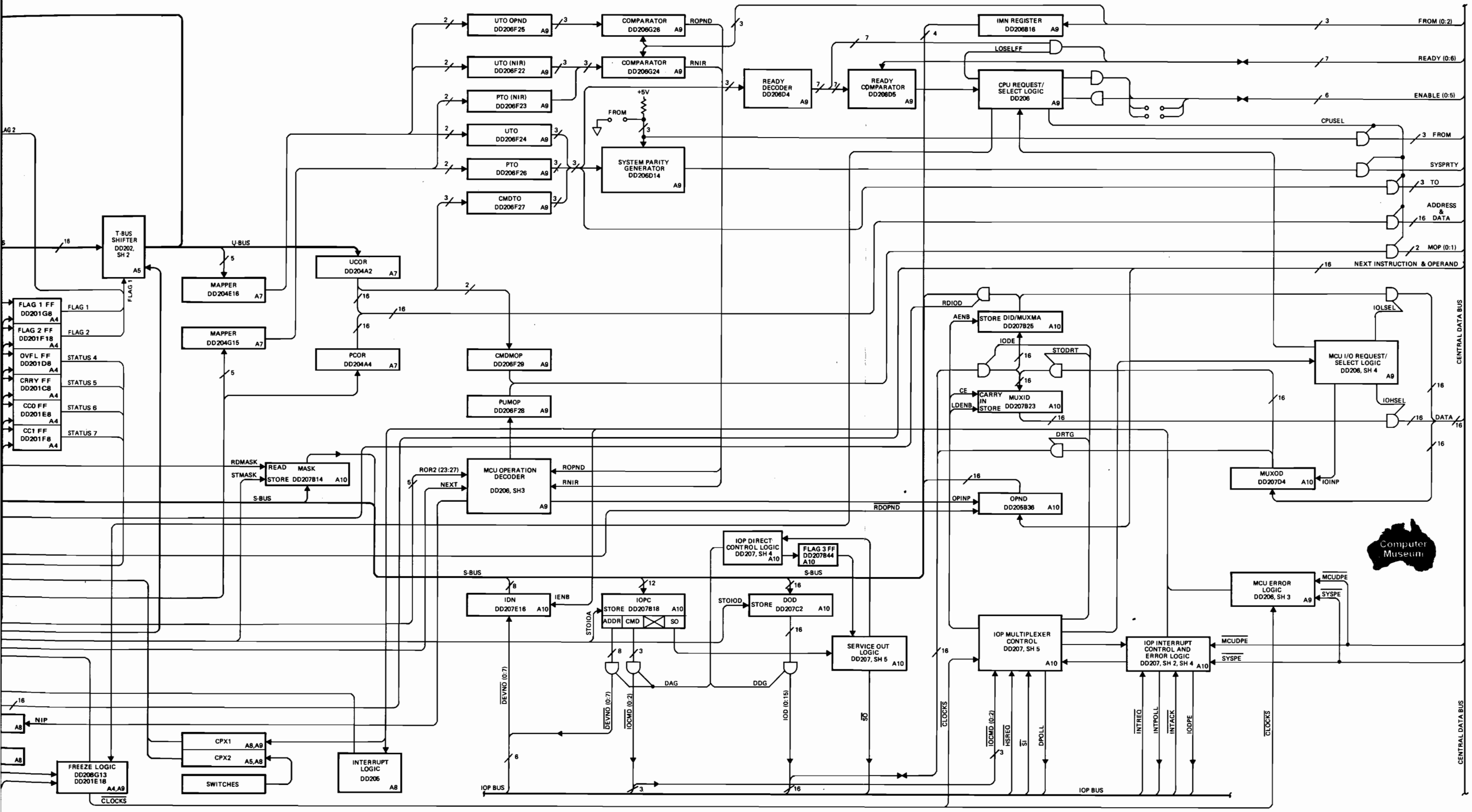
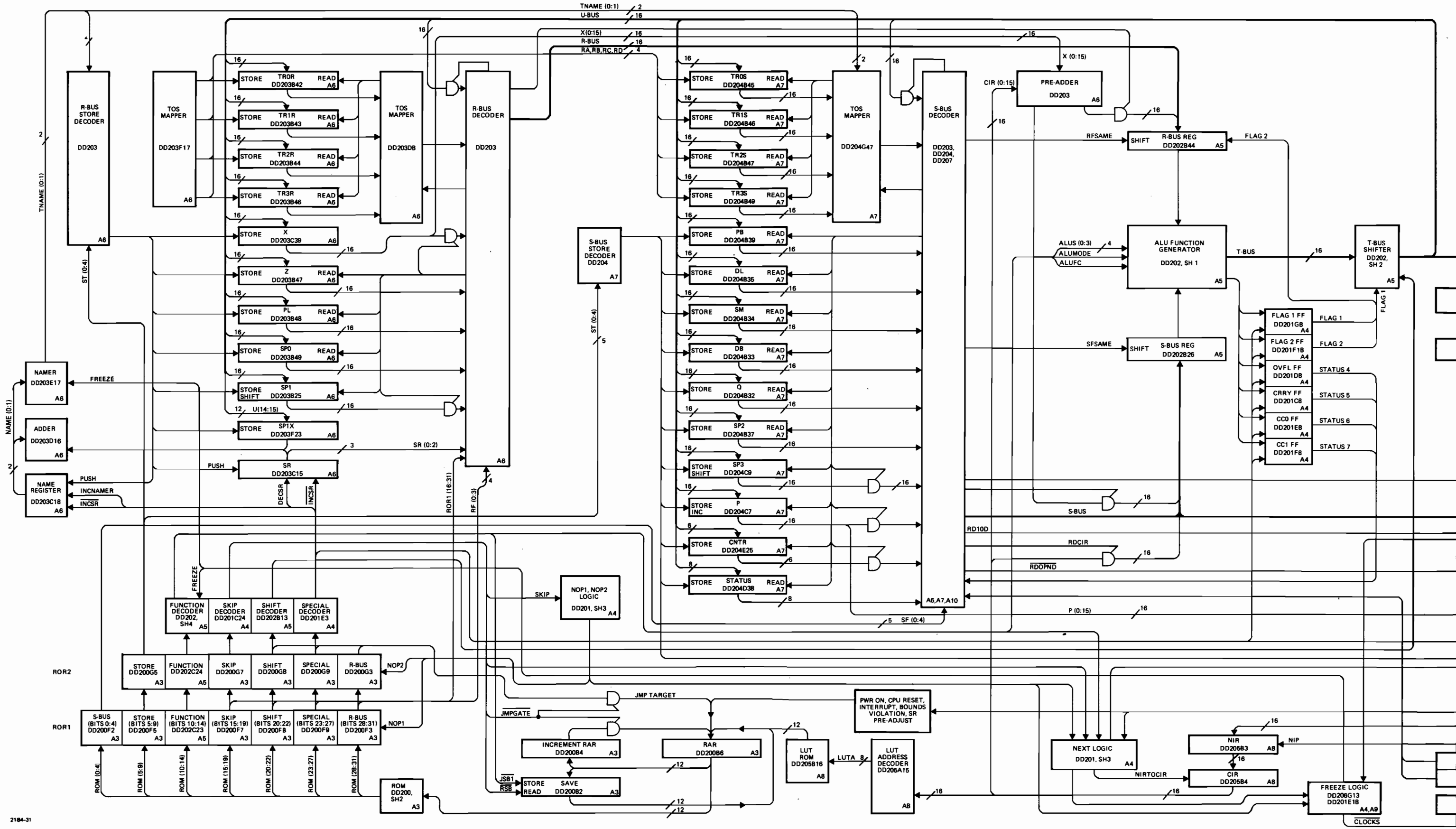


Figure 3-2. CPU/IOP Block Diagram


TNAME (0:1)
U-BUS
X(0:15)
R-BUS
RA,RB,RC,RD
R-BUS STORE DECODER DD203
TOS MAPPER DD203F17
TOS MAPPER DD203D8
R-BUS DECODER DD203
S-BUS STORE DECODER DD204 A7
TOS MAPPER DD204G47 A7
S-BUS DECODER DD203, DD204, DD207
X (0:15)
CIR (0:15)
PRE-ADDER DD203 A6
TR0R DD203B42
TR1R DD203B43
TR2R DD203B44
TR3R DD203B46
X DD203C39
Z DD203B47
PL DD203B48
SP0 DD203B49
SP1 DD203B25
SP1X DD203F23
SR DD203C15
TR0S DD204B45
TR1S DD204B46
TR2S DD204B47
TR3S DD204B49
PB DD204B39
DL DD204B35
SM DD204B34
DB DD204B33
Q DD204B32
SP2 DD204B37
SP3 DD204C9
P DD204C7
CNTR DD204E25
STATUS DD204D38
STORE
READ
SHIFT
INC
U(14:15)
SR (0:2)
ST (0:4)
RF (0:3)
ROR1 (16:31)
DECSR
INCSR
PUSH
FREEZE
NAMER DD203E17
ADDER DD203D16
NAME REGISTER DD203C18
INCNAMER
INCSR
RFSAME
SFSAME
R-BUS REG DD202B44
S-BUS REG DD202B26
SHIFT
ALU FUNCTION GENERATOR DD202, SH 1
ALUS (0:3)
ALUMODE
ALUFC
T-BUS
T-BUS SHIFTER DD202, SH 2
FLAG 1 FF DD201G8
FLAG 2 FF DD201F18
OVFL FF DD201D8
CRRY FF DD201C8
CC0 FF DD201E8
CC1 FF DD201F8
FLAG 1
FLAG 2
STATUS 4
STATUS 5
STATUS 6
STATUS 7
S-BUS
RD10D
RDCIR
RDOPND
P (0:15)
SF (0:4)
A6,A7,A10
FUNCTION DECODER DD202, SH4
SKIP DECODER DD201C24
SHIFT DECODER DD202B13
SPECIAL DECODER DD201E3
NOP1, NOP2 LOGIC DD201, SH3
SKIP
NOP2
NOP1
ROR2
ROR1
STORE DD200G5
FUNCTION DD202C24
SKIP DD200G7
SHIFT DD200G8
SPECIAL DD200G9
R-BUS DD200G3
S-BUS (BITS 0:4) DD200F2
STORE (BITS 5:9) DD200F5
FUNCTION (BITS 10:14) DD202C23
SKIP (BITS 15:19) DD200F7
SHIFT (BITS 20:22) DD200F8
SPECIAL (BITS 23:27) DD200F9
R-BUS (BITS 28:31) DD200F3
ROM (0:4)
ROM (5:9)
ROM (10:14)
ROM (15:19)
ROM (20:22)
ROM (23:27)
ROM (28:31)
ROM DD200, SH2
JMP TARGET
JMPGATE
INCREMENT RAR DD20084
RAR DD20086
STORE SAVE DD20082
JSB1
RSB
PWR ON, CPU RESET, INTERRUPT, BOUNDS VIOLATION, SR PRE-ADJUST
LUT ROM DD205816
LUTA
LUT ADDRESS DECODER DD205A15
NEXT LOGIC DD201, SH3
NIRTOCIR
NIR DD20583
CIR DD20584
NIP
FREEZE LOGIC DD208G13 DD201E18
CLOCKS
2184-31

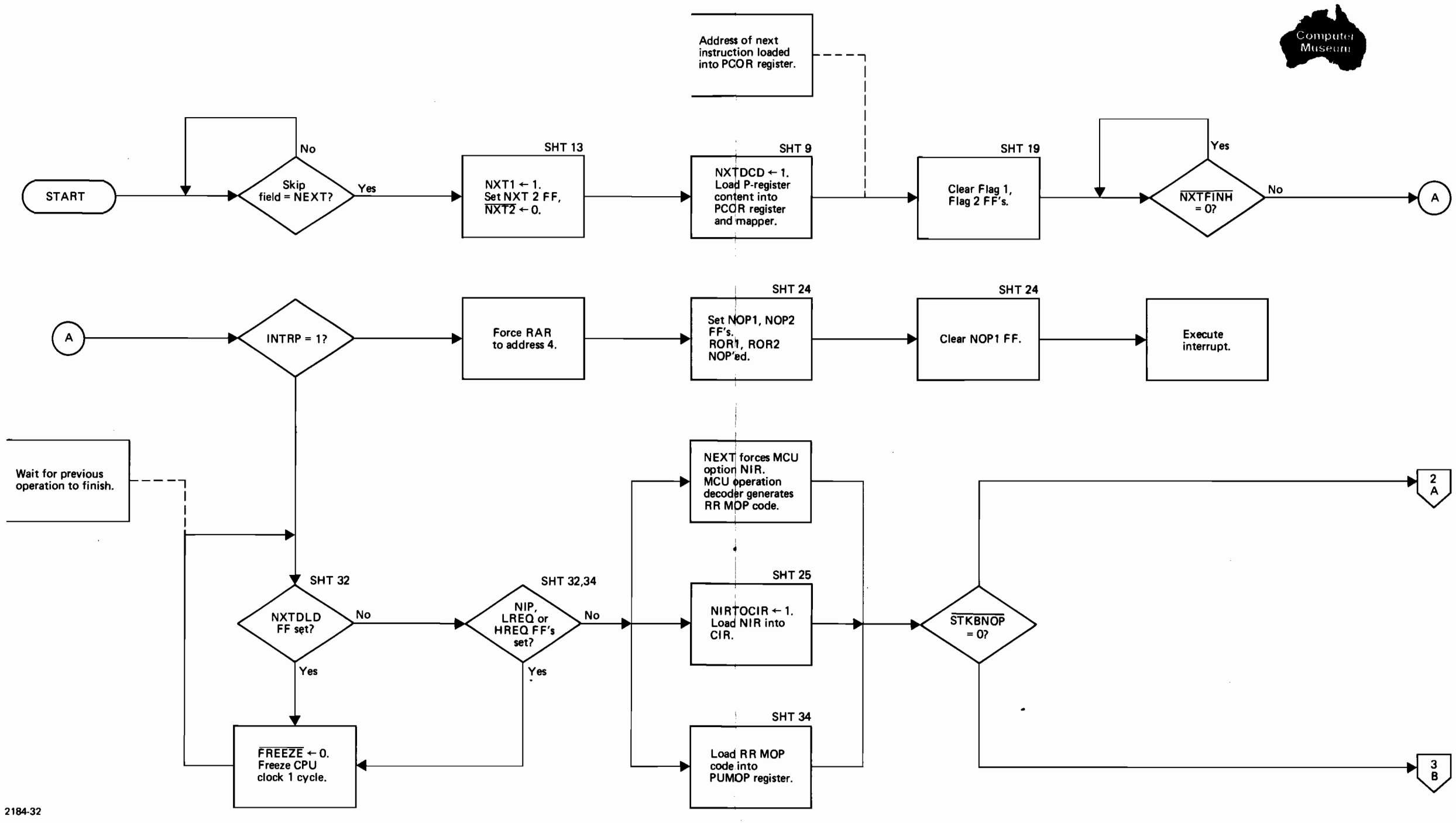


Figure 3-4. Next Instruction Fetch Operational Flow Diagram (Sheet 1 of 4)

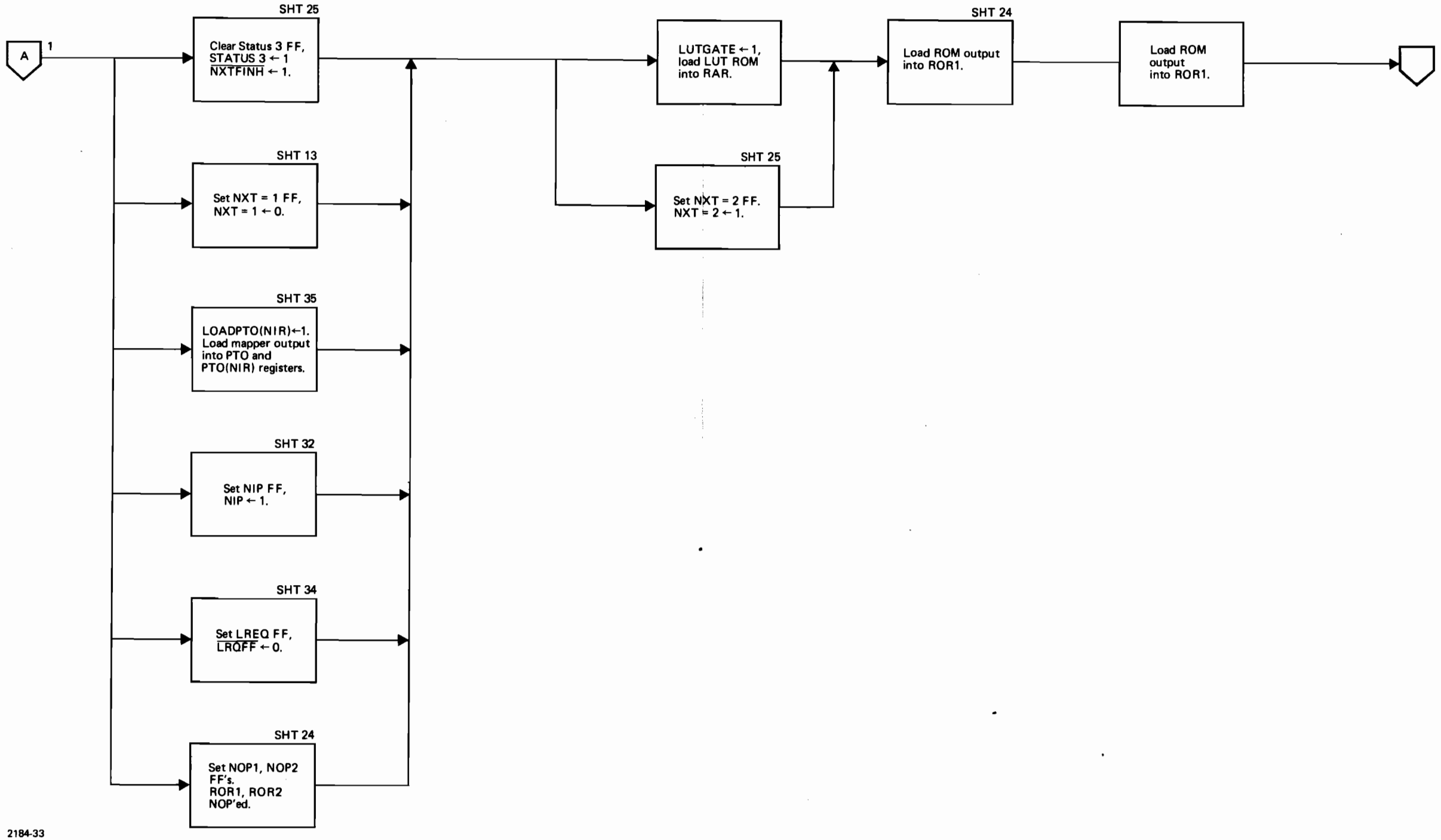
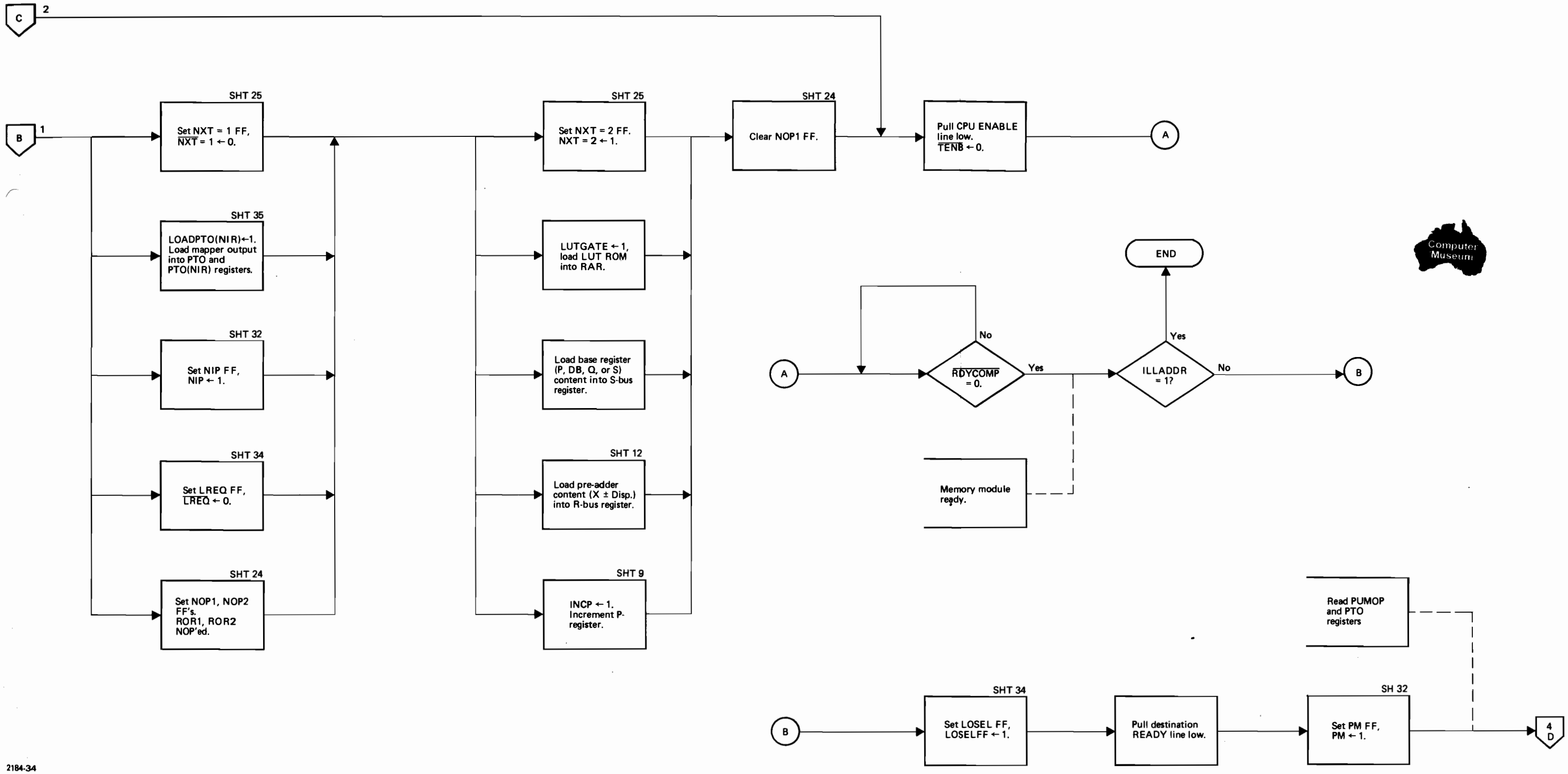


Figure 3-4. Next Instruction Fetch Operational Flow Diagram (Sheet 2 of 4)



2184-34

Figure 3-4. Next Instruction Fetch Operational Flow Diagram (Sheet 3 of 4)

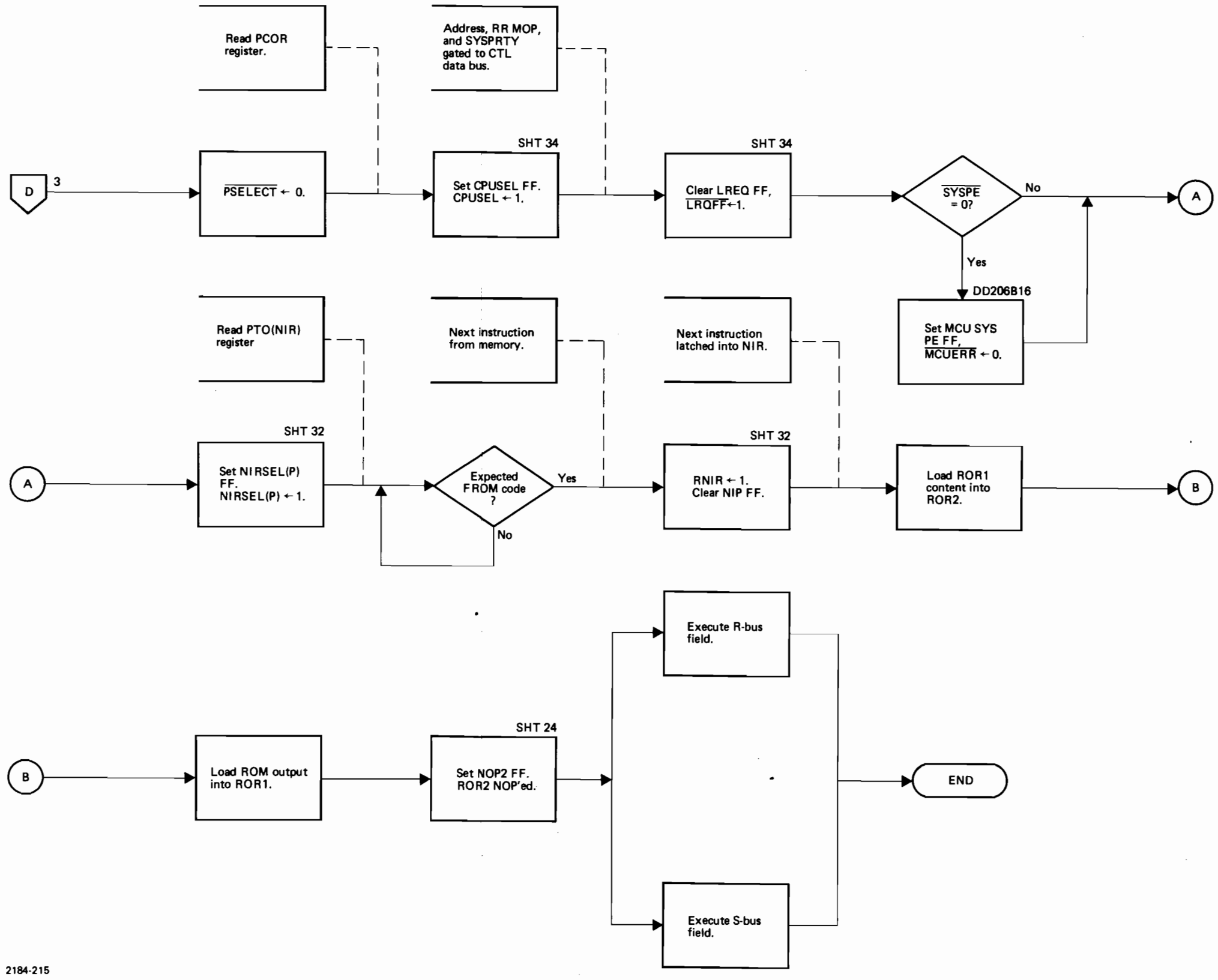


Figure 3-4. Next Instruction Fetch Operational Flow Diagram (Sheet 4 of 4)

3-197. The base register (P, DB, Q, or S) called for by the instruction is loaded into the S-bus register and the pre-adder, containing the relative address portion of the CIR, is loaded into the R-bus register. The U-bus becomes the sum of the S-bus plus the R-bus register contents and the result is loaded into the S-bus register, if the microcode so specifies. The NOP2 flip-flop remains set for one clock cycle after the NXT=2 flip-flop is cleared.

3-198. When a NEXT code is decoded, the CPU sets the LREQ flip-flop. The ready comparator compares the READY lines against the TO code for the intended memory module and, if the memory is ready to receive, the CPU pulls the TENB signal low. If the maximum memory bounds are exceeded, the ILLADDR signal goes high, terminating the operation. If ILLADDR is not high, the Low Select (LOSEL) flip-flop is set, and the ENABLE line is released. The P-to-Memory (PM) flip-flop is set, the PM level goes high and PSELECT goes low. The PM level reads the PUMOP register, containing the RR MOP code, and the PTO register, containing the TO code of the memory module. The PSELECT level reads the PCOR register, which contains the memory address. The CPU Select (CPUSEL) flip-flop is set and the memory address, RR MOP code, and SYSPRTY are gated to memory via the central data bus. The LREQ flip-flop is cleared. If a SYSPE signal is received, the MCU System Parity Error (MCU SYS PE) flip-flop is set and the MCUERR level is set to "0".

3-199. The NIRSEL(P) level, set to "1" by the NIR Select (P) flip-flop, reads the contents of the PTO(NIR) register to the TO-FROM comparator, which compares the saved TO code with the incoming FROM code from the memory module. If the two match, the RNIR level goes high, clearing the NIP flip-flop and latching the next instruction into NIR.

3-200. OPERAND FETCH.

3-201. An operand fetch operational diagram is shown in figure 3-5. The procedure for fetching an operand is similar to the procedure for fetching the next instruction. The main differences are that an operand fetch is initiated by a store field code BUSL and the operand is stored in the operand register instead of the next instruction register.

3-202. When a store field code BUSL is decoded, the special field is disabled and the MCU option field is enabled. If a previous MCU operation is still in process, the FREEZE signal will be low until the operation is completed. The U-bus, containing the memory address, is loaded into the UCOR register and mapper (bits 0, 1, 2, 14, and 15) and the RR MOP code is loaded into the PUMOP register.

3-203. The TO code, generated by the mapper, is loaded into the UTO and UTO OPND registers. The LREQ and Operand In Process (OPINP) flip-flops are set. The OPINP flip-flop enables the OPND register store function so that this register will load all central data bus transmissions. The READY comparator compares the TO lines with the READY lines to assure that the destination module is ready to accept an address. If the result is true, the CPU ENABLE line is pulled low, setting TENB to "0". The LOSEL flip-flop is set, releasing the ENABLE line and clearing the LREQ flip-flop. The P-to-Memory (PM) flip-flop is set, setting the PM level to "1" and PSELECT to "0", and reading the PUMOP and UCOR registers. When the CPUSEL flip-flop is set, the memory address and RR MOP code are gated to memory.

3-204. The saved TO code, which had been stored in the UTO OPND register, is compared with the incoming FROM code by the TO-FROM comparator. If the result is true, the ROPND level is set to "1", clearing the OPND flip-flop and latching the operand into the OPND register.

3-205. OPERAND STORE.

3-206. An operand store operation is identical to an operand fetch operation except that there are two transmissions to memory from the CPU, an address and data. In addition, the MOP code is Clear-Write instead of Read-Restore. The address is loaded into the UCOR register and sent to memory and, when the LOSEL flip-flop is reset at the completion of the address transfer cycle, the operand is loaded into UCOR and sent to memory. An operational flow diagram for an operand store function is shown in figure 3-6.

3-207. DIRECT I/O.

3-208. The CPU has eight instructions, or commands, for direct I/O control. The commands are three-bit codes carried over the IOCMD00, IOCMD01 and IOCMD02 lines. The commands and the three-bit IOCMD codes are shown in table 3-11.

Table 3-11. Direct I/O Commands

I/O COMMAND	CODE		
	IOCMD02	IOCMD01	IOCMD00
Set Interrupt (SIN)	0	0	0
Reset Interrupt (RIN)	0	0	1
Start I/O (SIO)	0	1	0
Set Mask (SMSK)	0	1	1
Control I/O (CIO)	1	0	0
Test I/O (TIO)	1	0	1
Write I/O (WIO)	1	1	0
Read I/O (RIO)	1	1	1

3-209. The following paragraphs describe the direct I/O commands and those subroutines which are called during execution of the commands. The descriptions are supported by operational flow diagrams. The text presents an overall description of operation for the various I/O commands; the detailed, step-by-step sequence of events is contained in the flow diagrams and it is these diagrams which should be consulted if a detailed description is required. The microprogram listing should be used in conjunction with the flow diagrams to follow the sequence of operation for the CPU/IOP. Simplified diagrams are referenced on the flow diagrams for key circuit elements.

3-210. READ I/O. A Read I/O (RIO) instruction transfers data from the I/O subsystem to the TOS in the CPU. The I/O subsystem device number is fetched from the stack at location S-K and a Test I/O (TIO) command is sent to this I/O subsystem. The I/O subsystem returns its status to the CPU, where it is checked to determine if the I/O subsystem is ready for RIO. If it is ready (status word bit 0=1), an RIO command is sent to the I/O subsystem, the incoming data word is loaded onto the stack, and the condition code is set to CCE. If the I/O subsystem is not ready (status word bit 0=0), the status word from the I/O subsystem is pushed onto the stack and the condition code is set to CCG. If the I/O subsystem does not respond to the TIO (I/O timeout), the condition code is set to CCL and the instruction is terminated.

3-211. An operational flow diagram for a RIO command is shown in figure 3-7. If the instruction in CIR contains an RIO, the LUT decoder generates a microprogram ROM address of 2252_8. The microinstruction at this ROM address is loaded into ROR1 and a function field JSB and skip field code UNC cause a jump to the device address fetch subroutine (FS-K). The device address fetch subroutine (see figure 3-8) fetches the device address (DEVNO) from the stack at location S-K. The address value S is obtained by adding the contents of the SR register to the SM register, the constant K (contained in bits 12:15 of the instruction) is subtracted from this value to point to the stack address where DEVNO is located.

3-212. The device address (DEVNO), when fetched, is loaded, with a Test I/O (TIO) command, into the SP3 register and is transferred to the IOP when the CIOP subroutine (see figure 3-9) is executed. The TIO IOCMD and DEVNO are loaded into the IOPC register in the IOP and are transferred, with SO, to the I/O subsystem. The I/O subsystem decodes the TIO IOCMD and returns its status and SI to the IOP. The IOP terminates SO and loads the status word into the DID/MUXMA register, and sets the Flag 3 flip-flop, notifying the CPU that the status word is present. An RDIOD signal from the CPU gates the DID/MUXMA register contents to the S-bus register and from there to the U-bus. A store field code, SP2, loads the U-bus (containing the status word) into the SP2 register.

3-213. If the I/O subsystem has not responded by the time the I/O timeout occurs, CPX2 bit 10 will be set. The microcode checks the status of CPX2 bit 10 and sets Flag 2 flip-flop if the bit is logic 1. If Flag 2 flip-flop is set (signifying I/O timeout), a jump to ROM address 2251_8 (I/O Kill) is executed, the condition code is set to CCL, and the operation is terminated. If the Flag 2 flip-flop is not set, the status word is gated from the SP2 register to the S-bus register and bit 0 is checked. If bit 0 is logic 1 (I/O subsystem not ready for WIO), a jump to ROM address 2247_8 (I/O Abort) is executed, the status word is pushed onto the stack, the condition code is set to CCG, and the operation is terminated.

3-214. If the status word bit 0 is "0", a RIO IOCMD is formed and loaded into the SP3 register for transfer to the IOP. The CIOP subroutine is executed once more, transferring the RIO IOCMD, DEVNO and SO to the I/O subsystem via the IOP bus. The I/O subsystem responds with a data word and SI. The IOP loads the data word into the DID/MUXMA register, terminates SO and sets Flag 3 flip-flop, notifying the CPU that the data word is present. The CPU generates the RDIOD signal and the data word is gated to the S-bus register and from there to the U-bus. A store field code SP2 loads the data word on the U-bus into the SP2 register.

3-215. Flag 2 flip-flop is checked again and if it is set (signifying I/O timeout), a jump to ROM address 2251_8 (I/O Kill) is executed, the condition code is set to CCL and the operation is terminated. If Flag 2 flip-flop is not set, the SR register content is checked to determine if the TOS registers are full. If the SR register content equals four, denoting that the TOS registers are full, a jump to the PSHM subroutine (ROM address 3425_8) is executed. The PSHM subroutine (see figure 3-10) pushes one word from the TOS registers down to memory to make room for the incoming data word from the I/O subsystem.

3-216. The data word from the I/O subsystem is gated from the SP2 register into the S-bus register, and from the S-bus register to the U-bus. When a store field code PUSH is decoded, the data word is loaded into the TOS register. The condition code is then set to CCE, a next instruction fetch is initiated, and the RIO instruction is complete.

3-217. FS-K SUBROUTINE. An operational flow diagram for the FS-K subroutine is shown in figure 3-8. FS-K is a subroutine to fetch the contents at address S-K (the I/O subsystem device number). The S quantity is obtained by adding the contents of the SR register to the SM register; K is a constant contained in bits 12 through 15 of the instruction.

3-218. If the content of the SR register is less than or equal to the constant K, the device address is in memory and an operand fetch is initiated. The address in memory is contained in the SM register and is gated from this register to the S-bus register. An operand fetch is then initiated to fetch the contents of memory location S-K (see paragraph 3-201).

3-219. The device address is received from memory and loaded into the OPND register. The S-bus field code OPND reads the OPND register out to the S-bus and into the S-bus register. A shift field code RRZ places the right byte of the T-bus (containing the device address) in the right byte of the U-bus (the left byte is set to all zeros) and, finally, the store field code SP1 loads the U-bus into the SP1 register.

3-220. If the content of the SR register is not less than or equal to the constant K, the device address is in a TOS register and an R-bus field code MREG loads this TOS register into the S-bus register. The shift field code RRZ loads the right byte of the word into the T-bus (the left byte is not used for device address and is set to all zeros), and a store field code SP1 loads these eight bits of the right byte (device address) into the SP1 register.

3-221. CIOP SUBROUTINE. An operational flow diagram for the CIOP subroutine is shown in figure 3-9. CIOP is a subroutine that performs the CPU-IOP communication. The CPU checks the Flag 2 flip-flop to determine if an outbound transfer is in progress, and, if Flag 2 is set (signifying an outbound transfer), the data word which had been stored in the SP2 register is loaded onto the S-bus. If the SIO Active (SIO ACT), Data Poll (DPOLL), or Interrupt Acknowledge (INTACK) flip-flops are set, signifying SIO or interrupt in process, the CIOP subroutine waits for the completion of these operations by repeatedly checking for Flag 1. When these flip-flops are all cleared, the CIOP subroutine can proceed and a store field code IOD (if an outbound transfer is in process) sets the STOIOD level to "1" and loads the data word on the S-bus into the DOD register. The store field code IOA sets the STOIOA level to "1" and the IOCMD and DEVNO are loaded into the IOPC register. The Direct Active (DRCT ACT) flip-flop is set, gating the IOPC register content (IOCMD and DEVNO) to the IOP bus. If an outbound transfer is in process, the Direct Data (DRCT DATA) flip-flop also is set and the DOD register content is gated to the IOP bus.

3-222. When the I/O subsystem receives the IOCMD, DEVNO and SO (which has been asserted by the IOP), it loads the data on the IOP bus into its data-in register and returns SI if an outbound transfer is in process, or gates a status word or data word onto the IOP bus with SI if an inbound transfer is in process. In the latter case (inbound transfer), the IOP loads the incoming data into the DID/MUXMA register, terminates SO, and sets the Flag 3 flip-flop, notifying the CPU that the data word has been received. The S-bus field code IOD sets the RDIOD logic level to "1" and the DID/MUXMA register content is gated to the S-bus register.

3-223. PSHM SUBROUTINE. An operational flow diagram for the PSHM subroutine is shown in figure 3-10. PSHM is a subroutine which pushes one word from the TOS registers down to memory. The SM register, which contains the address of the top element of the data stack in memory, is incremented and loaded into the S-bus register. An operand store operation (see paragraph 3-206) is initiated that stores the content of the lowest TOS register into the memory location contained in the SM register.

3-224. The top memory location available to the data stack is checked by loading the Z-register content into the R-bus register, the SM register content into the S-bus register, and subtracting the S-bus register content from the content of the R-bus register. If the SM register content exceeds the Z-register content (ALU carry out bit equals "0") a jump to ROM address 3432_8 (stack overflow subroutine) is executed.

3-225. PUL1 SUBROUTINE. An operational flow diagram of the PUL1 subroutine is shown in figure 3-11. The PUL1 subroutine pulls one word from the data stack in memory to a TOS register. The address of the data stack top element is contained in the SM register. An S-bus field code SM loads the SM register content (memory address) into the S-bus register and an operand fetch operation is initiated, using BUSL in the store field (see paragraph 3-201). The SM register is decremented to contain the address of the new data stack top element, and loaded into the R-bus register and the data base (DB) register is loaded into the S-bus register. An R-bus minus S-bus subtract function is executed and if the SM register content is less than the address contained in the DB register, the ALU carry out bit will equal "1", signifying a stack underflow and a jump to ROM address 2772_8 (stack underflow subroutine) is executed.

3-226. When the data word is received from memory an S-bus field code OPND loads the OPND register into the S-bus register and onto the U-bus. The U-bus data (the word just received from memory) is loaded onto the TOS at the location SM plus one, and the SR register is incremented to point to this TOS location.

3-227. WRITE I/O. A Write I/O (WIO) instruction transfers data from a TOS register to an I/O subsystem. An I/O subsystem device number is fetched from the stack at location S-K and a TIO command is sent to this I/O subsystem. The I/O subsystem returns its status to the CPU where it is checked to determine if the I/O subsystem is ready for WIO. If the I/O subsystem is ready for WIO (status word bit 0=1), a data word is transmitted from the TOS to the I/O subsystem with a WIO command and the condition code is set to CCE. If the I/O subsystem is not ready for WIO (status word bit 0=0), the status word from the I/O subsystem is pushed onto the stack and the condition code is set to CCG. If the I/O subsystem has not responded to the TIO command or to the WIO command (I/O timeout), the condition code is set to CCL and the instruction is terminated.

3-228. An operational flow diagram for a WIO instruction is shown in figure 3-12. When the machine instruction in CIR is a WIO, the LUT decoder generates a microprogram ROM address of 2265_8 and the microinstruction at this ROM address is loaded into ROR1. The function field code JSB and skip field code UNC cause a jump to the device address fetch (FS-K) subroutine (see figure 3-8). The FS-K subroutine fetches the device number from the stack at location S-K.

3-229. The device address (DEVNO), when fetched, is loaded, with a TIO command, into the SP3 register and is transferred to the IOP when the CIOP subroutine (see figure 3-9) is executed. The TIO IOCMD and DEVNO are loaded into the IOPC register, and are transferred, with SO, to the I/O subsystem. The I/O subsystem decodes the TIO IOCMD and returns its status and SI to the IOP. The IOP terminates SO and loads the status word into the DID/MUXMA register and sets the Flag 3 flip-flop, notifying the CPU that the status word is present. An RDIOD level from the CPU gates the DID/MUXMA register contents to the S-bus register and from there to the U-bus. A store field code SP2 loads the U-bus (containing the status word) into the SP2 register.

3-230. If the I/O subsystem has not responded to the TIO command, Flag 2 flip-flop will be set. If Flag 2 flip-flop is set (signifying I/O timeout), a jump to ROM address 2251_8 (I/O Kill) is executed, the condition code is set to CCL, and the instruction is terminated. If the Flag 2 flip-flop is not set, the status word is gated from the SP2 register to the S-bus register and bit 1 is checked. (Bit 2 is shifted left and becomes bit 0.) If bit 0 is logic 1 (I/O subsystem not ready for WIO), a jump to ROM address 2247_8 (I/O Abort) is executed, the status word is pushed onto the stack, the condition code is set to CCG, and the instruction is terminated.

3-231. If the I/O subsystem is ready for WIO (bit 1=0), the condition of the stack is checked before the WIO IOCMD is formed and the data word is transferred. If the SR register equals zero, a jump to ROM address 3420_8 (PUL1 subroutine, see figure 3-11) is executed and a data word is pulled from memory to a TOS register.

3-232. The WIO IOCMD is formed and loaded into the SP3 register and an S-bus field code RA loads the TOS register (containing the data word for transfer to the I/O subsystem) into the S-bus register and from there to the SP2 register. The CIOP subroutine is executed again and the WIO IOCMD and data word are transferred to the IOP. The IOP loads the WIO command into the IOPC register and the data word into the DOD register. The WIO IOCMD, DEVNO and data word are then gated to the IOP bus with SO. The addressed I/O subsystem decodes the WIO command and loads the data word into its input register, then returns SI to the IOP.

3-233. After the return from the CIOP subroutine, the CPU checks the Flag 2 flip-flop. If this flip-flop is set, denoting I/O timeout, the condition code is set to CCL and the instruction is terminated. If Flag 2 flip-flop is not set, the condition code is set to CCE. The stack is POP'ed, the SR register is decremented and the name register is incremented to point to the new TOS register. A next instruction fetch is initiated and the WIO instruction is complete.

3-234. TEST I/O. A Test I/O (TIO) instruction fetches an I/O subsystem device number from the stack at location S-K and sends a TIO command to this I/O subsystem. The I/O subsystem responds by gating its status to the CPU. The status word is pushed onto the stack and the condition code is set to CCE. If the I/O subsystem does not respond to the TIO command (I/O timeout), the condition code is set to CCL and the instruction is terminated.

3-235. An operational diagram for a TIO instruction is shown in figure 3-13. A TIO instruction causes the LUT decoder to generate a microprogram ROM address of 2300_8 and the microinstruction at this ROM address is loaded into ROR1. The device address is fetched by executing an FS-K subroutine (see figure 3-8) and is loaded into the SP1 register, and from there to the SP3 register with the TIO IOCMD. A CIOP subroutine (see figure 3-9) transfers the TIO command and DEVNO to the IOP. The TIO command, DEVNO, and SO are then transferred to the I/O subsystem by the IOP. The I/O subsystem decodes the TIO command and its DEVNO and returns a status word to the IOP.

3-236. The IOP loads the status word into the DID/MUXMA register, terminates SO, and sets the Flag 3 flip-flop, notifying the CPU that the status word has been received. The CPU issues the RDIOD logic level, which gates the contents of the DID/MUXMA register to the S-bus register and from there a store field code SP2 loads the status word into the SP2 register.

3-237. If the I/O subsystem has not responded to the TIO command, an I/O timeout will have occurred and the Flag 2 flip-flop will be set. If this flip-flop is set, a jump to ROM address 2251_8 is initiated and an I/O Kill subroutine is executed. The condition code is set to CCL and the instruction is terminated. If the Flag 2 flip-flop is not set, the CPU checks the content of the SR register to determine if the TOS registers are full. If the SR register content equals four, a jump to ROM address 3425_8 is executed (PSHM subroutine, see figure 3-10) and one word from the TOS registers is pushed down to memory.

3-238. An S-bus field code SP2 loads the Status word from the SP2 register into the S-bus register and the store field code PUSH loads the status word into the TOS register. The condition code is set to CCE and the TIO instruction is complete.

3-239. CONTROL I/O. A Control I/O (CIO) instruction obtains an I/O subsystem device number from the stack at location S-K and sends a control word from the TOS, and a CIO command, to the I/O subsystem. If the I/O subsystem responds to the CIO command and control word, the TOS is POP'ed and the condition code is set to CCE. If the I/O subsystem does not respond (I/O timeout), the condition code is set to CCL and the instruction is terminated.

3-240. An operational flow diagram for a CIO instruction is shown in figure 3-14. A CIO instruction causes the LUT decoder to generate a microprogram ROM starting address of 2306_8 and the microinstruction at this address is loaded into ROR1. A function field code JSB and a skip field code UNC cause a jump to ROM address 3533_8 (FS-K subroutine, see figure 3-8), and the device address is fetched from the stack and stored in the SP1 register. An S-bus field code SP1 gates the SP1 register to the S-bus register, and, after the CIO command is formed, the device address and CIO command are loaded into the SP3 register for transfer to the IOP.

3-241. The CPU determines if the control word is in a TOS register or in memory by checking the SR register. If SR equals zero, the control word is in memory and a PUL1 subroutine (see figure 3-11) is executed which pulls one word from memory and loads it in the TOS register. An S-bus field code RA loads the control word from the TOS register into the S-bus register and a store field code SP2 gates it from the S-bus register to the SP2 register for transfer to the IOP. When the CIOP subroutine (see figure 3-9) is executed, the CIO command, DEVNO and control data word are sent to the IOP which transfers this information, with SO, to the I/O subsystem. The addressed I/O subsystem loads the control word into its control register and returns SI to the IOP. The IOP terminates SO and sets the Flag 3 flip-flop, notifying the CPU that the transfer is complete.

3-242. When the CPU returns from the CIOP subroutine, the Flag 2 flip-flop is checked to determine if I/O timeout has occurred. If the flip-flop is set, a jump to ROM address 2251_8 is executed (I/O Kill subroutine), the condition code is set to CCL and the instruction is terminated. If Flag 2 flip-flop is not set, the condition code is set to CCE. The stack is POP'ed, and the SR register is decremented and the name register is incremented to point to the new TOS register.

3-243. SET INTERRUPT. The Set Interrupt (SIN) instruction sends a Set Interrupt command to an addressed I/O subsystem. The SIN command sets an Interrupt Request flip-flop in the I/O subsystem's device controller, causing the I/O subsystem to send an INTREQ to the IOP (unless masked).

3-244. An operational flow diagram for a SIN instruction is shown in figure 3-15. The SIN instruction causes the microinstruction at ROM address 2323_8 to be loaded into ROR1. The address of the I/O subsystem which is to receive the SIN command is obtained by executing an FS-K subroutine (see figure 3-8). An S-bus field code SP1 loads the SP1 register (which now contains the I/O subsystem address) into the S-bus register and, after the SIN IOCMD is formed, the I/O subsystem address and SIN command are loaded into the SP3 register. The CIOP subroutine (see figure 3-9) transfers DEVNO and the SIN command to the IOP where they are stored in the IOPC register. The IOP transfers DEVNO and the SIN IOCMD to the IOP bus and the addressed I/O subsystem.

3-245. If the I/O subsystem has not responded to the SIN command, the Flag 2 flip-flop will be set and the CPU checks this flip-flop after returning from the CIOP subroutine. If Flag 2 flip-flop is set, a jump to ROM address 2251_8 (I/O Kill) is initiated; the condition code is set to CCL and the instruction is terminated. If Flag 2 flip-flop is not set, the condition code is set to CCE.

3-246. SET ENABLE/DISABLE EXTERNAL INTERRUPTS. The interrupt system is set, to enable or disable external interrupts, by the Set Enable/Disable External Interrupts (SED) command. The interrupt system is enabled or disabled according to the least significant bit (bit 15) of the instruction. If bit 15 is a logic 1, bit 1 of the status register is set to logic 1, enabling the external interrupts. If bit 15 is "0", bit 1 of the status register is set to logic 0, disabling the external interrupts.

3-247. An operational flow diagram for a SED instruction is shown in figure 3-16. A SED instruction causes the LUT decoder to generate a microprogram ROM address of 2315_8 and the microinstruction at this address is loaded into ROR1. SED is a privileged instruction and if the CPU is operating in the non-privileged mode, the non-privilege skip condition is met, a jump to ROM address 2763_8 (Trap 6) is executed, and the instruction is terminated. If the non-privilege condition is not met, an S-bus field code STA loads the contents of the status register into the S-bus register. ROR1 bits 16:31 (= 137777_8) are loaded into the R-bus register and the T-bus becomes the R-bus register content "and'ed" with the S-bus register content, resetting the external interrupt bit (bit 15) of the status word. The status word is then loaded back into the status register by a store field code STA.

3-248. The current instruction is loaded from the CIR into the S-bus register, an add is executed, and the U-bus becomes the sum of the S-bus register plus the R-bus register. (Since the R-bus = 0, the U-bus actually becomes the content of CIR.) If U-bus bit 15 is a logic 0, ROR2 is NOP'ed, and status bit 1 is left at "0". If U-bus bit 15 is a "1", the status bit 1 is set to "1" (by STA←STA+040000_8) and the external interrupts are enabled.

3-249. SET MASK. The Set Mask (SMSK) instruction causes a SMSK command to be sent, with the mask word, to all I/O subsystem device controllers. Each "1" bit in the mask word sets the Mask flip-flop in the group of device controllers which are wired to be controlled by that specific bit. Each "0" bit in the mask word clears the Mask flip-flop in the device controller group.

3-250. An operational flow diagram for a SMSK command is shown in figure 3-17. The SMSK instruction causes the LUT decoder to generate a microprogram ROM starting address of 2332_8 and the microinstruction at this address is loaded into ROR1. SMSK is a privileged instruction and if the CPU is operating in the non-privileged mode the non-privilege skip condition is met, a skip to ROM address 2763_8 (Trap 6) is initiated, and the instruction is terminated. If the non-privilege skip condition is not met, the SR register content is checked to determine if the mask word is in a TOS register. If the SR register content equals zero, a jump to ROM address 3420_8 is initiated and a PUL1 subroutine is executed (see figure 3-11), pulling the mask word from memory to the TOS register. An S-bus field code RA loads the TOS register, containing the mask word, into the S-bus register and from there it is stored in the SP2 register by a store field code SP2. The SMSK command is formed and loaded into the SP3 register and, with the mask word, is transferred to the IOP by a CIOP subroutine (see figure 3-9). The IOP transfers the SMSK IOCMD and the mask word to the IOP bus and the I/O subsystems and informs the CPU of the transfer by setting the Flag 1 flip-flop.

3-251. The CPU checks the Flag 2 flip-flop to determine if I/O timeout has occurred. If the Flag 2 flip-flop is set (I/O timeout), a jump to ROM address 2251_8 (I/O Kill) is executed, the condition code is set to CCL, and the instruction is terminated. If I/O timeout has not occurred (Flag 2 flip-flop is not set), the mask word in the TOS register is loaded into the CPU mask register and the stack is POP'ed. The SR register is decremented and the name register is incremented to point to the new TOS register. The special field code CF1 clears the Flag 1 flip-flop, the condition code is set to CCE and the instruction is complete.

3-252. START I/O. The Start I/O (SIO) instruction starts the programmed I/O mode of operation for an addressed I/O subsystem. Once the I/O subsystem receives the SIO command, the I/O control program (in memory) is executed by the I/O subsystem in conjunction with the IOP, multiplexer channel, and memory, or in conjunction with a selector channel and memory. The CPU is not involved after SIO is received and accepted by the I/O subsystem. The starting address of the I/O program is on the top of the stack, either in a TOS register or in memory. If the starting address is not in a TOS register (SR = 0), a PUL1 subroutine (see figure 3-11) is executed and the address is fetched from memory and loaded in the TOS register. The address of the I/O subsystem is in the stack at location S-K, where S is the sum of the SR register and SM register contents and K is a constant contained in bits 12 through 15 of the SIO instruction. The I/O subsystem address (DEVNO) and a TIO command are sent to the I/O subsystem. The I/O subsystem decodes its address and the TIO IOCMD and gates its status to the IOP. The IOP transfers the status word to the CPU where it is checked to determine if the I/O subsystem is ready for SIO. If the I/O subsystem is ready, the address in the TOS register is sent to memory (in particular, the first word of the subsystem's DRT) to point to the first instruction word for the I/O program and an SIO command is sent to the I/O subsystem, telling it to begin executing its I/O program. The TOS is deleted and the condition code is set to CCE. If the I/O subsystem is not ready for SIO, its status word is pushed onto the stack and the condition code is set to CCG.

3-253. An operational flow diagram for a Start I/O instruction is shown in figure 3-18. An SIO instruction causes the LUT decoder to generate a microprogram starting address of 2233_8 and the microinstruction at this address is loaded into ROR1. A function field code JSB and a skip field code UNC cause a jump to ROM address 3533_8 and a FS-K subroutine is executed (see figure 3-8). The I/O subsystem address (DEVNO) at stack location S-K is fetched and loaded into the SP1 register and from there to the S-bus register. A TIO command is formed and loaded, with DEVNO, into the SP3 register for transfer to the IOP. When the CIOP subroutine is executed (see figure 3-9), DEVNO and the TIO IOCMD are transferred to the IOP and loaded into the IOPC register. The IOP transfers the DEVNO, TIO IOCMD, and SO to the IOP bus. The addressed I/O subsystem decodes DEVNO and TIO and gates its status word, with SI, to the IOP. The IOP loads the status word into the DID/MUXMA register and ends SO. A RDIOD level from the CPU reads the DID/MUXMA to the S-bus and from there to the SP2 register.

3-254. If the addressed I/O subsystem has not responded to the TIO IOCMD (I/O timeout), Flag 2 flip-flop will be set. The CPU checks Flag 2 flip-flop and, if it is set, a jump to ROM address 2251_8 (I/O Kill) is executed, the condition code is set to CCL and the instruction is terminated. If Flag 2 flip-flop is not set, an S-bus field code SP2 loads the SP2 register content (I/O subsystem status word) into the S-bus register where its bit 0 is checked. If bit 0 is a logic 0, signifying that the I/O subsystem is not ready for SIO, a jump to ROM address 2247_8 (I/O Abort) is executed, and the instruction is terminated. I/O Abort pushes the subsystem's status word onto the stack and sets the condition code to CCG.

3-255. If the I/O subsystem is ready for SIO (bit 0 = 1), the CPU checks the content of the SR register to determine if the address of the first I/O program instruction word is in a TOS register or in memory. When SR equals zero, the address of the first I/O program word is in memory and a jump to ROM address 3420_8 (PUL1 subroutine, see figure 3-11) is executed. The PUL1 subroutine pulls one word (containing the address of the first instruction word) from memory and loads it into the TOS register. The CPU computes the DRT address (DEVNO times 4) by adding the R-bus and S-bus register contents (which have been loaded from the SP1 register), in effect multiplying the SP1 register contents by two, and then

shifting the T-bus left 1, which multiplies the amount by two again. The result is the DEVNO multiplied by four, which is the DRT location. This address is loaded into the UCOR register by the store field code BUSL. The MCU option field CWA causes the MCU operation decoder to generate a Clear-Write (CW) MOP code and this code is loaded into the PUMOP register. The TO code, generated by the mapper from U-bus bits 0, 1, 2, 14, and 15, is loaded into the UTO register. The store field code BUSL also sets the LOREQ flip-flop. When the module READY line designated by the TO code in the UTO register is high (destination READY) and when no module of higher priority than the CPU has its ENABLE line pulled low, the LOSEL and CPUSEL flip-flops are set. The LOSEL signal pulls the destination module's READY line low. The CPUSEL signal gates the TO, FROM, MOP and UCOR to the central data bus, thus transferring the address and CW MOP code to the memory module.

3-256. The address of the first word of the I/O program is then transferred to this DRT address. The S-bus field code RA loads the TOS register (containing the SIO program address) into the S-bus register and the store field code DATA loads it into the UCOR register. The condition code is set to CCE. The HREQ flip-flop is set and the PSELECT level, when low, reads the UCOR register. When the CPUSEL flip-flop is set, the CPUSEL logic level reads the UCOR register contents to memory via the central data bus. Once the starting address of the SIO program has been transferred to the DRT, the CPU forms the SIO command and loads this command into the SP3 register. The CIOP subroutine (see figure 3-9) is executed again, transferring the SIO command to the IOPC register in the IOP. When the STOIOA signal is issued, the IOPC register is loaded. If an SIO or interrupt is not current, the Direct Address and Flag 1 flip-flops are set, and the DAG and F1 logic levels become "1", gating the IOCMD to the IOP bus and causing SO. The microcode must wait for F1 before terminating STOIOA. If the I/O subsystem does not respond to SIO (I/O timeout), CPX2 bit 10 is set to "1". The microcode checks this bit and, if it is logic 1, a jump to ROM address 2251_8 (I/O Kill) is executed by the CPU. The condition code is set to CCL and the instruction is terminated. If Flag 2 flip-flop is not set, the stack is POP'ed, the SR register is decremented and the name register is incremented to point to the new TOS register, and the instruction is completed.

3-257. PROGRAMMED I/O.

3-258. Programmed I/O operation begins for any particular I/O subsystem when the CPU issues an SIO command for that I/O subsystem. The I/O subsystem, IOP and multiplexer channel (or subsystem and selector channel) then execute the I/O program without further assistance from the CPU. The IOP transfers an I/O program, one instruction double word at a time, from the memory to the multiplexer channel. The multiplexer channel then controls operation of the I/O subsystem.

3-259. Once an I/O subsystem decodes and accepts an SIO command, it sends a Service Request signal to its multiplexer channel. The multiplexer channel sends a High Service Request (HSREQ) to the IOP. If the DRCT ACT flip-flop is set, the HSREQ from the subsystem will have no effect on the IOP because issuance of Data Poll is inhibited. When an SI is received by the IOP, however, the DRCT ACT flip-flop is cleared, terminating the direct active state, and the IOP issues a Data Poll. The multiplexer channel having highest priority stops the Data Poll and returns a DRT Fetch command and SI to the IOP. The IOP requests the DRT entry from memory and begins transferring the I/O program, one instruction at a time, to the I/O subsystem. The order contained in the instruction double word determines whether data is transferred to the I/O subsystem or from the I/O subsystem to memory.

3-260. DRTE FETCH/STORE. An operational flow diagram for a DRTE fetch/store operation is shown in figure 3-19. A HSREQ signal from the multiplexer sets the I/O High Service Request (IOHSREQ) flip-flop, which in turn sets the DPOLL and SIO ACT flip-flops. If HSREQ, STOIOA and STOIOD are active at the same time, the Data Poll, DRCT ADDR and DRCT DATA flip-flops are all set. The Data Poll flip-flop, however, nullifies all effects that the DRCT ADDR and DRCT DATA flip-flops would have, and clears them on the next clock. The SI(D) latch guarantees that SI arrives during the first half of the clock cycle. If SI does not arrive during the first half of the clock, SI(D) waits for one clock cycle before latching SI into the IOP.

3-261. The SI(D) level clears the DPOLL flip-flop, Data Poll is terminated and the IOCMP and Data Cycle flip-flops are set. The DRT2 flip-flop is set, setting the IODE level to "1" and gating the data (DRT address) on the IOP bus into the DID/MUXMA register. The DRT Fetch command from the multiplexer channel causes a Read-Restore (RR) MOP code to be generated and sets the DRT1 flip-flop. The LDENB logic level, set to "1" when the Low Request Initializing (LOREQ INIT) flip-flop was set, enables the MUXID register store function. The three most significant bits (bits 0, 1, 2) of the memory address (contained in the DID/MUXMA register) are checked to determine if the memory bounds are exceeded. The three bits comprise a 3-bit binary number. Three jumpers are set to correspond to the three high order bits of the last legal address. (For example, a 32K machine's highest legal address is 77777_8, therefore the jumpers would be set to 011.) The memory address in question and the jumpered number are connected into an arithmetic comparator, which determines if the three most significant bits of the memory address are greater than the jumpered value. An address such as 5_8 is legal because $000 < 011$. Address 100005_8, however, is illegal because $100 > 011$. If an illegal address is determined, the operation is terminated. Transfer error is sent out by the IOP, causing an external interrupt.

3-262. The IOINP, IOLHSEL, and IOLSEL flip-flops are set and, when the memory module is ready, the memory address and RR MOP code are transferred to memory. A SYSPE or MCUDPE signal, if received from memory, will set the Address Parity Error (APE) flip-flop, the IOERROR level goes high, clearing the IOINP flip-flop (disabling the MUXOD register store function) and terminating the operation. If SYSPE or MCUDPE are not received, the MCU Complete (MCU CMP) flip-flop is set. When the DRT entry is received from memory, the FROM code is compared with the saved TO code and, when they are true, IOSTROBE goes high, clearing the IOINP flip-flop and latching the DRT entry into the MUXOD register. The DRT GATE and SO1 flip-flops are set and the DRT entry and SO are gated onto the IOP bus for transfer to the I/O subsystem.

3-263. The address of the next I/O program double word is computed by setting the DRT2 flip-flop, which causes the current DRT entry to be stored in the MUXID register. The CE flip-flop is set and the contents of the MUXID register are incremented by two. The IOHREQ level, set to "1" when the CE flip-flop was set, sets the IOHREQ flip-flop which, in turn, sets the IOHSEL flip-flop. The DRT entry, now incremented by two, is then transferred back to the DRT.

3-264. MEMORY BOUND TRANSFER. Data transfers between memory and an I/O subsystem consist of two steps: the transfer of an address from the multiplexer channel to memory and the transfer of data from the I/O subsystem to the memory address. An operational flow diagram for a memory bound transfer is shown in figure 3-20.

3-265. The HSREQ signal from the multiplexer channel sets the IOHSREQ and DPOLL flip-flops, causing the IOP to issue a Data Poll and to start the IOTIMER. In addition, the SIO ACT flip-flop is set. If a multiplexer channel does not respond to the Data Poll before the IOTIMER flip-flop sets (signifying I/O timeout) the CPX2 register bit 10 is set to "1" and the transfer is terminated. If an SI is received before I/O timeout, the SI(D) and IOLREQ flip-flops are set, the AENB level goes low and the DID/MUXMA register store function is enabled so that it will load the IOP bus transmissions. If the IOCMD code from the multiplexer channel is MEM BND, the Memory Bound (MEM BND) flip-flop is set. The Data Cycle (DATA CYC) flip-flop is set and the DPOLL flip-flop is cleared, terminating the Data Poll. The AENB level goes high, latching the memory address on the IOP bus into the DID/MUXMA register. The IOP generates a Clear-Write (CW) MOP code for transfer to memory and checks the three most significant bits (0, 1, 2) of the memory address in the DID/MUXMA register to determine if the memory bounds are being exceeded. If an illegal address is detected the IOLSEL and IOLHSEL flip-flops are inhibited, and the IODPE signal is sent to the multiplexer channel. When the memory module is ready to receive the address, the IOLSEL and IOLHSEL flip-flops are set and the address and CW MOP code are gated to the central data bus and the IOP sets the SO1 flip-flop and sends SO to the multiplexer channel. If a SYSPE or MCUDPE signal is received from the memory module, the XFER ERROR level goes high and the IODPE signal is sent to the multiplexer channel.

3-266. When an SI signal is received by the IOP (signifying that the inbound data is on the IOP bus), the SI(D) flip-flop is set and the LDENB level goes to "1", enabling the MUXID register store function. The IOHREQ level goes high and the IOHREQ flip-flop is set. The IODE level gates the data on the IOP bus into the MUXID register and the IOP terminates SO by clearing the SO1 flip-flop. When the SO1 flip-flop is cleared, LDENB goes low and the inbound data is latched into the MUXID register. If there is no IOERROR, the IOHSEL flip-flop is set and the data in the MUXID register is gated to the central data bus and, when the MEM BND, DATA CYC and SIO ACT flip-flops are cleared, the data transfer is complete.

3-267. DEVICE BOUND TRANSFER. As with the memory bound transfer, a device bound transfer also consists of two separate steps: a transfer of a memory address from the multiplexer channel to memory and a data transfer from the I/O subsystem (device) to memory.

3-268. An operational flow diagram for a device bound transfer is shown in figure 3-21. During the address transfer phase, operation is the same as that described for a memory bound transfer (see paragraph 3-262) except that a Read-Restore (RR) MOP code is generated and sent to memory instead of a CW MOP code. When the RR MOP code is transferred to the central data bus, the IOINP flip-flop is set, IOINP goes to "1" and the MUXOD store function is enabled.

3-269. When the memory transfers the data to the IOP, it is loaded into the MUXOD register. The TO/FROM comparator compares the TO-FROM lines and when the result is true, sets IOSTROBE to "1". The IOSTROBE level sets the DRT Gate flip-flop and the data is gated out to the IOP bus for transfer to the I/O subsystem. The SO1 flip-flop is set, asserting SO, and starting the IOTIMER. When SI is received (signifying that the I/O subsystem has received the data), the SI(D) and IOCMP flip-flops are set and the DATA CYC flip-flop is cleared, terminating SO and, when the SIO ACT, DRT Gate, and IOCMP flip-flops are cleared, the device bound transfer is complete.

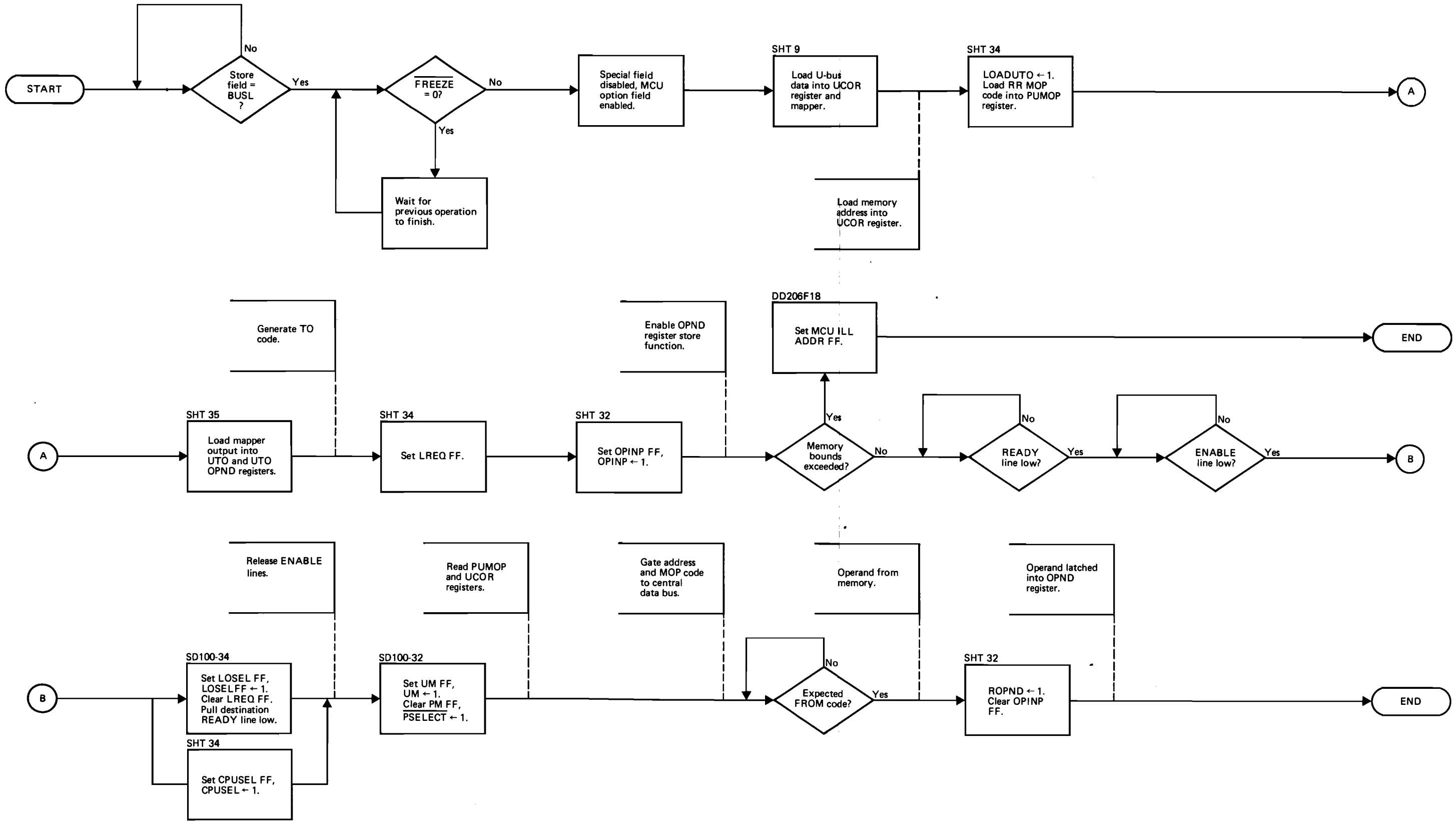
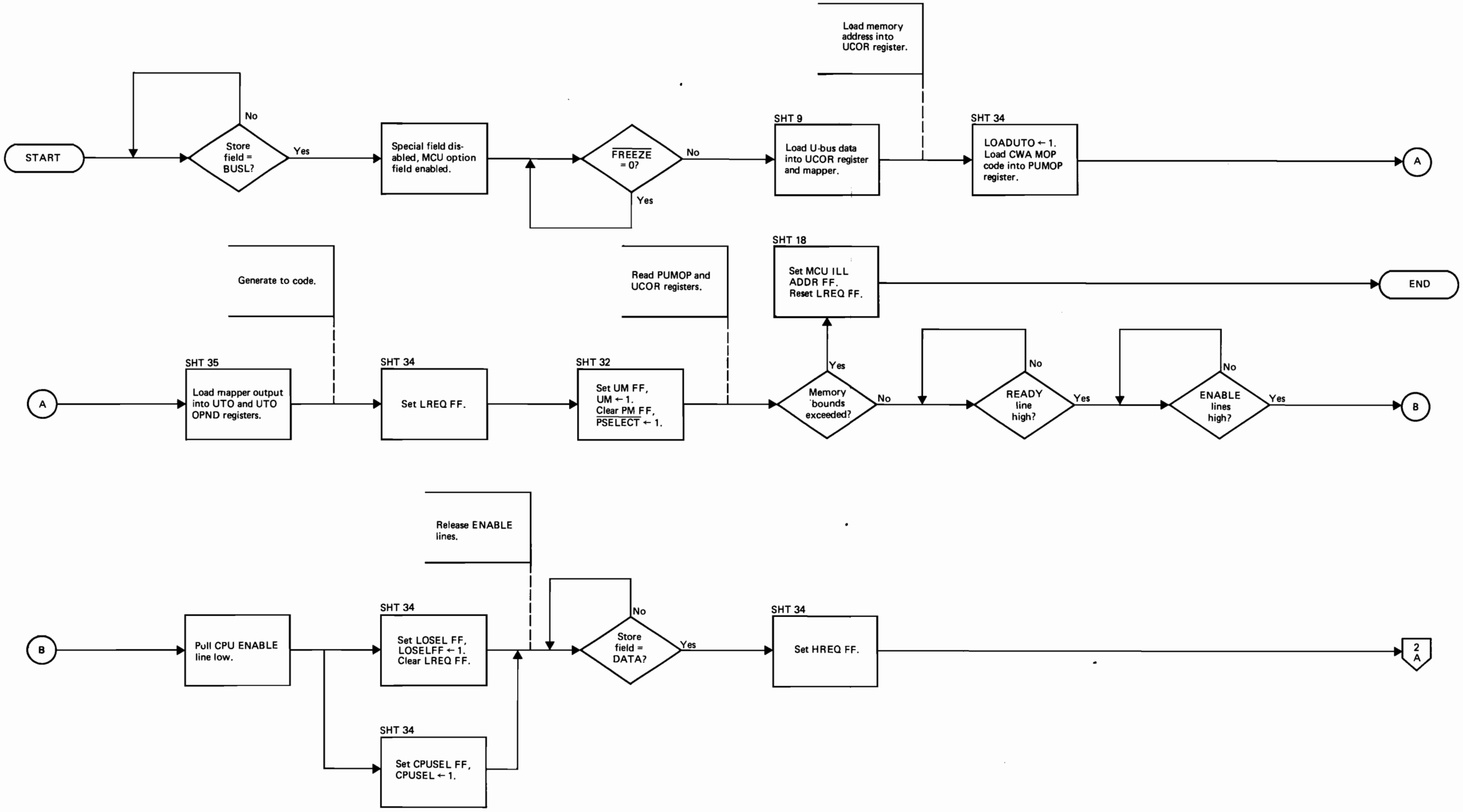
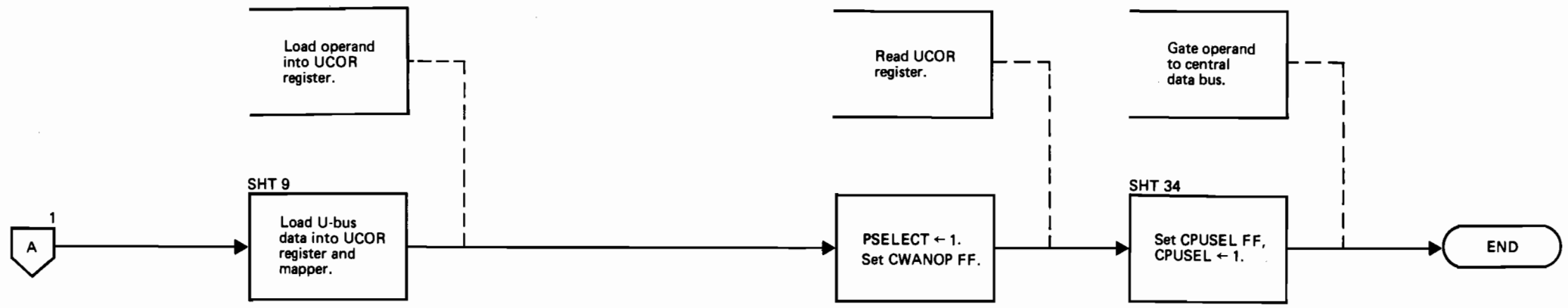


Figure 3-5. Operand Fetch Operational Flow Diagram



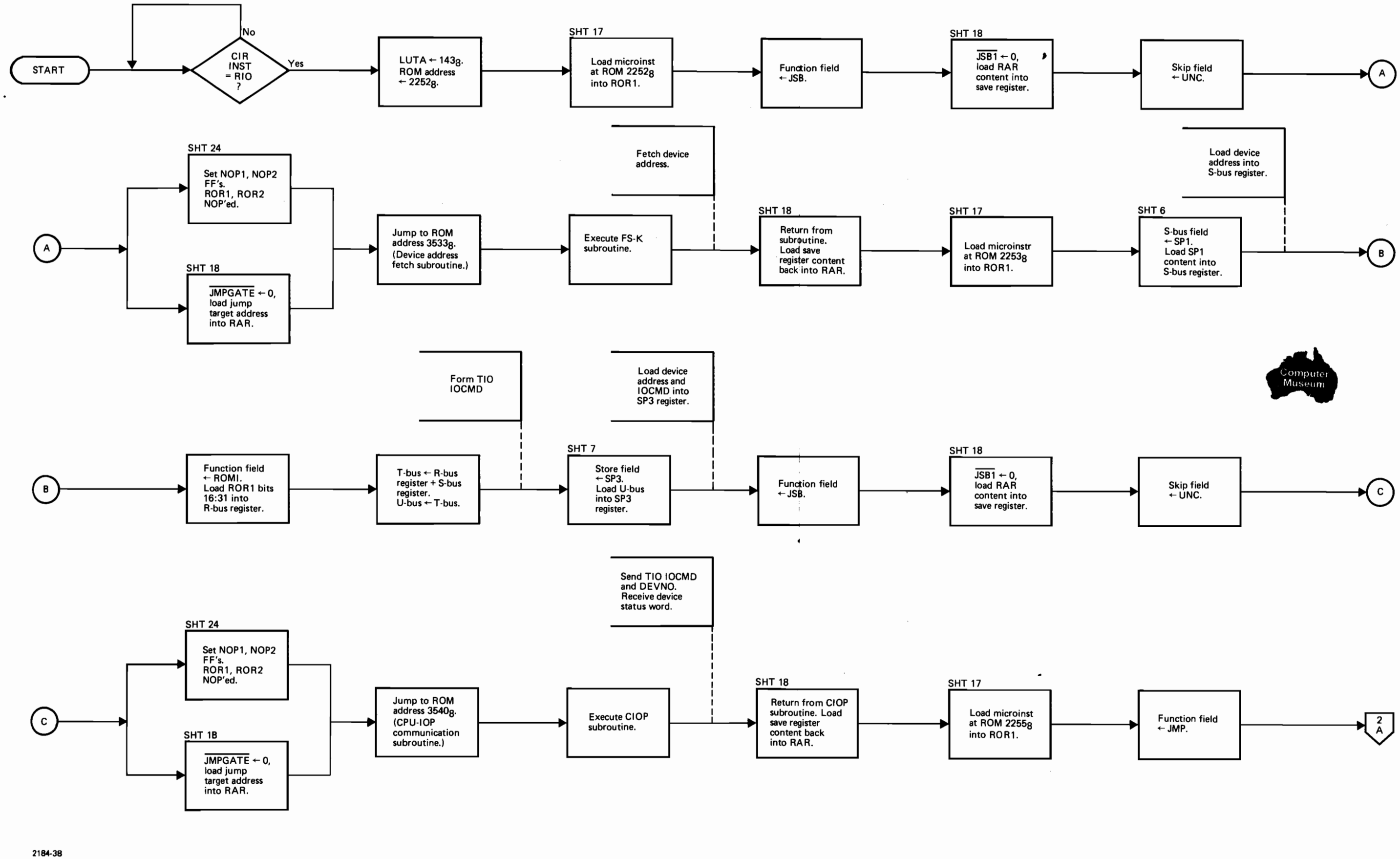
2184-36

Figure 3-6. Operand Store Operational Flow Diagram (Sheet 1 of 2)



2184-37

Figure 3-6. Operand Store Operational Flow Diagram (Sheet 2 of 2)



2184-38

Figure 3-7. RIO Command Operational Flow Diagram (Sheet 1 of 4)

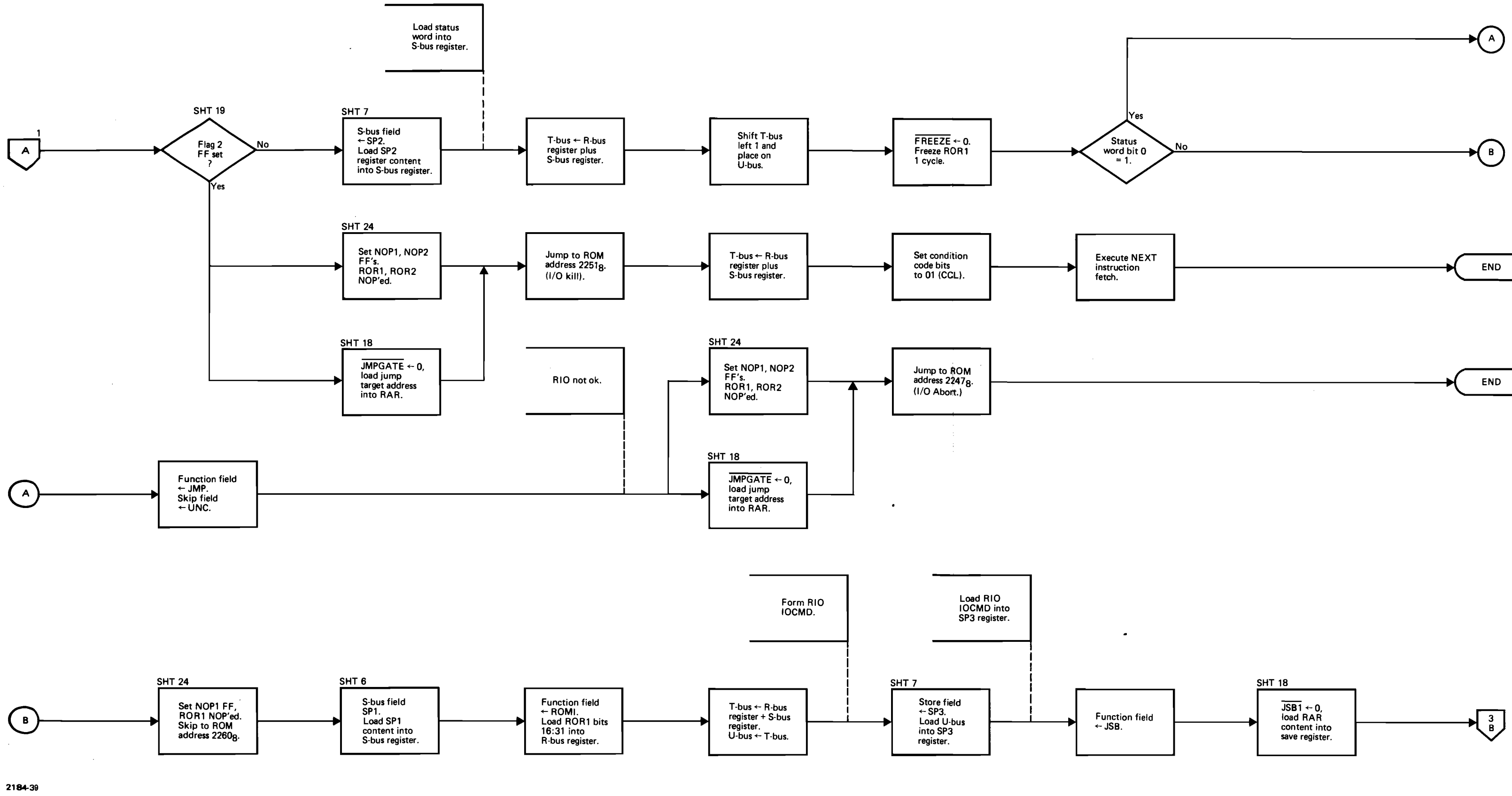
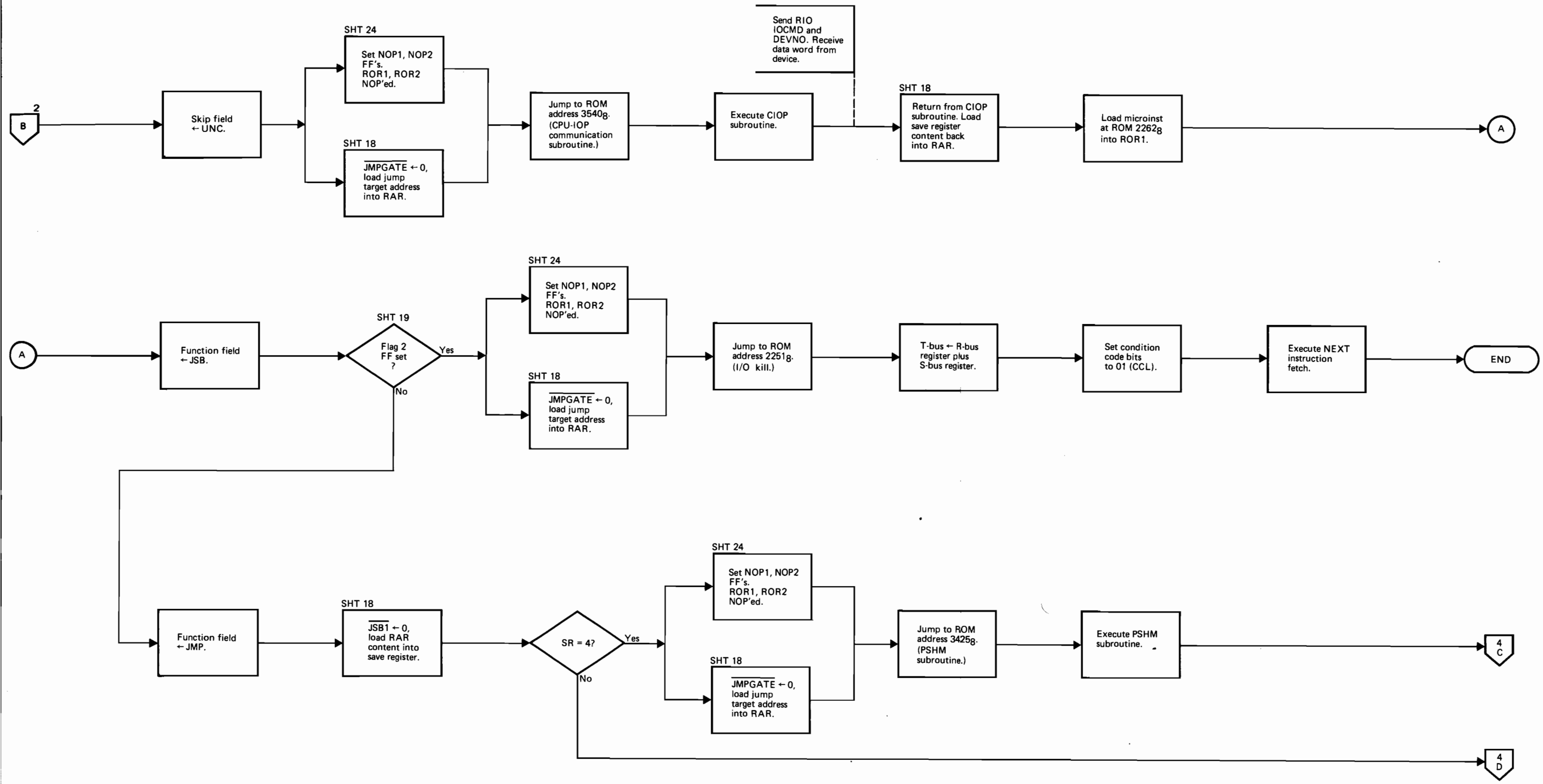
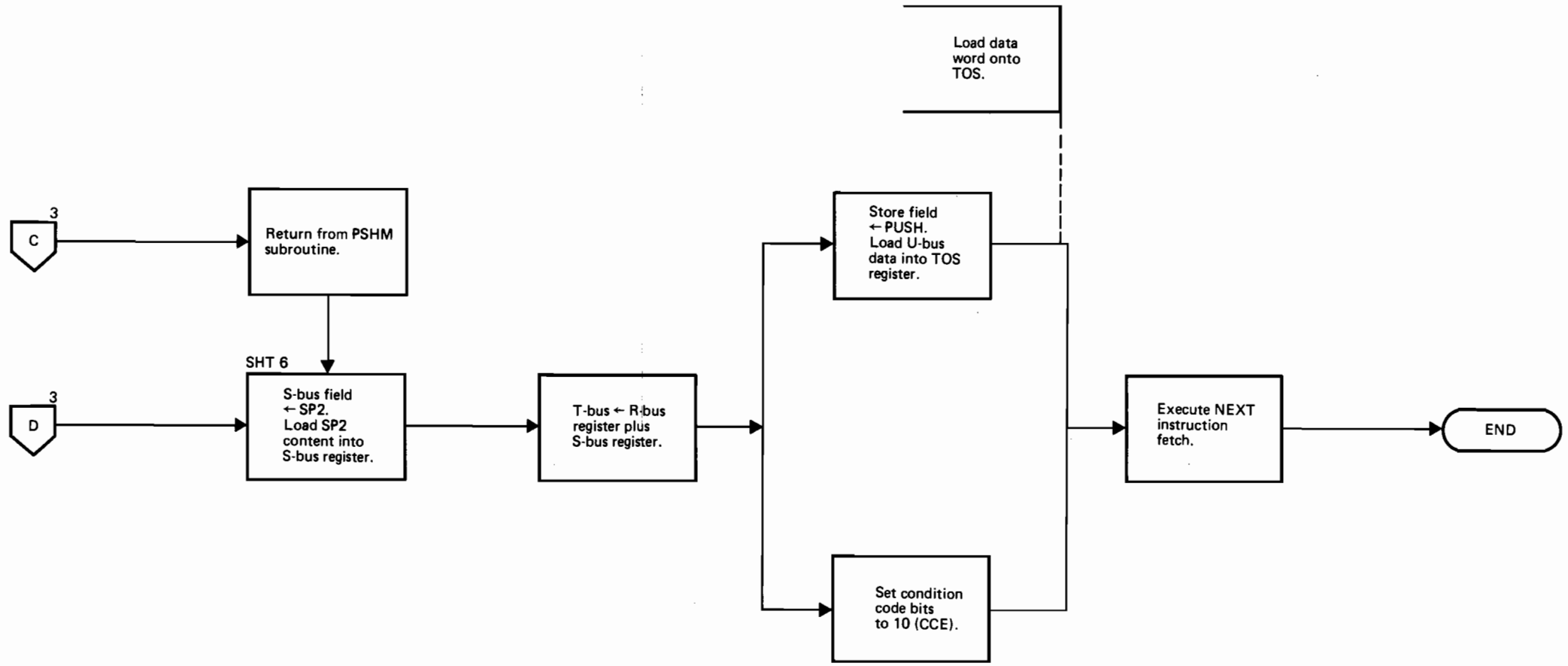


Figure 3-7. RIO Command Operational Flow Diagr
(Sheet 2 of 4)



2184-40

Figure 3-7. RIO Command Operational Flow Diagram (Sheet 3 of 4)



2184-41

Figure 3-7. RIO Command Operational Flow Diagram (Sheet 4 of 4)

