IBM Field Engineering

Manual of Instruction

System/360 Model 40

Functional Units

SY22-2843-1

Preface

This manual describes functional units of the IBM 2040 Processing Unit. Storage and functional units that control data flow are explained in detail. An explanation of the control and documentation for microprogramming is also contained in this manual.

A knowledge of computer fundamentals, System/360 philosophy, and the Model 40 data paths are useful in understanding this manual. The following manuals supply this information:

1. *System/360 Model 40 Comprehensive Introduction, Field Engineering Manual of Instruction,* Order Number 223-2840-0

2. *System/360 Model 40 Theory of Operation, Field Engineering Manual of Instruction,* Order Number 223-2844-0.

3. *System/360 Model 40 Power Supplies, Features, and Appendix, Field Engineering Manual of Instruction,* Order Number 223-2845-0

4. *System/360 Model 40, 2040 Processing Unit, Field Engineering Maintenance Manual,* Order Number 223-2841-0

5. *System/360 Model 40, 2040 Processing Unit, Field Engineering Diagrams Manual,* Order Number 223-2842-0

This manual is written to engineering change level 254814 for ALD's and CLD level 255263.

All three-digit figure references in this publication refer to figures in the Diagrams Manual.

The words "memory" and "storage" may be used interchangeably between this publication and the machine ALD's.

Fifth Edition (March, 1970)
This edition, Order Number SY22-2843-1, is a reprint of Form Y22-2843-0 incorporating changes released in the following FE supplement:

FES Number	*Date*
Y22-6784	January 30, 1969

Changes are periodically made to the specifications herein; any such change will be reported in subsequent revisions or FE Supplements.

This manual has been prepared by the IBM Systems Development Division, Product Publications, Dept. B96, PO Box 390, Poughkeepsie, N.Y. 12602

A form is provided at the back of this publication for readers' comments. If the form has been removed, comments may be sent to the above address.

Contents

Clock ... 7
Purpose and Function of Clock ... 7
A Gate Clock Generation ... 7
B Gate Clock Generation ... 9
T Clock Start-Stop ... 9

Registers ... 11
A Register ... 11
B Register ... 14
C Register ... 14
D Register ... 14
R Register (R Bus) ... 15
ALU Registers ... 18
ALU Control Register ... 18
ALU Function Register ... 18
P Register (P Bus) ... 18
Q Register (Q Bus) ... 18
Extension Register ... 19
Skew Select Register and Skew Buffer ... 19
Storage Protect Registers ... 19
Storage Protect Data Register ... 19
Storage Protect Key Register ... 19
Multiplex or Channel Registers ... 19
Multiplex Data Register ... 19
Multiplex Error Register ... 19
Selector Channel Registers ... 20
S Register ... 20
T Register ... 20
Channel Flags Register ... 20
Channel Checks and Status Register ... 20
Channel Key Register ... 20
W Registers (Byte Buffers) ... 20
TROS Address Registers ... 20
ROAR (Read Only Address Register) ... 20
ROSCAR (Read Only Storage Channel Address Register) 20
ROBAR (Read Only Back-up Address Register) ... 21

Staticizer's (Stats) ... 22

Arithmetic and Logical Unit (ALU) ... 30
ALU control ... 30
TROS Control ... 32
Function Control ... 36
Arithmetic ... 39
Binary Arithmetic ... 39
Decimal Arithmetic – True Add ... 39
Decimal Arithmetic – Complement Add ... 40
Treatment of Negative Decimal Numbers ... 40
ALU Operations ... 40
Logical Operations ... 40
Shifts ... 40
Arithmetic Operations ... 40
Carry Control ... 42
Carry Latches YCI and YCD ... 42
Tests and Conditions ... 43
B Condition Tests, CPU State and CD = 0 or 2 ... 43
C Condition Tests, CPU State ... 43
Additional Conditions ... 43
Summary of ALU Conditions ... 45
ALU Circuits ... 45
Two-Wire Logic ... 45
P and Q Input Latches ... 45
Skew Feature ... 46
Decimal Filler ... 46
Right Shift Unit ... 46
Connect Box ... 47
Carry Box ... 48
XOR Box ... 48
Decimal Correction ... 49
ALU Output Parity ... 49
ALU Extended Carry ... 49
Compress ALU Output ... 50
Checking ... 51
ALU Input Checks ... 51
ALU Internal Checks ... 51
ALU Output Checks ... 51
ALU Timing ... 51
Glossary for ALU ... 51

Transformer Read Only Storage (TROS) ... 52
Principles of Operation ... 52
Physical Description ... 53
The Transformer (Figure 34) ... 53
Tapes (Figure 35) ... 53
The Module ... 54
Core-Carrier Assembly ... 56
Laminar Bus ... 58
The Array ... 58
Functional Description ... 58
TROS Timing ... 58
TROS Addressing ... 58
TROS Drive Scheme ... 62
Decoding of ROSAB (See Figure 43) ... 62
TROS Control Word and Data Flow ... 64
Data Registers ... 64
Emit Circuits ... 66
TROS Control Word Parity Bits ... 66
Control Latches ... 66
Examples for Data Flow Control ... 66
Determining the Next TROS Address ... 66
FNB Format ... 66
CD = 0, 1, 3 Format ... 67
CD = 2 Format ... 67
Undump Format ... 67
Special Formats ... 67
Forced Addresses ... 67
Examples for Generating Next TROS Address ... 68
TROS Checking ... 69
ROBAR (Figure 48) ... 69
TROS Address Check (Figure 49) ... 69
TROS Word Data Check (Figure 50) ... 69
Control Field Decoder Checks ... 69
ROAR Format Checks (Figure 51) ... 69

Microprogram ... 74
Overall Microprogram Block Diagram ... 74
Individual Microroutines ... 74
Circuit Functions ... 76
System Operation ... 76
Error Handling ... 76
Manual Operations ... 77
CAS Logic Diagrams ... 77
Interpretation of a Single CAS Block ... 78
Determination of Microinstruction Sequence ... 79
Additional Information on CLD's ... 84
CLD Example of I-Fetch ... 84
Entry to I-Fetch ... 85
Exits of I-Fetch ... 85
First Halfword I-Fetch (Figure 54) ... 85

Sequence of I-Fetch Using CLD Example 85
Other Microinstructions on Figure 54 86
Machine Instructions – CLFC's 86
Instruction Fetch Microprogram 87
Second-Level Instruction Fetch: RX Half and Full Word Fixed Point 87
Second-Level Instruction Fetch: RX Floating Point 88
Second-Level Instruction Fetch: RS and SI Operations 88
Second-Level Instruction Fetch: SS Logical 88
Second-Level Instruction Fetch: SS Decimal 89
Branch and Link 89
Branch on Count 90
Branch on Condition 90
Convert to Binary 90
Convert to Decimal 91
Fixed-Point Halfword Add and Subtract 92
Fixed-Point Multiply 93
Floating-Point Multiply 94
Shift Instructions 95

Theory of Magnetic Core Storage 97
Ferrite Cores 97
Magnetic Properties 97
Hysteresis Loop 98
Controlling the Core 98
Coincident Current Addressing 100
Split Cycle Operation 100
Phase Reversal 102

Local Storage 103
External Machine Circuitry 103
Local Storage Address Register (LSAR) 103
Local Storage Address Loop (Figure 62) 103
Registers in the Address Loop 103
Incrementer 103
Address Loop Controls 103
Address Loop Checking 106
Timing 108
Local Storage Data Register 108
Local Storage Unit 108
Read Operation 112
Write Operation 112
Addressing 114
Drive Scheme 114
X Line Driving and Gating 114
Y Line Driving and Gating 115
Inhibit 115
Sense 115
Read and Write Controls 120
Timing 120
Read Cycle 122
Write Cycle 122
Circuit Description 122

Storage Protect 123
Functional Unit Circuits 123
Storage Protect Address Bus 123
SPLS Data Register 125
CPU/Mpx Channel Key Register 125
Selector Channel Key Registers 125
Parity Checking 125
Assignment of Storage 125
Examining the Storage Keys in SPLS 125
Storage Protect for CPU Operation 125
Storage Protect for Selector Channel 125
Storage Protect for Multiplex Channel 127
Protect Storage Address (PSA) Detection 127
Invalid Storage Address (ISA) Detection 131
Storage Address Tests (SAT and TRAP) 131
Storage Protect Local Storage 131
General Write Scheme 133
Reading Out a Location 133
Addressing 134
X-Y Drive 134
Sense 136
Inhibit 136
Read and Write Controls 136
Timing Logic 136
Storage Protect Block Diagram 139
Circuit Description 142

Main Storage 143
Capacity 143
Speed 143
Storage Cycle 145
64K Halfword Main Storage 145
Address Registers 145
Addressing 146
Drive Scheme 148
Common Sense/Inhibit Line 157
Control and Timing Logic 159
Block Diagram 164
Timing 164
External Controls 166
Power On Reset 166
Data Register (D Register) 167
Checking Storage Address Bus 167
Checking D Register 167
Circuit Description 167
1024K Halfword Multiplex Storage 168
Multiplex Storage Address 168
Circuit Description 169
Main Storage Power Supplies 169
Physical Structure of Main Storage 170
Arrays 172
Core Planes 172
X Drive Line Routing 176
Y Drive Line Routing 176
Sense/Inhibit Line Connections 176
Array End Boards D1 and D3 176
Jumper Boards D2 and D4 180
Socket Panel C1 180
Board A1 180
Board B1 180
Board A2 180

Checking 186
Error Detection 186
Logic Detected Errors 186
Microprogram Detected Errors 186
Start Log Out 186
Stop T Clock 186
Set Interface Control Check 186
System Operation After Error Detection 186
Normal Processing Mode, Errors Enabled 187
Normal Processing Mode, Errors Disabled 187
Checks in Relation to Timing (Figure 135) 187
Other Machine Malfunctions 190
Microprogram Looping 190
Machine Malfunctions Indicated as Program Check 190