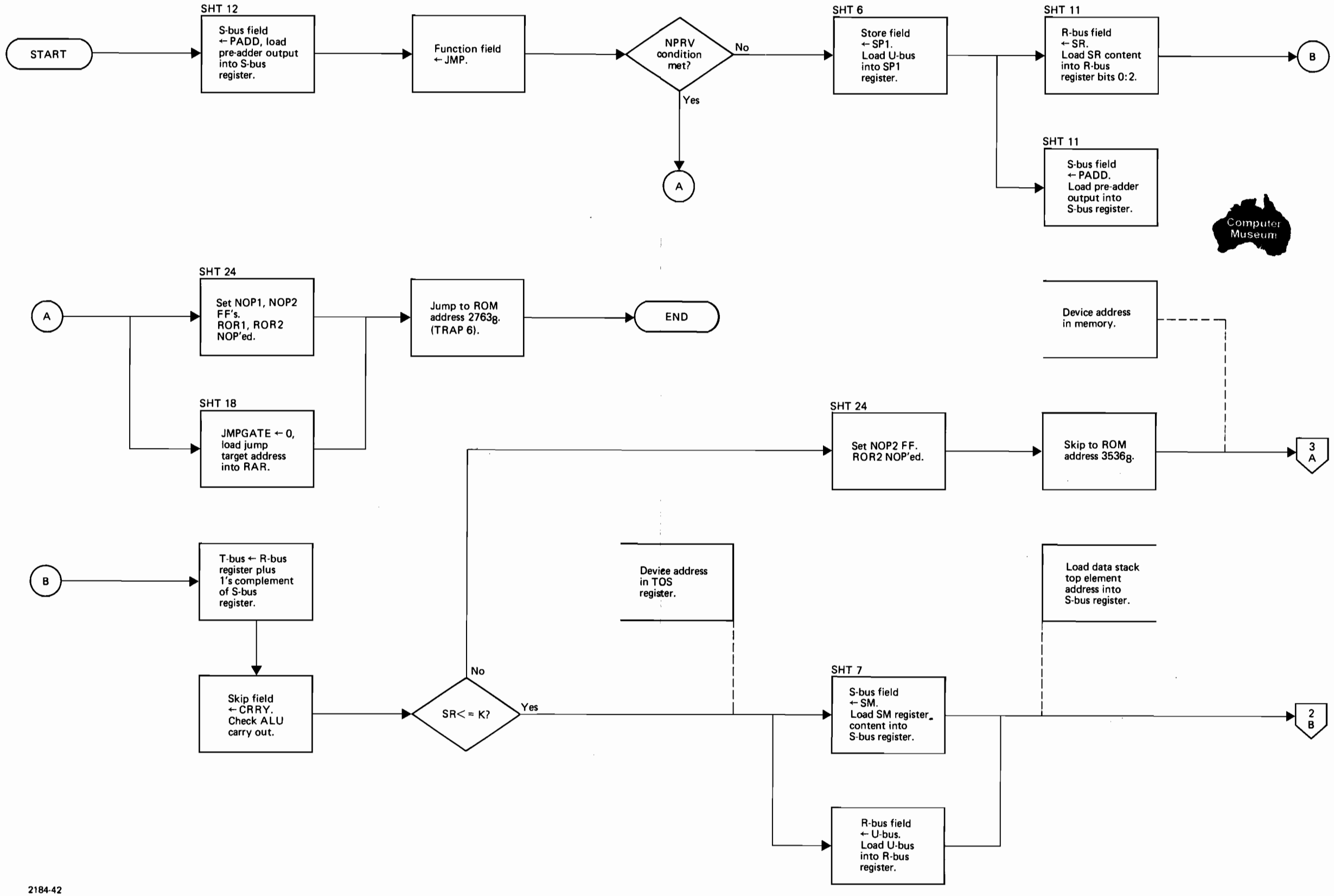
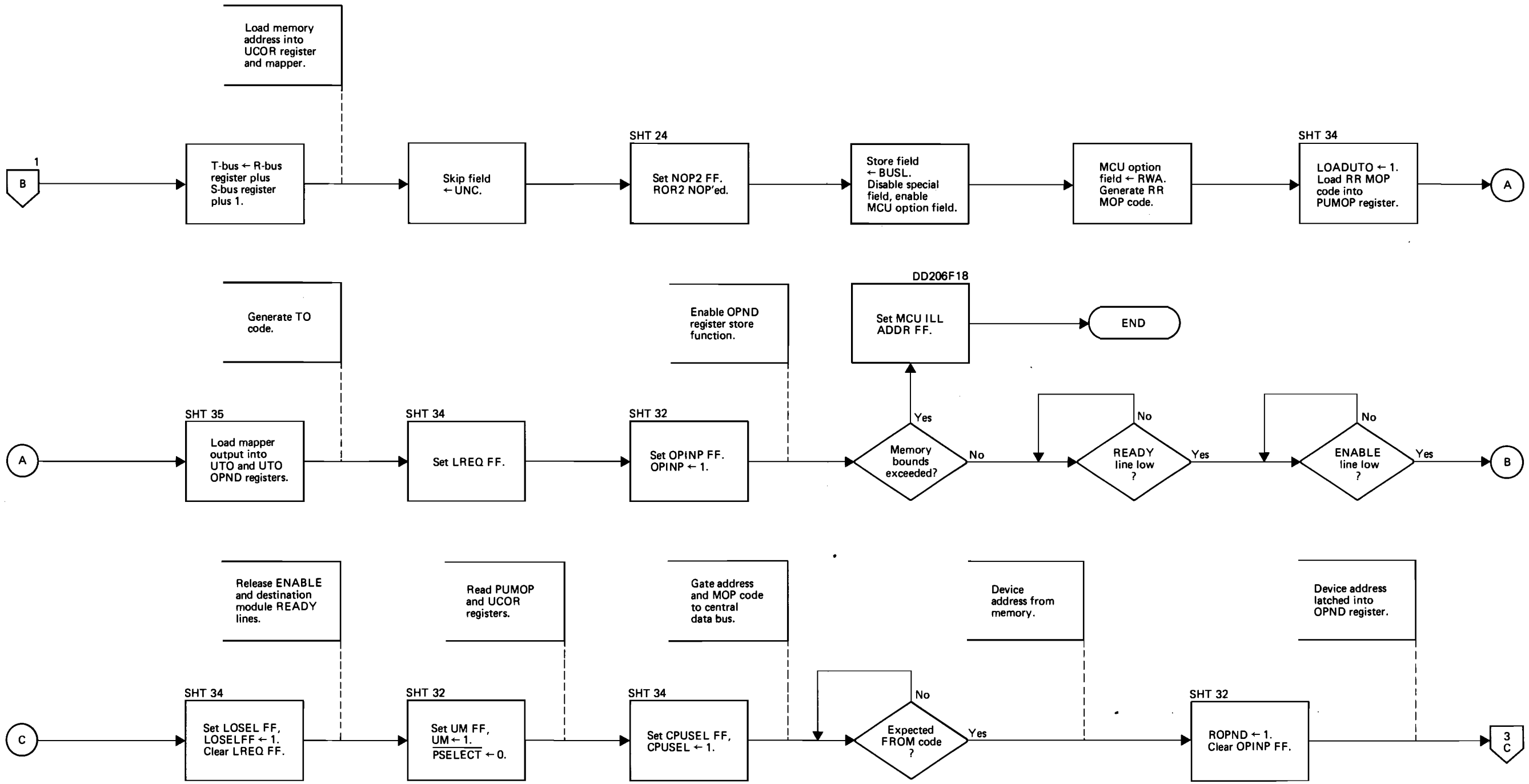


Figure 3-8. FS-K Subroutine (Device Address Fetch) Operational Flow Diagram (Sheet 1 of 3)



2184-43

Figure 3-8. FS-K Subroutine (Device Address Fetch) Operational Flow Diagram (Sheet 2 of 3)

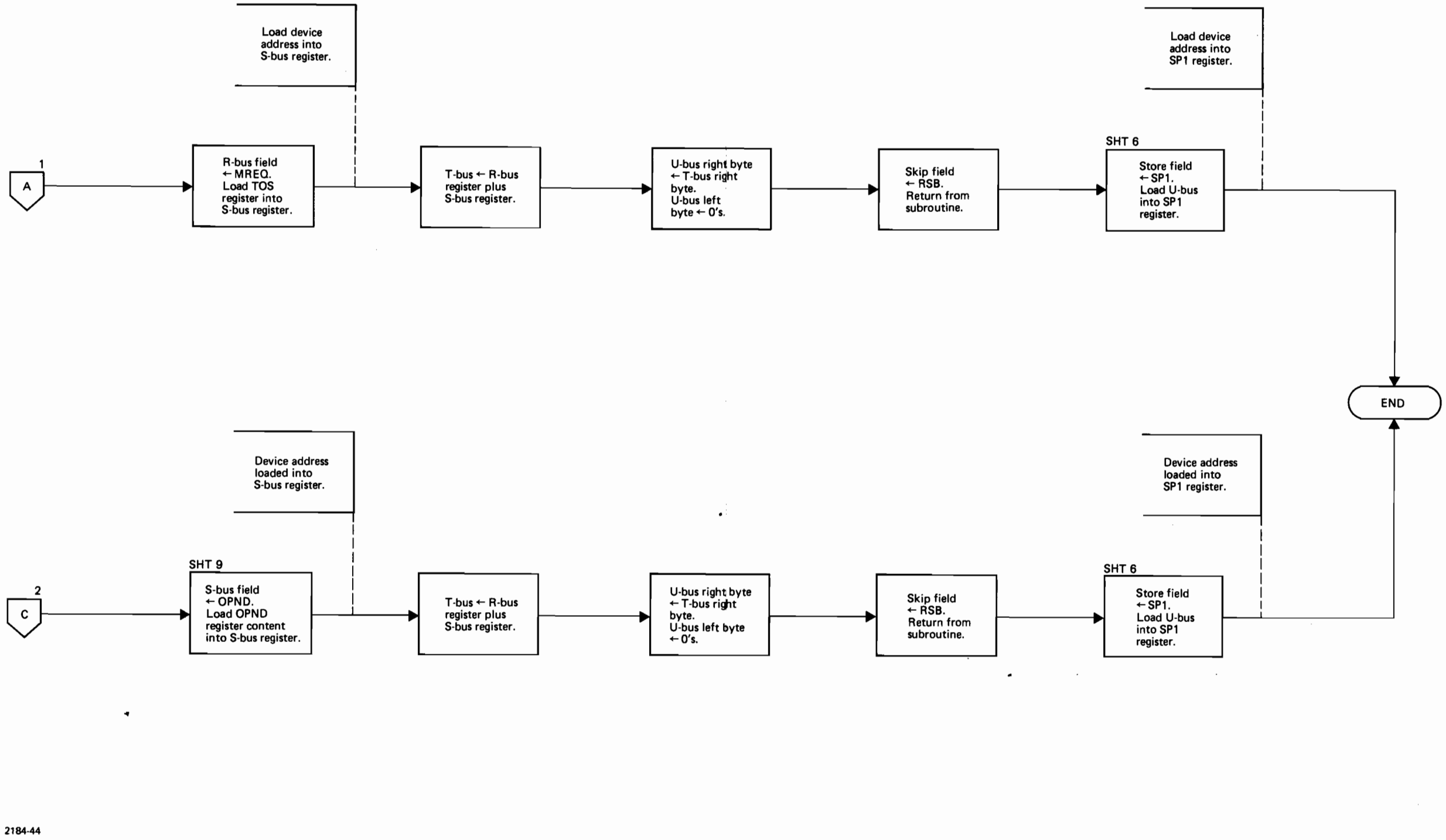


Figure 3-8. FS-K Subroutine (Device Address Fetch) Operational Flow Diagram (Sheet 3 of 3)

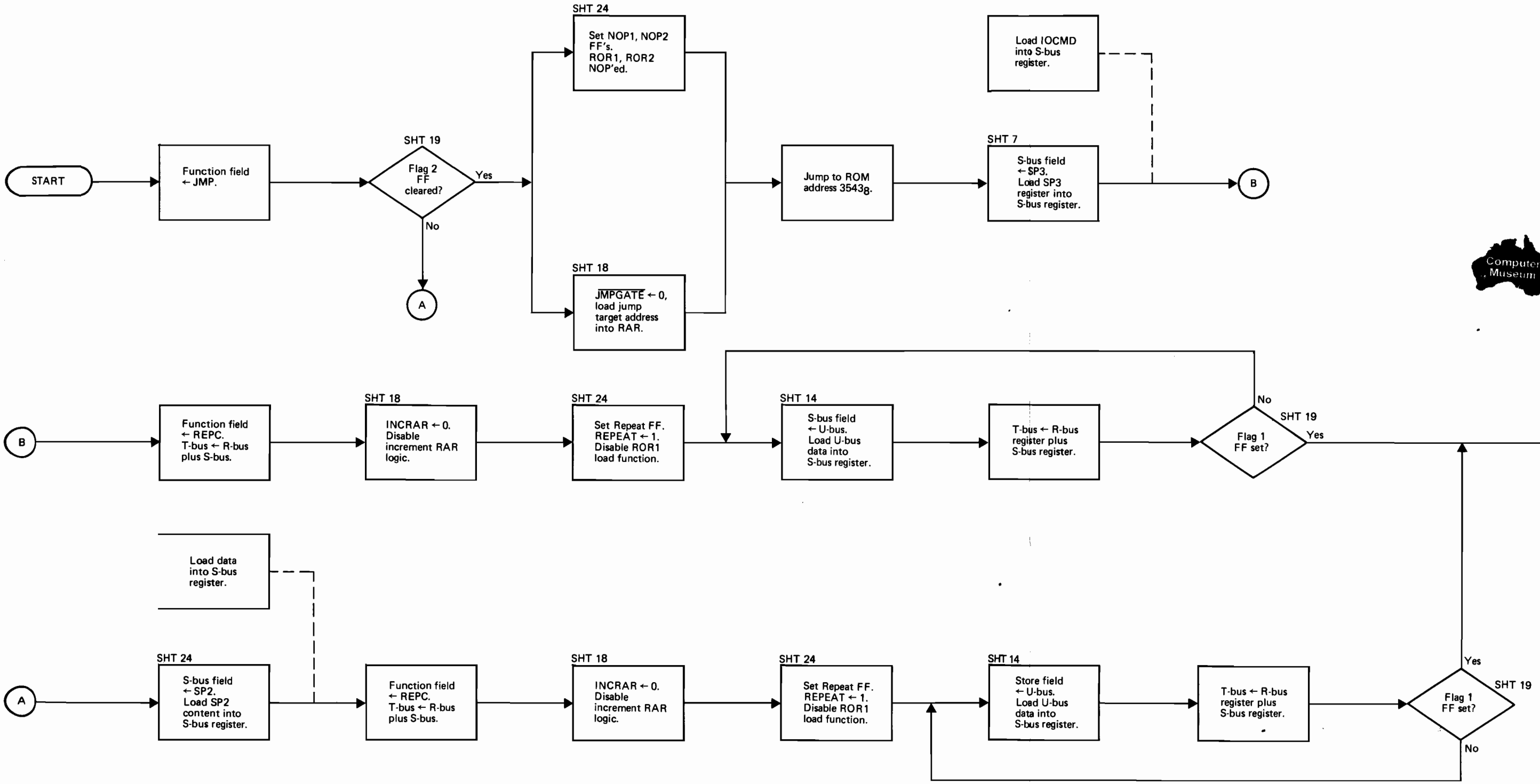
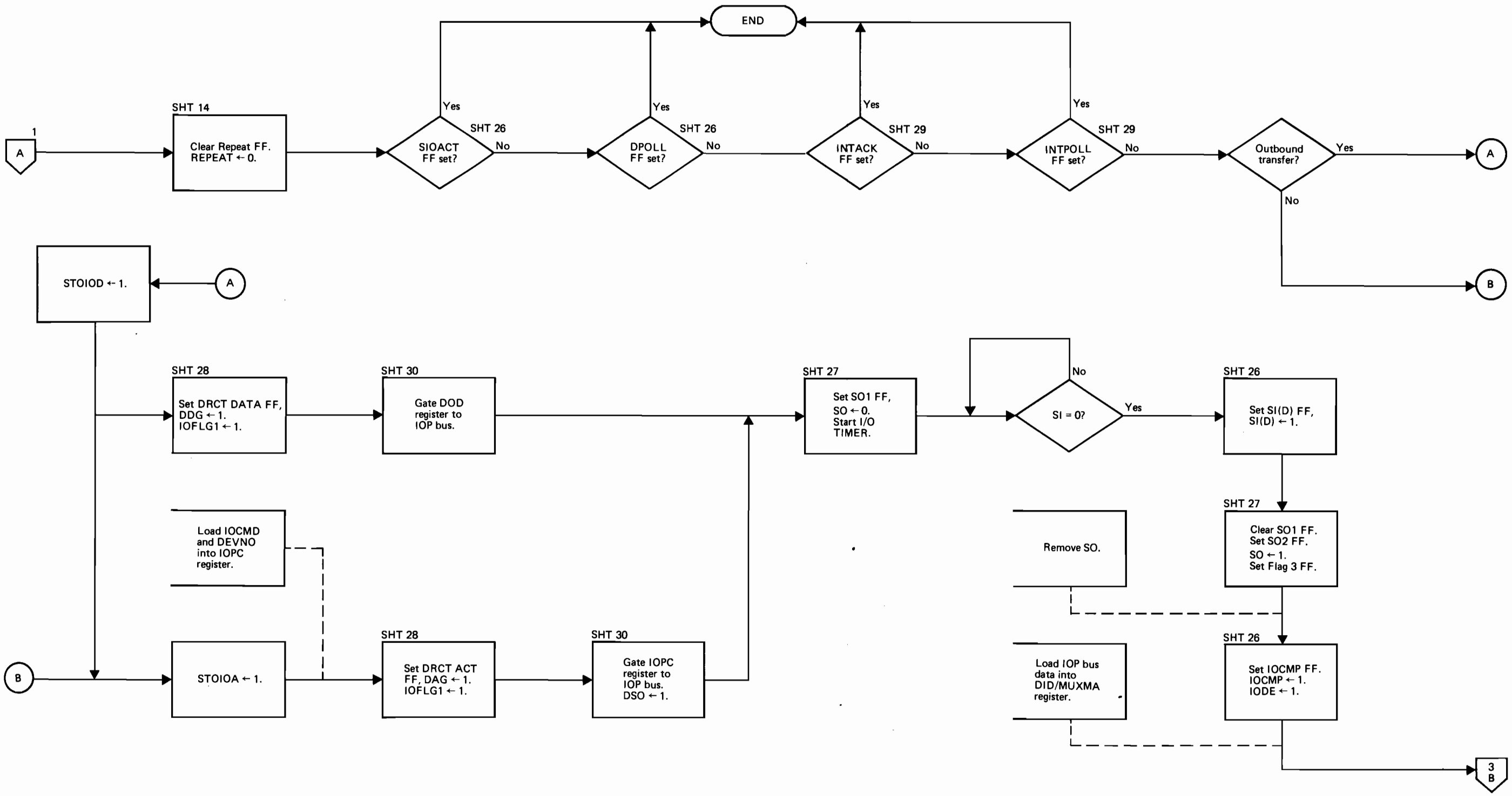
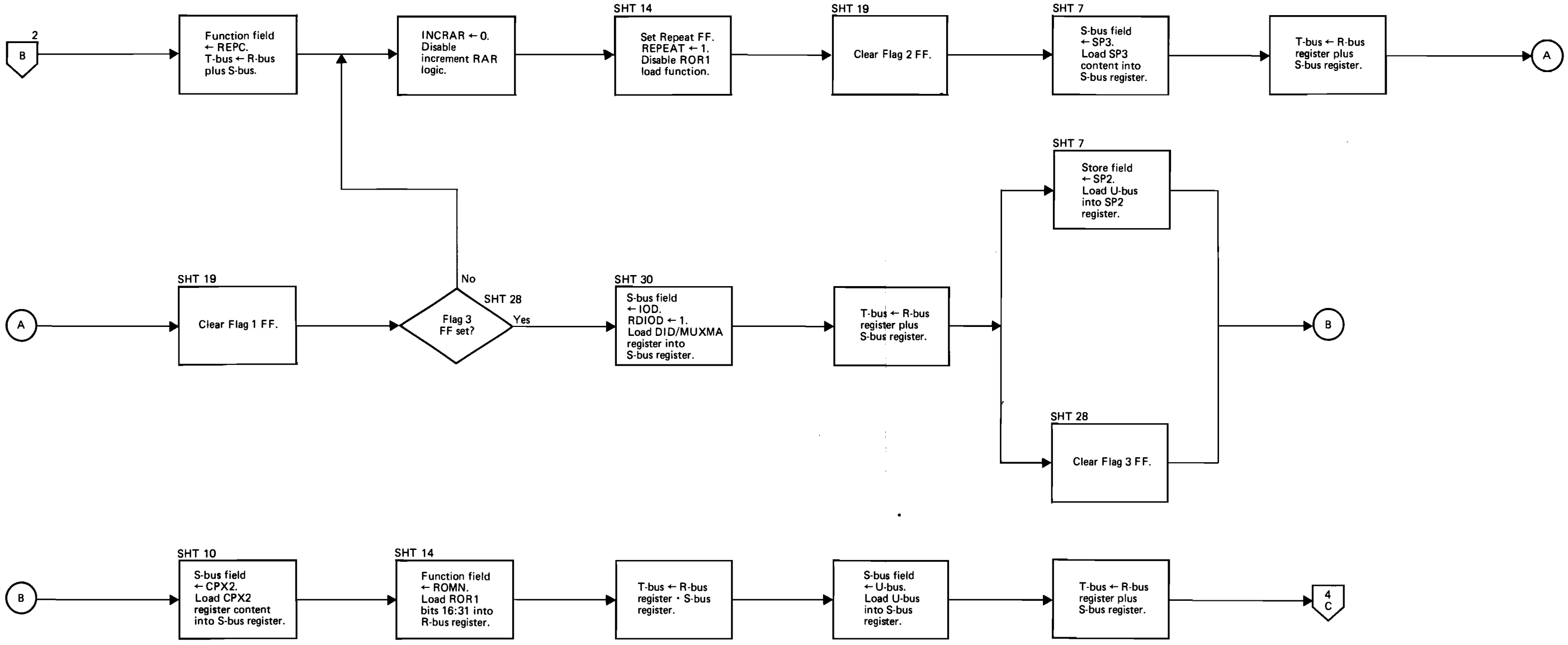


Figure 3-9. CIOP Subroutine (CPU-IOP Communication) Operational Flow Diagram (Sheet 1 of 4)



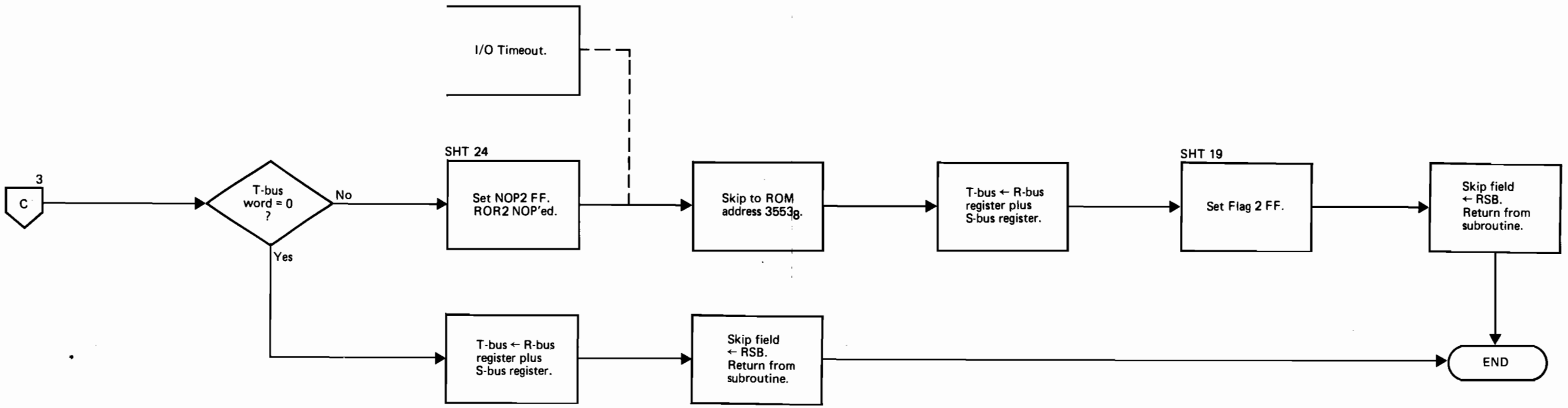
2184-46

Figure 3-9. CIOP Subroutine (CPU-IOP Communication) Operational Flow Diagram (Sheet 2 of 4)



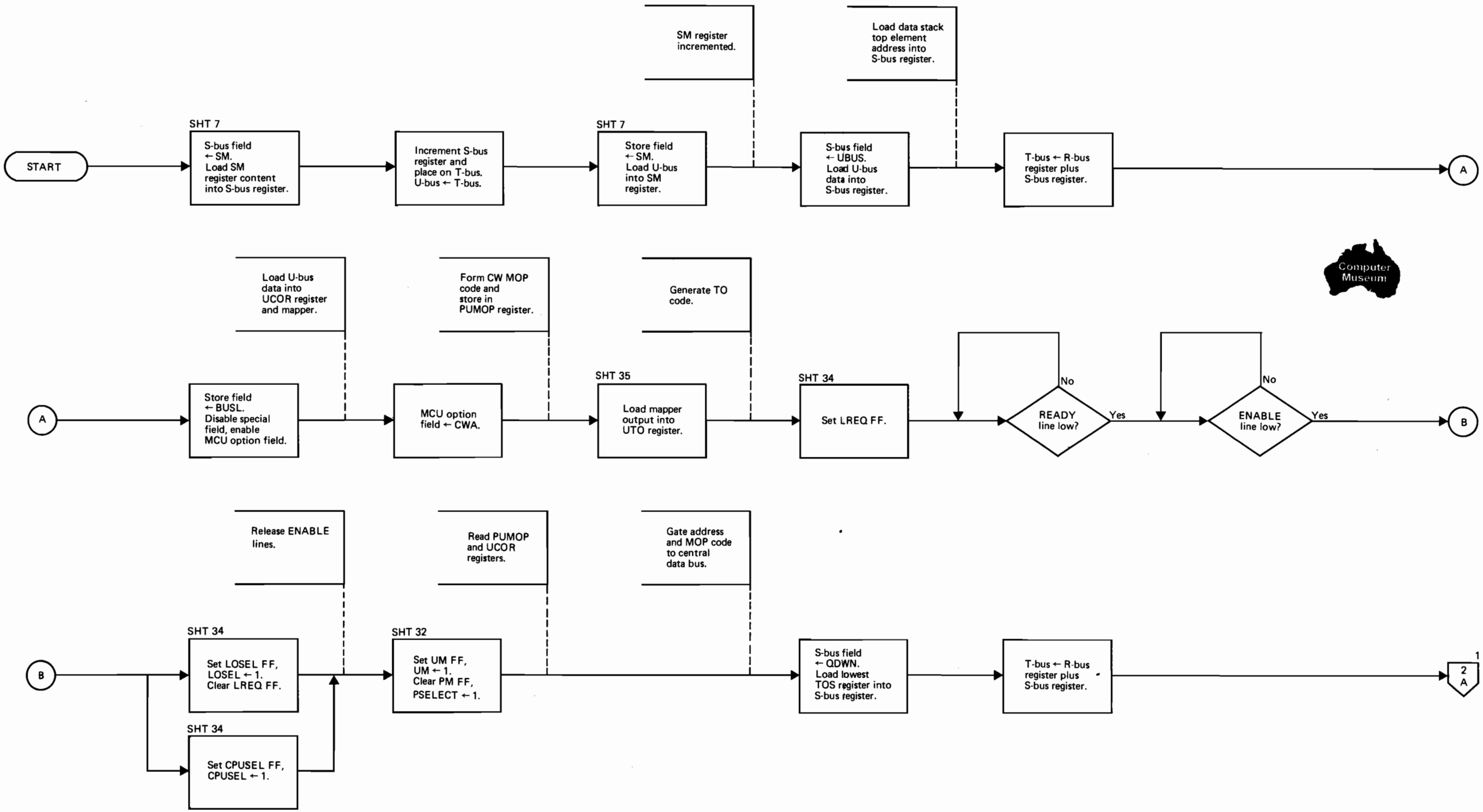
2184-47

Figure 3-9. CIOP Subroutine (CPU-IOP Communication) Operational Flow Diagram (Sheet 3 of 4)



2184-48

Figure 3-9. CIOP Subroutine (CPU-IOP Communication) Operational Flow Diagram (Sheet 4 of 4)



2184-49

Figure 3-10. PSHM Subroutine Operational Flow Diagram (Sheet 1 of 2)

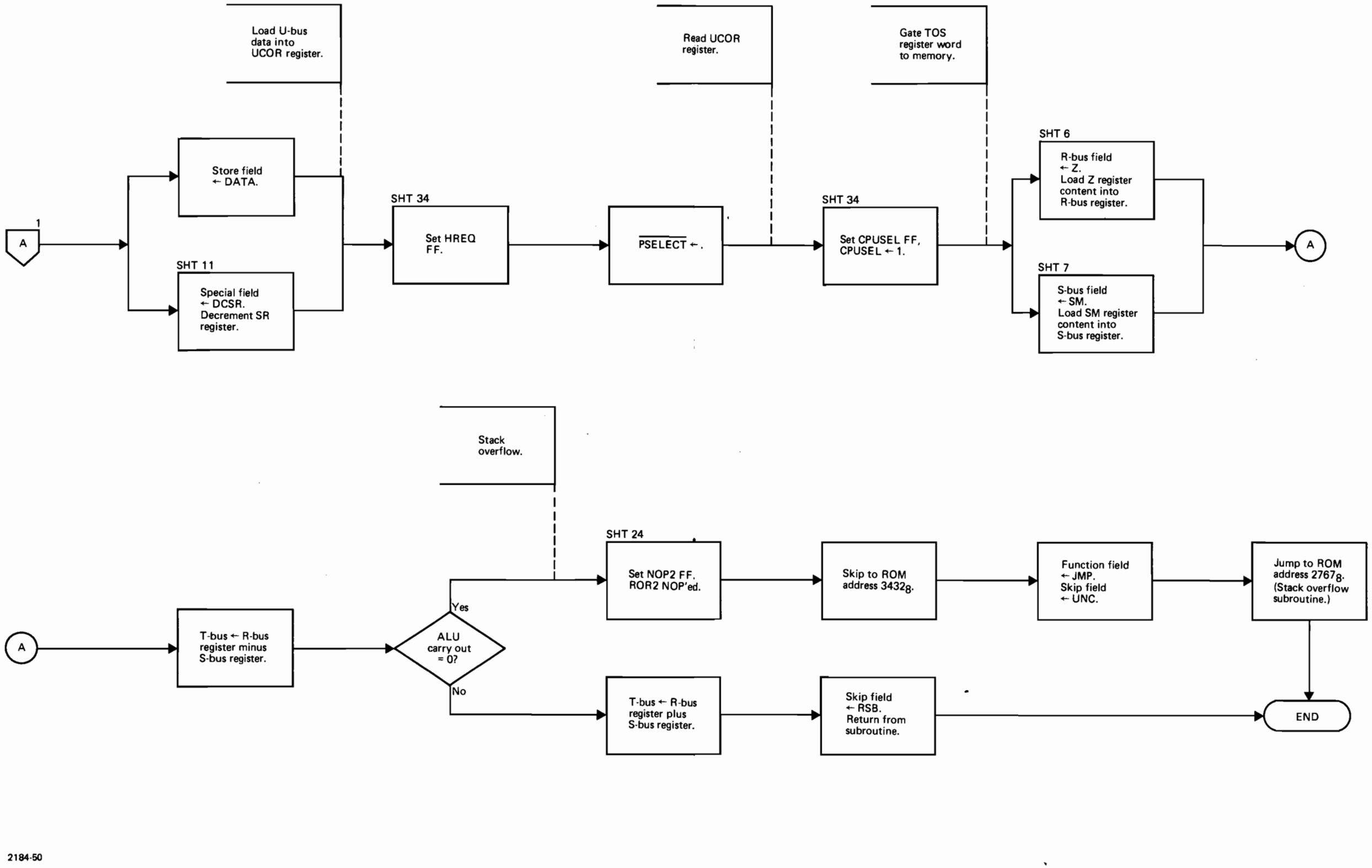
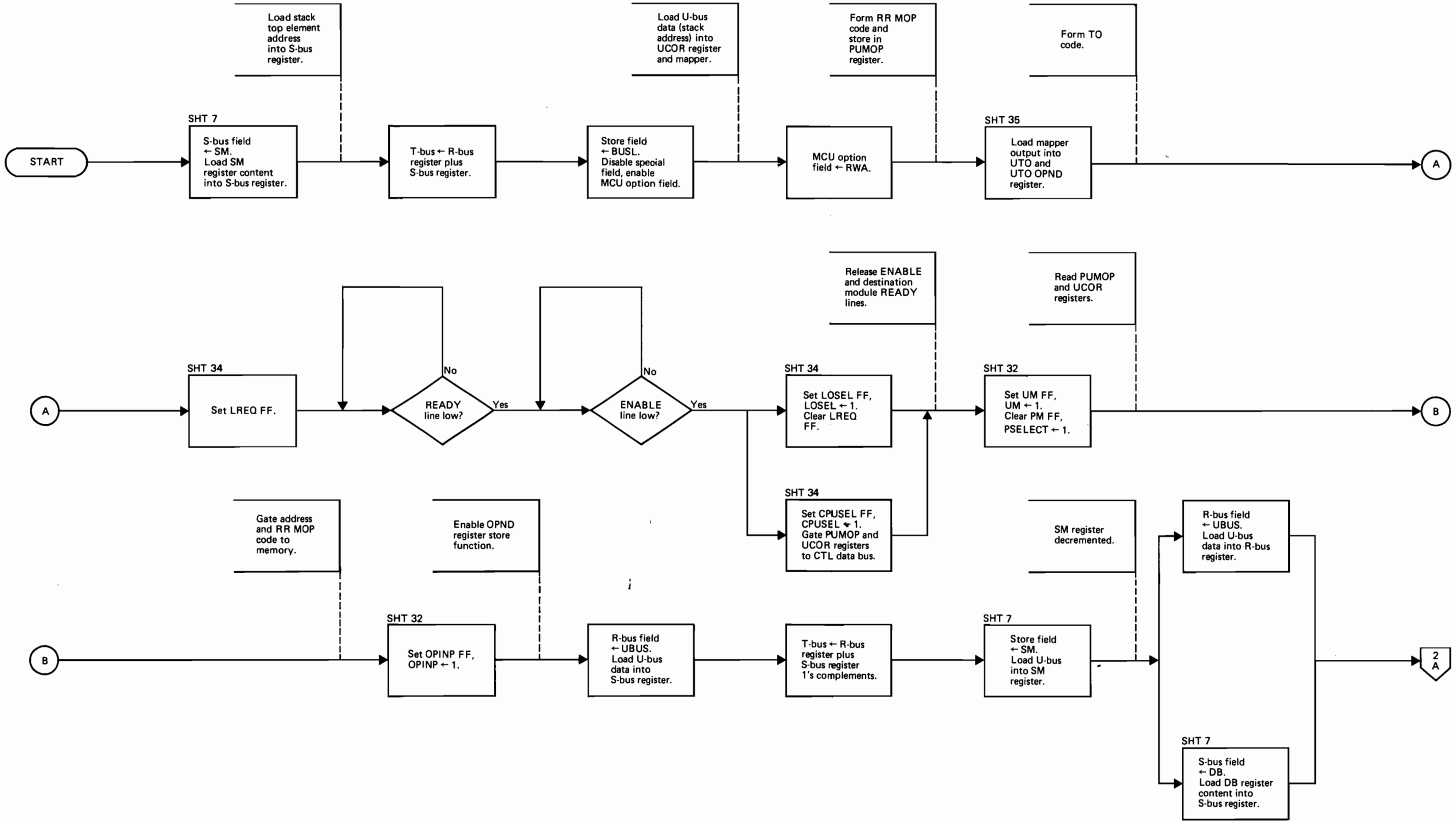


Figure 3-10. PSHM Subroutine Operational Flow Diagram (Sheet 2 of 2)



2184-96

Figure 3-11. PUL1 Subroutine Operational Flow Diagram (Sheet 1 of 2)

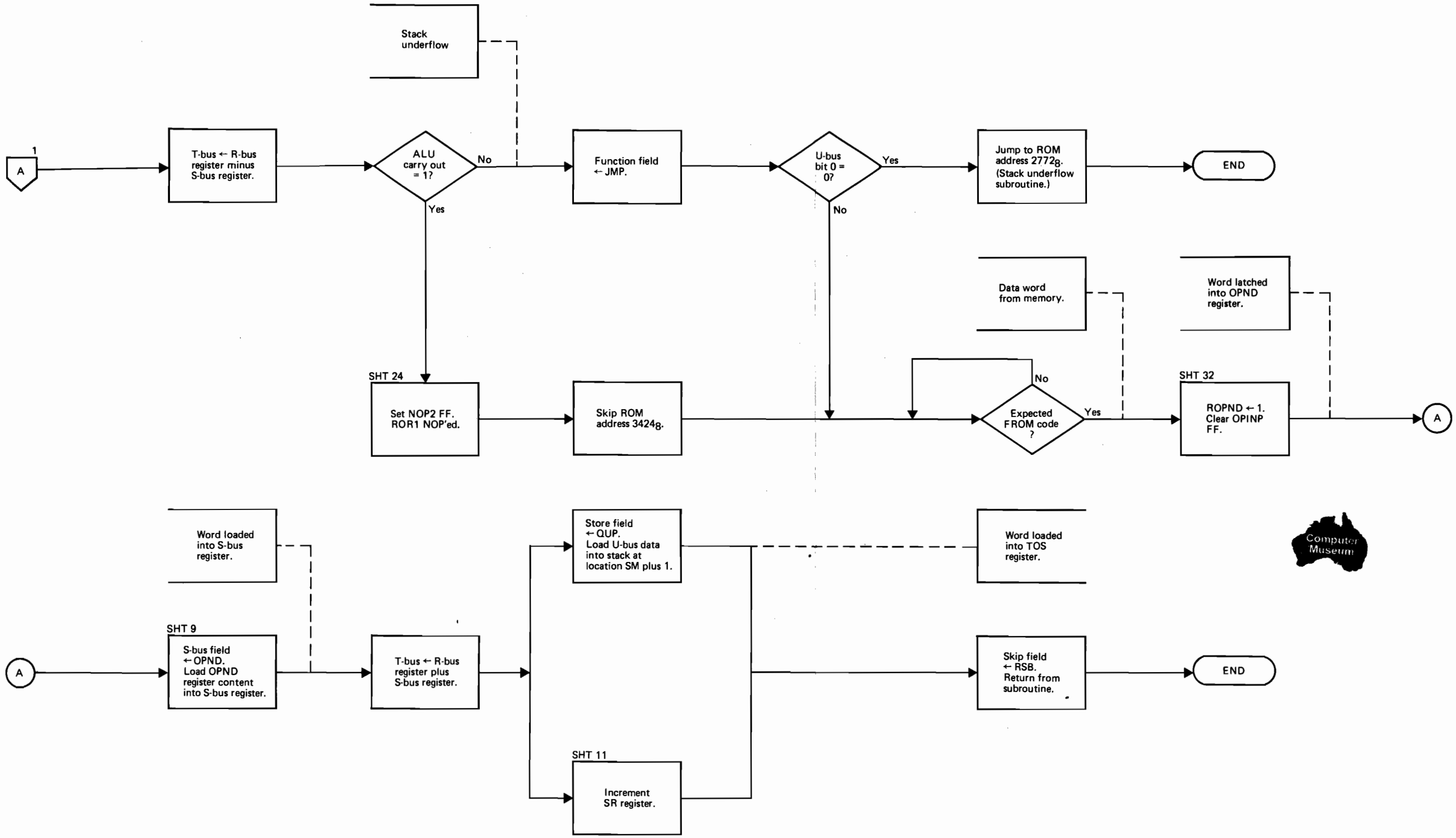


Figure 3-11. PUL1 Subroutine Operational Flow Diagram (Sheet 2 of 2)

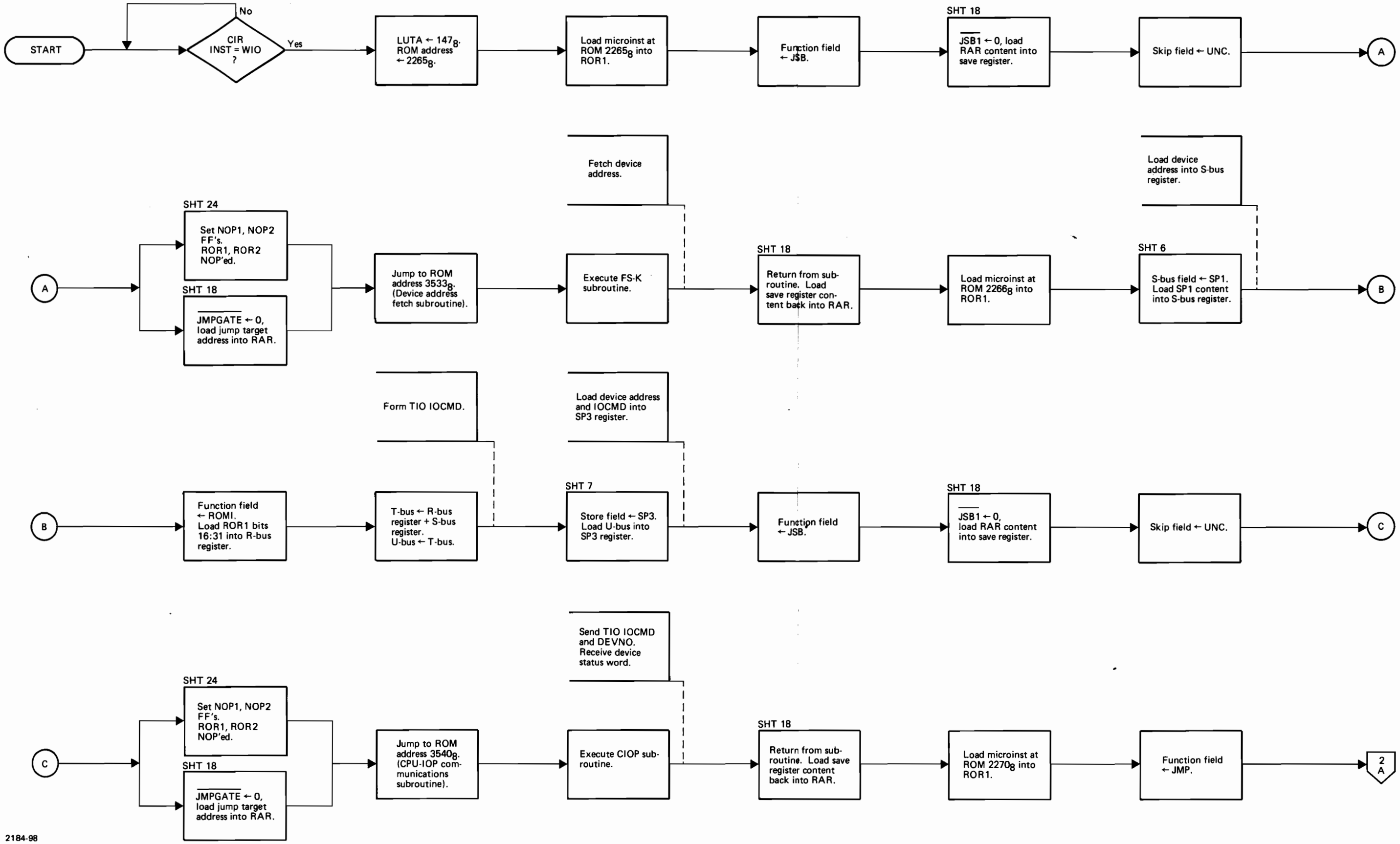


Figure 3-12. WIO Command Operational Flow Diagram (Sheet 1 of 4)

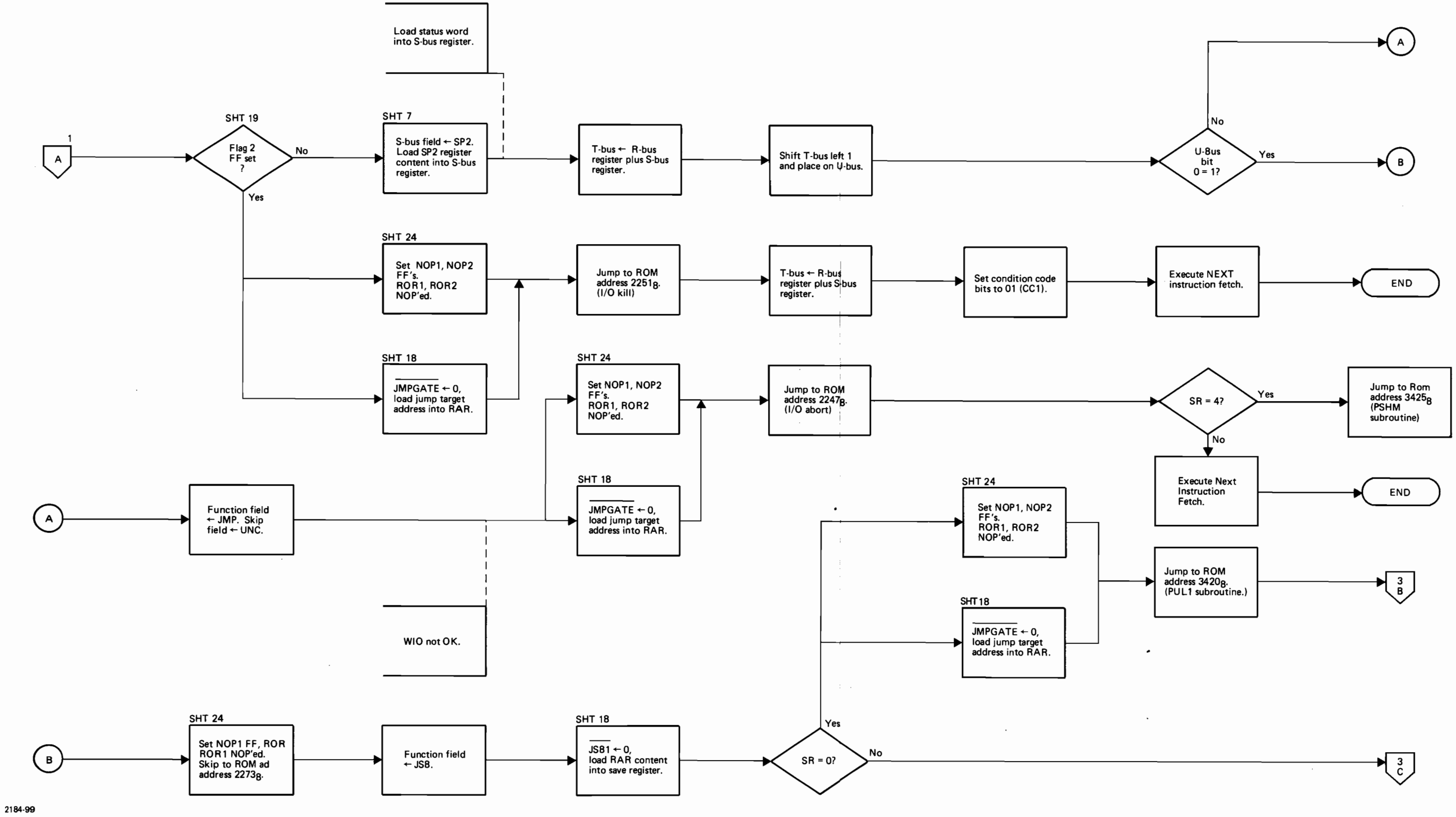
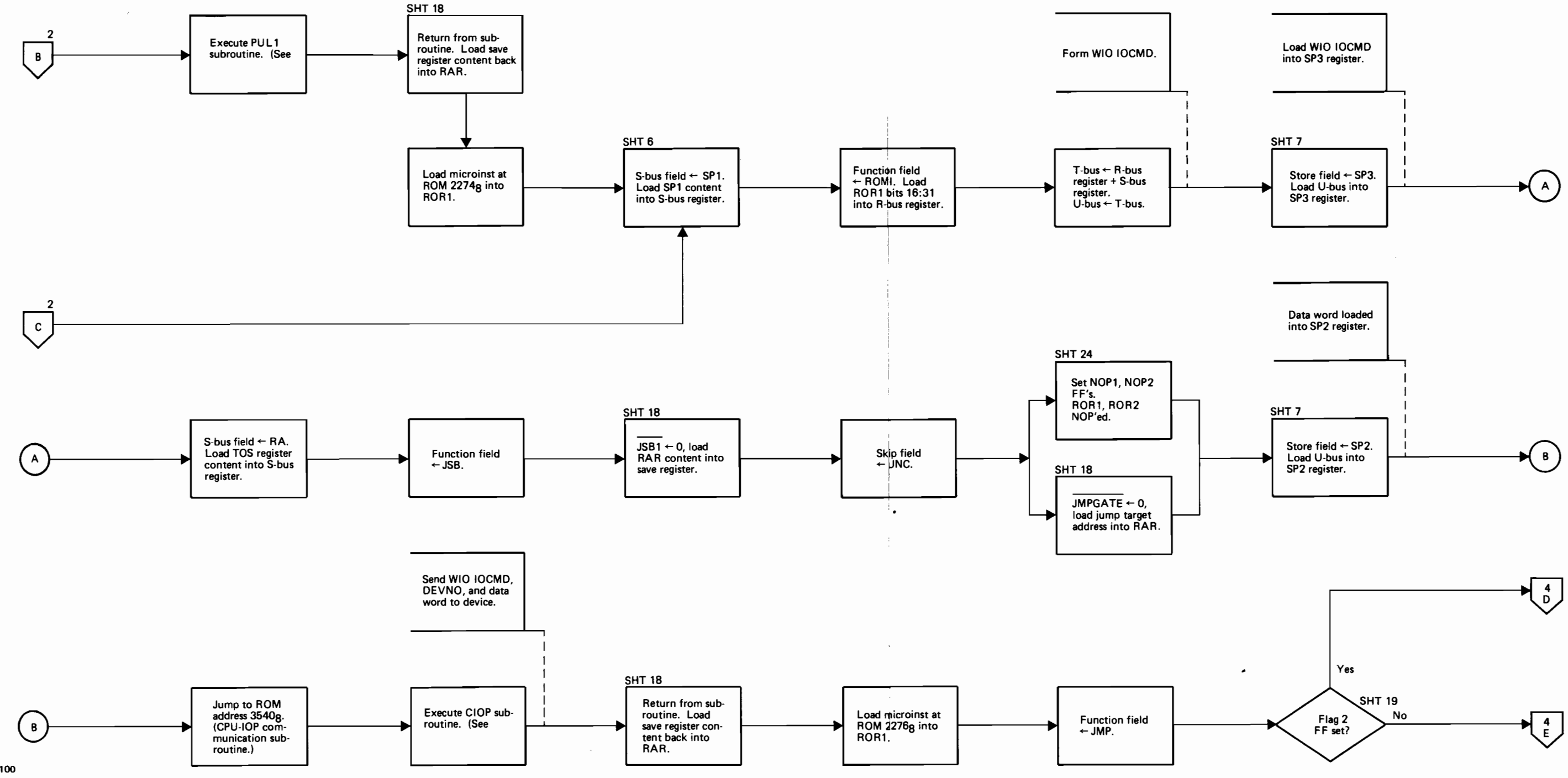
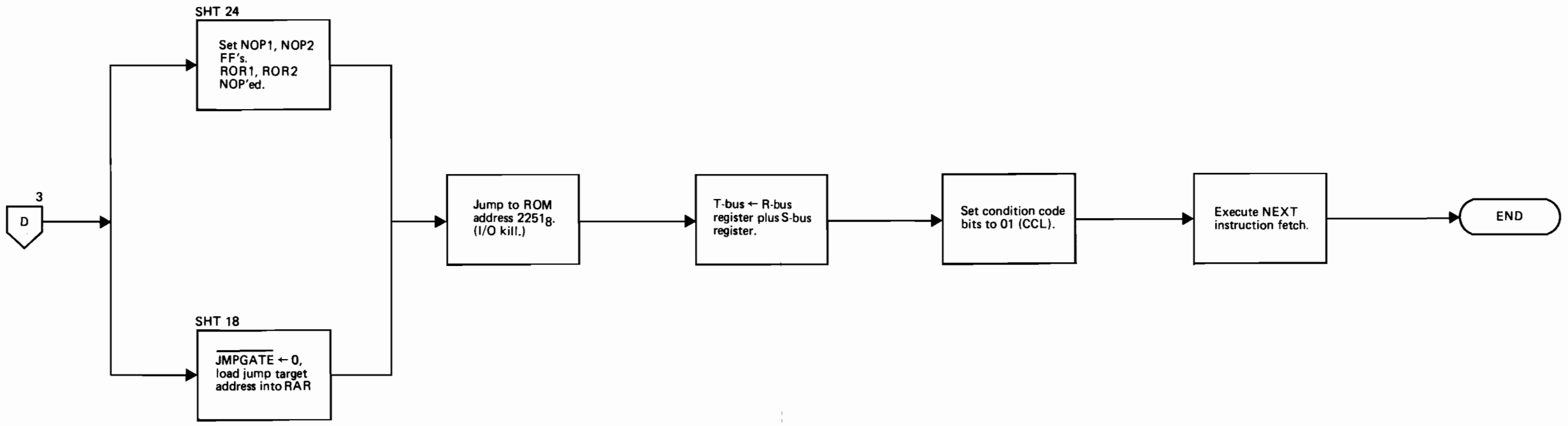


Figure 3-12. WIO Command Operational Flow Diagram (Sheet 2 of 4)



2184-100

Figure 3-12. WIO Command Operational Flow Diagram (Sheet 3 of 4)



2184-101

Figure 3-12. WIO Command Operational Flow Diagram (Sheet 4 of 4)

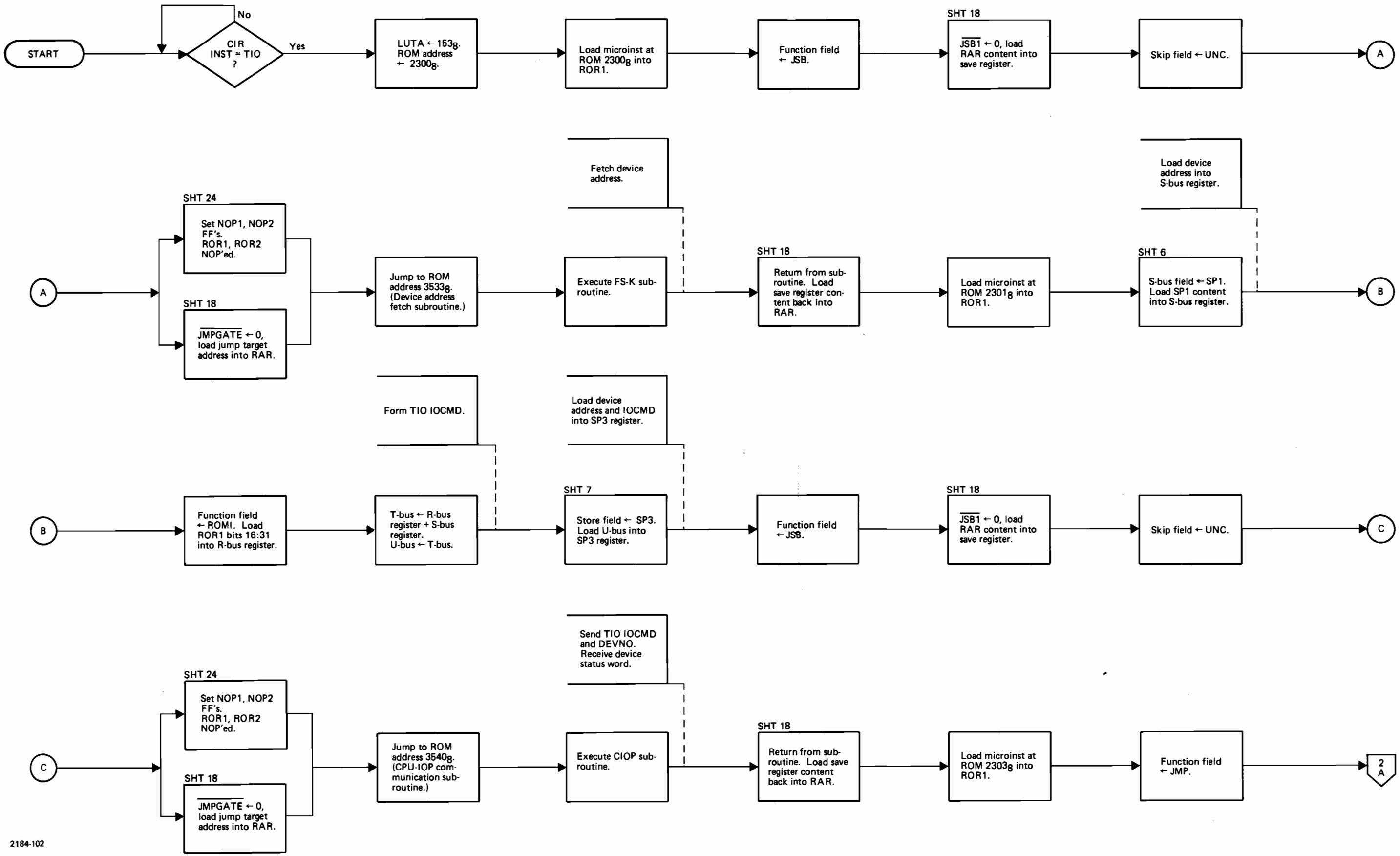


Figure 3-13. TIO Command Operational Flow Diagram (Sheet 1 of 2)

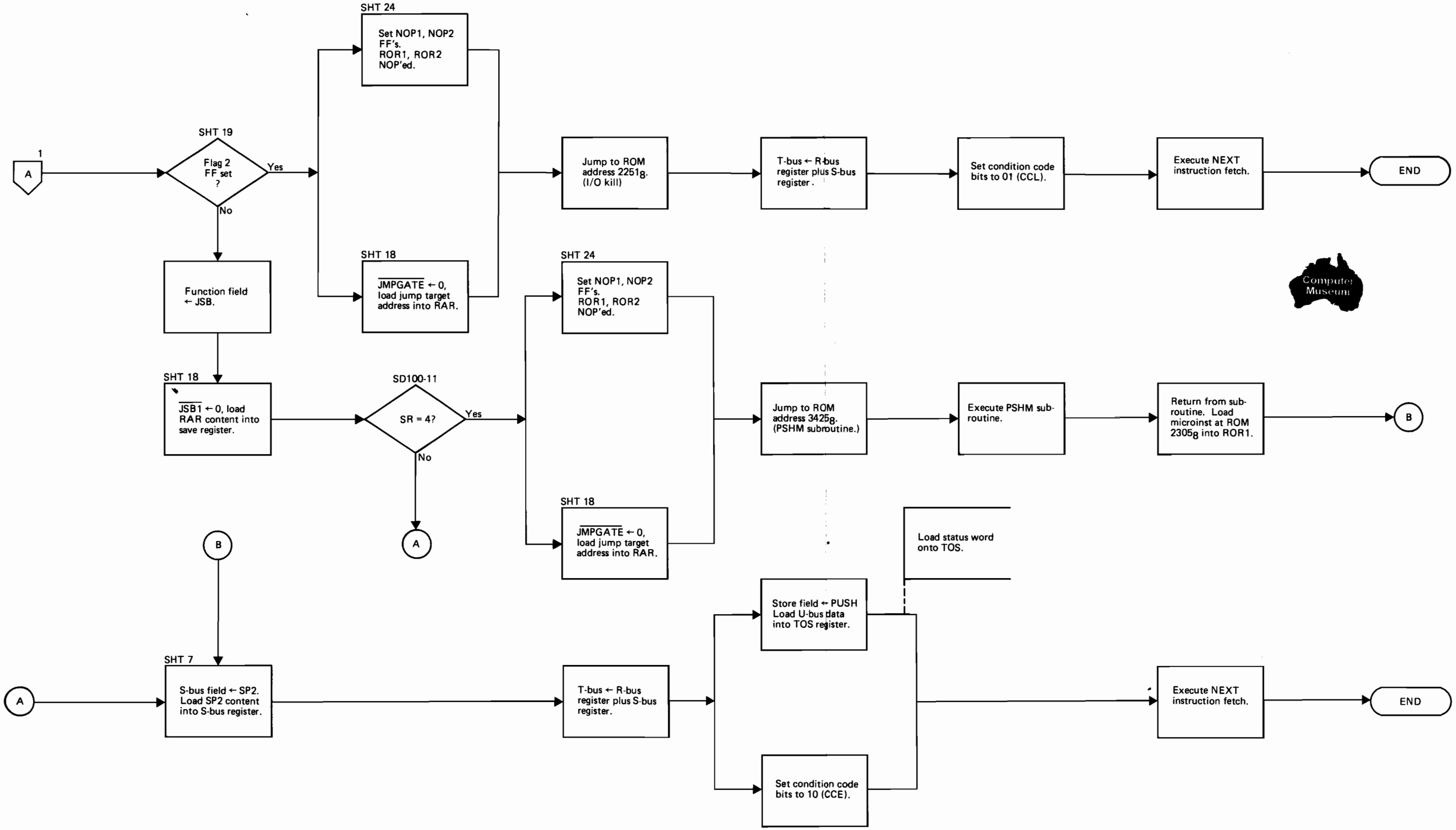


Figure 3-13. TIO Command Operational Flow Diagram (Sheet 2 of 2)

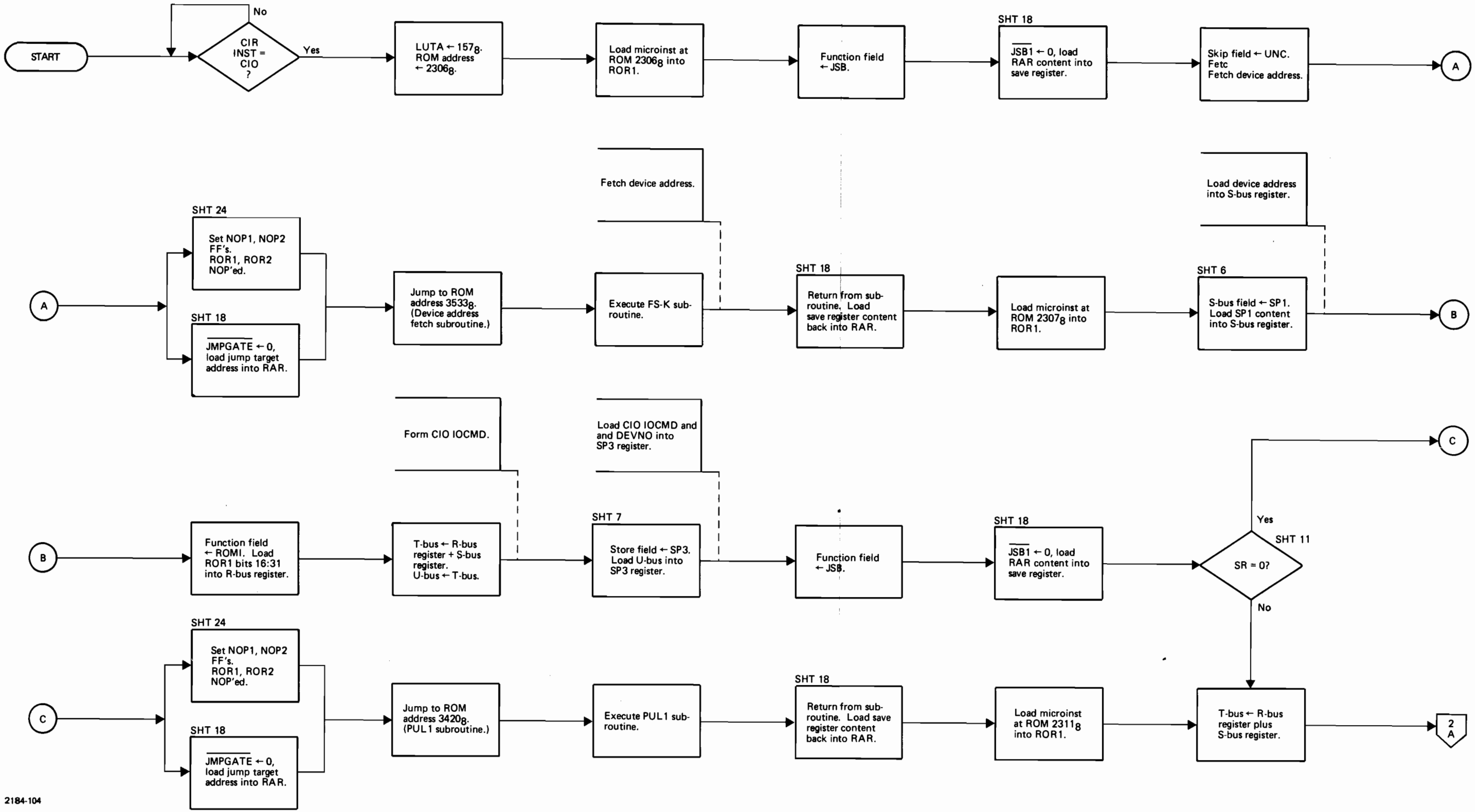
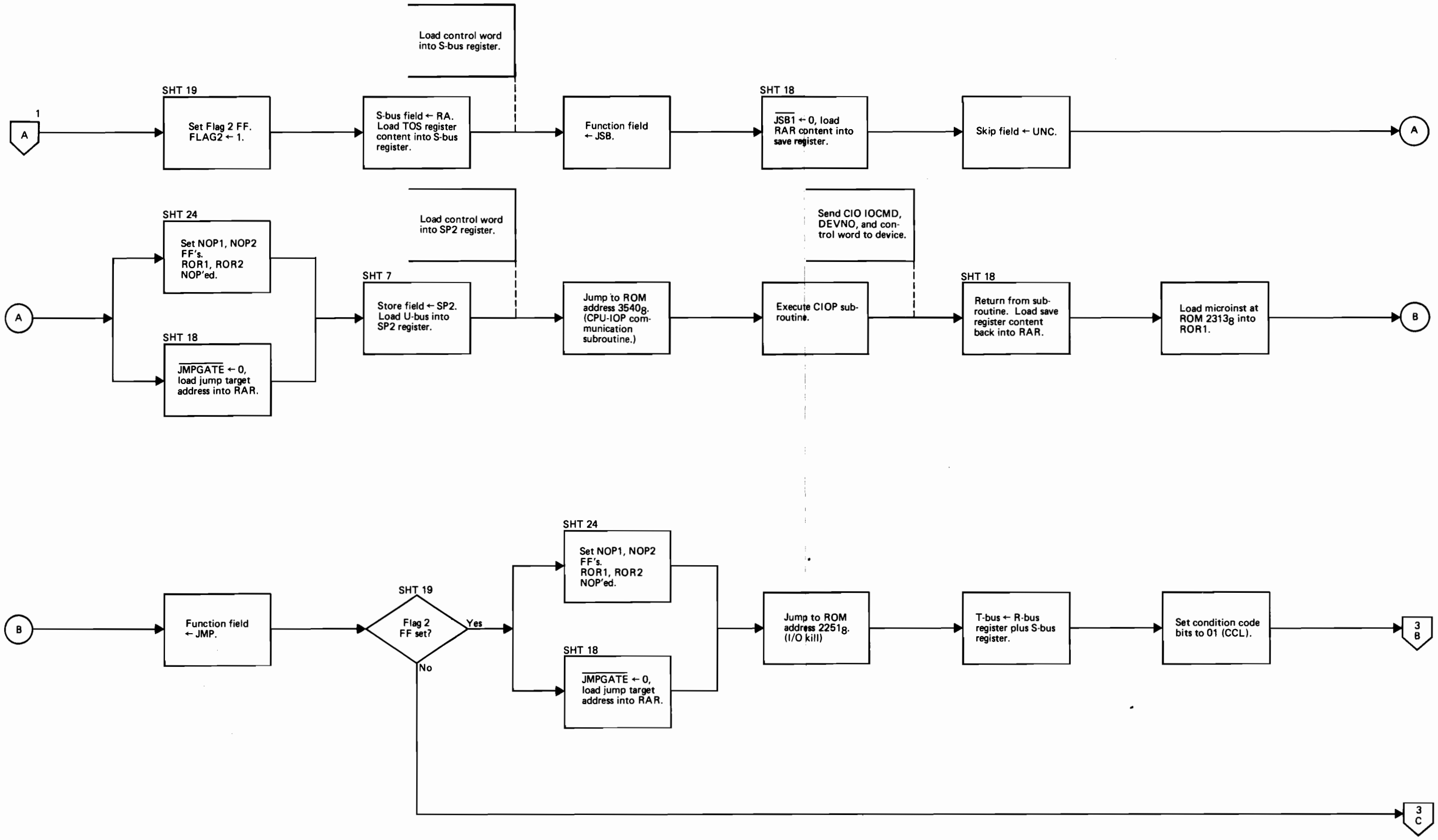


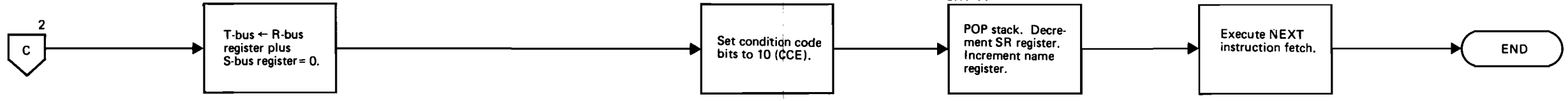
Figure 3-14. CIO Command Operational Flow Diagram (Sheet 1 of 3)



2184-105

Figure 3-14. CIO Command Operational Flow Diagram (Sheet 2 of 3)

SHT 11



2184-106

Figure 3-14. CIO Command Operational Flow Diagram (Sheet 3 of 3)

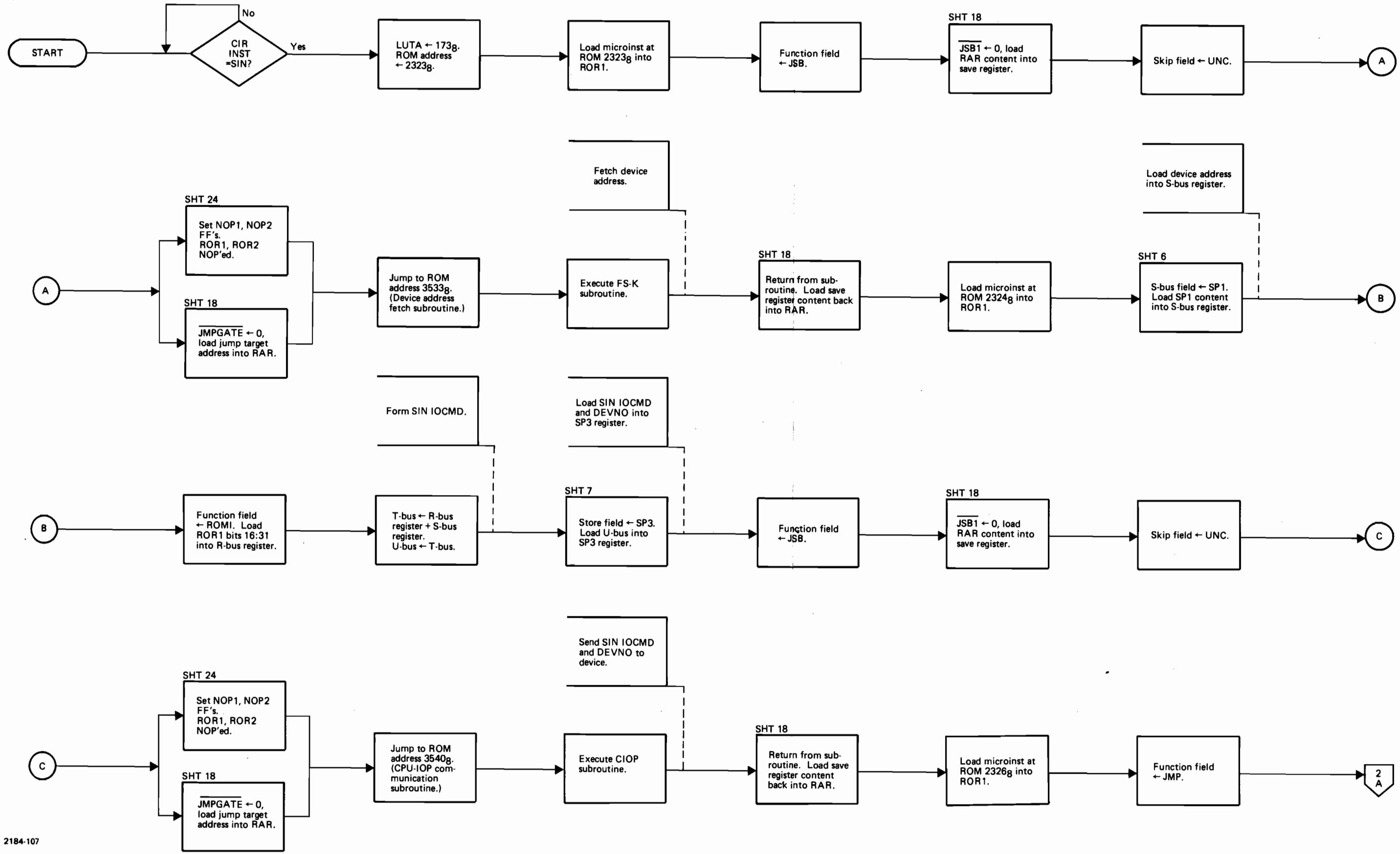
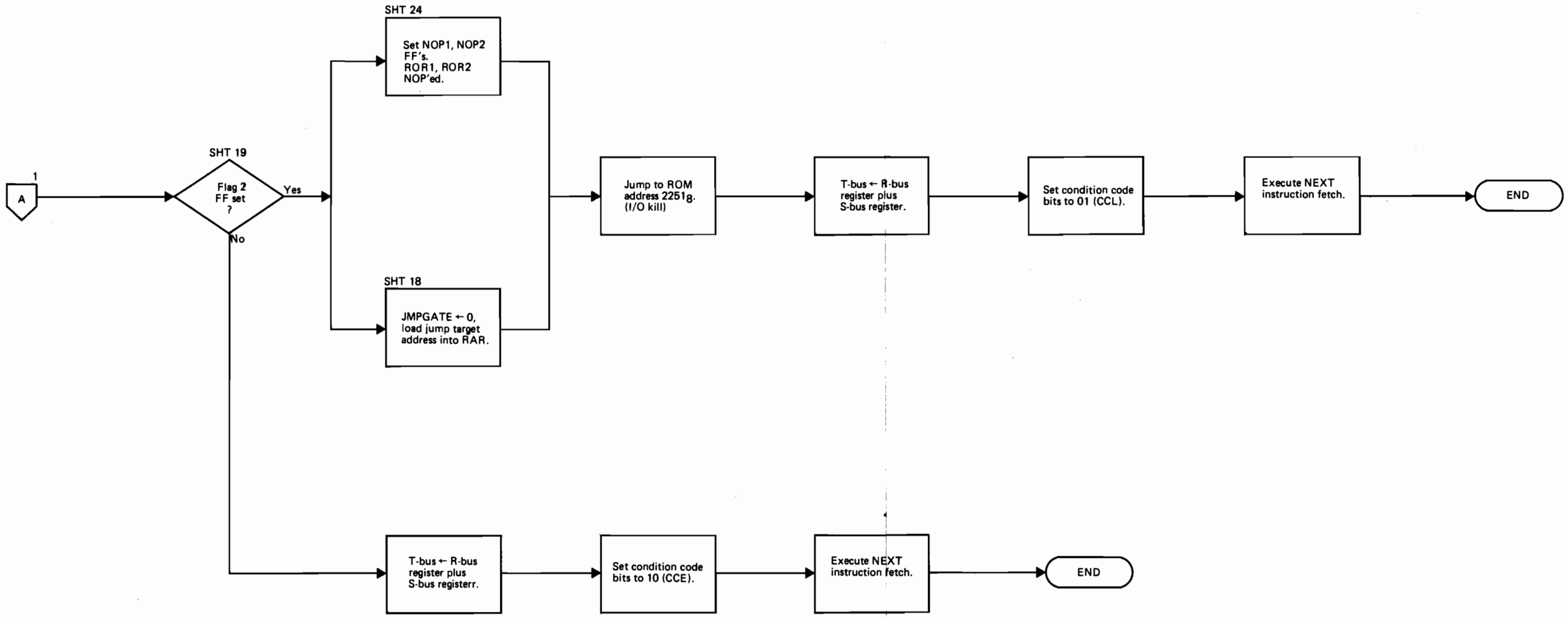
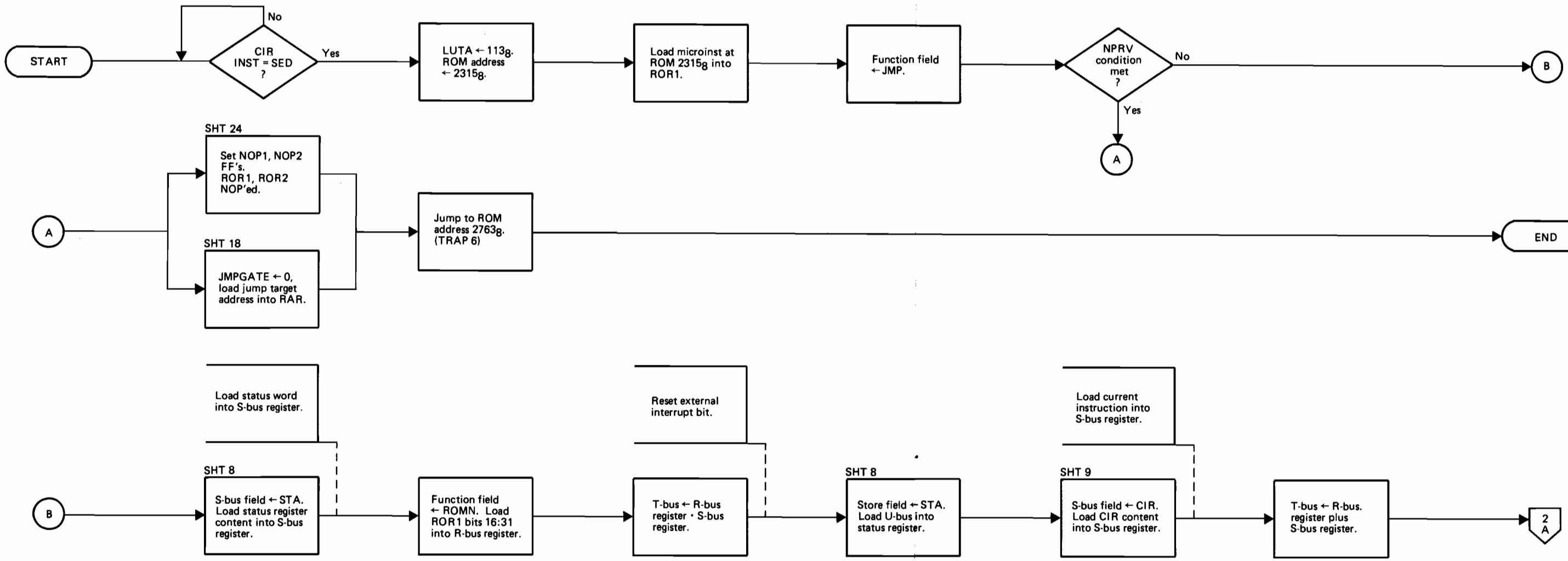


Figure 3-15. SIN Command Operational Flow Diagram (Sheet 1 of 2)



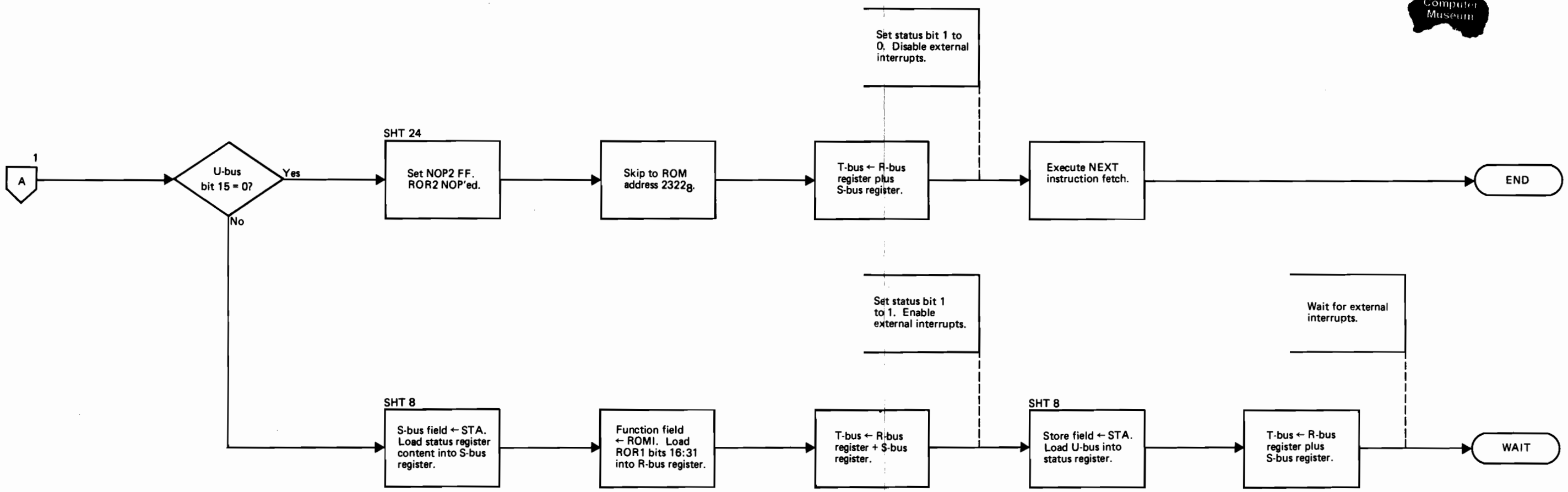
2184-108

Figure 3-15. SIN Command Operational Flow Diagram (Sheet 2 of 2)



2184-109

Figure 3-16. SED Command Operational Flow Diagram (Sheet 1 of 2)



2184-110

Figure 3-16. SED Command Operational Flow Diagram (Sheet 2 of 2)

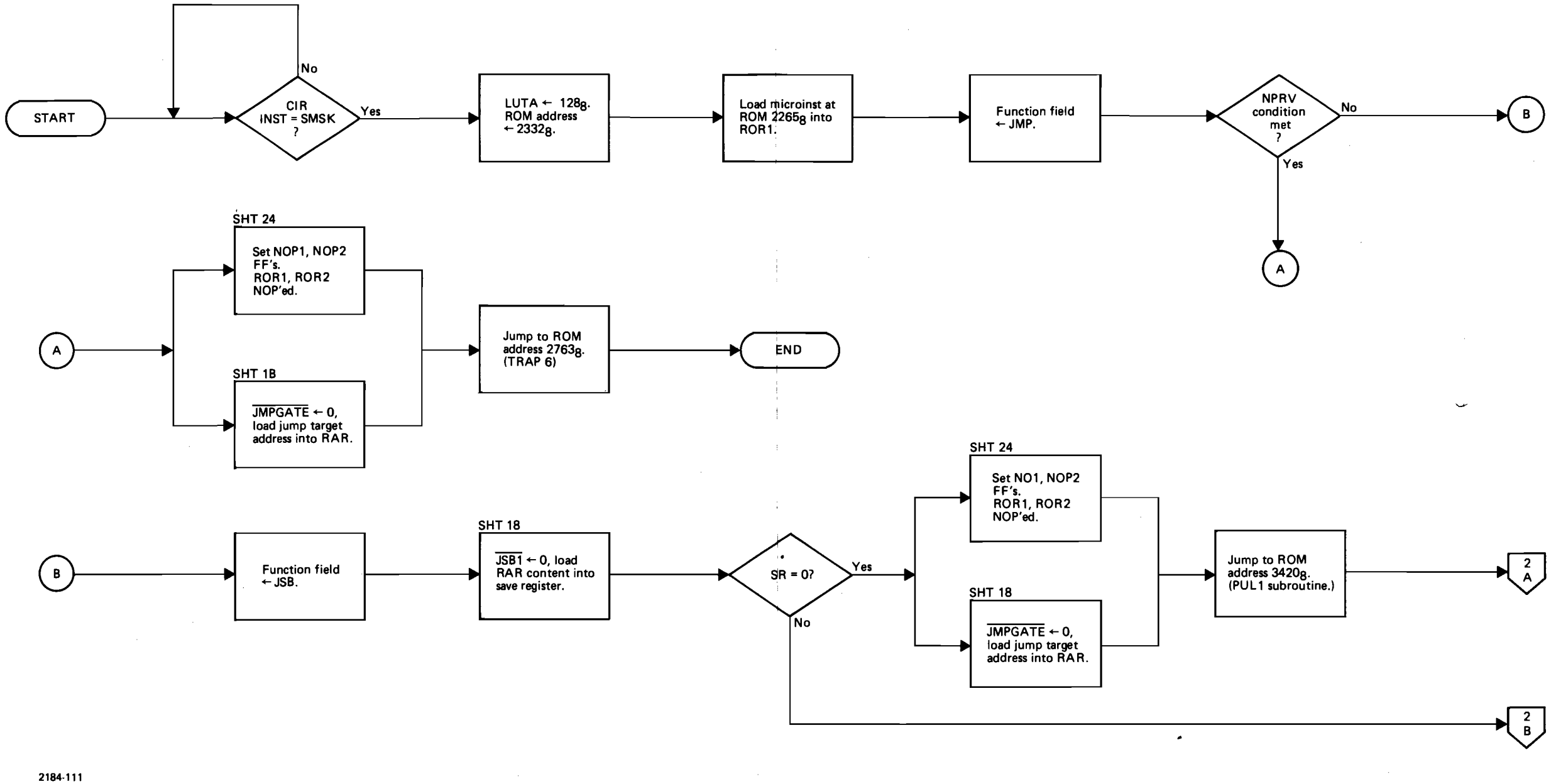


Figure 3-17. SMSK Command Operational Flow Diagram (Sheet 1 of 3)

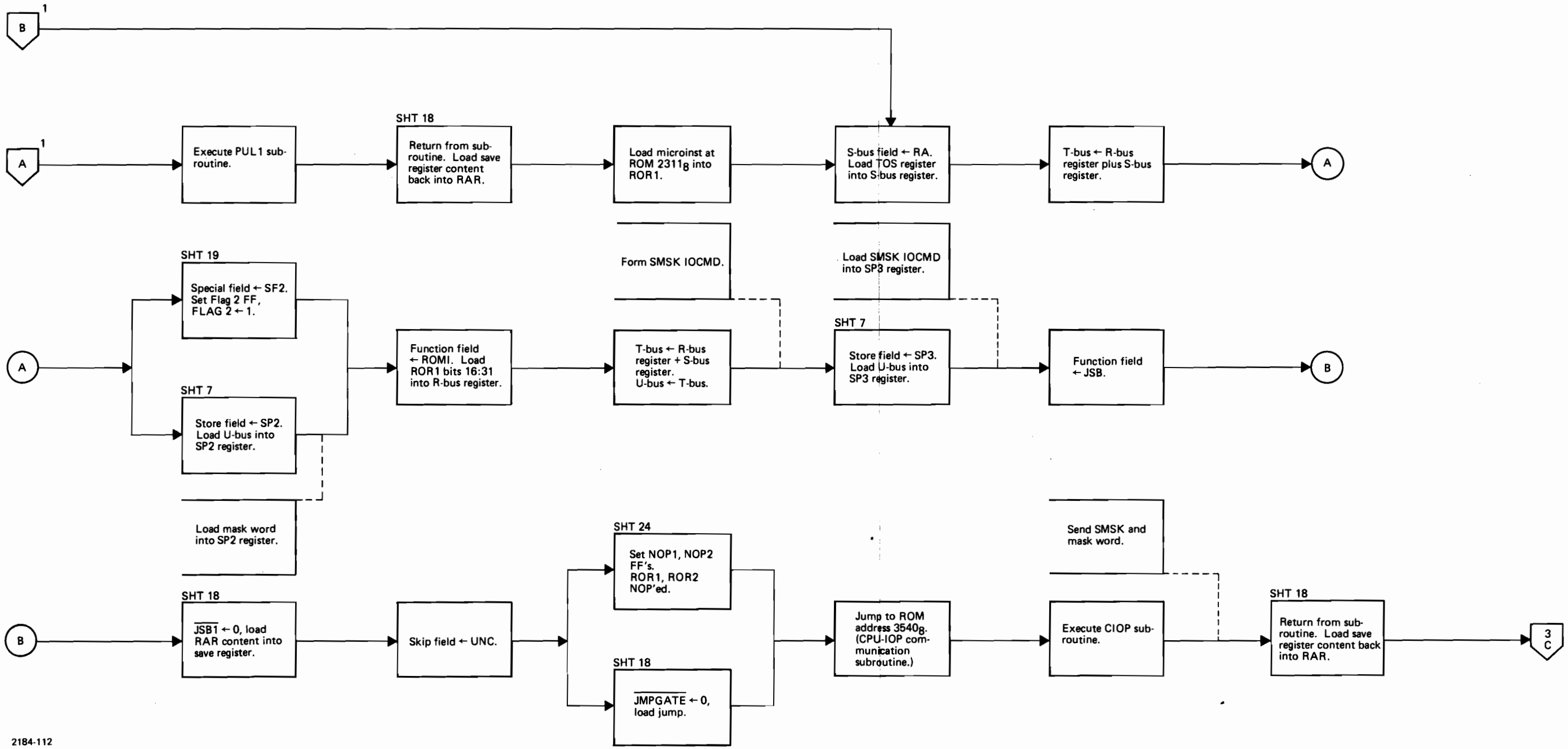


Figure 3-17. SMSK Command Operational Flow Diagram (Sheet 2 of 3)

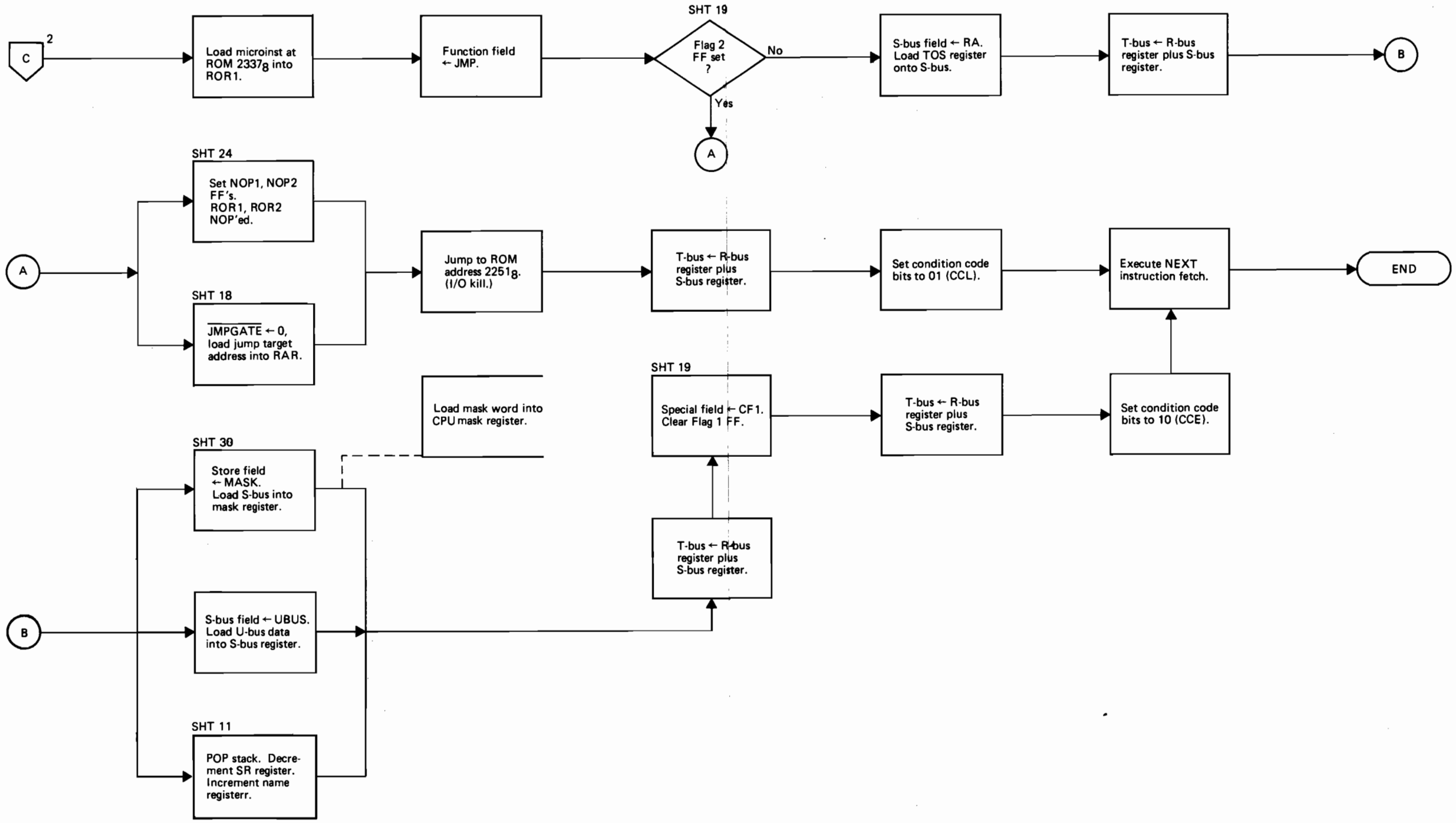


Figure 3-17. SMSK Command Operational Flow Diagram (Sheet 3 of 3)

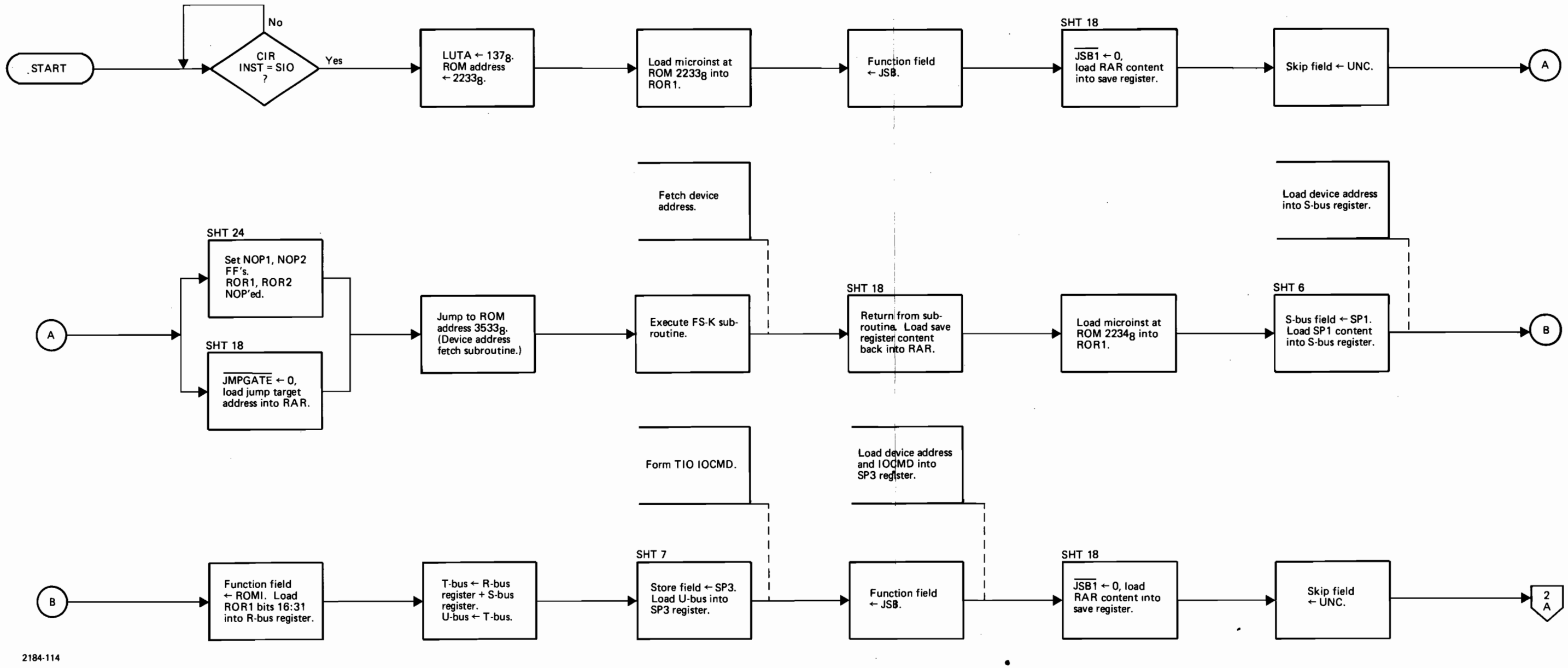


Figure 3-18. SIO Command Operational Flow Diagram (Sheet 1 of 5)