Index 191

Illustrations

FIGURE	TITLE	PAGE
Clock		
1	Basic Clock Pulses	7
2	Clock System	8
3	Clock Start/Stop Circuit for A Gate	9
4	T Clock Start/Stop Controls	10
5	Start/Stop for B Gate Clocks	10
Registers		
6	Registers	12
7	A Register Circuits	13
8	D Register Set (Only D0 Shown)	15
9	R Bus Entry	17
10	Hardware Entry to Q Bus	18
11	Set and Reset of ROBAR	21
Staticizers		
12	Staticizer Parity Bit Generation	23
13	YA and YB Parity Change	24
14	Y Stats, Function and Control	25
15	YA Stats	26
16	YD Stats	27
17	YE Stats	28
18	Stats (Halt State, Wait, Enable, ASCII, I/O)	29
Arithmetic and Logical Unit (ALU)		
19	ALU Schematic	31
20	ALU Control Signals and Functions	32
21	Effects of ROS Fields on ALU	33
22	Function and Control Registers	37
23	Function Check Signals	38
24	Examples of Fixed-Point Binary Subtraction	39
25	ALU Work Sheet (2 Sheets)	41
26	ALU Carry Latches	44
27	ALU Timing Chart	46
28	Skew Select Schematic	46
29	Right Shift Box (Example Right Shift)	47
* 30	ALU Sum Generator	47
32	ALU Parity Generation	50
Transformer Read Only Storage (TROS)		
33	Principle of TROS	52
34	Tape with "U" and "I" Core	53
35	Tape Layout	53
36	Tape Stagger	54
37	TROS Module (Exploded View)	55
38	Core-carrier Assembly	56
39	General Arrangement of TROS Hardware	57
40	4K TROS Block Diagram	59
41	TROS Timing and Sense Current Waveforms	60
42	Principle of Driving and Gating	61
43	Decoding of ROAR	63
44	TROS Control Word and Basic Functions	64
45	IBM System/360 Model 40 Data Flow	65
46	Basic TROS Address Bit Generation	67
47	Determining Next TROS Address	68
48	ROBAR and ROBAR Timing (4K TROS)	70
49	TROS Parity Bit Generation and TROS Parity Check (4K TROS)	71
50	Data Check and Control Field Decoder Checks	72
51	DAQX to ROAR Transfer Check	73

FIGURE	TITLE	PAGE
Microprogram		
52	Microprogram Block Diagram	75
53	CAS Block Frame	79
54	CLD Sample	81
55	Essential Timings for Interpretation of CLD's	83
Theory of Magnetic Core Storage		
56	Two Stable States of Magnetic Core	97
57	1 Changing to 0	97
58	Core Field While Changing States	98
59	Normal and Distorted Hysteresis Loops	99
60	Coincident Current Addressing	101
61	Cores Involved in Phase Reversal	102
Local Storage		
62	Local Storage Address Loop	104
63	CH Field Control Chart	105
64	Set and Reset LSAR	107
65	LSAR Bits	108
66	Incrementer and Incrementer Control	109
67	Parity Check/Change	110
68	LSAR Destination, H and J Register	111
69	Local Storage Address Loop Timing	112
70	Local Storage Block Diagram	113
71	Address Decode Diagram	114
72	Layout of One Core Plane	115
73	X Drive Diagram	116
74	Y Drive Diagram	117
75	Inhibit Drive Diagram	118
76	Sense Diagram	119
77	Local Storage Read and Write Controls	120
78	Local Storage Timing Circuit	121
79	Local Storage Timing Diagram	121
Storage Protect		
80	Storage Protection Symbolically	124
81	SPLS Addressing Scheme	125
82	Storage Protect Data Flow	126
83	Storage Protect Address Bus	127
84	Storage Protect Local Storage Data Register, CPU/Mpx Channel Key Register	128
85	Parity Inhibit, Parity Generation, Parity Check	129
86	PSA and ISA Detection	130
87	SAT and TRAP	132
88	Storage Protect Core Plane Bit Sections	133
89	Storage Protect Address Decode	134
90	X Line Driving and Gating	135
91	Y Line Driving and Gating	137
92	Sense Schematic	138
93	Inhibit Drive Schematic	139
94	Storage Protect Timing Circuit	140
95	Storage Protect	141
Main Storage		
96	Core Storage Locations	144
97	Phase Reversal Control	146
98	Main Storage Arrays	147
99	Plane Layout	148
100	Routing of X and Y Wires	149
101	Routing of X Lines	150
102	Phase Reversal	151
103	Storage Address Bus Bit Allocation	152

*Figure 31. Carry Box and Look-ahead Equations (deleted)

FIGURE	TITLE	PAGE
104	Connection of Four Basic Drive Circuits	152
105	Main Storage Unit Drive	153
106	Storage X Dimension Drive	155
107	Main Storage Y Dimension Drive	156
108	Core Wires (Three Wire System)	157
109	Main Storage Sense Amplifier and Inhibit Driver Allocation	158
110	Inhibit Drive	160
111	Principle of Inhibit Drive	161
112	Main Storage Sense	162
113	Storage Select and Gate Select Pulses	163
114	Main Storage Timing Chart	165
115	Main Storage Addressing Parity	167
116	D Register Parity Generation and Parity Check	168
117	Storage Address Bit Allocation for Mpx Storage	169
118	Multiplex Y Drive	170
119	Multiplex Storage Y Dimension	171
120	64K Halfword Storage Unit	172
121	Array No. 1	173
122	Plane Arrangement (Array 1, Invert for Array 2)	174
123	Sense/Inhibit Wire Routing	175
124	Core Winding Three Wire	176
125	X Line Routing	177
126	Y Line Routing	178
127	Array End Board D1	179
128	Interface Area of Board D1	179
129	Jumper Board D2	181
130	Socket Panel C1	182
131	Board A1	183
132	Board B1 Layout	184
133	Board A2 Layout	185

Checking

FIGURE	TITLE	PAGE
134	System Operation After Error Detection	188
135	Checks in Relation to Timing	189

Purpose and Function of Clock

- **Provides timed pulses to the system**
- **Three types of clock pulses are generated: P, T, and gated T**
- **Separate clock systems for A and B logic gates**

The purpose of the clock is to provide timing pulses. Two separate timing systems are generated, one for each logic gate. The A gate system synchronizes the B gate system. The A gate system has two sets of pulses: the continuously running P clock and the T clock. The P clock pulses control main storage and TROS. T clock pulses can be started and stopped by CPU controls.

The B gate system has a T clock, which runs in synchronism with the A gate T clock, and a modified clock called gated T clock, which is used in selector channel operations.

A Gate Clock Generation

- **Free running oscillator 3.2 mc**
- **Paraphase amplifier produces clock odd and clock even**
- **Gate generator**
- **P and T pulse generator**

For basic clock pulse timing see Figure 1. A diagram of the clock system is shown in Figure 2. Circuits are on ALD pages KC001 through KC081.

The basic clock time is derived from an oscillator and delay card which produces control pulses at a frequency of 3.2mc (megacycles, 10^6 cycles/second) with a tolerance of + or −1 kilocycle (10^3 cycles/second). The delay on the card produces a pulse called delayed clock, which occurs 80 nanoseconds after clock.

The clock pulse is fed into a paraphase amplifier and emitter follower stage which produces the pulses clock odd and clock even. Clock odd and clock even are 180 degrees out of phase.

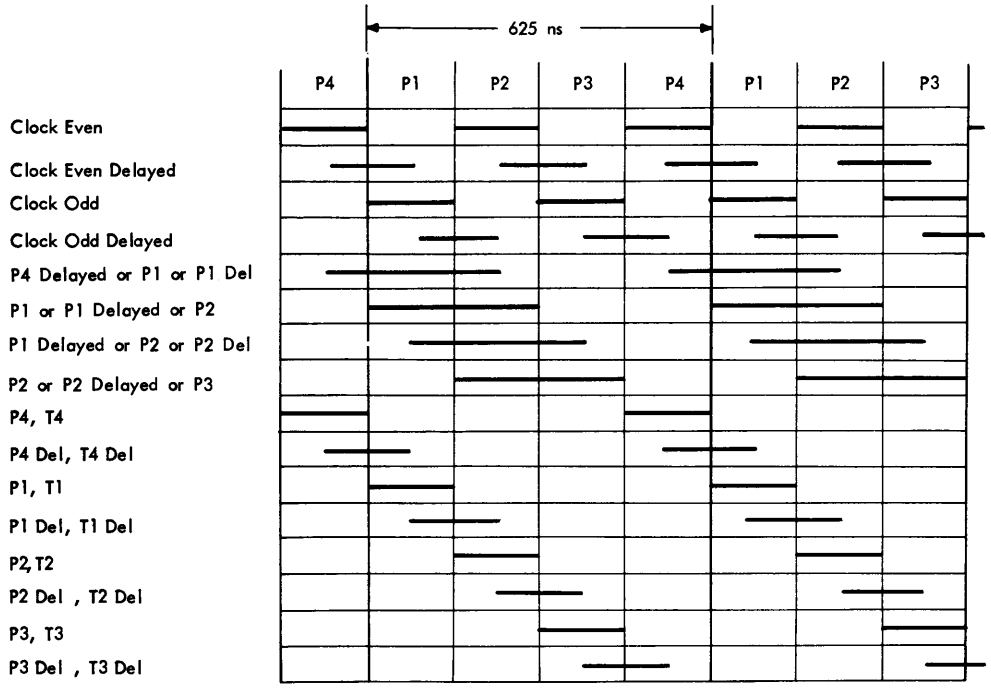


Figure 1. Basic Clock Pulses

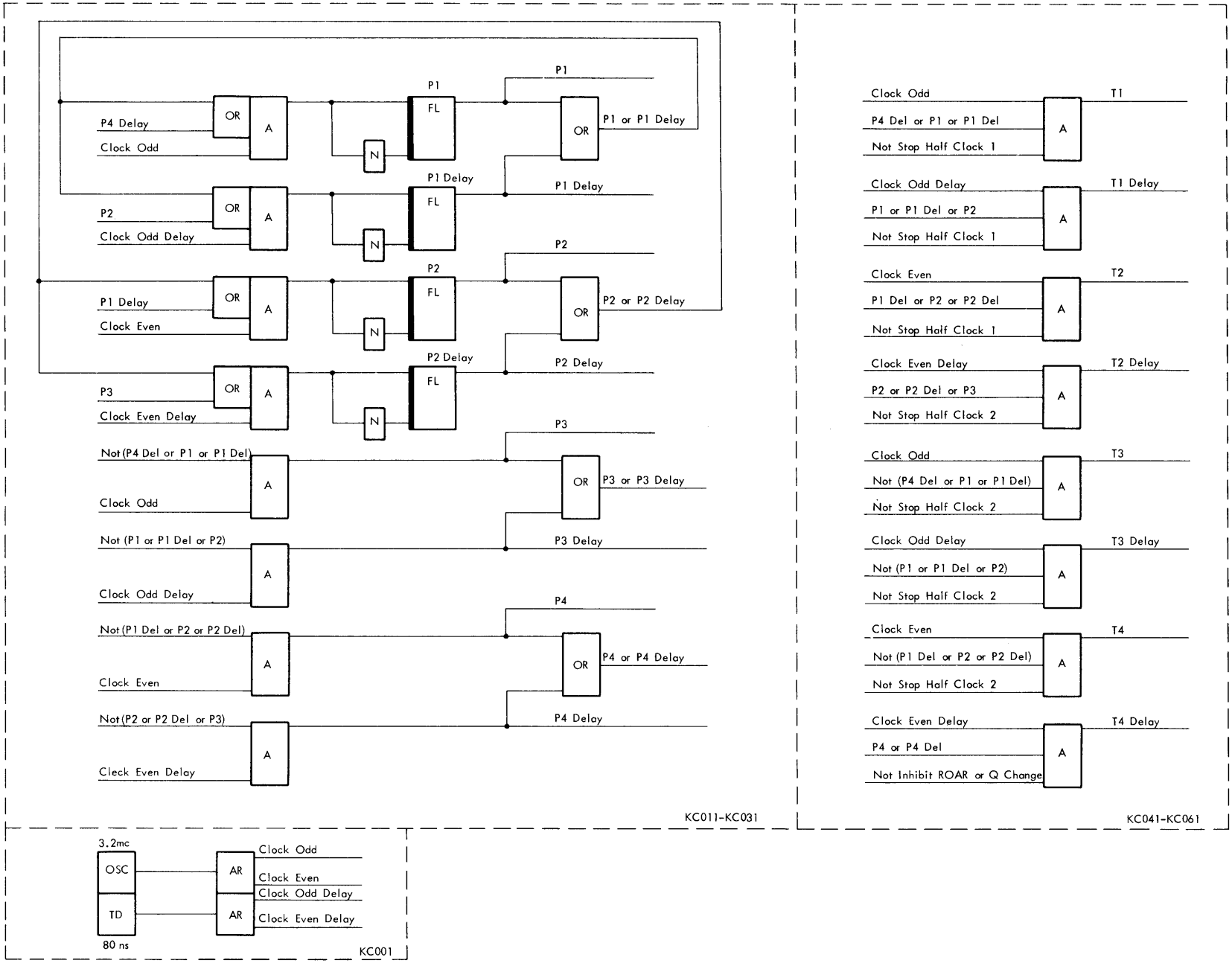


Figure 2. Clock System

The delayed clock pulse is fed into a second identical stage, which produces delayed clock odd and delayed clock even.

The gate generator gates the clock signals in the P pulse and T pulse generators to produce the individual timing pulses (P1, P2, T1, T2, etc.) used throughout the machine.

Gate generator circuits are built with AND and OR circuits gated by the P pulse generator output. All circuits are designed so that the proper pulse sequence is established automatically after power is brought up.

The P pulse and T pulse generators are also combinations of AND and OR circuits, fed by the clock signals and the gates provided by the gate generator. The stop clock, when active, blocks the T clock and controls the T pulse generator.

B Gate Clock Generation

- **Controlled by A gate oscillator**
- **Paraphase amplifier produces clock odd and clock even**
- **Gate generator uses latches and is synchronized by the A gate P clock**
- **T pulse generator and gated T pulse generator**

Basic clock pulses and a block diagram are shown in Figures 1 and 2. Circuits are on ALD pages GX501.

The basic timing is derived from the oscillator of the A gate system. As a result of the time delay in the wiring from A to B gate, the B gate timing system is approximately 10 nanoseconds later than the A gate.

A set of paraphase amplifiers identical to the A gate system, produces clock odd and clock even and the corresponding delayed clock signals.

The gate generator consists of two pairs of interconnected latches supplied with the paraphase amplifier outputs. Each of the four latches is set and reset alternately at periods of approximately 310 nanoseconds, equal to two consecutive P pulses. The set times of the latches are spaced at half P pulse intervals and are staggered by half a P pulse, producing identical gates as used in the A gate.

A synchronizing signal, synch B gate clock, is supplied from the A gate to ensure that A gate P1 is coincident with B gate P1.

The gate generator waveforms in conjunction with the clock signals are combined in logic circuits to provide the B gate T clock and gated T clock.

The B gate clocks are stopped and started in parallel with the A gate T clock. The gated T clock is:
1. Stopped during logout microprogram to prevent selector channel interference.
2. Running during hardware system reset.

T Clock Start-Stop

- **Clocks always stop after T4 del (delayed)**
- **The clock for A gate is controlled by stop half clock lines**
- **B gate clocks controlled by A gate T clock**

Figure 3 shows the simplified circuits for developing the two signals, stop half clock 1 and stop half clock

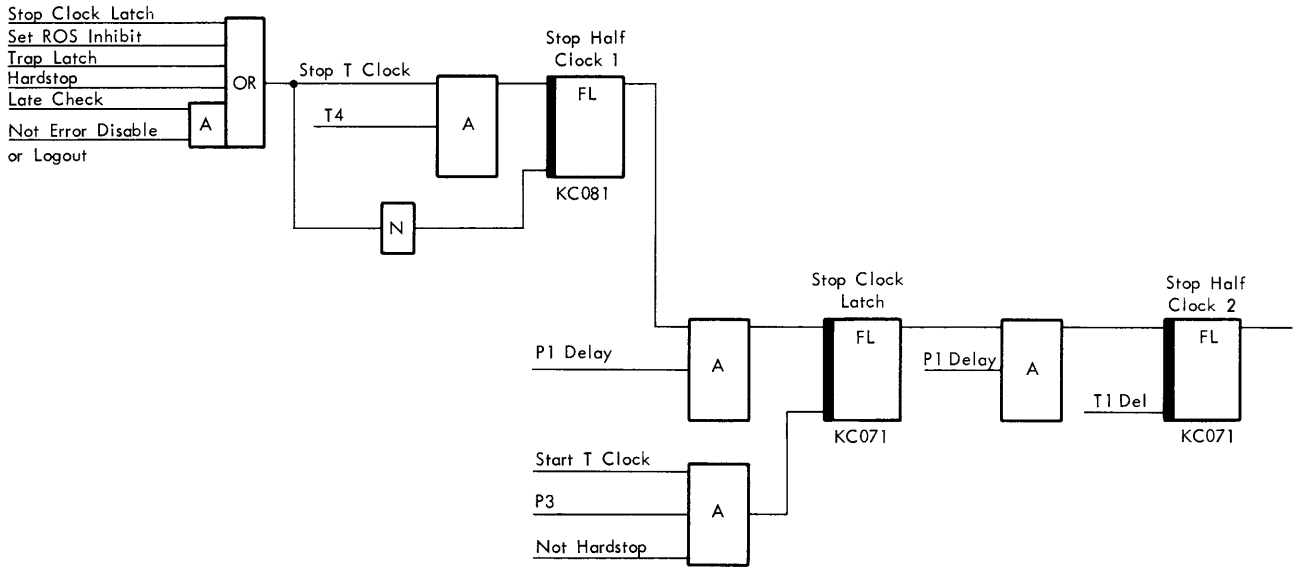


Figure 3. Clock Start/Stop Circuit for A Gate

2, which are responsible for blocking the A gate T clock generator.

Stop half clock 1 blocks T1 and T2, stop half clock 2 blocks T2 del to T4. The line labelled stop T clock becomes active for all possible conditions that stop the clock and can come on at any time during the cycle. In practice, only errors detected before T4 del can generate this signal and can stop the clock at the end of the cycle in which they are detected. Example: Local storage read check cannot stop in the same cycle. Similarly the line start T clock is the OR condition of all signals that resume T clock operation.

In Figure 4 the last pulse generated after stop T clock is present is T4 del; the first pulse blocked is T1.

To restart the T clock, start T clock resets the stop clock latch and the first timing pulse generated is T1.

The B gate clocks are directly controlled by the A gate clock. The signal, B gate clock to run, is essentially T1 and T2 of the A gate clock. (Figure 5).

For the B gate T clock, this signal is directly latched to provide gating of the T clock generator. Not logout is needed for gated T clock pulses to be active. Channel system reset, in addition, allows the gated T clock to run.