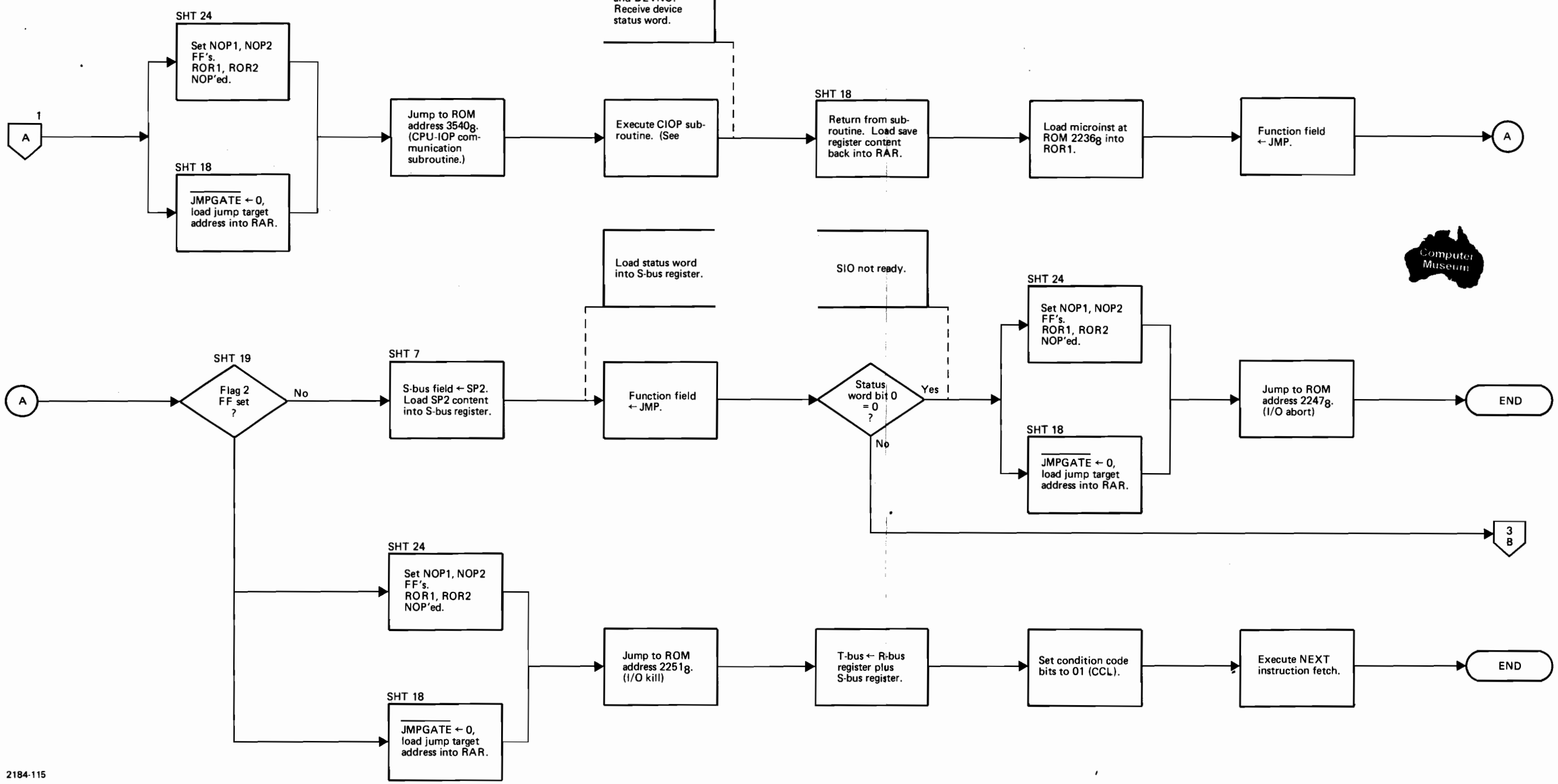
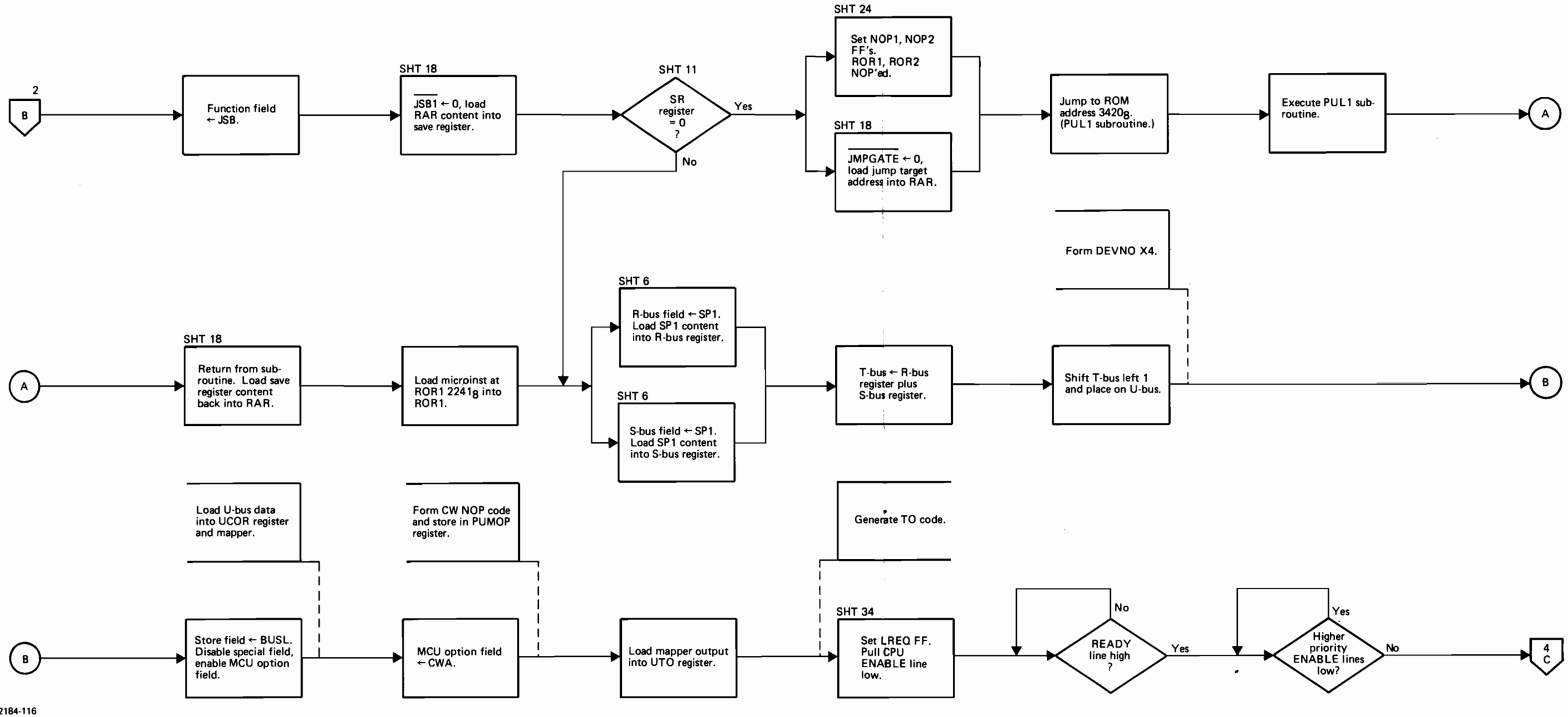
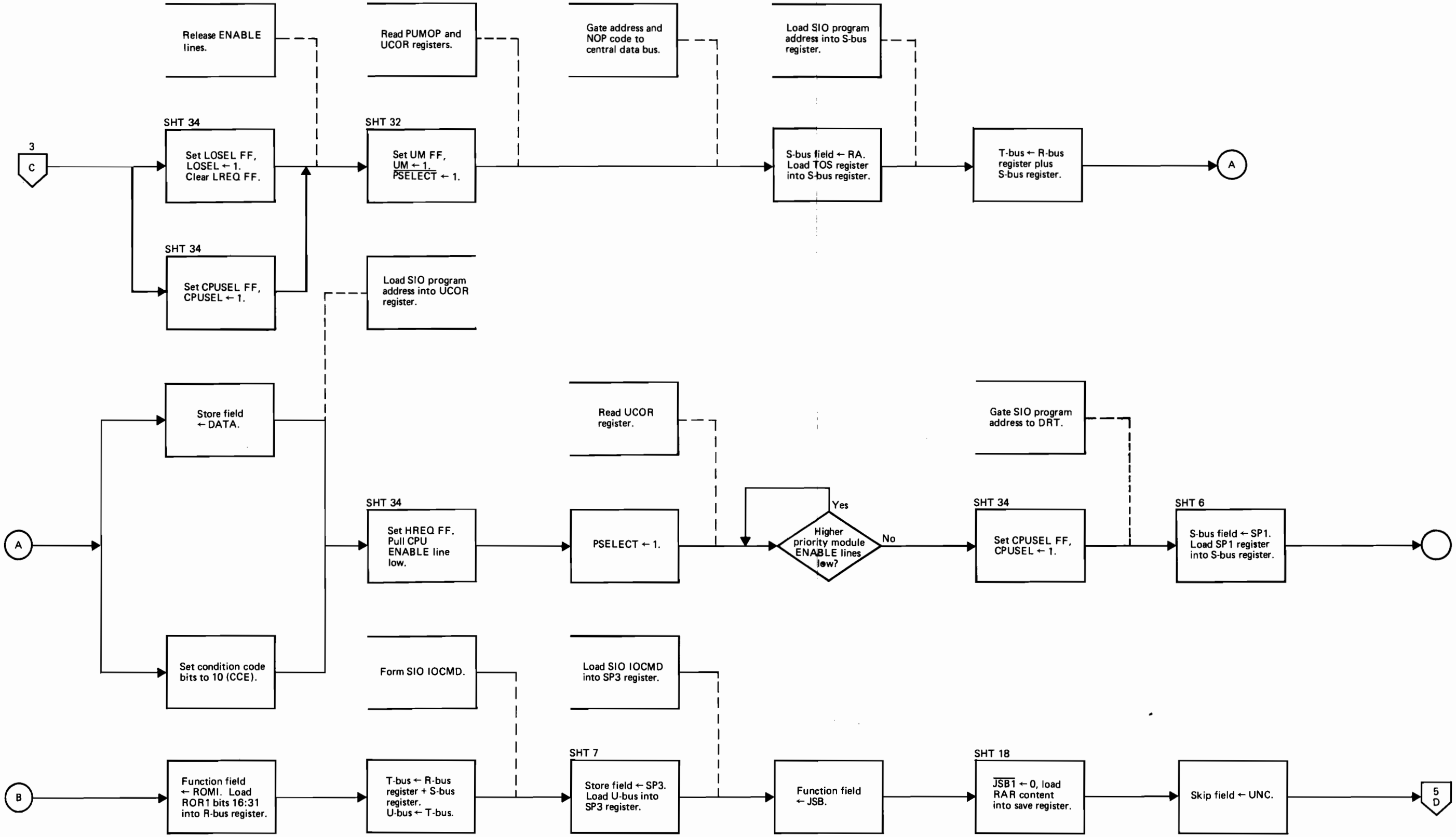


Figure 3-18. SIO Command Operational Flow Diagram (Sheet 2 of 5)



2184-116

Figure 3-18. SIO Command Operational Flow Diagram (Sheet 3 of 5)



2184-117

Figure 3-18. SIO Command Operational Flow Diagram (Sheet 4 of 5)

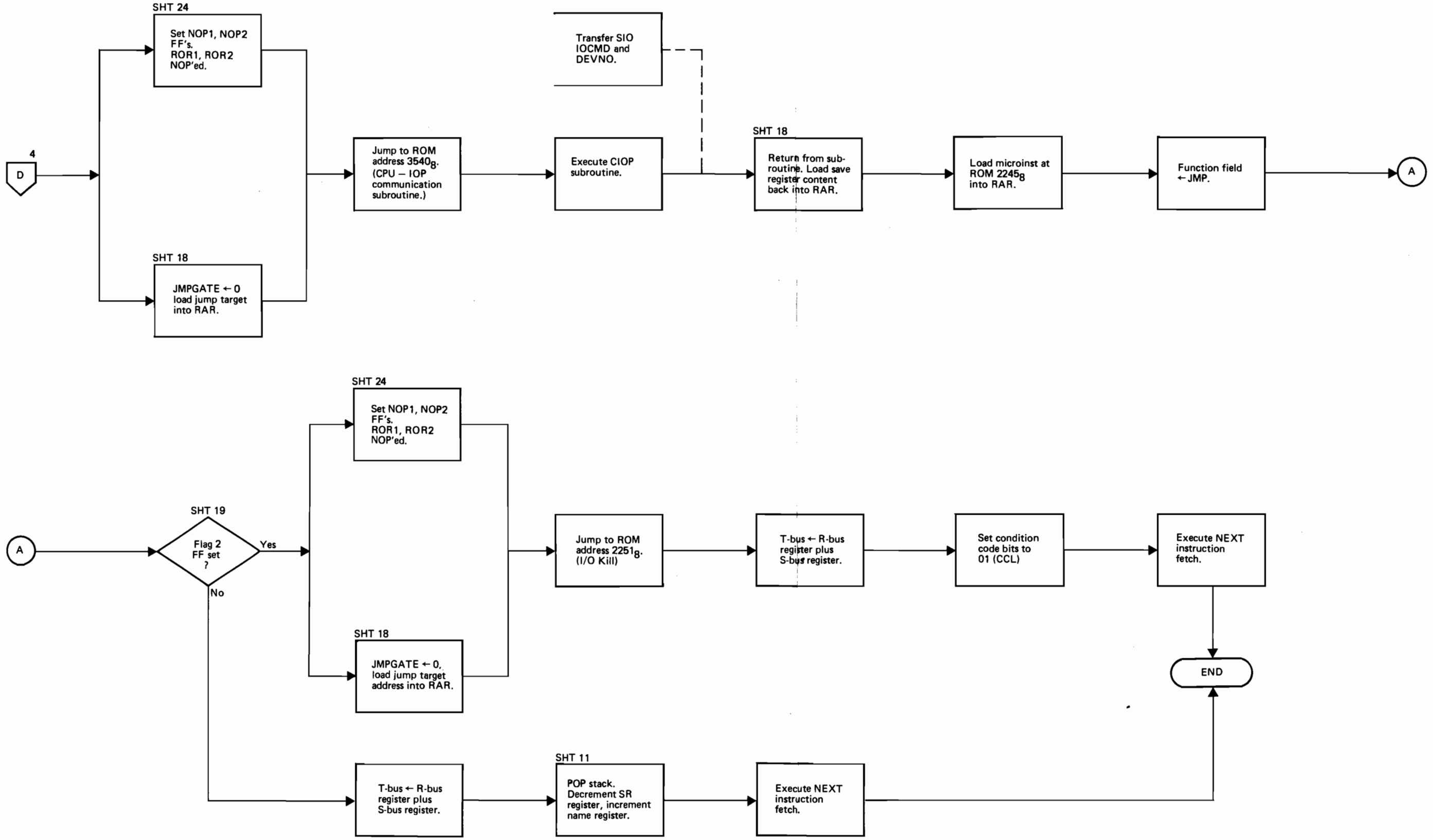
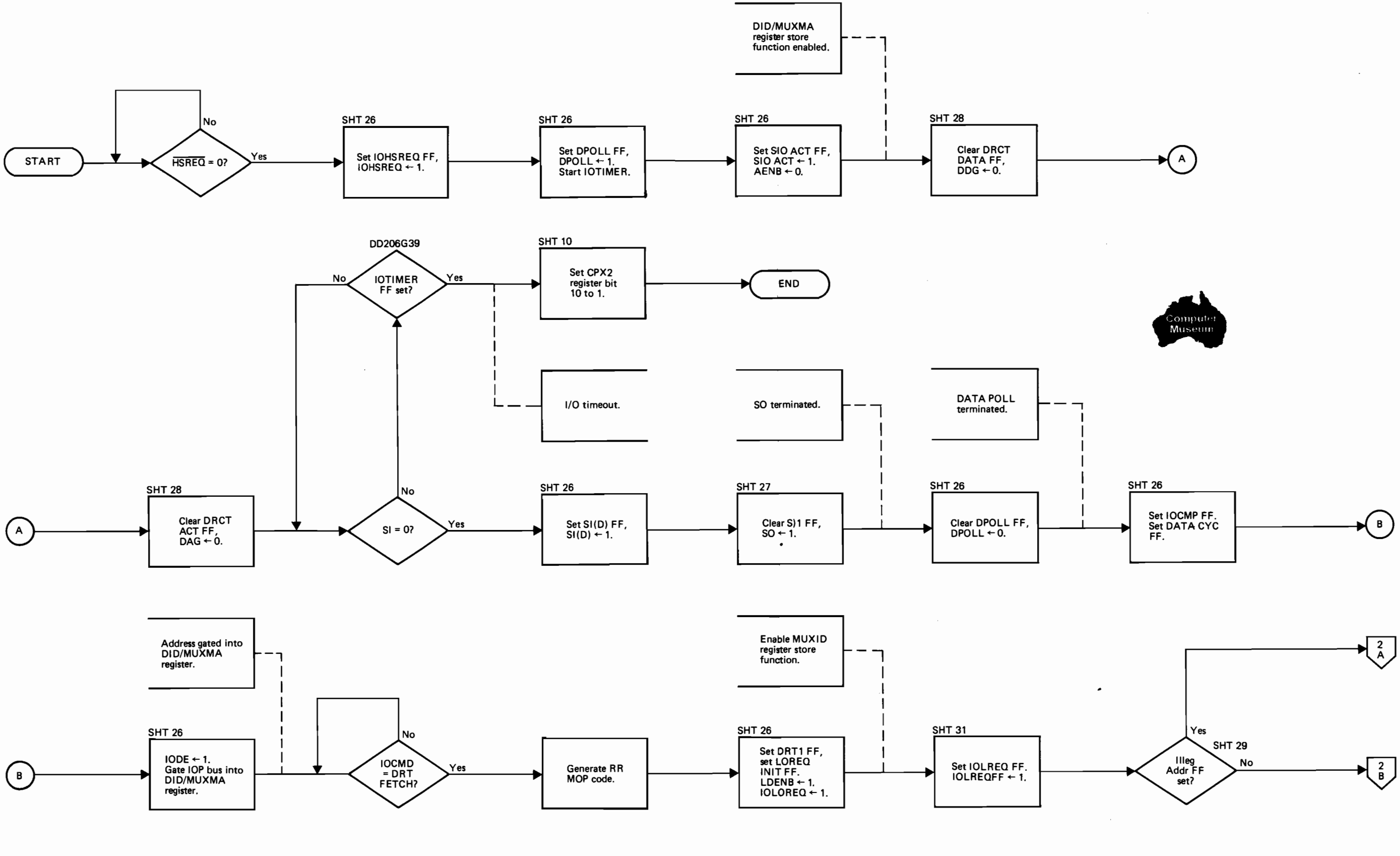
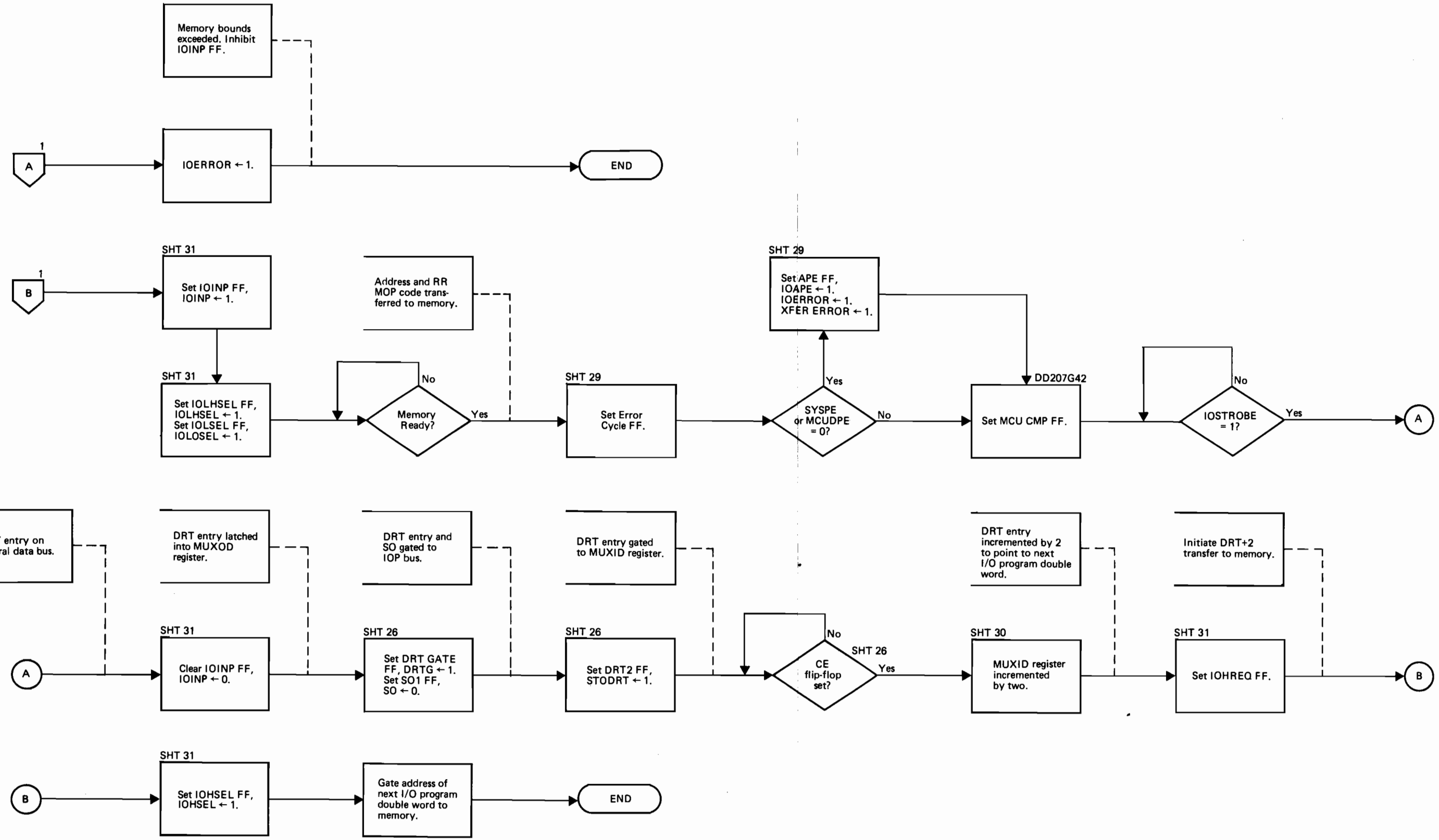


Figure 3-18. SIO Command Operational Flow Diagram (Sheet 5 of 5)



2184-119

Figure 3-19. DRTE Fetch/Store Operational Flow Diagram (Sheet 1 of 2)



2184-120

Figure 3-19. DRTE Fetch/Store Operational Flow Diagram (Sheet 2 of 2)

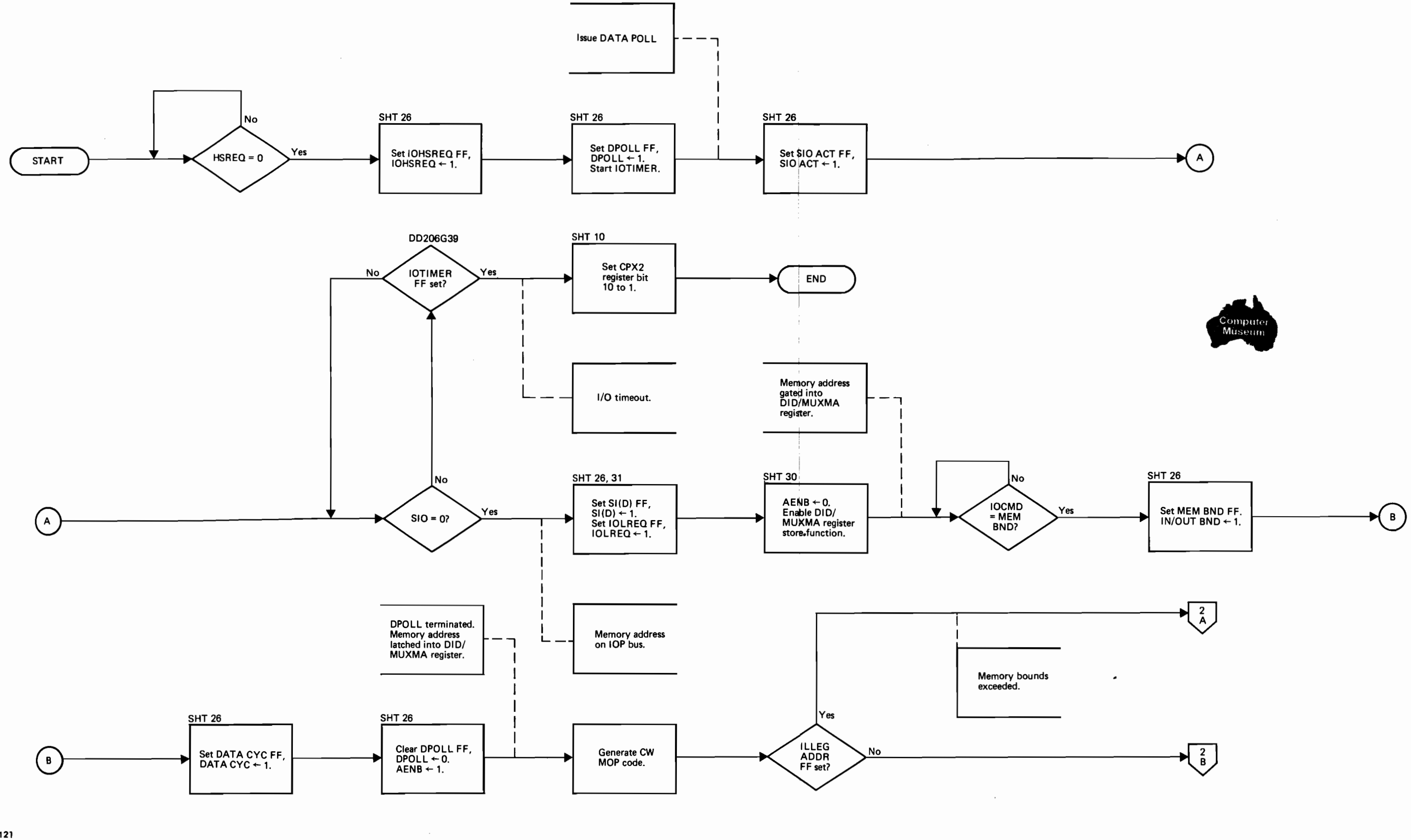
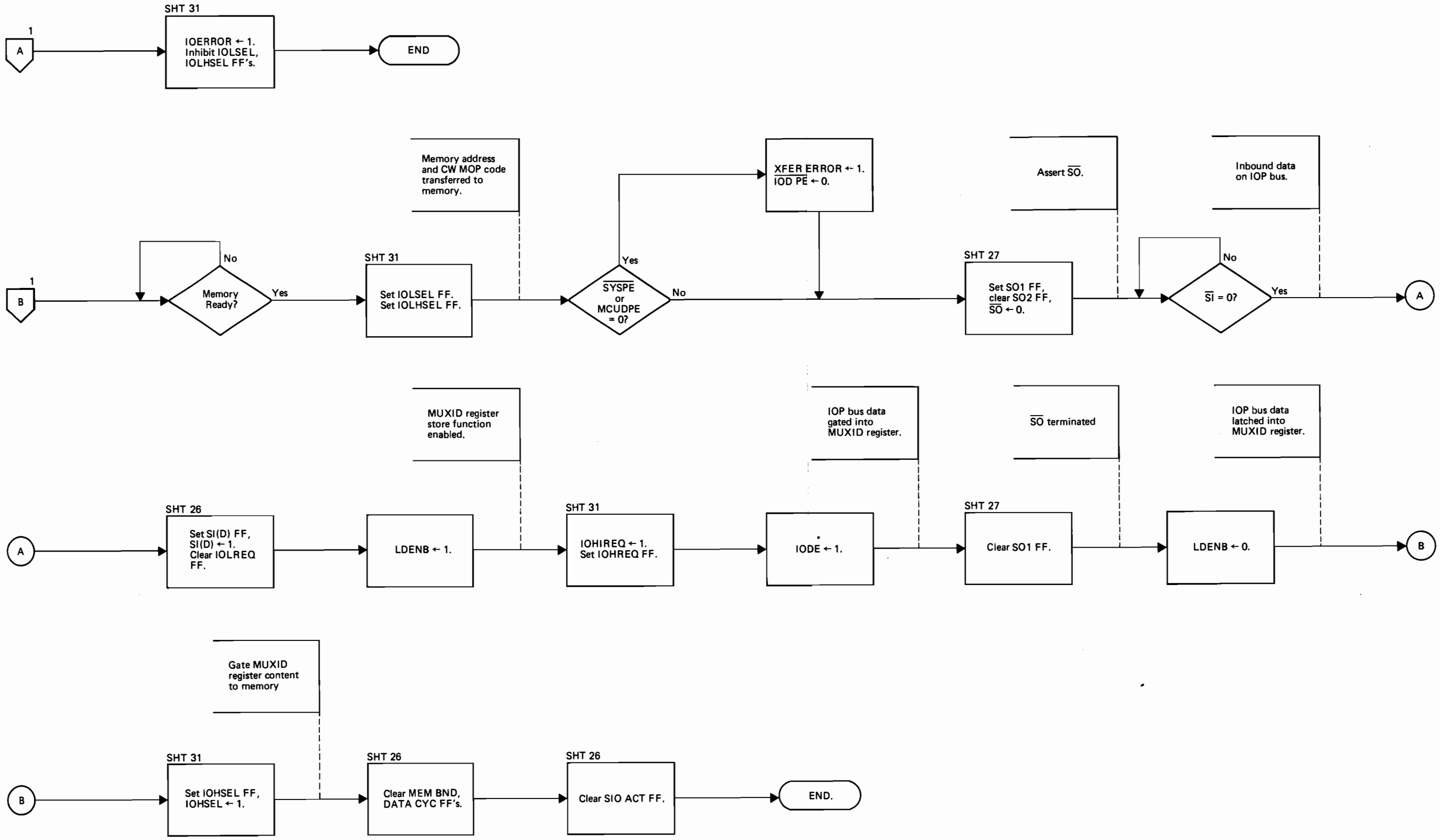


Figure 3-20. Memory Bound Transfer Operational Flow Diagram (Sheet 1 of 2)



2184-122

Figure 3-20. Memory Bound Transfer Operational Flow Diagram (Sheet 2 of 2)

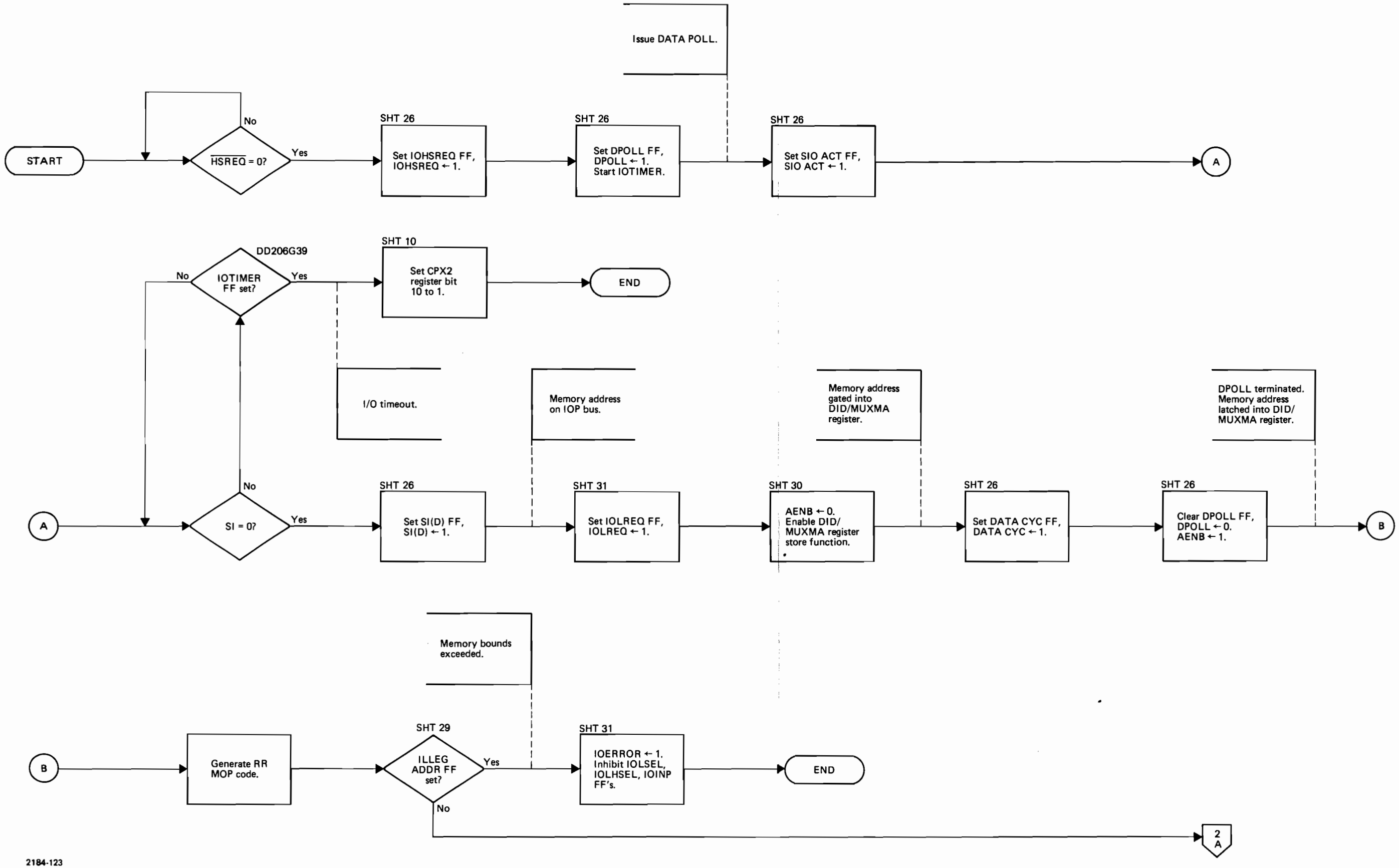


Figure 3-21. Device Bound Transfer Operational Flow Diagram (Sheet 1 of 2)

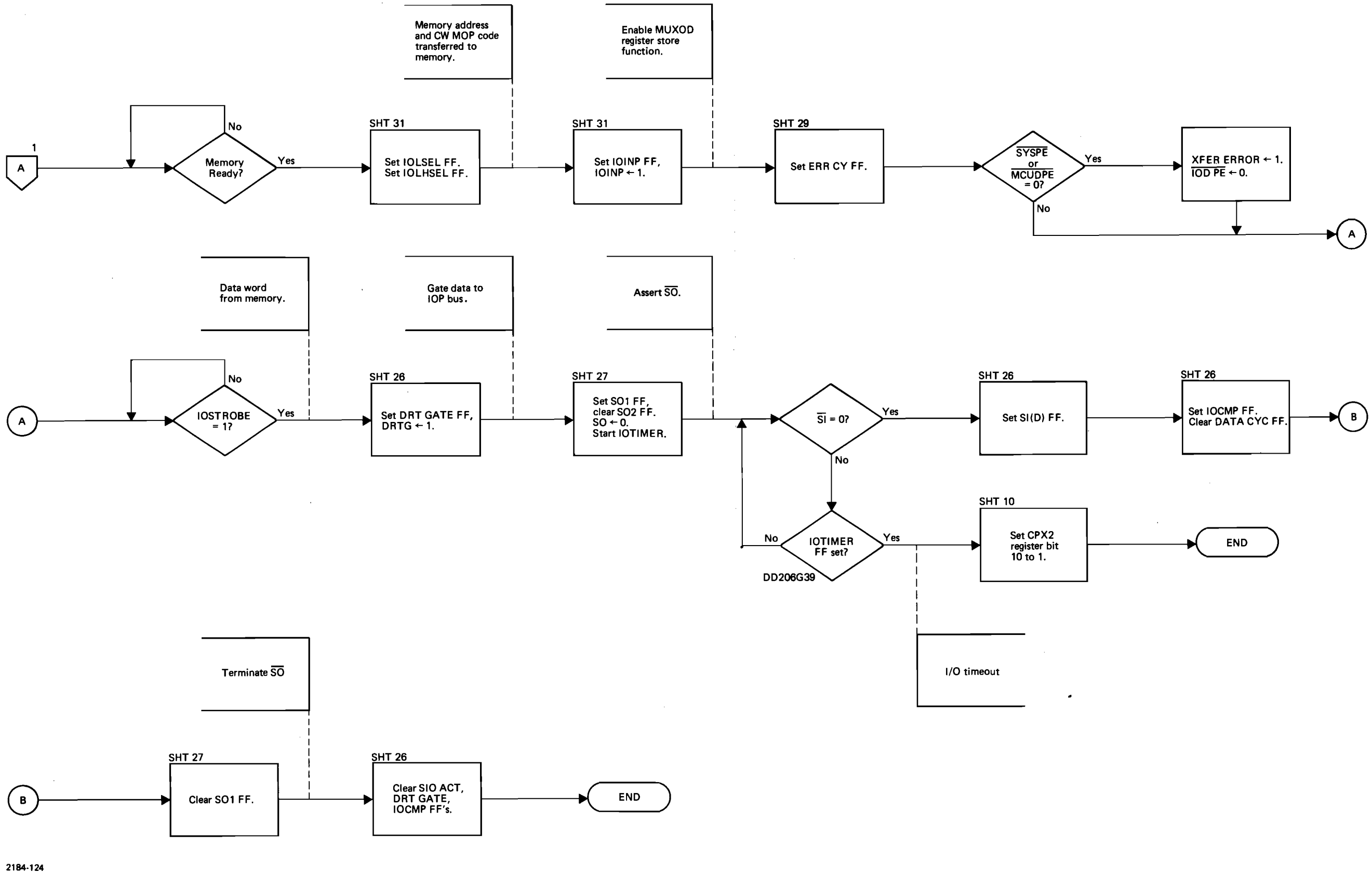


Figure 3-21. Device Bound Transfer Operational Flow Diagram (Sheet 2 of 2)

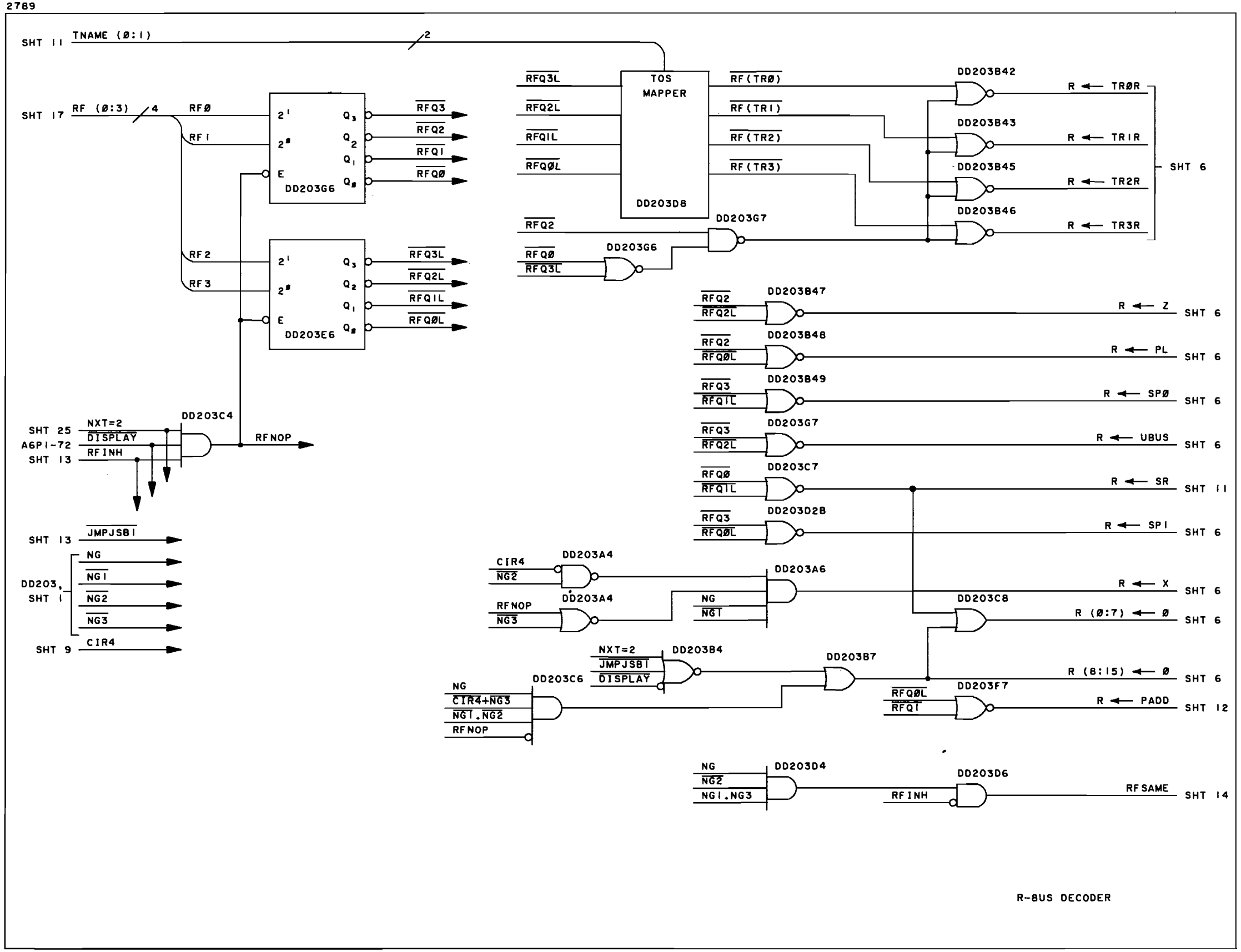


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 1 of 35)

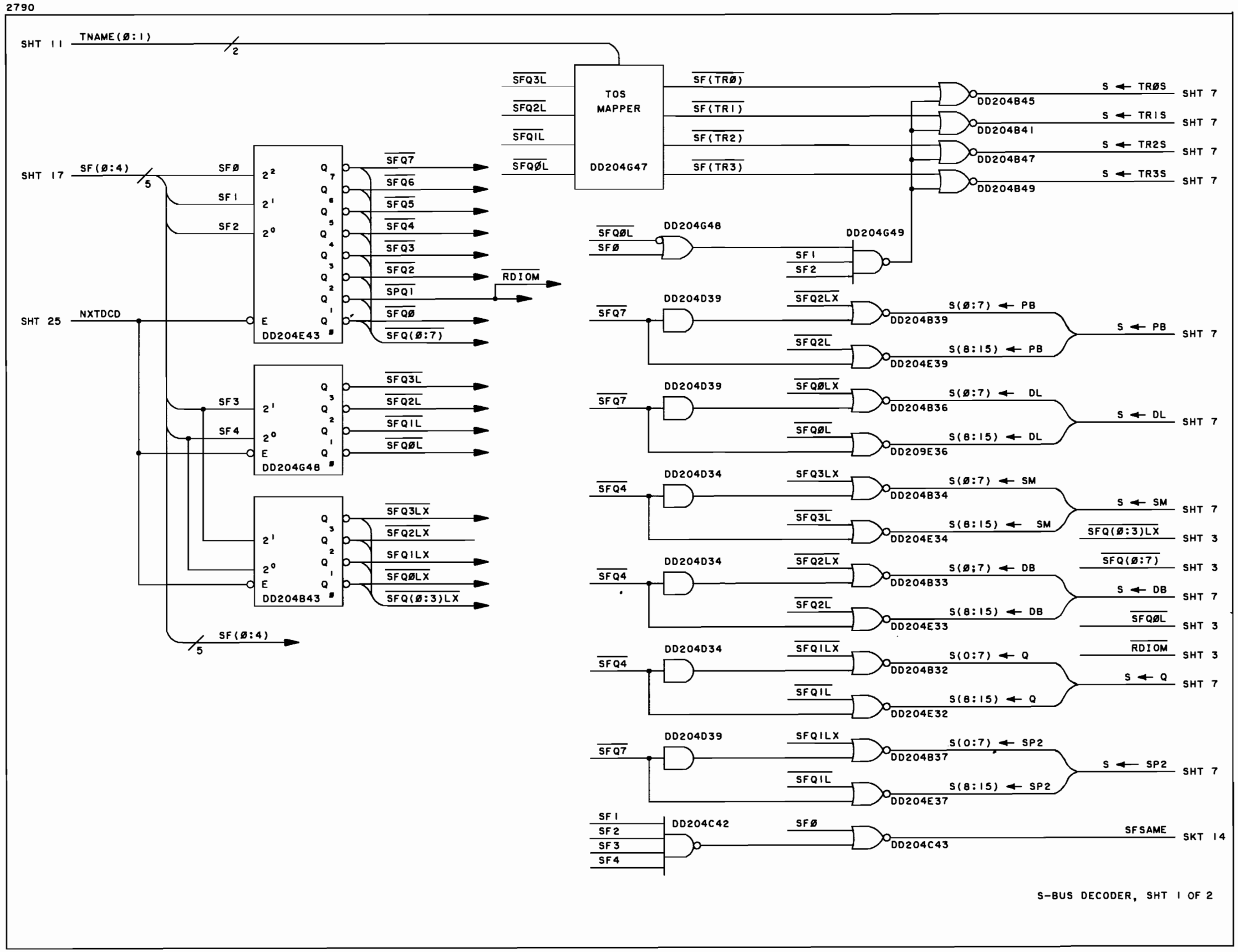


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 2 of 35)

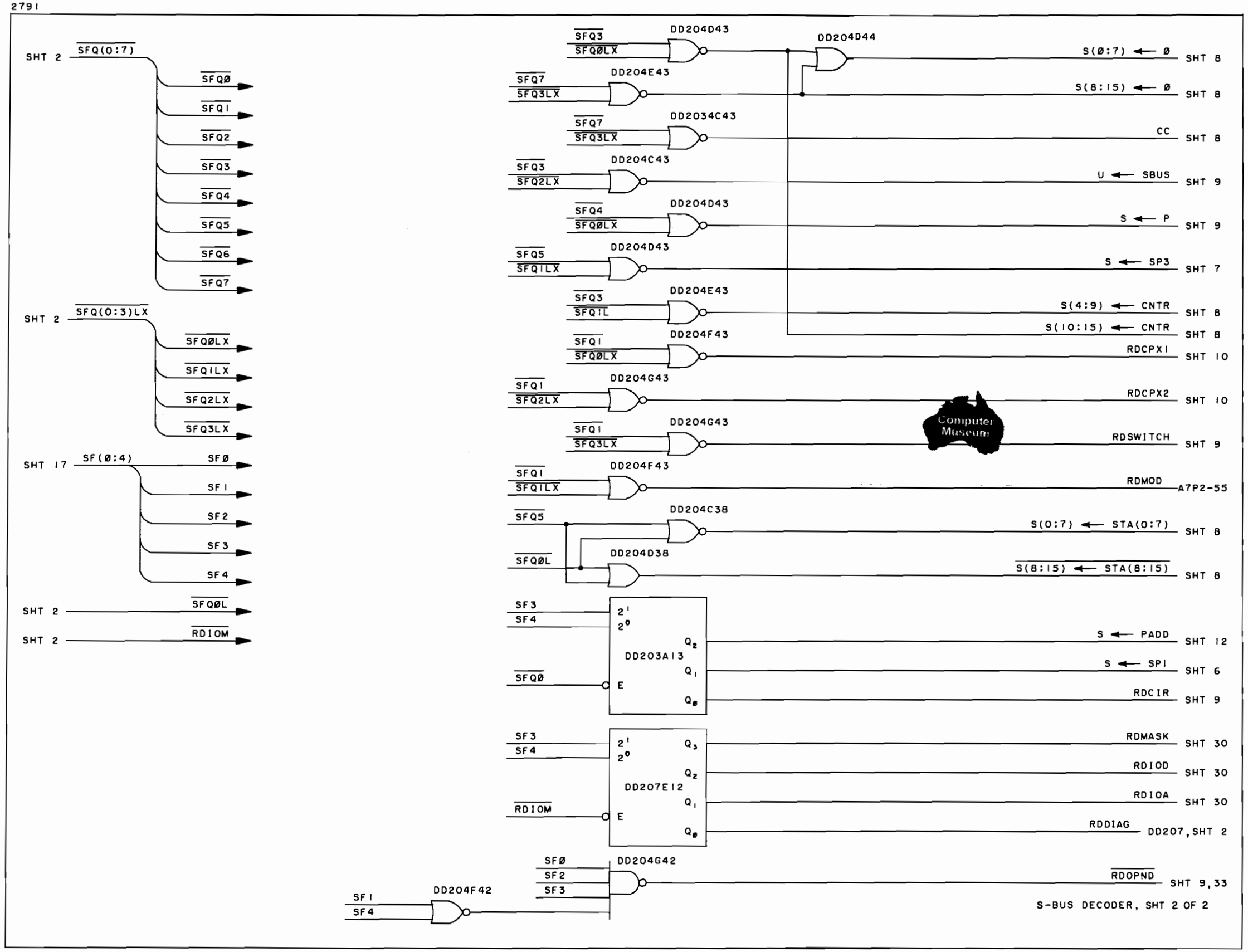


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 3 of 35)

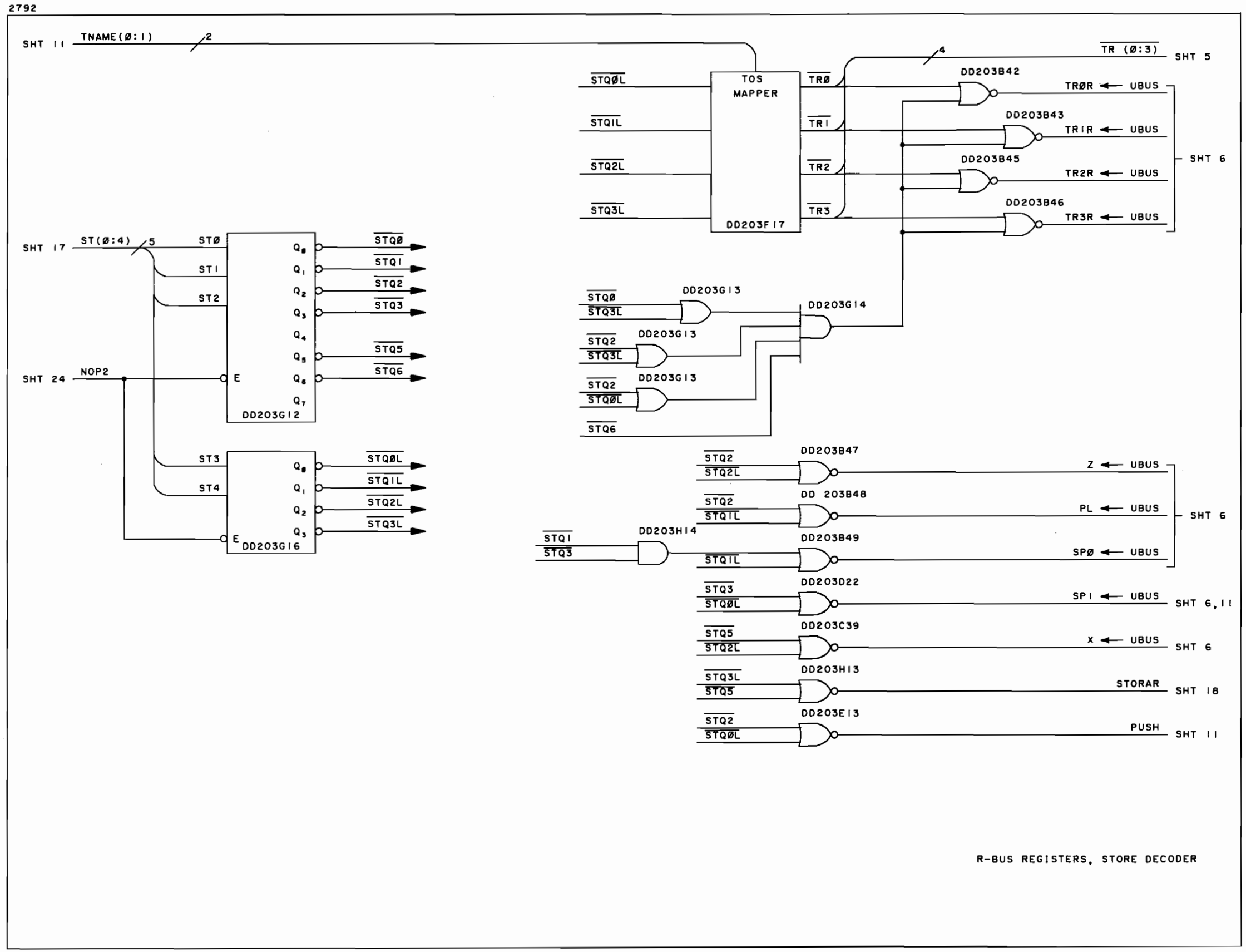


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 4 of 35)

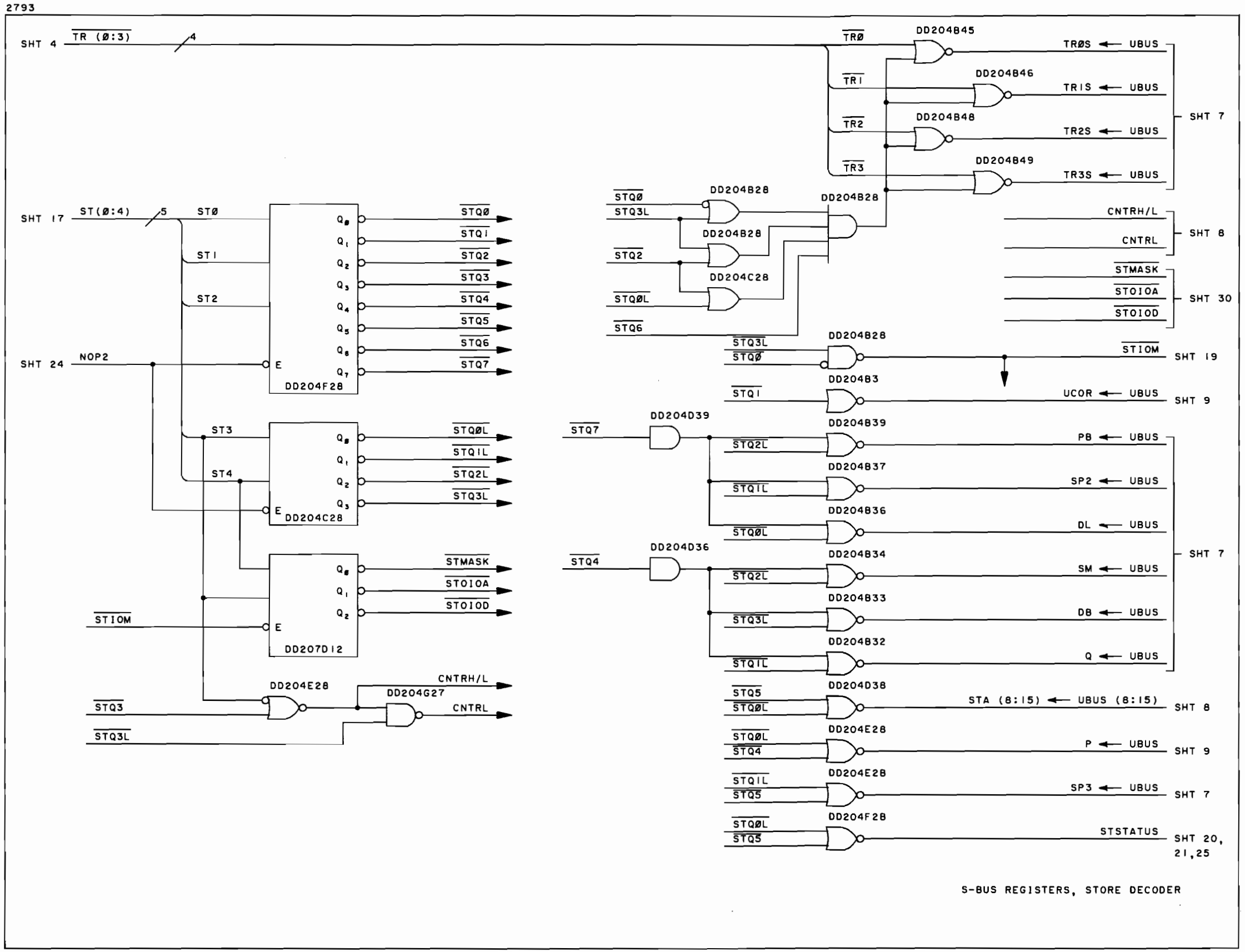


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 5 of 35)

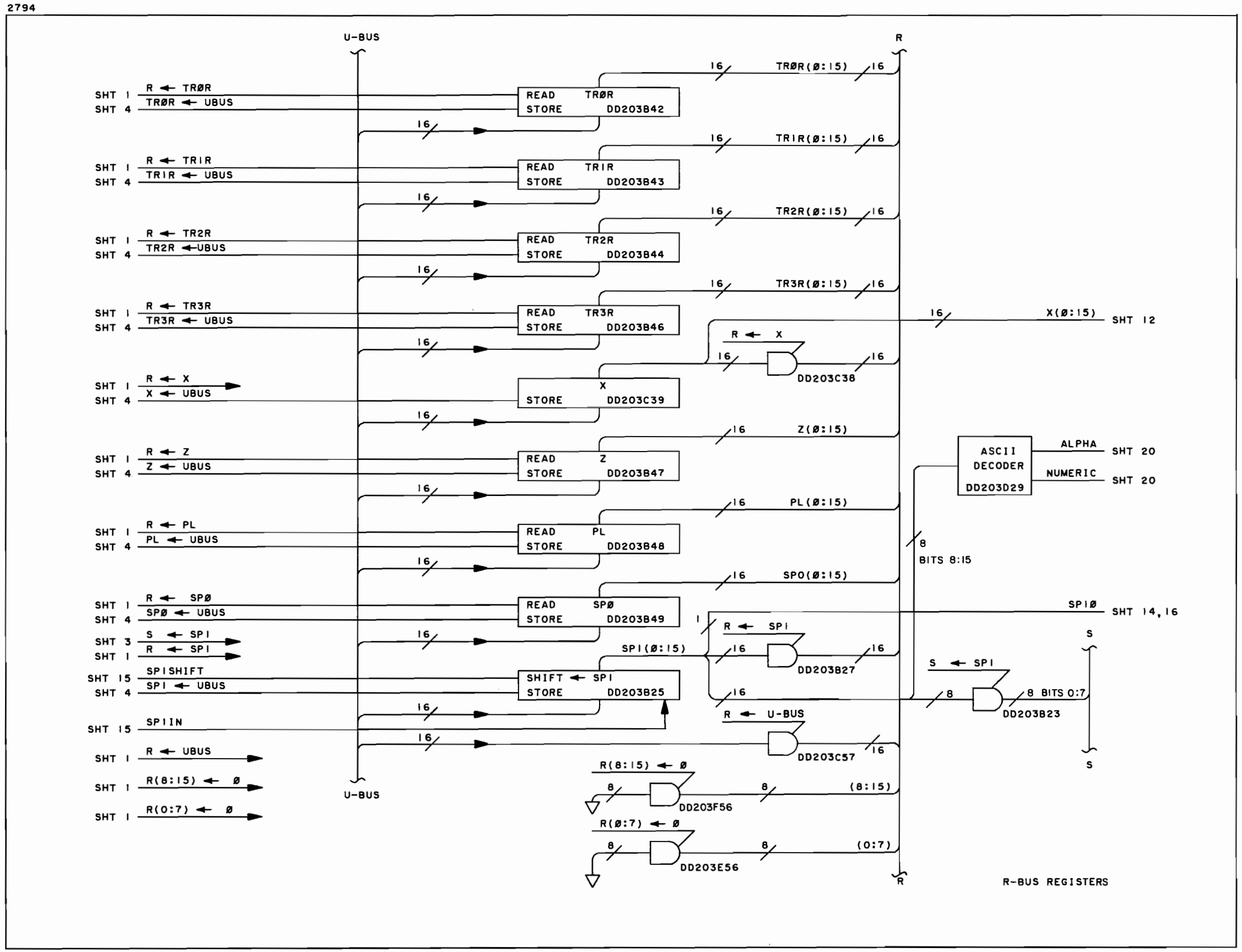


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 6 of 35)

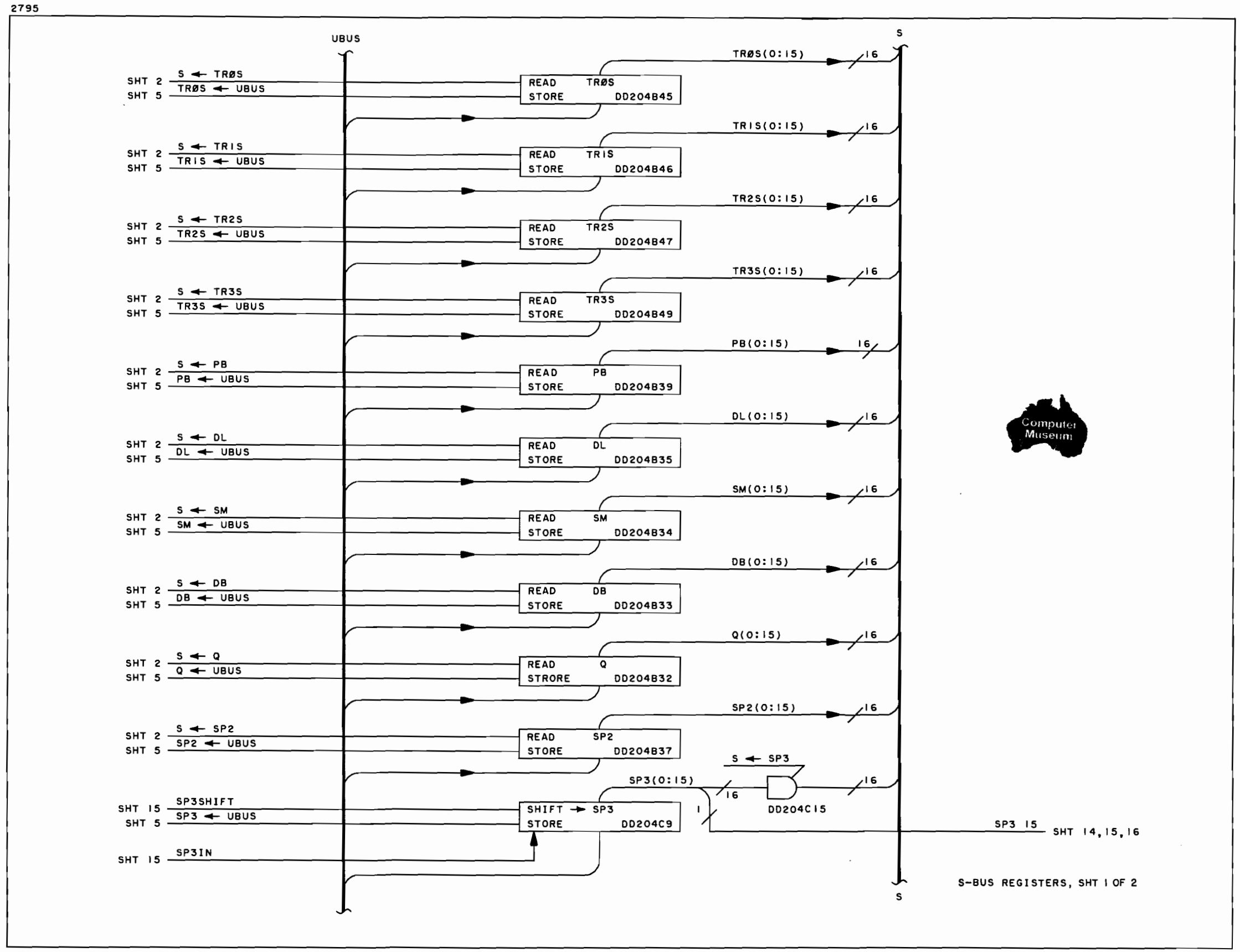


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 7 of 35)

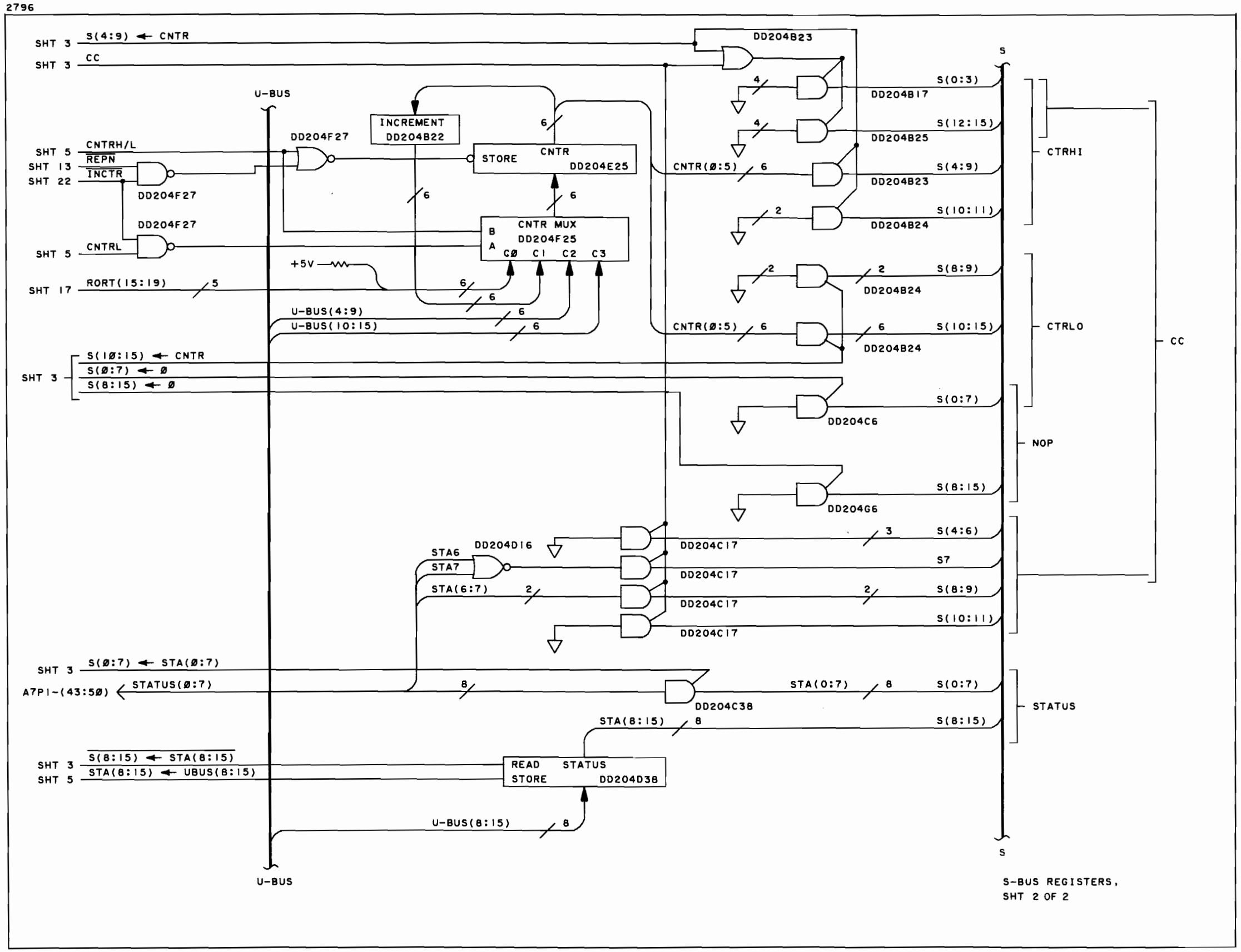


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 8 of 35)

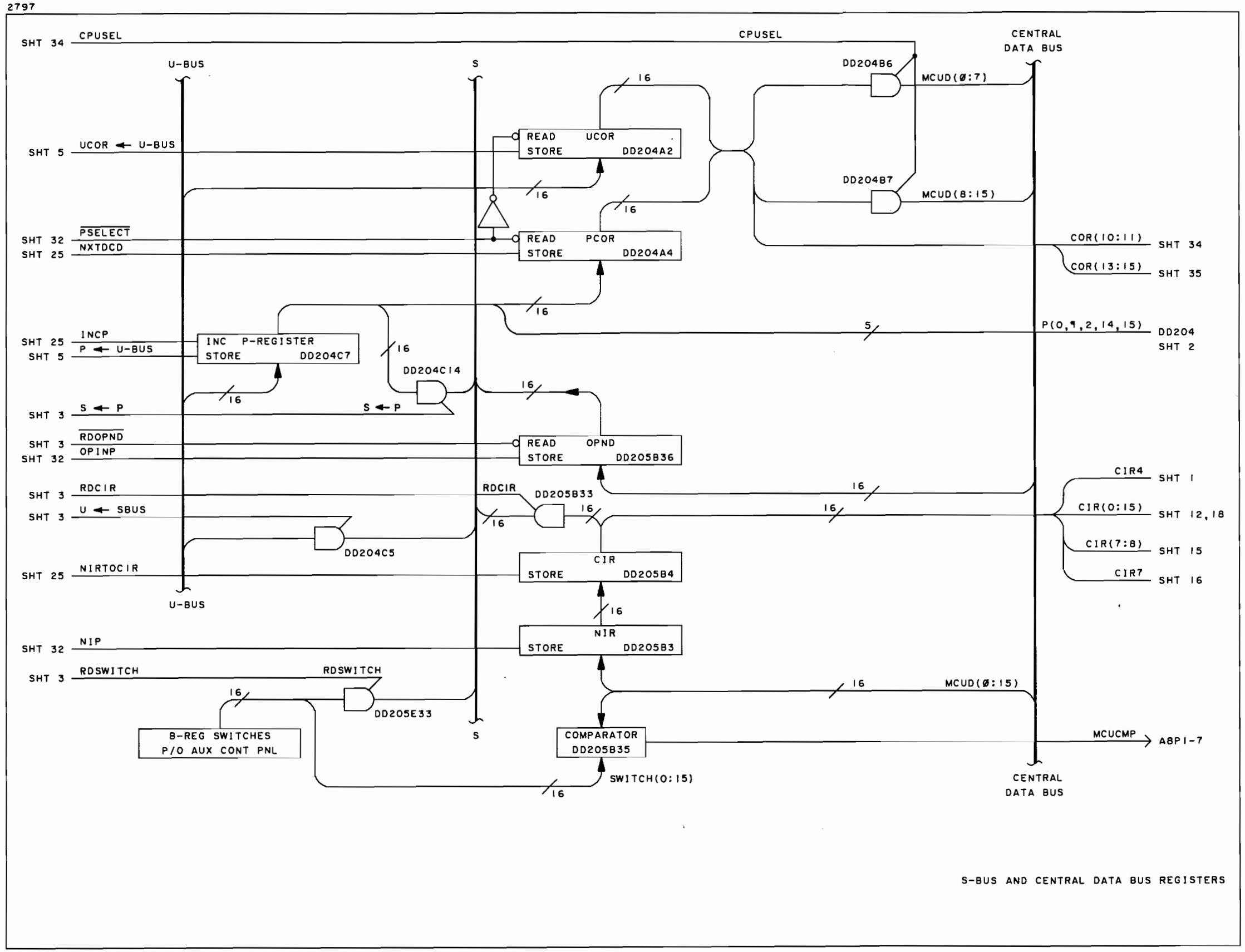


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 9 of 35)

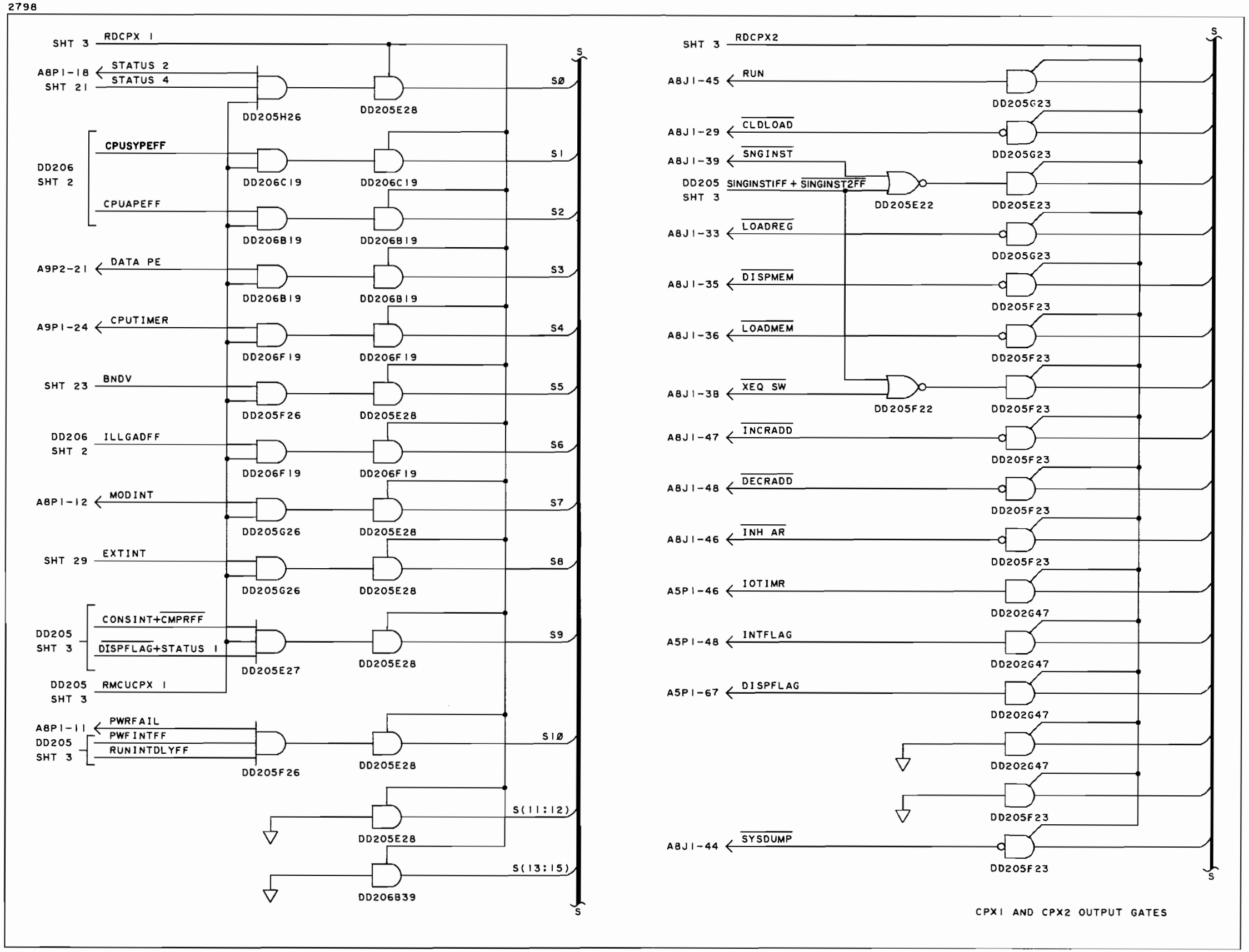


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 10 of 35)

2799

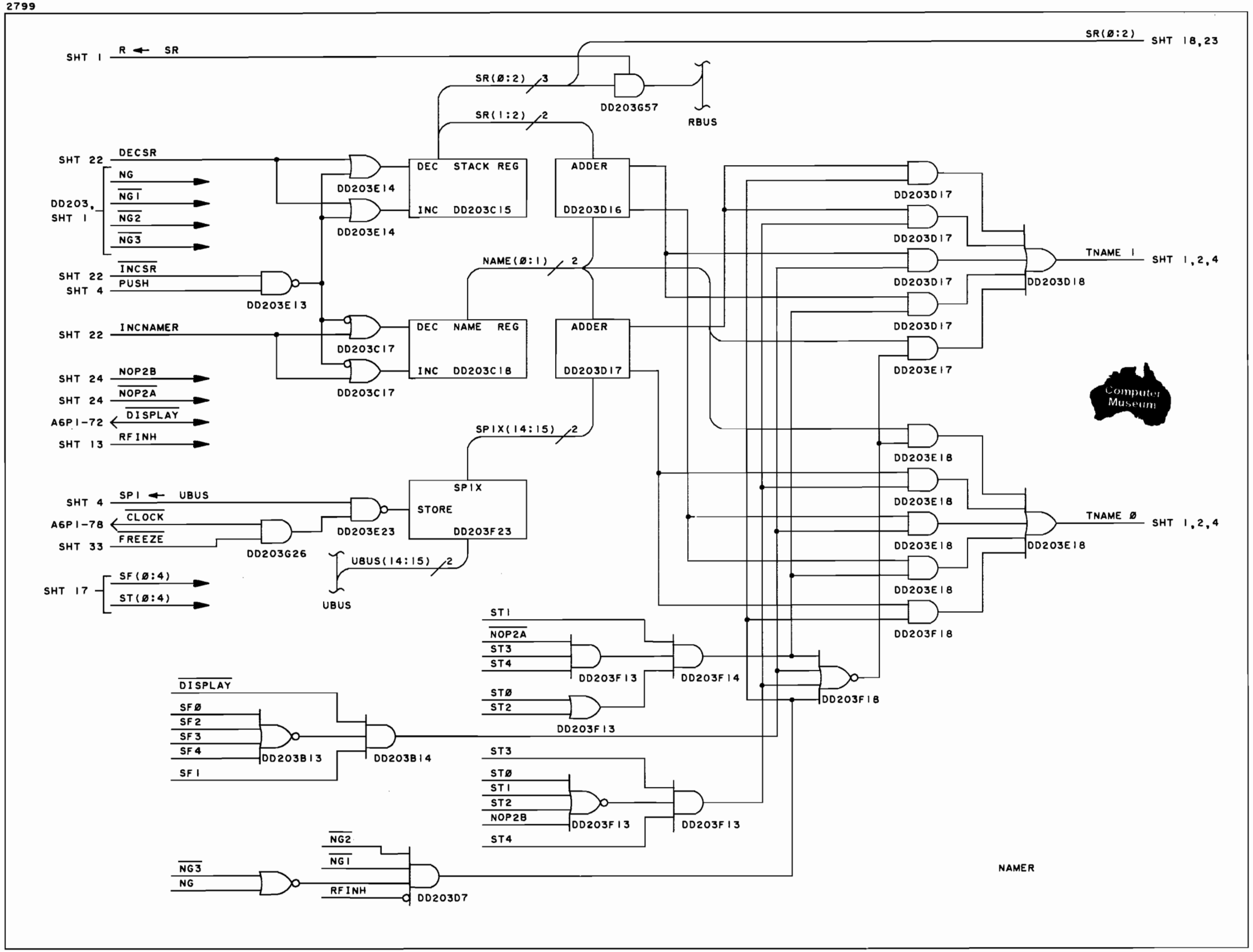
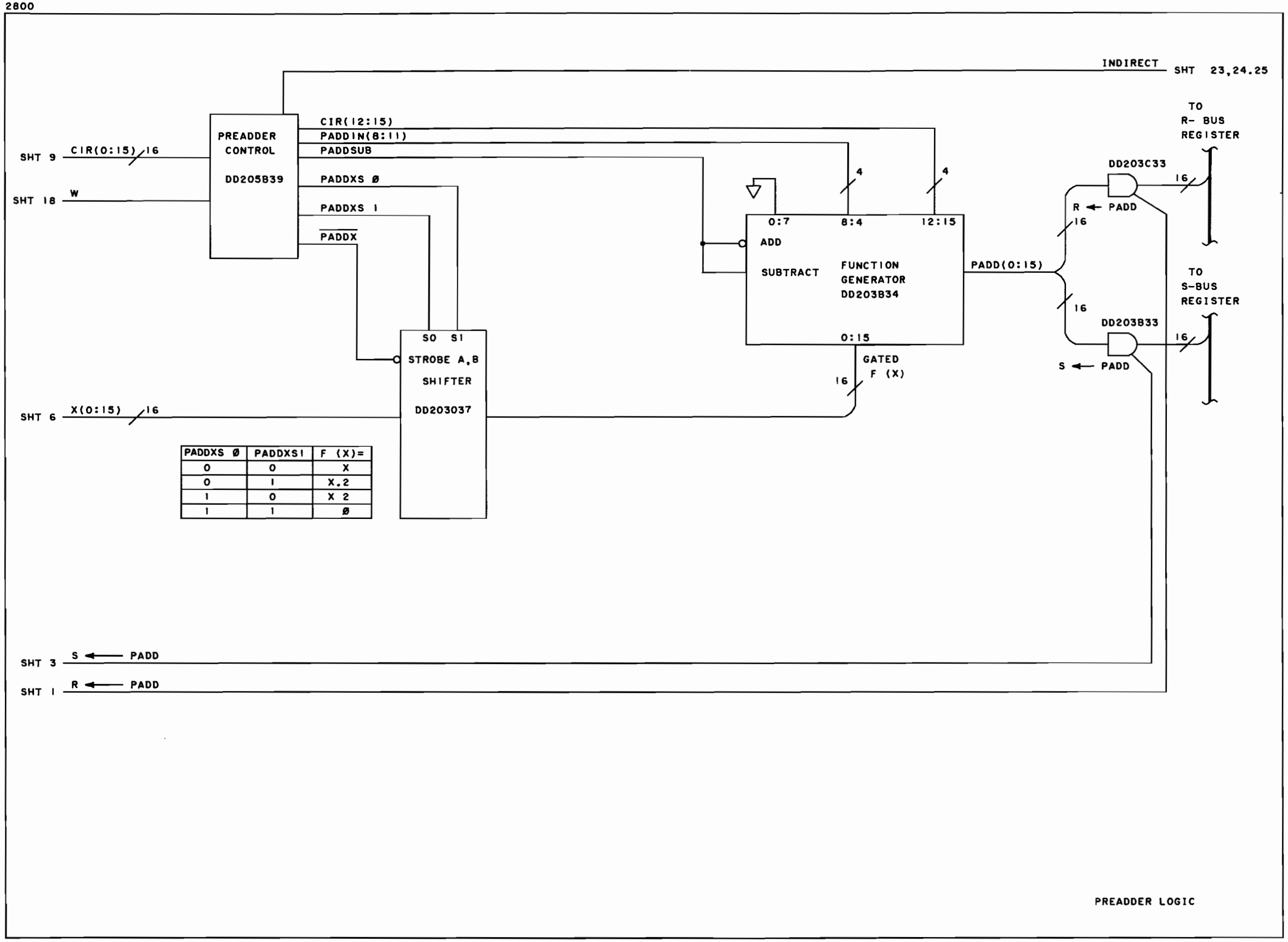


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 11 of 35)



PADDXS Ø	PADDXS1	F (X)=
0	0	X
0	1	X.2
1	0	X 2
1	1	Ø

Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 12 of 35)

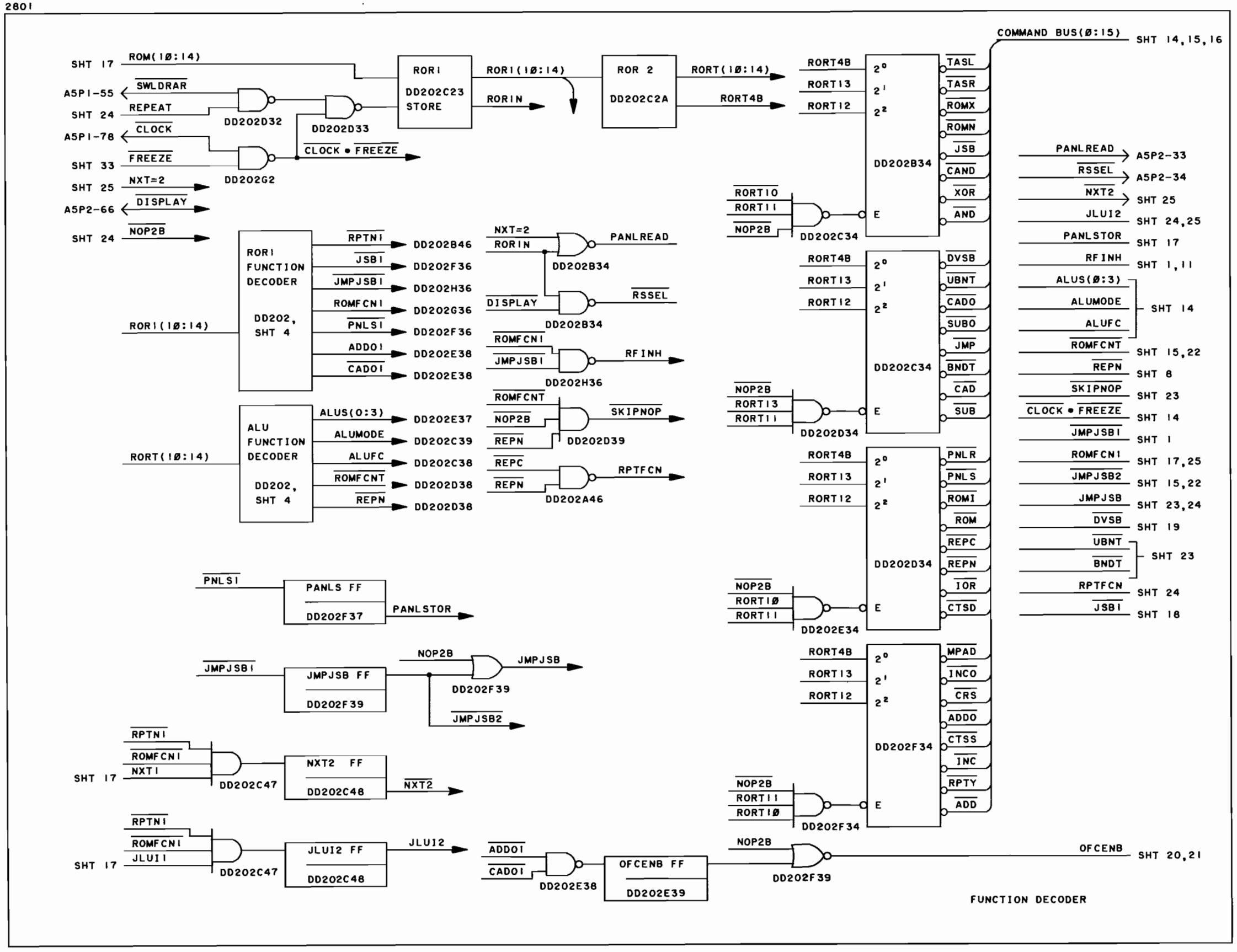
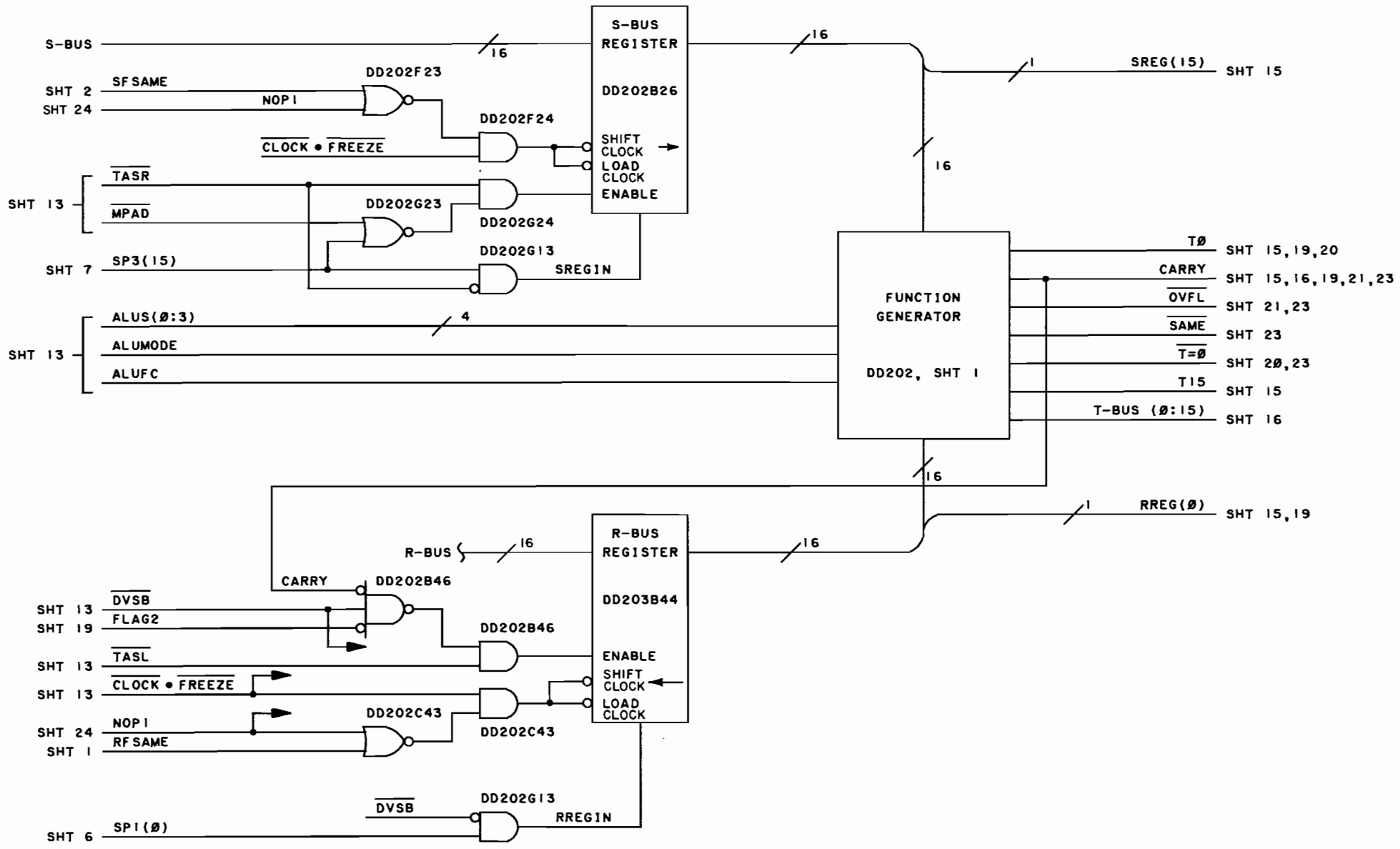


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 13 of 35)

2802



FUNCTION GENERATOR

Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 14 of 35)

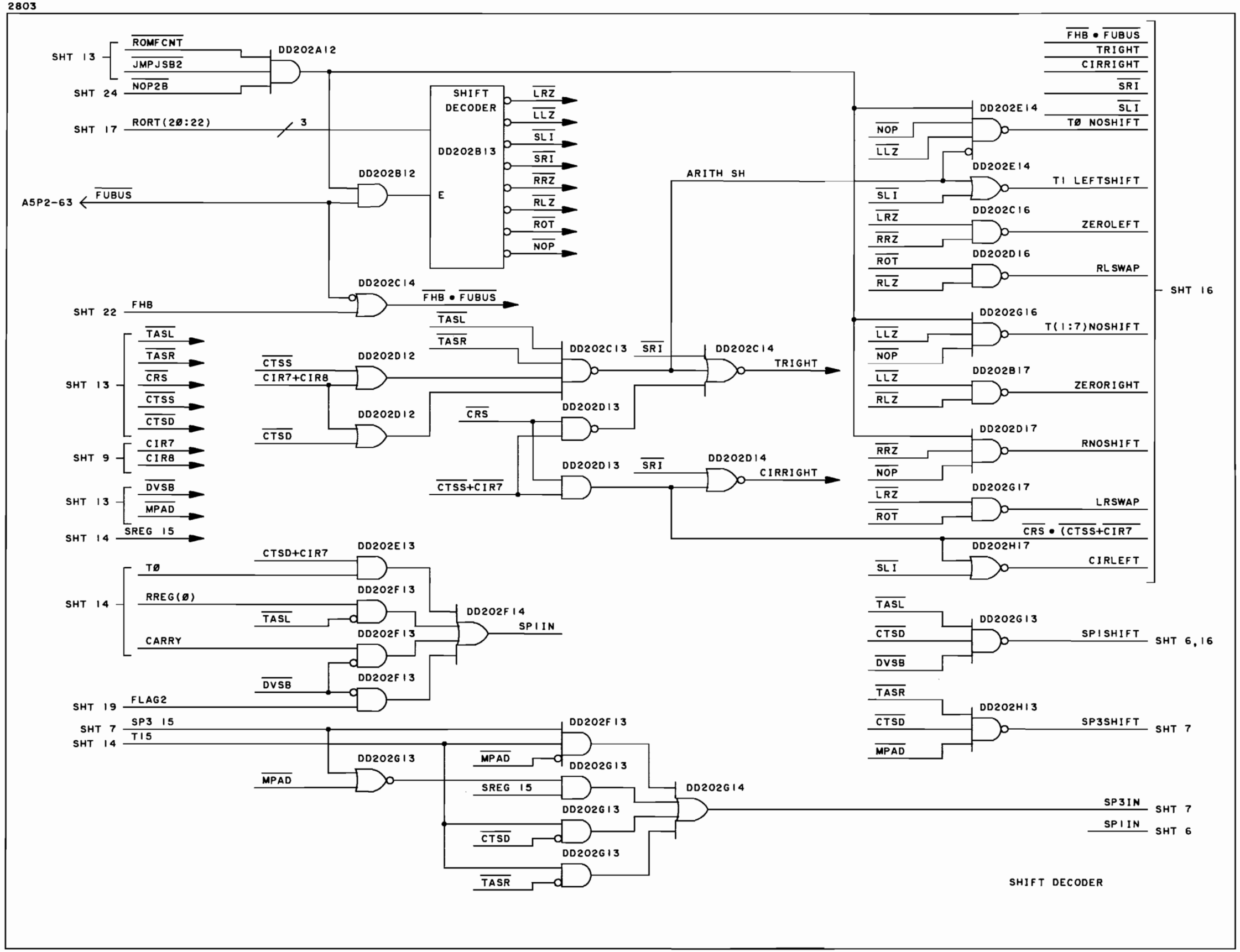


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 15 of 35)