The gated T pulses and T pulses will be present together, except in the following circumstances:

1. No gated T pulses during microprogram logout.
2. Gated T pulses during hardware system reset.

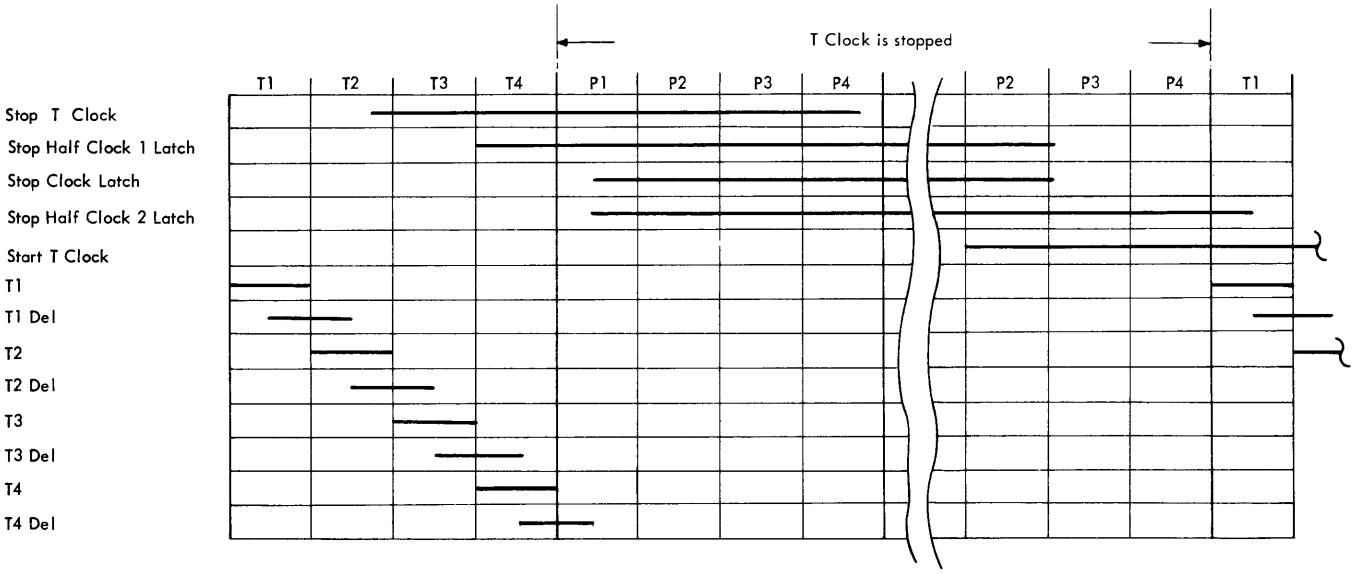


Figure 4. T Clock Start/Stop Controls

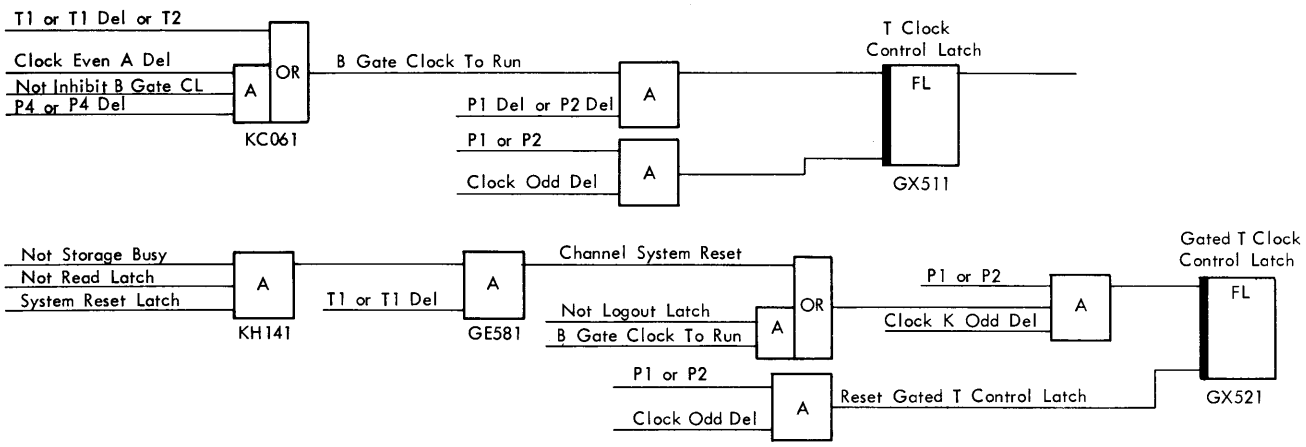


Figure 5. Start/Stop for B Gate Clocks

- **Latch-type circuits which temporarily hold several bits of data or control information are called registers**
- **Relationship of the various registers is shown on the over-all data flow**
- **Registers can store information for any number of machine cycles until new information is read in**
- **A register that is reset in every cycle is also referred to as a data bus**

A group of latches which is used temporarily to store data or control information is called a register. If a certain latch of the group is on, the position is considered to contain a "one" bit (1). When the position is reset or off, it contains a "zero" bit (0).

Read-in and read-out of registers is normally under control of the microprogram, but some of the registers have special logic controls incorporated. This section gives the characteristics of every register in the machine.

Registers will normally hold their contents until new information is read into them, which may be after one or any number of machine cycles. Some specific registers, however, are reset at the beginning of every cycle. The purpose of these registers is to buffer information during data transfers (either on the 16-bit or 8-bit data flow) where the source register may be changed during the present cycle, but the old contents may be used for the entire duration of the cycle.

This type of register is often referred to as a data bus. (Examples: R bus, P bus, Q bus.) Figure 6 shows the basic controls and purposes of the registers.

A Register

- **Two bytes (A0, A1) plus extension (AX) each with parity bit**
- **Temporary work register**
- **Main storage address register for CPU operations**
- **Set from R bus and ALU output**
- **Output to main storage address bus (SAB), R bus, and P bus**
- **Register is not parity checked, but during CPU operations is always gated to SAB which is checked**

Figure 7 shows the A register circuits.

The capacity of the A register is two bytes plus extension, each part having its own parity bit. The A register may be used as a temporary work register, but its main purpose is to address main storage during CPU or multiplexor channel operations.

The A register is read in under microprogram control from either the R bus or the ALU output. Logic circuit controls are included for manual console operations. When the register is read in, the old content of the register is reset (all bit latches turned off).

During system reset all latches are turned off, but a special entry to the parity latches becomes active which insures that system reset forces "good zeros" (all latches off, parity on) into the register.

The output of the A register can be gated to one or more of the following units: storage address bus (SAB), R bus, or P bus. When the system is not working as a selector channel the output is gated to the SAB.

In accordance with main storage addressing, bit 7 of A1 is not gated to SAB. Connection of A register to SAB also depends on the multiplexor (Mpx) stat, which provides a changed data path for addressing multiplex store.

The A register itself is not parity checked, but parity checking exists on the SAB which is almost continuously fed by the A register. Additionally, the other destinations of A register information are parity checked (R bus, P bus).

The source of A register data can be specified from the R bus and the ALU output in the same microinstruction. In this case, the ALU output should override R bus data.

R bus gating is blocked for the register part which is intended to receive the ALU output:

1. Manually loading the A register from the console data switches (which are fed to the R bus) is controlled by the AND switch, display and store A, AND'ed with manual set from R.
2. For display and store of main storage, the console address switches are fed to the R bus and gated into A by the switch, address switches to R, AND'ed with manual store-display.

Register	Set	Reset	Length	Main function	Special controls	ALD's
A Reg	R bus, ALU	On read in	19 bits	Main storage addressing		RA001
B Reg	R bus, ALU	On read in	16 bits	Temporary data storage		RB001
C Reg	R bus, ALU	On read in	19 bits	Temporary storage only; source to Mpx data reg		RC001
D Reg	R bus, ALU, MS data out	On read in	16 bits	Temporary storage, MS data register	Coincidence circuitry controls priority of incoming data: ALU, R bus output, MS read data	RD021
R Reg (Bus)	From source specified in CJ field	Every machine cycle	19 bits	Buffer register for two-byte data flow. Data register for local storage		RR001
ALU Control	From ALU function register or CE or CG fields	Every machine cycle	4 bits	Determines the ALU function to be performed		KP023
ALU Function	From CE field if CN=15	On read in	5 bits	Stores ALU function for indirect operations		KP001
P Reg (Bus)	From source specified in CP field	Every machine cycle	8 bits	ALU entry for one operand		RP001
Q Reg (Bus)	From source specified in CQ field	Every machine cycle	8 bits	ALU entry for one operand		RQ001
Extension	By microprogram	Every machine cycle	3 bits	Contains the entry to ALU if a register with an extension is gated to P or Q bus	If both P and Q specify an extension, P blocks Q	RE001
Skew Select	From CK=1	Every machine cycle	8 bits	Provides a four-bit left shift of the Q bus data	Controlled by ALU function register	AQ011
Skew Buffer	From CK=1	On read in or microprogram (CR = 6)	4 bits	Holds the skewed data	Controlled by ALU function register	AQ011
Storage Protect	Logic or microprogram	Logic or microprogram	4 bits	Holds the storage key when MS is accessed		KU011
Mpx Data	Data (C reg)	On read in	8 bits	Holds data to bus out		FA023
Mpx Error	Channel conditions	I/O mode, CB=7	6 latches	Set by error conditions in channel, can be displayed		FN011
S Reg	R bus, ALU	On read in	19 bits	Addresses MS during channel operations		HS501
T Reg	ALU output	On read in	16 bits	Records the number of bytes transmitted by the channel		HT501
Chan Flags	ALU, ROS, logic circuits	I/O mode, clear channel	8 bits	Holds channel control flags		GF501
Chan Checks and Stat Regs	From channel logic circuits	Clear SC errors	8 bits	Holds channel checks and status bits		GG543
Chan Key	Microprogram	On read in	4 bits	Holds the protect key during selector channel operations		GU501
W Regs	Interface, ALU, R bus	Every machine cycle	5 bytes	Data buffer between MS and interface		HW501
ROAR	Logic or microprogram	Every machine cycle	12 bits	Used to address read only storage		RX001
ROSCAR	Logic or microprogram	Microprogram	13 bits	Used to address read only storage during selector channel operations		GC501
ROBAR	ROSAB	Every cycle unless inhibit ROBAR is active	13 bits	Holds the ROS address used in the previous cycle		RX201