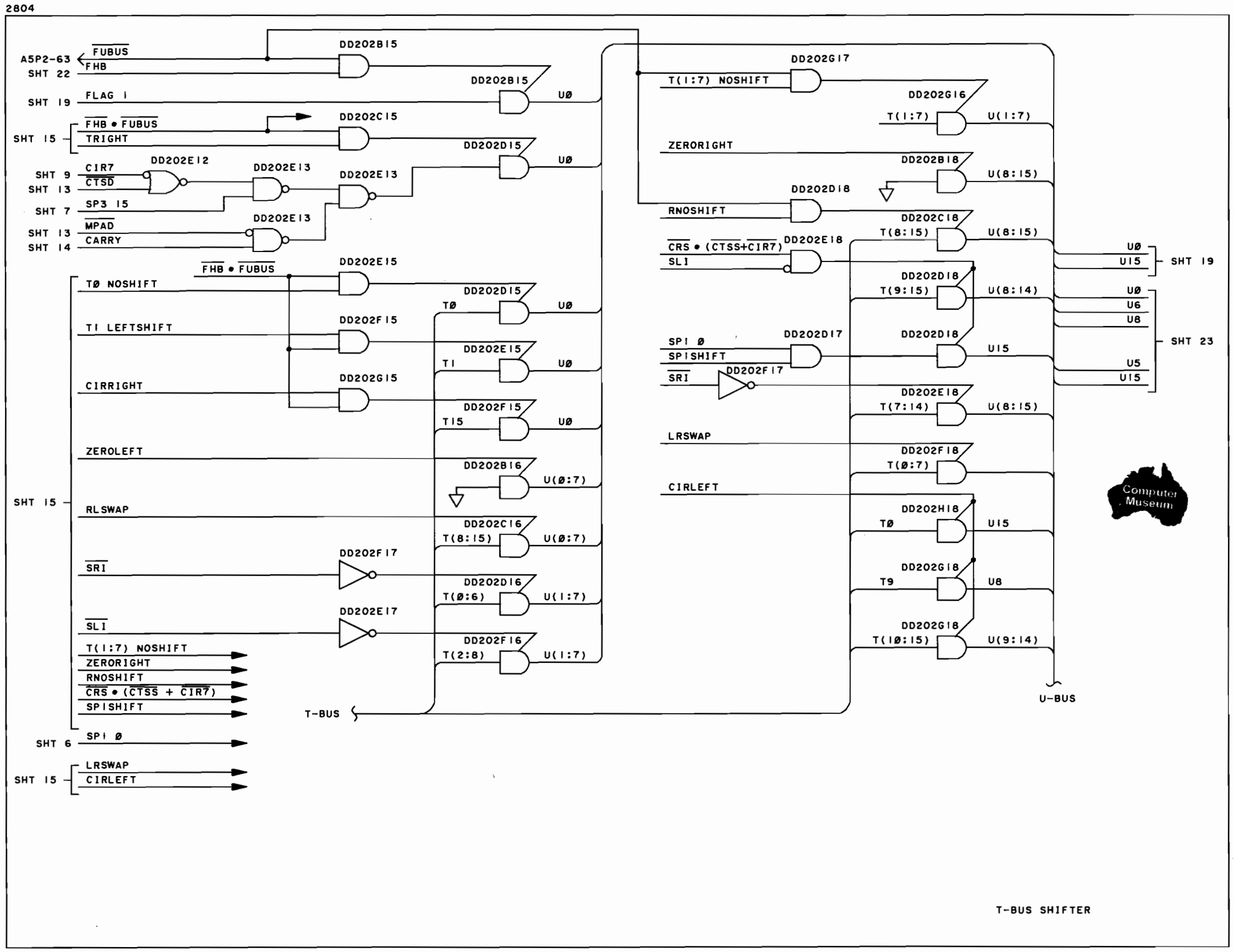


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 16 of 35)

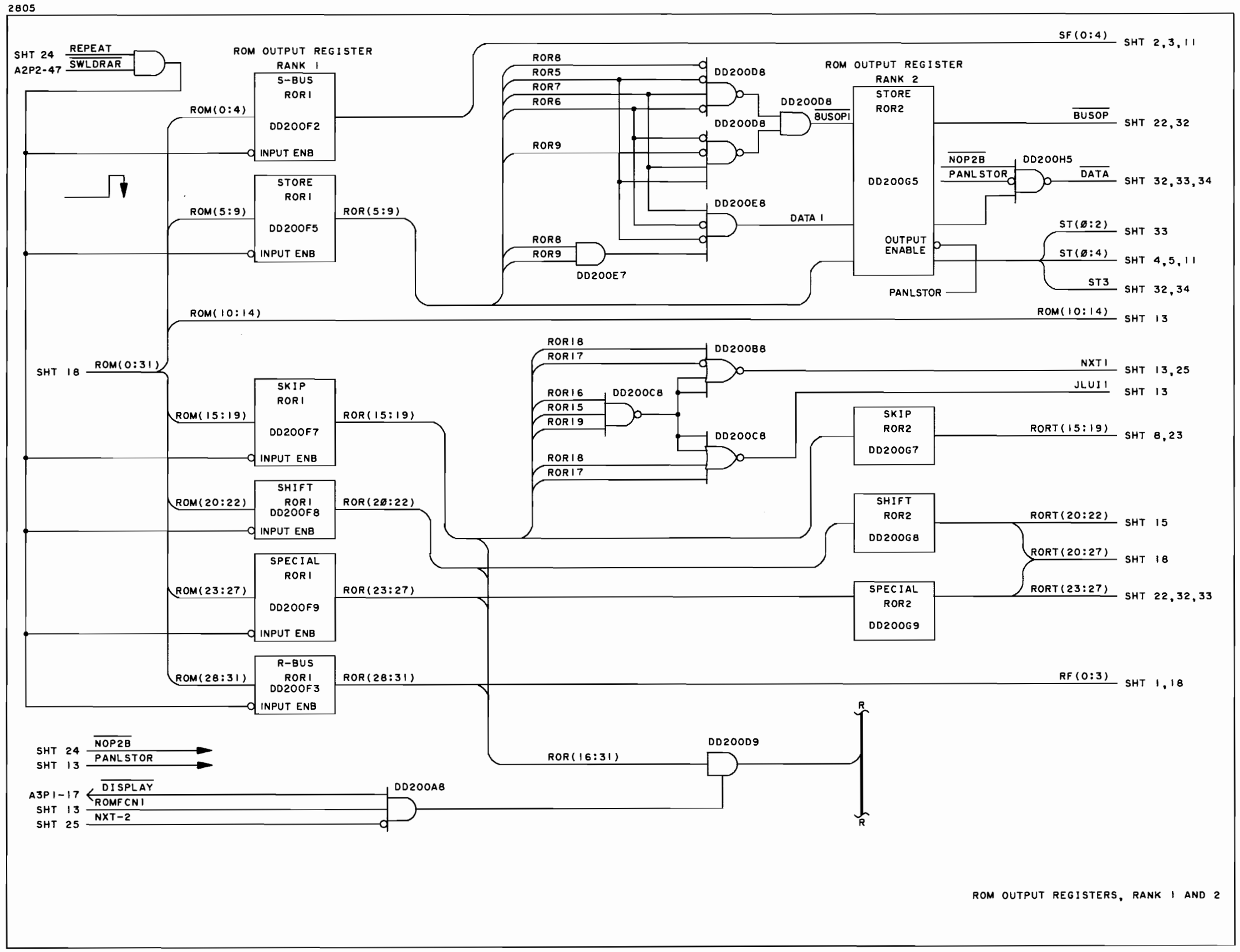


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 17 of 35)

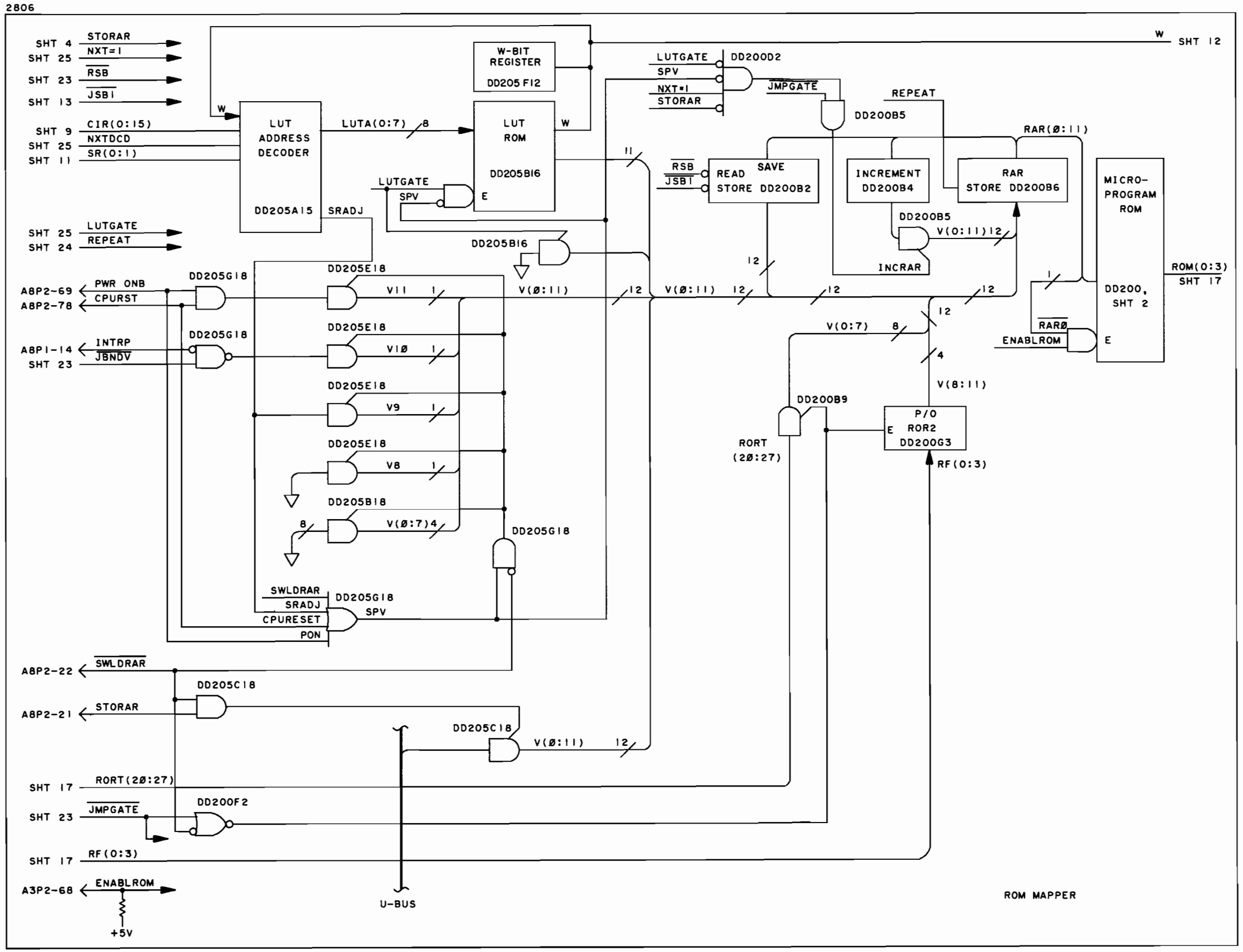
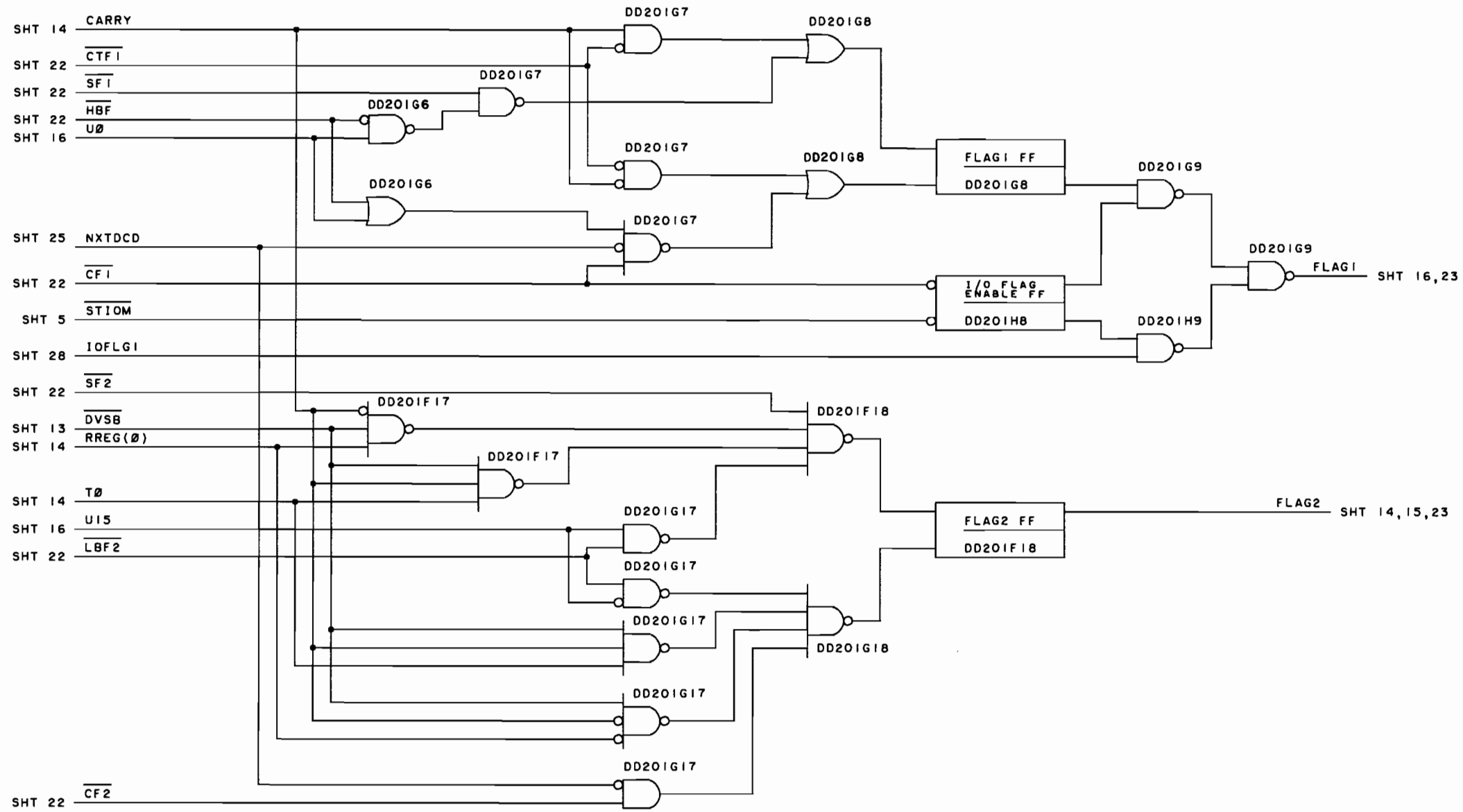


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 18 of 35)

2807



FLAG1 AND FLAG2 LOGIC

Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 19 of 35)

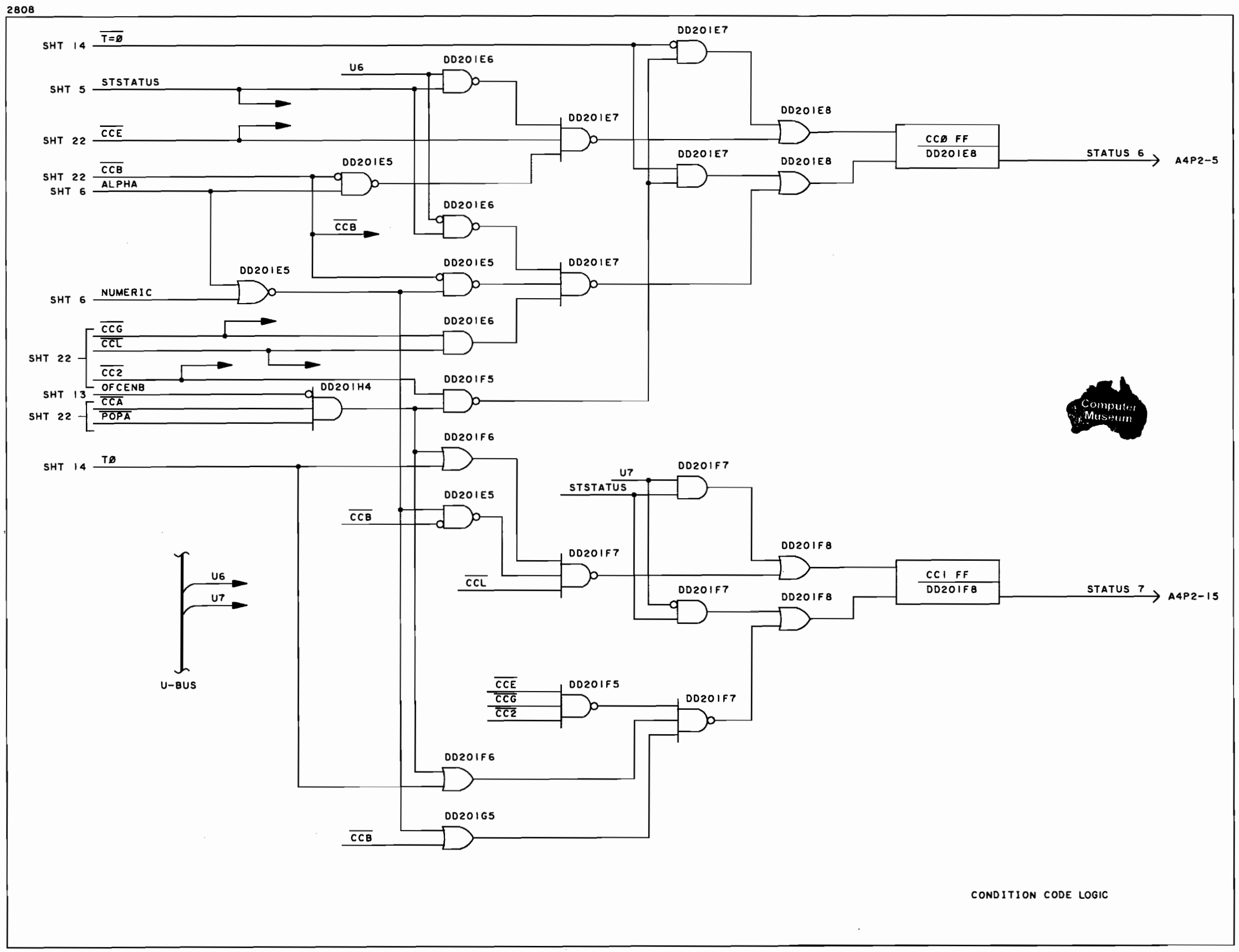


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 20 of 35)

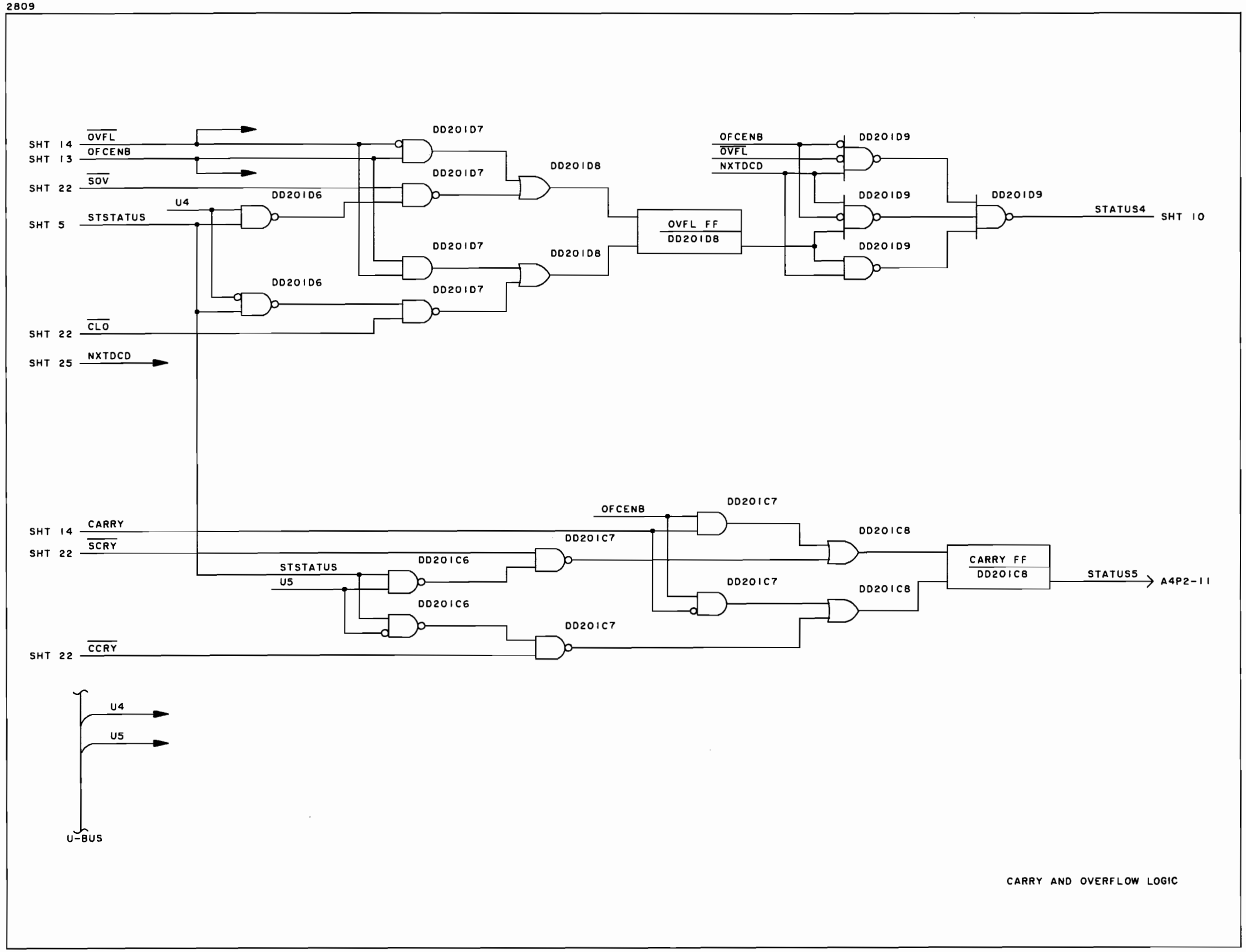


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 21 of 35)

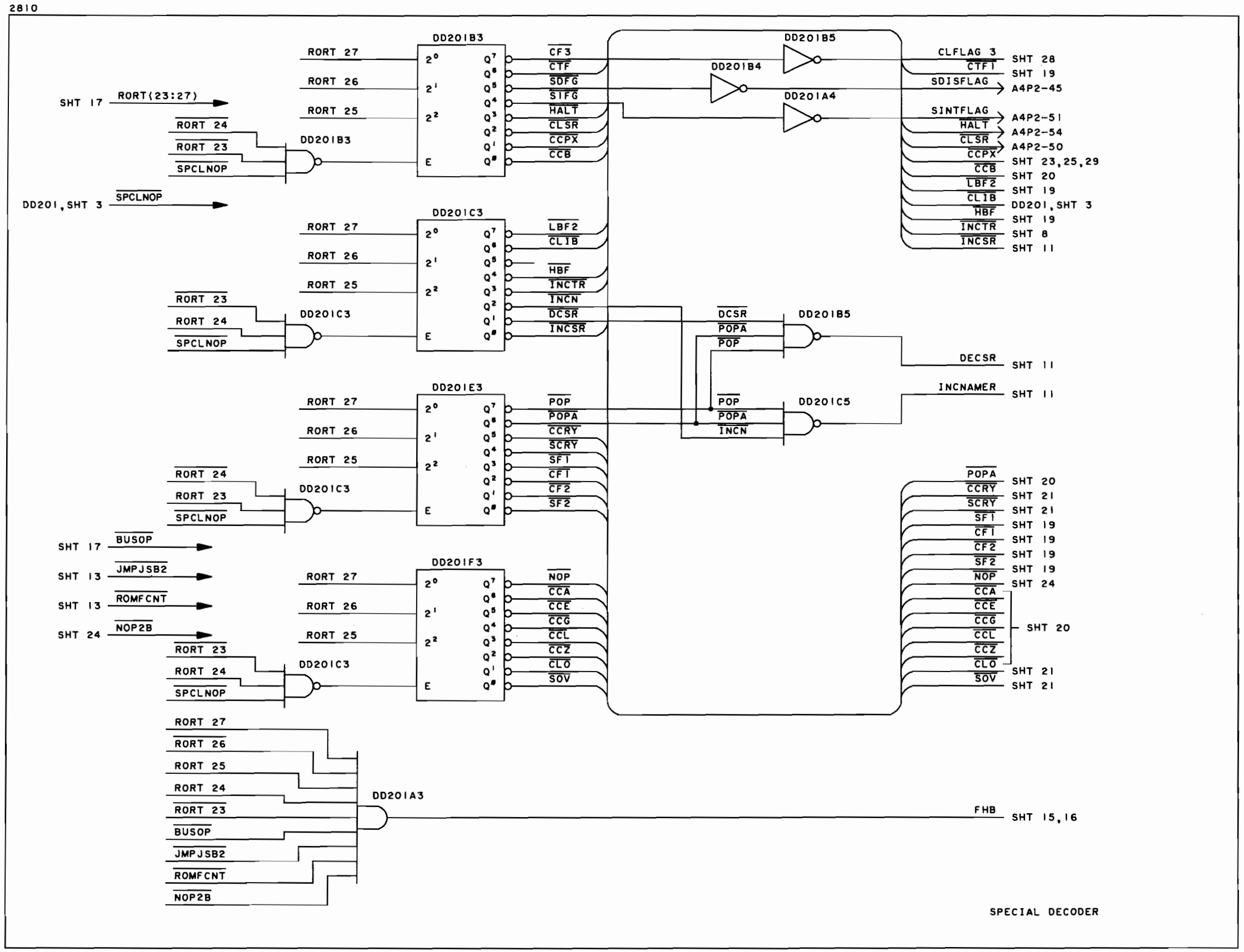


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 22 of 35)

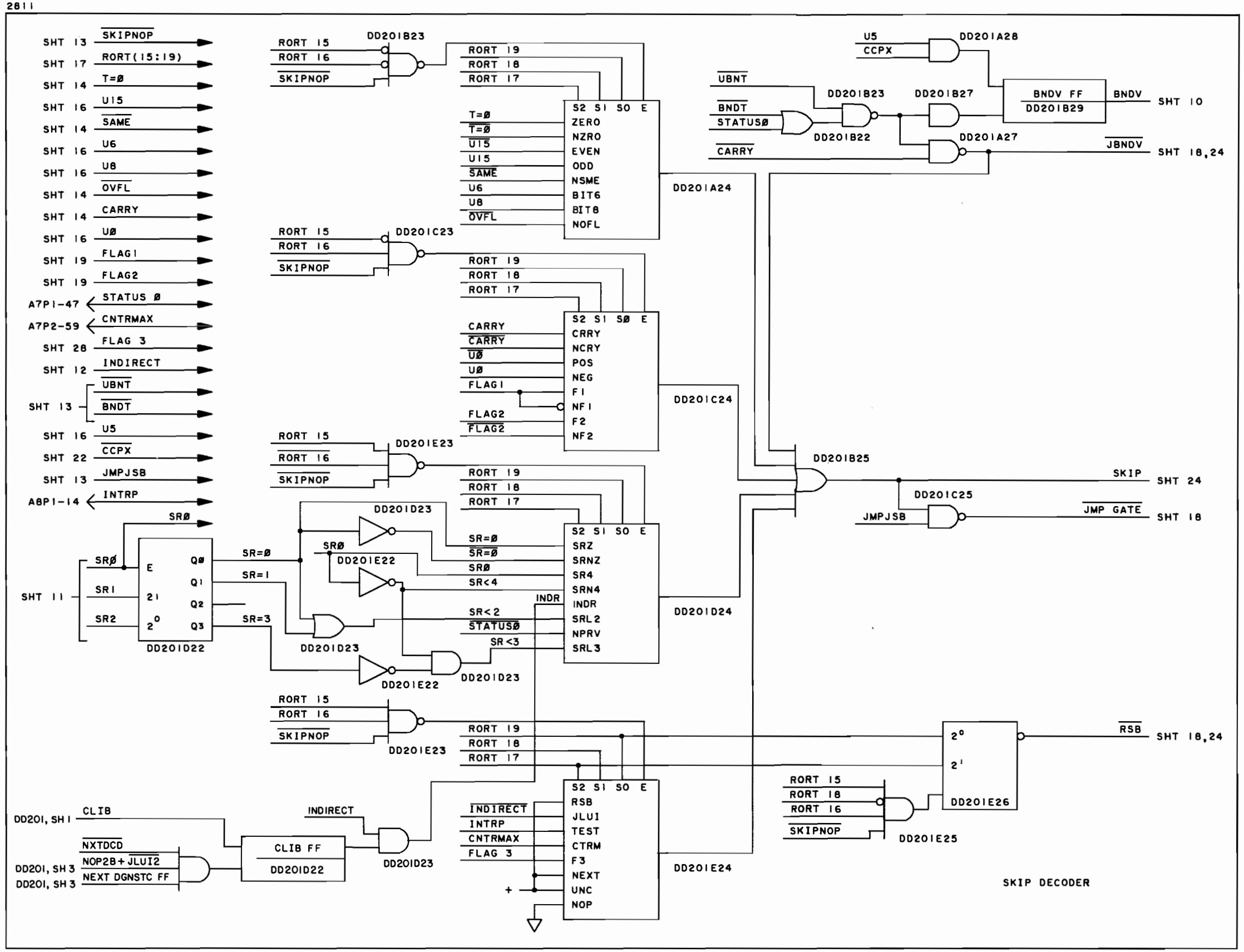


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 23 of 35)

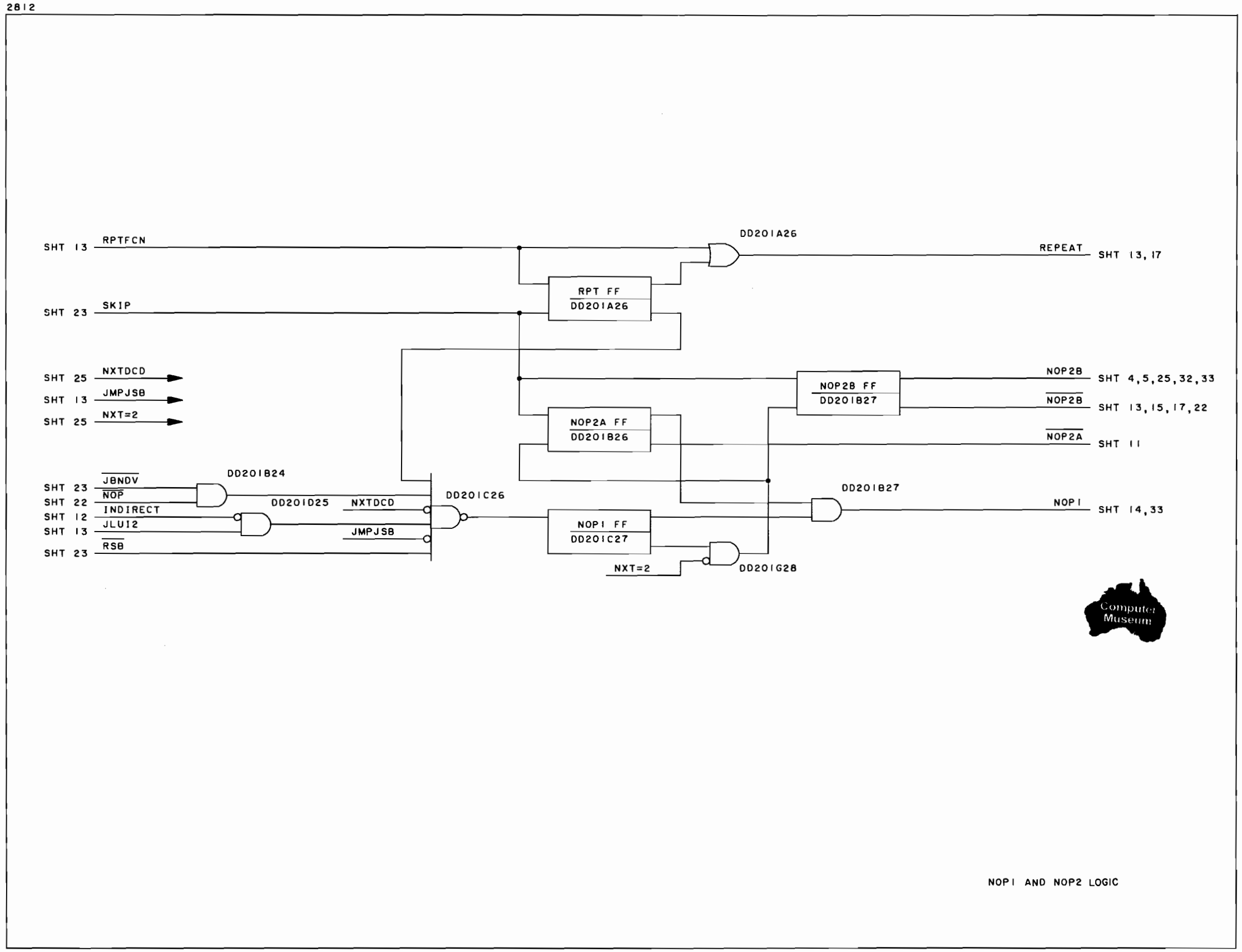


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 24 of 35)

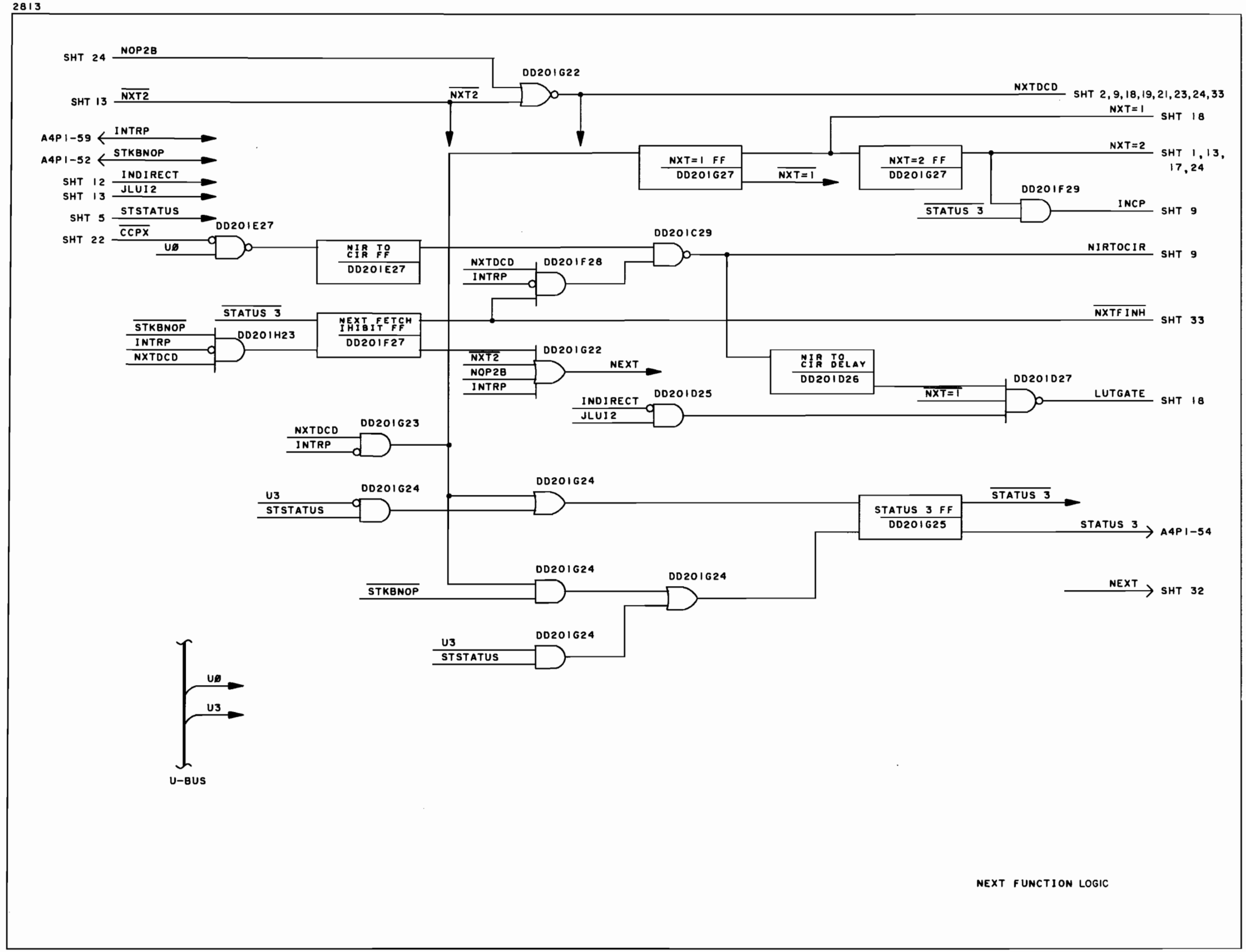


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 25 of 35)

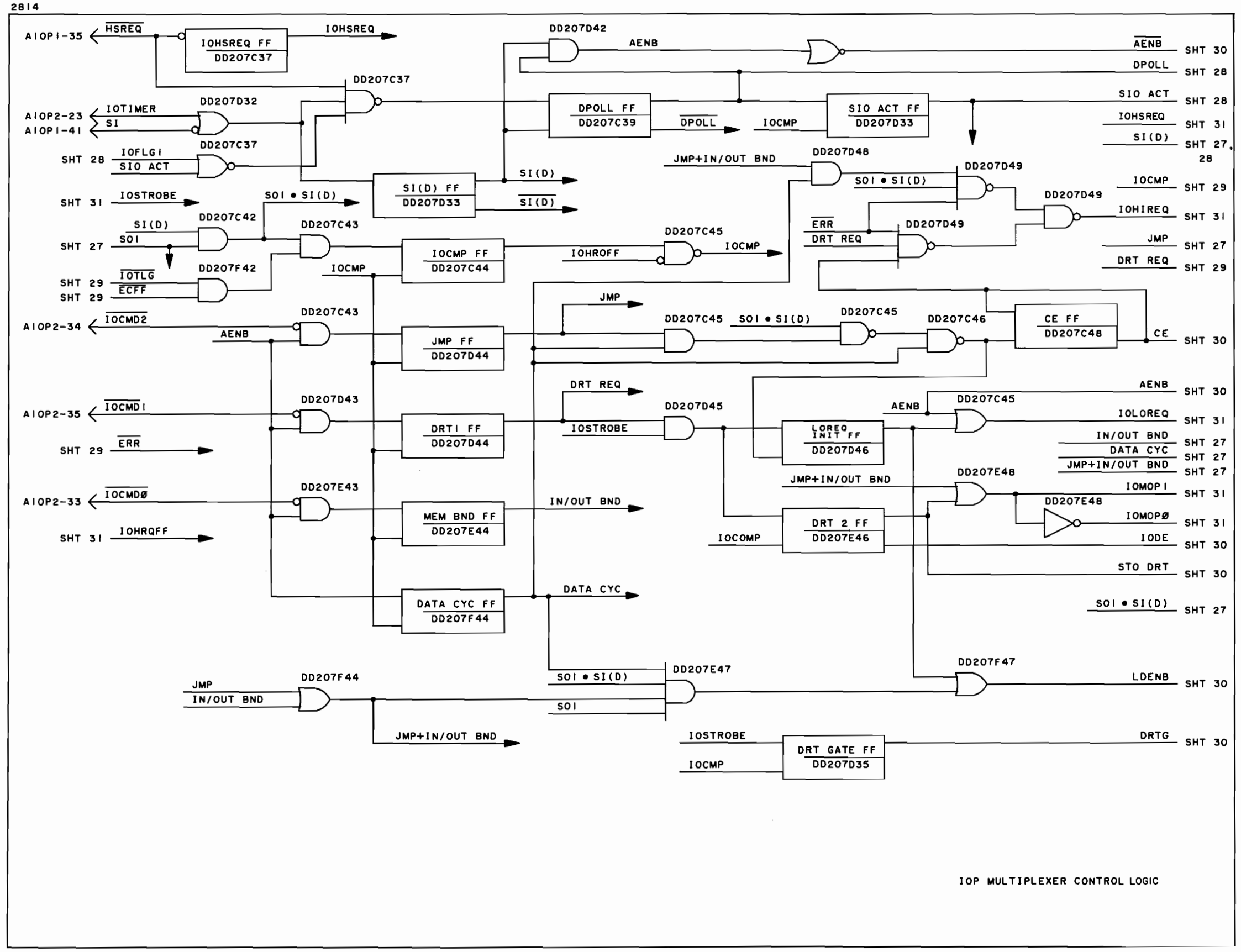


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 26 of 35)

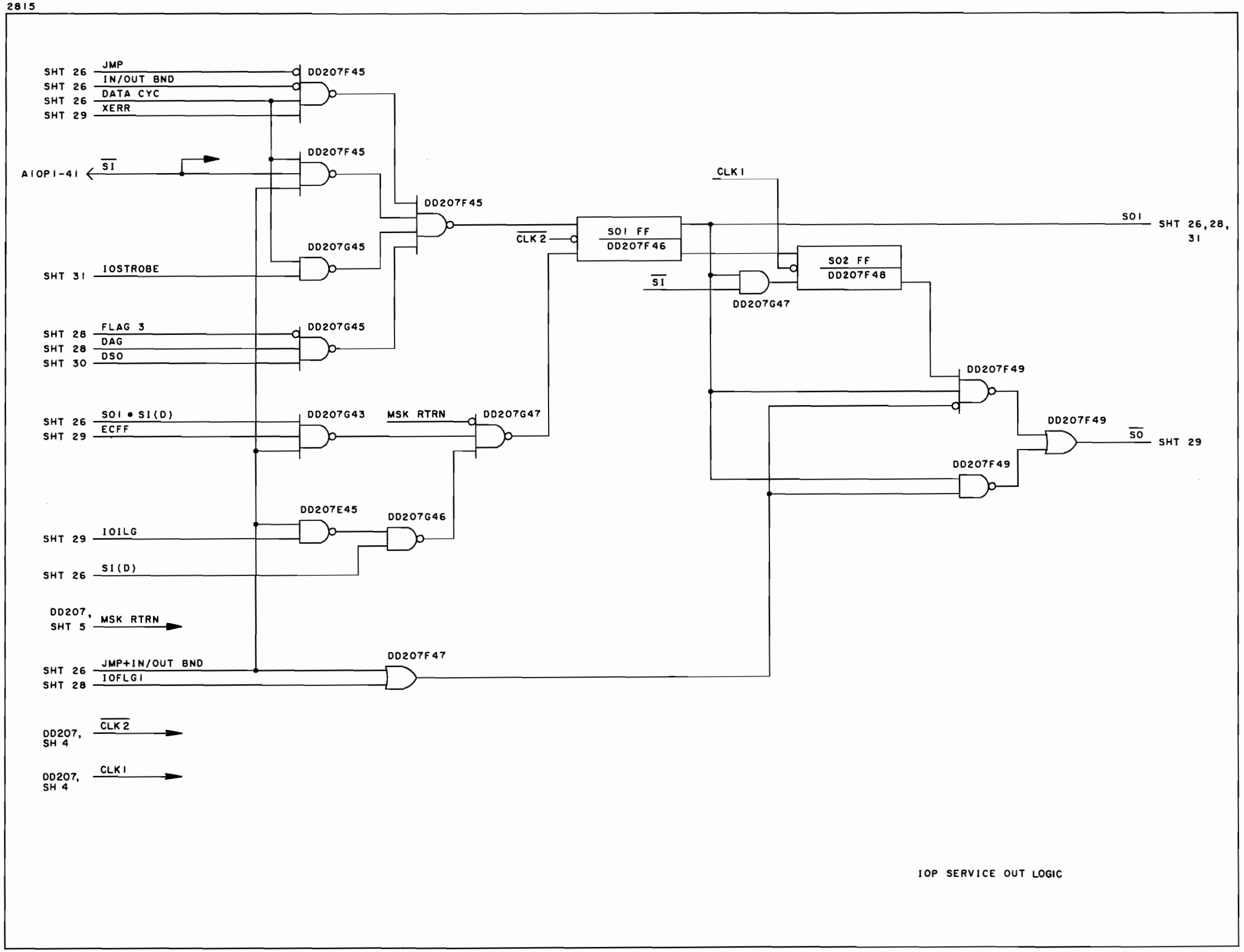


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 27 of 35)

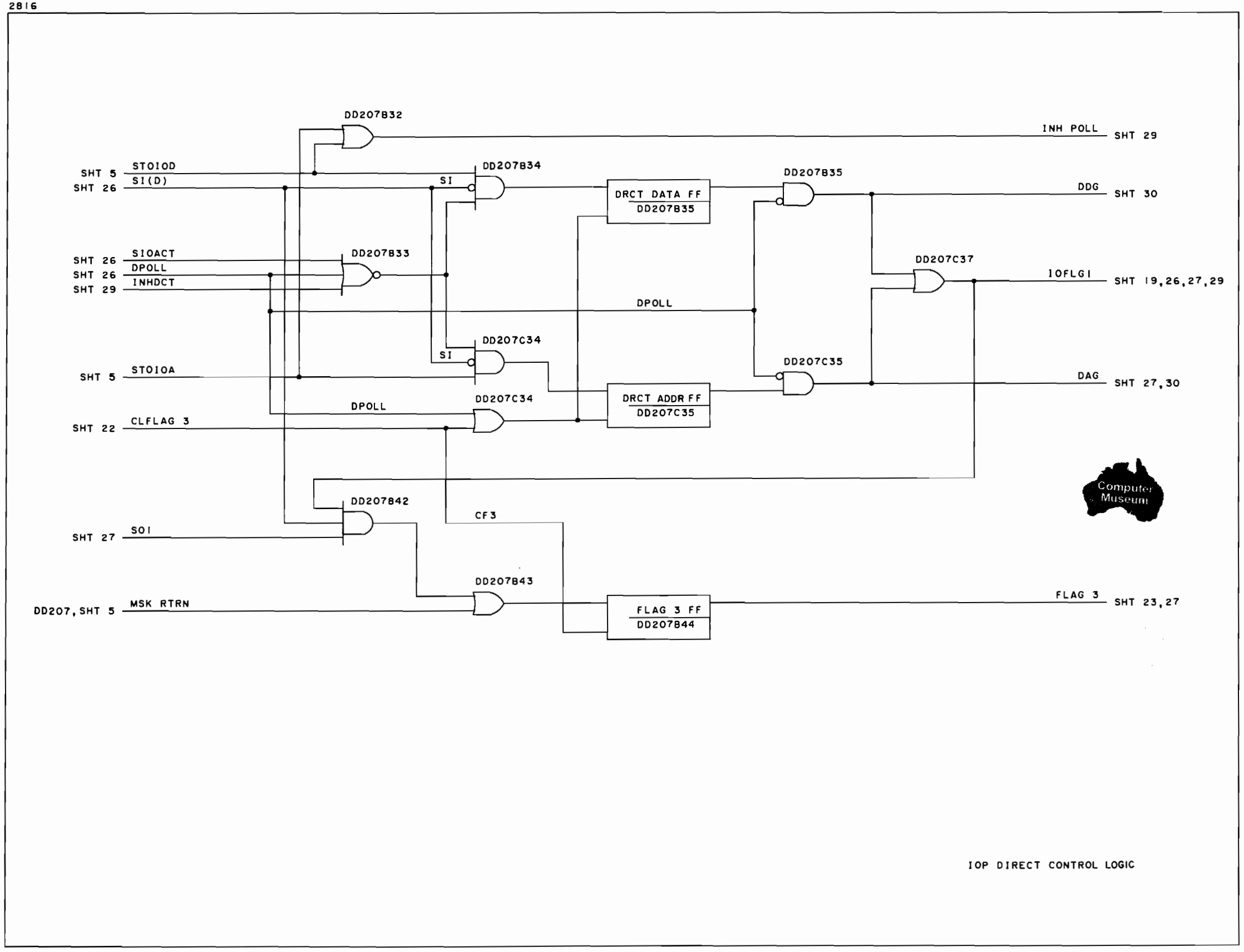


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 28 of 35)

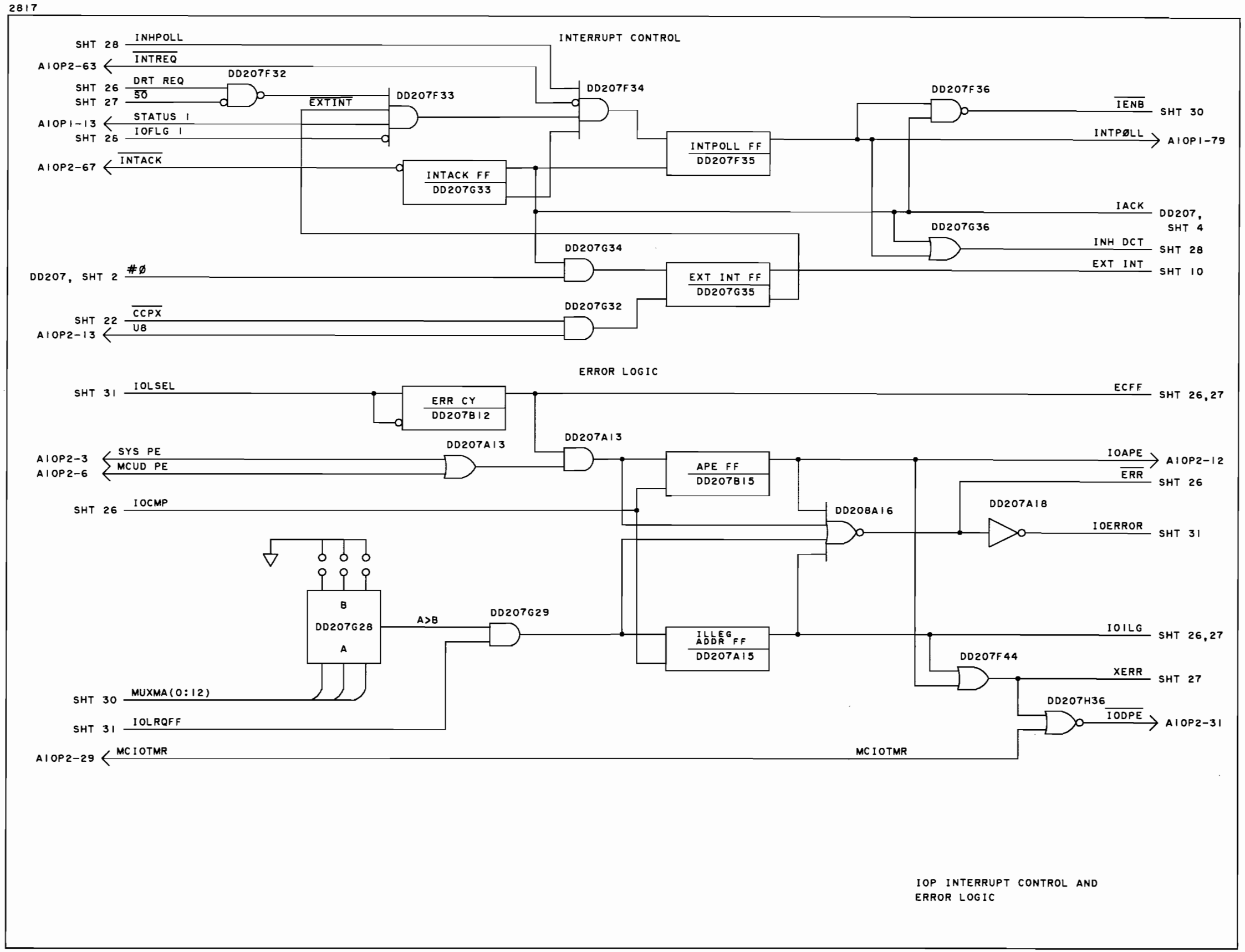


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 29 of 35)

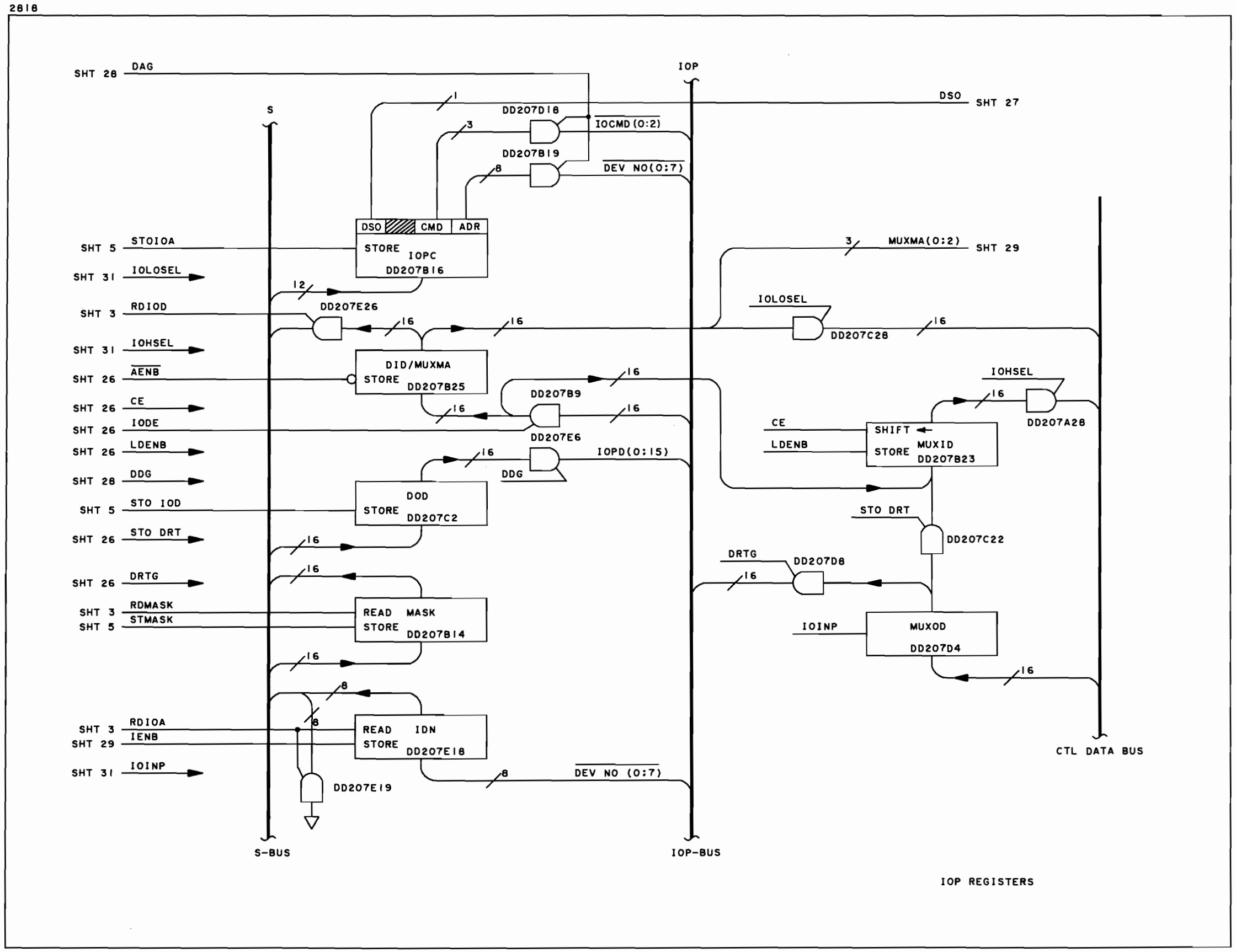


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 30 of 35)

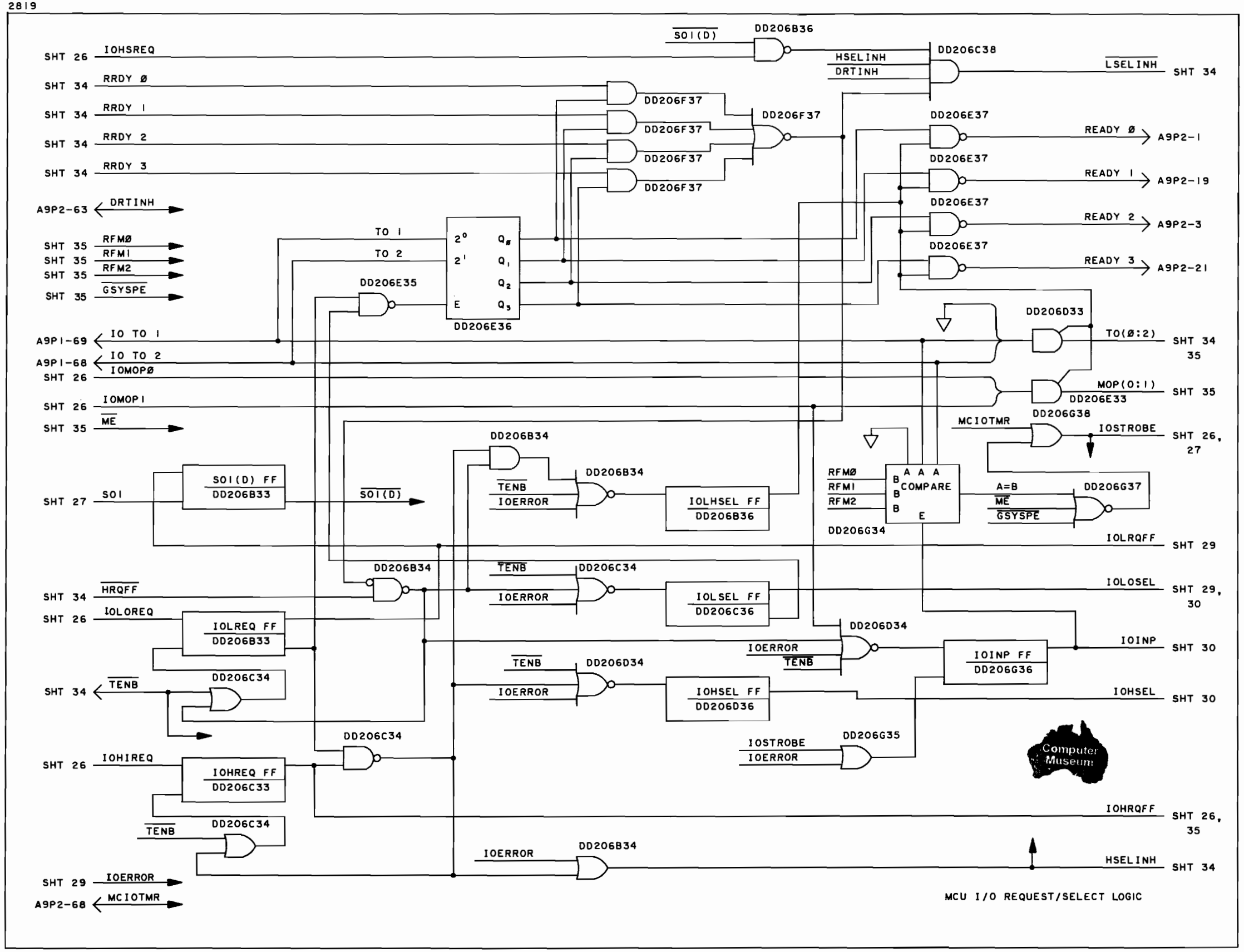


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 31 of 35)

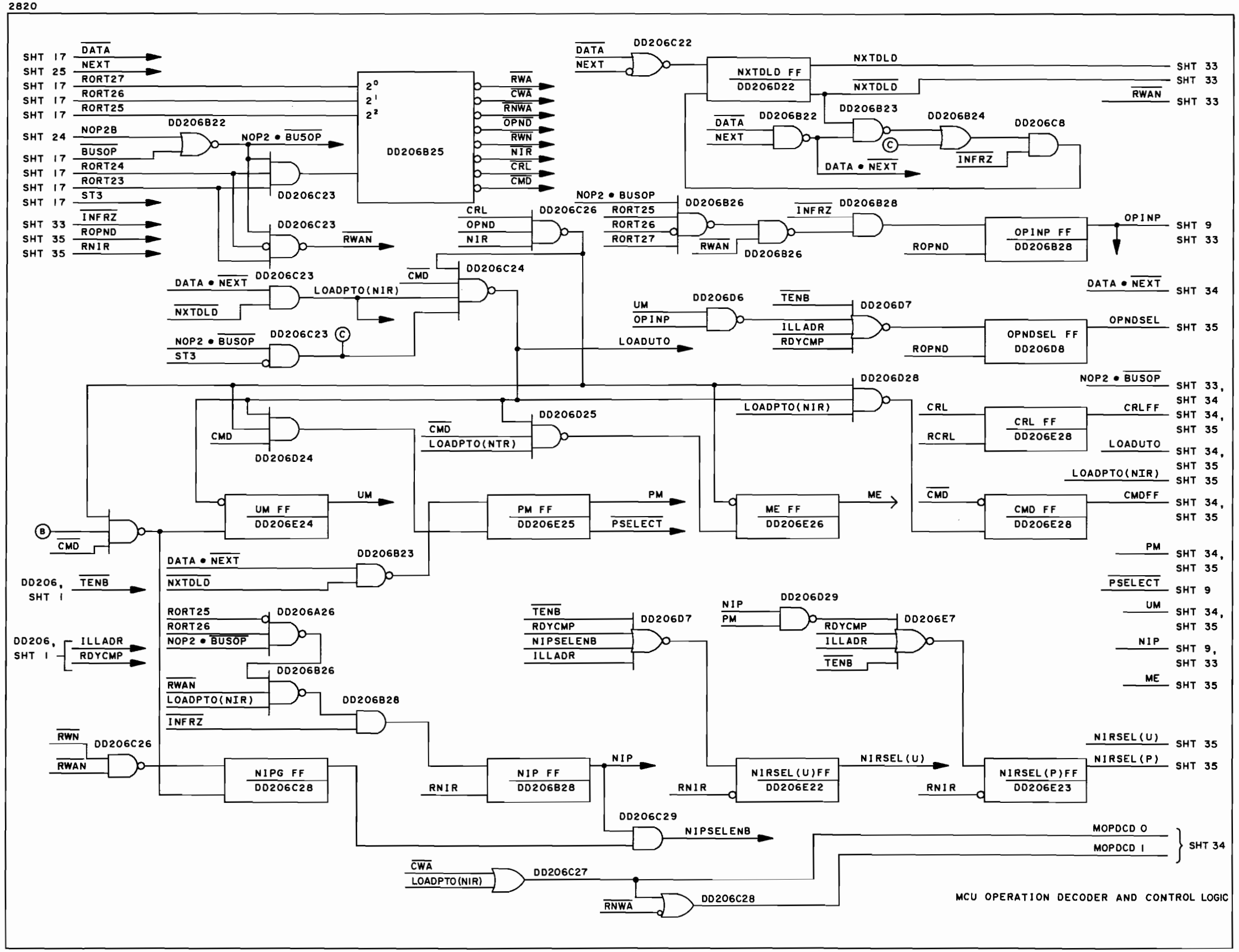


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 32 of 35)

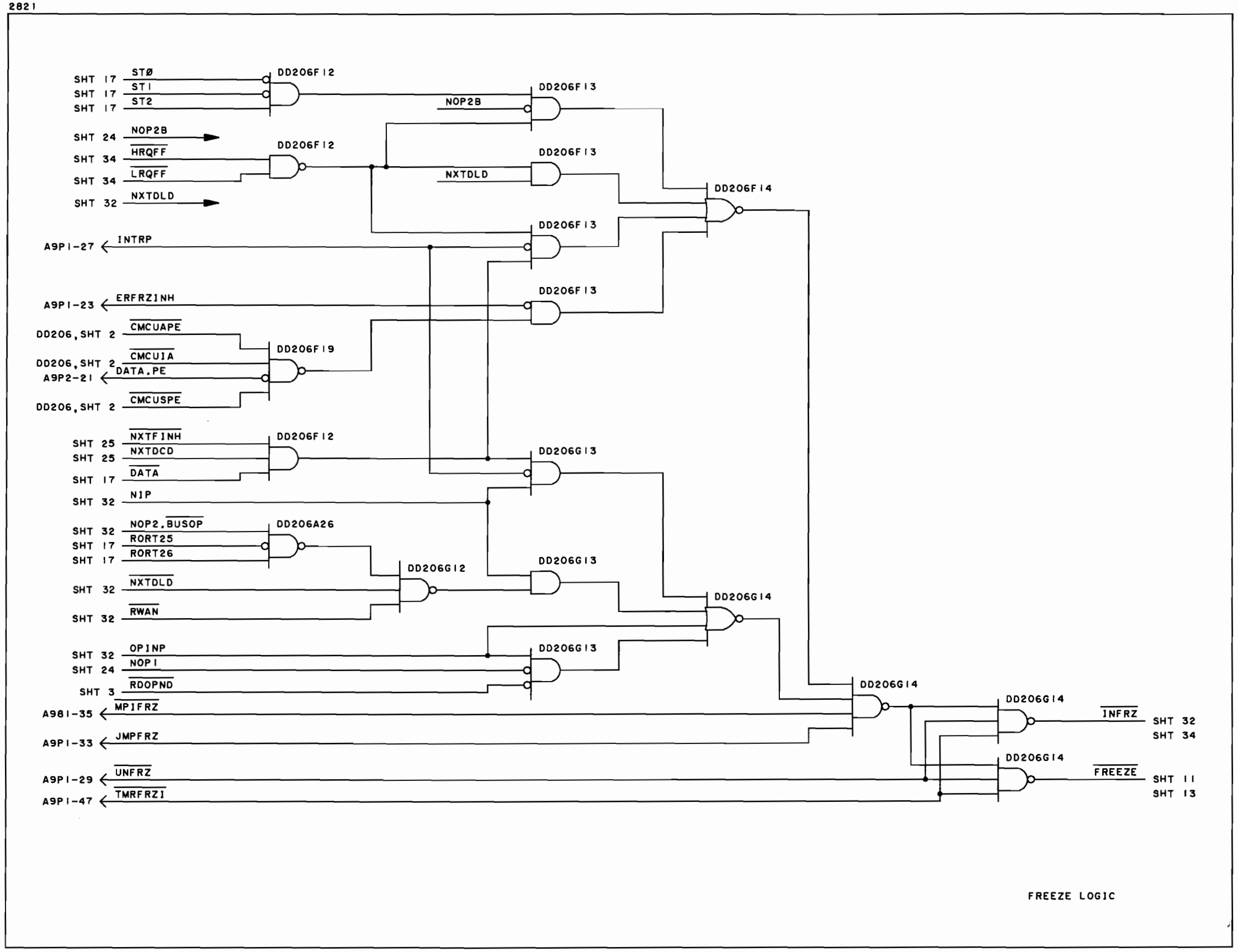


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 33 of 35)

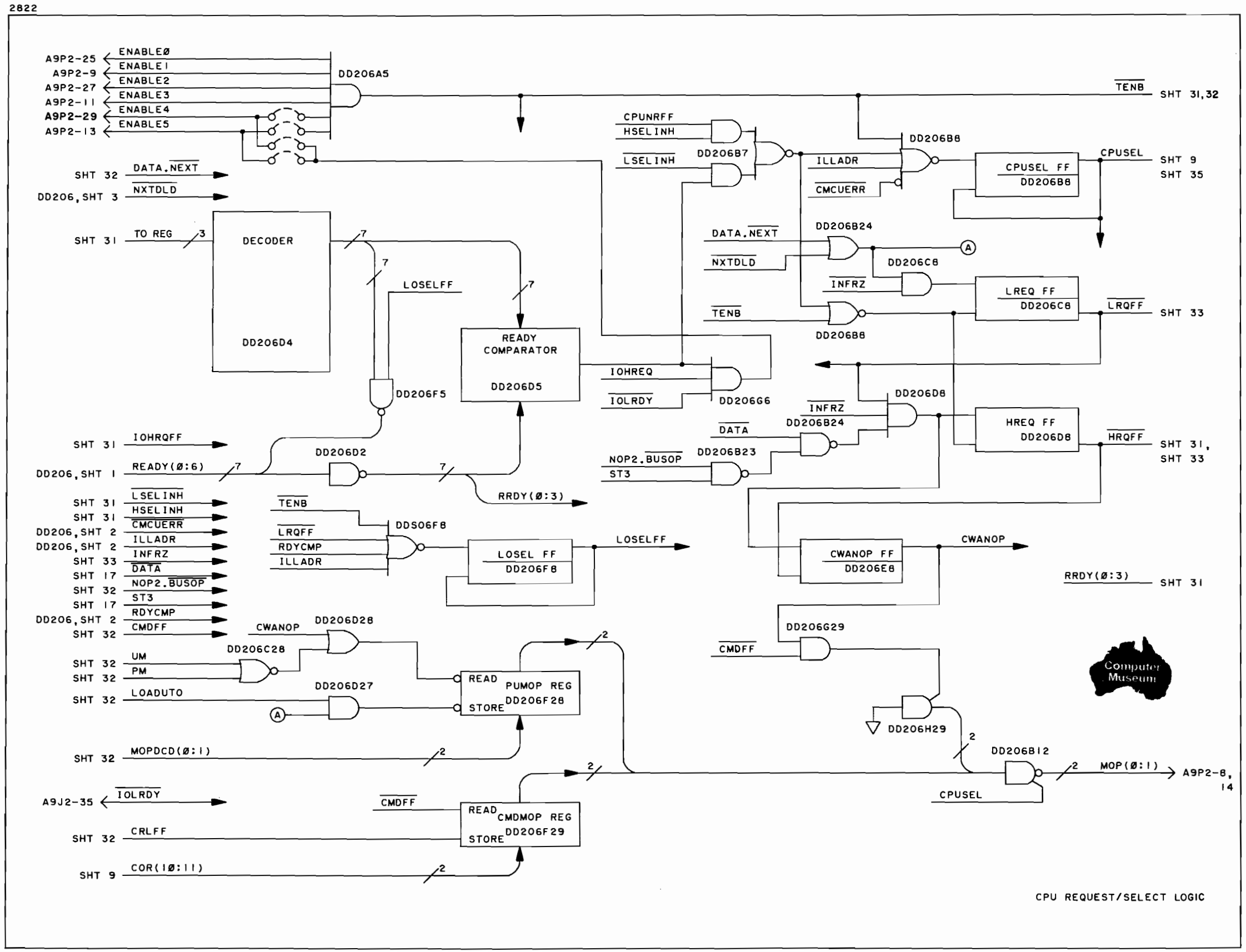


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 34 of 35)

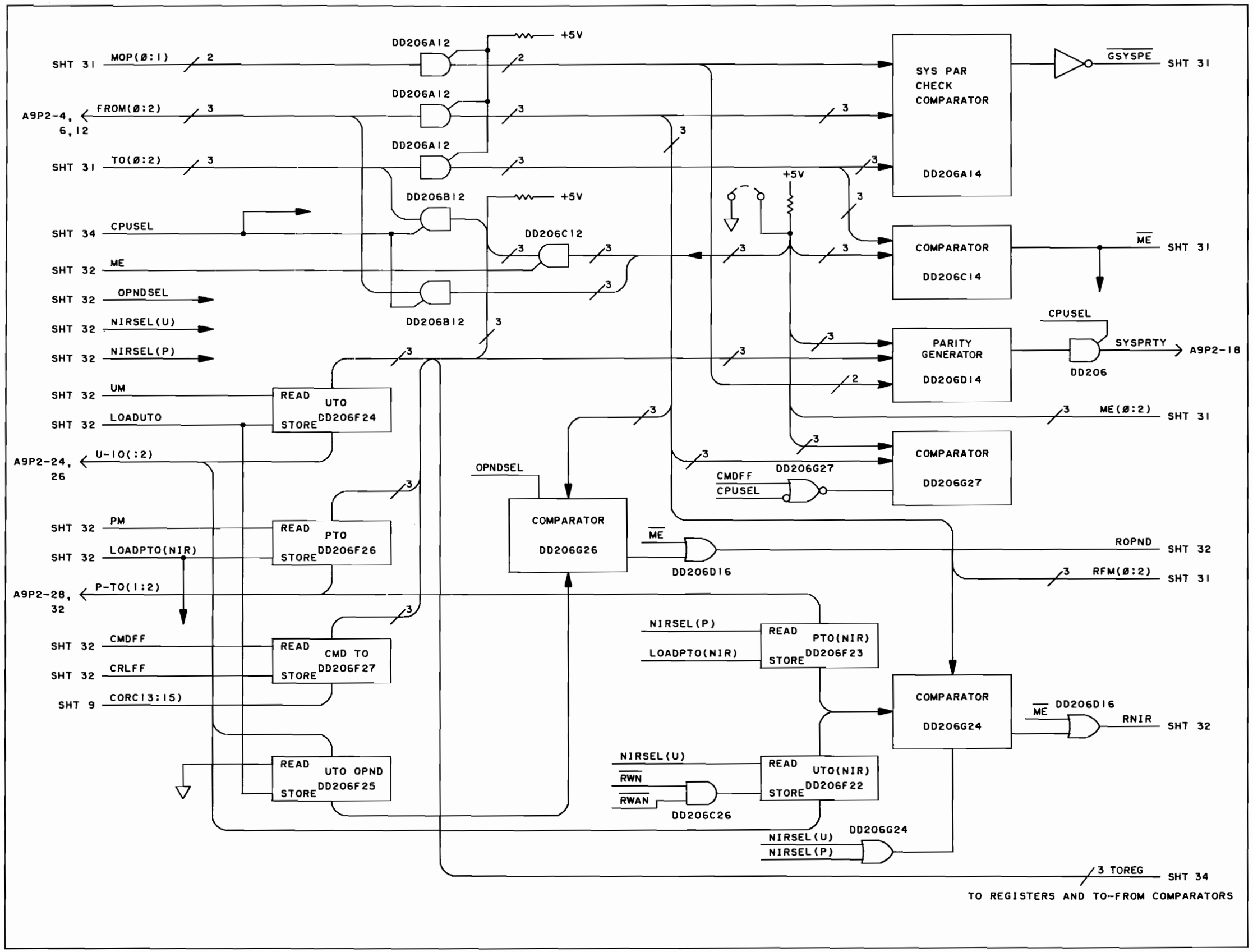


Figure 3-22. CPU/IOP Simplified Logic Diagrams (Sheet 35 of 35)

4-1. INTRODUCTION.

4-2. This section contains general servicing information, preventive maintenance information, and troubleshooting information applicable to the CPU/IOP. Detailed diagrams for the CPU/IOP are set numbers DD200 through DD211, part nos. 30001-90005 through 30001-90019, and are contained in the *HP 3000 Computer System Detailed Diagrams Manual*, part no. 03000-90023. Parts information for servicing and replacement of components is contained in the *HP 3000 Computer System Illustrated Parts Breakdown Manual*, part no. 03000-90021. The CPU stand-alone diagnostic is described in the *HP 3000 Manual of Stand-Alone Diagnostics*, part no. 03000-90027. The microdiagnostics for the CPU/IOP are as follows:

a. CPU/1 Microdiagnostic, part no. 32300-60001.

b. CPU/2 Microdiagnostic, part no. 32300-60002.

c. CTL/1 Microdiagnostic, part no. 32300-60003.

d. CTL/2 Microdiagnostic, part no. 32300-60004.

e. IOP/MUX Microdiagnostic, part no. 32300-60005.

4-3. In addition to the above data, the microprogram listing for the CPU should be referenced when troubleshooting the CPU/IOP.

4-4. GENERAL SERVICING INFORMATION.

4-5. The following paragraphs contain safety precautions to observe when servicing the CPU/IOP, wiring information, signal information, and a list of servicing equipment required for testing the CPU/IOP.

4-6. SAFETY PRECAUTIONS.

CAUTION

Failure to observe the following precautions could result in damage to components of the CPU/IOP PCA's or other components in the computer system.

4-7. When any CPU/IOP PCA is being installed, removed, or placed on an extender PCA for maintenance and troubleshooting procedures, the computer SYSTEM DC POWER switch must be set to the STANDBY position to remove power from the CPU/IOP connectors. Failure to observe this precaution could result in damage to the CPU/IOP or computer system connectors.

4-8. WIRING INFORMATION.

4-9. Wiring information for the CPU/IOP is contained in the *HP 3000 Computer System Detailed Diagrams Manual*, part no. 03000-90023.

4-10. CPU/IOP SIGNALS AND MNEMONICS.

4-11. A list of CPU/IOP signals and mnemonics is presented in table 4-1. In addition to the signal name and mnemonic, table 4-1 lists the location of the source of each signal (the detailed diagram number and the coordinates where the signal originates). For those signals which do not originate in the CPU/IOP, the source column of table 4-1 lists the diagram number and coordinates where the signal enters the CPU/IOP.

4-12. TEST EQUIPMENT AND DATA REQUIRED.

4-13. Test equipment and data (in addition to this manual) required to service and troubleshoot the CPU/IOP consists of the following items:

a. HP 30350A Auxiliary Control Panel.

b. HP 30352A Hardware Maintenance Panel.

c. HP 30353A ROM Simulator (Electronic Tool ET-6710).

d. Dummy Controller (Electronic Tool ET-6722).

e. TTL Logic Tool.

f. CPU Stand-Alone Diagnostic Magnetic Tape, HP 32320.

g. CPU/1 Microdiagnostic Paper Tape, part no. 32300-60001.

h. CPU/2 Microdiagnostic Paper Tape, part no. 32300-60002.

i. CTL/1 Microdiagnostic Paper Tape, part no. 32300-60003.

j. CTL/2 Microdiagnostic Paper Tape, part no. 32300-60004.

k. IOP/MUX Microdiagnostic Paper Tape, part no. 32300-60005.

l. *HP 30001A CPU/IOP Microprogram Listing Manual*, part no. 03000-90022.

m. CPU Microdiagnostic Listings, part no. 32300-90001A, 32300-90002A, 32300-90003A, 32300-90004A, 32300-90005A.

n. *HP 3000 Manual of Stand-Alone Diagnostics, Stand-Alone HP 30001A CPU Diagnostic*, part no. 03000-90027.

o. *HP 3000 Computer System Detailed Diagrams Manual*, part no. 03000-90023.

Table 4-1. CPU/IOP/MCU Signals

MNEMONIC	SIGNAL NAME	SOURCE
AENB	Address Enable	'DD207B45
ALPHA	Alphabetic Character	DD203D29
ALUFC	ALU Force Carry	DD202D38
ALUMODE	ALU Mode	DD202C39
ALUS (0:3)	ALU Input S0:S3	DD202D37
APEFF	Address Parity Error FF	DD207B15
BMCUPRTY	Buffered MCU Parity	DD206G17
BNDT	Bounds Test	DD202C34
BNDV	Bounds Violation	DD201B29
BUSOP	Bus Operation	DD200G6
CARRY	Carry	DD202C6
CCPX	Clear CPX	DD201B5
CE	Count Enable	DD207C48
CHACT	Channel Active	DD207D33
CHECK CYCLE	Check Cycle	DD206G9
CIR (0:15)	Current Instruction Register, Bits 0:15	DD205B10
CLFLAG3	Clear Flag 3 FF	DD201B5
CLIB	Clear Indirect Bit	DD201C3
CLK MODE 0	Clock Mode 0	DD201A14
CLK MODE 1	Clock Mode 1	DD201A15
CLOCK	Clock	DD201C18
CLOCK DIVIDER 0	Clock Divider 0	DD201C16
CLOCK DIVIDER 1	Clock Divider 1	DD201C17
CLOCKENB	Clock Enable	A4P1-27 DD201C11
CLOCKS	Clocks	DD201C18
CLSR	Clear SR Register	DD201B3
CMCU ILLADD	Clear MCU Illegal Address FF	DD206H14
CMD	Command	DD206C25
CMPRFF	Compare FF	DD205B27
CNTRMAX	Counter Maximum	DD204G23
COR (0,1,2,10,11,13,14,15)	CPU Output Register, Bits 0,1,2,10,11,13,14,15	DD204B5
CPUHRFF	CPU High Request FF	DD206D9
CPUIN	CPU In	DD206E18
CPULRFF	CPU Low Request FF	DD206C9
CPU RESET DLY	CPU Reset Delay	DD205G16
CPURST	CPU Reset	DD201E19
CPUSEL	CPU Select	DD206B8
CPU TIMER	CPU Timer	DD201D17
CRL	Control	DD206C25
CWANOP	Clear-Write Address No Operation	DD206E9
DAG	Direct Address Gate	DD207C35
DATA	Data	DD200H6
DATA CYC	Data Cycle	DD207F44
DATAPE	Data Parity Error	DD205B7
DATAPOLL	Data Poll	DD207E42

Table 4-1. CPU/IOP/MCU Signals (Continued)

MNEMONIC	SIGNAL NAME	SOURCE
DDG	Direct Data Gate	DD207B35
DECSR	Decrement SR Register	DD201C5
DEVNO (0:7)	Device Number (0:7)	DD207B19, DD207F11
DISPFLAG	Dispatcher Flag	DD202G46
DISPIOP	Display IOP	DD204G50
DISPLAY	Display	DD200A1, DD201F29, DD202B32, DD203C2, DD207G21
DPOLL	Data Poll	DD207C39
DRTG	Device Reference Table Gate	DD207D35
DRTINH	Device Reference Table Inhibit	DD207D46
DRT1	Device Reference Table 1	DD207D44
DS	DB or S-Relative Addressing	DD205G13
ENABLE (0:5)	Enable (0:5)	DD206A2
ENABLROM	Enable ROM	DD200C12
ENTIMER	Enable Timer	DD201E11, DD206G31
ERFRZINH	Error Freeze Inhibit	DD206F11
ERR CYC	Error Cycle	DD207B12
EXTCLK	External Clock	DD201B12
EXTINT	External Interrupt	DD207G35
FHB	Flag to High Bit	DD201A4
FLAG1	Flag 1	DD201G9
FLAG2	Flag 2	DD201F18
FLAG3	Flag 3	DD207B44
FPNLOS	Front Panel One-Shot	DD205B24
FREEZE	Freeze	DD206H15
FROM (0:2)	From Lines 0, 1, 2	DD206A11
FRUNCLK	Free Running Clock	DD201D18
FUBUS	Force U-bus	DD202B11
HALT	Halt	DD201B3
HSREQ	High Service Request	DD207E41
IENB	Interrupt Enable	DD207F36
INCNAMER	Increment Namer	DD201C5
INCP	Increment P-Register	DD201F29
INCSR	Increment SR Register	DD201C3
INCTR	Increment Counter	DD201C3
INDIRECT	Indirect	DD205F15
INHROMJ1	Inhibit ROM	DD200F11
INSTSEL	Instruction Select	DD207G24
INTACK	Interrupt Acknowledge	DD207G32
INTFLAG	Interrupt Flag	DD202G46
INTPOLL	Interrupt Poll	DD207F37
INTREQ	Interrupt Request	DD207G32
INTRP	Interrupt	DD205G28
IOAPE	I/O Address Parity Error	DD207B15

Table 4-1. CPU/IOP/MCU Signals (Continued)

MNEMONIC	SIGNAL NAME	SOURCE
IOCMD (0:2)	I/O Command (0:2)	DD207E41, DD207D18
IOCMP	I/O Compare	DD207C45
IOD (0:15)	I/O Data (0:15)	DD207C6
IODPE	I/O Data Parity Error	DD207F37
IODPRTY	I/O Data Parity	DD207B6
IOERROR	I/O Error	DD207A18
IOFLG1	I/O Flag 1	DD207C36
IOFRZ	I/O Freeze	DD206F2, DD207E31
IOHIREQ	I/O High Request	DD207D49
IOHRQFF	I/O High Request FF	DD206C33
IOHSEL	I/O High Select	DD206D36
IORSREQ	I/O High Service Request	DD207B37
IOILG	I/O Illegal Address Error	DD207A15
IOINP	I/O In Process	DD206G36
IOLHSEL	I/O Low High Select	DD206B36
IOLOREQ	I/O Low Request	DD207C45
IOLOSEL	I/O Low Select	DD207C36
IOLRQFF	I/O Low Request FF	DD206B33
IOLRDY	I/O Low Ready	DD206C38
IOMOP (0:2)	I/O Memory Operation Code (0:2)	DD207E48
IORESET	I/O Reset	DD201E19
IORSTSW	I/O Reset Switch	DD201E11
IOSTROBE	I/O Strobe	DD206G38
IOTIMER	I/O Timer	DD206G39
IOTO ((1:2)	I/O To Line Bits 1 and 2	DD207G16
JBNDV	Jump Bounds Violation	DD201A27
JLUI1	Jump Look-Up Table Indirect, Rank 1	DD200C9
JLUI2	Jump Look-Up Table Indirect, Rank 2	DD202C48
JMP	Jump	DD202D35
JMPFRZ	Jump Freeze	DD201E29
JMPGATE	Jump Gate	DD201C25
JMPJSB	Jump or Jump to Subroutine	DD202F39
JMPJSB1	Jump or Jump to Subroutine, Rank 1	DD202G36
JUMPER (0:2)	Jumpers (0:2)	DD204H3
JSB	Jump to Subroutine	DD202C35
JSB1	Jump to Subroutine, Rank 1	DD202F36
LDENB	Load Enable	DD207E47
LOREQINIT	Low Request Initializer	DD207D46
LOSEL	Low Select	DD206F8
LREQ	Low Request	DD206C8
LUTGATE	Look-Up Table Gate	DD201D27
MCIOTMR	MC I/O Timer	DD206G38
MCUADDRPE	MCU Address Parity Error	DD206B19
MCUCLKS	MCU Clocks	DD201C19
MCUCLK 1:5	MCU Clock 1:5	DD201B19

Table 4-1. CPU/IOP/MCU Signals (Continued)

MNEMONIC	SIGNAL NAME	SOURCE
MCUCMP	MCU Compare	DD205B36
MCUCMPH	MCU Compare Halt	DD205C21
MCUCMPL	MCU Compare Light	DD205D21
MCUDZ (0:15)	MCU Data Bits 0:15	DD204B6, DD205D2, DD207B30
MCUDPE	MCU Data Parity Error	DD206A11, DD207B11
MCUDPRTY	MCU Data Parity	DD206H17
MCUERR	MCU Error	DD206A19
MCUHINT	MCU Halt Interrupt	DD205G21
MCUINT	MCU Interrupt	DD206E19
MCURST	MCU Reset	DD206G3
MCUSYSPE	MCU System Parity Error	DD206B16
MDPARITY	Memory Data Parity	DD204H4
ME	Me	DD206E27
MEMBND	Memory Bound	DD207E42
MODINT	Module Interrupt	DD206D19
MOP (0:1)	Memory Operation Code Bits 0, 1	DD206A11
MPIFRZ	Maintenance Panel Interface Freeze	DD206G11
MSKRTRN	Mask Return	DD207C19
NAMER (0:1)	Namer Bits 0:1	DD203C18
NEXT	Next	DD201G22
NIP	Next In Process	DD206B29
NIPG	Next In Process Gate	DD206C29
NIPSELENB	Next In Process Select Enable	DD206C29
NIRSEL (P)	Next Instruciton Register Select (P)	DD206E23
NIRSEL (U)	Next Instruction Register Select (U)	DD206E22
NIRTOCIR	NIR to CIR	DD201C26
NIRTOCIRDLY	NIR to CIR	DD201C26
NOP	No Operation	DD205H18
NOP1	No Operation, Rank 1	DD201C27
NOP2	No Operation, Rank 2	DD201B27
NUMERIC	Numeric Character	DD203D29
NXT1	Next, Rank 1	DD200B9
NXT2	Next, Rank 2	DD202C48
NXT=1	Next Counter=1	DD201G27
NXT=2	Next Counter=2	DD201G28
NXTDCD	Next Decoded	DD201G22
NXTDLD	Next Delayed	DD206D23
NXTFINH	Next Fetch Inhibit	DD201F28
NXTGATE	Next Gate	DD201F29
OFCENB	Overflow Carry Eanble	DD202E39
OPINP	Operand In Process	DD206B29
OPNDSEL	Operand Select	DD206D8
OUTBND	Out Bound	DD207F38
OVFL	Overflow	DD202D8

Table 4-1. CPU/IOP/MCU Signals (Continued)

MNEMONIC	SIGNAL NAME	SOURCE
PADDIN (8:11)	Pre-Adder Input, Bits 8:11	DD205C40
PADDSUB	Pre-adder Add/Subtract Control	DD205E40
PADDX	Pre-adder X Shift Control	DD205F39
PADDXS (0:1)	Pre-adder X Shift Control, Bits 0:1	DD205G39
PANLREAD	Panel Read	DD202B34
PANLSTOR	Panel Store	DD202F38
PAUSE	Pause	DD205H8
PFWARN	Power Fail Warning	DD207H31
PFWARNB	Power Fail Warning, Buffered	DD207H38
PM	P-to-Memory	DD206E25
POLLORSO	Poll or Service Out	DD207G47
PRTYMODE	Parity Mode	DD202G38
PSELECT	P Select	DD206E25
PTO1	PCOR Register to Line, Bit 1	DD204G13
PTO2	PCOR Register to Line, Bit 2	DD204E13
PWRFAIL	Power Fail	DD201E13
PWRFAILRST	Power Fail Reset	DD201D14
PWRON	Power On	DD201E12
QS	Q or S Relative Addressing	DD205G13
QUP	Queue Up	DD204E28
RAR 0:11	ROM Address Register Bits 0:11	DD200A10
RARDIS	ROM Address Register Display	DD200B1
RDIOA	Read I/O Address	DD207E12
RDIOD	Read I/O Data	DD207E12
RDCIR	Read Current Instruction Register	DD203A15
RDCPX1	Read CPX1 Register	DD204F44
RDCPX2	Read CPX2 Register	DD204G43
RDIOM	Read I/O Memory Data	DD204E42
RDJMPR	Read Jumpers	DD203B15
RDMSK	Read Mask Register	DD207E12
RDMOD	Read Module Number (Interrupt Module Number Register)	DD204F43
RDOPND	Read Operand Register	DD204G43
RDSWITCH	Read Switch Register	DD204G43
READY ((0:6)	Ready Lines 0:6	DD206D2
REPEAT	Repeat	DD201A26
REPC	Repeat Until Condition	DD202D34
REPN	Repeat N Times	DD202D38
RF (0:3)	R-bus Field Bits 0:3	DD200H4
RFINH	R-bus Field Inhibit	DD202G36
RFSAME	R-bus Field Same-Preserve R-bus Register	DD203D6
RMCUCPX1	Read MCU CPX1	DD205D27
ROM(0:31)	ROM Bits 0:31	DD200A14
ROMFCNT	ROM Function	DD202D38
ROMFCN1	ROM Function, Rank 1	DD202H36
RORT	ROM Output Register, Rank 2	DD202B40

Table 4-1. CPU/IOP/MCU Signals (Continued)

MNEMONIC	SIGNAL NAME	SOURCE
RPTFCN	Repeat Function	DD202B46
RREG0	R-bus Register, Bit 0	DD202B46
RSB	Return From Subroutine	DD201E26
RSSEL	R-bus or S-bus Field Select	DD202B34
RUN	Run	
RUNFF	Run FF	DD205C29
R(0:15)	R-bus Bits 0:15	DD200D10
SAME	Same	DD202F9
SCIR	Serialized Current Instruction Register	DD205D9
SDISFLAG	Set Dispatcher Flag	DD201B4
SFSAME	S-bus Field Same-Preserve S-bus Register	DD204C43
SFQO	S-bus Field High Decode=0	DD204E42
SF 0:4	S-bus Bits 0:4	DD200H2
SI	Service In	DD207E42
SI (D)	Service In Delayed	DD207C33
SINGINST1	Single Instruction Bit 1	DD205E24
SINGINST2	Single Instruction Bit 2	DD205E26
SINTFLAG	Set Interrupt Flag	DD201A4
SIOACT	SIO Active	DD207D33
SIOP	Serialized IOP Signals	DD207B4
SKIP	Skip	DD201B25
SKIPNOP	Skip Field No Operation	DD202D39
SMCUDATA	Serialized MCU Data	DD205E7
SO	Service Out	DD207F49
SO1	Service Out 1	DD207F46
SO1 (D)	Service Out FF1 Delayed	DD206B33
SO2	Service Out FF2	DD207F48
SP1X14	Scratch Pad 1X Register, Bit 14	DD203F23
SP1X15	Scratch Pad 1X Register, Bit 15	DD203F22
SPV	Special V-bus	DD205G18
SP1IN	Scratch Pad 1 Register Shift Input	DD202F14
SP1SHIFT	Scratch Pad 1 Register Shift Control	DD202G13
SP3IN	Scratch Pad 3 Register Shift Input	DD202F14
SP3SHIFT	Scratch Pad 3 Register Shift Control	DD202H13
SP3(15)	Scratch Pad 3 Register Bit 15	DD204G9
SRBUS	Serialized R-bus	DD203D55
SRREG	Serialized R-bus Register	DD202E47
SR(0:2)	Stack Register, Bits 0:2	DD203D15
SSBUS	Serialized S-bus	DD203D53
SSREG	Serialized S-bus Register	DD202E28
STATUS 0	Status Bit 0	DD201F26
STATUS 1	Status Bit 1	DD201F26
STATUS 2	Status Bit 2	DD201F26
STATUS 3	Status Bit 3	DD201G25
STATUS 4	Status Bit 4	DD201D9

Table 4-1. CPU/IOP/MCU Signals (Continued)

MNEMONIC	SIGNAL NAME	SOURCE
STATUS 5	Status Bit 5	DD201C8
STATUS 6	Status Bit 6	DD201E8
STATUS 7	Status Bit 7	DD201F8
STATUS (8:15)	Status Bits 8:15	DD204D38
STIOA	Store I/O Address	DD207D12
STIOD	Store I/O Data	DD207D12
STIOM	Store I/O Memory Data	DD204B28
STKBNOP	Stack Op B, No Operation	DD205G9
STMSK	Store Mask	DD207D12
STOFROM	Store From	DD206D39
STORAR	Store ROM Address Register	DD203H13
STSTATUS	Store Status	DD204F30
ST(0:4)	Store Bits 0:4	DD200H5
SWLDRAR	Switch Load ROM Address Register	DD200F1
SYSDUMP	System Dump	DD205C21
SYSHALT	System Halt	DD205C21
SYSPE	System Parity Error	DD207A11, DD206E11
SYSPRTY	System Parity	DD206H17
S(0:15)	S-bus, Bits 0:15	DD204A50
TEST1	Test Point 1, Clock Circuits	DD201B12
TEST2	Test Point 2, Clock Circuits	DD201B12
TINT	Test Interrupt	DD201F21
TMRFRZI	Timer Freeze Inhibit	DD201E19
TNAME(0:1)	Tname, Bits 0:1	DD203D18
TO(0:2)	To Lines, 0:2	DD206A11
T=0	T-bus=0	DD202F8
T(0)	T-bus, Bit 0	DD202C7
T(15)	T-bus, Bit 15	DD202C7
UBNT	Unconditional Bounds Test	DD202D34
UM	U to Memory	DD206E24
UNFRZ	Unfreeze	DD206H11
U-TO1	UCOR to Line, Bit 1	DD204C13
U-TO2	UCOR to Line, Bit 2	DD204B13
U(0:15)	U-bus, Bits (0:15)	DD202A20
V(0:7)	V-bus, Bits (0:7)	DD200C10
W	W-Bit	DD205F12
XERR	Transfer Error	DD207F44

p. HP 3000 Computer System Diagnostic Monitor, part no. 03000-90016.

q. *HP 3000 Computer System Illustrated Parts Breakdown Manual*, part no. 03000-90021.