Figure 6. Registers

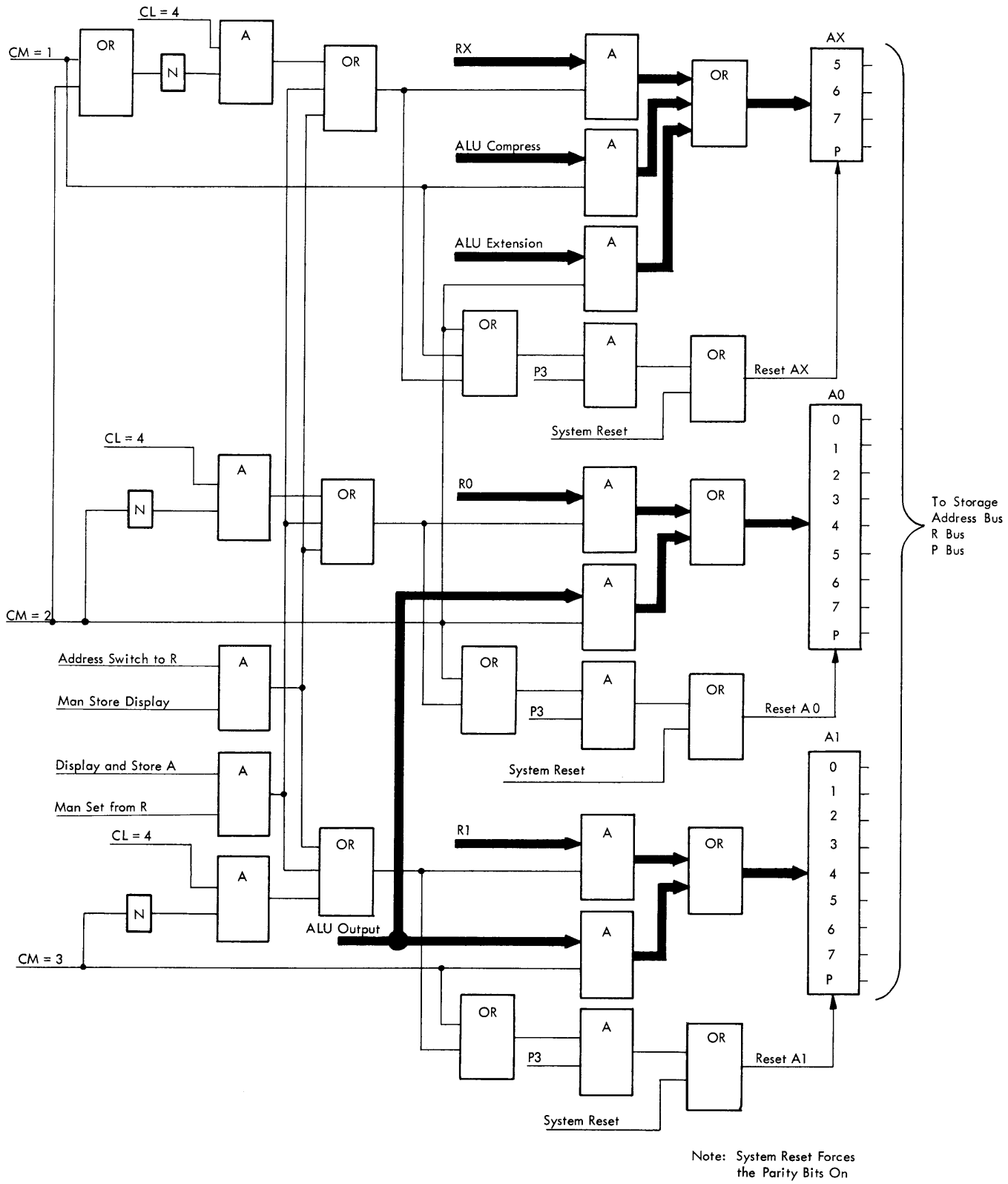


Figure 7. A Register Circuits

B Register

- **Two bytes (B0 and B1) each with a parity bit**
- **B register is microprogram controlled to serve as a work area and temporary storage**
- **Set from R bus and ALU output**
- **Output to R bus, P bus, Q bus**
- **Register is not parity checked**

The B register is used as temporary storage for two bytes of data. It is under control of the microprogram when handling data between registers and the ALU, etc.

B register circuits are similar to the circuits shown for the A register (Figure 7).

TROS field values control the gating and the coincidence circuits.

The B register is not parity checked; however, its contents are eventually checked when the data are moved to the R, P, or Q registers.

C Register

- **Two bytes (C0 and C1) plus an extension (CX) each with a parity bit**
- **Temporary work register, can hold an updated A register address**
- **Set from R bus and ALU output**
- **Output to R bus and Q bus**
- **Byte C1 is the source of all output data of the multiplex channel**
- **Not parity checked**

The C register, used as a temporary storage, contains two bytes (C0 and C1) and an extension (CX). All parts have separate parity bits, although the register is not parity checked. Invalid parity is eventually detected when the C register data are moved to either the R or Q registers.

C register circuits are similar to those shown for the A register. The extension CX is provided for more efficient access to main storage. During a main storage cycle, the A register may not be changed. This would mean that address updating could be done only after the MS cycle. With CX available, updating is done during the MS cycle, and the modified address is stored in C.

Byte C1 is the only source which is fed to the multiplex data register. All output information of the multiplexor channel, therefore, has to be transferred to C1.

D Register

- **Two bytes (D0, D1) each with parity bit**
- **Temporary work register**
- **MS data register**
- **Set from R bus, ALU output or MS output (sense)**
- **Output to R bus, Q bus, or MS entry (inhibit)**
- **Parity checked**
- **For output to MS, good parity is always generated**
- **D register is directly displayed on the console**
- **D register assigns priority to system data flow**

The D register can be used as temporary storage for two bytes of data, but its principal function is to act as the main storage data register.

During the second machine cycle of a main storage read operation, the MS sense amplifier output is gated by logic circuits to the D register. Entry of R bus and ALU is under microprogram control.

Circuitry is provided which assigns highest priority to the ALU output, second highest priority to R bus output and lowest priority to main storage read data.

Main storage read data are completely lost if the microinstruction following a MS read call specifies R bus to D register.

No provisions are made to override one byte of main storage data by an ALU output. Specification of ALU output to D in the second read cycle of main storage is an invalid microprogram statement.

In case a MS read call is given for an invalid storage address, or if the storage protection is violated, R and ALU entries to the D register are blocked to ensure that storage data are not destroyed. (Figure 8 shows the entry gates for D0).

The D register output to the main storage inhibit drivers is always in good parity, i.e. the parity bits to main storage are generated according to the contents of bits 0-7. The D register parity bits are not used in this circuit.

The D register is parity checked, incorrect parity sets the early check latch.

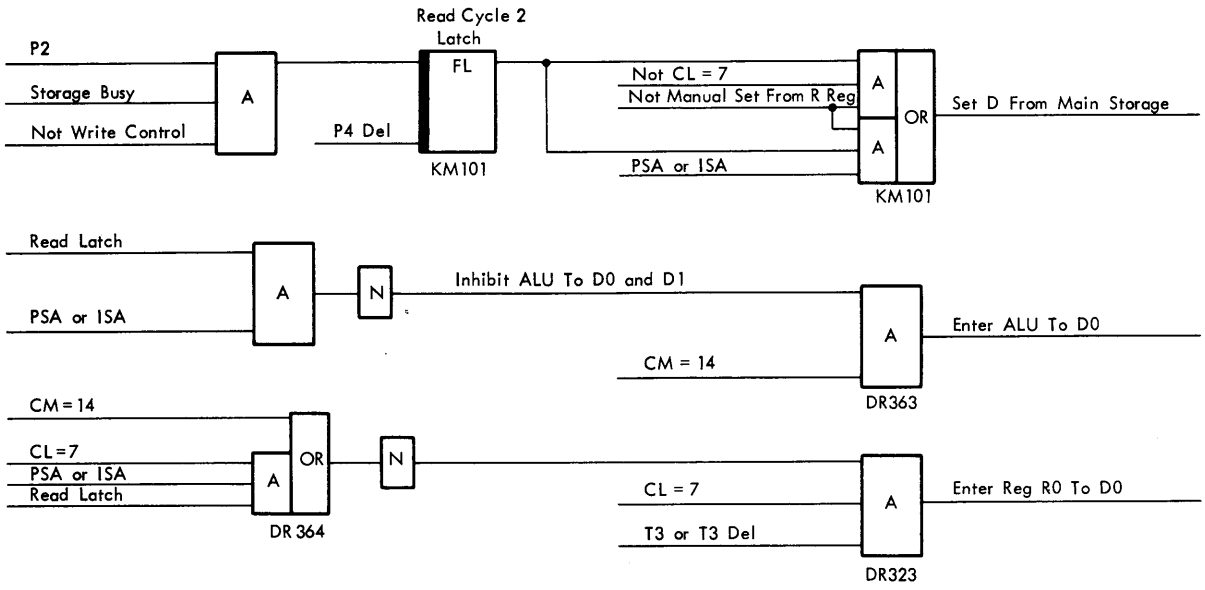


Figure 8. D Register Set (Only D0 Shown)

R Register (R Bus)

- **Two bytes (R0, R1) plus extension (RX) each with parity bit**
- **Buffer register for the two-byte data flow (R bus); special purpose: data register for local storage and buffer register for display**
- **Set by microprogram from the sources specified in the CJ field. Set by logic circuits during logout, manual store-display and dump from various other sources**
- **Output to destinations specified in the CL field and by logic circuits**
- **Parity checked**

The capacity of the R register is two bytes plus extension, each part having its own parity bit. The R register is reset in every machine cycle and is therefore also referred to as R bus.

The main purpose of this register is to provide a buffer for data transfer between various parts of the system.

Special purposes of the R register are:

1. Data register for local storage
2. Display buffer for the console lights associated with the roller switches.

During microprogram cycles the R bus is fed from the various sources specified in the CJ field; during hardware cycles or manual store-display operations various other sources are gated to the R bus.