4-14. PREVENTIVE MAINTENANCE.

4-15. Preventive maintenance for the CPU/IOP should be performed each time the preventive maintenance procedures are performed for the HP 3000 Computer System. Preventive maintenance consists of inspecting the CPU/IOP PCA's and the interconnecting cable assemblies for burned or broken components, loose connections, and deteriorated insulating materials.

4-16. TROUBLESHOOTING.

4-17. Once a malfunction is detected, troubleshooting the CPU/IOP can be accomplished most efficiently by performing the following steps in the order given:

a. Run the on-line diagnostics from the system disc.

b. Load and run the stand-alone diagnostics. The stand-alone diagnostics require that the system be capable of cold loading.

c. If the system cannot be cold loaded, the microdiagnostics should be run.

d. The CPU microdiagnostics are written in microcode, allowing each field and function to be exercised. The microdiagnostic references the signal indicators on the hardware maintenance panel and auxiliary control panel. When a specific signal indicator does not light, refer to table 4-3, LUT-to-Microprogram ROM Index; figure 4-1, Instruction-to-Microprogram Index; and the servicing diagrams, figures 4-2 through 4-74 to isolate the cause of the malfunction. The servicing diagrams show the source of each signal or register capable of being displayed on the hardware maintenance panel or the auxiliary control panel.

4-18. ON-LINE DIAGNOSTICS.

4-19. The on-line diagnostics are resident on the system disc and can be called by entering the following command from the terminal: :RUN SDM

4-20. A detailed description of the on-line diagnostics is contained in the *HP 3000 Computer System Diagnostic Monitor*, part no. 03000-90016.

4-21. STAND-ALONE DIAGNOSTIC.

4-22. The CPU stand-alone diagnostic is described in the *HP 3000 Manual of Stand-Alone Diagnostics*, part no. 03000-90027. The CPU stand-alone diagnostic is divided into five sections. The five sections are on magnetic tape in cold load format. To cold load the magnetic tape, proceed as follows:

a. Press the RUN-HALT switch (on the auxiliary control panel) if the computer is running.

b. Install the reel of tape on the tape unit to be used.

c. Set the tape at load point by means of the LOAD POINT pushbutton on the tape unit.

d. If there is more than one tape unit in the system, press the UNIT SELECT 0 pushbutton on the tape unit to be used. Ensure that no other tape unit has this pushbutton lighted.

e. Press the ON LINE pushbutton on the tape unit.

f. Set into bit positions 0 thru 7 of the B-register switches (on the auxiliary control panel) the cold load control byte for magnetic tape. This number is 006 octal (logic 1 in bit positions five and six).

g. Set into bit positions 8 thru 15 of the B-register switches (on the auxiliary control panel) the device number associated with the magnetic tape unit. (Normally, one controller is used for each four tape units. The tape units are numbered zero through three: unit zero is used for cold loading.)

h. Press, in turn, the I/O RESET and CPU RESET switch on the auxiliary control panel. This resets the I/O system and selects tape unit zero.

i. Press the COLD LOAD switch on the auxiliary control panel.

j. Press the RUN-HALT switch on the auxiliary control panel.

4-23. MICRODIAGNOSTICS.

4-24. The microdiagnostics for the CPU/IOP consist of five parts: CPU microdiagnostic parts one and two, central data bus microdiagnostic parts one and two, and IOP/multiplexer microdiagnostic. All five microdiagnostics use Electronic Tool ET-6710 (ROM Simulator). In addition, the IOP/MUX microdiagnostic uses Electronic Tool ET-6722 (Dummy Controller). The microdiagnostics perform the following tests:

a. CPU Microdiagnostic, Part one - Tests single cycle operations and a kernel group of microcode.

b. CPU Microdiagnostic, Part two - Tests the CPU microcode further using the verified single cycle operations and verified microcode.

c. Central Data Bus Microdiagnostic, Part one - Tests that portion of the microcode that performs bus operations.

d. Central Data Bus Microdiagnostic, Part two - Performs an address and checkerboard test on memory.

e. IOP/MUX Microdiagnostic - Tests the data in/out, device number in/out, and command out lines of the IOP bus. Interrupts from any device number can be simulated. The I/O data, device number, command lines, and key control signals are displayed.

4-25. ROM SIMULATOR. Before running the CPU microdiagnostics, connect the ROM simulator as follows:

a. Remove the back panel of the ROM simulator and remove the RAM PCA (part no. 03000-92031) from the card storage slot.

b. Turn computer system power off by setting the SYSTEM DC POWER switch (located in right-hand bay, upper right-hand corner) to STANDBY. Open upper front panel for access to this switch.

c. Plug the RAM PCA into the 1A2 slot of the CPU (see figure 1-1).

d. Connect the auxiliary control panel and hardware maintenance panel to the system.

e. Connect the cable (supplied with the ROM simulator) between J4 on the ROM simulator and J3 on the PCA's located in slots 1A1 (Auxiliary Control Panel Interface PCA) and 1A2 (RAM PCA).

f. Plug the ROM simulator into a 115 Vac, 60 Hz source.

g. Turn computer system power on by setting the SYSTEM DC POWER switch to ON.

h. Press POWER switch on ROM simulator.

1. REMOTE POWER indicator should light.

2. READY indicator should light.

i. Set the FUNCTION switch on ROM simulator to CPU SELF TEST.

j. Press the RESET switch on ROM simulator.

k. Press the LOAD switch on ROM simulator. PASS indicator should light indicating that the ROM simulator self-test is finished and the ROM simulator is operational.

4-26. Load the microdiagnostic paper tape into the ROM simulator paper tape reader as follows:

a. Set the FUNCTION switch on the ROM simulator to the CPU MDIAG position.

b. Press the RESET switch on the ROM simulator.

c. Load paper tape into paper tape reader.

d. Set the PAGE NUMBER switches on the ROM simulator to value of test pages to be loaded.

e. Set the CLOCK SINGLE CYCLE/FREE RUN switch on hardware maintenance panel to the SINGLE CYCLE position.

f. Press, in turn, the CPU RESET and I/O RESET switches on auxiliary control panel.

g. Press the EXECUTE SINGLE CYCLE switch (on hardware maintenance panel) twice.

h. Press the LOAD switch on the ROM simulator.

i. Clear the RAR LOAD register on the hardware maintenance panel by setting all RAR LOAD REGISTER switches to the down position.

j. Press the LOAD RAR switch on hardware maintenance panel. Press the EXECUTE SINGLE CYCLE switch. RAR should equal zero.

4-27. DUMMY CONTROLLER. The dummy controller (Electronic Tool ET-6722) is used when the IOP/MUX microdiagnostic is run. Connect the dummy controller as follows:

a. Connect interrupt poll wires to the INT POLL terminals on the dummy controller.

b. Set the device number of the multiplexer channel in the left byte of the B-register switches (bit positions 0 through 7) on the auxiliary control panel.

c. Set the device number of the dummy controller in the right byte of the B-register switches (bit positions 8 through 15) on the auxiliary control panel.

Note: The device numbers set into the B-register switches must total less than 150 octal for both the multiplexer channel and the dummy controller.

4-28. To aid in troubleshooting the CPU/IOP, a LUT-to-microprogram ROM index (table 4-3), instruction-to-microprogram index (figure 4-1) and servicing diagrams (figures 4-2 through 4-74) are included at the end of this section. The microdiagnostics specify which indicators should be lighted on the hardware maintenance panel and what the contents of the registers displayed on the auxiliary control panel should contain. If any of these indications are not as specified in the microdiagnostic, refer to the appropriate servicing diagram to further isolate the cause of the malfunction. The servicing diagrams show the logic source for each of the displayed signals. Table 4-2 contains a cross-reference index listing the circuit name and the figure number which contains the logic for the signal generated by the circuit. Where possible, the circuit name in table 4-2 matches the label of the signal indicator on the auxiliary control panel and hardware maintenance panel.

Table 4-2. Circuit-to-Servicing Diagram Cross Reference Index

CIRCUIT NAME	SERVICING DIAGRAM FIGURE NO.
ALU CARRY	4-54
ALU OVFLO	4-54
BNDV	4-54
CHECK CYCLE	4-69
CNTR	4-24
CPU HIRQ	4-48
CPU LORQ	4-48
CPU SELECT	4-48
CPU TIMER	4-56
CPX1	4-25
CPX2	4-26
CURRENT INSTRUCTION REGISTER	4-2
DATA CYCLE	4-69
DATA POLL	4-68
DB	4-37
DIRECT ACTIVE	4-68
DIR I/O DATA IN	4-23
DISP FLAG	4-58
DL	4-36
DRTE+2	4-70
DRT REQ	4-66
DRT STORE	4-69
DS	4-62
ENB 0, ENB 1, ENB 2, ENB 3, ENB 4, ENB 5	4-50
EXT INT	4-55
FLAG 1, FLAG 2, FLAG 3	4-51
FREEZE	4-57
FROM 0, FROM 1, FROM 2	4-44
ICS FLAG	4-58
IN BOUND	4-66
INSTR SELECT	4-65
INSTR WAIT	4-60
INTRP	4-55
INTRPT ACK	4-71
INTRPTTG DVC NO.	4-22
INTRPT POLL	4-71
IOB ENABLE	4-70
IOHREQ	4-50
IOLOREQ	4-50
I/O SELECT	4-49
IOTIMER	4-56
I/O WAIT	4-60
JUMP	4-66
LUTGATE	4-64
MASK	4-40

Table 4-2. Circuit-to-Servicing Diagram Cross Reference Index (Continued)

CIRCUIT NAME	SERVICING DIAGRAM FIGURE NO.
MCUD PARITY	4-42
MCUD PE	4-47
MOD NO.	4-28
MOP 0, MOP 1	4-45
MPIFRZ	4-50
MUX I/O DATA	4-27
NOP 1, NOP 2	4-53
NXT-1, NXT-2	4-59
OPND	4-21
OPND SELECT	4-65
OPND WAIT	4-60
P	4-34
PADD	4-29
PB	4-33
PL	4-35
Q	4-38
QS	4-62
RA	4-3, 4-4, 4-5, 4-6
RB	4-7, 4-8, 4-9, 4-10
RC	4-11, 4-12, 4-13, 4-14
RD	4-15, 4-16, 4-17, 4-18
RDY 0, RDY 1, RDY 2, RDY 3, RDY 4, RDY 5, RDY 6	4-50
REPEAT	4-53
RUN-HALT	4-41
Serializer A	4-72
Serializer B	4-50
Serializer C	4-73
SERVICE IN	4-66
SERVICE OUT	4-67
SIO ACTIVE	4-68
SKIP	4-63
SPI 1	4-74
SP2	4-19
SP3	4-20
SR	4-30
STATUS	4-31
SYSTEM PARITY	4-46
SYSTEM PE	4-47
TNAME 0, TNAME 1	4-52
TO 0, TO 1, TO 2	4-43
W	4-61
X	4-32
XFR ERROR	4-69
Z	4-39

Table 4-3. LUT-to-Microprogram ROM Index

LUT ENTRY ADDR	OPCODE	INST	ACTION LABEL	REMARKS	ROM ADDR
000	OP 00-03		TRP7	NOT USED	
017				NOT USED	
020	OP04	LOAD	AC1D	ADDR COMP – DB, Q RELATIVE	13
021	OP04		AC1S	ADDR COMP – S RELATIVE	12
022	OP04		LOAD	JUMP IF NOT INDIRECT	122
023	OP04		AC1P	ADDR COMP – P RELATIVE	27
024	OP05	STOR	AC1D	ADDR COMP – DB, Q RELATIVE	13
025	OP05		AC1S	ADDR COMP – S RELATIVE	12
026	OP05		STOR	JUMP IF NOT INDIRECT	220
027	OP05	TBA, MTBA TBX, MTBX	LCBI		416
030	OP06	CMPM	AC1D	ADDR COMP – DB, Q RELATIVE	13
031	OP06		AC1S	ADDR COMP – S RELATIVE	12
032	OP06		CMPM	JUMP IF NOT INDIRECT	400
033	OP06		AC1P	ADDR COMP – P RELATIVE	27
034	OP07	ADDM	AC1D	ADDR COMP – DB, Q RELATIVE	13
035	OP07		AC1S	ADDR COMP – S RELATIVE	12
036	OP07		ADDM	JUMP IF NOT INDIRECT	254
037	OP07		AC1P	ADDR COMP – P RELATIVE	27
040	OP10	SUBM	AC1D	ADDR COMP – DB, Q RELATIVE	13
041	OP10		AC1S	ADDR COMP – S RELATIVE	12
042	OP10		SUBM	JUMP IF NOT INDIRECT	257
043	OP10		AC1P	ADDR COMP – P RELATIVE	27
044	OP11	MPYM	AC1D	ADDR COMP – DB, Q RELATIVE	13
045	OP11		AC1S	ADDR COMP – S RELATIVE	12
046	OP11		MPYM	JUMP IF NOT INDIRECT	1700
047	OP11		AC1P	ADDR COMP – P RELATIVE	27
050	OP12	DECM	AC1D	ADDR COMP – DB, Q RELATIVE	13
051	OP12		AC1S	ADDR COMP – S RELATIVE	12
052	OP12		IDMY		262
053	OP12	INCM	INCM	INCM MEM – DB, Q, S RELATIVE	35
054	OP13	LDX	AC1D	ADDR COMP – DB, Q RELATIVE	13
055	OP13		AC1S	ADDR COMP – S RELATIVE	12
056	OP13		LDX	JUMP IF NOT INDIRECT	121
057	OP13		AC1P	ADDR COMP – P RELATIVE	27
060	OP14	BR or BCC	BRD		272
061	OP14	BR or BCC	BRS		271

Table 4-3. LUT-to-Microprogram ROM Index (Continued)

LUT ENTRY ADDR	OPCODE	INST	ACTION LABEL	REMARKS	ROM ADDR
062	OP14	unused	TRP7	NOT USED	
063	OP14	BR	BRP		302
064	OP15	LDD	AC3D	ADDR COMP – DB, Q RELATIVE	62
065	OP15		AC3S	ADDR COMP – S RELATIVE	61
066	OP15		LDD	JUMP IF NOT INDIRECT	130
067	OP15	LDB	LDB	LOAD BYTE – DB, Q, S RELATIVE	143
070	OP16	STD	AC4D	ADDR COMP – DB, Q RELATIVE	102
071	OP16		AC4S	ADDR COMP – S RELATIVE	101
072	OP16		STD	JUMP IF NOT INDIRECT	225
073	OP16	STB	LDB	LOAD BYTE – DB, Q, S RELATIVE	143
074	OP17	LRA	AC2D	ADDR COMP – DB, Q RELATIVE	44
075	OP17		AC2S	ADDR COMP – S RELATIVE	43
076	OP17		LRA	JUMP IF NOT INDIRECT	124
077	OP17		AC2P	ADDR COMP – P RELATIVE	55
100	Sub 2 = 00	–	TRP7	NOT USED	
101	Spec 2 = 00	MOVE PB	PROG		2027
102	Sub 3 = 00	–	TRP7	NOT USED	
103	Spec 3 = 00	–	%3373	SPARE. MODULE 6 ENTRY	
104	Sub 2 = 01	–	%3372	SPARE. MODULE 6 ENTRY	
105	Spec 2 = 01	MOVE DB	DATA	MOVE ENTRY -- DB RELATIVE	2030
106	Sub 3 = 01	SCAL	SCAL		2545
107	Spec 3 = 01	PAUS	PAUS		1541
110	Sub 2 = 02	LDI	LPNI		1521
111	Spec 2 = 02	MVB PB	PROG		2027
112	Sub 3 = 02	PCAL	PCAL		2546
113	Spec 3 = 02	SED	SED		2315
114	Sub 2 = 03	LDXI	XPNI		1527
115	Spec 2 = 03	MVB DB	DATA		2030
116	Sub 3 = 03	EXIT	EXIT		2621
117	Spec 3 = 03	XCHD	XCHD	INSURE A TOS REG	1674
120	Sub 2 = 04	CMPI	CPNI		460
121	Spec 2 = 04	MVBL	MVBL	MOVE FROM DB+ TO DL+	2000
122	Sub 3 = 04	SXIT	SXIT		1531
123	Spec 3 = 04	SMSK	SMSK		2332
124	Sub 2 = 05	ADDI	ASI		1523
125	Spec 2 = 05	SCW	SCAM		2126

Table 4-3. LUT-to-Microprogram ROM Index (Continued)

LUT ENTRY ADDR	OPCODE	INST	ACTION LABEL	REMARKS	ROM ADDR
126	Sub 3 = 05	ADXI	ASXI		1530
127	Spec 3 = 05	RMSK	RMSK		2330
130	Sub 2 = 06	SUBI	ASI		1523
131	Spec 2 = 06	MVLB	MVLB	MOVE FROM DL+ TO DB+	2003
132	Sub 3 = 06	SBXI	ASXI		
133	Spec 3 = 06	XEQ	XEQ		1545
134	Sub 2 = 07	MPYI	MPYI		1703
135	Spec 2 = 07	SCU	SCAM		2126
136	Sub 3 = 07	LLBL	LLBL		3407
137	Spec 3 = 07	SIO	SIO		2233
140	Sub 2 = 10	DIVI	DIVI		566
141	Spec 2 = 10	MVBW	SCAM		2126
142	Sub 3 = 10	LDPP	LODP		206
143	Spec 3 = 10	RIO	RIO		2252
144	Sub 2 = 11	PSHR	PSHR		1557
145	Spec 2 = 11	MVBW	SCAM		2126
146	Sub 3 = 11	LDPN	LODP		206
147	Spec 3 = 11	WIO	WIO		2265
150	Sub 2 = 12	LDNI	LPNI		1521
151	Spec 2 = 12	CMPB PB	CPBP		2023
152	Sub 3 = 12	ADDS	ADDS		1661
153	Spec 3 = 12	TIO	TIO		2300
154	Sub 2 = 13	LDXN	XPNI		1524
155	Spec 2 = 13	CMPB DB	CPBD		2024
156	Sub 3 = 13	SUBS	SUBS		1660
157	Spec 3 = 13	CIO	CIO		2306
160	Sub 2 = 14	CMPN	CPNI		460
161	Spec 2 = 14	RSW/LLSH	RSW		351
162	Sub 3 = 14	TSBM	TSBM		1443
163	Spec 3 = 14	CMD	CMD		2356
164	Sub 2 = 15	EXF	EXF		1475
165	Spec 2 = 15	PLDA/PSTA	PLDA		753
166	Sub 3 = 15	ORI	ORI		1524
167	Spec 3 = 15	SIRF	SIRF		2343
170	Sub 2 = 16	DPF	DPF		1501
171	Spec 2 = 16	—	%3377	SPARE. MODULE 6 ENTRY	
172	Sub 3 = 16	XORI	XORI		1525
173	Spec 3 = 16	SIN	SIN		2323

Table 4-3. LUT-to-Microprogram ROM Index (Continued)

LUT ENTRY ADDR	OPCODE	INST	ACTION LABEL	REMARKS	ROM ADDR
174	Sub 2 = 17	SETR	SETR		1605
175	Spec 2 = 17	–	%3374	SPARE. MODULE 6 ENTRY	
176	Sub 3 = 17	ANDI	ANDI		1526
177	Spec 3 = 17	HALT	HALT		2733
200	Subop 1 = 00	ASL	SHFL	SHIFT LEFT IF SR > = 2	732
201	Subop 1 = 01	ASR	SHFR	SHIFT RIGHT IF SR > = 2	724
202	Subop 1 = 02	LSL	SHFL	SHIFT LEFT IF SR > = 2	732
203	Subop 1 = 03	LSR	SHFR	SHIFT RIGHT IF SR > = 2	724
204	Subop 1 = 04	CSL	SHFL	SHIFT LEFT IF SR > = 2	732
205	Subop 1 = 05	CSR	SHFR	SHIFT RIGHT IF SR > = 2	724
206	Subop 1 = 06	SCAN	SCAN		740
207	Subop 1 = 07	IABZ	IABZ		2363
210	Subop 1 = 10	TASL	TASL		1417
211	Subop 1 = 11	TASR	TASR		1432
212	Subop 1 = 12	IXBZ	IXBZ		3401
213	Subop 1 = 13	DXBZ	DXBZ		3404
214	Subop 1 = 14	BCY	BCY		336
215	Subop 1 = 15	BNCY	BNCY		342
216	Subop 1 = 16	TNSL	TNSL		1342
217	Subop 1 = 17	–	%3375	SPARE	
220	Subop 1 = 20	DASL	SHDL		1400
221	Subop 1 = 21	DASR	SHDR		1410
222	Subop 1 = 22	DLSL	SHDL		1400
223	Subop 1 = 23	DLSR	SHDR		1410
224	Subop 1 = 24	DCSL	SHDL		1400
225	Subop 1 = 25	DCSR	SHDR		1410
226	Subop 1 = 26	CPRB	CPRB		405
227	Subop 1 = 27	DABZ	DABZ		2366
230	Subop 1 = 30	BOV	BOV		345
231	Subop 1 = 31	BNOV	BNOV		1770
232	Subop 1 = 32	TBC	TBC		1465
233	Subop 1 = 33	TRBC	TRBC		1467
234	Subop 1 = 34	TSBC	TSBC		1471
235	Subop 1 = 35	TCBC	TCBC		1473
236	Subop 1 = 36	BRO	BRO		2371
237	Subop 1 = 37	BRE	BRE		1773
240	Subop 0 = 20	ADD	ADD		463

Table 4-3. LUT-to-Microprogram ROM Index (Continued)

LUT ENTRY ADDR	OPCODE	INST	ACTION LABEL	REMARKS	ROM ADDR
241	Subop 0 = 21	SUB	SUB		464
242	Subop 0 = 22	MPY	MPY		1705
243	Subop 0 = 23	DIV	DIV		577
244	Subop 0 = 24	NEG	NEG		465
245	Subop 0 = 25	TEST	TEST		632
246	Subop 0 = 26	STBX	STBX		717
247	Subop 0 = 27	DTST	DTST		636
250	Subop 0 = 30	DFLT	DFLT		1234
251	Subop 0 = 31	BTST	BTST		644
252	Subop 0 = 32	XCH	XCH		702
253	Subop 0 = 33	INCA	INCA		652
254	Subop 0 = 34	DECA	DECA		651
255	Subop 0 = 35	XAX	XAX		704
256	Subop 0 = 36	ADAX	ADAX		720
257	Subop 0 = 37	ADXA	ADXA		722
260	Subop 0 = 60	LADD	LADD		504
261	Subop 0 = 61	LSUB	LSUB		1
262	Subop 0 = 62	LMPY	LMPY		511
263	Subop 0 = 63	LDIV	LDIV		517
264	Subop 0 = 64	NOT	NOT		537
265	Subop 0 = 65	OR	OR		646
266	Subop 0 = 66	XOR	XOR		647
267	Subop 0 = 67	AND	AND		650
270	Subop 0 = 70	FIXR	FIXR		1277
271	Subop 0 = 71	FIXT	FIXT		1276
272	Subop 0 = 72	—	%3376	SPARE. MODULE 6 ENTRY	
273	Subop 0 = 73	INCB	INCB		654
274	Subop 0 = 74	DECB	DECB		653
275	Subop 0 = 75	XBX	XBX		706
276	Subop 0 = 76	ADBX	ADBX		721
277	Subop 0 = 77	ADXB	ADXB		723
300	Subop 0 = 00	NOP	NOP	Unused. See entry 2 ↑	1466
301	Subop 0 = 01	DELB	DELB	Entry if SR ≥ 2	667
302	Subop 0 = 02	DDEL	DDEL	Entry if SR ≥ 2	770
303	Subop 0 = 03	ZROX	ZROX	Unused. See entry 2	665
304	Subop 0 = 04	INCX	INCX	Unused. See entry 2	656
305	Subop 0 = 05	DECX	DECX	Unused. See entry 2 ↓	655

ENTRY NO. 1

Table 4-3. LUT-to-Microprogram ROM Index (Continued)

LUT ENTRY ADDR	OPCODE	INST	ACTION LABEL	REMARKS	ROM ADDR
306	Subop 0 = 06	ZERO	ZRO1	Entry if SR = 4	663
307	Subop 0 = 07	DZRO	ZRO2	Entry if SR > 2	657
310	Subop 0 = 10	DCMP	SRP	Entry until SR = 4	4
311	Supop 0 = 11	DADD	SRP	Entry until SR = 4	4
312	Subop 0 = 12	DSUB	SRP	Entry until SR = 4	4
313	Subop 0 = 13	MPYL	SRP	Entry until SR = 4	4
314	Subop 0 = 14	DIVL	SRP	Entry until SR = 4	4
315	Subop 0 = 15	DNEG	SRP	Entry until SR = 4	4
316	Subop 0 = 16	DXCH	SRP	Entry until SR = 4	4
317	Subop 0 = 17	CMP	SRP	Entry until SR = 4	4
320	Subop 0 = 40	DEL	DEL	Entry if SR ⩾ 2	471
321	Subop 0 = 41	ZROB	ZROB	Entry if SR ⩾ 2	666
322	Subop 0 = 42	LDXB	LDXB	Entry if SR ⩾ 2	715
323	Subop 0 = 43	STAX	STAX	Entry if SR ⩾ 2	716
324	Subop 0 = 44	LDXA	LDX1	Entry if SR ⩾ 2	713
325	Subop 0 = 45	DUP	DUP1	Entry if SR ⩾ 2	670
326	Subop 0 = 46	DDUP	DDP1	Entry if SR ⩾ 2	673
327	Subop 0 = 47	FLT	FLT1	Entry if SR ⩾ 2	1226
330	Subop 0 = 50	FCMP	SRP	Entry until SR = 4	4
331	Subop 0 = 51	FADD	SRP	Entry until SR = 4	4
332	Subop 0 = 52	FSUB	SRP	Entry until SR = 4	4
333	Subop 0 = 53	FMPY	SRP	Entry until SR = 4	4
334	Subop 0 = 54	FDIV	SRP	Entry until SR = 4	4
335	Subop 0 = 55	FNEG	SRP	Entry until SR = 4	4
336	Subop 0 = 56	CAB	SRP	Entry until SR = 4	4
337	Subop 0 = 57	LCMP	SRP	Entry until SR = 4	4
340	Subop 0 = 00	NOP	NOP	Normal entry	1466
341	Subop 0 = 01	DELB	SRP	Entry if SR < 2	4
342	Subop 0 = 02	DDEL	DDL1	Entry if SR < 2	472
343	Subop 0 = 03	ZROX	ZROX	Normal entry	665
344	Subop 0 = 04	INCX	INCX	Normal entry	656
345	Subop 0 = 05	DECX	DECX	Normal entry	655
346	Subop 0 = 06	ZERO	ZERO	Entry if SR < 4	644
347	Subop 0 = 07	DZRO	DZRO	Entry if SR ⩽ 2	662
350	Subop 0 = 10	DCMP	DCMP	Entry if SR = 4	554
351	Subop 0 = 11	DADD	DADD	Entry if SR = 4	550
352	Subop 0 = 12	DSUB	DSUB	Entry if SR = 4	540
353	Subop 0 = 13	MPYL	MPYL	Entry if SR = 4	1706

ENTRY NO. 1

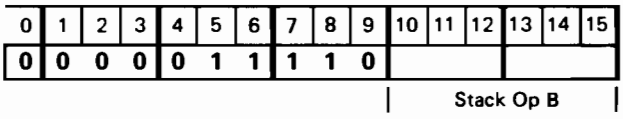
ENTRY NO. 2

Table 4-3. LUT-to-Microprogram ROM Index (Continued)

LUT ENTRY ADDR	OPCODE	INST	ACTION LABEL	REMARKS	ROM ADDR
354	Subop 0 = 14	DIVL	DIVL	Entry if SR = 4	604
355	Subop 0 = 15	DNEG	DNEG	Entry if SR = 4	544
356	Subop 0 = 16	DXCH	DXCH	Entry if SR = 4	633
357	Subop 0 = 17	CMP	CMP	Entry if SR = 4	453
360	Subop 0 = 40	DEL	DEL1	Entry if SR < 2	476
361	Subop 0 = 41	ZROB	SRP	Entry if SR < 2	4
362	Subop 0 = 42	LDXB	SRP	Entry if SR < 2	4
363	Subop 0 = 43	STAX	SRP	Entry if SR < 2	4
364	Subop 0 = 44	LDXA	LDXA	Entry if SR < 2	714
365	Subop 0 = 45	DUP	DUP	Entry if SR < 2	671
366	Subop 0 = 46	DDUP	DDUP	Entry if SR < 2	676
367	Subop 0 = 47	FLT	FLT	Entry if SR < 2	1231
370	Subop 0 = 50	FCMP	FCMP	Entry if SR = 4	1255
371	Subop 0 = 51	FADD	FADD	Entry if SR = 4	1001
372	Subop 0 = 52	FSUB	FSUB	Entry if SR = 4	1000
373	Subop 0 = 53	FMPY	FMPY	Entry if SR = 4	1110
374	Subop 0 = 54	FDIV	FDIV	Entry if SR = 4	1141
375	Subop 0 = 55	FNEG	FNEG	Entry if SR = 4	1104
376	Subop 0 = 56	CAB	CAB	Entry if SR = 4	710
377	Subop 0 = 57	LCMP	LCMP	Entry if SR = 4	466

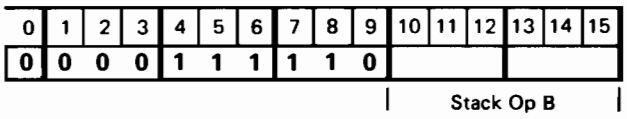
ENTRY NO. 2

ADAX



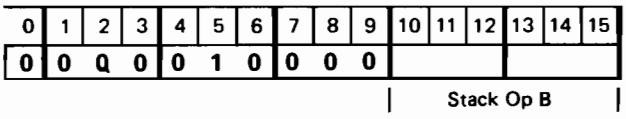
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	256	ADAX	0720	1

ADBX



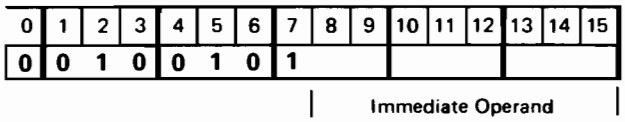
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	276	ADBX	0721	1

ADD



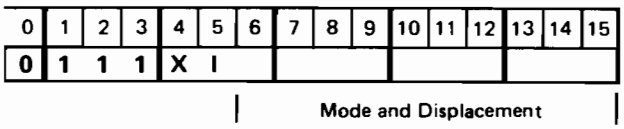
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	240	ADD	0463	1

ADDI



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	124	ASI	1523	0

ADDM

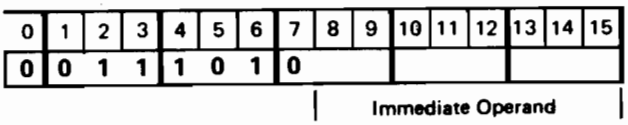


COND	LUTA	LABEL	RAR	W
P REL.	037	AC1P	0027	1
DB REL.	034	AC1D	0013	1
Q REL.	034	AC1D	0013	1
S REL.	035	AC1S	0012	1
JLUI	036	ADDM	0254	1

2184-51

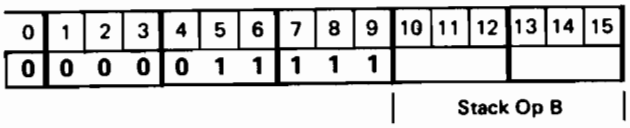
Figure 4-1. Instruction to Microprogram Index (Sheet 1 of 34)

ADDS



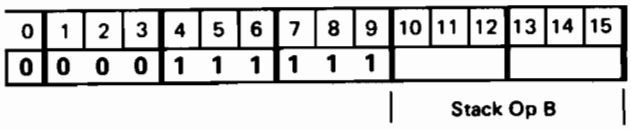
COND	LUTA	LABEL	RAR	W
UNC.	152	ADDS	1661	0

ADXA



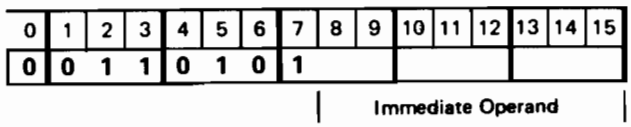
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	257	ADXA	0722	1

ADXB



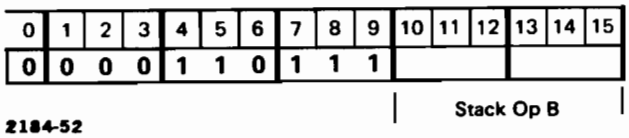
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	277	ADXB	0723	1

ADXI



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	126	ASXI	1530	0

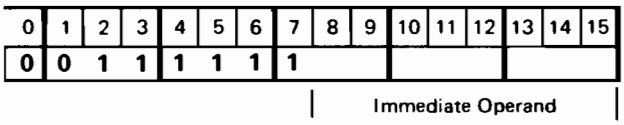
AND



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	267	AND	0650	1

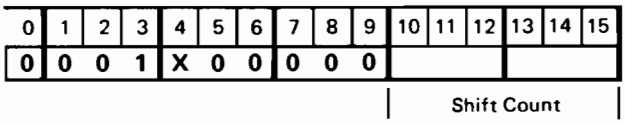
Figure 4-1. Instruction to Microprogram Index (Sheet 2 of 34)

ANDI



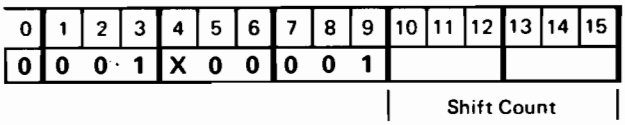
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	176	ANDI	1526	0

ASL



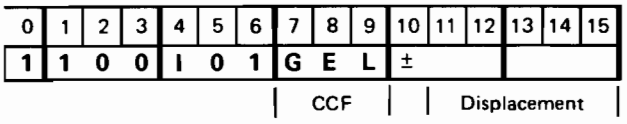
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	200	SHFL	0732	0

ASR



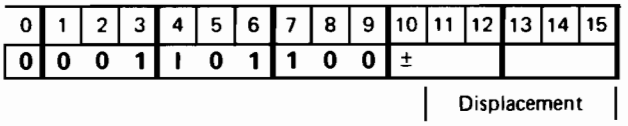
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	201	SHFR	0724	0

BCC



COND	LUTA	LABEL	RAR	W
UNC.	060	BRD	0272	0

BCY

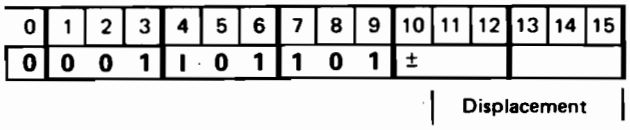


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	214	BCY	0336	1

2184-53

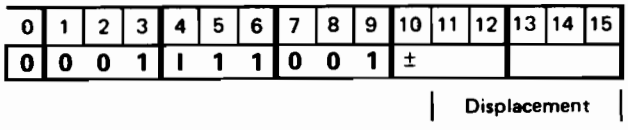
Figure 4-1. Instruction to Microprogram Index (Sheet 3 of 34)

BNCY



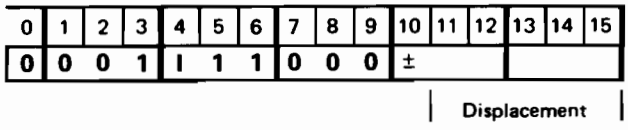
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	215	BNCY	0342	1

BNOV



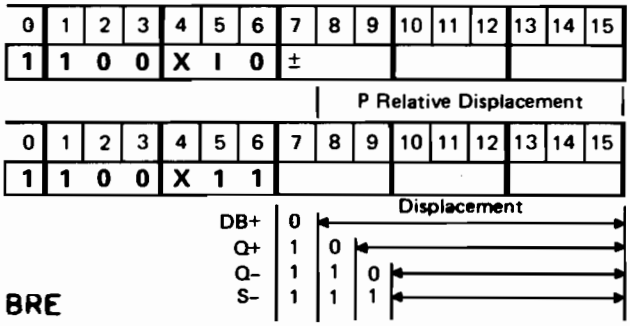
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	231	BNOV	1770	1

BOV



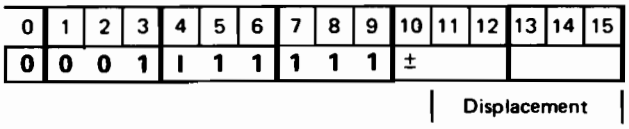
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	230	BOV	0345	1

BR



COND	LUTA	LABEL	RAR	W
P REL.	063	BRP	0302	0
DB REL.	060	BRD	0272	0
Q REL.	060	BRD	0272	0
S REL.	061	BRS	0271	0

BRE

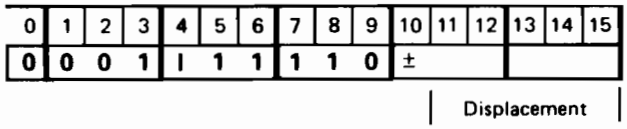


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	237	BRE	1773	1

2184-54

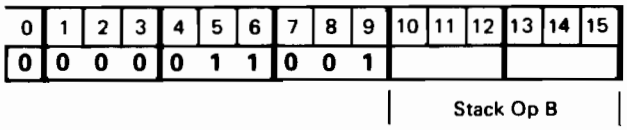
Figure 4-1. Instruction to Microprogram Index (Sheet 4 of 34)

BRO



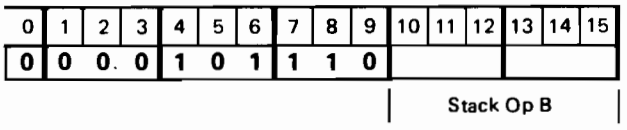
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	236	BRO	2371	1

BTST



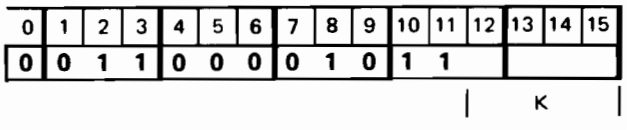
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	251	BTST	0644	1

CAB



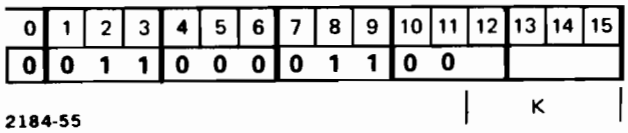
COND	LUTA	LABEL	RAR	W
SR<4	336	SRP	0004	1
SR=4	376	CAB	0710	1

CIO



COND	LUTA	LABEL	RAR	W
UNC.	157	CIO	2306	0

CMD

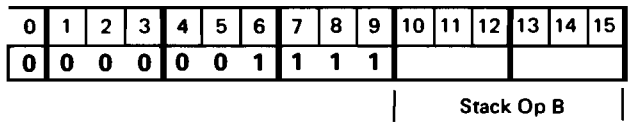


COND	LUTA	LABEL	RAR	W
UNC.	163	CMD	2356	0

2184-55

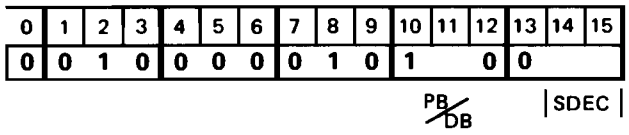
Figure 4-1. Instruction to Microprogram Index (Sheet 5 of 34)

CMP



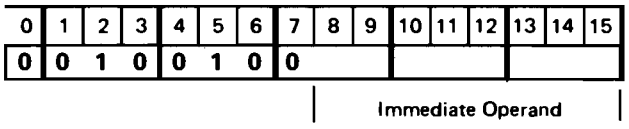
COND	LUTA	LABEL	RAR	W
SR<4	317	SRP	0004	1
SR=4	357	CMP	0453	1

CMPB



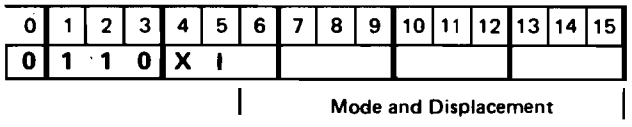
COND	LUTA	LABEL	RAR	W
P REL.	151	CPBP	2023	0
DB REL.	155	CPBD	2024	0

CMPI



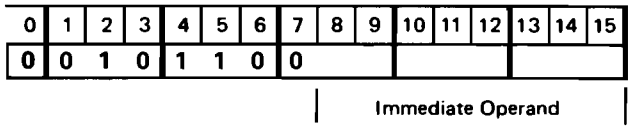
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	120	CPNI	0460	1

CMPM



COND	LUTA	LABEL	RAR	W
P REL.	033	AC1P	0027	1
DB REL.	030	AC1D	0013	1
Q REL.	030	AC1D	0013	1
S REL.	031	AC1S	0012	1
JLUI	032	CMPM	0400	1

CMPN

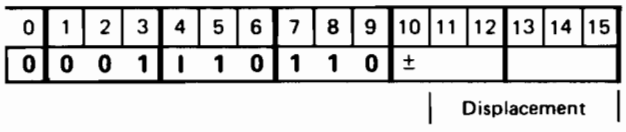


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	160	CPNI	0460	0

2184-56

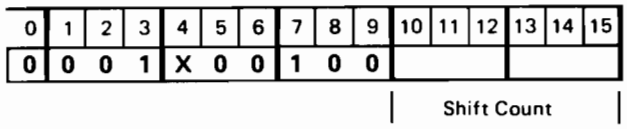
Figure 4-1. Instruction to Microprogram Index (Sheet 6 of 34)

CPRB



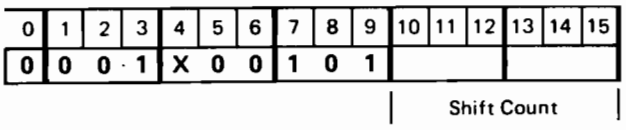
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	226	CPRB	0405	1

CSL



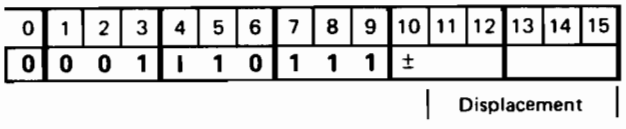
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	204	SHFL	0732	0

CSR



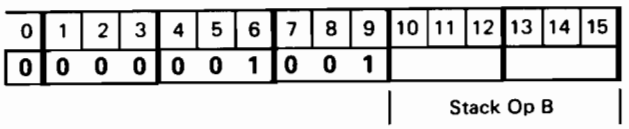
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	205	SHFR	0724	0

DABZ



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	227	DABZ	2366	1

DADD

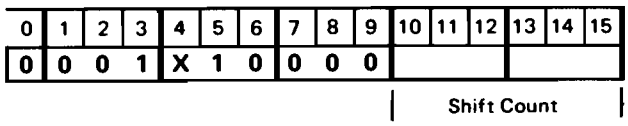


COND	LUTA	LABEL	RAR	W
SR<4	311	SRP	0004	1
SR=4	351	DADD	0550	1

2184-57

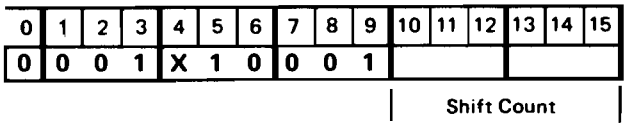
Figure 4-1. Instruction to Microprogram Index (Sheet 7 of 34)

DASL



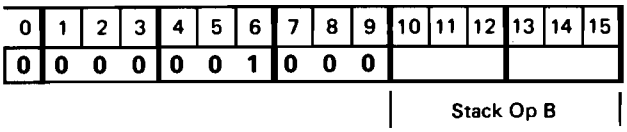
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	220	SHDL	1400	0

DASR



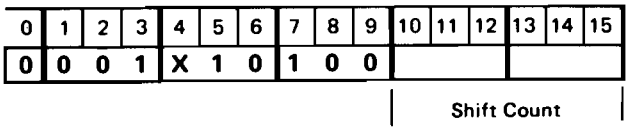
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	221	SHDR	1410	0

DCMP



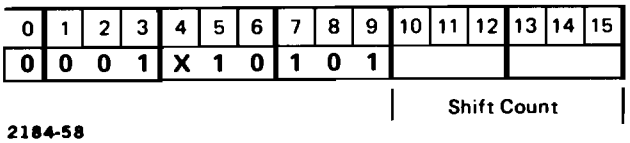
COND	LUTA	LABEL	RAR	W
SR<4	310	SRP	0004	1
SR=4	350	DCMP	0554	1

DCSL



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	224	SHDL	1400	0

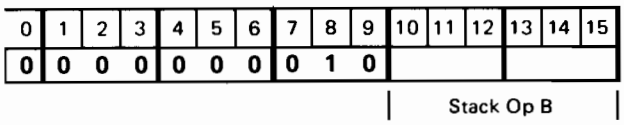
DCSR



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	225	SHDR	1410	0

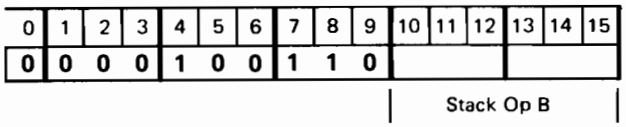
Figure 4-1. Instruction to Microprogram Index (Sheet 8 of 34)

DDEL



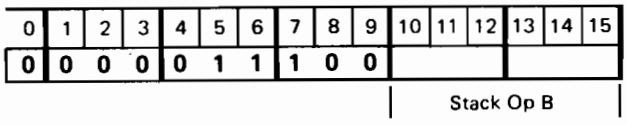
COND	LUTA	LABEL	RAR	W
SR<2	342	DDL1	0472	1
SR>=2	302	DDEL	0770	1

DDUP



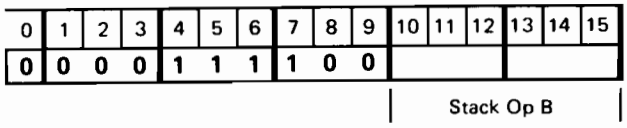
COND	LUTA	LABEL	RAR	W
SR<2	366	DDUP	0676	1
SR>=2	326	DDP1	0673	1

DECA



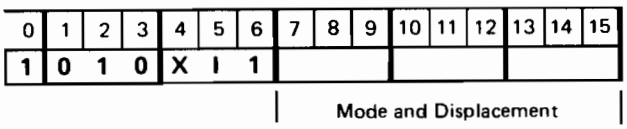
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	254	DECA	0651	1

DECB



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	274	DECB	0653	1

DECM

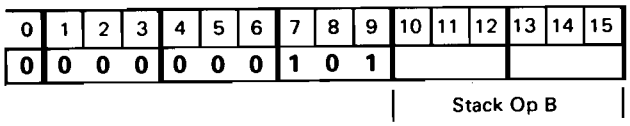


COND	LUTA	LABEL	RAR	W
DB REL.	050	AC1D	0013	1
Q REL.	050	AC1D	0013	1
S REL.	051	AC1S	0012	1
JLUI	052	IDMY	0262	1

2184-59

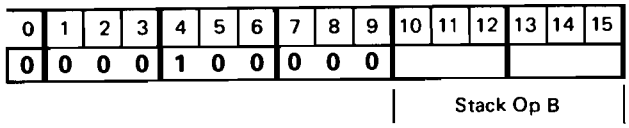
Figure 4-1. Instruction to Microprogram Index (Sheet 9 of 34)

DECX



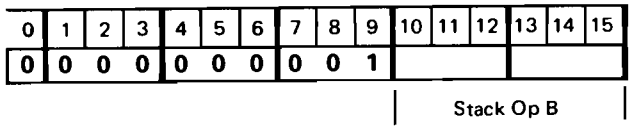
COND	LUTA	LABEL	RAR	W
SR<2	345	DECX	0655	1
SR>=2	305	DECX	0655	1

DEL



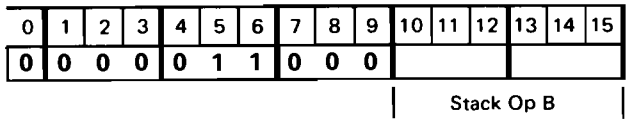
COND	LUTA	LABEL	RAR	W
SR<2	360	DEL1	0476	1
SR>=2	320	DEL	0471	1

DELB



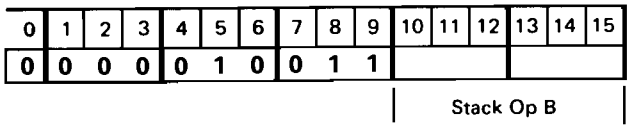
COND	LUTA	LABEL	RAR	W
SR<2	341	SRP	0004	1
SR>=2	301	DELB	0667	1

DFLT



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	250	DFLT	1234	1

DIV

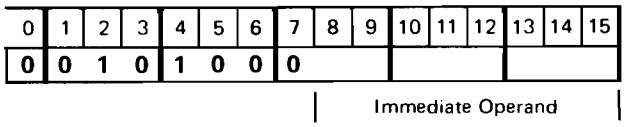


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	243	DIV	0577	1

2184-60

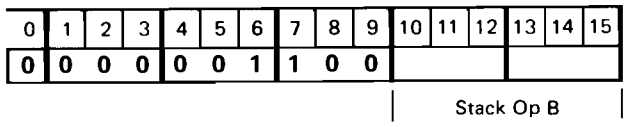
Figure 4-1. Instruction to Microprogram Index (Sheet 10 of 34)

DIVI



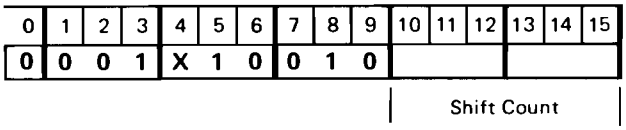
COND	LUTA	LABEL	RAR	W
UNC.	140	DIVI	0566	0

DIVL



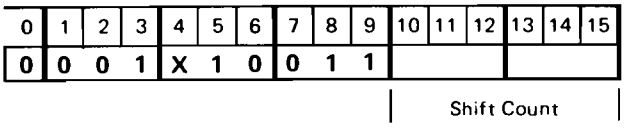
COND	LUTA	LABEL	RAR	W
SR<4	314	SRP	0004	1
SR=4	354	DIVL	0604	1

DLSL



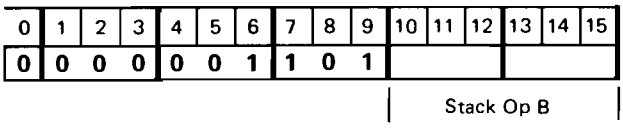
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	222	SHDL	1400	0

DLSR



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	223	SHDR	1410	0

DNEG

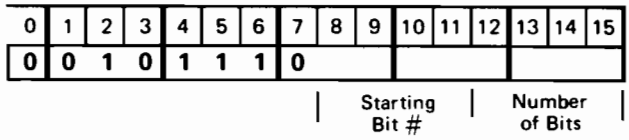


COND	LUTA	LABEL	RAR	W
SR<4	315	SRP	0004	1
SR=4	355	DNEG	0544	1

2184-61

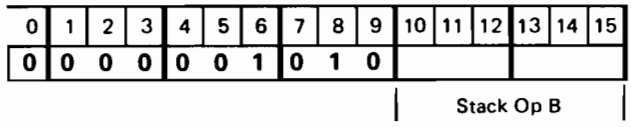
Figure 4-1. Instruction to Microprogram Index (Sheet 11 of 34)

DPF



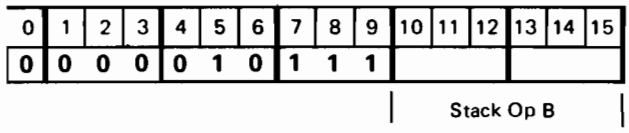
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	170	DPF	1501	0

DSUB



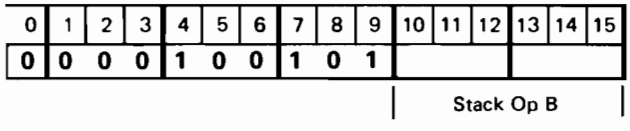
COND	LUTA	LABEL	RAR	W
SR<4	312	SRP	0004	1
SR=4	352	DSUB	0540	1

DTST



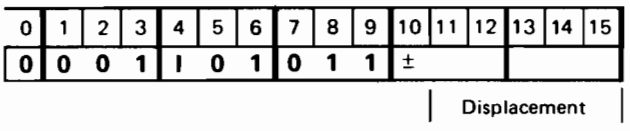
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	247	DTST	0636	1

DUP



COND	LUTA	LABEL	RAR	W
SR<2	365	DUP	0671	1
SR>=2	325	DUP1	0670	1

DXBZ

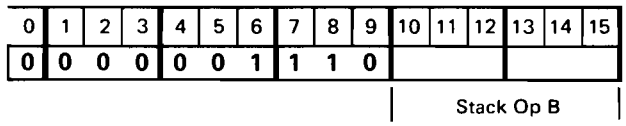


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	213	DXBZ	3404	1

2184-62

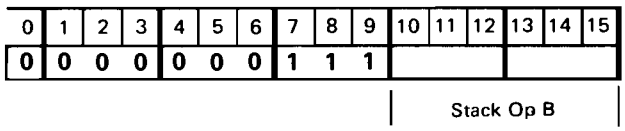
Figure 4-1. Instruction to Microprogram Index (Sheet 12 of 34)

DXCH



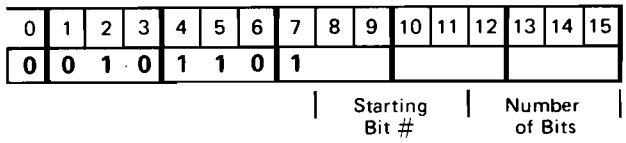
COND	LUTA	LABEL	RAR	W
SR<4	316	SRP	0004	1
SR=4	356	DXCH	0633	1

DZRO



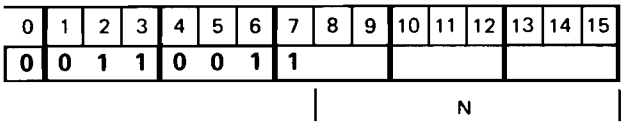
COND	LUTA	LABEL	RAR	W
SR<2	347	DZRO	0662	1
SR>=2	307	ZRO2	0657	1

EXF



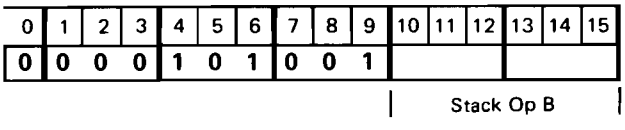
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	164	EXF	1475	0

EXIT



COND	LUTA	LABEL	RAR	W
UNC.	116	EXIT	2621	1

FADD

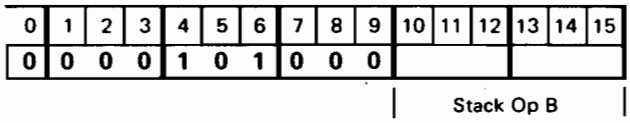


COND	LUTA	LABEL	RAR	W
SR<4	331	SRP	0004	1
SR=4	371	FADD	1001	1

2814-63

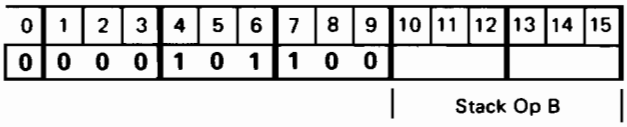
Figure 4-1. Instruction to Microprogram Index (Sheet 13 of 34)

FCMP



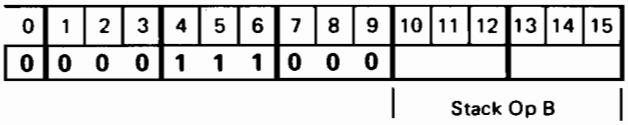
COND	LUTA	LABEL	RAR	W
SR<4	330	SRP	0004	1
SR=4	370	FCMP	1255	1

FDIV



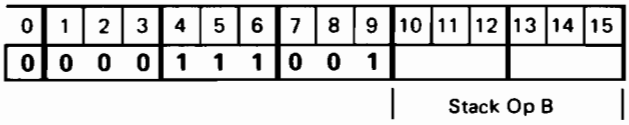
COND	LUTA	LABEL	RAR	W
SR<4	334	SRP	0004	1
SR=4	374	FDIV	1141	1

FIXR



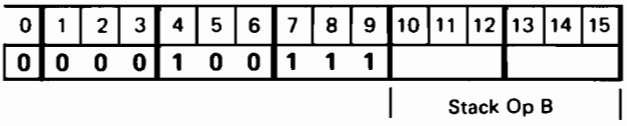
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	270	FIXR	1277	1

FIXT



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	271	FIXT	1276	1

FLT

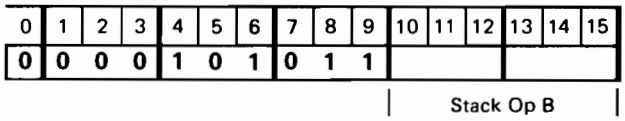


COND	LUTA	LABEL	RAR	W
SR<2	367	FLT	1231	1
SR>=2	327	FLT1	1226	1

2184-64

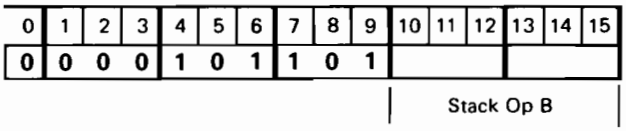
Figure 4-1. Instruction to Microprogram Index (Sheet 14 of 34)

FMPY



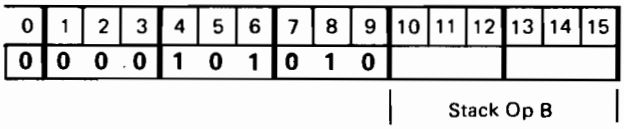
COND	LUTA	LABEL	RAR	W
SR<4	333	SRP	0004	1
SR=4	373	FMPY	1110	1

FNEG



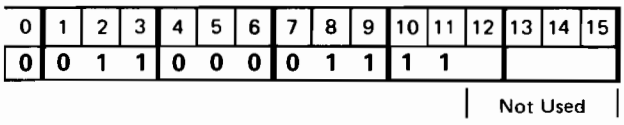
COND	LUTA	LABEL	RAR	W
SR<4	335	SRP	0004	1
SR=4	375	FNEG	1104	1

FSUB



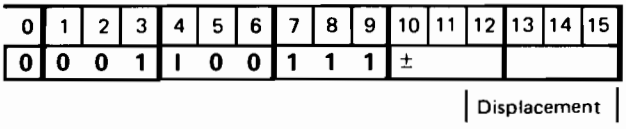
COND	LUTA	LABEL	RAR	W
SR<4	332	SRP	0004	1
SR=4	372	FSUB	1000	1

HALT



COND	LUTA	LABEL	RAR	W
UNC.	177	HALT	2733	1

IABZ

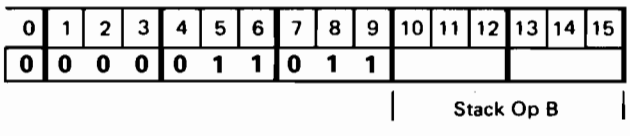


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	207	IABZ	2363	1

2184-65

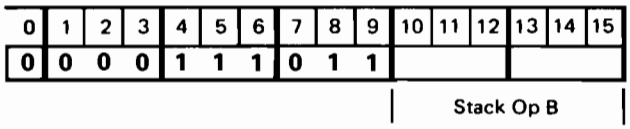
Figure 4-1. Instruction to Microprogram Index (Sheet 15 of 34)

INCA



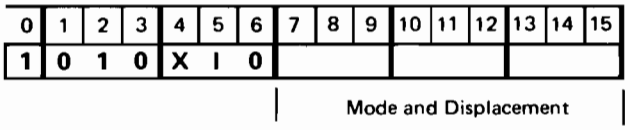
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	253	INCA	0652	1

INCB



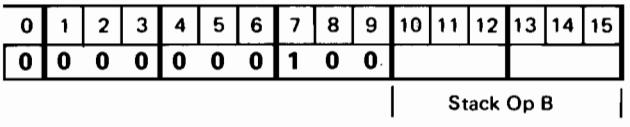
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	273	INCB	0654	1

INCM



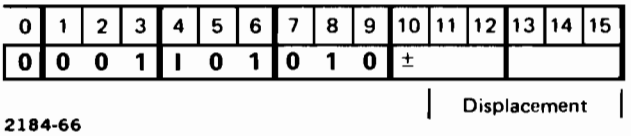
COND	LUTA	LABEL	RAR	W
DB REL.	053	INCM	0035	1
Q REL.	053	INCM	0035	1
S REL.	053	INCM	0035	1
JLUI	052	IDMY	0262	1

INCX



COND	LUTA	LABEL	RAR	W
SR<2	344	INCX	0656	1
SR>=2	304	INCX	0656	1

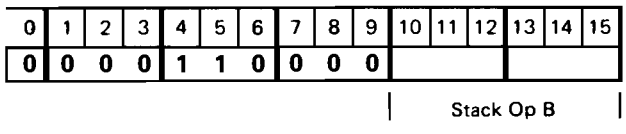
IXBZ



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	212	IXBZ	3401	1

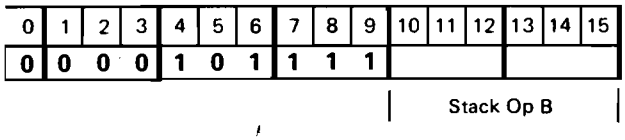
Figure 4-1. Instruction to Microprogram Index (Sheet 16 of 34)

LADD



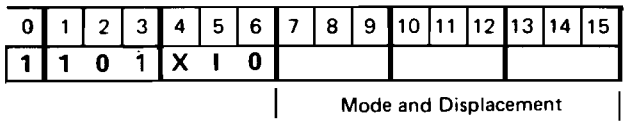
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	260	LADD	0504	1

LCMP



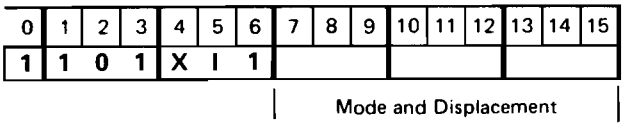
COND	LUTA	LABEL	RAR	W
SR<4	337	SRP	0004	1
SR=4	377	LCMP	0466	1

LDB



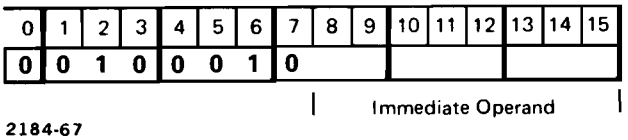
COND	LUTA	LABEL	RAR	W
DB REL.	067	LDB	0143	0
Q REL.	067	LDB	0143	0
S REL.	067	LDB	0143	0

LDD



COND	LUTA	LABEL	RAR	W
DB REL.	064	AC3D	0062	1
Q REL.	064	AC3D	0062	1
S REL.	065	AC3S	0061	1
JLUI	066	LDD	0130	1

LDI

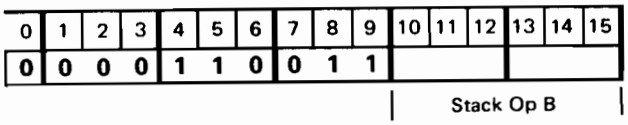


2184-67

COND	LUTA	LABEL	RAR	W
UNC.	110	LPNI	1521	0

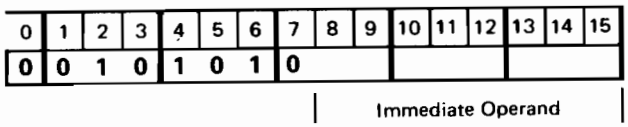
Figure 4-1. Instruction to Microprogram Index (Sheet 17 of 34)

LDIV



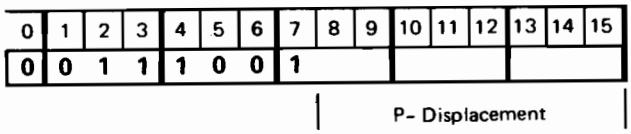
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	263	LDIV	0517	1

LDNI



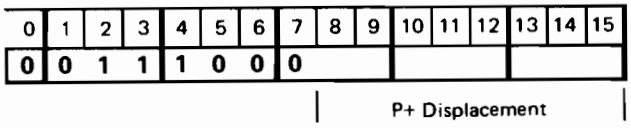
COND	LUTA	LABEL	RAR	W
UNC.	150	LPNI	1521	1

LDPN



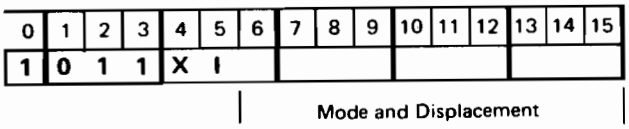
COND	LUTA	LABEL	RAR	W
UNC.	146	LODP	0206	1

LDPP



COND	LUTA	LABEL	RAR	W
UNC.	142	LODP	0206	0

LDX

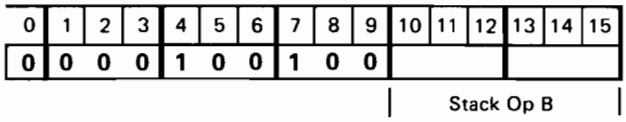


COND	LUTA	LABEL	RAR	W
P REL.	057	AC1P	0027	1
DB REL.	054	AC1D	0013	1
Q REL.	054	AC1D	0013	1
S REL.	055	AC1S	0012	1
JLUI	056	LDX	0121	1

2184-68

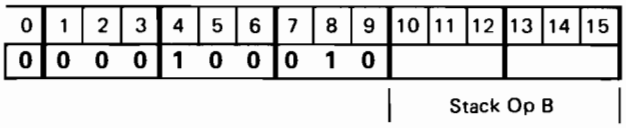
Figure 4-1. Instruction to Microprogram Index (Sheet 18 of 34)

LDXA



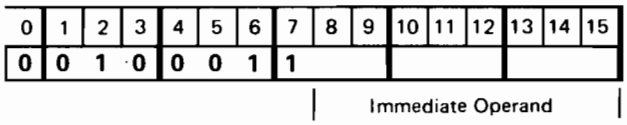
COND	LUTA	LABEL	RAR	W
SR<2	364	LDXA	0714	1
SR>=2	324	LDX1	0713	1

LDXB



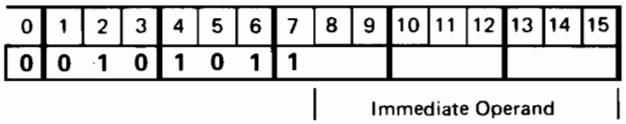
COND	LUTA	LABEL	RAR	W
SR<2	362	SRP	0004	1
SR>=2	322	LDXB	0715	1

LDXI



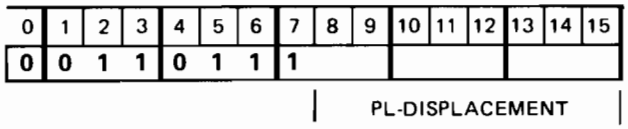
COND	LUTA	LABEL	RAR	W
UNC.	114	XPNI	1527	0

LDXN



COND	LUTA	LABEL	RAR	W
UNC.	154	XPNI	1527	1

LLBL

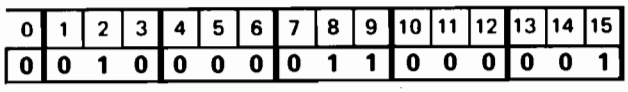


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	136	LLBL	3407	1

2184-69

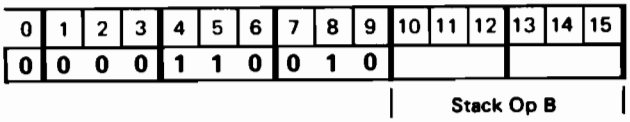
Figure 4-1. Instruction to Microprogram Index (Sheet 19 of 34)

LLSH



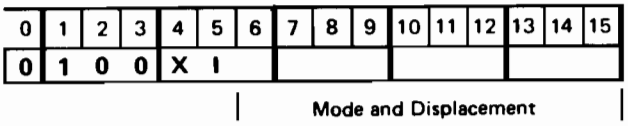
COND	LUTA	LABEL	RAR	W
UNC.	161	RSW	0351	0

LMPY



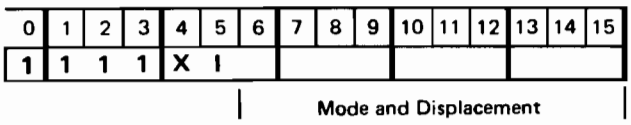
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	262	LMPY	0511	1

LOAD



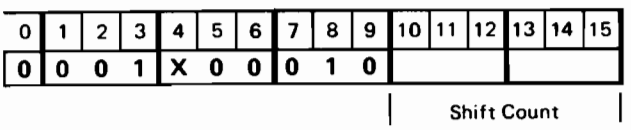
COND	LUTA	LABEL	RAR	W
P REL.	023	AC1P	0027	1
DB REL.	020	AC1D	0013	1
Q REL.	020	AC1D	0013	1
S REL.	021	AC1S	0012	1
JLUI	022	LOAD	0122	1

LRA



COND	LUTA	LABEL	RAR	W
P REL.	077	AC2P	0055	1
DB REL.	074	AC2D	0044	1
Q REL.	074	AC2D	0044	1
S REL.	075	AC2S	0043	1
JLUI	076	LRA	0124	1

LSL

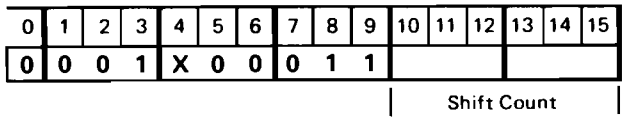


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	202	SHFL	0732	0

2184-70

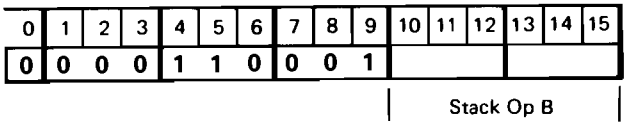
Figure 4-1. Instruction to Microprogram Index (Sheet 20 of 34)

LSR



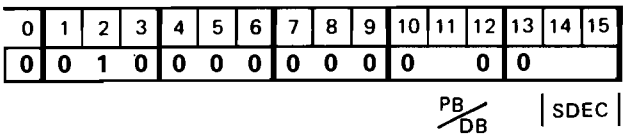
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	203	SHFR	0724	0

LSUB



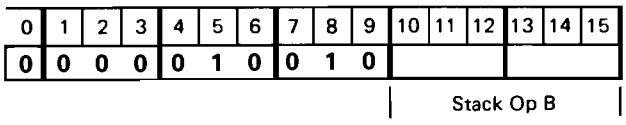
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	261	LSUB	0505	1

MOVE



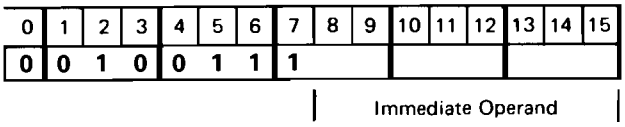
COND	LUTA	LABEL	RAR	W
P REL.	101	PROG	2027	0
DB REL.	105	DATA	2030	0

MPY



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	242	MPY	1705	1

MPYI

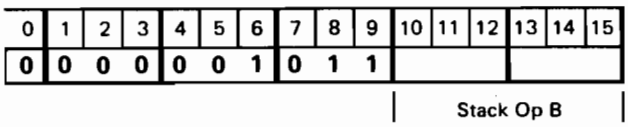


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	134	MPYI	1703	0

2184-71

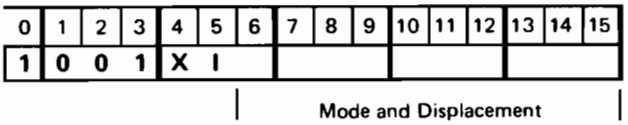
Figure 4-1. Instruction to Microprogram Index (Sheet 21 of 34)

MPYL



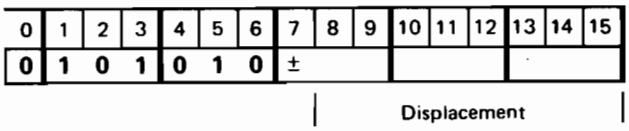
COND	LUTA	LABEL	RAR	W
SR<4	313	SRP	0004	1
SR=4	353	MPYL	1706	1

MPYM



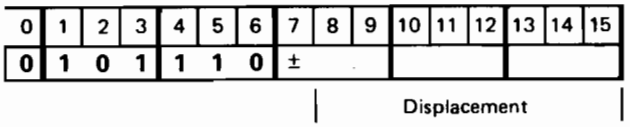
COND	LUTA	LABEL	RAR	W
P REL.	047	AC1P	0027	1
DB REL.	044	AC1D	0013	1
Q REL.	044	AC1D	0013	1
S REL.	045	AC1S	0012	1
JLUI	046	MPYM	1700	1

MTBA



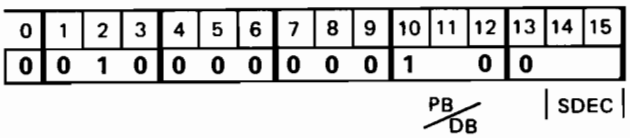
COND	LUTA	LABEL	RAR	W
P REL.	027	LCB1	0416	0

MTBX



COND	LUTA	LABEL	RAR	W
P REL.	027	LCB1	0416	0

MVB

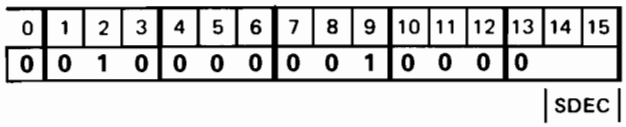


COND	LUTA	LABEL	RAR	W
P REL.	111	PROG	2027	0
DB REL.	115	DATA	2030	0

2184-72

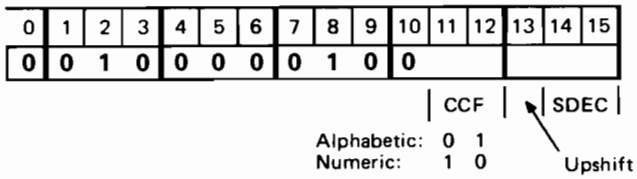
Figure 4-1. Instruction to Microprogram Index (Sheet 22 of 34)

MVBL



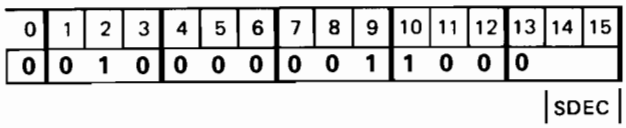
COND	LUTA	LABEL	RAR	W
UNC.	121	MVBL	2000	0

MVBW



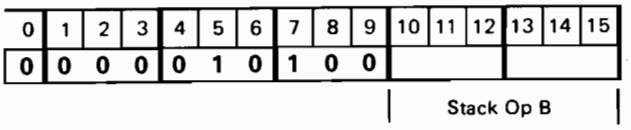
COND	LUTA	LABEL	RAR	W
P REL.	141	SCAM	2126	0
DB REL.	145	SCAM	2126	0

MVLB



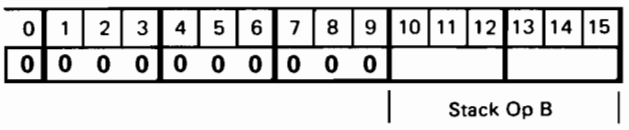
COND	LUTA	LABEL	RAR	W
UNC.	131	MVLB	2003	0

NEG



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	244	NEG	0465	1

NOP

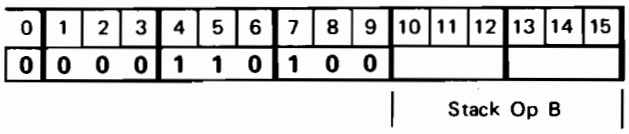


COND	LUTA	LABEL	RAR	W
SR<2	340	NOP	1466	1
SR>=2	300	NOP	1466	1

2184-73

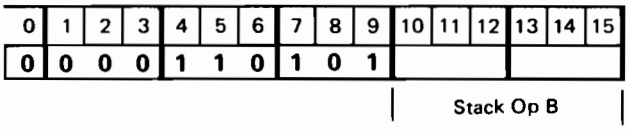
Figure 4-1. Instruction to Microprogram Index (Sheet 23 of 34)

NOT



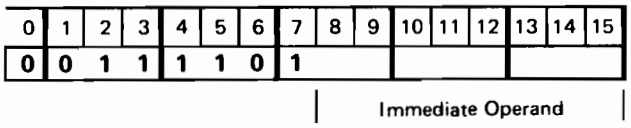
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	264	NOT	0537	1

OR



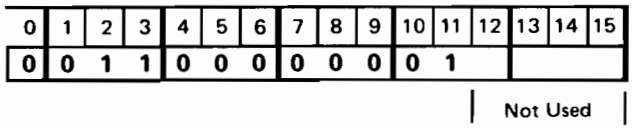
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	265	OR	0646	1

ORI



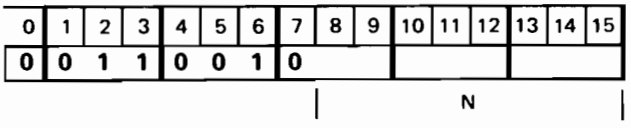
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	166	ORI	1524	0

PAUS



COND	LUTA	LABEL	RAR	W
UNC.	107	PAUS	1541	1

PCAL



COND	LUTA	LABEL	RAR	W
UNC.	112	PCAL	2546	1

2184-74

Figure 4-1. Instruction to Microprogram Index (Sheet 24 of 34)

PLDA

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

COND	LUTA	LABEL	RAR	W
UNC.	165	PLDA	0753	0

PSHR

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	0	1	0	0	1	0							
										DB	DL	Z	Sta	X	Q	S

COND	LUTA	LABEL	RAR	W
UNC.	144	PSHR	1557	0

PSTA

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

COND	LUTA	LABEL	RAR	W
UNC.	165	PLDA	0753	0

RIO

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

COND	LUTA	LABEL	RAR	W
UNC.	143	RIO	2252	0

RMSK

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	1	0	0	0	0	0	1	0	1				
													Not Used		

COND	LUTA	LABEL	RAR	W
UNC.	127	RMSK	2330	1

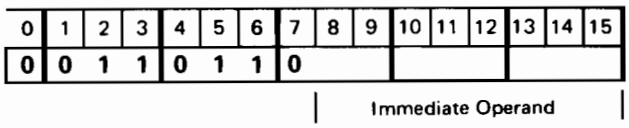
2184-75

Figure 4-1. Instruction to Microprogram Index (Sheet 25 of 34)

RSW

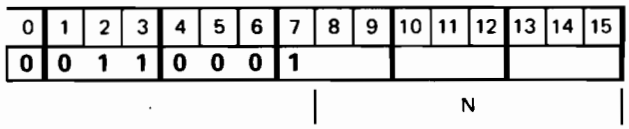
COND	LUTA	LABEL	RAR	W
UNC.	161	RSW	0351	0

SBXI



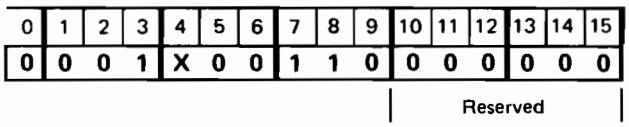
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	130	ASI	1523	1

SCAL



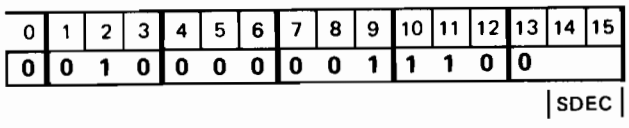
COND	LUTA	LABEL	RAR	W
UNC.	106	SCAL	2545	1

SCAN



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	206	SCAN	0740	1

SCU

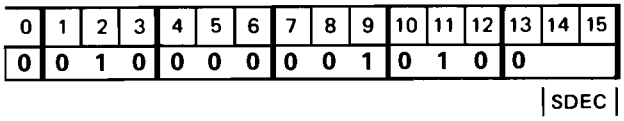


COND	LUTA	LABEL	RAR	W
UNC.	135	SCAM	2126	0

2184-76

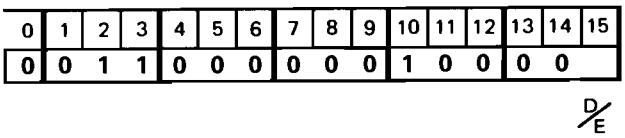
Figure 4-1. Instruction to Microprogram Index (Sheet 26 of 34)

SCW



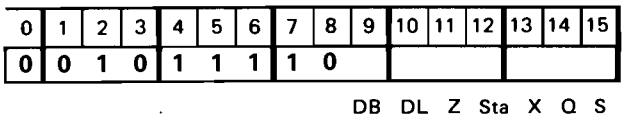
COND	LUTA	LABEL	RAR	W
UNC.	125	SCAM	2126	0

SED



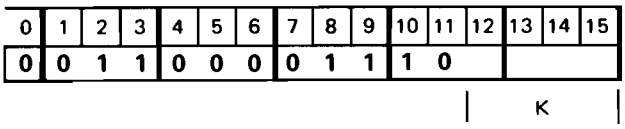
COND	LUTA	LABEL	RAR	W
UNC.	113	SED	2315	0

SETR



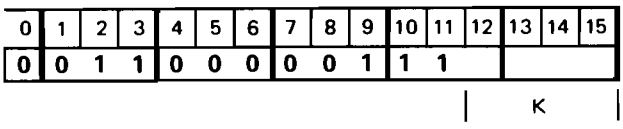
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	174	SETR	1605	0

SIN



COND	LUTA	LABEL	RAR	W
UNC.	173	SIN	2323	0

SIO

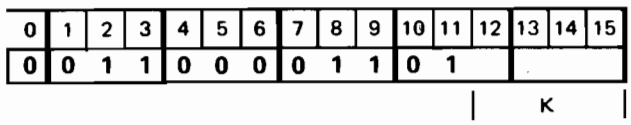


COND	LUTA	LABEL	RAR	W
UNC.	137	SIO	2233	0

2184-77

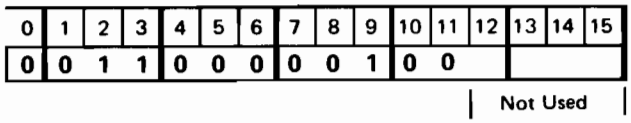
Figure 4-1. Instruction to Microprogram Index (Sheet 27 of 34)

SIRF



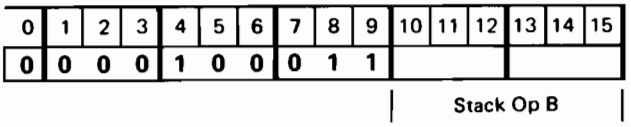
COND	LUTA	LABEL	RAR	W
UNC.	167	SIRF	2343	0

SMSK



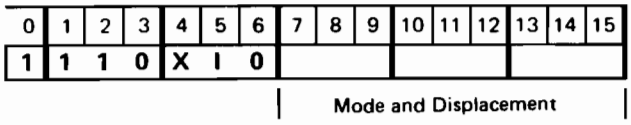
COND	LUTA	LABEL	RAR	W
UNC.	123	SMSK	2332	1

STAX



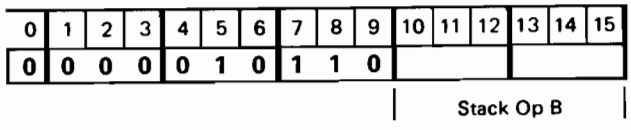
COND	LUTA	LABEL	RAR	W
SR<2	363	SRP	0004	1
SR>=2	323	STAX	0716	1

STB



COND	LUTA	LABEL	RAR	W
DB REL.	073	LDB	0143	0
Q REL.	073	LDB	0143	0
S REL.	073	LDB	0143	0

STBX

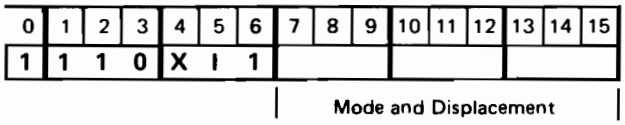


COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	246	STBX	0717	1

2184-78

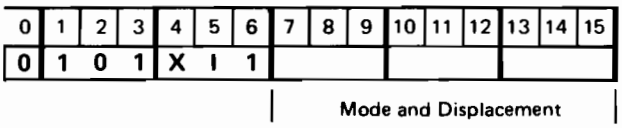
Figure 4-1. Instruction to Microprogram Index (Sheet 28 of 34)

STD



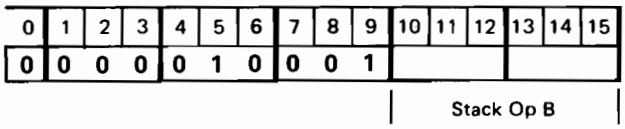
COND	LUTA	LABEL	RAR	W
DB REL.	070	AC4D	0102	1
Q REL.	070	AC4D	0102	1
S REL.	071	AC4S	0101	1
JLUI	072	STD	0225	1

STOR



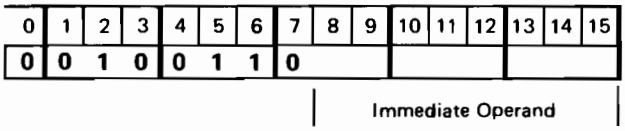
COND	LUTA	LABEL	RAR	W
DB REL.	024	AC1D	0013	1
Q REL.	024	AC1D	0013	1
S REL.	025	AC1S	0012	1
JLUI	026	STOR	0220	1

SUB



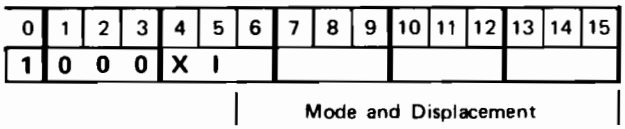
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	241	SUB	0464	1

SUBI



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	130	ASI	1523	1

SUBM

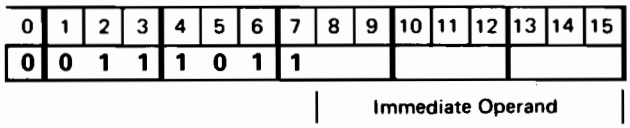


COND	LUTA	LABEL	RAR	W
P REL.	043	AC1P	0027	1
DB REL.	040	AC1D	0013	1
Q REL.	040	AC1D	0013	1
S REL.	041	AC1S	0012	1
JLUI	042	SUBM	0257	1

2184-79

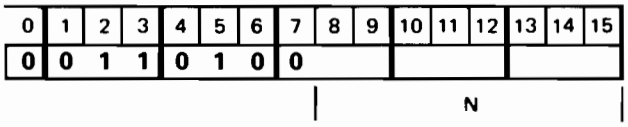
Figure 4-1. Instruction to Microprogram Index (Sheet 29 of 34)

SUBS



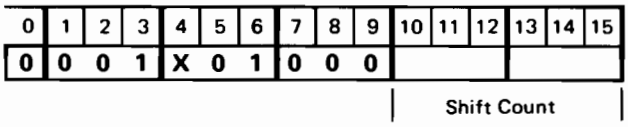
COND	LUTA	LABEL	RAR	W
UNC.	156	SUBS	1660	1

SXIT



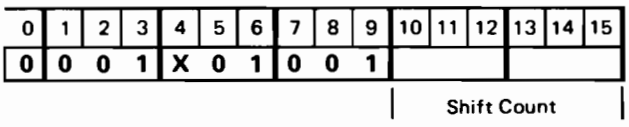
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	122	SXIT	1531	1

TASL



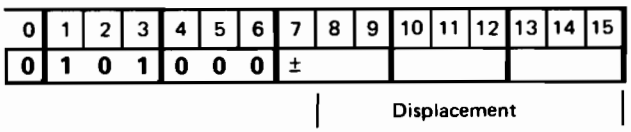
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	210	TASL	1417	0

TASR



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	211	TASR	1432	0

TBA

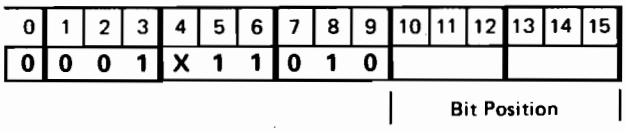


COND	LUTA	LABEL	RAR	W
P REL.	027	LCB1	0416	0

2184-80

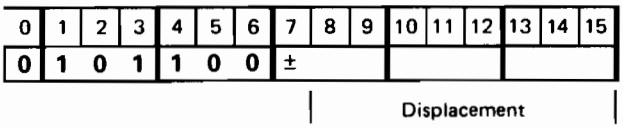
Figure 4-1. Instruction to Microprogram Index (Sheet 30 of 34)

TBC



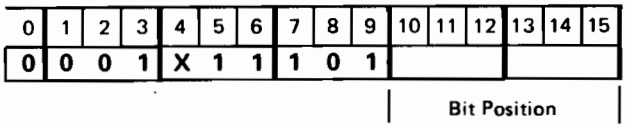
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	232	TBC	1465	0

TBX



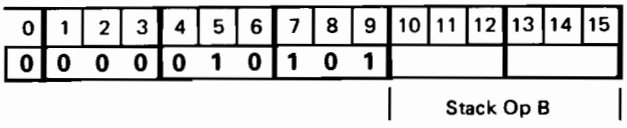
COND	LUTA	LABEL	RAR	W
P REL.	027	LCB1	0416	0

TCBC



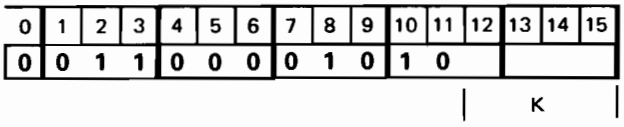
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	235	TCBC	1473	0

TEST



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	245	TEST	0632	1

TIO

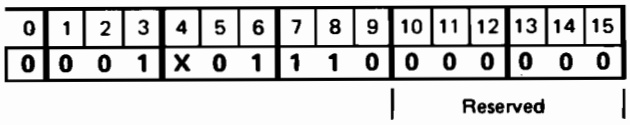


COND	LUTA	LABEL	RAR	W
UNC.	153	TIO	2300	0

2184-81

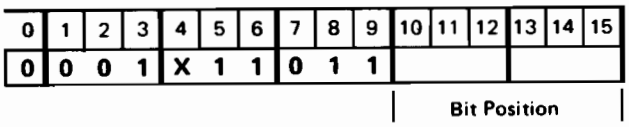
Figure 4-1. Instruction to Microprogram Index (Sheet 31 of 34)

TNSL



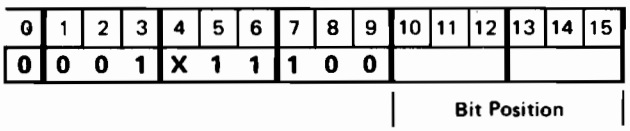
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	216	TNSL	1342	1

TRBC



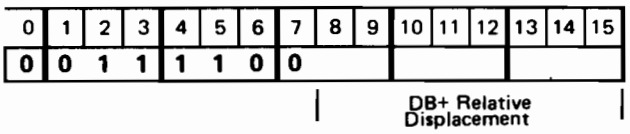
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	233	TRBC	1467	0

TSBC



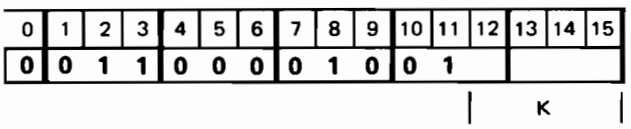
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	234	TSBC	1471	0

TSBM



COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	162	TSBM	1443	0

WIO

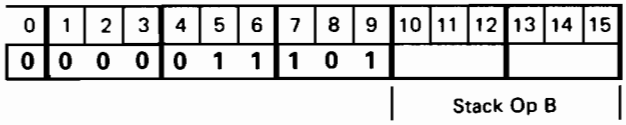


COND	LUTA	LABEL	RAR	W
UNC.	147	WIO	2265	0

2184-82

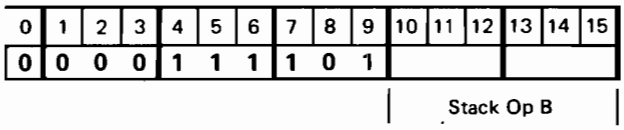
Figure 4-1. Instruction to Microprogram Index (Sheet 32 of 34)

XAX



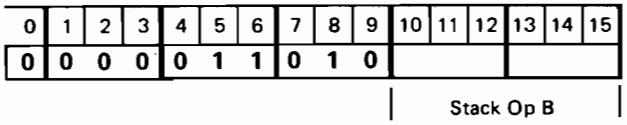
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	255	XAX	0704	1

XBX



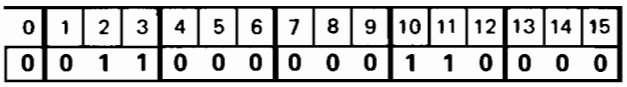
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	275	XBX	0706	1

XCH



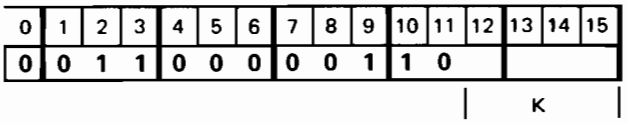
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	252	XCH	0702	1

XCHD



COND	LUTA	LABEL	RAR	W
UNC.	117	XCHD	1674	1

XEQ

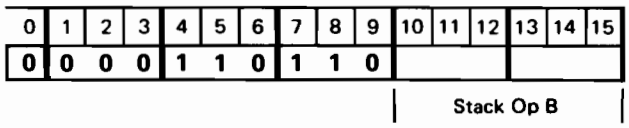


COND	LUTA	LABEL	RAR	W
UNC.	133	XEQ	1545	0

2184-83

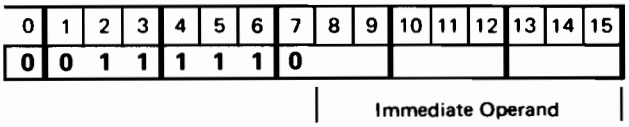
Figure 4-1. Instruction to Microprogram Index (Sheet 33 of 34)

XOR



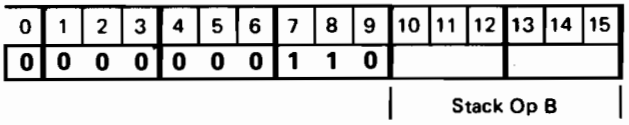
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	266	XOR	0647	1

XORI



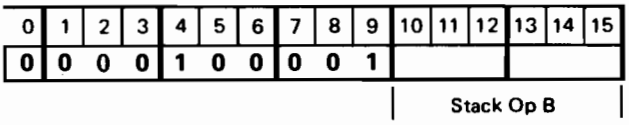
COND	LUTA	LABEL	RAR	W
SR<2	---	SRP	0004	-
SR>=2	172	XORI	1525	0

ZERO



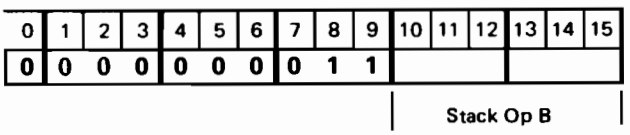
COND	LUTA	LABEL	RAR	W
SR<2	346	ZERO	0664	1
SR>=2	306	ZRO1	0663	1

ZROB



COND	LUTA	LABEL	RAR	W
SR<2	361	SRP	0004	1
SR>=2	321	ZROB	0666	1

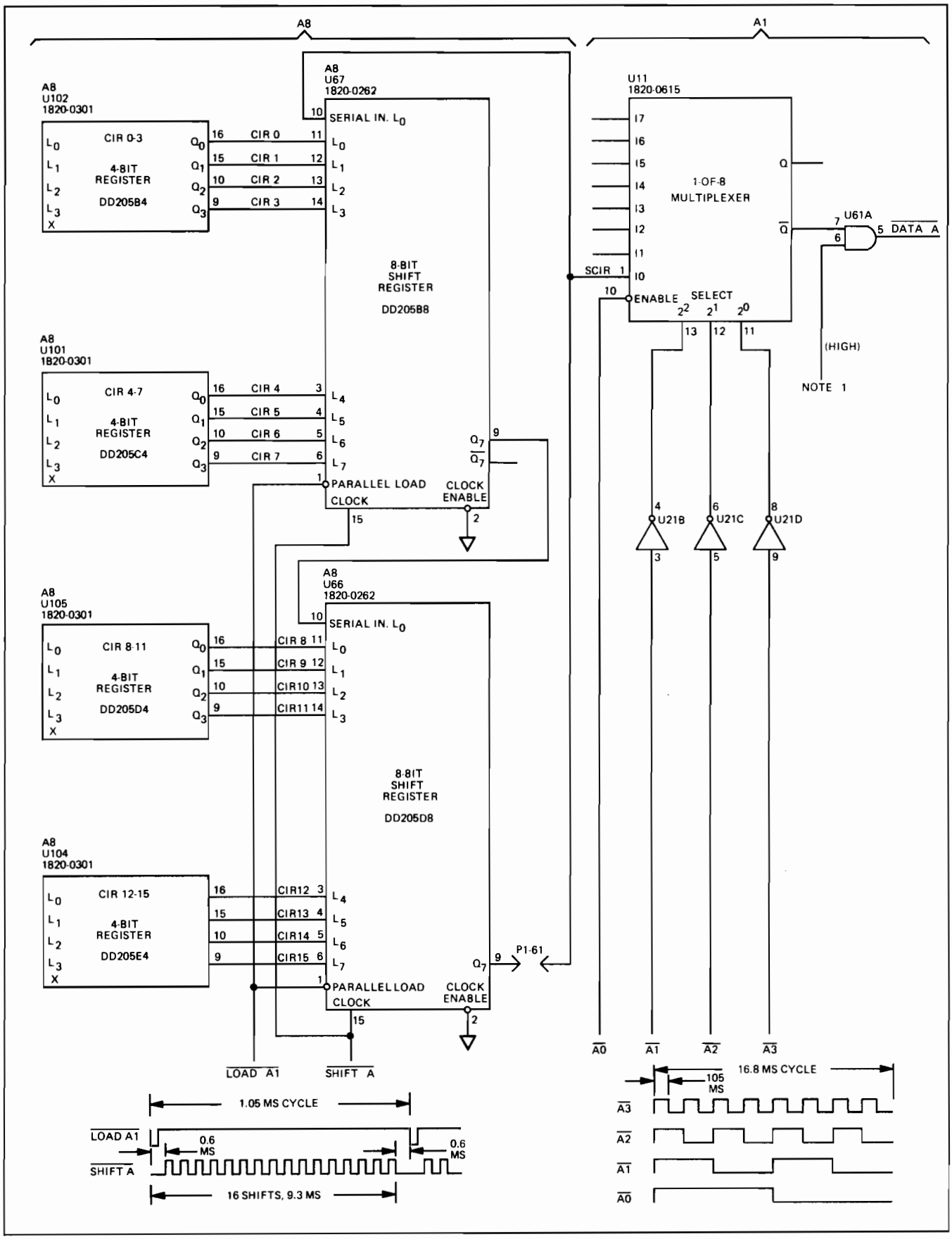
ZROX



COND	LUTA	LABEL	RAR	W
SR<2	343	ZROX	0665	1
SR>=2	303	ZROX	0665	1

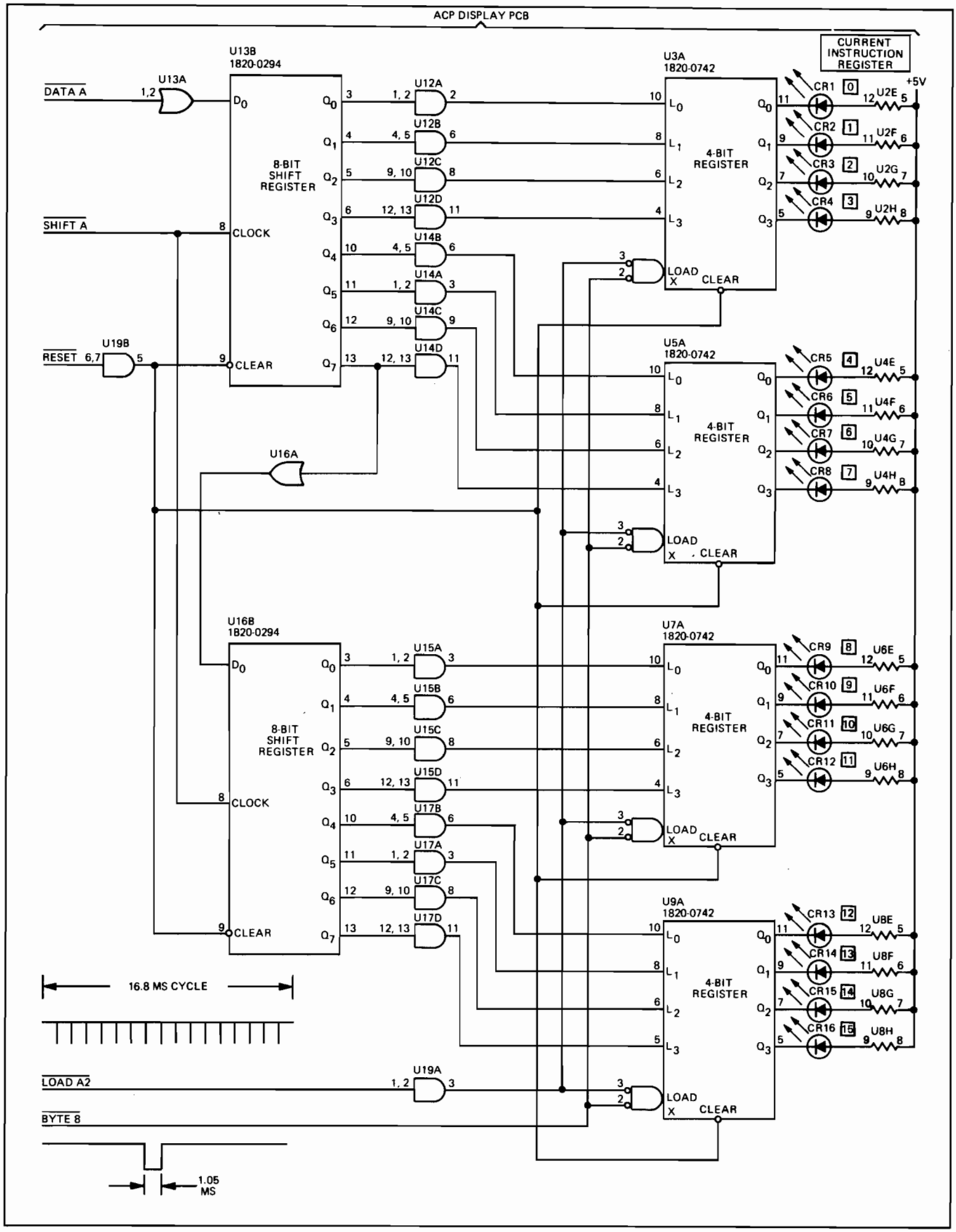
2184-84

Figure 4-1. Instruction to Microprogram Index (Sheet 34 of 34)



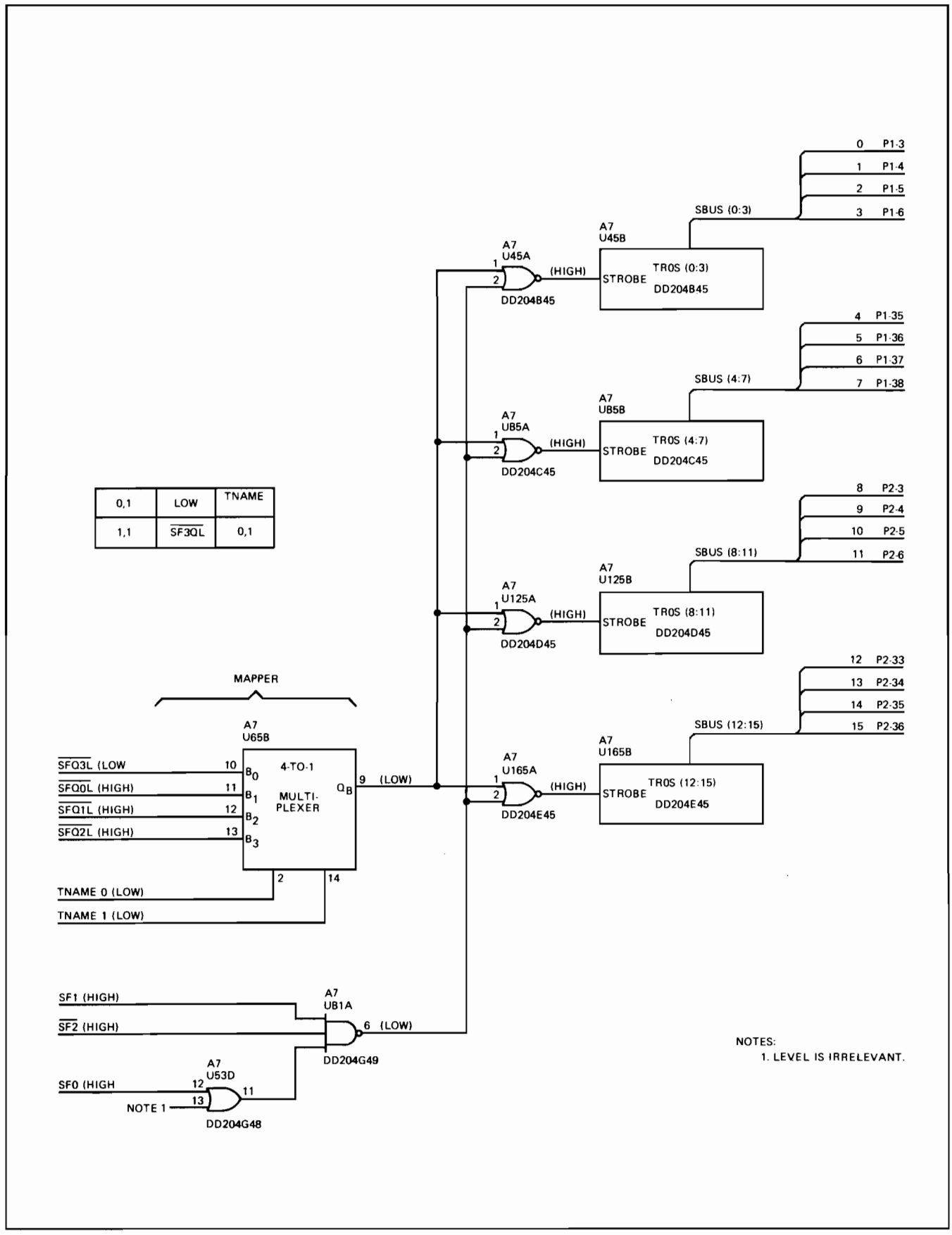
2184-138

Figure 4-2. Current Instruction Register Servicing Diagram (Sheet 1 of 2)



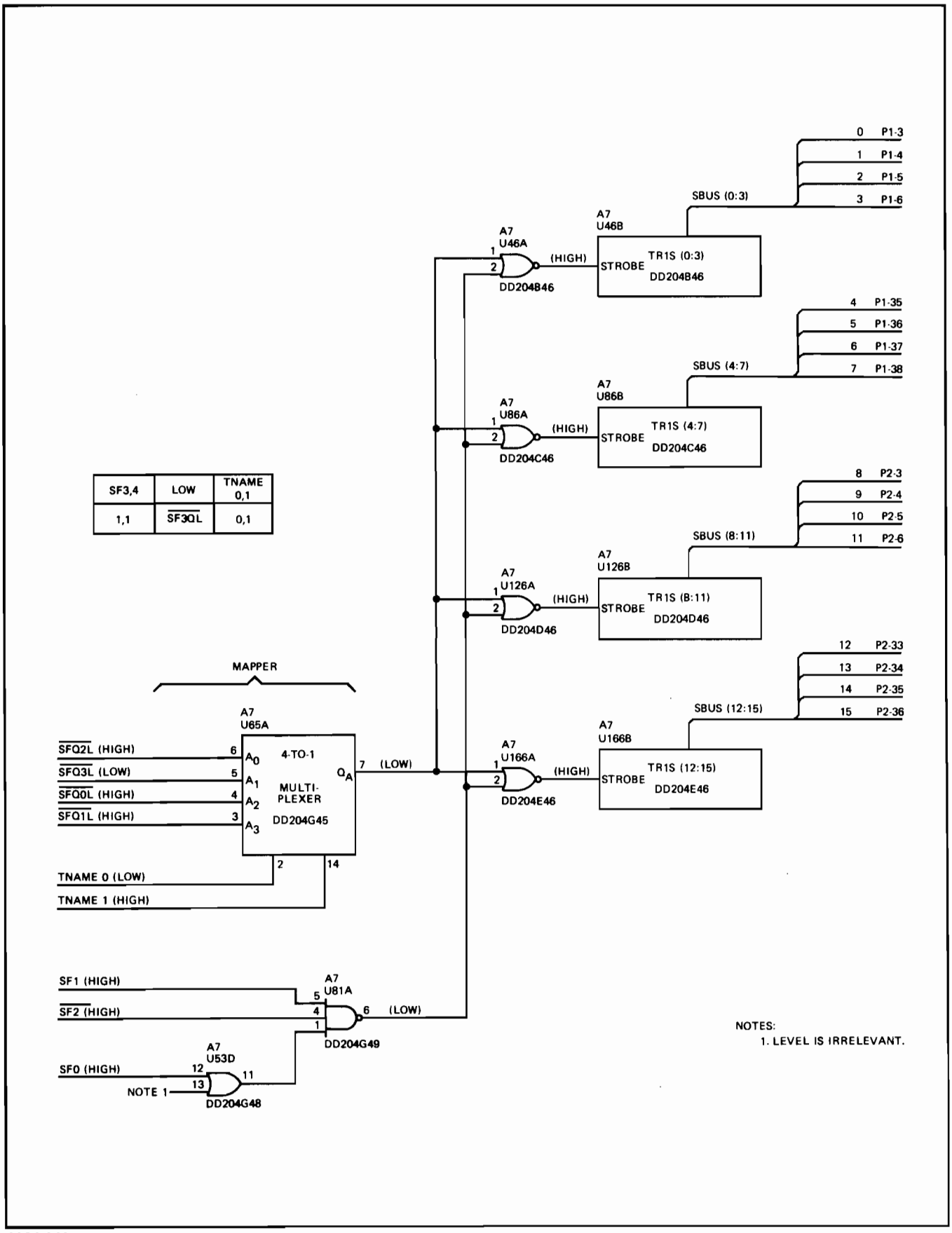
2184-139

Figure 4-2. Current Instruction Register Servicing Diagram (Sheet 2 of 2)



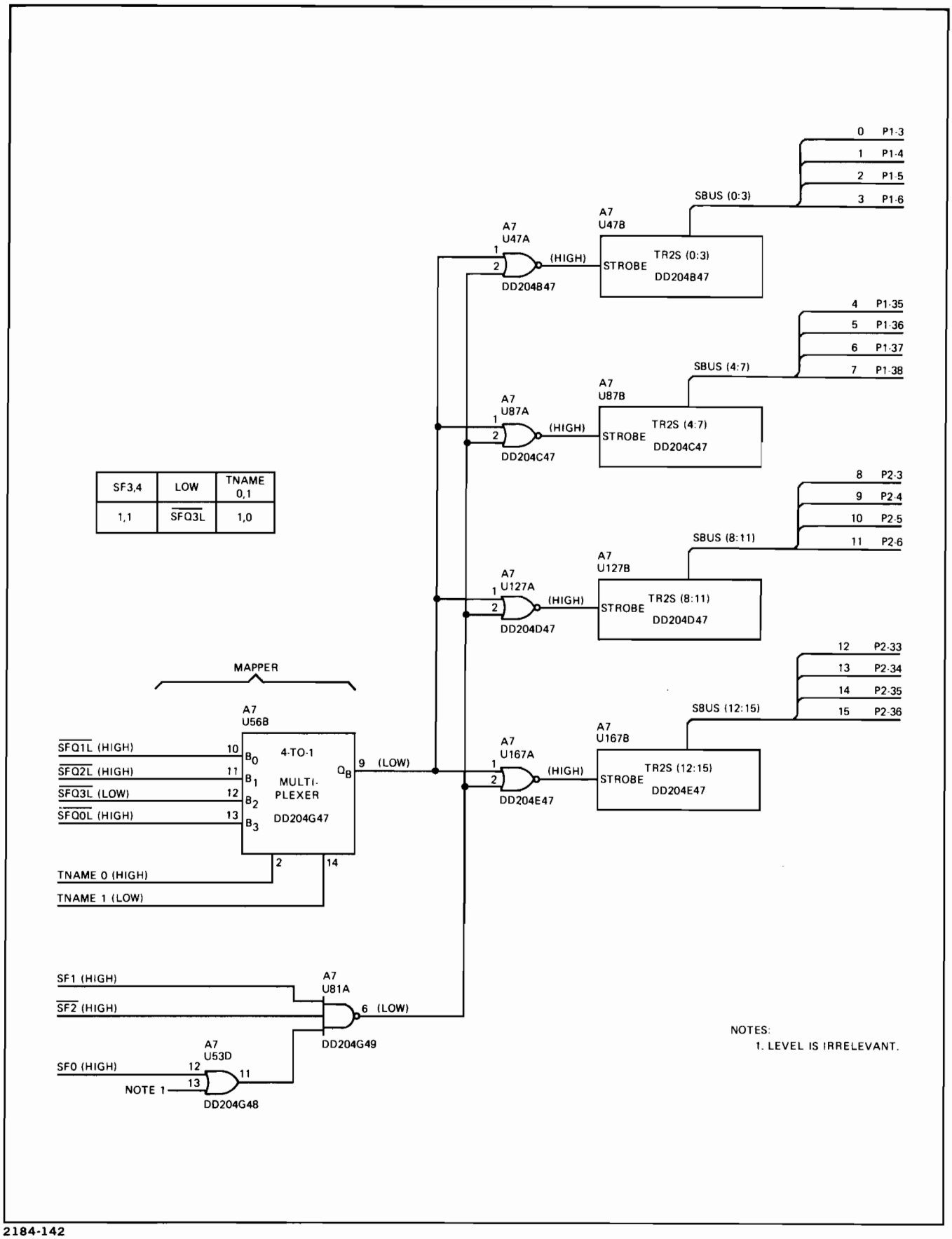
2184-140

Figure 4-3. RA (TROS) Register Servicing Diagram



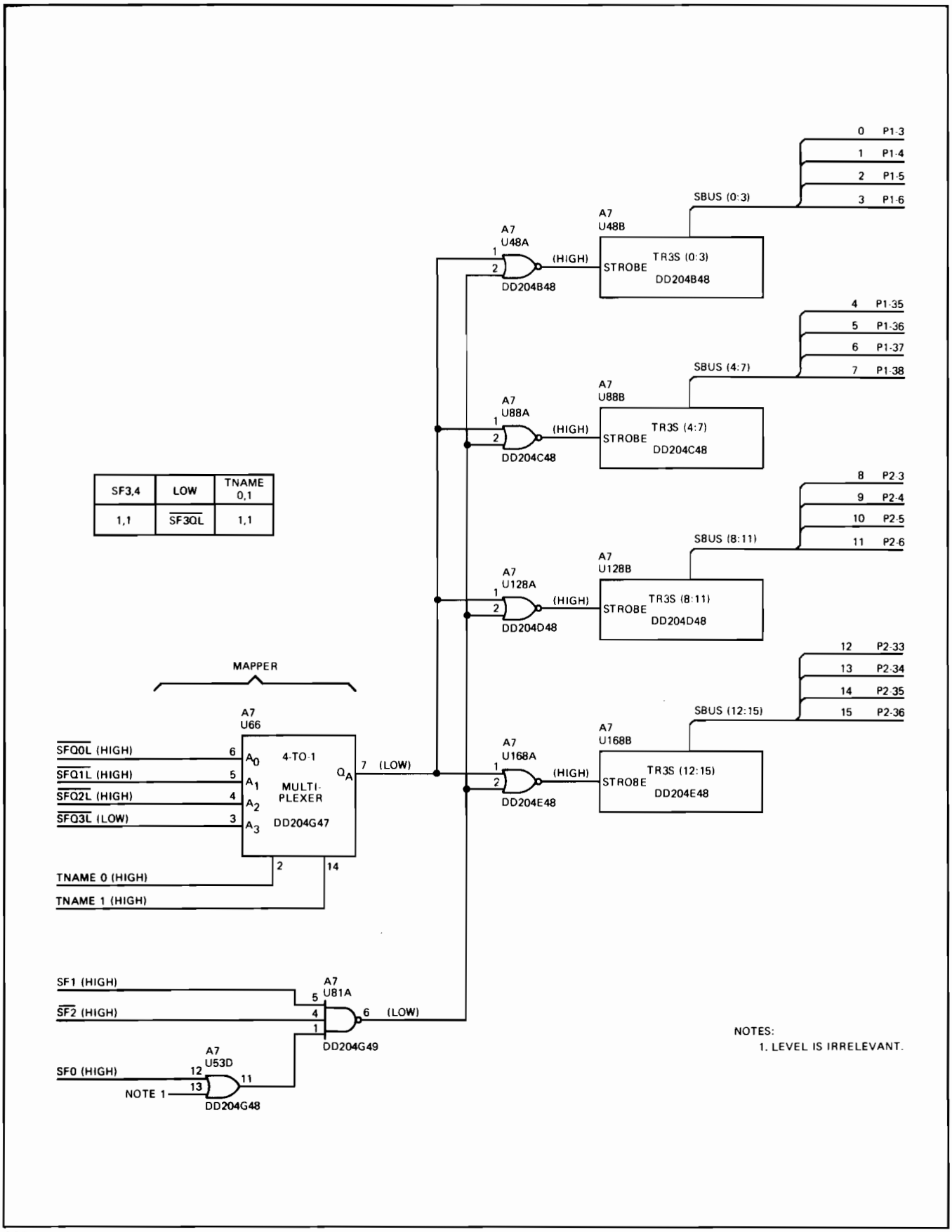
2184-141

Figure 4-4. RA (TRIS) Register Servicing Diagram



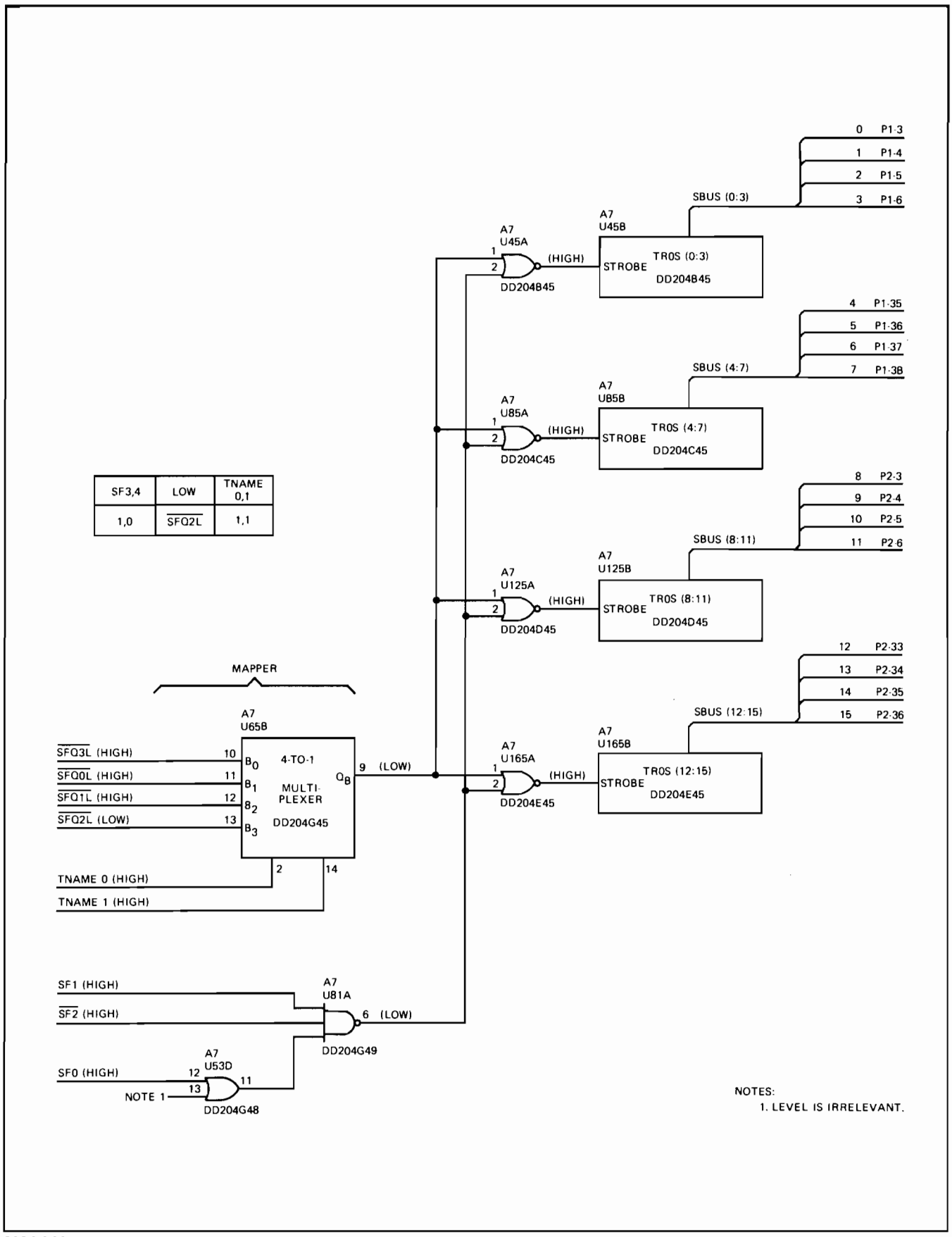
2184-142

Figure 4-5. RA (TR2S) Register Servicing Diagram



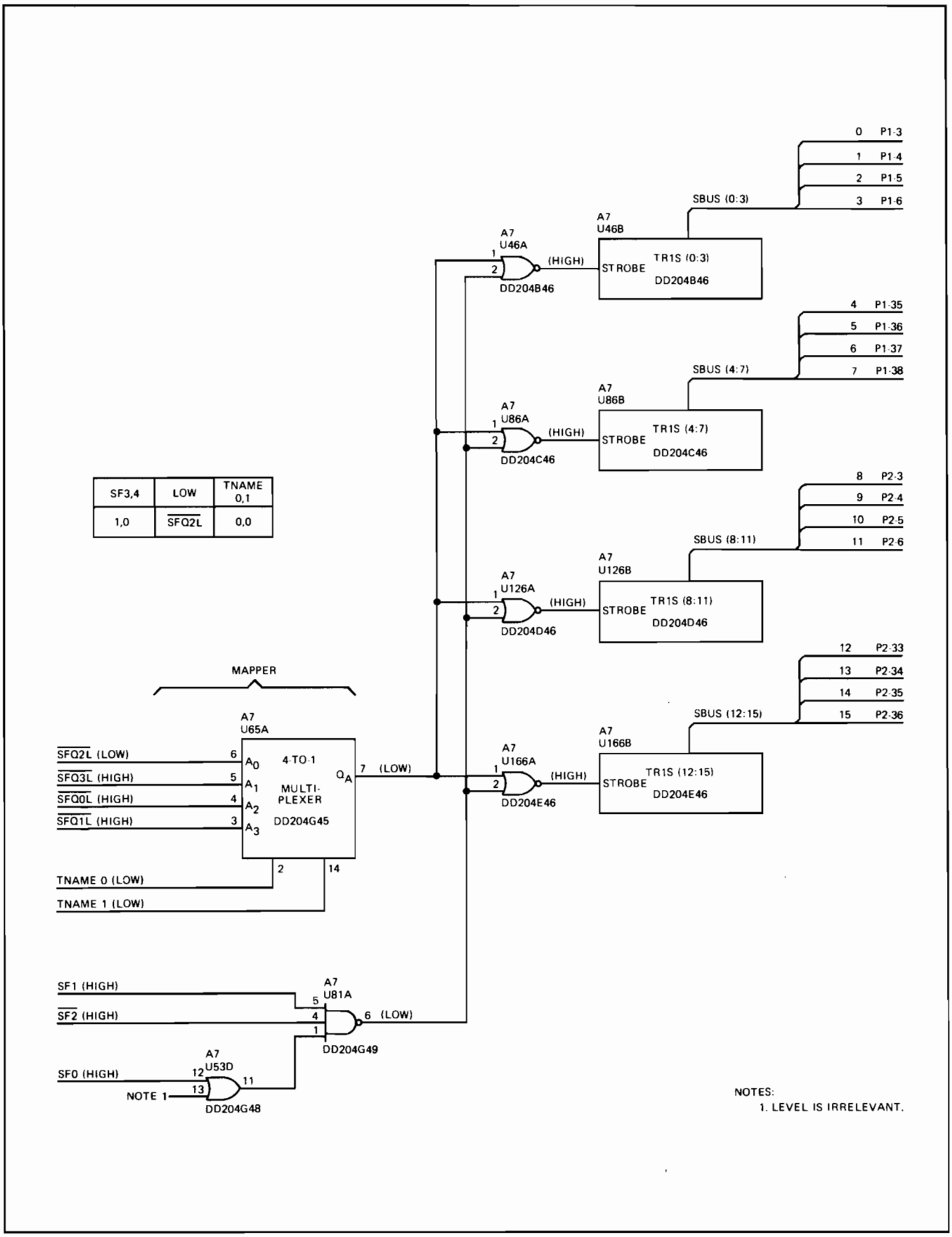
2184-143

Figure 4-6. RA (TR3S) Register Servicing Diagram



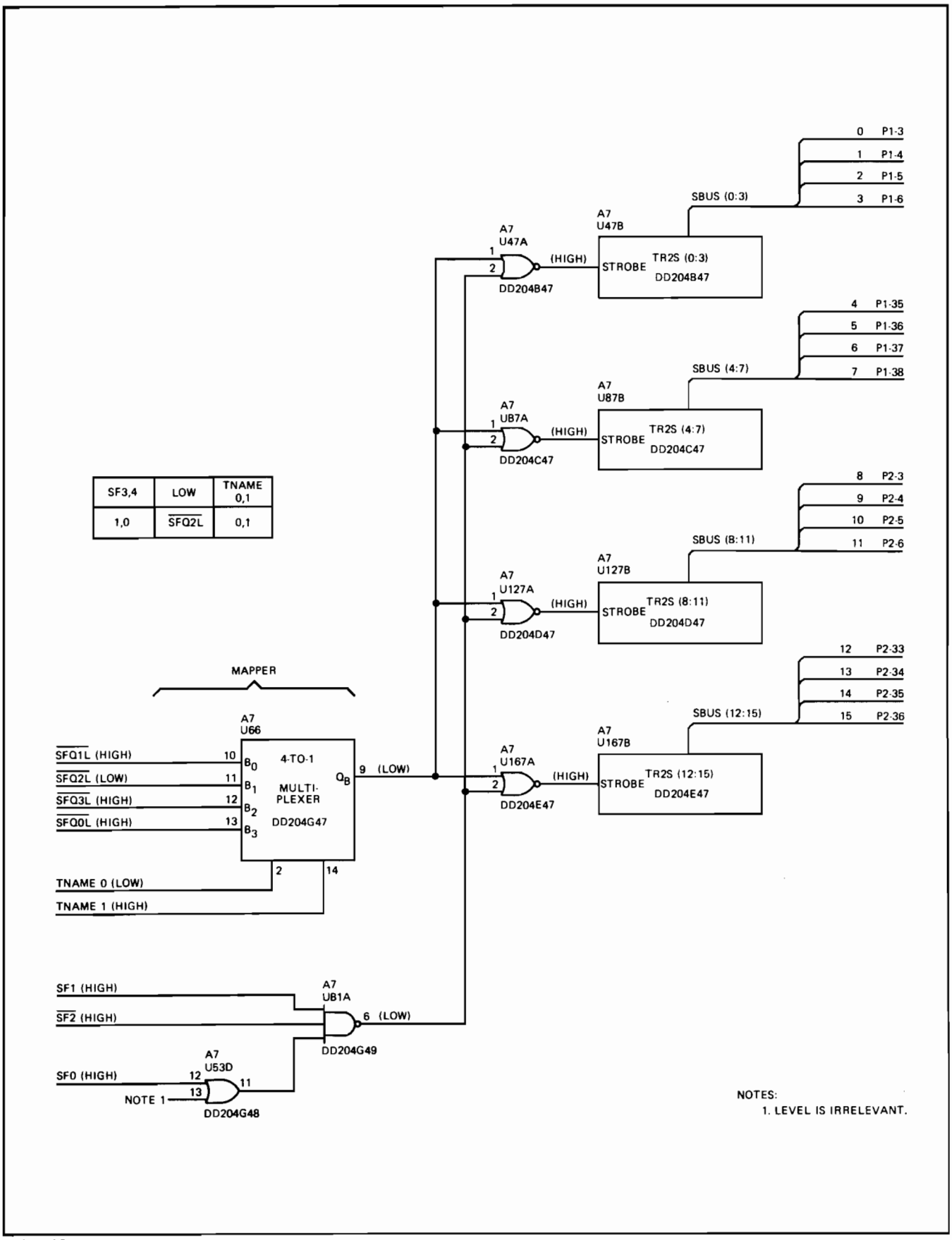
2184-144

Figure 4-7. RB (TROS) Register Servicing Diagram



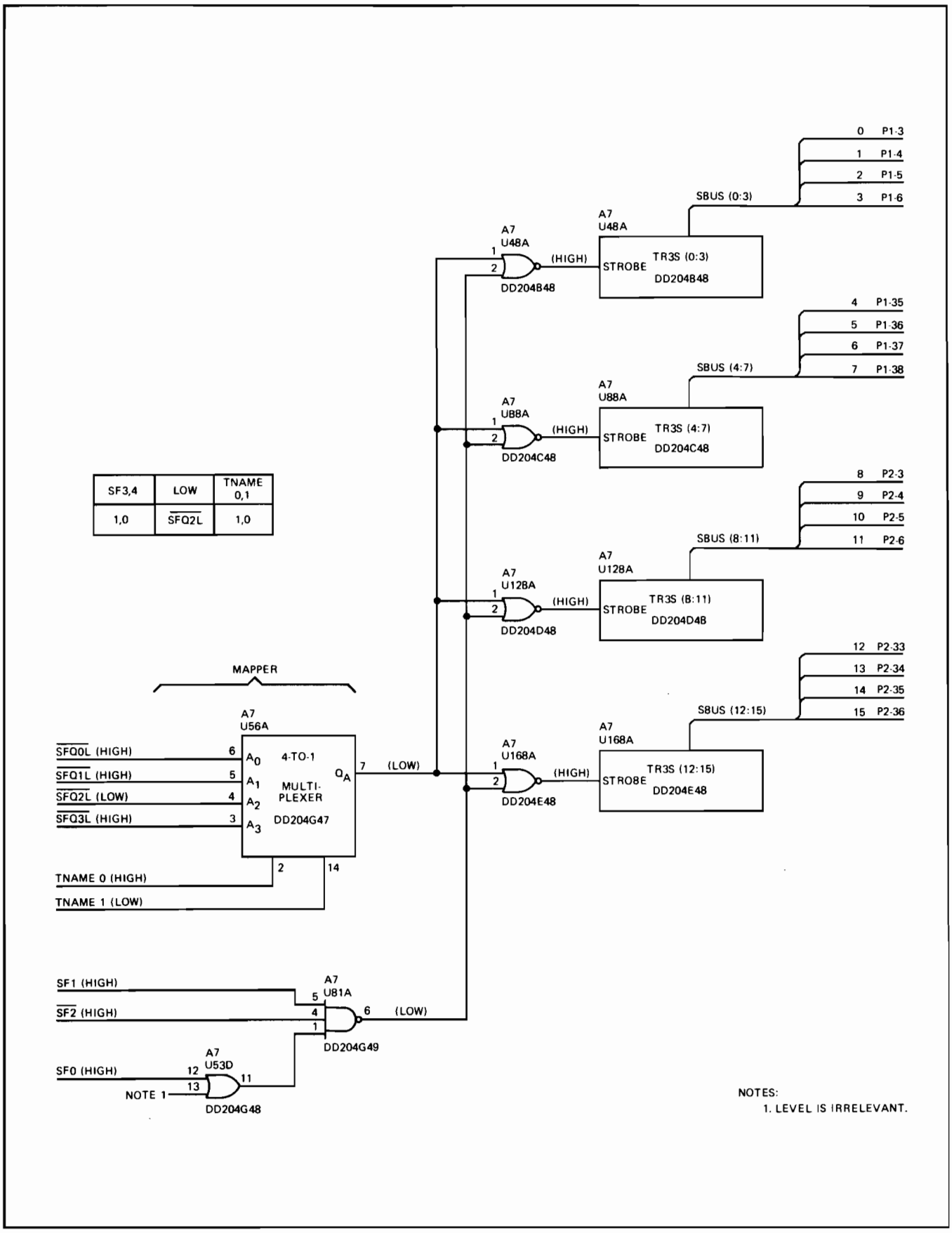
2184-145

Figure 4-8. RB (TR1S) Register Servicing Diagram



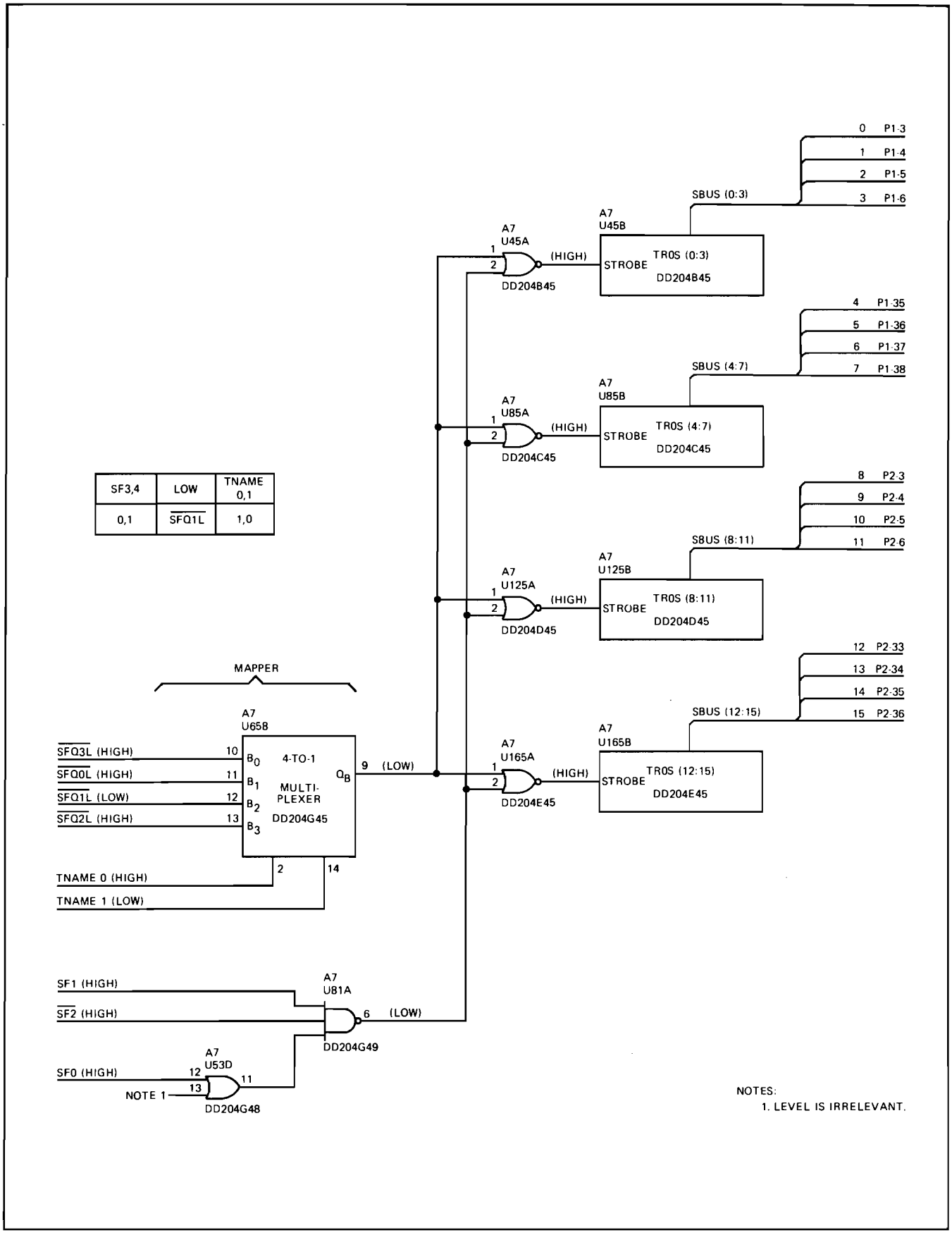
2184-146

Figure 4-9. RB (TR2S) Register Servicing Diagram



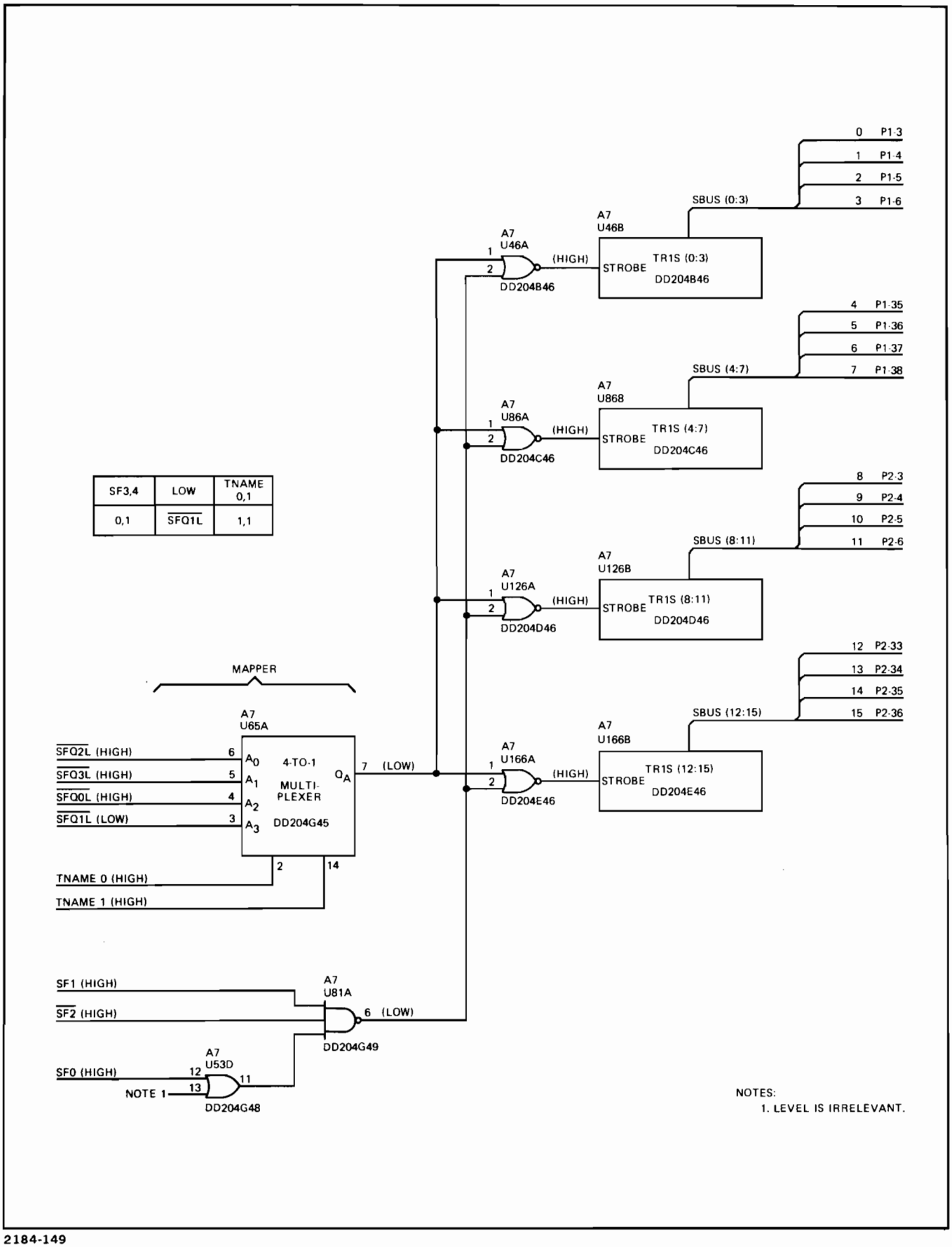
2184-147

Figure 4-10. RB (TR3S) Register Servicing Diagram



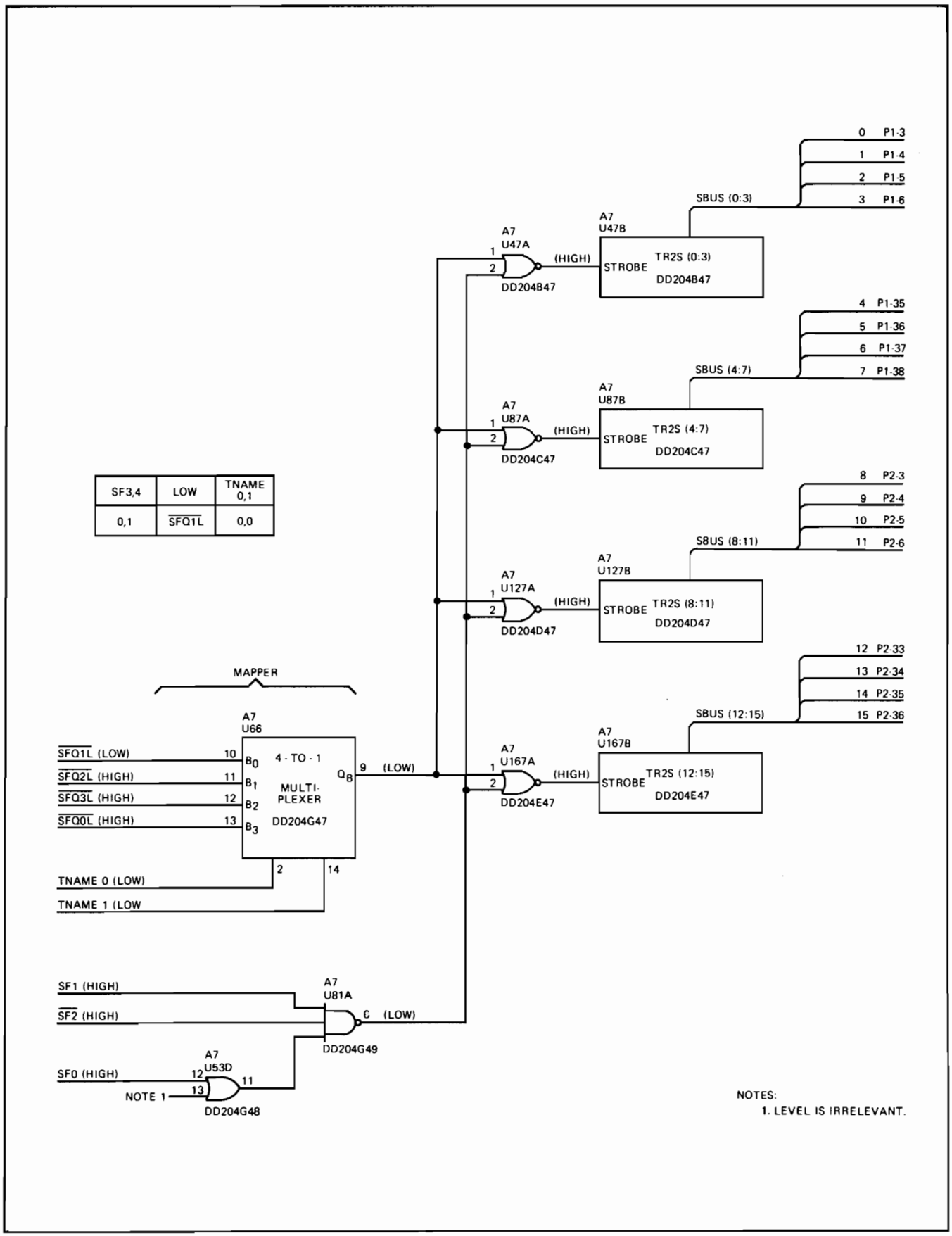
2184-148

Figure 4-11. RC (TROS) Register Servicing Diagram



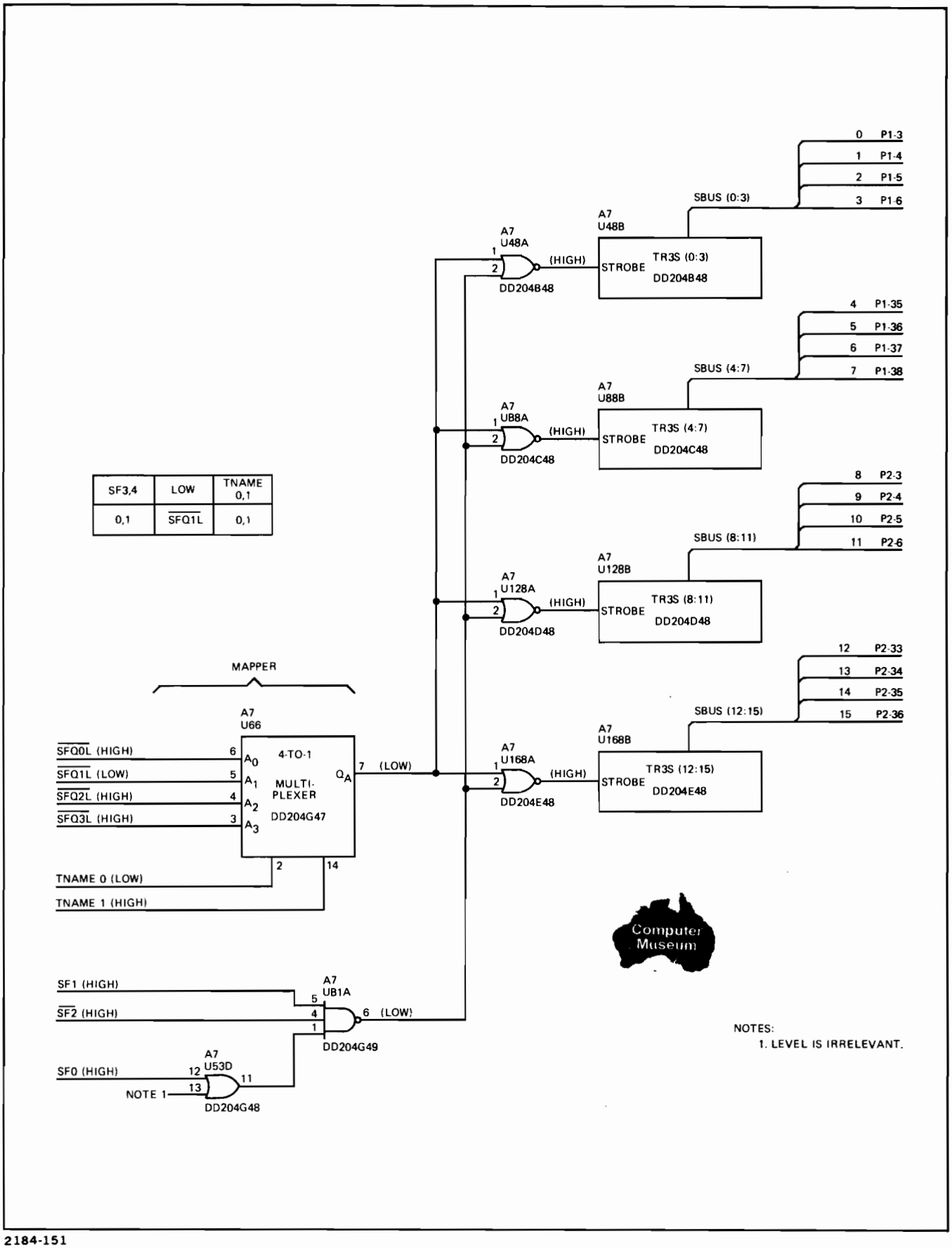
2184-149

Figure 4-12. RC (TR1S) Register Servicing Diagram



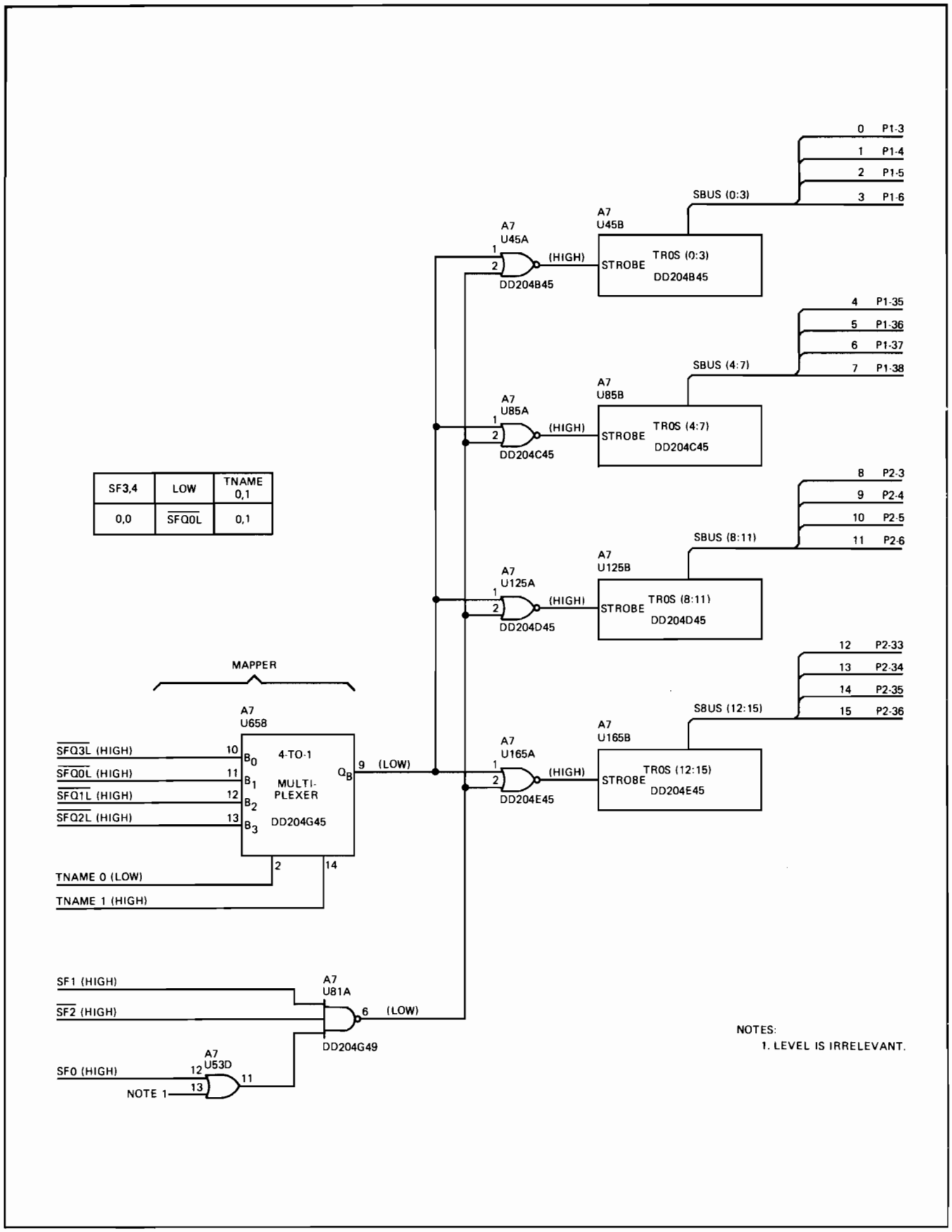
2184-150

Figure 4-13. RC (TR2S) Register Servicing Diagram



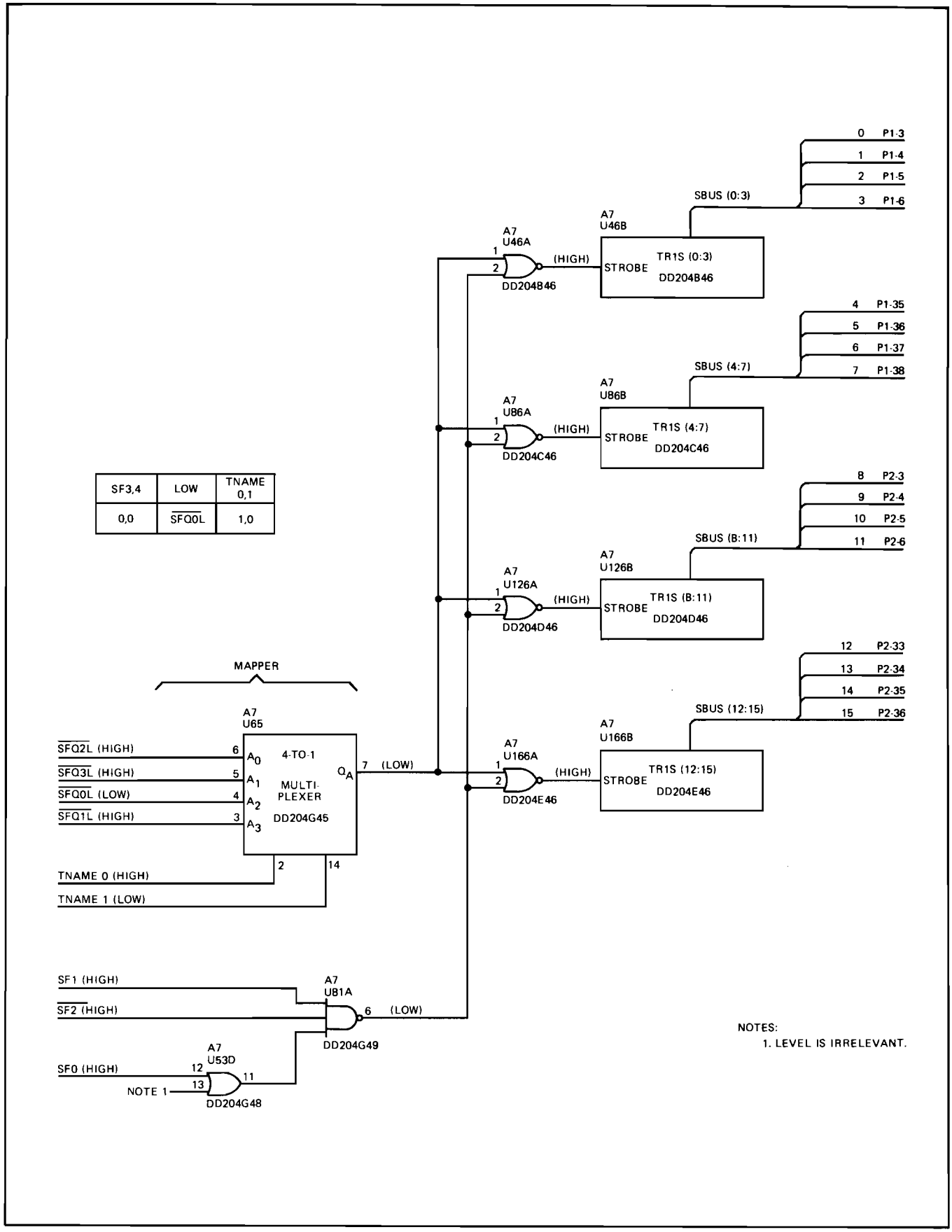
2184-151

Figure 4-14. RC (TR3S) Register Servicing Diagram



2184-152

Figure 4-15. RD (TR0S) Register Servicing Diagram



2184-153

Figure 4-16. RD (TR1S) Register Servicing Diagram

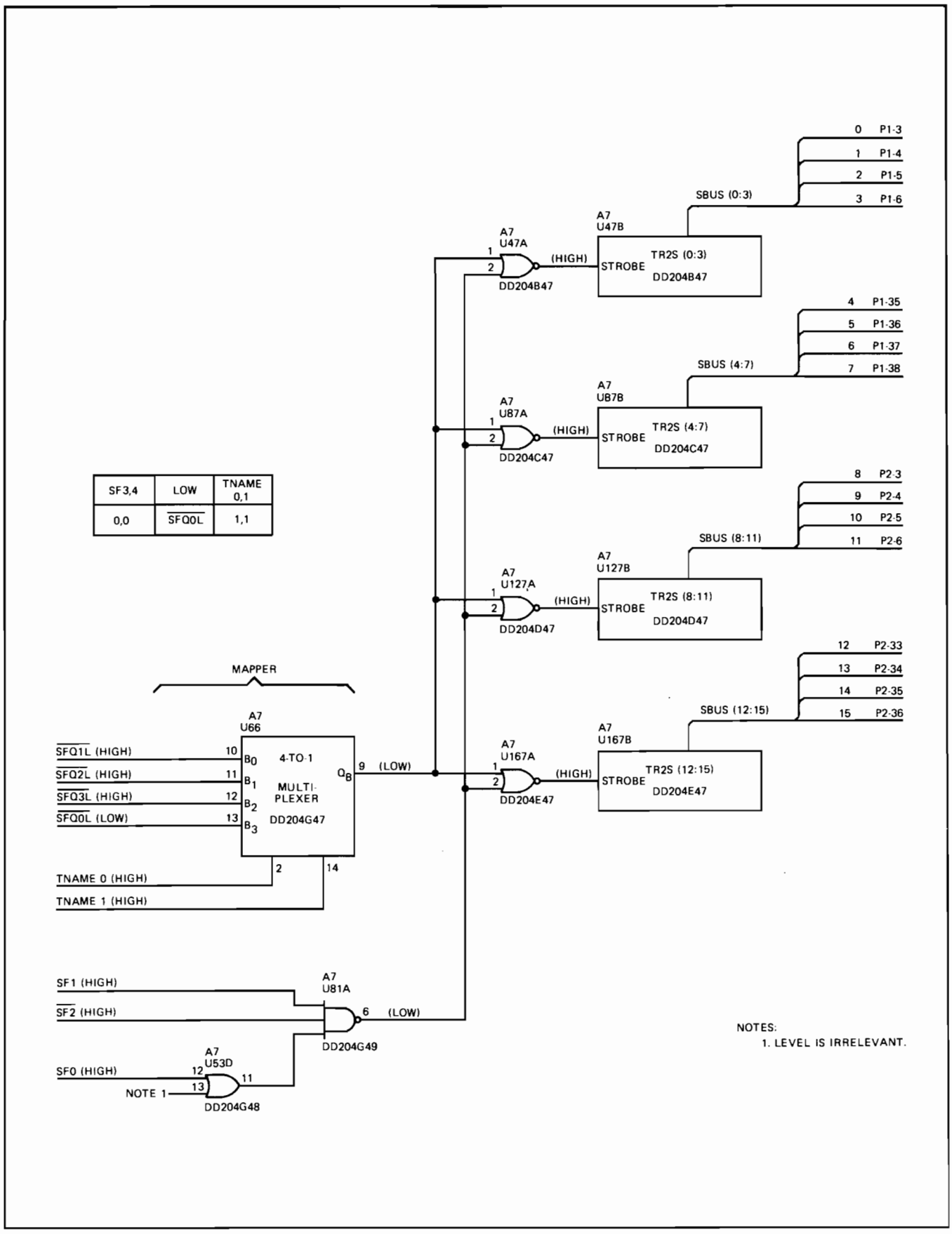
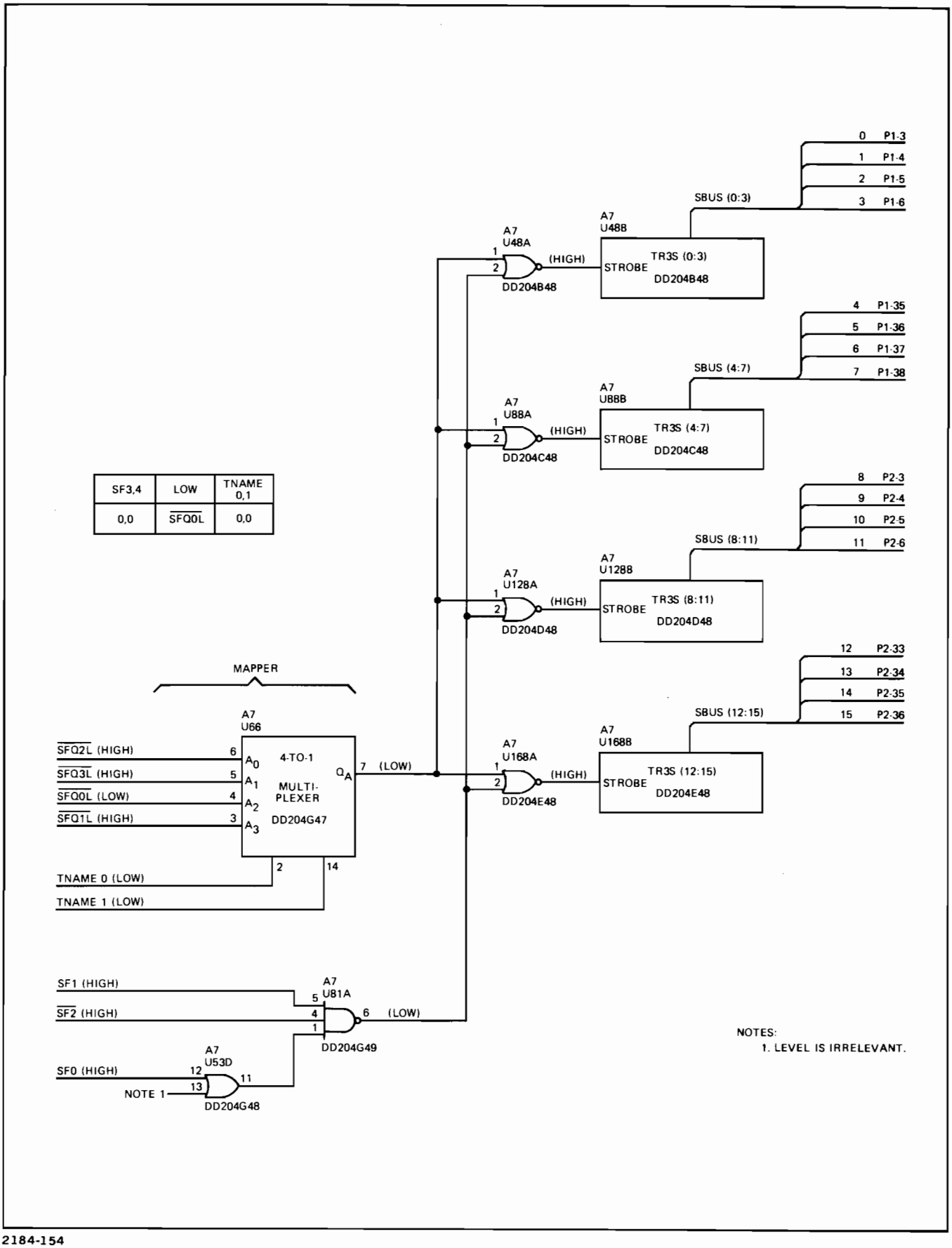
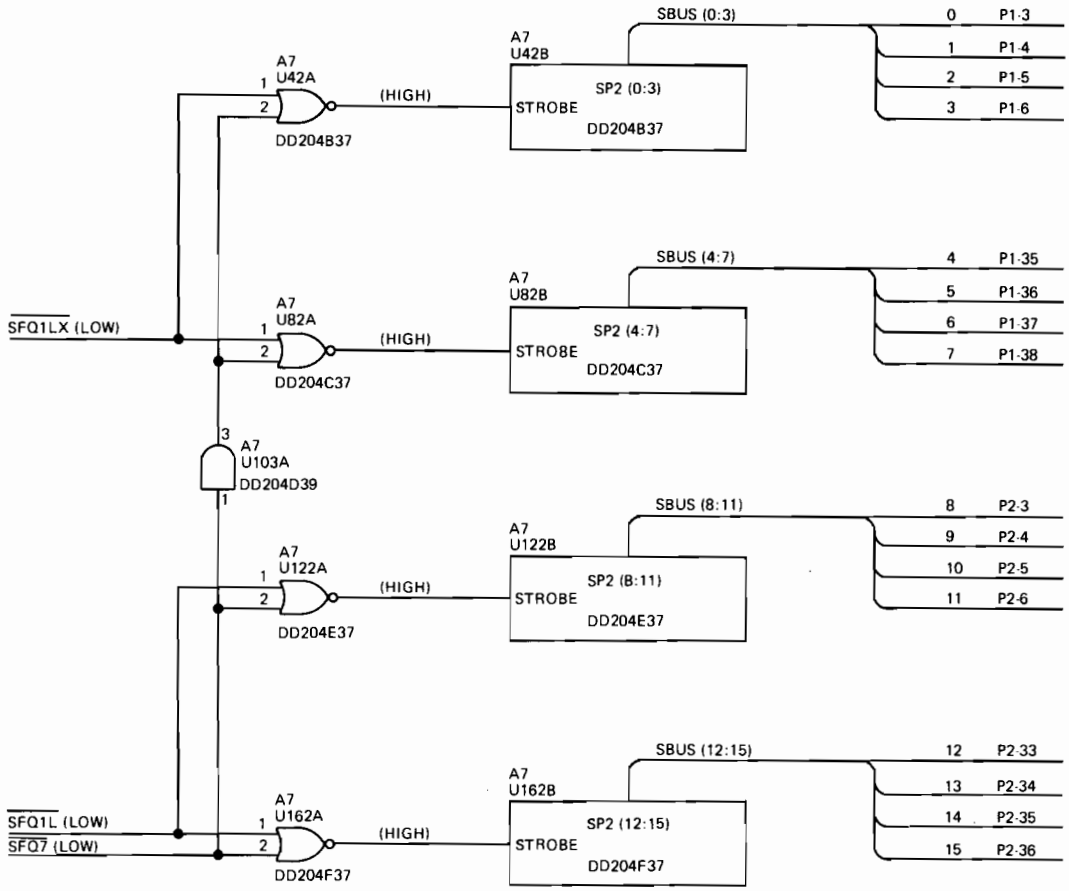


Figure 4-17. RD (TR2S) Register Servicing Diagram



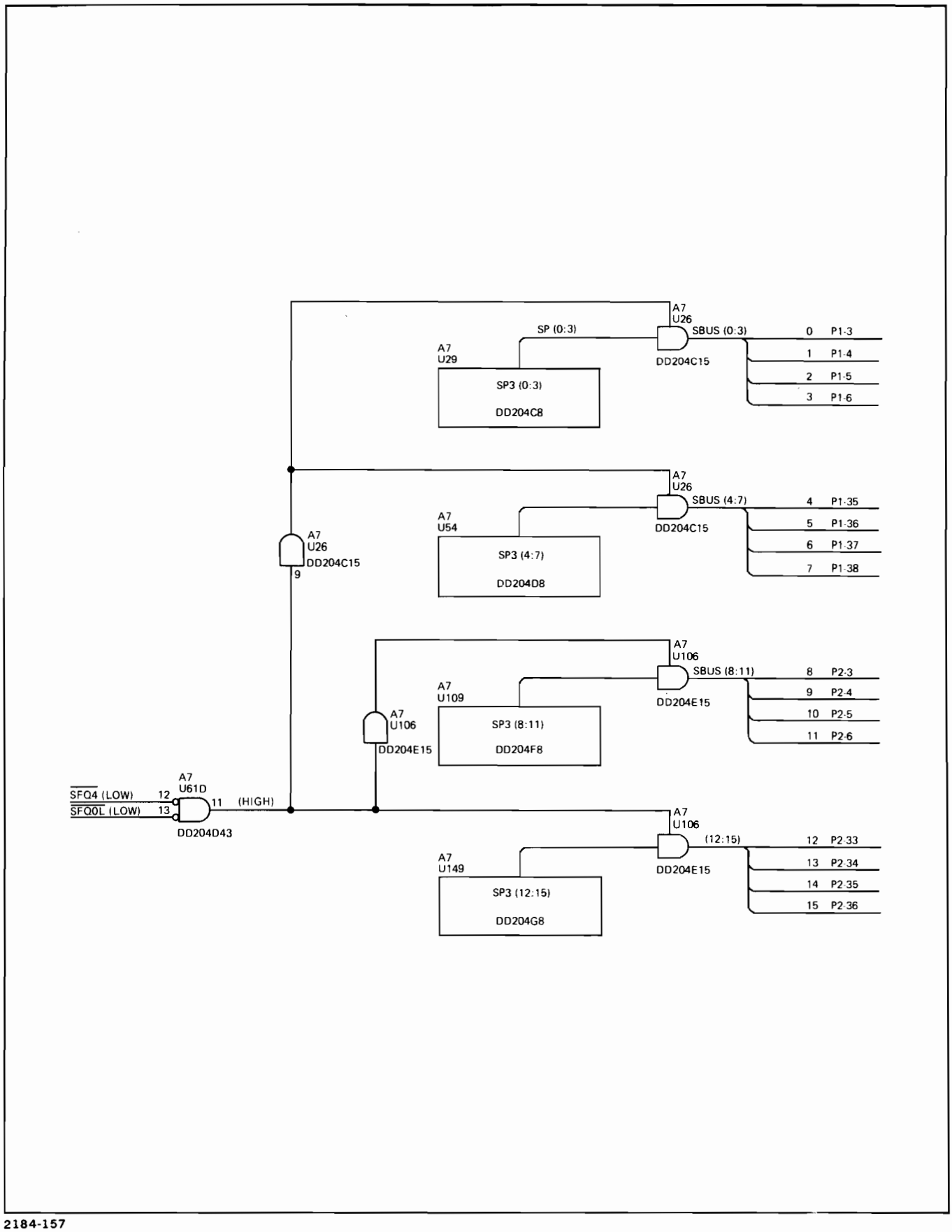
2184-154

Figure 4-18. RD (TR3S) Register Servicing Diagram



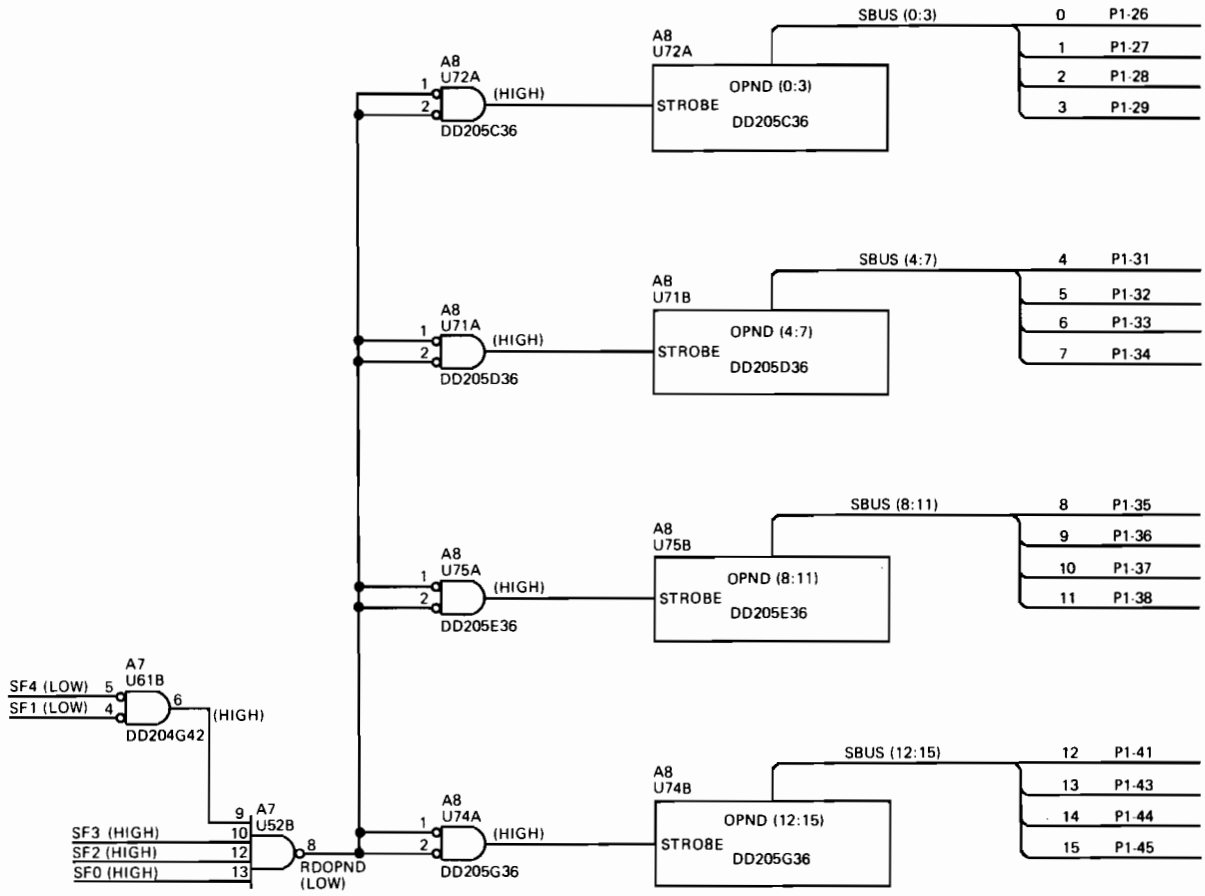
2184-156

Figure 4-19. SP2 Register Servicing Diagram



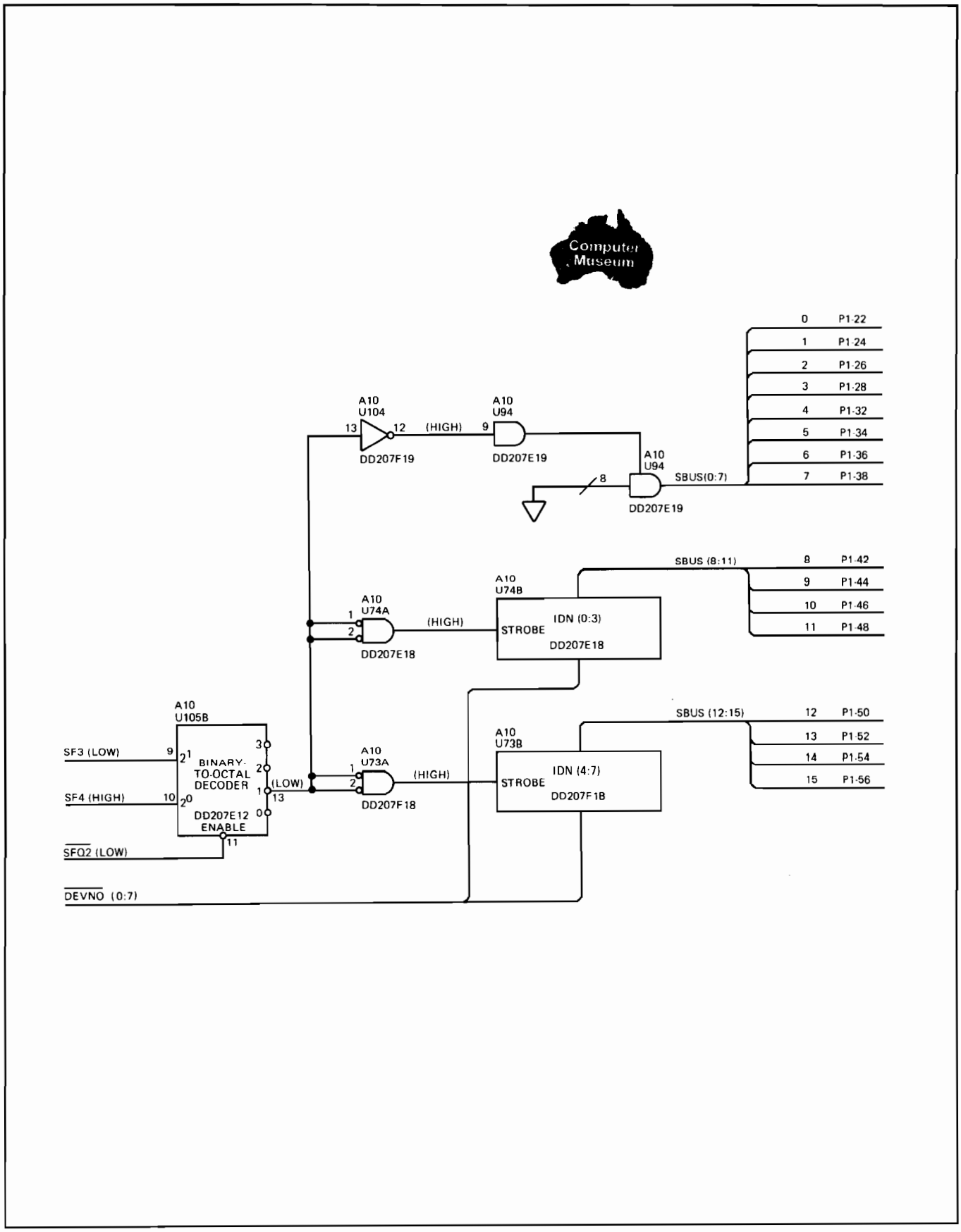
2184-157

Figure 4-20. SP3 Register Servicing Diagram



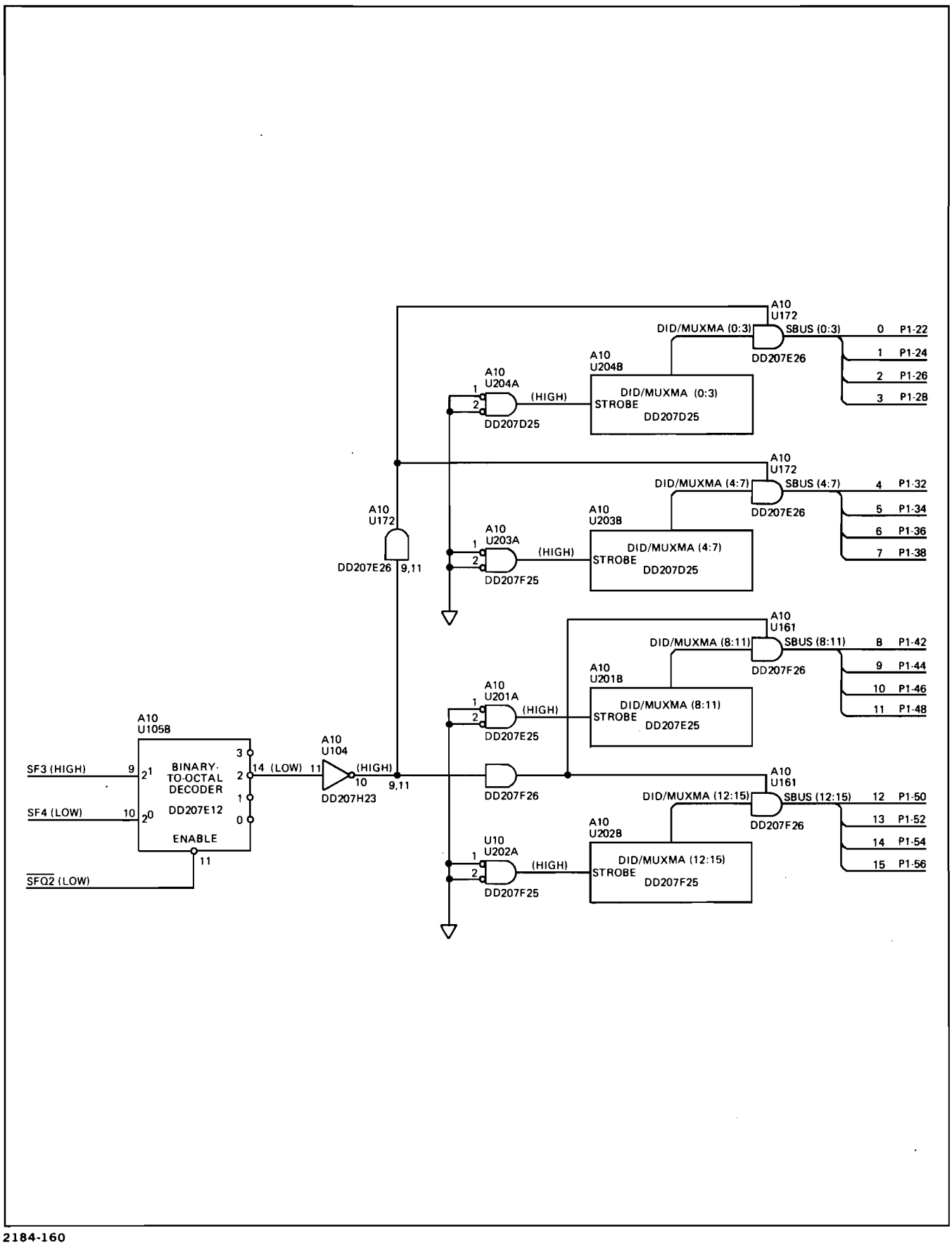
2184-158

Figure 4-21. OPND Register Servicing Diagram



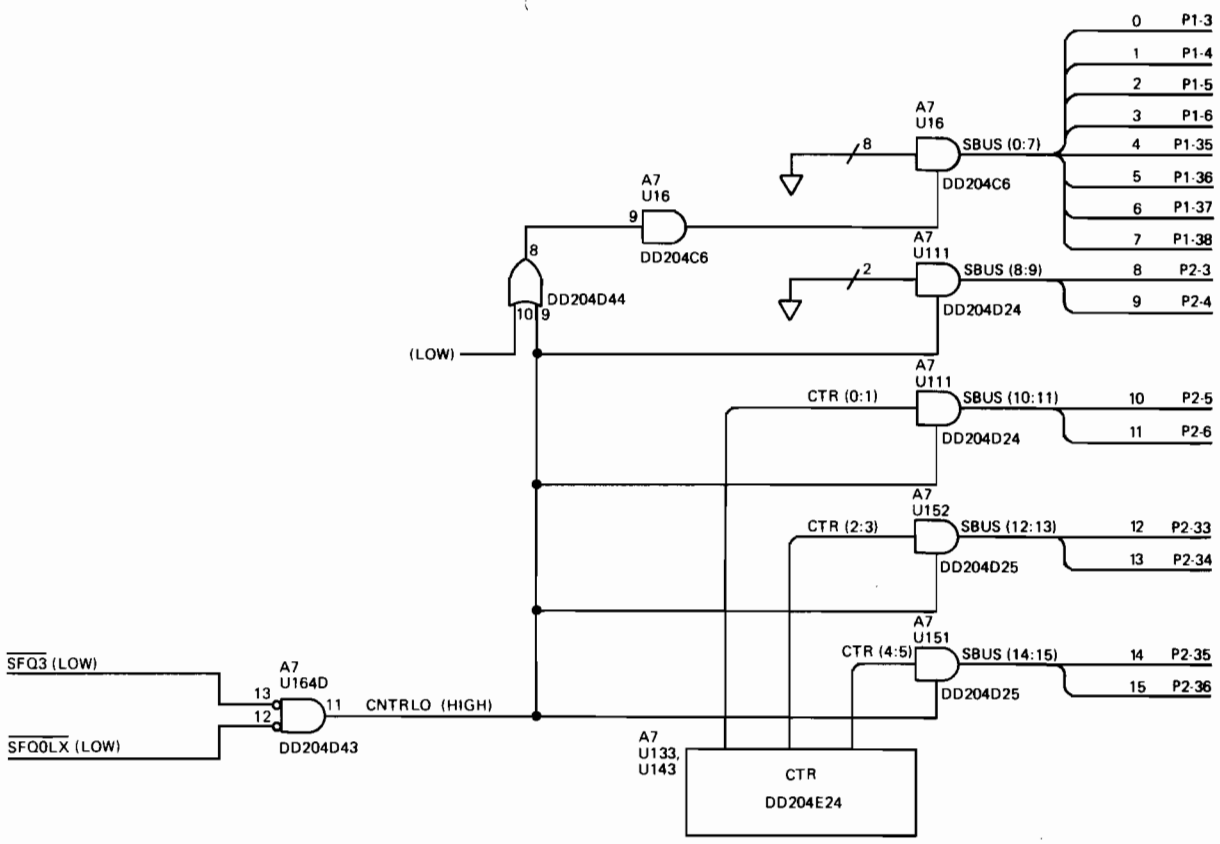
2184-159

Figure 4-22. IDN Register Servicing Diagram



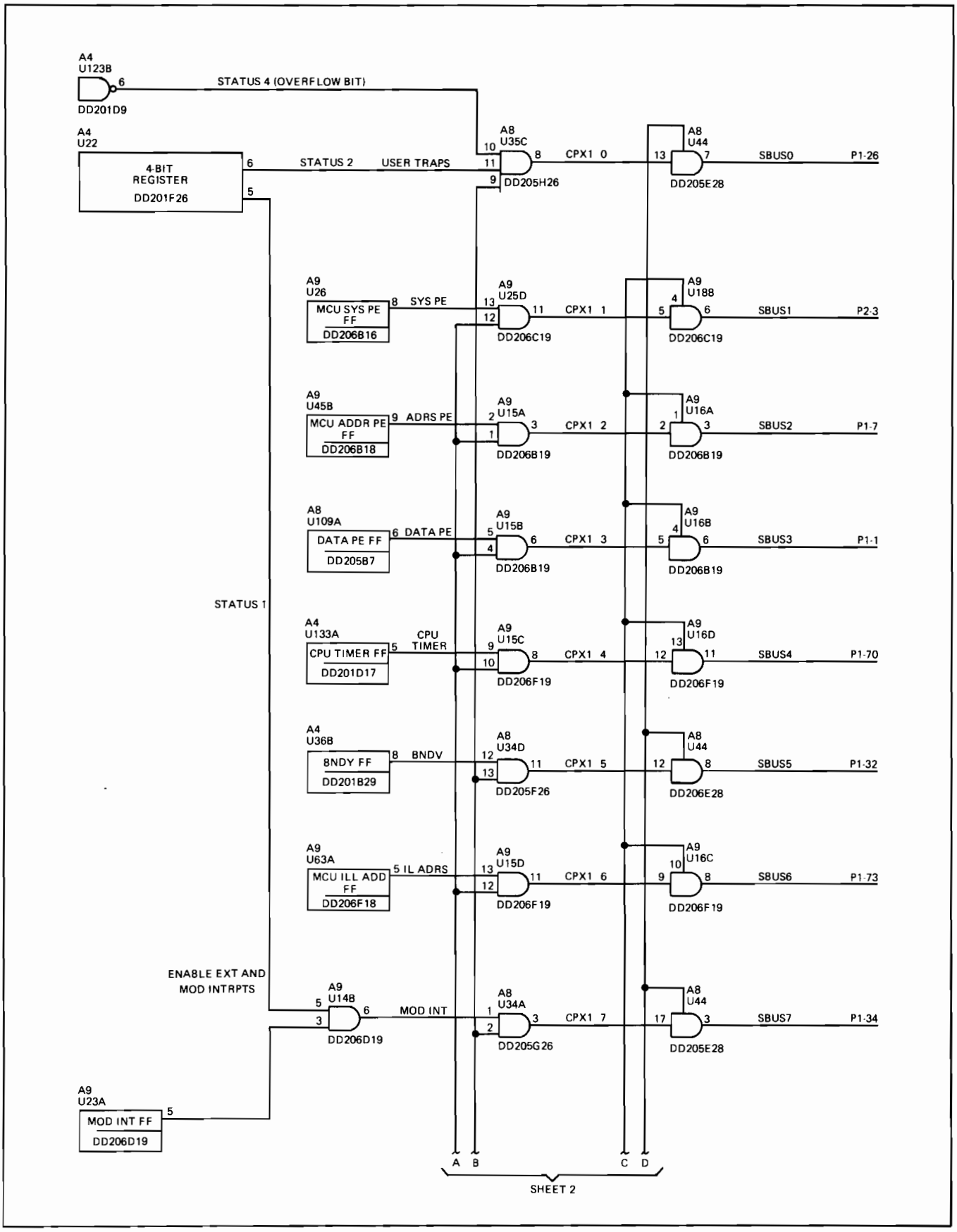
2184-160

Figure 4-23. DID/MUXMA Register Servicing Diagram



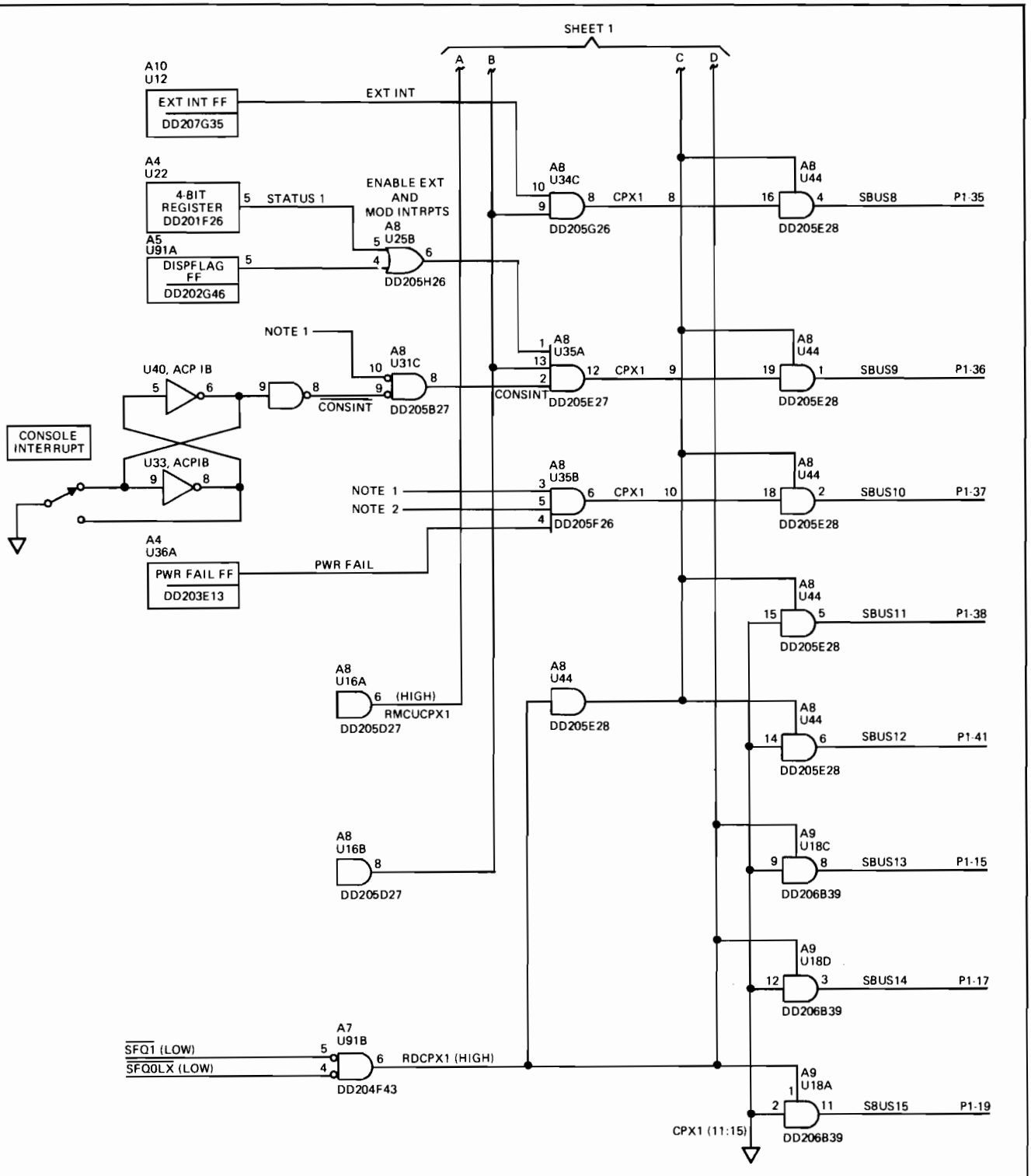
2184-161

Figure 4-24. CNTR Register Servicing Diagram



2184-162

Figure 4-25. CPX1 Register Servicing Diagram (Sheet 1 of 2)

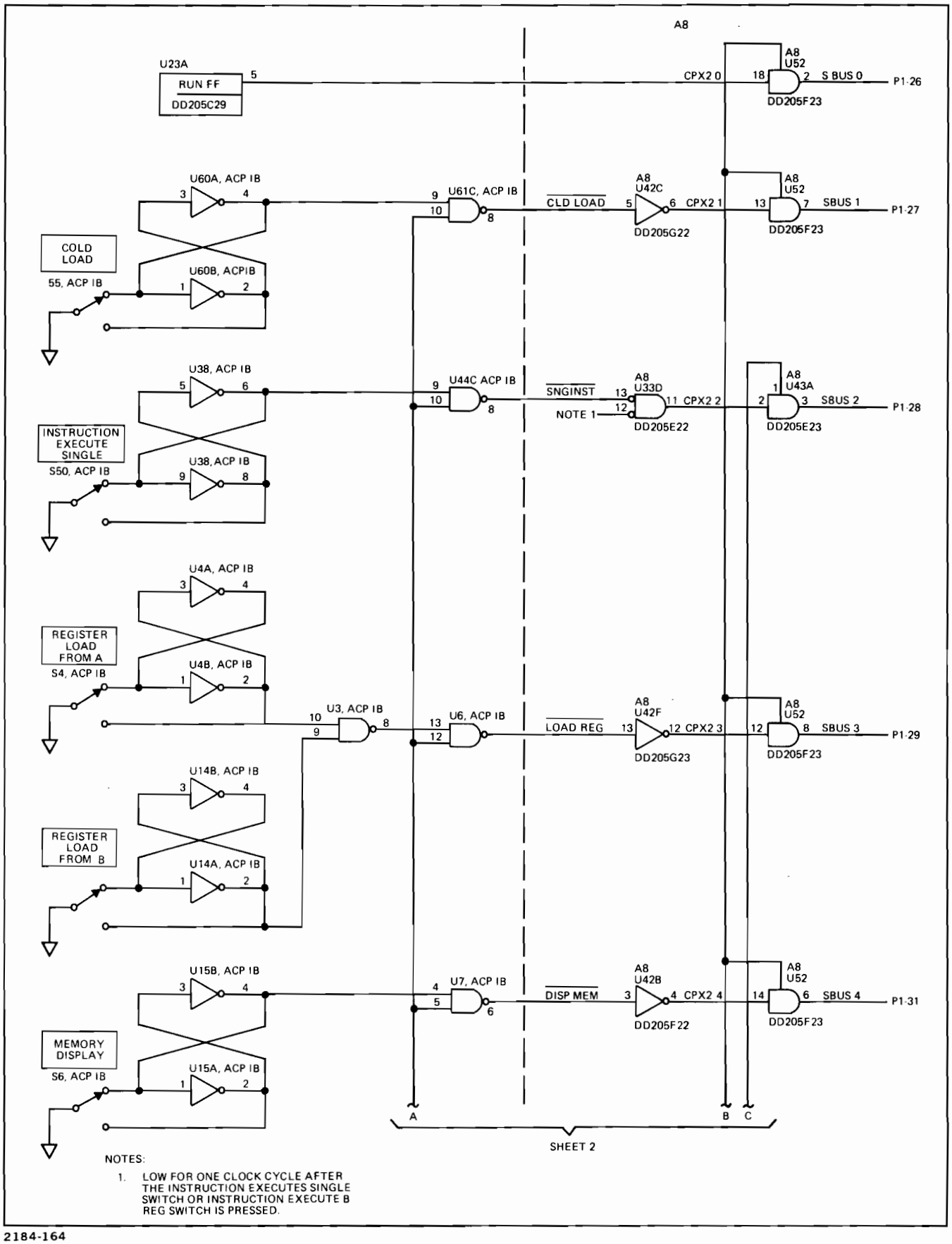


NOTES:

1. LOW WHEN ANY ACP SPRING-RETURN SWITCH IS PRESSED, WITH THE EXCEPTION OF THE CPU RESET SWITCH AND THE I/O RESET SWITCH. GOES HIGH WHEN CONSOLE INTERRUPT MICROROUTINE STARTS.
2. LOW WHEN ANY OF THE FOLLOWING CONDITIONS EXISTS:
   (A) INSTRUCTION EXECUTE SINGLE SWITCH IS BEING PRESSED.
   (B) INSTRUCTION EXECUTE 8 REG SWITCH IS BEING PRESSED.
   (C) POWER FAILURE ROUTINE HAS STARTED.