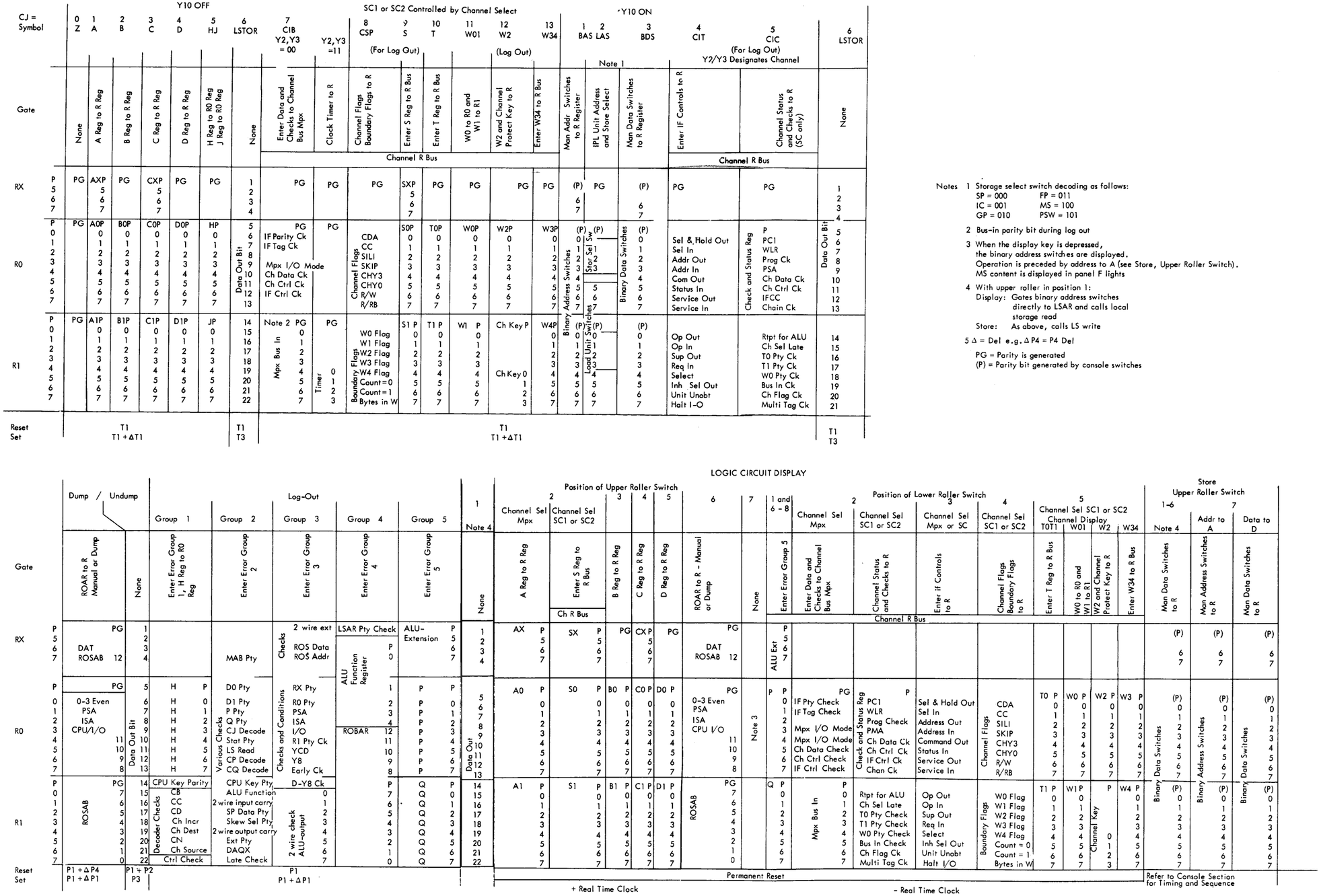
Example: During logout or display the P and Q bus can be gated directly to R. R bus entries for machine check conditions are also provided.

Figure 9 shows the various possibilities for R bus entries and the corresponding gates responsible. The main timings for reset and set are also indicated. The figure may be used as a starting point before entering the ALD's.

The output of the R bus is again distributed throughout the machine during microprogram cycles to the destinations specified in the CL field.

In the undump hardware cycle R is gated to ROAR. During hardware logout R is stored in local storage.

In manual store-display operations, R is gated by logic circuits to various parts of the system. Hardstop display with the upper roller switch in positions other than 1 (R bus display) gates the R bus to two sets of



Notes 1 Storage select switch decoding as follows:
SP = 000 FP = 011
IC = 001 MS = 100
GP = 010 PSW = 101

2 Bus-in parity bit during log out

3 When the display key is depressed, the binary address switches are displayed. Operation is preceded by address to A (see Store, Upper Roller Switch). MS content is displayed in panel F lights

4 With upper roller in position 1:
Display: Gates binary address switches directly to LSAR and calls local storage read
Store: As above, calls LS write

5 Δ = Del e.g. ΔP4 = P4 Del

PG = Parity is generated
(P) = Parity bit generated by console switches

Figure 9. R Bus Entry

display lights which are alternately gated by the line frequency (with two different sources gated into R, again under control of the line frequency).

The R register is parity checked and normally sampled at P3 time; an R register parity check forces early check. After a local storage read operation, the R register parity is sampled at P4 and forces the early check only in the next cycle.

The R register parity bits are normally set by the parity latches of the source register or, if no parity latches are available, correct parity may be generated. During logout or display, correct parity (Figure 9) is normally not maintained.

ALU Registers

ALU Control Register

- **Four bits plus parity**
- **Determines the ALU function to be performed**
- **Set from CE field, CG field or ALU function register**
- **Permanently displayed**

ALU Function Register

- **Five bits plus parity**
- **Stores ALU function for indirect operations**
- **Set by CE field if CN = 15**
- **Output used to set ALU control register during logout, gated to R bus**

P Register (P Bus)

- **One byte plus parity bit**
- **ALU entry for one operand**
- **Set by microprogram from the sources specified in the CP field**
- **Parity checked**

The capacity of the P register is one byte plus parity bit. The P register is reset in every machine cycle and is also referred to as the P bus.

The purpose of the P bus is to provide a buffer for one of the two ALU input bytes. With this design it is possible to alter the original data source in the same cycle in which it is used as the ALU input.

The P register is set only by microprogram from one of the sources specified in the CP field.

The register is checked for correct parity; detection of incorrect parity forces an early check.

During logout and manual display the P register is directly gated to the R bus (R0).

Q Register (Q Bus)

- **One byte plus parity bit**
- **Main purpose is ALU entry for one operand; buffer register for manual display**
- **Set by microprogram from the sources specified in the CQ field; set by circuitry during manual display**
- **Output normally to ALU during logout and manual display to R bus**
- **Parity checked**

Capacity of the Q register is one byte plus parity bit. The register is reset in every machine cycle and therefore is also referred to as the Q bus.

The main purpose of the Q bus is to provide a buffer for the second of the two ALU input bytes, as specified in the CQ field.

In addition, the Q register can be gated to the R bus (R1) during microprogram logout and manual display.

Manual display is controlled by the lower roller switch where, for positions 6, 7, and 8, the appropriate information is gated by logic to Q. The table in Figure 10 shows the logic entries of Q; for microprogram entries refer to CQ specifications.

The Q register is parity checked. Detection of incorrect parity forces an early check.

		Lower Roller in Position			
		6	7		8
Gate		External Interrupts to Q Bus	Store and CPU Tags to Q Bus		Direct Control to Q Bus
Q	P	External Interrupts	Note	PG	Direct Data Bits
	0	Timer	SPLS Data	0	0
	1	Console		1	1
	2	2		2	2
	3	3		3	3
	4	4	CPU or Mpx Channel Key	0	4
	5	5		1	5
	6	6		2	6
	7	7		3	7
Reset Set		MSS 1 Pulse See Console Section			

Note: Q Parity Bit is generated as follows:

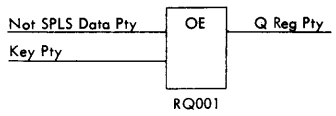


Figure 10. Hardware Entry to Q Bus

Extension Register

- Three bits plus parity bit
- ALU entry for one register extension if one or both ALU entries specify the 0 byte of a source that contains an extension
- Set by microprogram only
- Parity checked

Capacity of the extension register is three bits plus a parity bit. Like the P and Q registers, the extension is reset in every machine cycle.

The extension is used as an entry buffer for one of the extension positions if the 0 byte of a register with extension is gated to P or Q (e.g. A0, C0, S0).

The set of the extension register is controlled only by the appropriate CP or CQ field values.

Whenever both fields specify a source register with an extension (0 byte), the entry from the P bus source blocks the Q source extension, i.e. AX or SX override CX.

When A0, S0 or C0 are not specified for either ALU entry, the extension is reset to zeros (bits 5, 6, 7 off; parity on).

The extension register is gated to the ALU extended carry circuits. During logout or manual display it can be gated to the R bus.

The register is parity checked; incorrect parity forces an early check.

Skew Select Register and Skew Buffer

- Skew select register, eight bits plus parity bit
- Skew buffer four bits plus parity bit
- Skew select register is actual ALU entry from the Q bus. Transfer from Q bus to skew select provides a four-bit left shift when the microprogram specifies SKEW
- Both registers are controlled by ALU function control
- The skew select register is permanently displayed on the console and is parity checked

Storage Protect Registers

Storage Protect Data Register

- Four bits plus parity bit
- Data register of storage protect local storage (SPLS), i.e. it holds the storage key whenever main storage is accessed
- Set by logic circuits if main storage is accessed; under microprogram control if set from ALU output bits 0-3
- Output fed to compare circuits by logic circuits and to Q bus bits 0-3 by logic circuits and microprogram
- Displayed via Q bus
- Parity checked

Storage Protect Key Register

- Four bits plus parity bit
- Holds the protect key for CPU and multiplexor channel operations
- Set by microprogram from ALU output bits 4-7
- Output fed to compare bus by logic circuits and to Q bus bits 4-7 by logic circuits and microprogram
- Displayed via Q bus
- Parity checked only when fed to compare bus

Multiplexor Channel Registers

Multiplex Data Register

- One byte plus parity
- Buffer register feeding bus out of the interface
- Set by register C1 when an out tag is programmed (CB field)

Multiplex Error Register

- Six individual latches with common reset; no parity bit
- Set by various conditions detected in the channel
- Output gated to R0 with correct parity generated
- Can be displayed in hardstop

These latches are set by various error conditions detected during multiplexor channel operation.