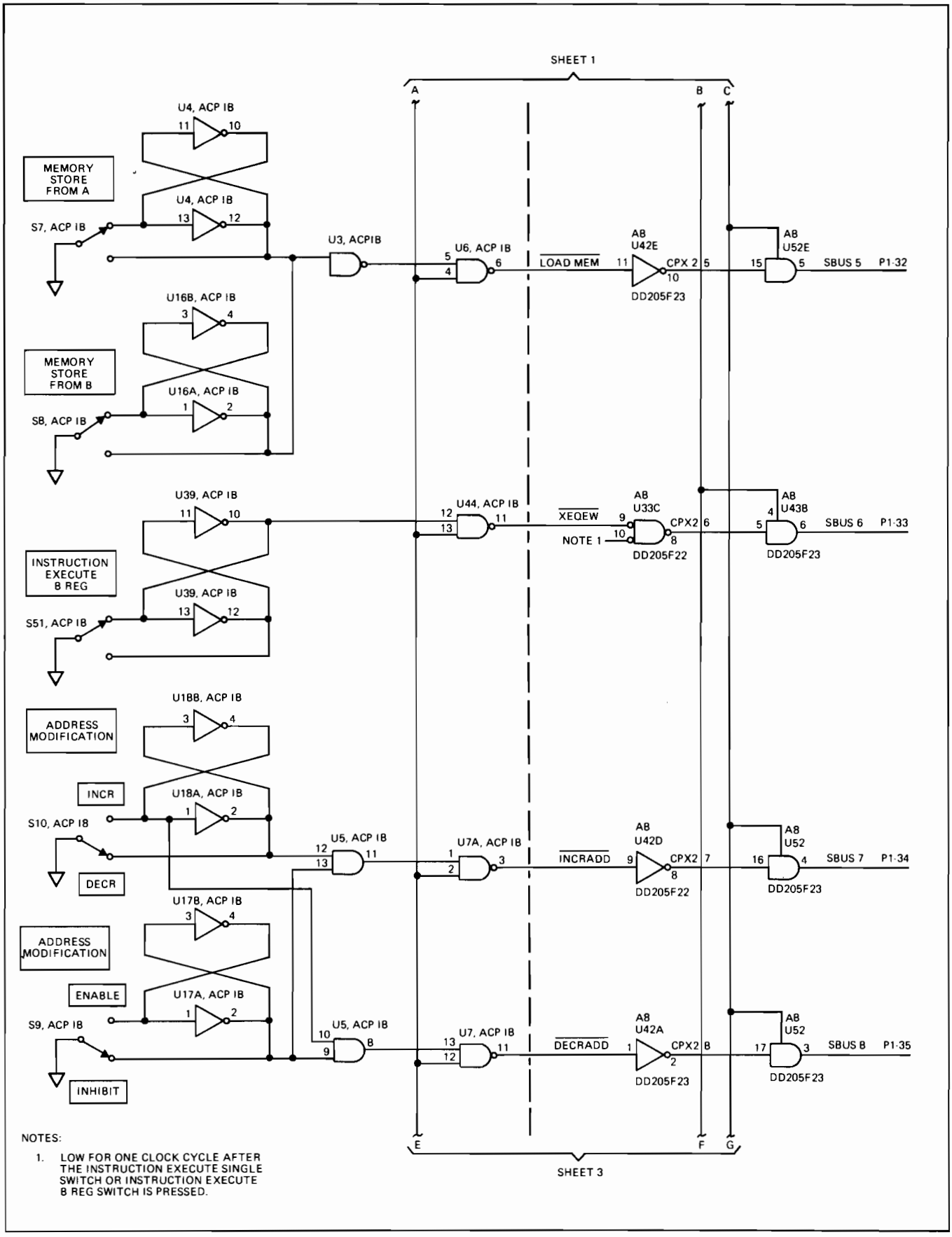
2184-163

Figure 4-25. CPX1 Register Servicing Diagram (Sheet 2 of 2)



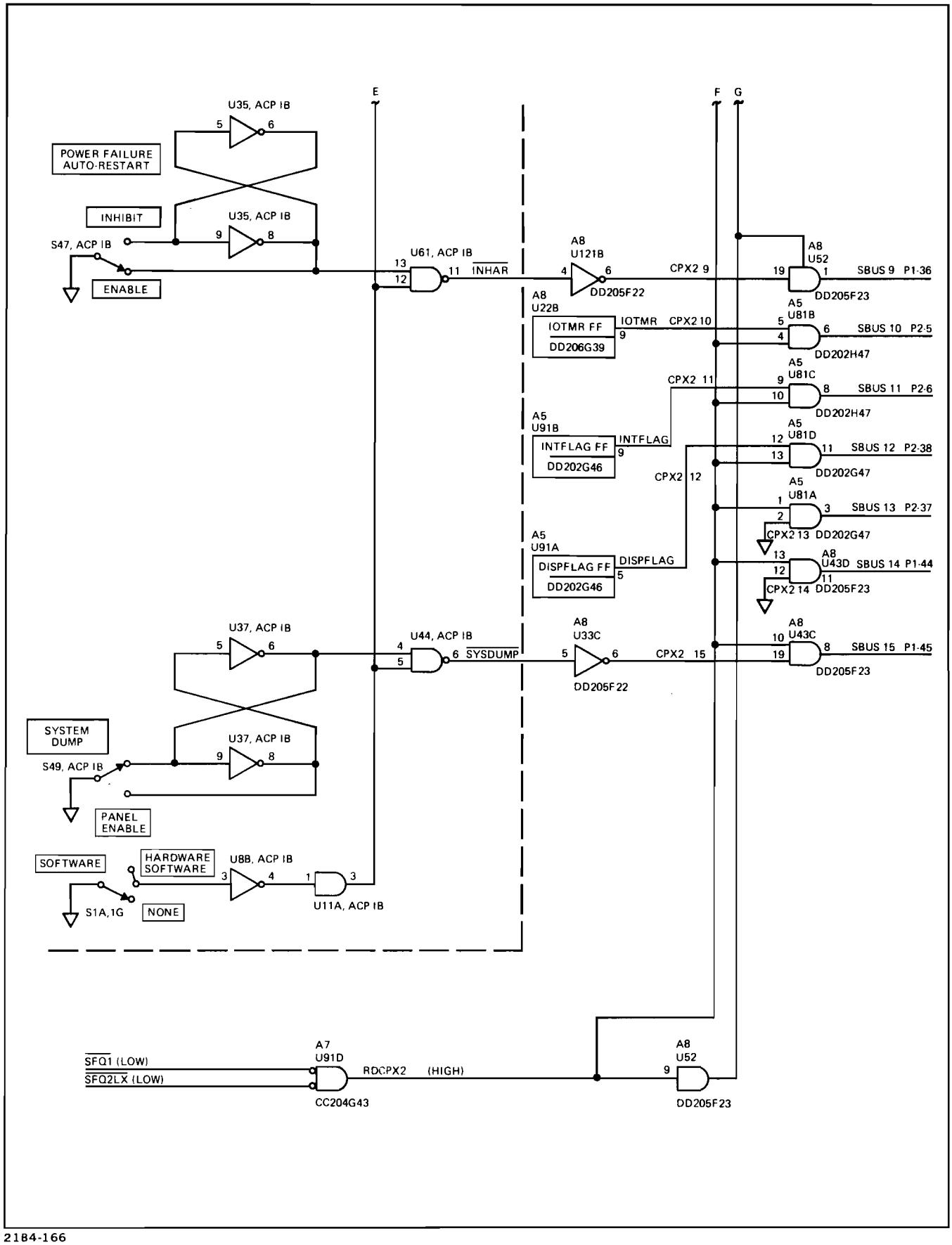
2184-164

Figure 4-26. CPX2 Register Servicing Diagram (Sheet 1 of 3)



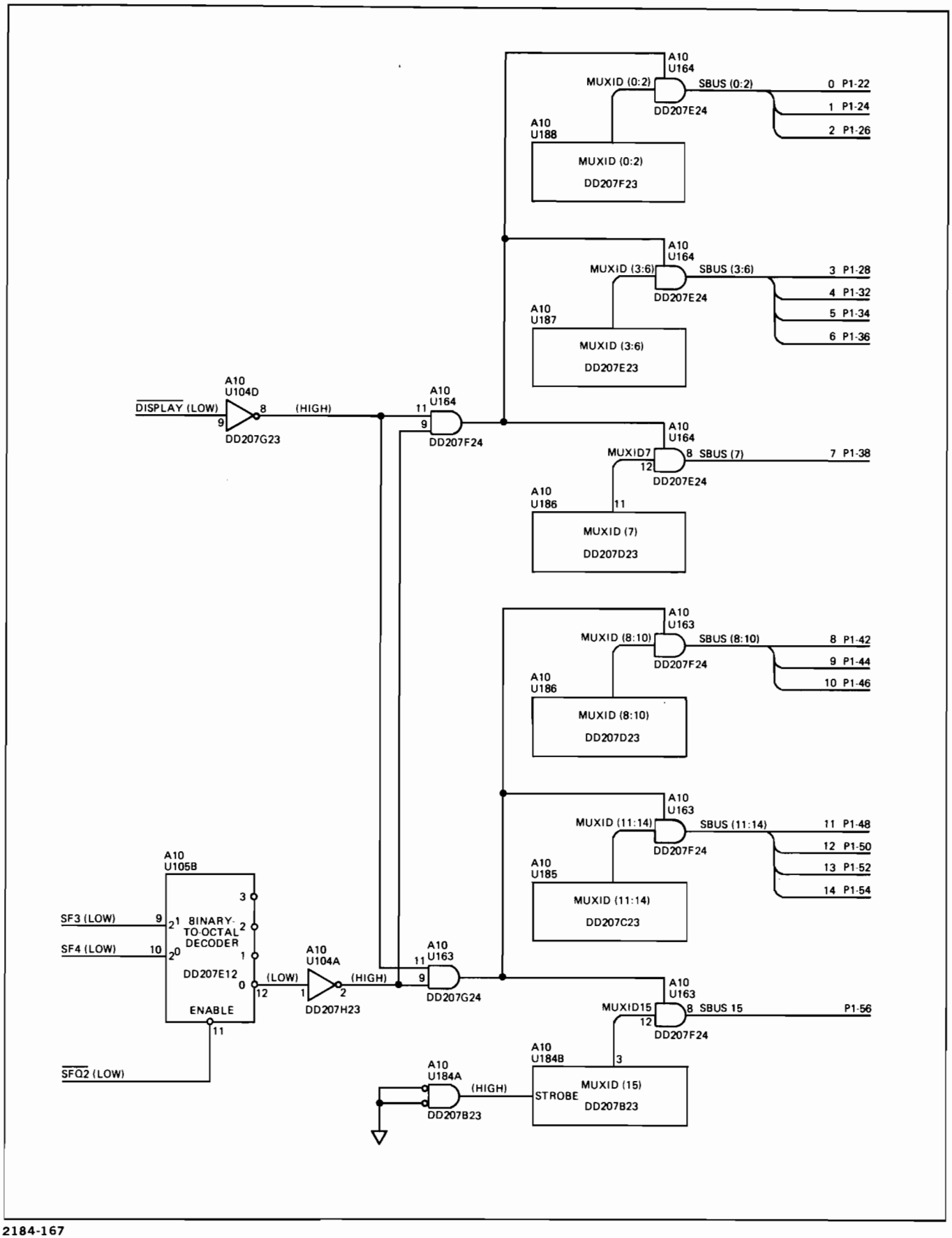
2184-165

Figure 4-26. CPX2 Register Servicing Diagram (Sheet 2 of 3)



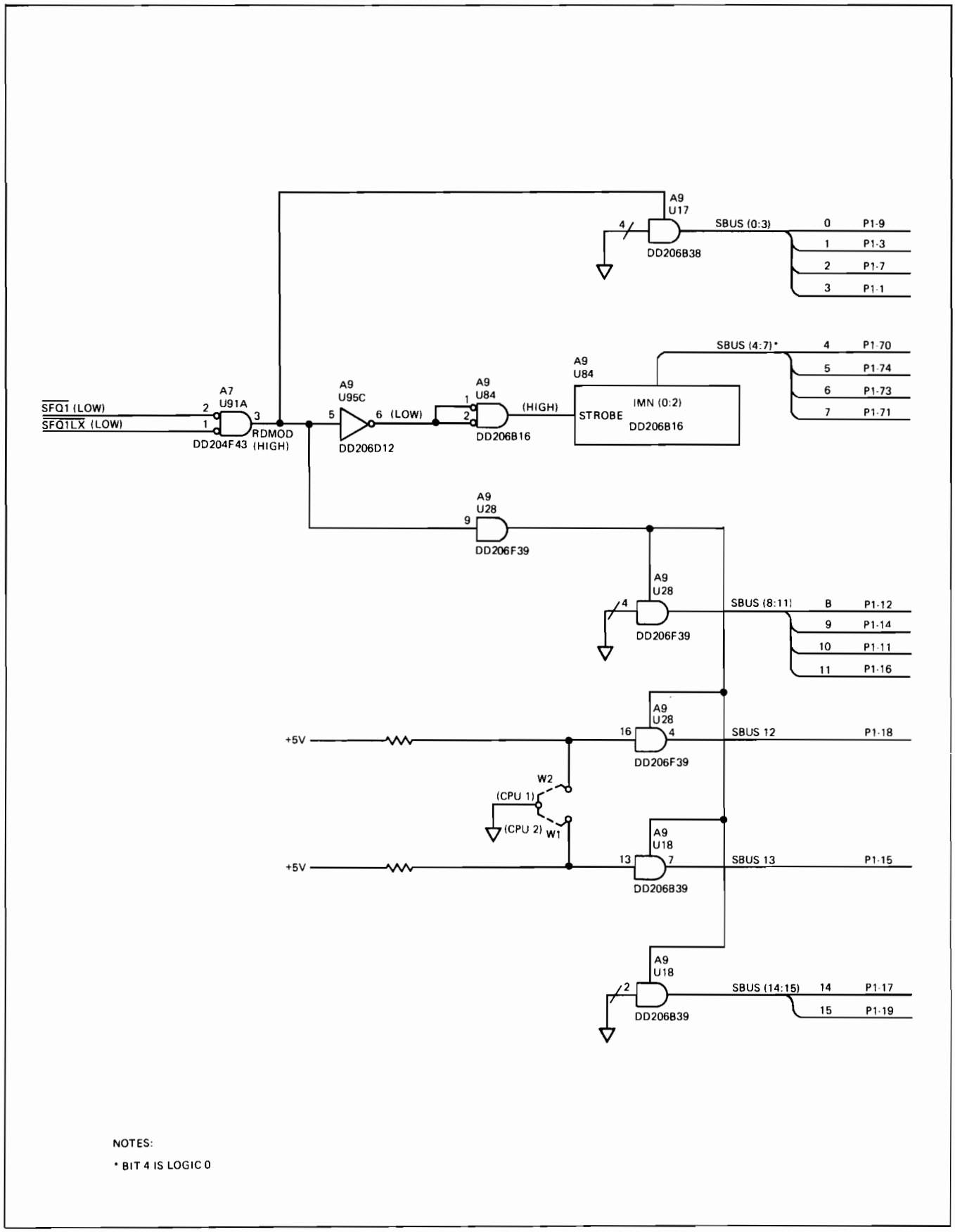
2184-166

Figure 4-26. CPX2 Register Servicing Diagram (Sheet 3 of 3)



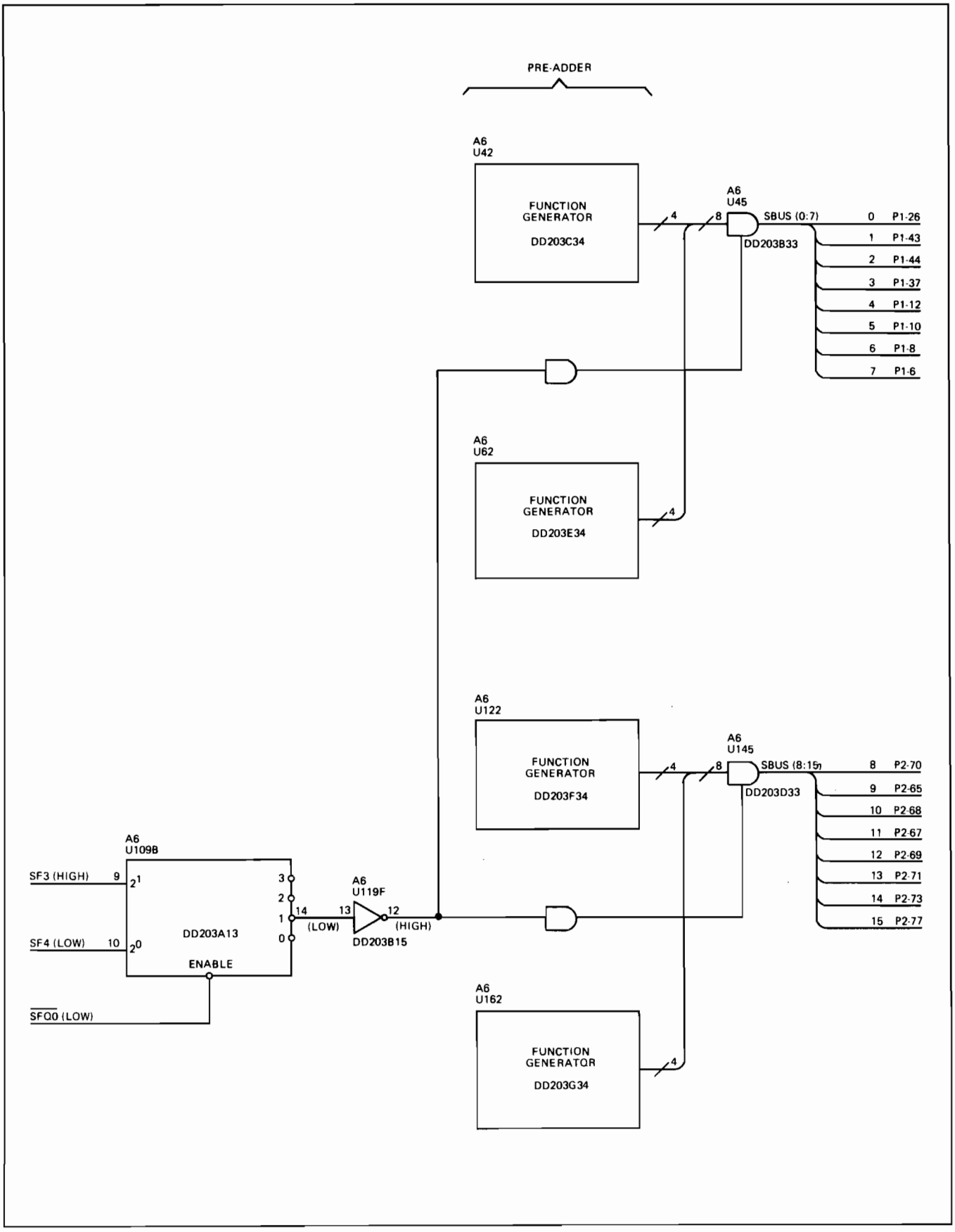
2184-167

Figure 4-27. MUXID Register Servicing Diagram



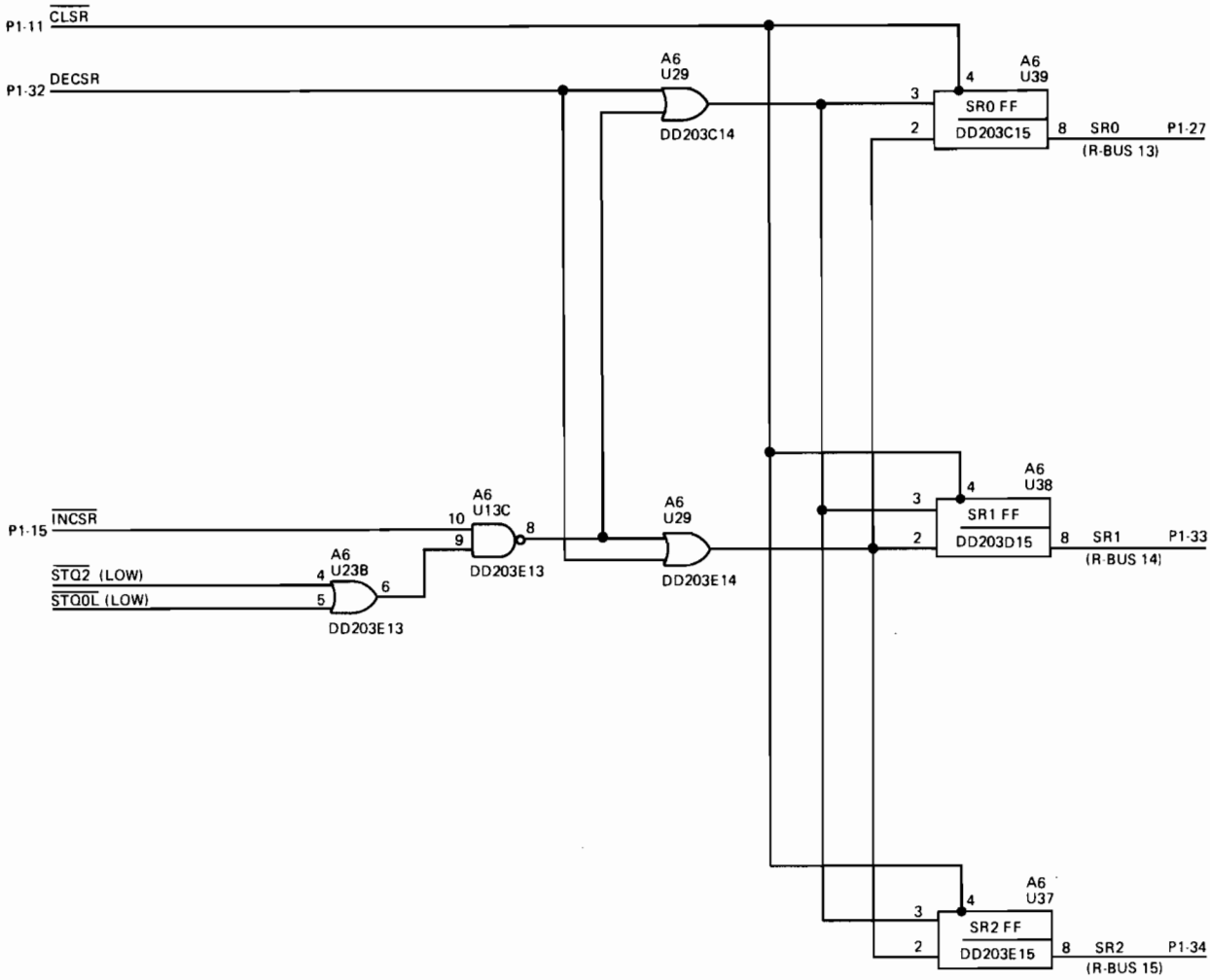
2184-168

Figure 4-28. MOD NO (IMN) Register Servicing Diagram



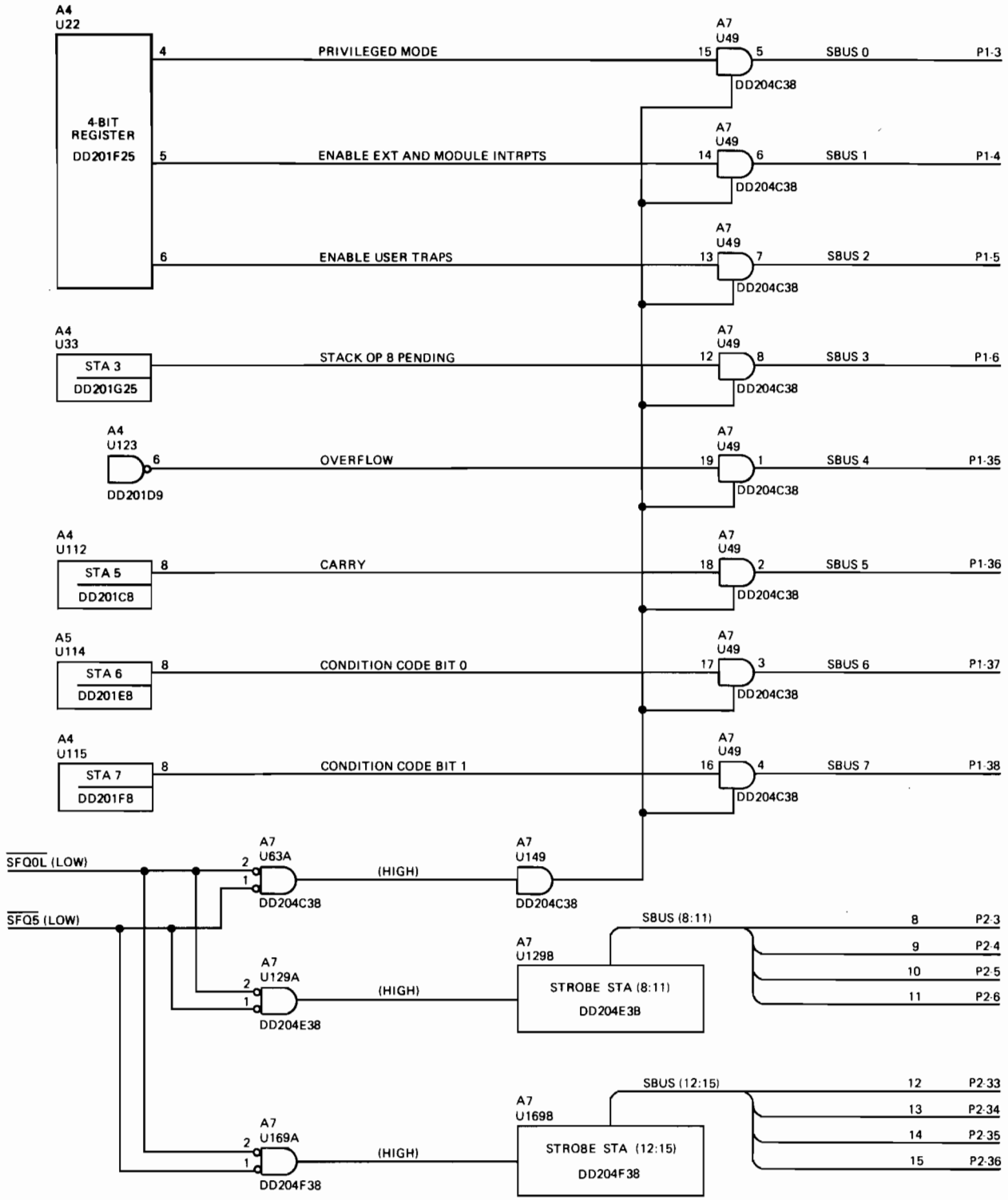
2184-169

Figure 4-29. PADD Servicing Diagram



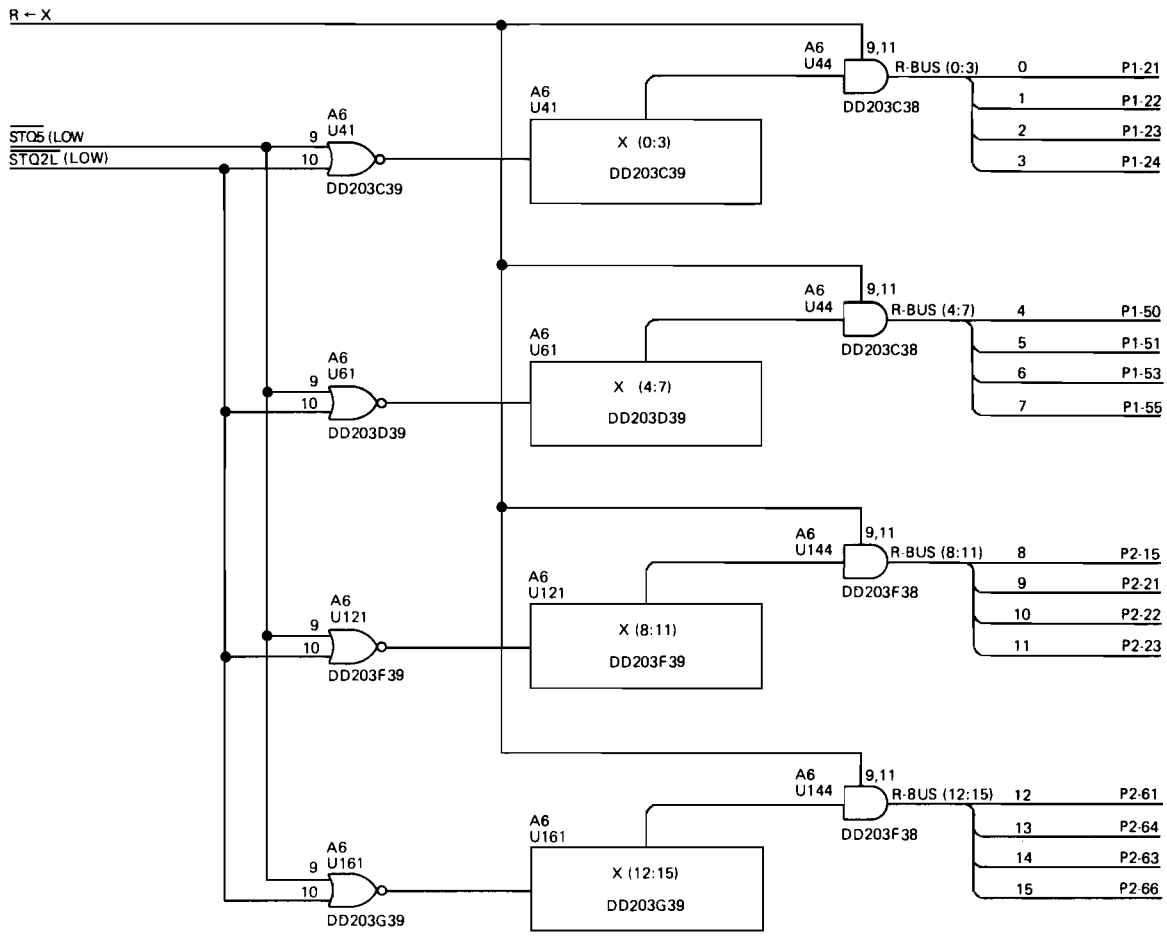
2184-189

Figure 4-30. SR Register Servicing Diagram



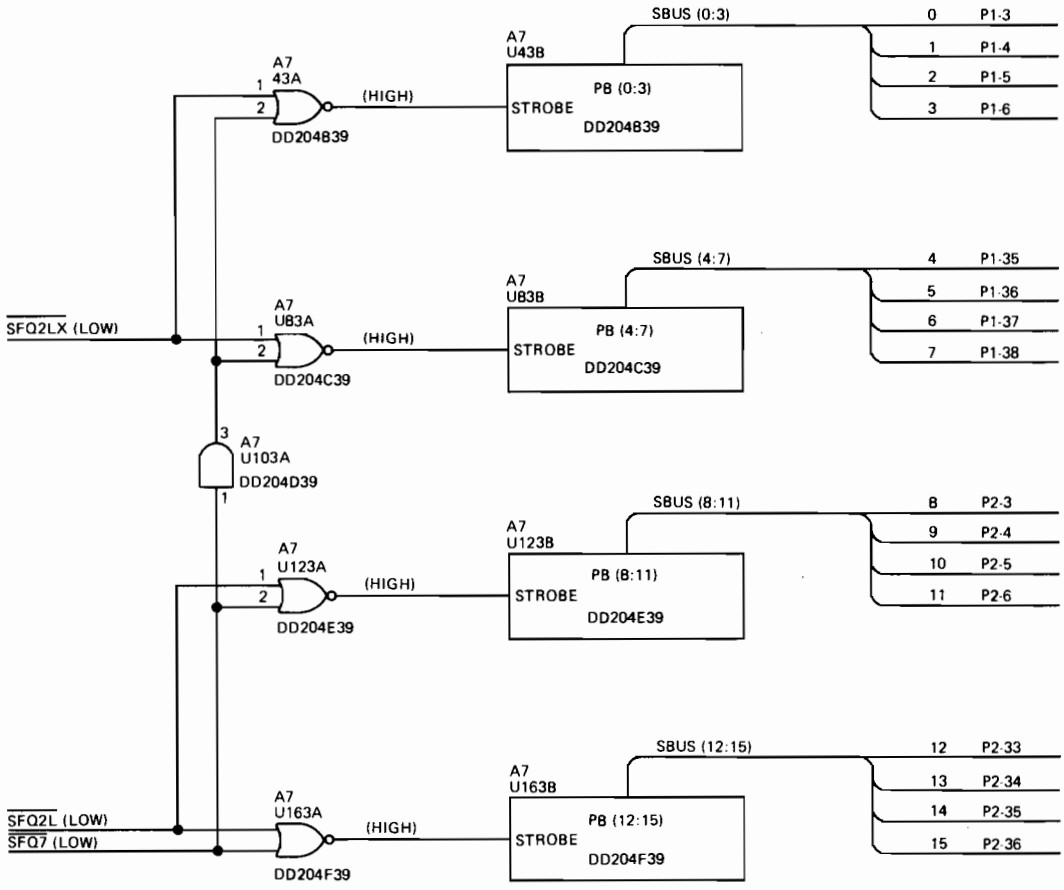
2184-170

Figure 4-31. Status Register Servicing Diagram



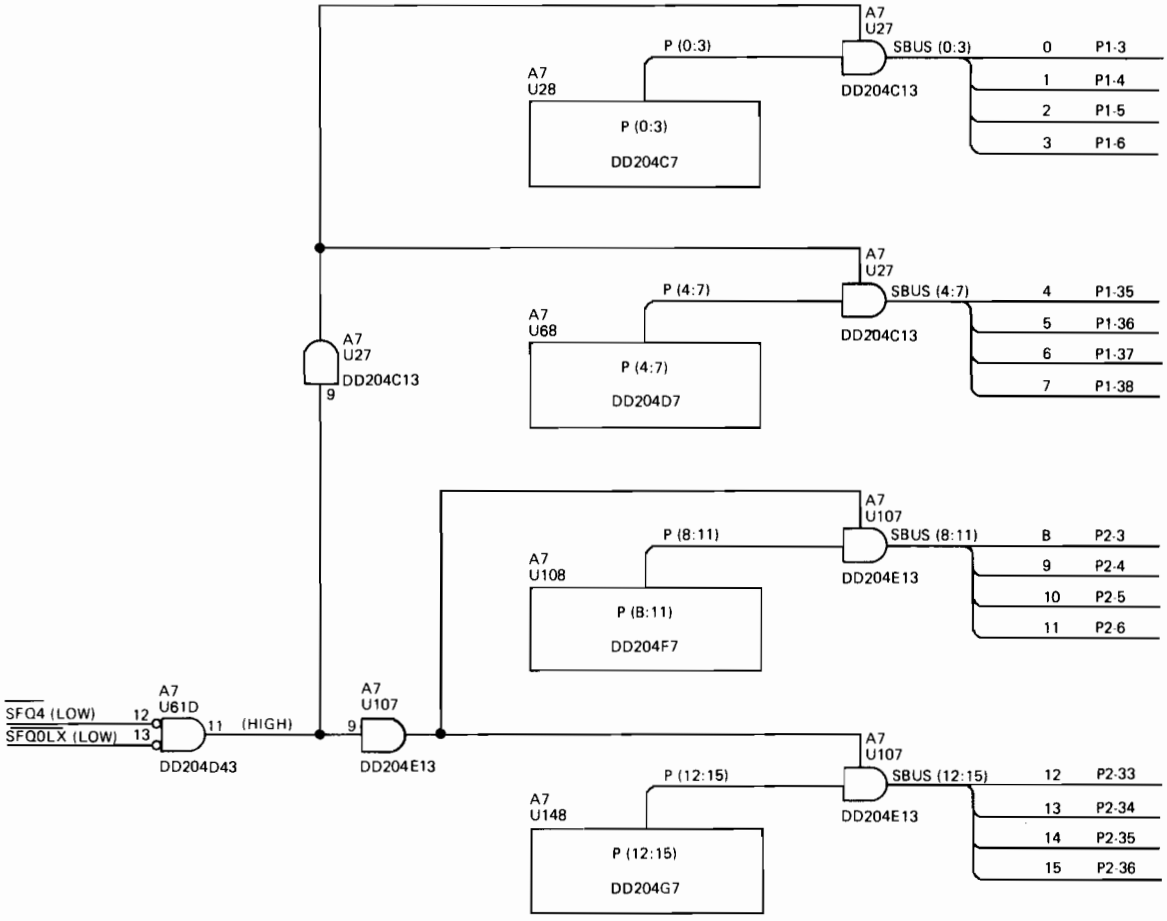
2184-191

Figure 4-32. X-Register Servicing Diagram



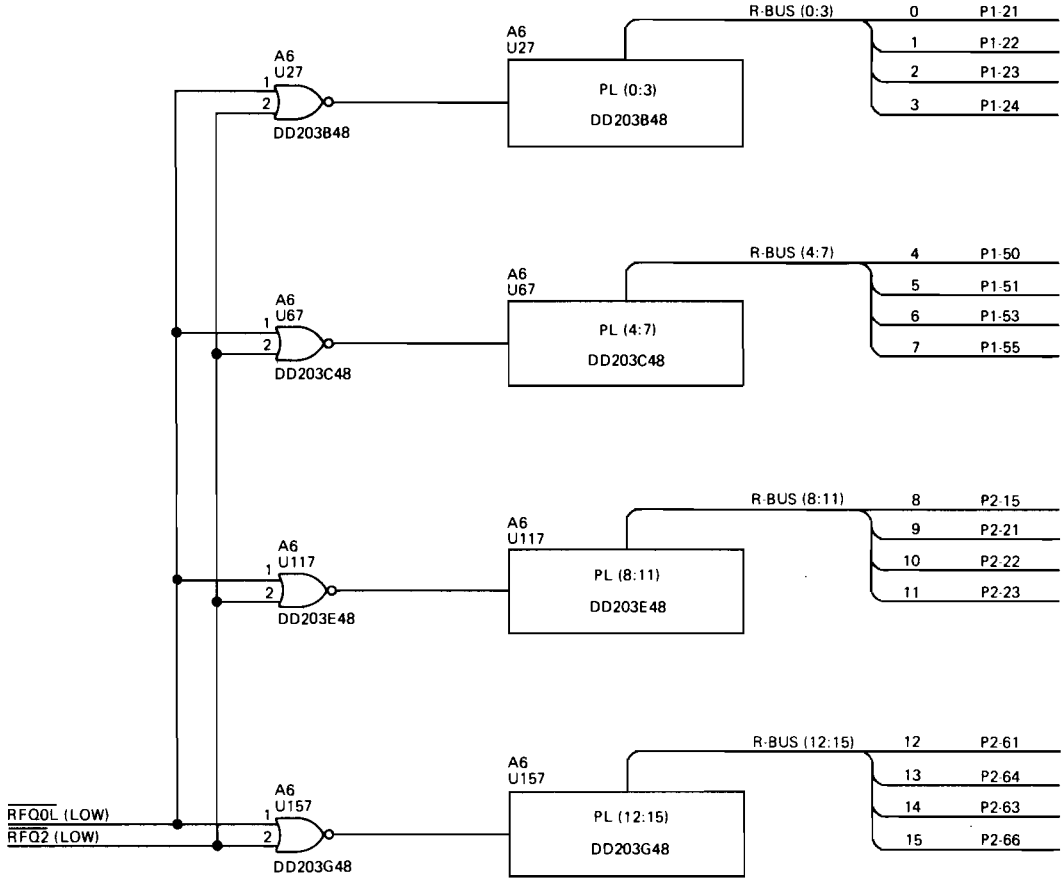
2184-171

Figure 4-33. PB Register Servicing Diagram



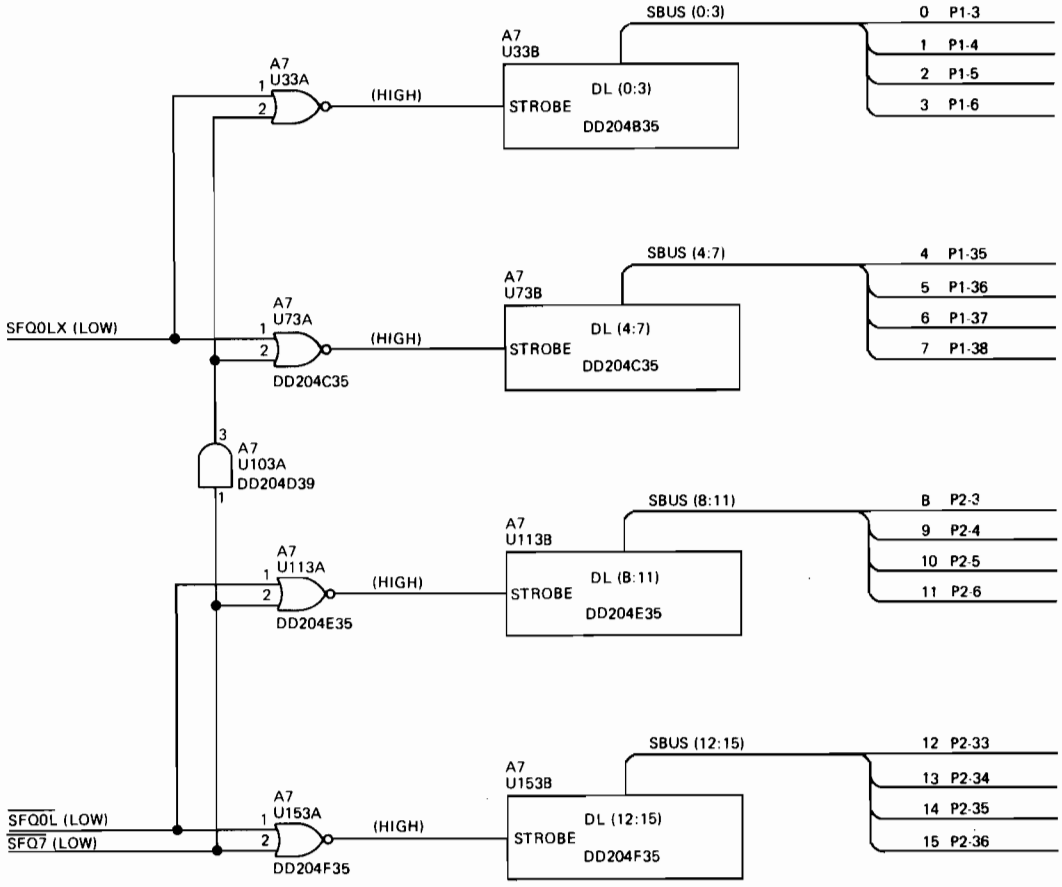
2184-172

Figure 4-34. P-Register Servicing Diagram



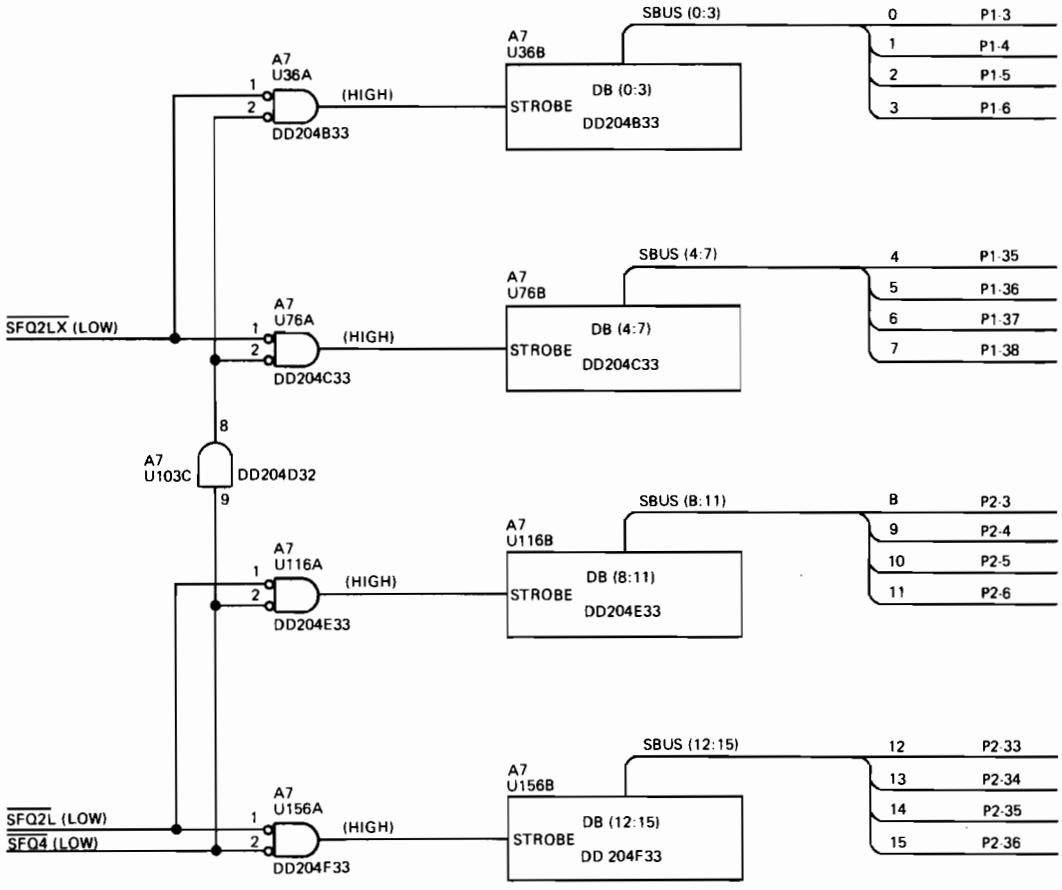
2184-190

Figure 4-35. PL Register Servicing Diagram



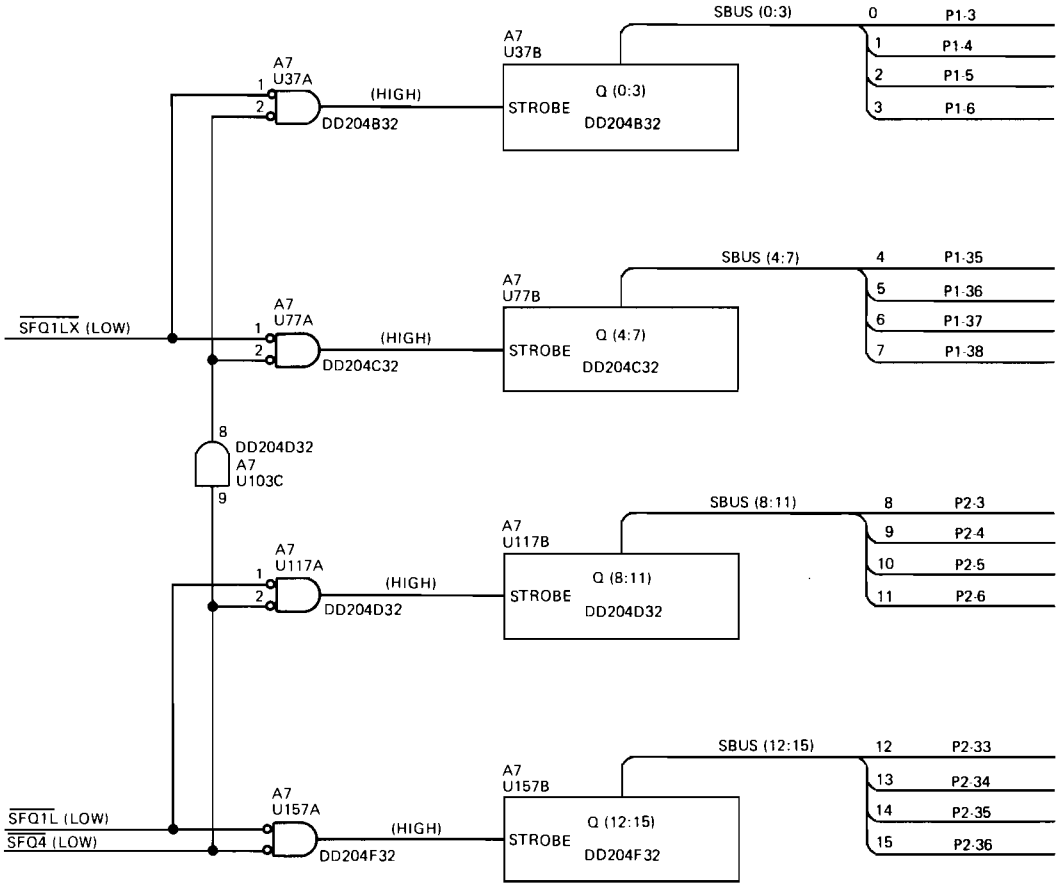
2184-173

Figure 4-36. DL Register Servicing Diagram



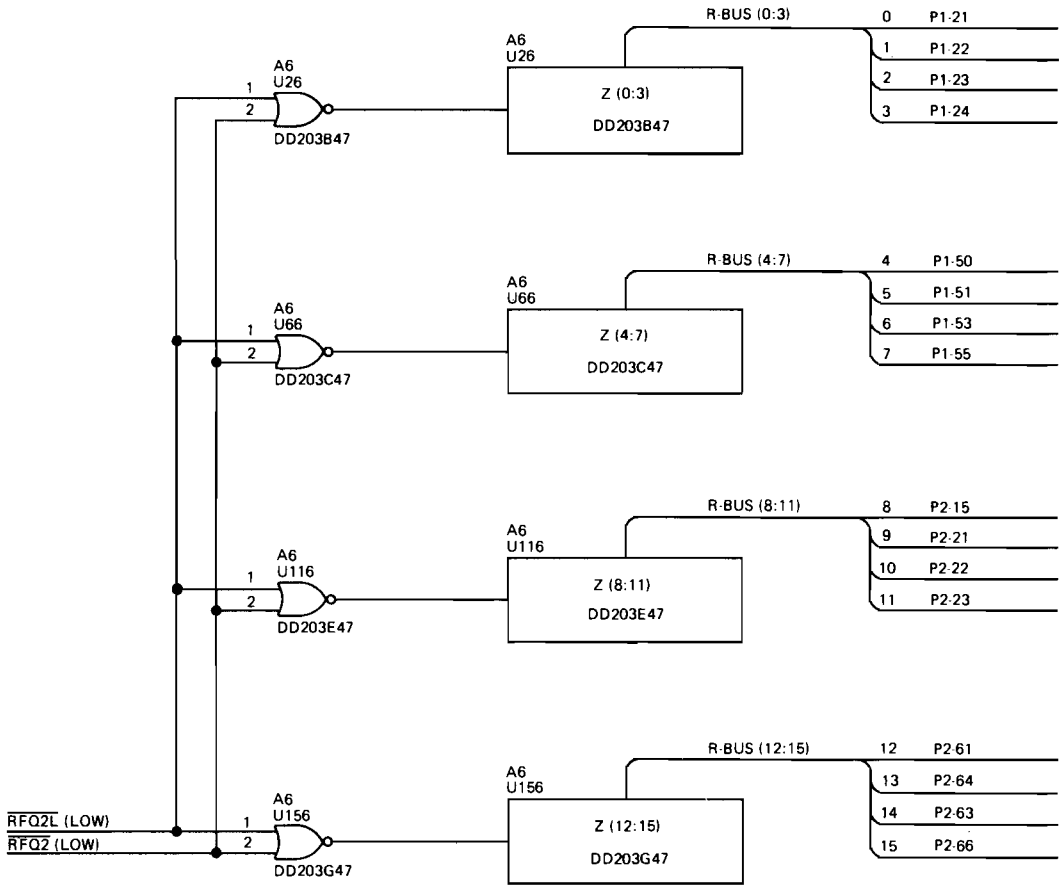
2184-174

Figure 4-37. DB Register Servicing Diagram



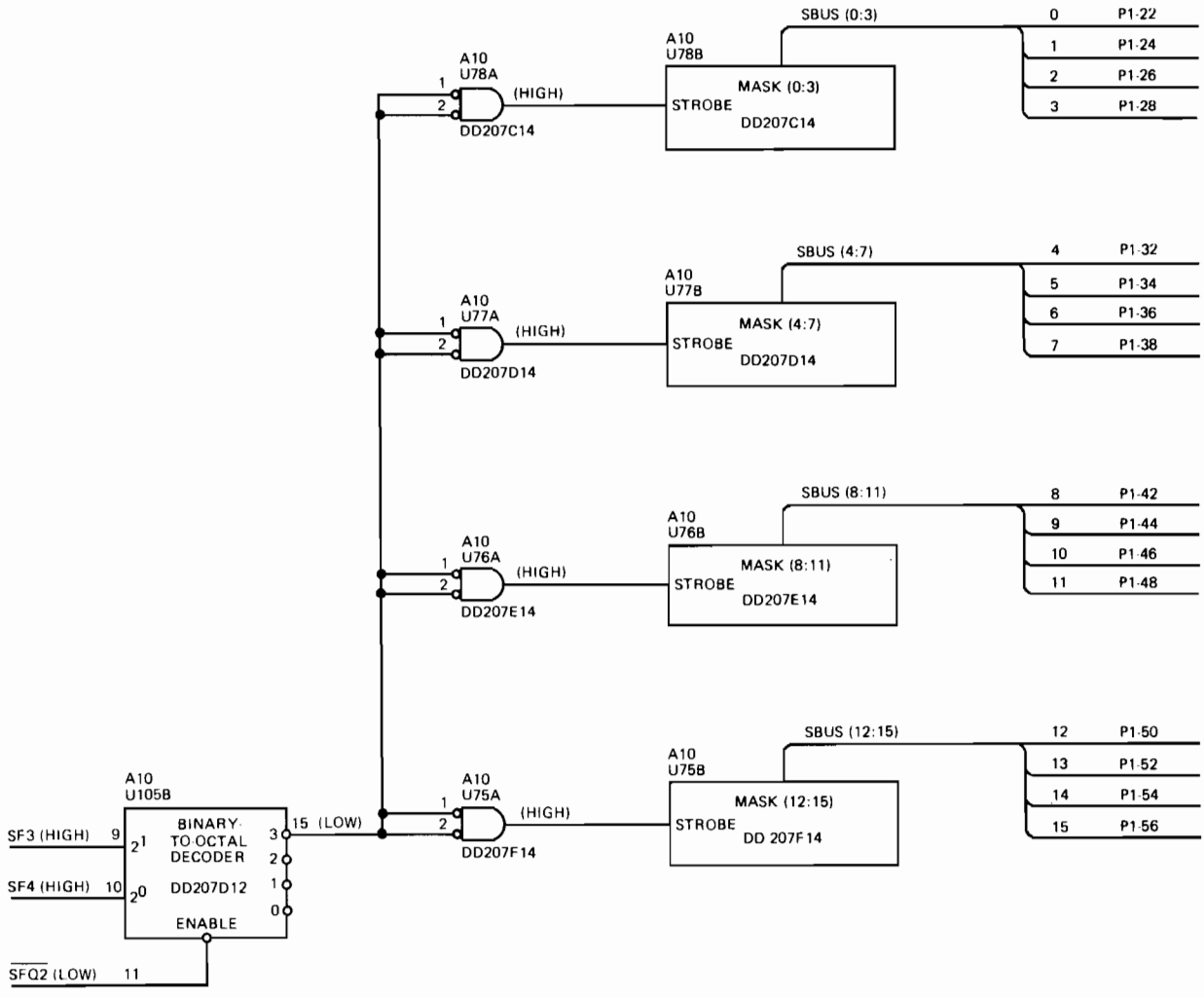
2184-175

Figure 4-38. Q-Register Servicing Diagram



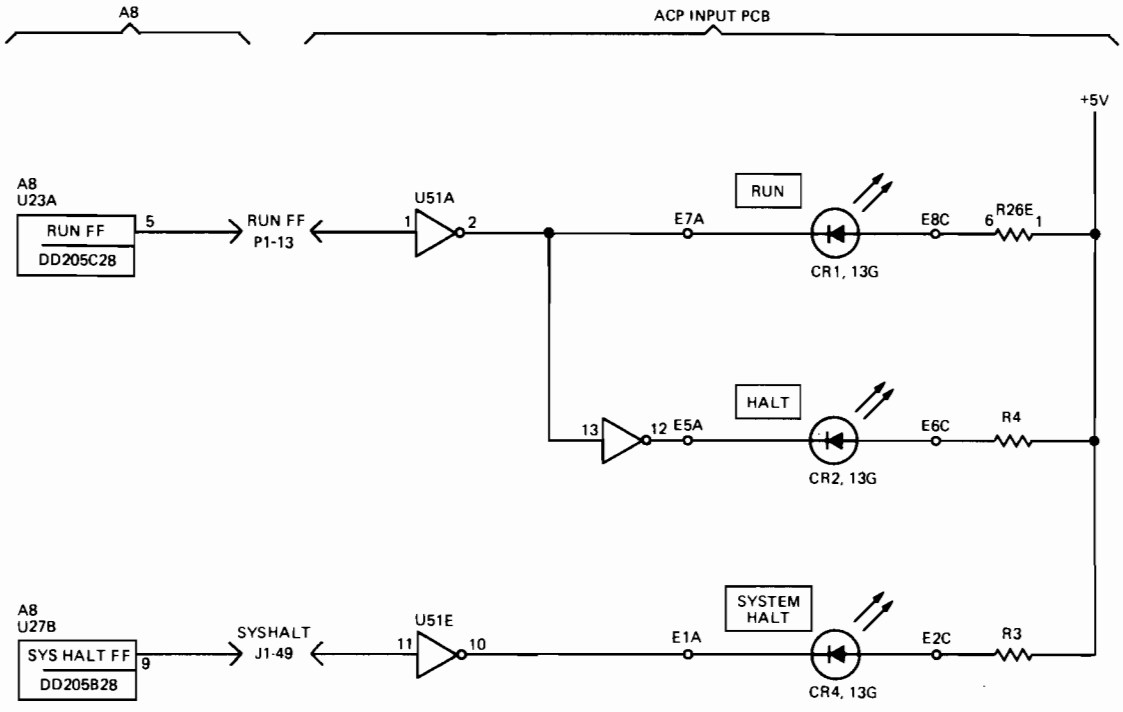
2184-188

Figure 4-39. Z-Register Servicing Diagram



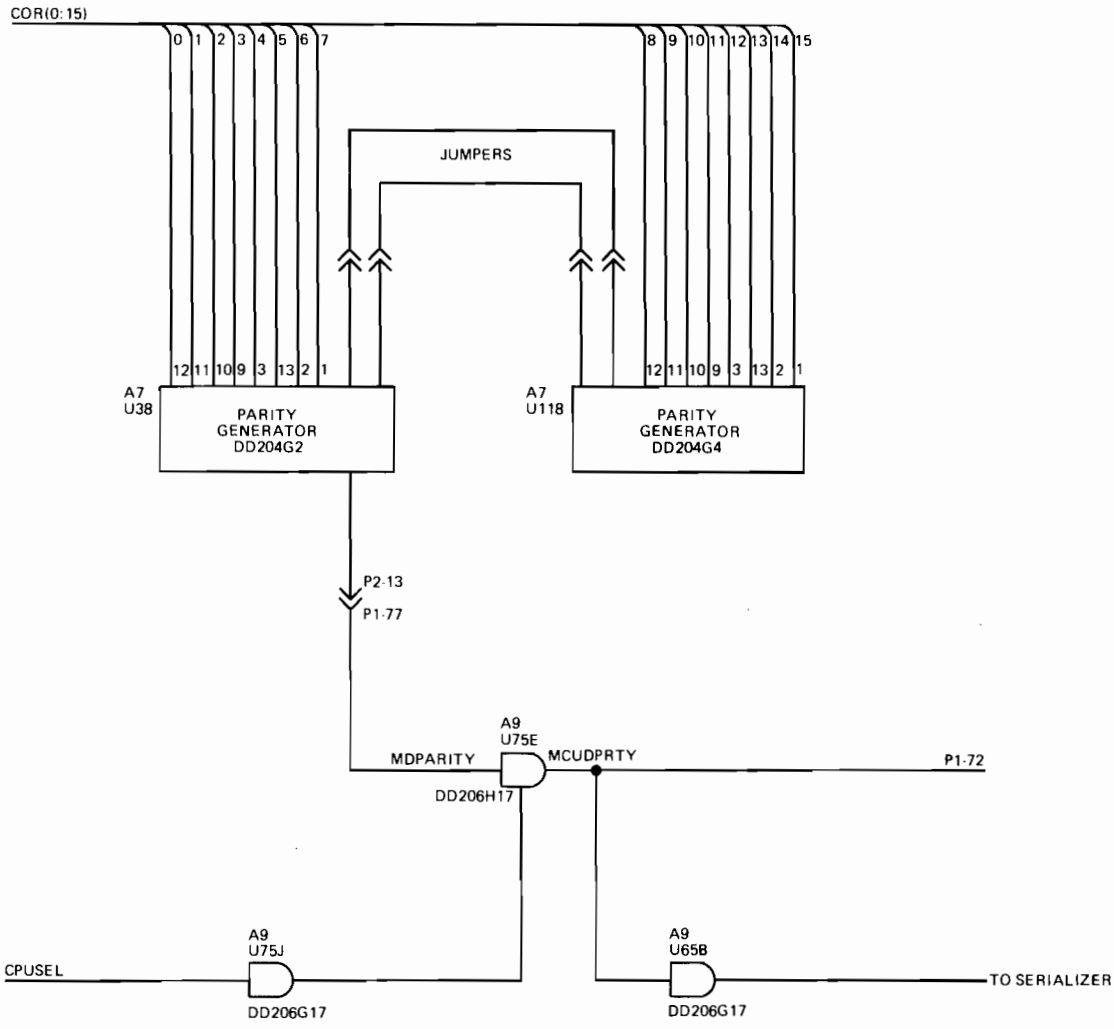
2184-176

Figure 4-40. Mask Register Servicing Diagram



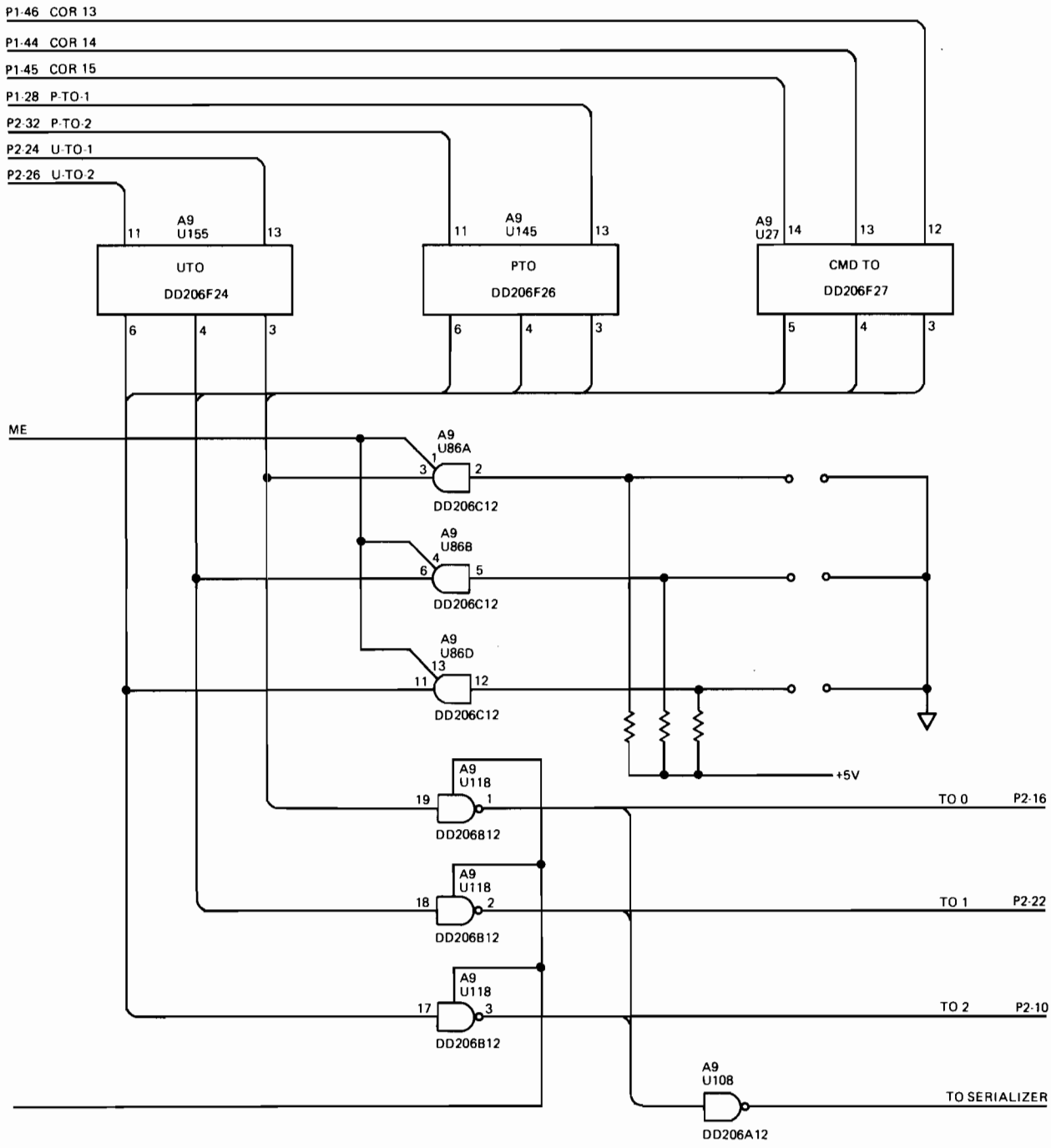
2184-178

Figure 4-41. RUN, SYSTEM HALT Servicing Diagram



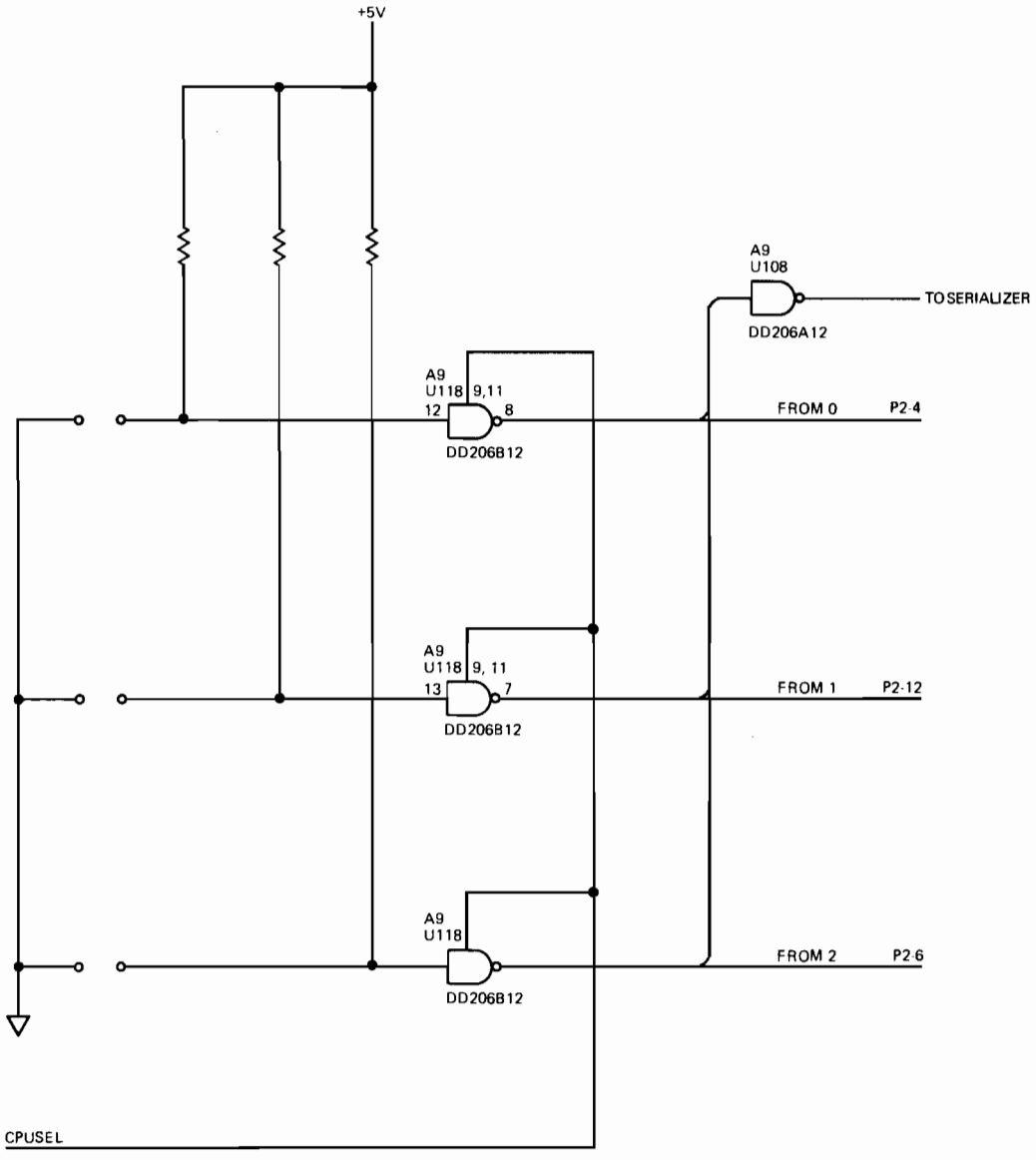
2184-187

Figure 4-42. MCUDPRTY Servicing Diagram



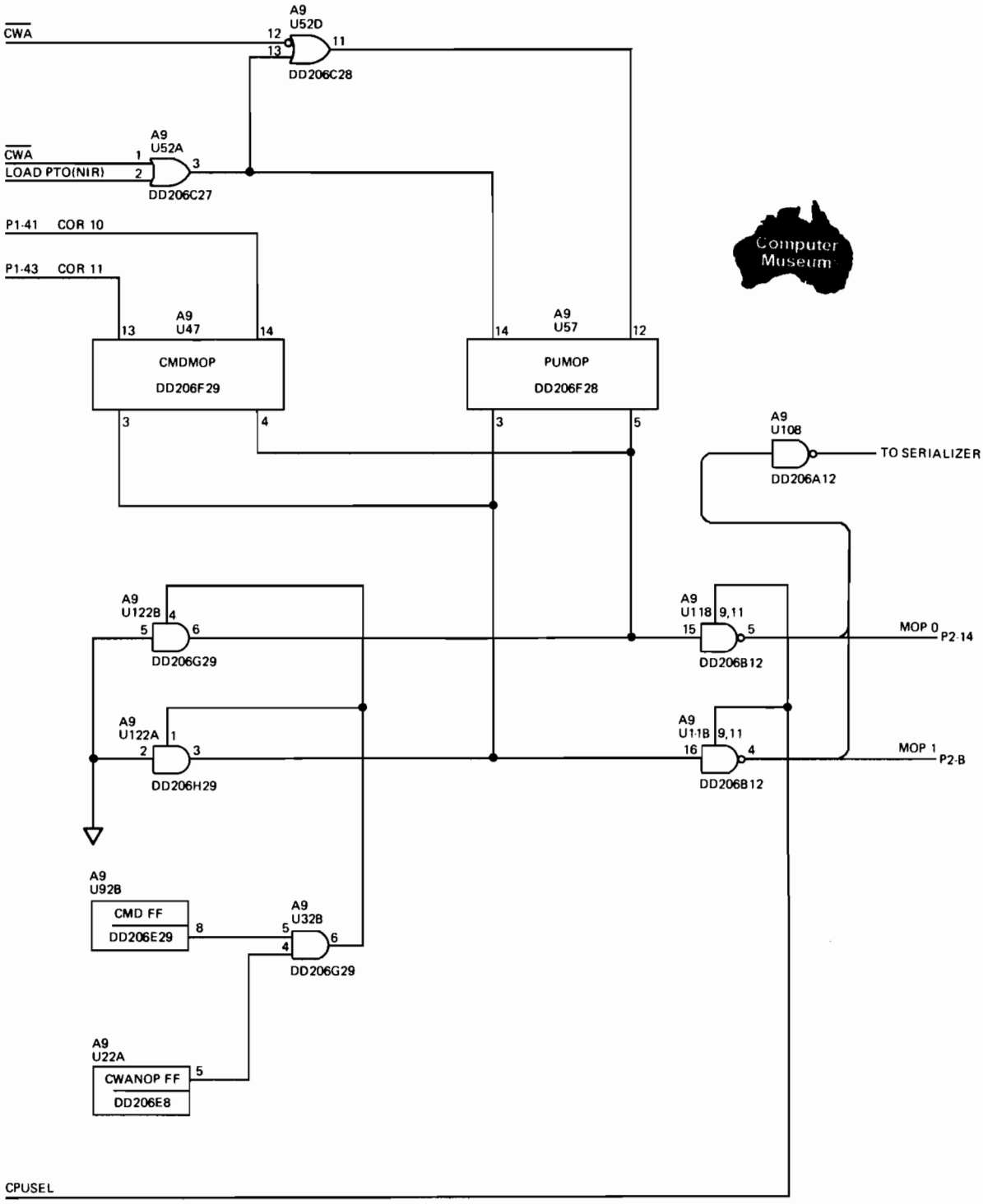
2184-183

Figure 4-43. TO Lines Servicing Diagram



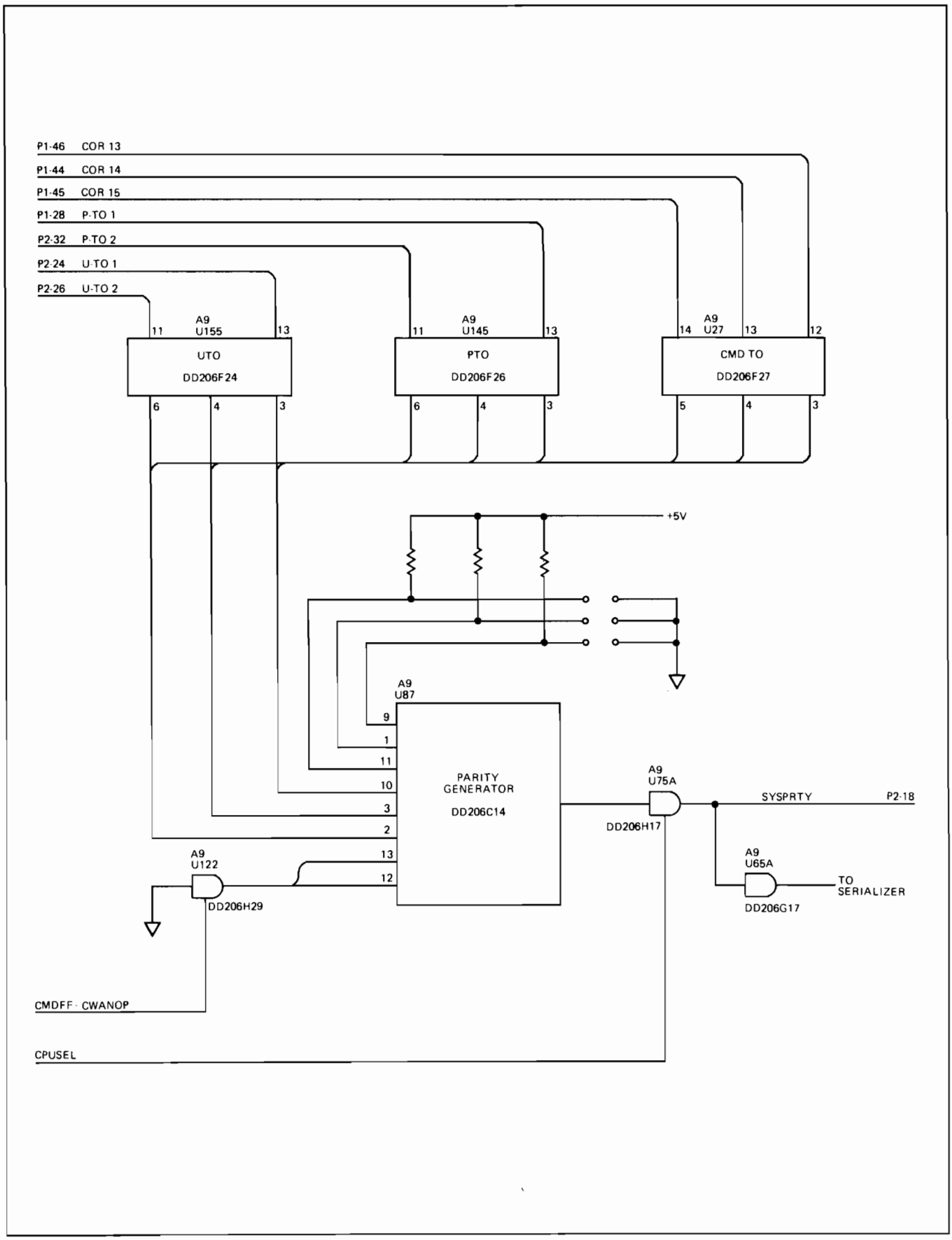
2184-185

Figure 4-44. FROM Lines Servicing Diagram



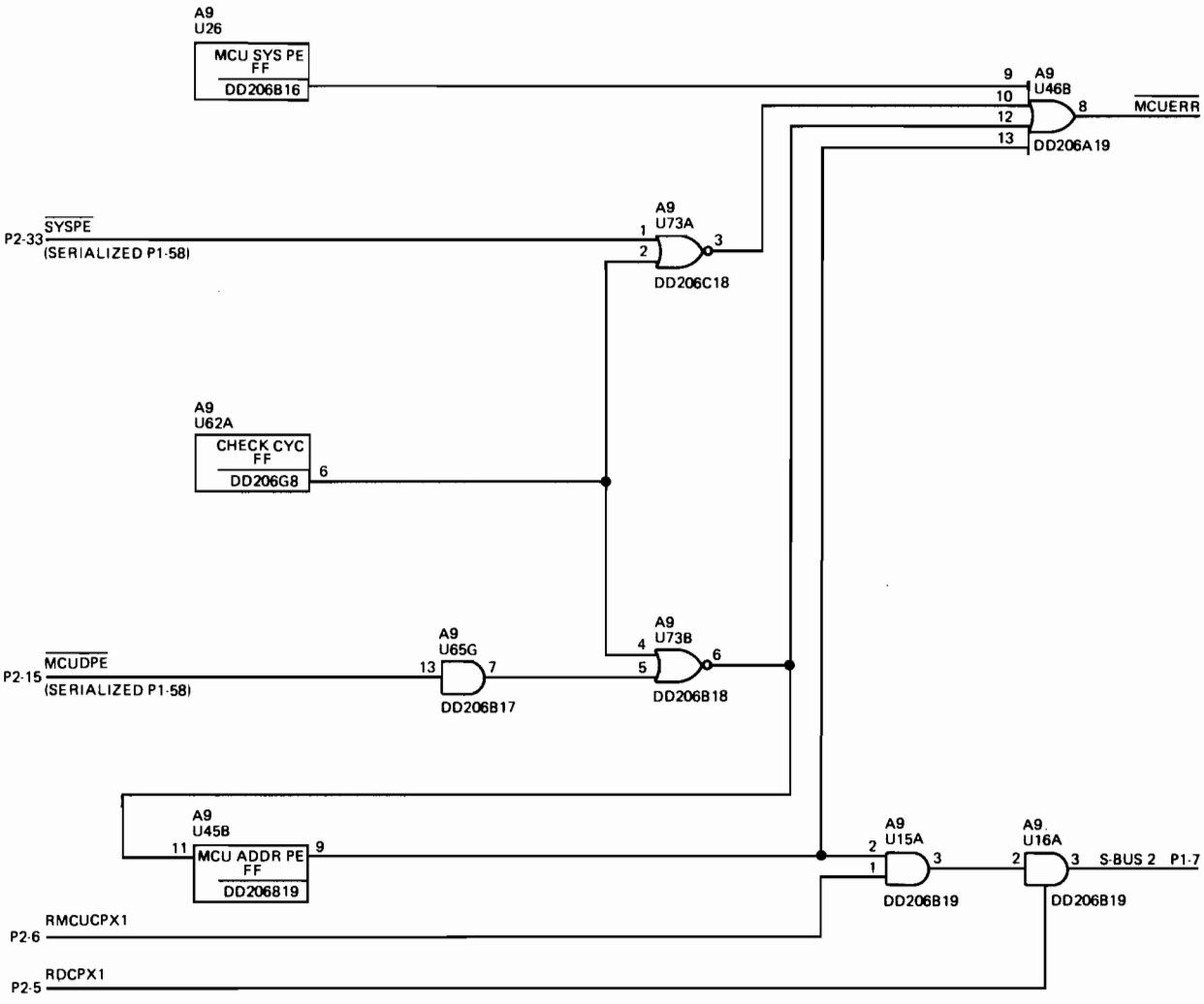
2184-184

Figure 4-45. MOP Lines Servicing Diagram



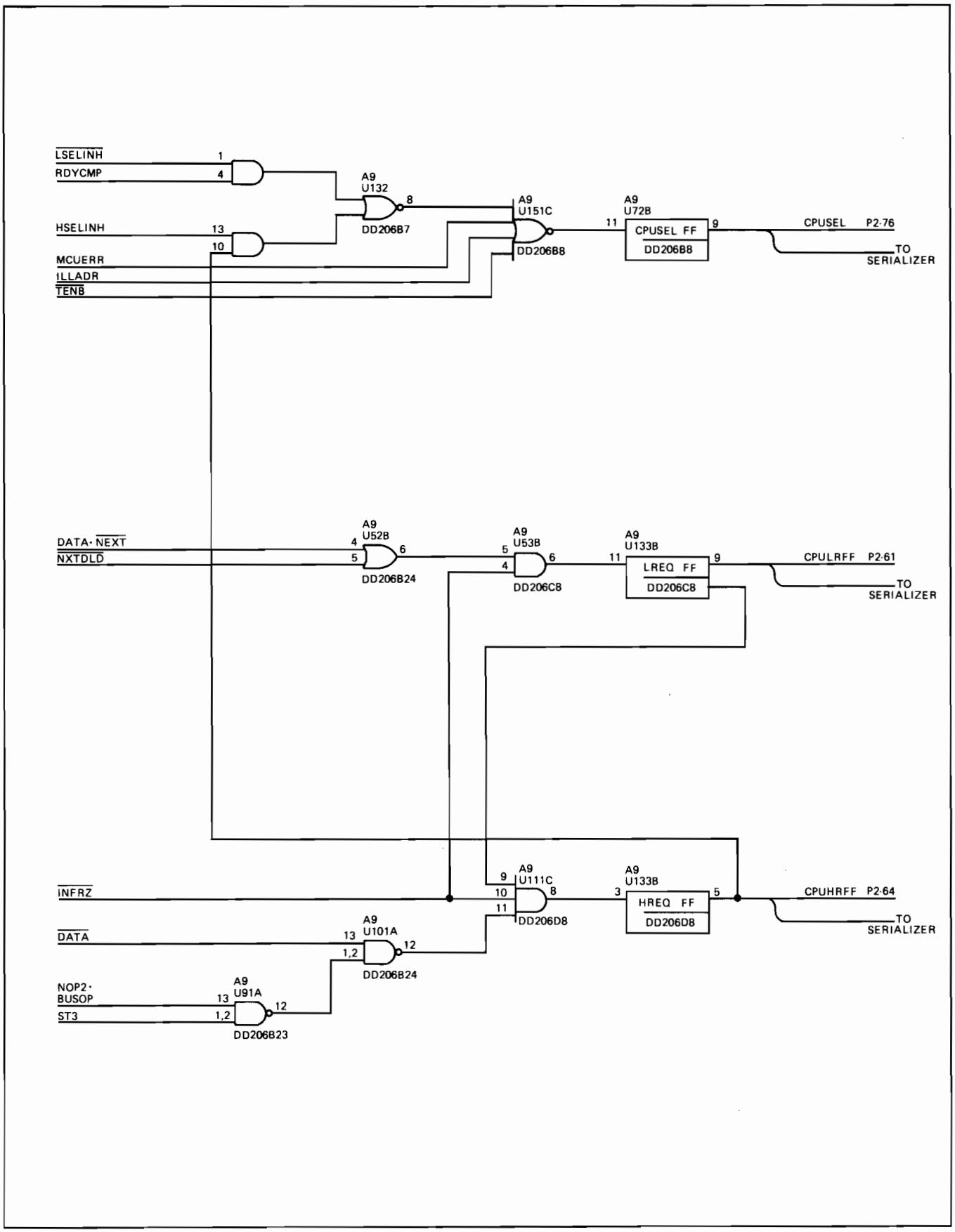
2184-186

Figure 4-46. SYSPRTY Servicing Diagram



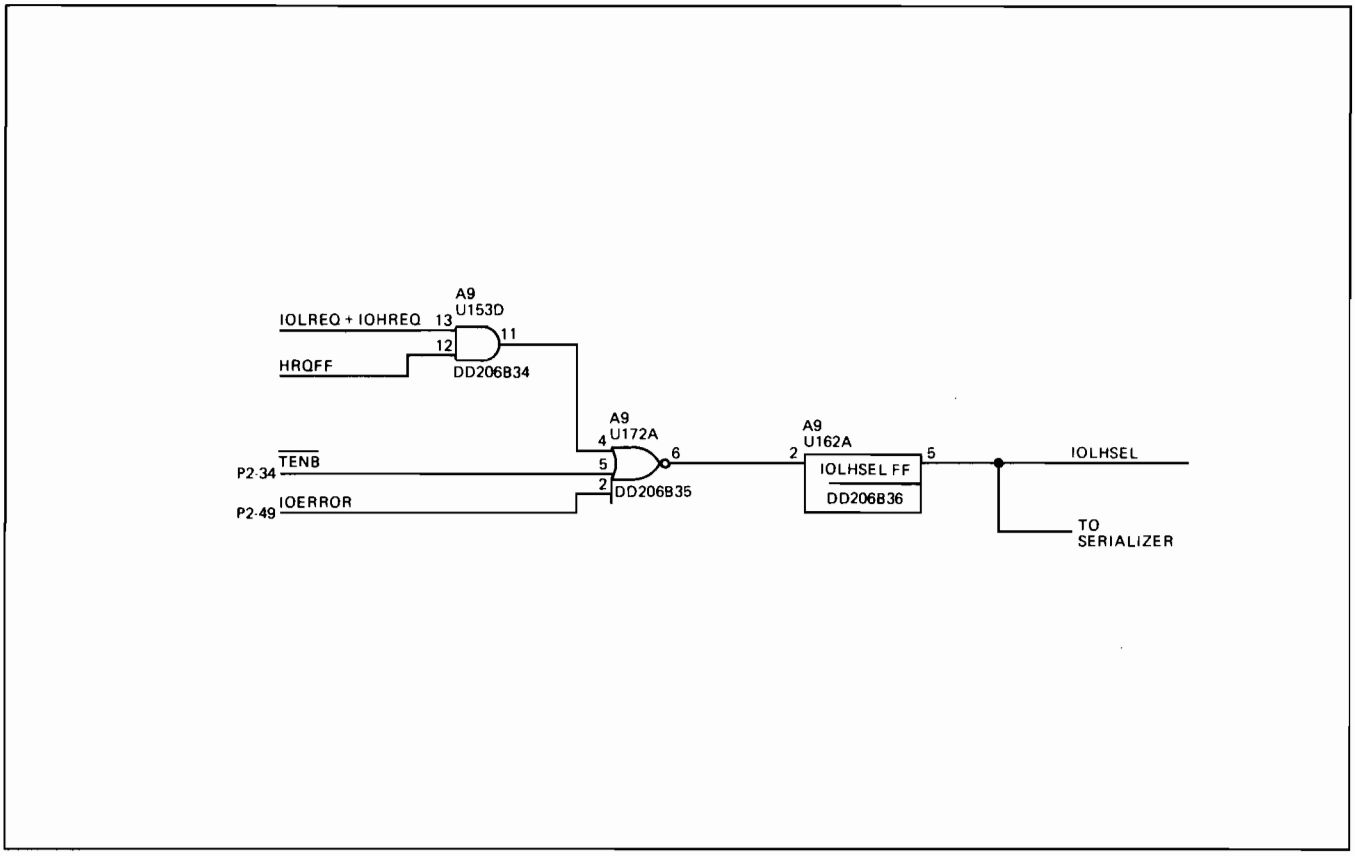
2184-182

Figure 4-47. SYSPE and MCUDPE Servicing Diagram



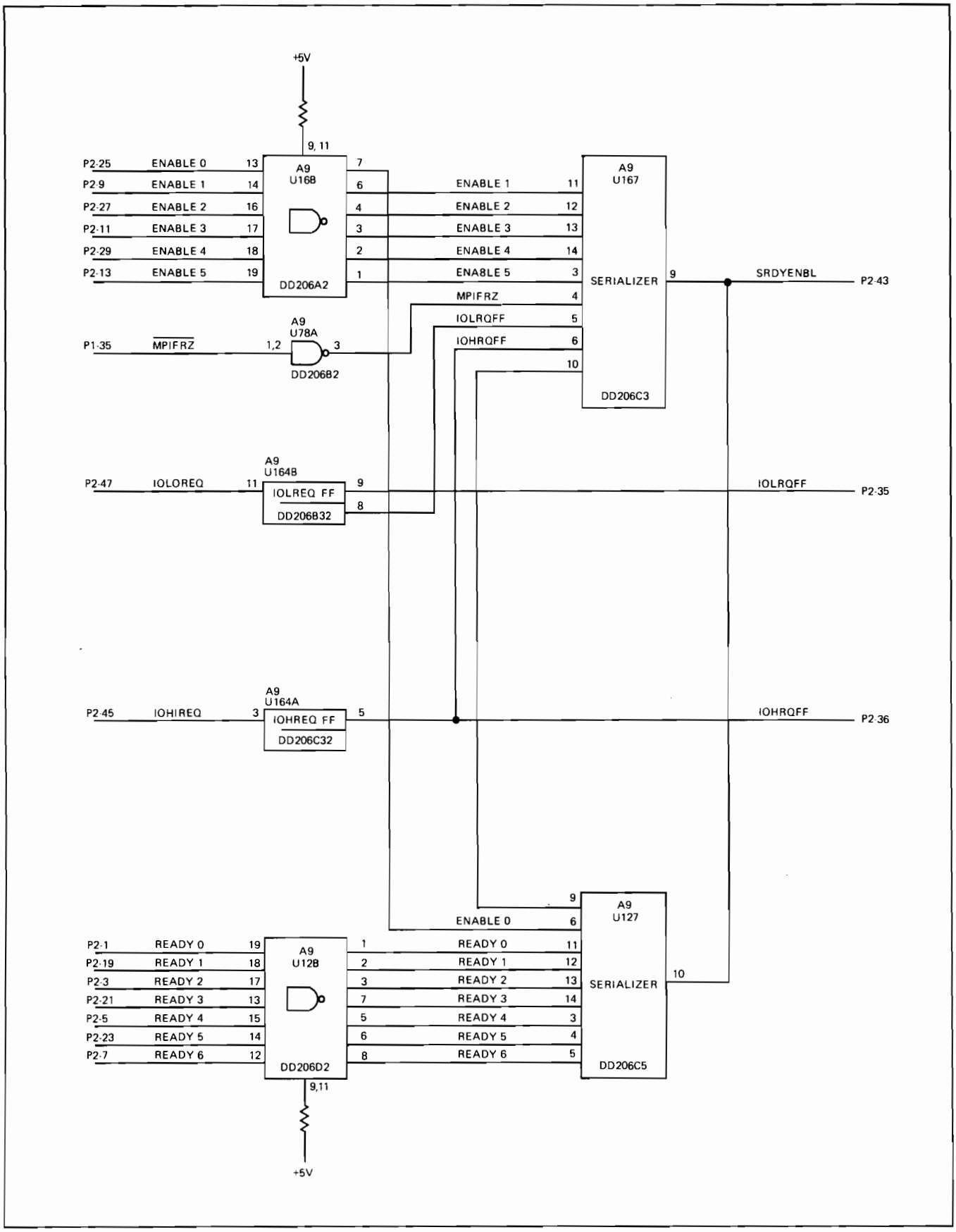
2184-181

Figure 4-48. CPUSEL, CPULRFF, CPUHRFF Servicing Diagram



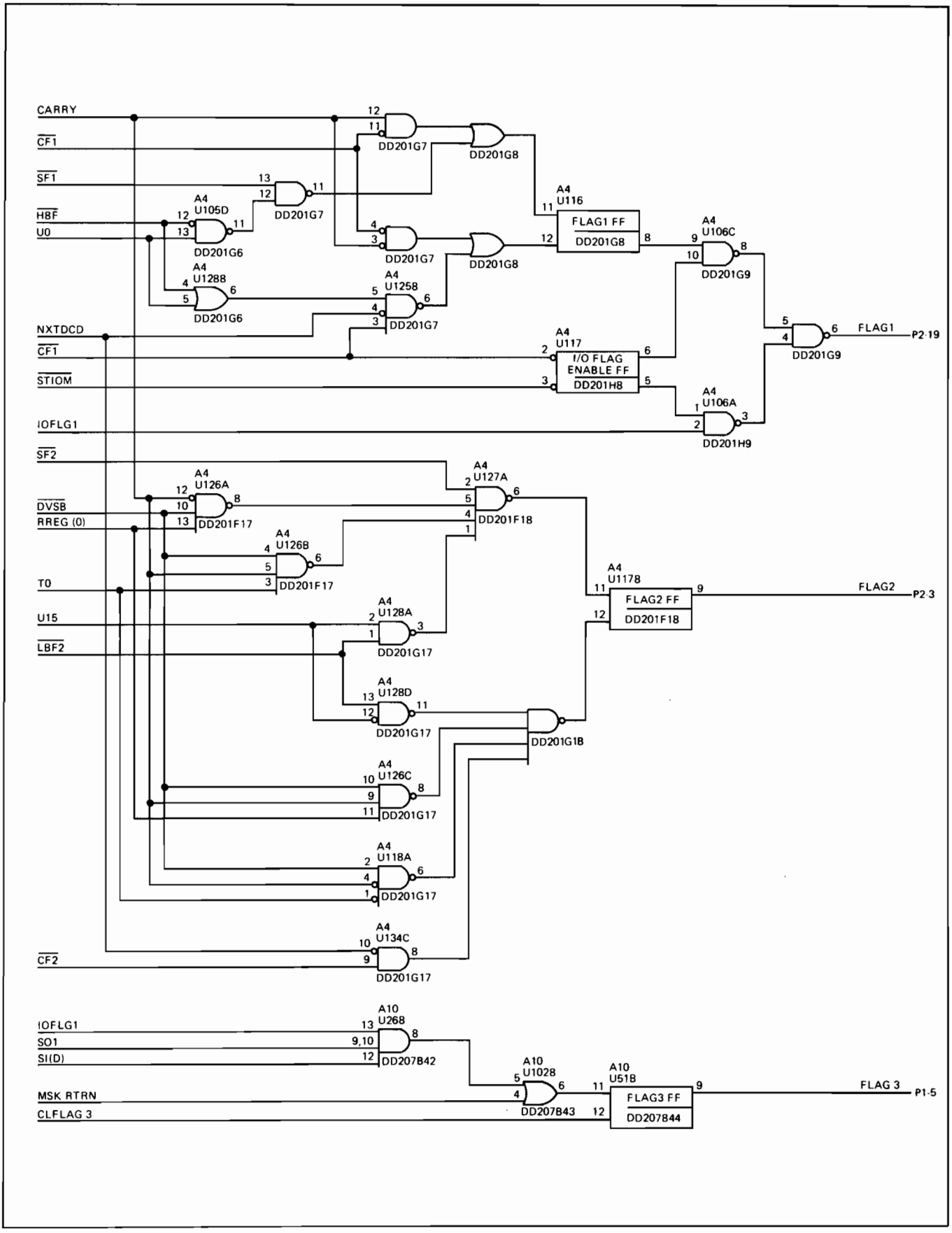
2184-199

Figure 4-49. IOSELECT Servicing Diagram



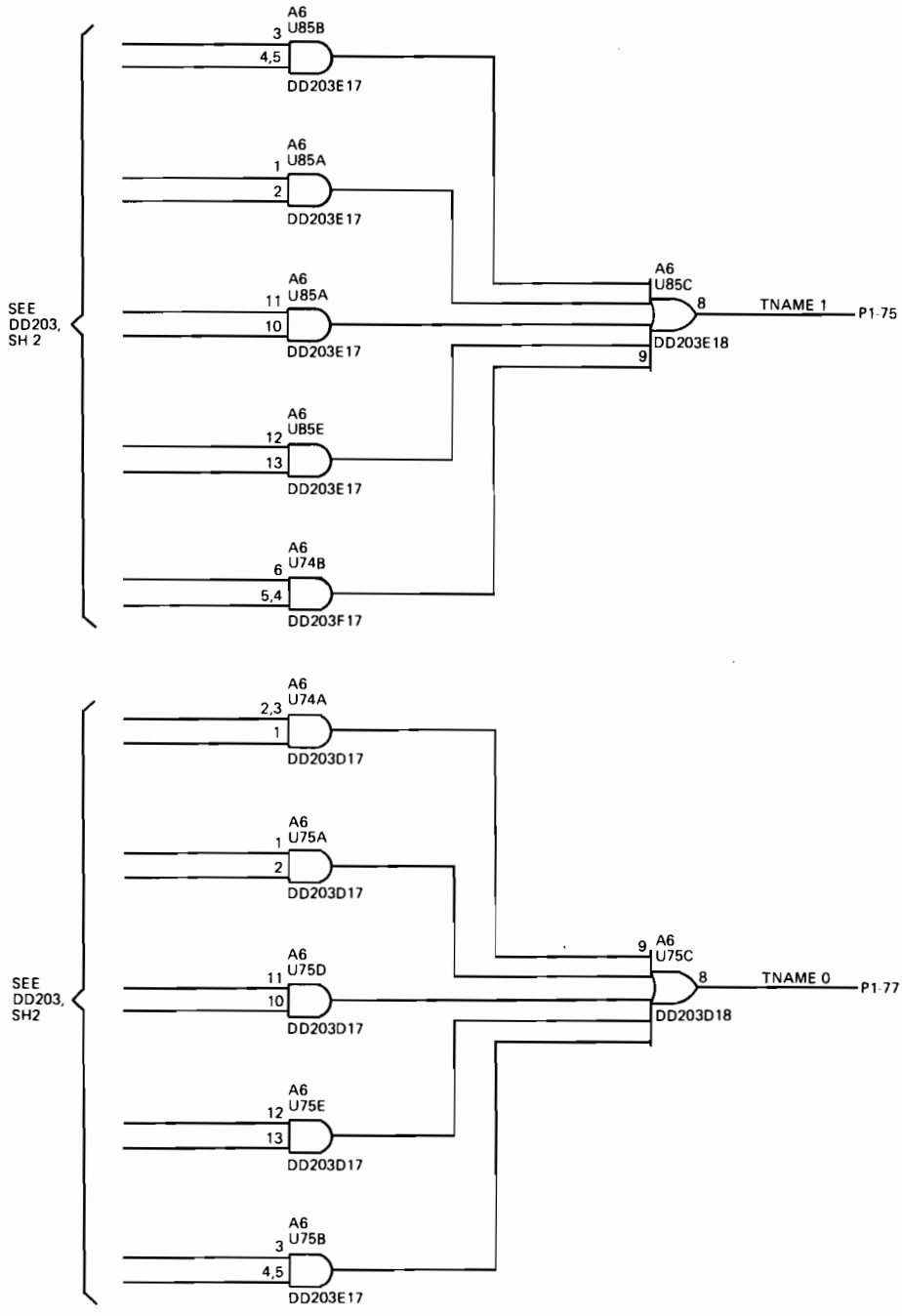
2184-179

Figure 4-50. ENABLE, READY, MPIFRZ, IOLOREQ, IOHREQ and Serializer B Servicing Diagram



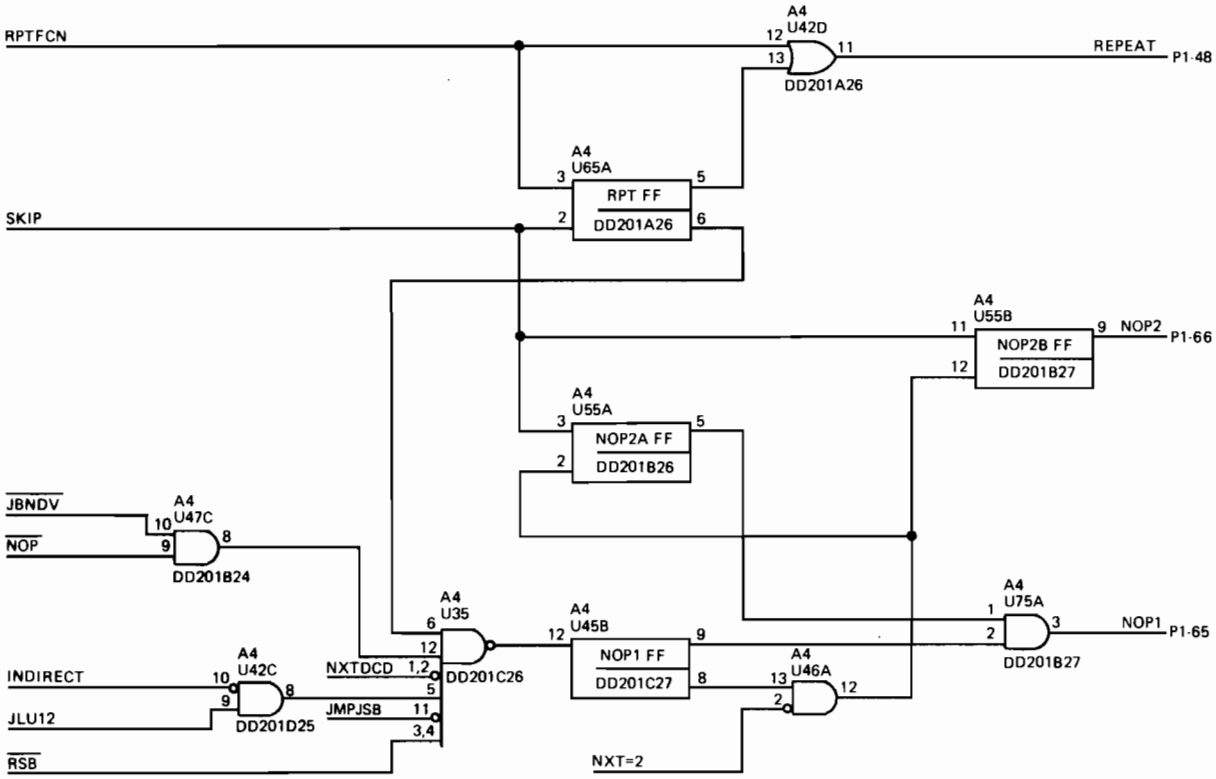
2184-200

Figure 4-51. Flag 1, Flag 2, Flag 3 Servicing Diagram



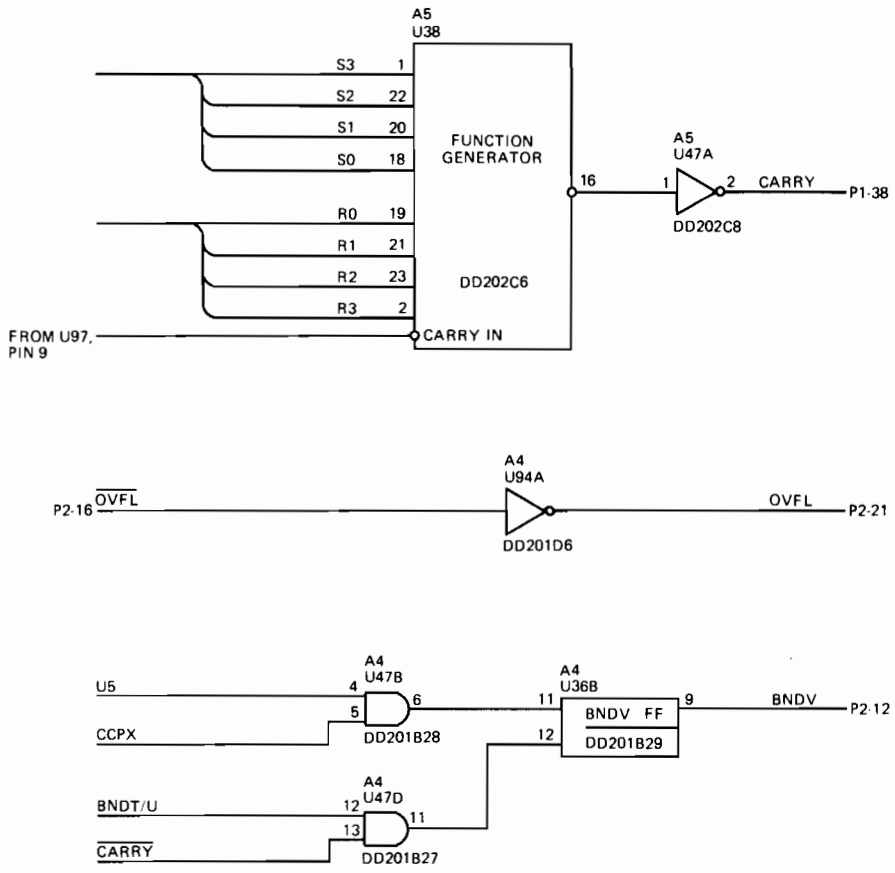
2184-198

Figure 4-52. TNAME (0:1) Servicing Diagram



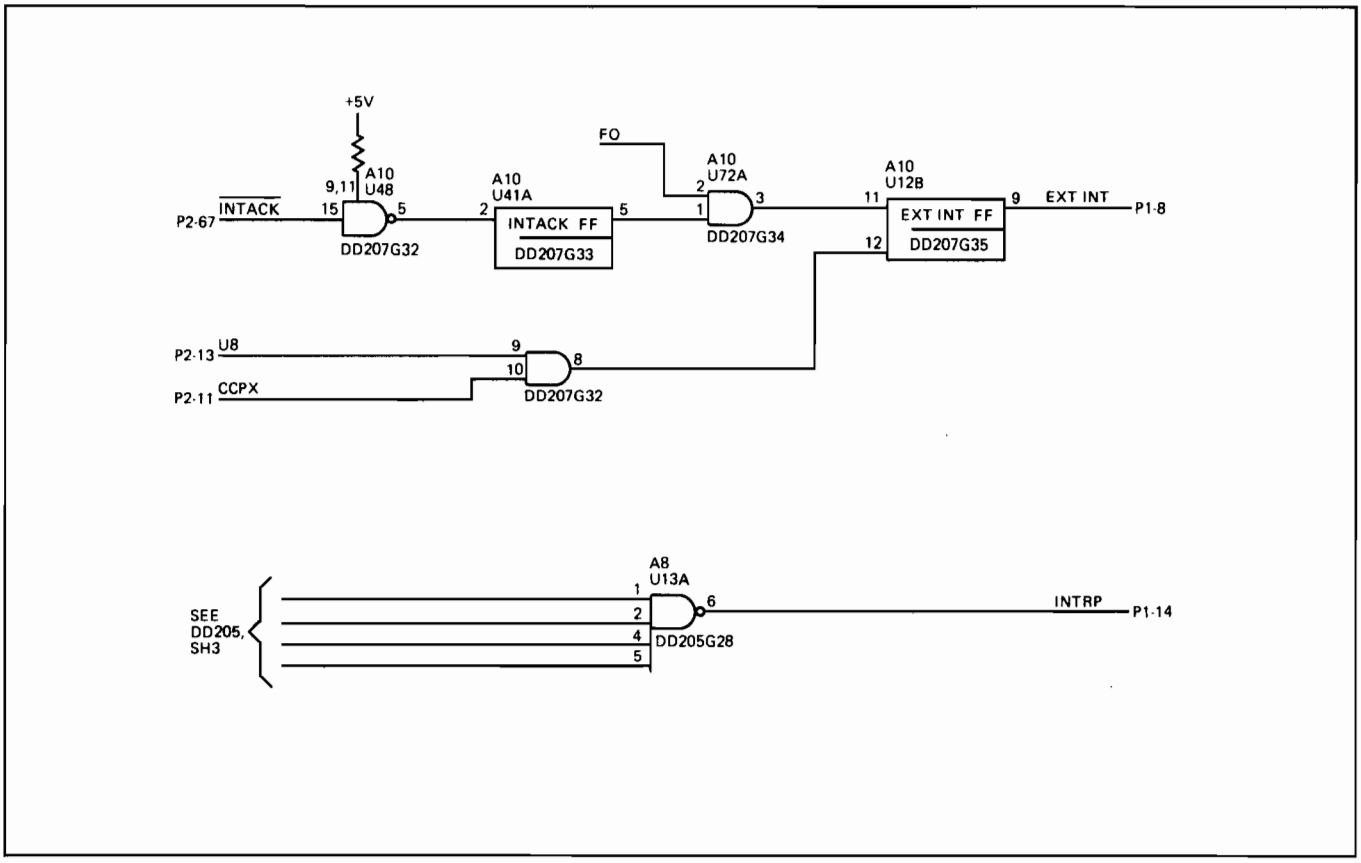
2184-201

Figure 4-53. NOP1, NOP2, REPEAT Servicing Diagram



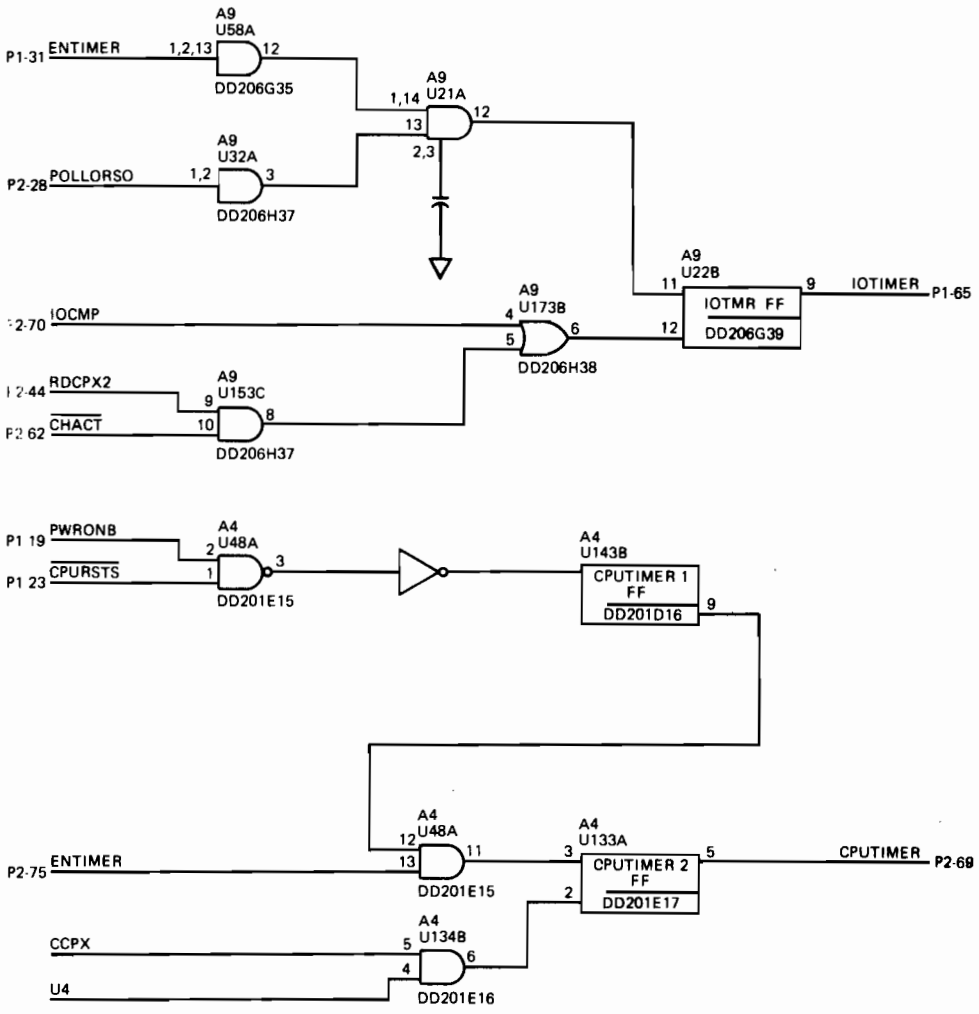
2184-202

Figure 4-54. ALU CARRY, ALU OVFL, and BNDV Servicing Diagram



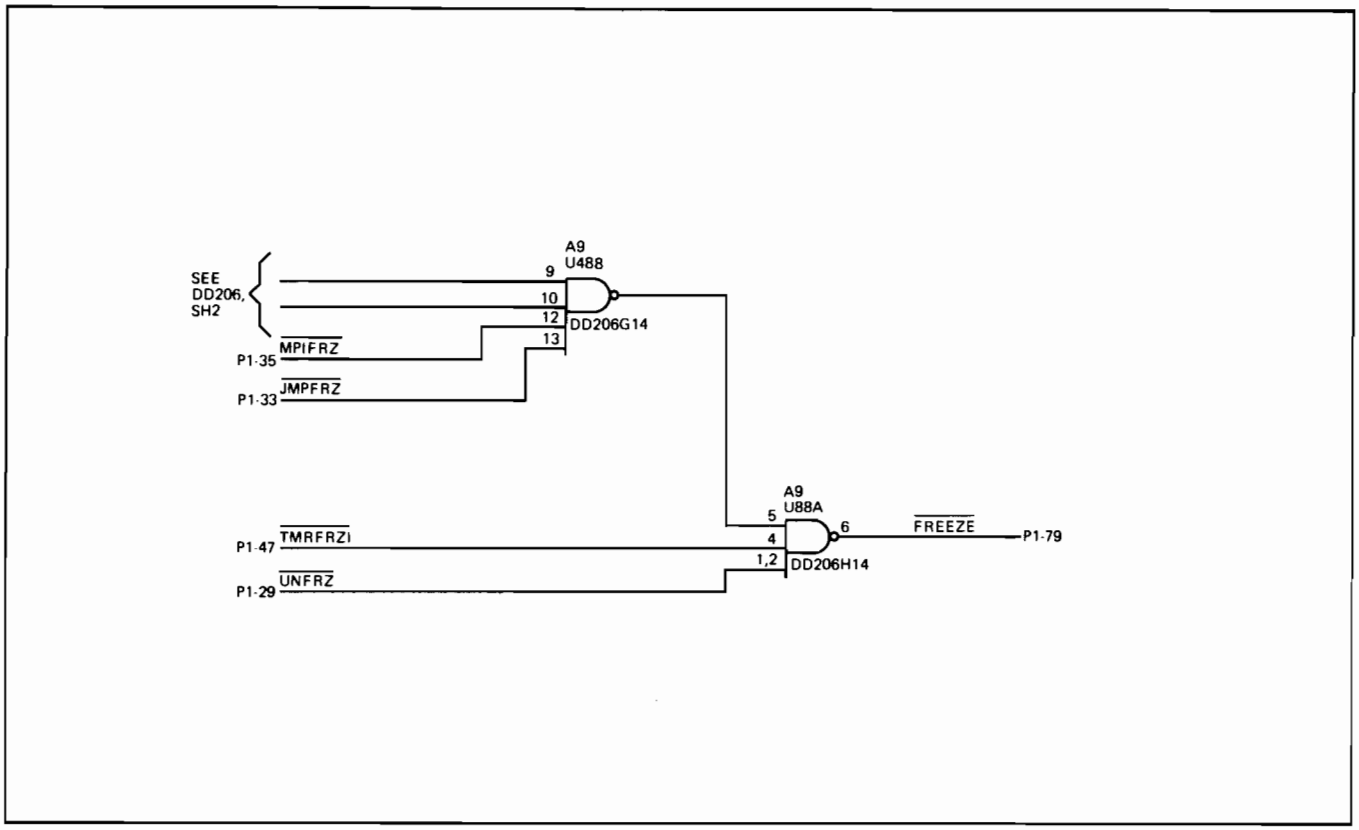
2184-203

Figure 4-55. EXT INT and INTRP Servicing Diagram



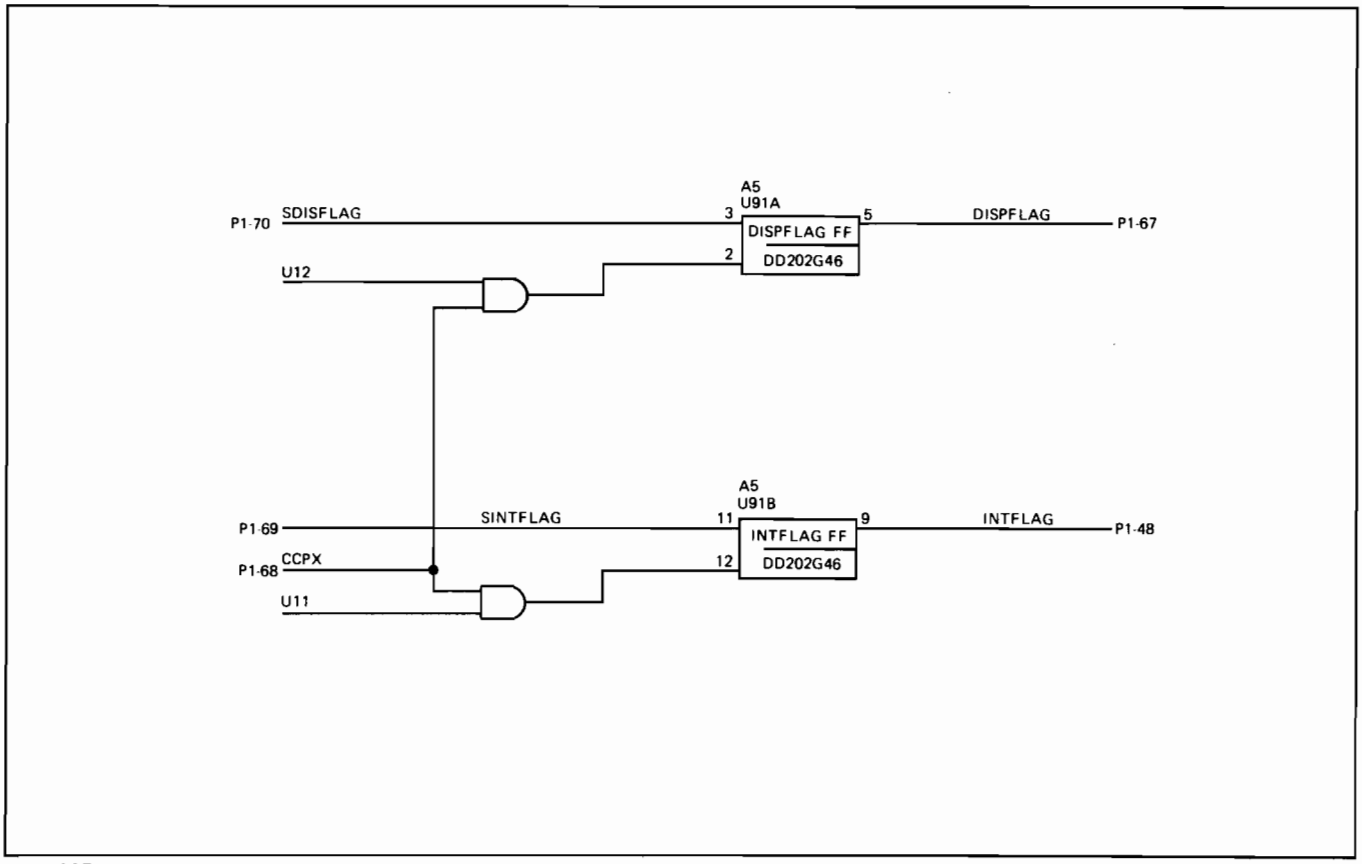
2184-205

Figure 4-56. IOTIMER and CPUTIMER Servicing Diagram



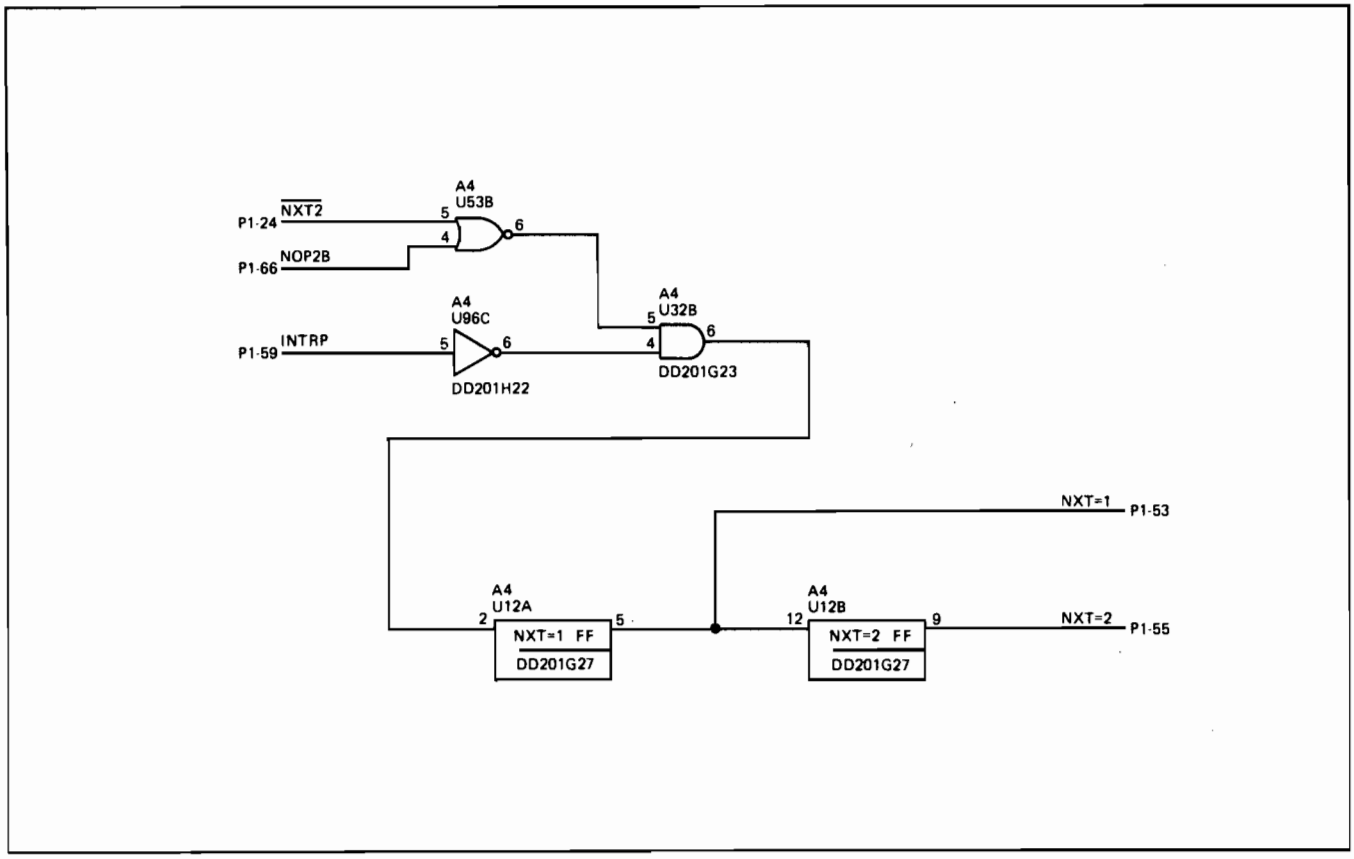
2184-204

Figure 4-57. Freeze Servicing Diagram



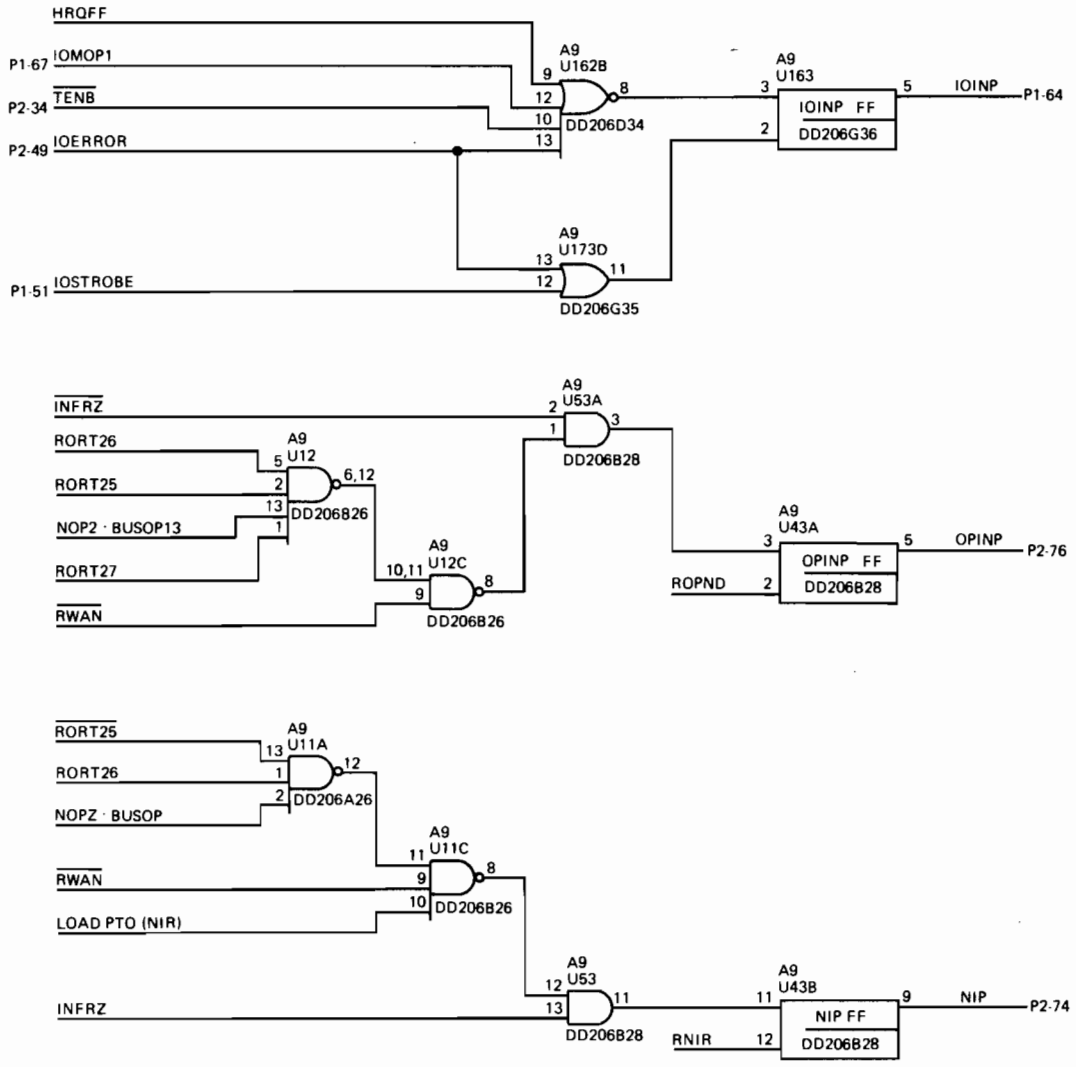
2184-197

Figure 4-58. DISPFLAG and INTFLAG Servicing Diagram



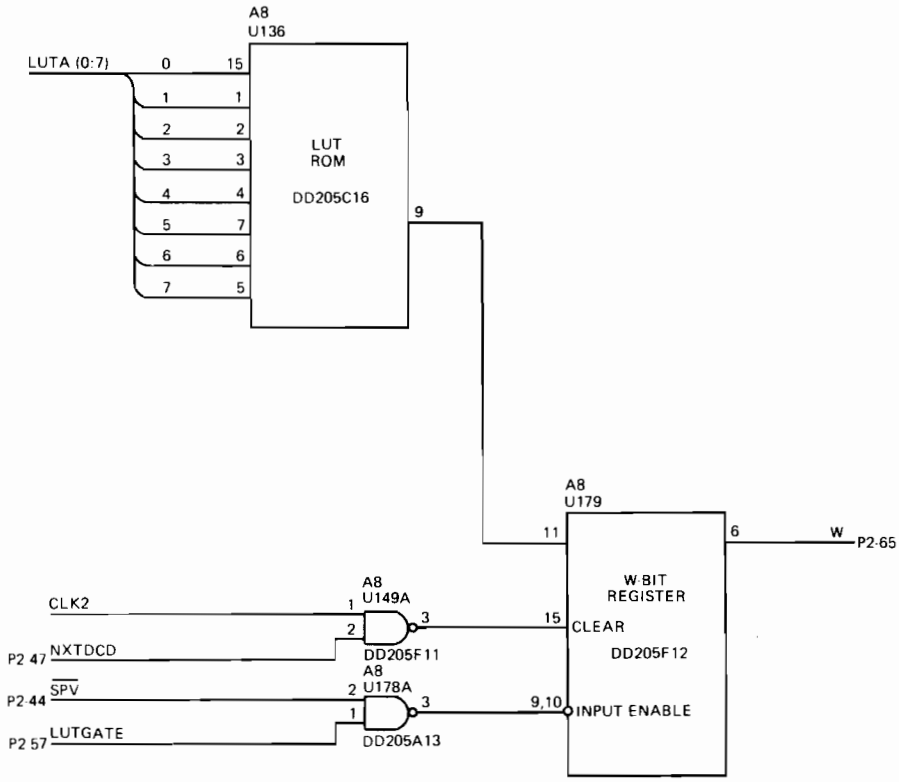
2184-196

Figure 4-59. NXT = 1 and NXT = 2 Servicing Diagram



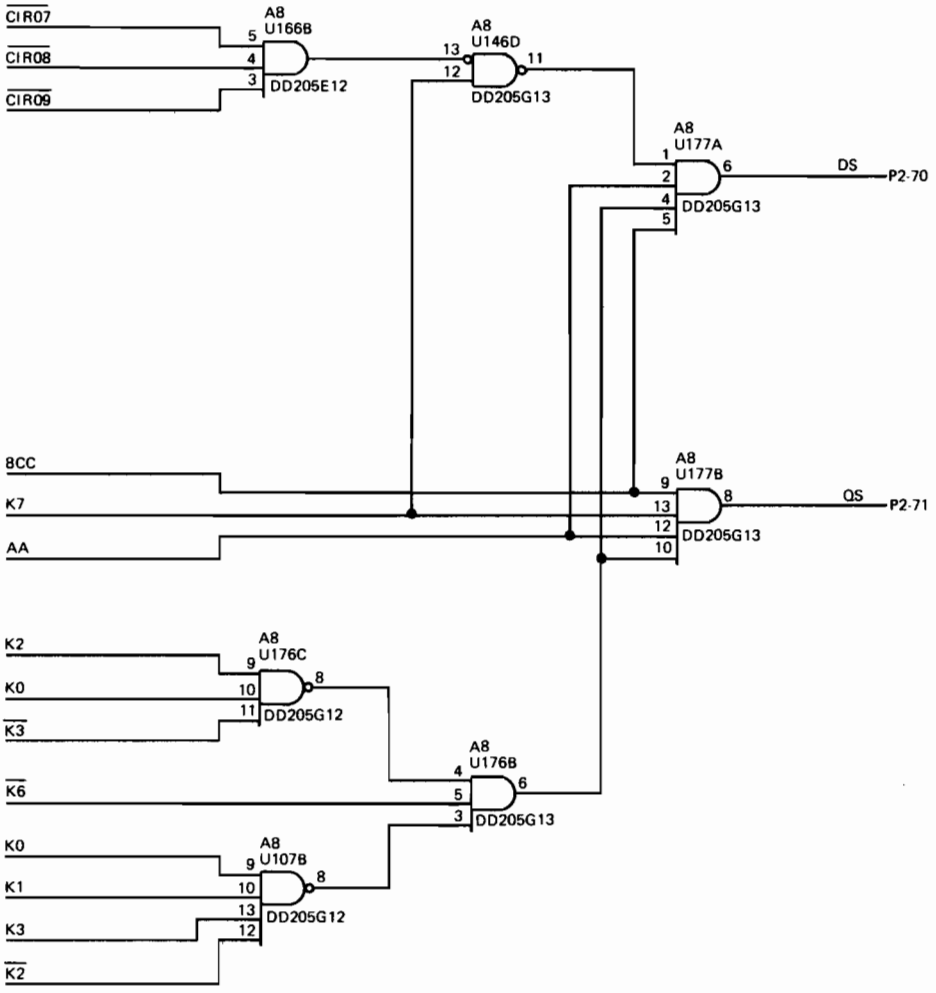
2184-195

Figure 4-60. IOINP, OPINP, and NIP Servicing Diagram



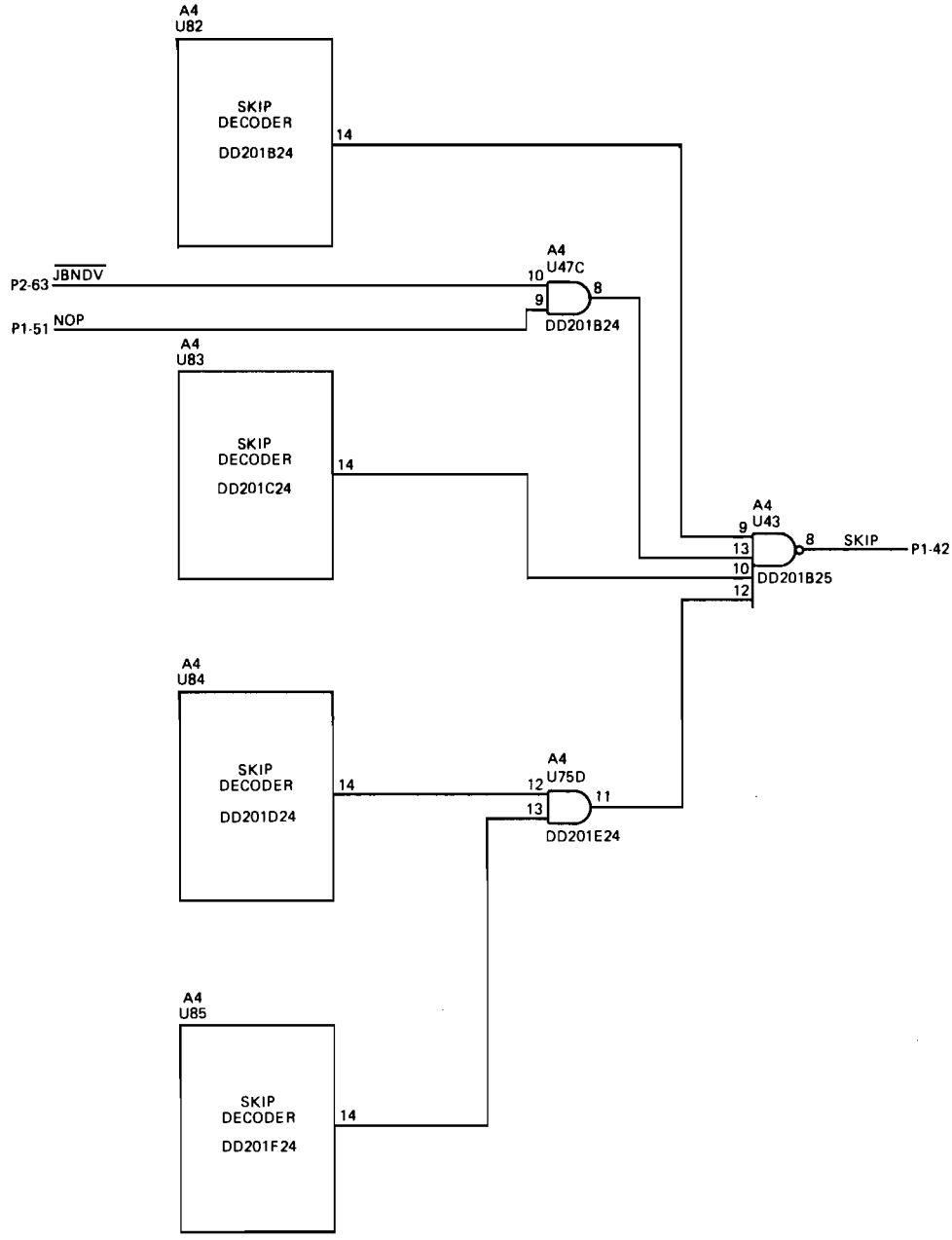
2184-194

Figure 4-61. W-Bit Servicing Diagram



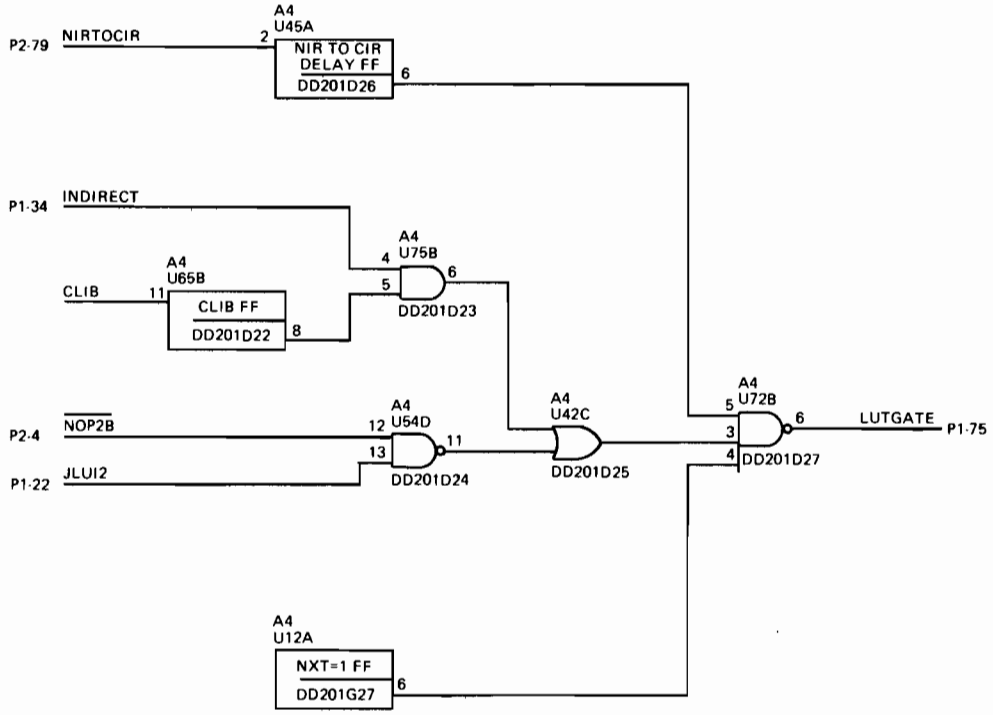
2184-193

Figure 4-62. DS and QS Servicing Diagram



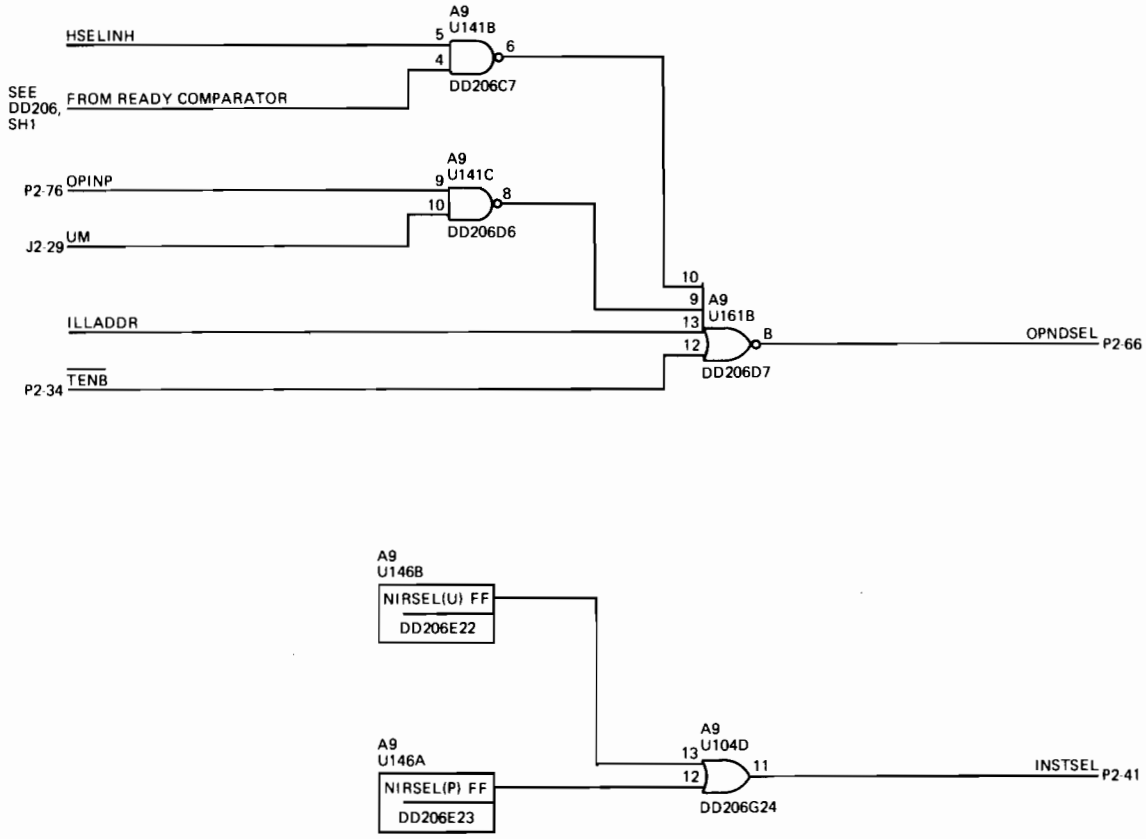
2184-192

Figure 4-63. Skip Servicing Diagram



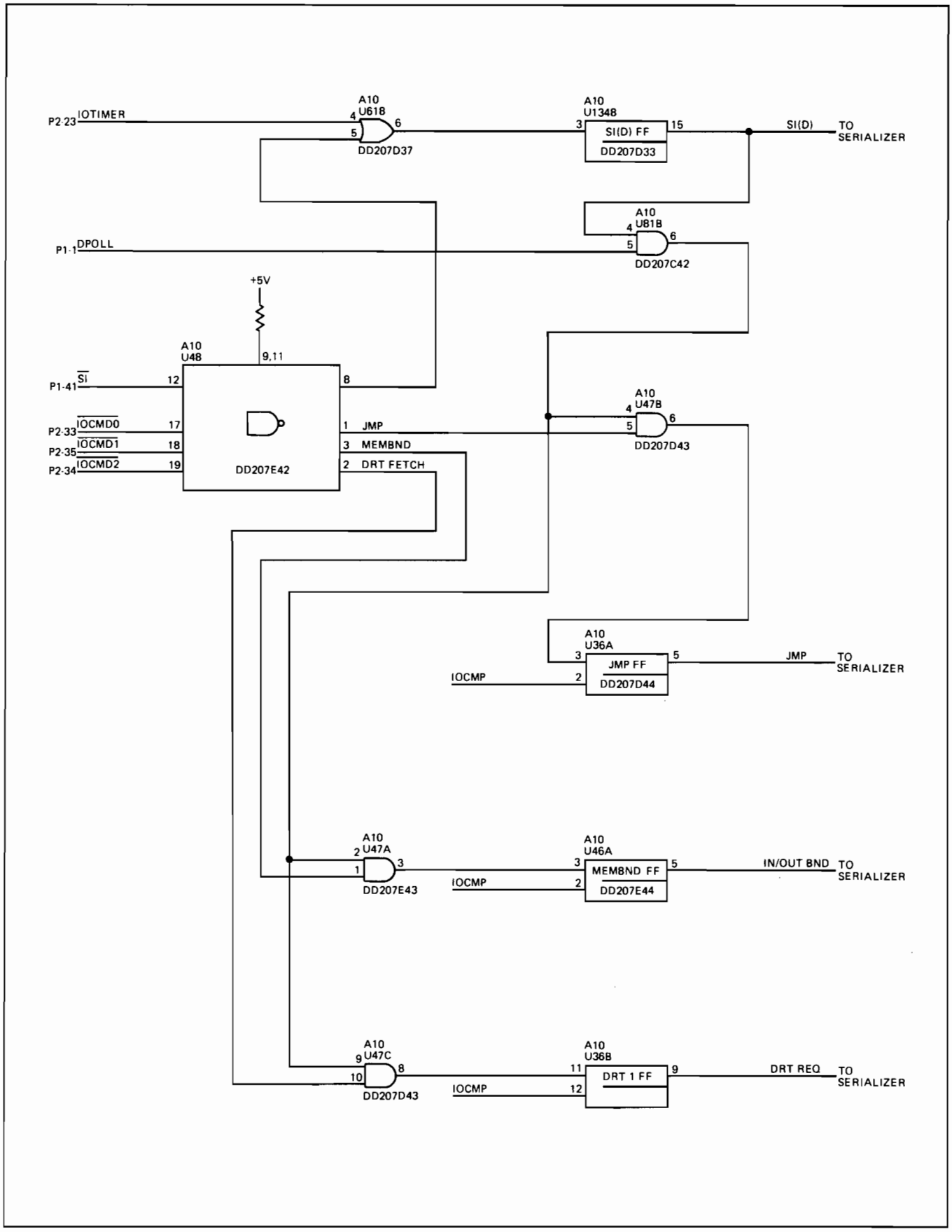
2184-206

Figure 4-64. LUTGATE Servicing Diagram



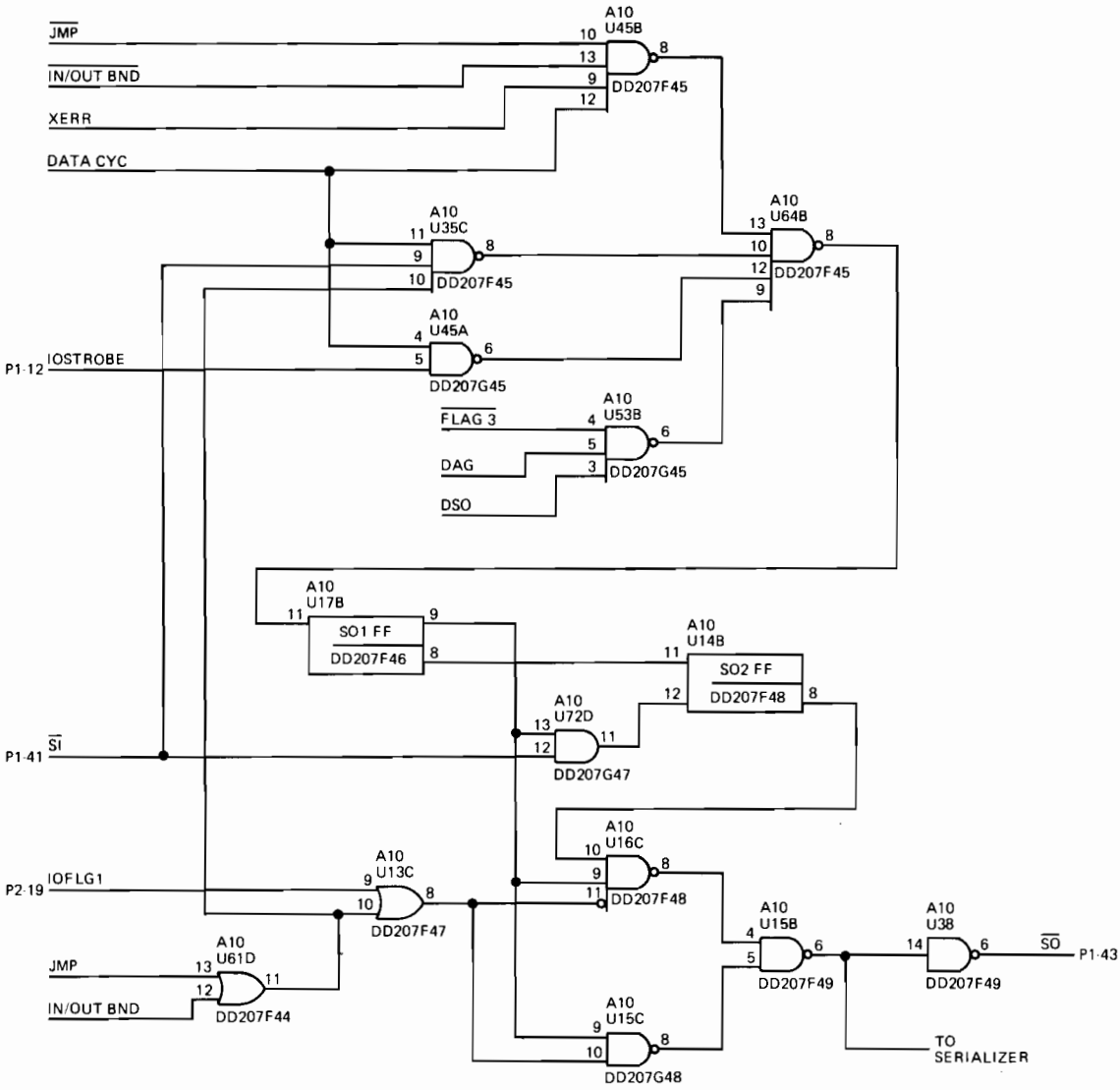
2184-214

Figure 4-65. OPNDSEL and INSTSEL Servicing Diagram



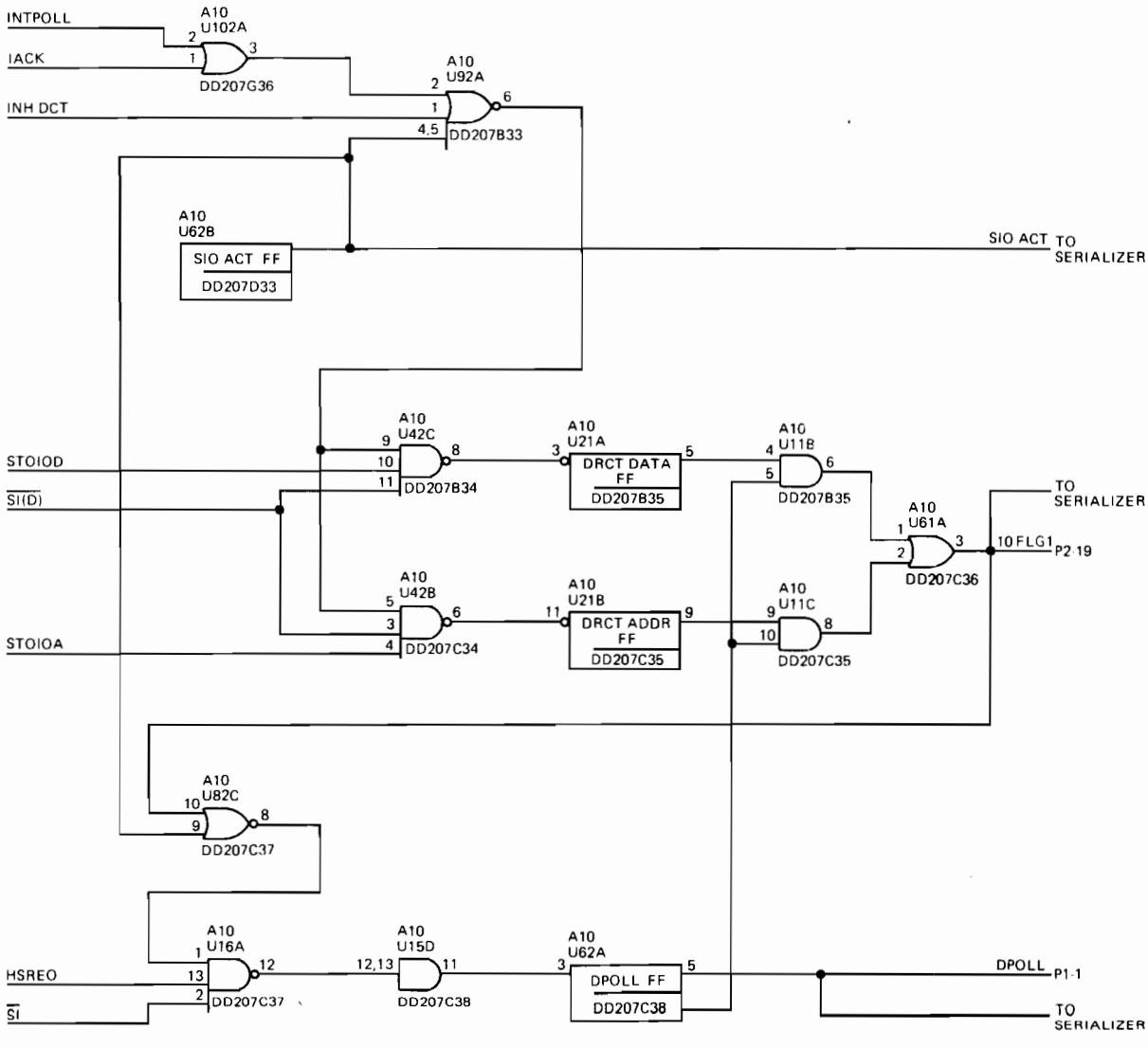
2184-212

Figure 4-66. SI, JMP, IN BND, and DRT REQ Servicing Diagram



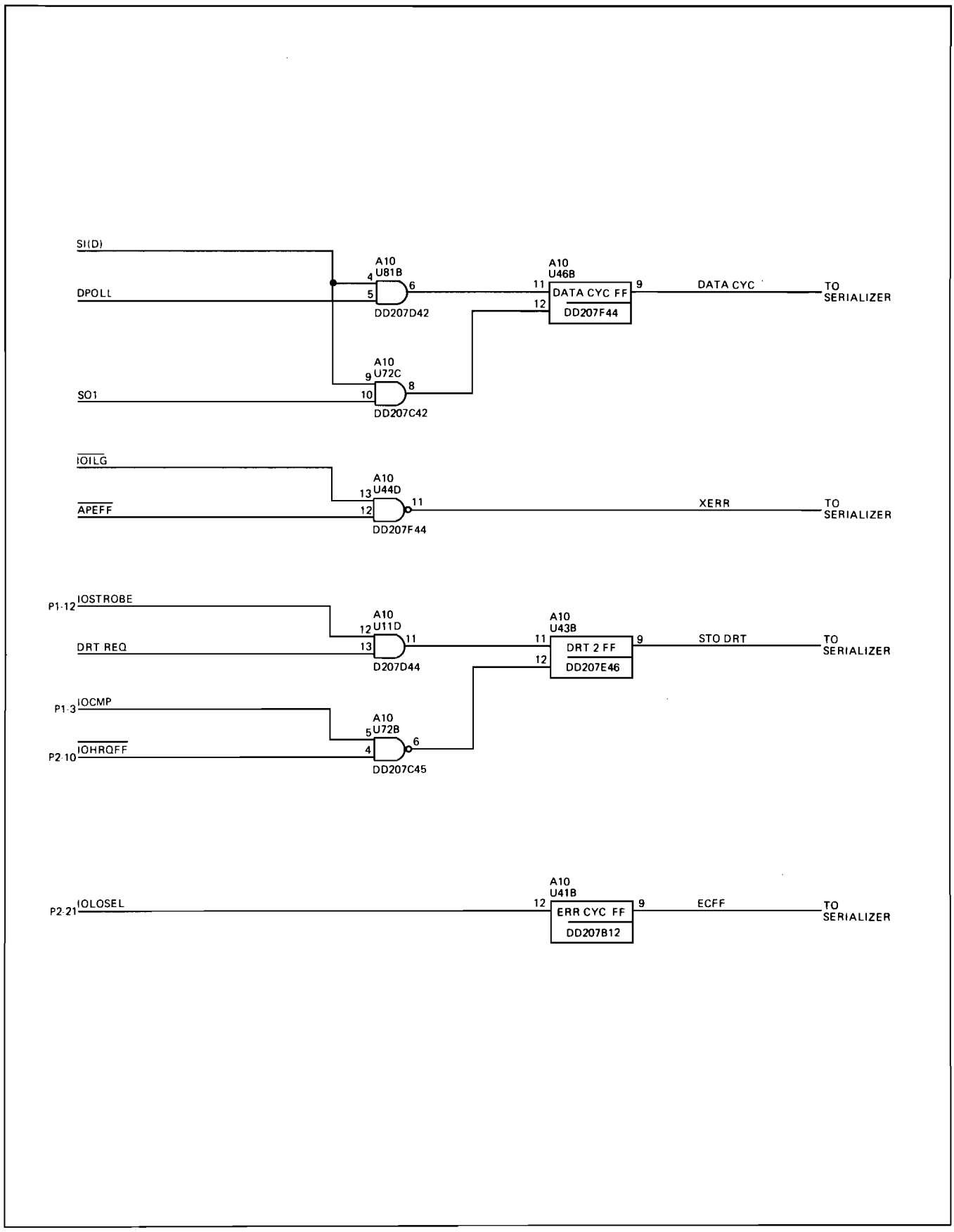
2184-211

Figure 4-67. Service Out Servicing Diagram



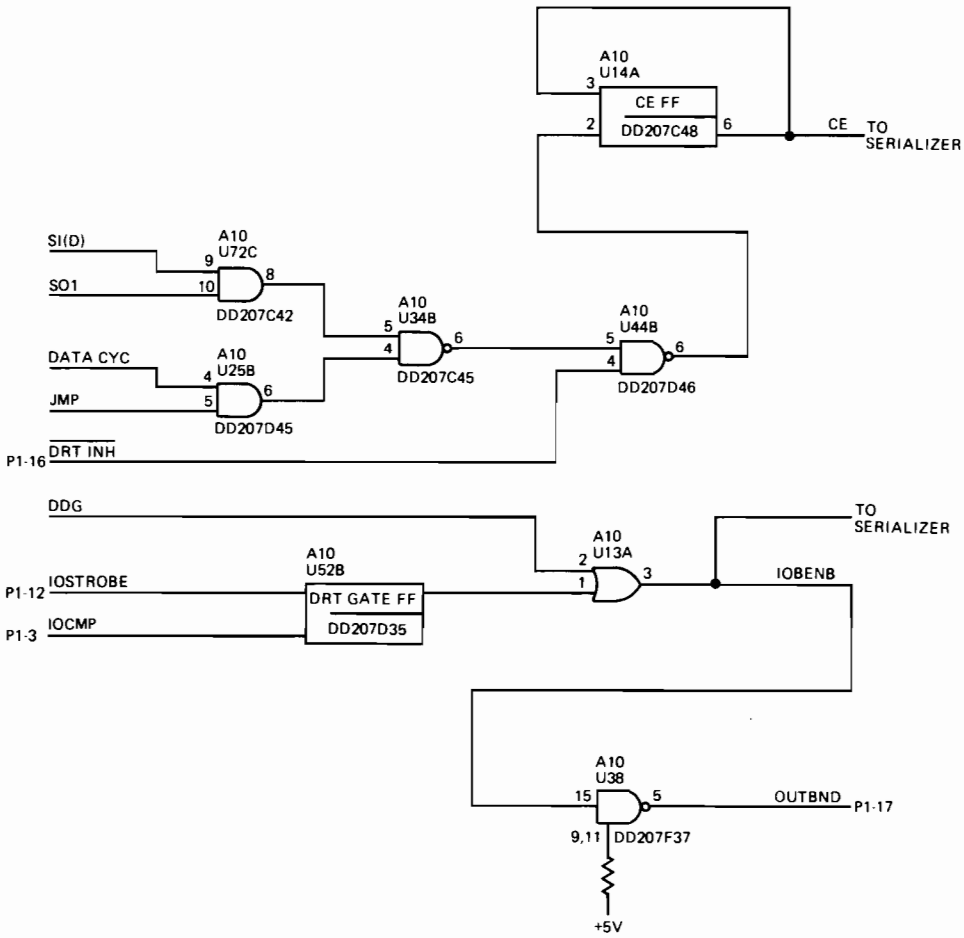
2184-210

Figure 4-68. SIOACT, IOFLG1, and DPOLL Servicing Diagram



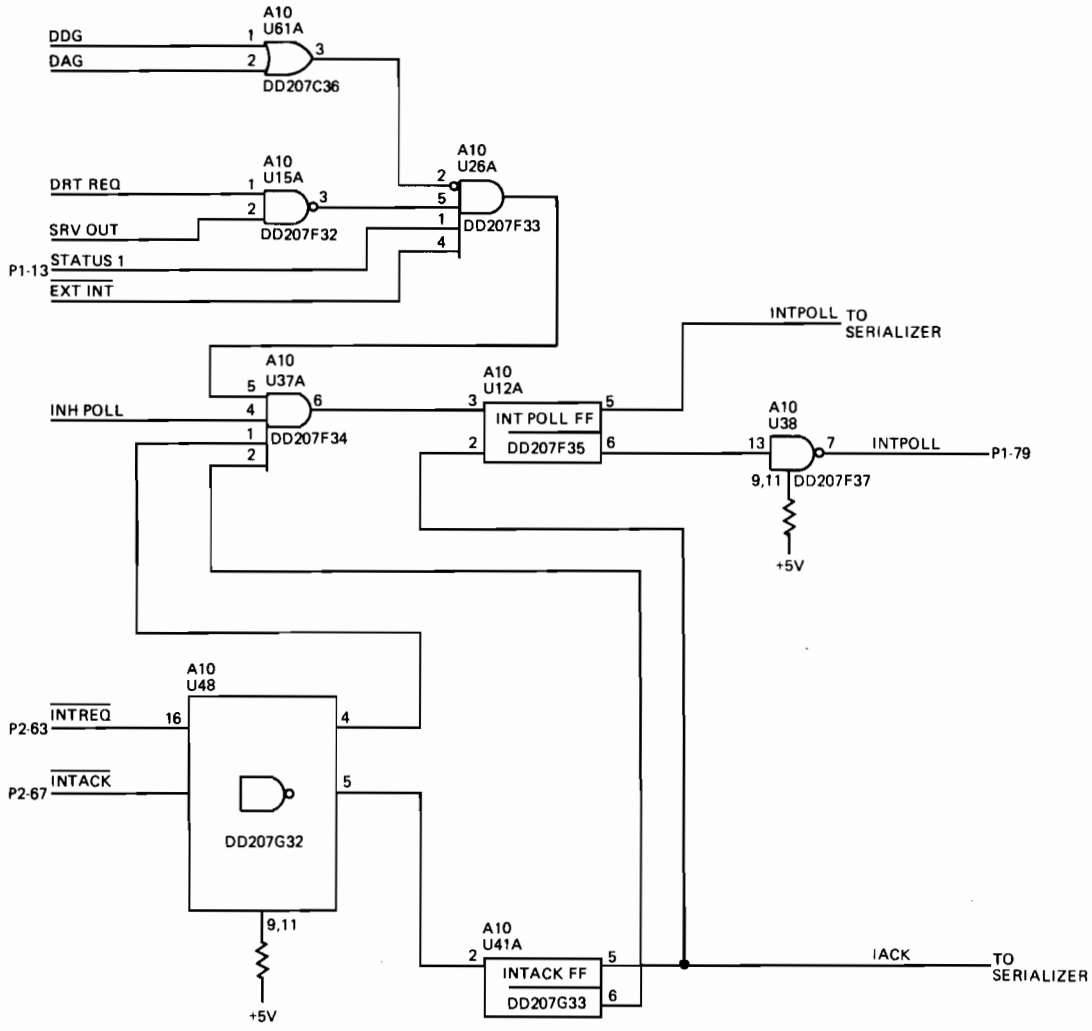
2184-213

Figure 4-69. DATA CYC, XERR, STO DRT, and ECFF Servicing Diagram



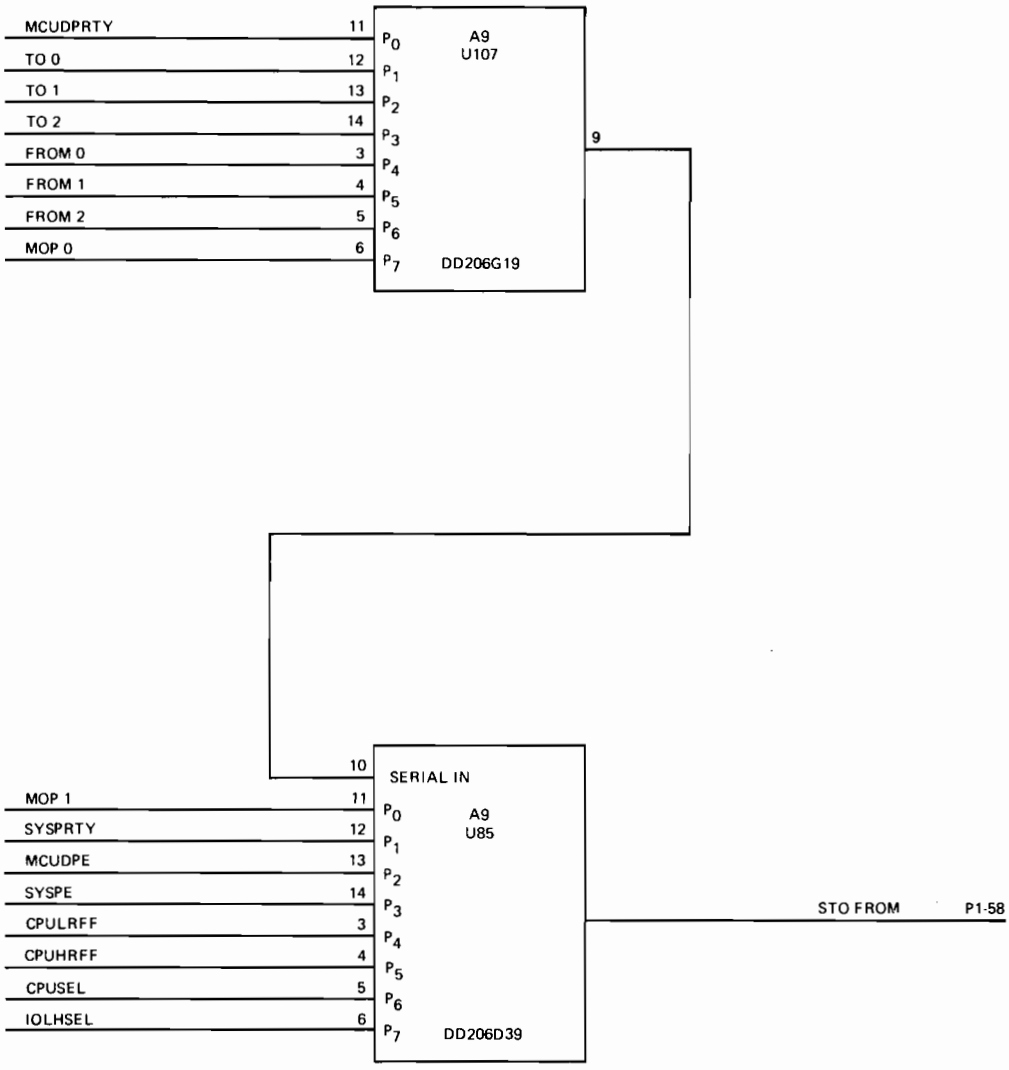
2184-209

Figure 4-70. CE and IOBENB Servicing Diagram



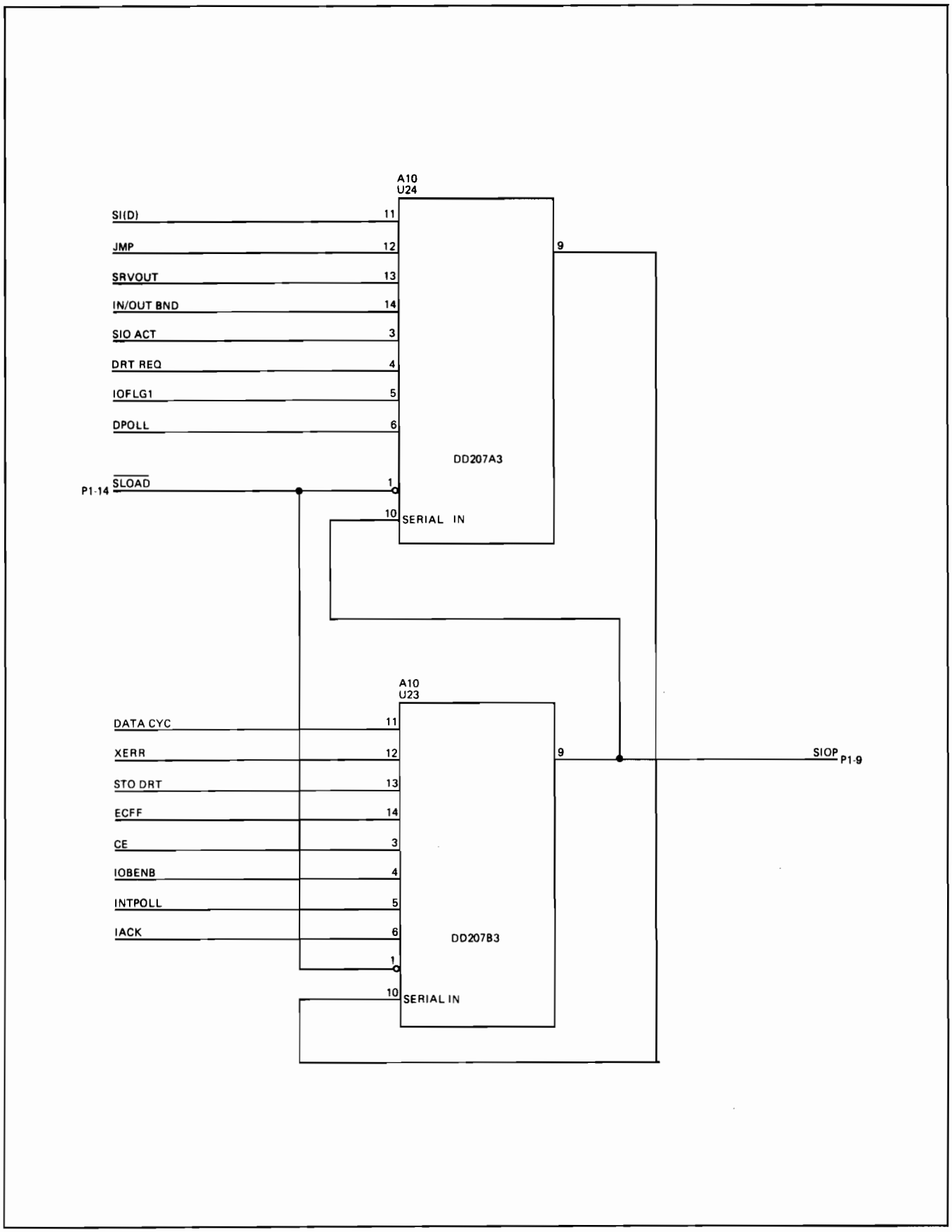
2184-208

Figure 4-71. INTPOLL and INTACK Servicing Diagram



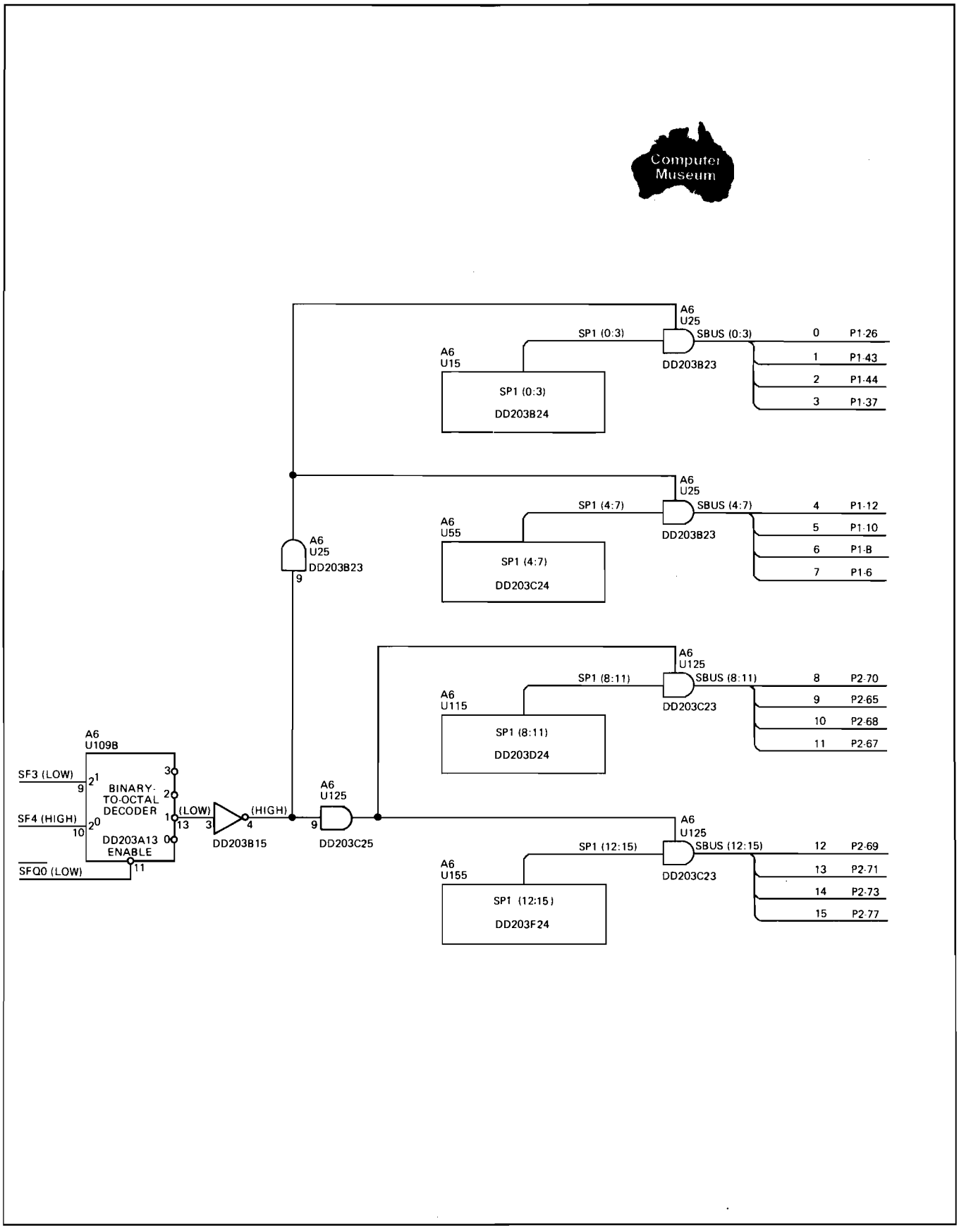
2184-180

Figure 4-72. Serializer A Servicing Diagram



2184-207

Figure 4-73. Serializer C Servicing Diagram



2184-177

Figure 4-74. SP1 Register Servicing Diagram

MICRODIAGNOSTICS

APPENDIX A

A-1 MICRODIAGNOSTICS

Stored in the microcode are diagnostics to test the CPU registers, memory, and the I/O channels. The diagnostics are accessed by a cold load procedure from the System Control Panel.

If an error occurs when running a diagnostic, the maintenance panel can be connected to the system to facilitate stepping through the microcode, or, the stand-alone diagnostic in the Manual of Diagnostics can be run to determine the malfunction.

A-2. CPU REGISTER TEST

This diagnostic tests the various CPU registers. To run this diagnostic, perform the following steps:

a. Set the SYSTEM SWITCH REGISTER to %000201.

b. Press the ENABLE and LOAD switches.

c. The program runs continuously until the HALT switch is pressed or until an error occurs.

When an error occurs, the CIR displays a coded register number that can be interpreted by referring to table A-1. Normal running time for a complete pass of the diagnostic is approximately one second.

NOTE

When the SYSTEM SWITCH REGISTER bit 8 is set to 0, all memory is initialized with a HALT %10 instruction (%030370) prior to executing the cold load. If bit 8 is set to 1, no initializing occurs prior to the cold load.

Table A-1. CPU Register Codes

CIR	REGISTER	CIR	REGISTER
00	SP1 (1) SEE NOTE	20	OPND (5)
01	PL (1)	21	DL (2)
02	Z (1)	22	SP2 (2)
03	X (1)	23	PB (2)
04	RD (R BUS) (1)	24	PCLK (2)
05	RC (R BUS) (1)	25	RD (R BUS) (2)
06	RB (R BUS))1)	26	RC (S BUS) (2)
07	RA (R BUS) (1)	27	RB (S BUS) (2)
10	SP0 (1)	30	RA (S BUS) (2)
11	CRTL (2)	31	CTRH (2)
12	P (2)	32	ABS (BANK) (3)
13	Q (2)	33	PB (BANK) (3)
14	DB (2)	34	DB (BANK) (3)
15	SM (2)	35	S (BANK) (3)
16	STA (4)		
17	SP3 (2)		

(1) Located on R-Bus PCA
(2) Located on S-Bus PCA
(3) Located on Skip and Special Field PCA
(4) Located on Skip and Special Field PCA and S-Bus PCA
(5) Located on Current Instruction Register PCA

NOTE

SP1 is the first register tested and the problem may not necessarily be in SP1 but somewhere previous in the data path (Store logic, Shifter, ALU, etc.).

A-3. MEMORY TEST

A memory configuration test is available from the microprogram for testing memory. Memory configuration test diagnostic time is approximately ten seconds.

To run the memory diagnostics, perform the following steps:

a. Set the SYSTEM SWITCH REGISTER, to %000200.
b. Press the ENABLE and LOAD switches.
c. Program runs until an error occurs.

When an error occurs the program pauses and the CIR contains the error data (lamp on = error bit). By pressing the RUN/HALT switch, the CIR then contains the address information shown in table A-2. The test should be continued so all memory is tested before any repairs are made. The test is terminated by pressing the HALT switch.

Table A-2. CIR Address Information

CIR BIT	FUNCTION
0:3	Address bits 0:3
6,7	Bank number
10:14	CPX1 register bits 2:6
10	(2) Illegal address
11	(3) CPU timer
12	(4) System Parity Error
13	(5) Address Parity Error
14	(6) Data Parity Error
15	Address bit 15

A-4. I/O TEST

A Test Input/Output (TIO) instruction is executed on each I/O device number (3 through %177) in sequence. Only those device numbers with a device connected will respond; empty device numbers are skipped. To run the I/O test, perform the following steps:

NOTE

If the HP 30354A Maintenance Panel is connected to the system, the TIMERS switch must be set to ENABLE.

a. Set the SYSTEM SWITCH REGISTER to %000202.

b. Press the ENABLE and LOAD switches. (When an existing device number is encountered, the program pauses with the device number in CIR. The RUN lamp will be illuminated.)

c. Press the RUN/HALT switch. CIR then displays the device status, the RUN lamp will be extinguished.

d. Press the RUN/HALT switch. Steps b and c are repeated until all device numbers have been interrogated. Diagnostic is finished when CIR displays %000200. The RUN indicator will be extinguished.

HEWLETT hp PACKARD

CERTIFICATION

The Hewlett-Packard Company certifies that this instrument was thoroughly tested and inspected and found to meet its published specifications when it was shipped from the factory. The Hewlett-Packard Company further certifies that its calibration measurements are traceable to the U.S. National Bureau of Standards to the extent allowed by the Bureau's calibration facility.

HEWLETT hp PACKARD

MANUAL PART NO. 30001-90003
MICROFICHE PART NO. 30001-90020

PRINTED IN U.S.A.