With the micro-order CJ = 7 or in hardstop display the latch outputs are gated to R0. In both cases the R0 parity bit is set according to the number of latches that are on.

Feed through is as follows:

R0	Parity	Generated
R0	Bit 0	Interface parity check latch
R0	Bit 1	Interface tag check latch
R0	Bit 2	0
R0	Bit 3	Multiplex I/O mode latch
R0	Bit 4	Channel data check latch
R0	Bit 5	Channel control check latch
R0	Bit 6	Interface control check latch
R0	Bit 7	0

Selector Channel Registers

In the following listing, the registers of one selector channel are included. With two selector channels installed, all registers are duplicated. Identification in the ALD's is by SC1 or SC2 preceding or following the line labels.

S Register

- Two bytes plus extension (SX) each with (S0, S1) parity bit
- Main purpose is addressing of main storage during selector channel operations; can also be used as temporary work register
- Set from R bus and ALU output
- Output to main storage address bus (SAB), R bus and P bus

The S register is similar to the A register, and is used during selector channel microprograms.

T Register

- Two bytes (T0, T1); each with parity bit
- Main purpose is to record the number of bytes transmitted by the channel
- Set only ALU output
- Output to P bus; for logout and display also to R bus
- Parity checked

Channel Flags Register

- Eight bits plus parity bit
- Used to hold channel control flags
- Individual bits are set independently, bits 0-3 from ALU output, bits 4 and 5 from CE field, bits 6 and 7 by channel circuits
- Output to R bus only for logout and display (R0)
- Only bits 0-3 are parity checked

Channel Checks and Status Register

- Eight individual latches
- Used to hold channel checks and status
- Set by channel logic circuits
- Output gated to P bus for logout and display R0

Channel Key Register

- Four bits plus parity bit
- Holds the protect key for selector channel operations
- Set by microprogram from ALU output bits 4-7
- Output fed to compare bus by logic circuits and to R1 for logout and display
- Parity checked when fed to compare bus

W Register (Byte Buffers)

- Five registers (W0, W4), each containing one byte plus parity
- Data buffer between main storage and interface
- Information is entered in W4 (or in W3 and W4 in parallel) from the interface, the ALU output, or the R bus
- Output is from W0 (or W0 and W1 in parallel), to the interface, the P bus or the R bus
- Transfer of information from W4 to W0 is controlled by logic circuits
- All bytes can be gated to the R bus for logout or display
- W0 is parity checked

TROS Address Registers

ROAR (Read Only Address Register)

- Standard capacity is 12 bits (13 bits with TROS of more than 4K); parity bit is generated
- Used to address TROS during CPU or multiplexor channel operations
- Set by microprogram or logic circuits
- Output gated to ROSAB (read only storage address bus)
- Check of ROAR setting by ROAR transfer check (DAQX check)
- DAQX check forces late check

ROSCAR (Read Only Storage Channel Address Register)

- Capacity is 13 bits plus parity
- Used to address TROS during selector channel operations
- Each selector channel has its own ROSCAR (ROSCAR 1 or ROSCAR 2)
- Set by microprogram or logic circuits
- Output gated to ROSAB

ROBAR (Read Only Back-up Address Register)

- **Only purpose: diagnostic tool for customer engineer**
- **Capacity is 13 bits plus parity bit**
- **Records the TROS address which was used in the previous machine cycle**
- **Set from ROSAB**
- **Directly displayed on console and gated to R bus for logout**

ROBAR has a capacity of 13 bits plus parity bit and is set in every machine cycle from ROSAB.

ROBAR is used to record the TROS address used in the cycle in which the machine is stopped.

ROAR or ROSCAR is set during a machine cycle to the TROS address required for the next cycle. Consequently ROAR or ROSCAR points to the cycle to be executed next if the machine is stopped after T4 (checks, or in single cycle mode). If this next cycle were a microinstruction that can be entered from various sources, there would be no means (without ROBAR) of finding out which cycle produced the condition that led to the stop.

ROBAR is set at P4 delayed if no errors were detected during that cycle. (Figure 11).

ROBAR is permanently displayed on the console and gated to the R bus for logout.

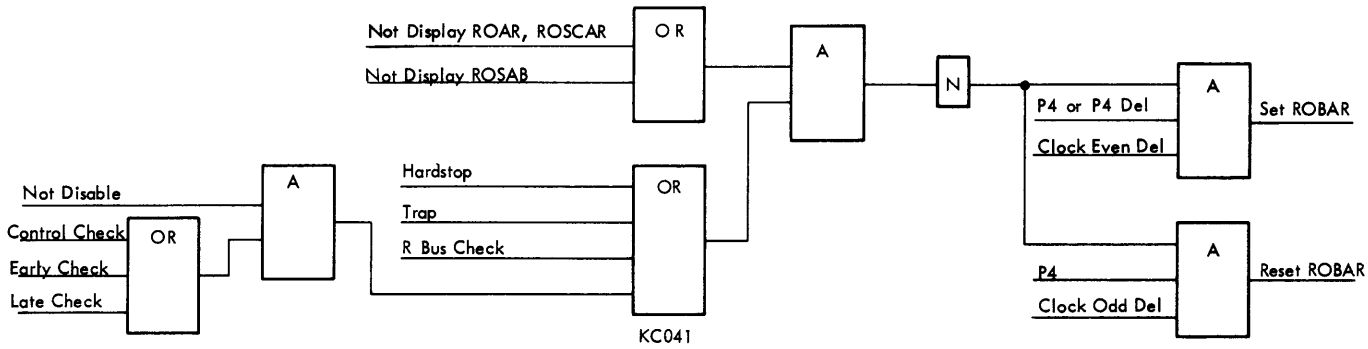


Figure 11. Set and Reset of ROBAR

Staticizers (Stats)

- **Stats are used to indicate machine conditions or status**
- **Stats are set by microprogram or circuitry**
- **Stats can be tested by the microprogram**
- **Y stats are in four groups (YA, YB, YD, and YE) with four stats in each group**

Stats are used to indicate machine conditions. Stats can be set by microprogram, or by logic circuitry. They are displayed on the console and can be tested at microprogram level. As a result of the microprogram tests, data flow paths and microinstruction sequence can be altered.

Of the four groups (referred to as Y stats) only the YA and YB stats are parity checked. Overall odd parity is maintained for Y0-Y7 (YA and YB). Figure 12 shows the parity bit generation; Figure 13 shows parity bit change. The purpose and control for the individual Y stats are shown in Figure 14.

Logic for the YA stats circuit is shown in Figure 15. The YB circuits are similar and are not shown. YD stats circuits are shown in Figure 16; Figure 17 shows the same for the YE stats.

Other stats (Figure 18), not called Y stats, are also displayed on the console. They serve a similar function to the Y stats. The other stats are:

Halt: This stat indicates the machine is in the stop loop with the T clock running. It is set by the micro-order MANUAL. The stat is reset by the console switches or a channel break-in.

Wait: This stat is set by the micro-order SWEA (set wait, enable, ASCII) and ALU output bit 6. The stat indicates that the system is in the wait loop and can be reset by SWEA and not ALU bit 6.

Enable: When this latch is on, a machine error causes either a logout or hardstop. The overall check is ignored when enable is off. Enable is set by SWEA and ALU output bit 5, and is reset by SWEA and not ALU output bit 5.

ASCII: This stat controls the system to operate on data in the ASCII mode. This stat is set by SWEA and ALU output bit 4, and is reset by SWEA and not ALU output bit 4.

CPU Stat: Interpretation of this latch controls whether the data flow paths act as a CPU or a channel. The stat also provides for different interpretation of the CB and CC fields of the ROS word. When the CPU stat is off, the system is operating in I/O mode.

PSA (Protected Storage Address): This stat is set when a protected storage location is addressed.

ISA (Invalid Storage Address): This stat is set when an invalid storage location is addressed.

YCD or *YCI:* These are ALU carry latches for the direct or indirect functions of ALU.

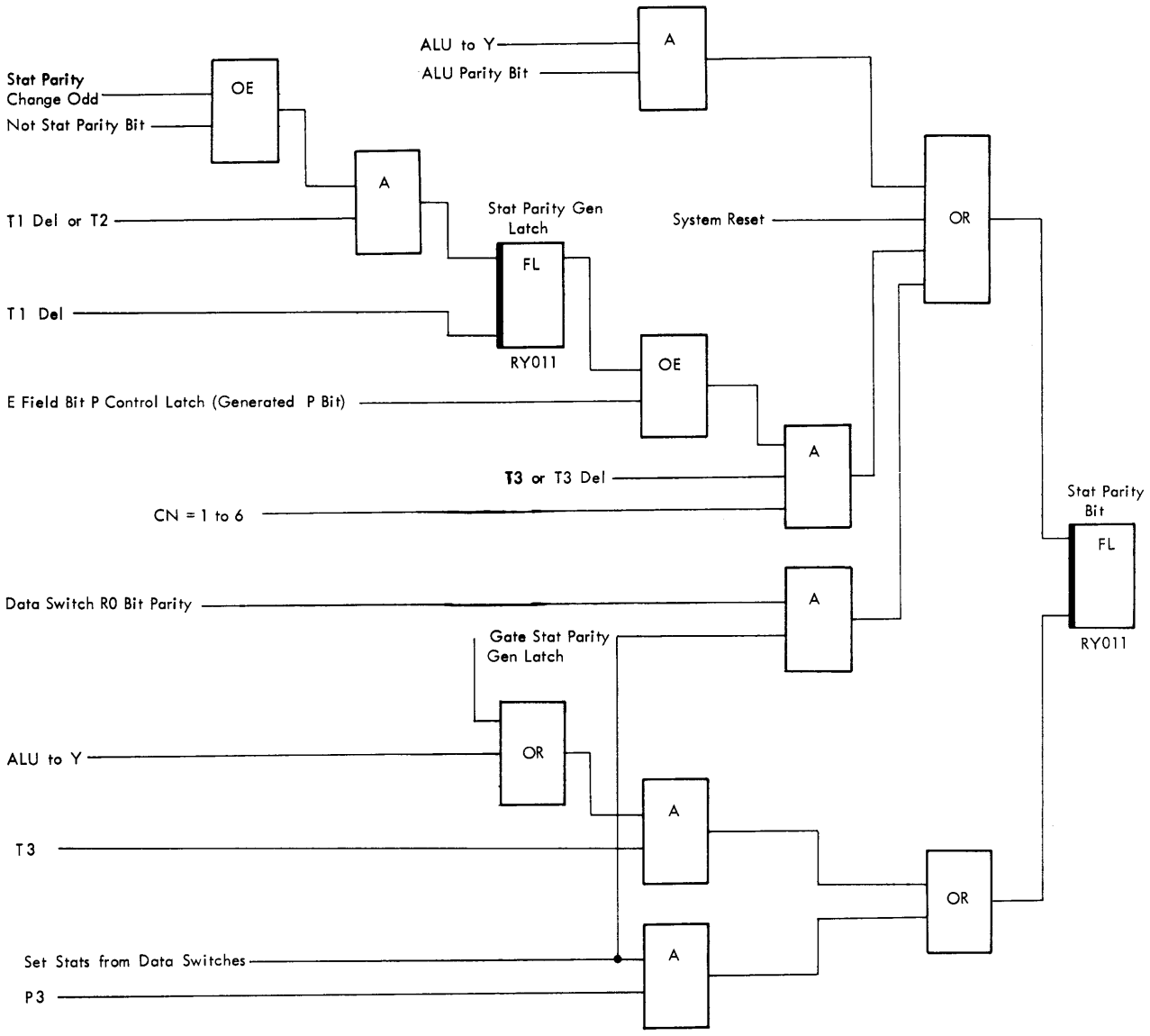


Figure 12. Staticizer Parity Bit Generation

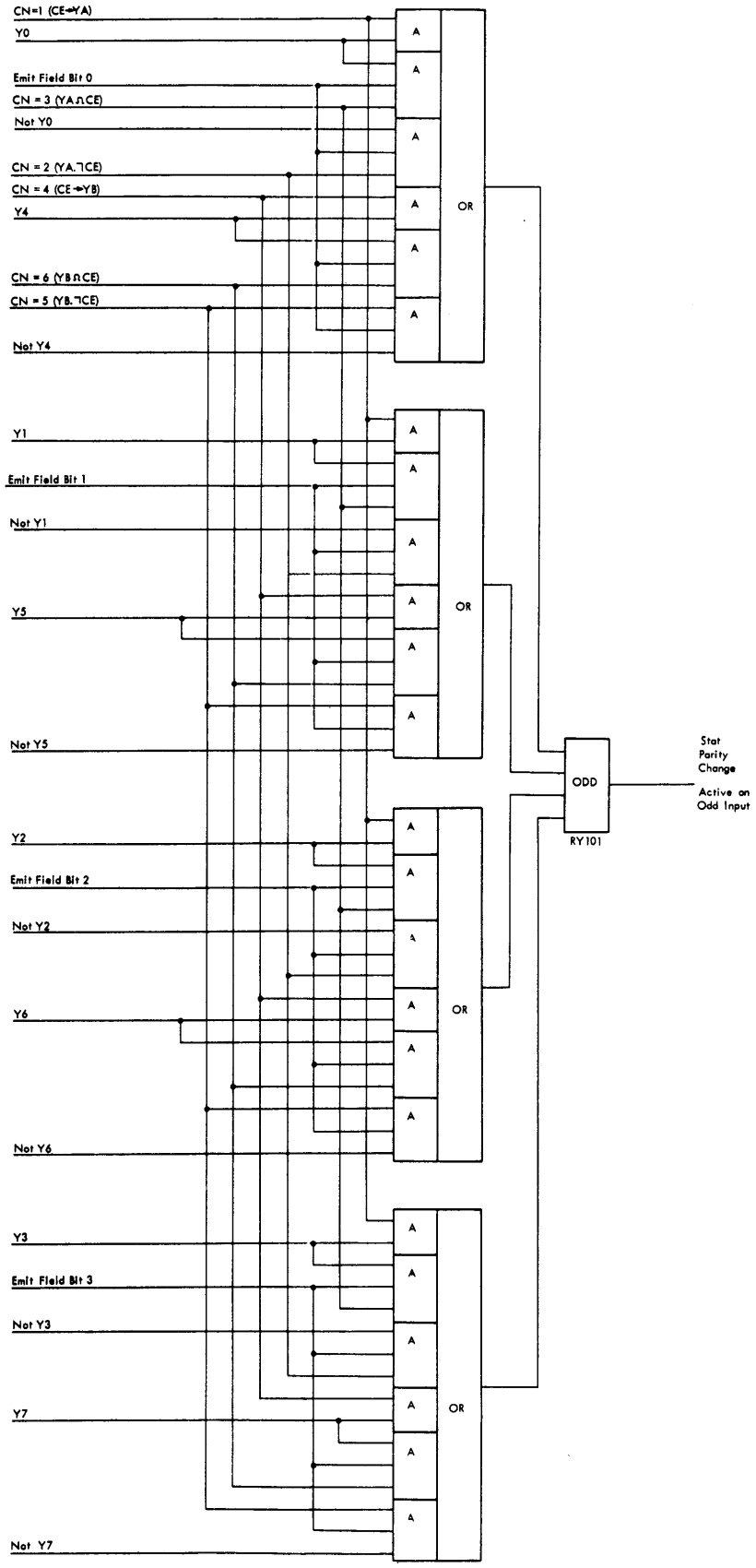
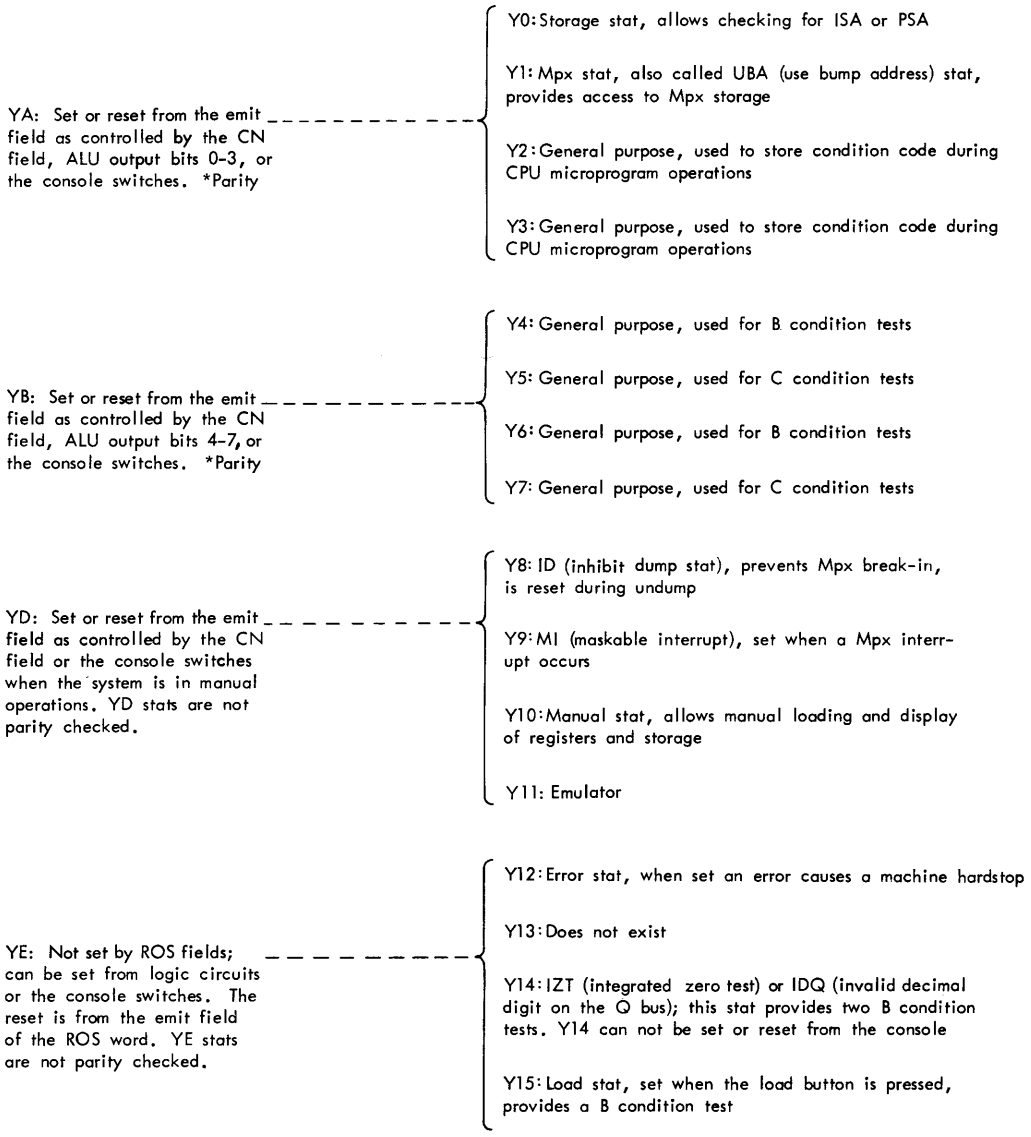


Figure 13. YA and YB Parity Change



*Parity is controlled for stats Y0 - Y7. The parity bit is set by ALU output parity, data switch parity, or the emit field.

Figure 14. Y Stats, Function and Control

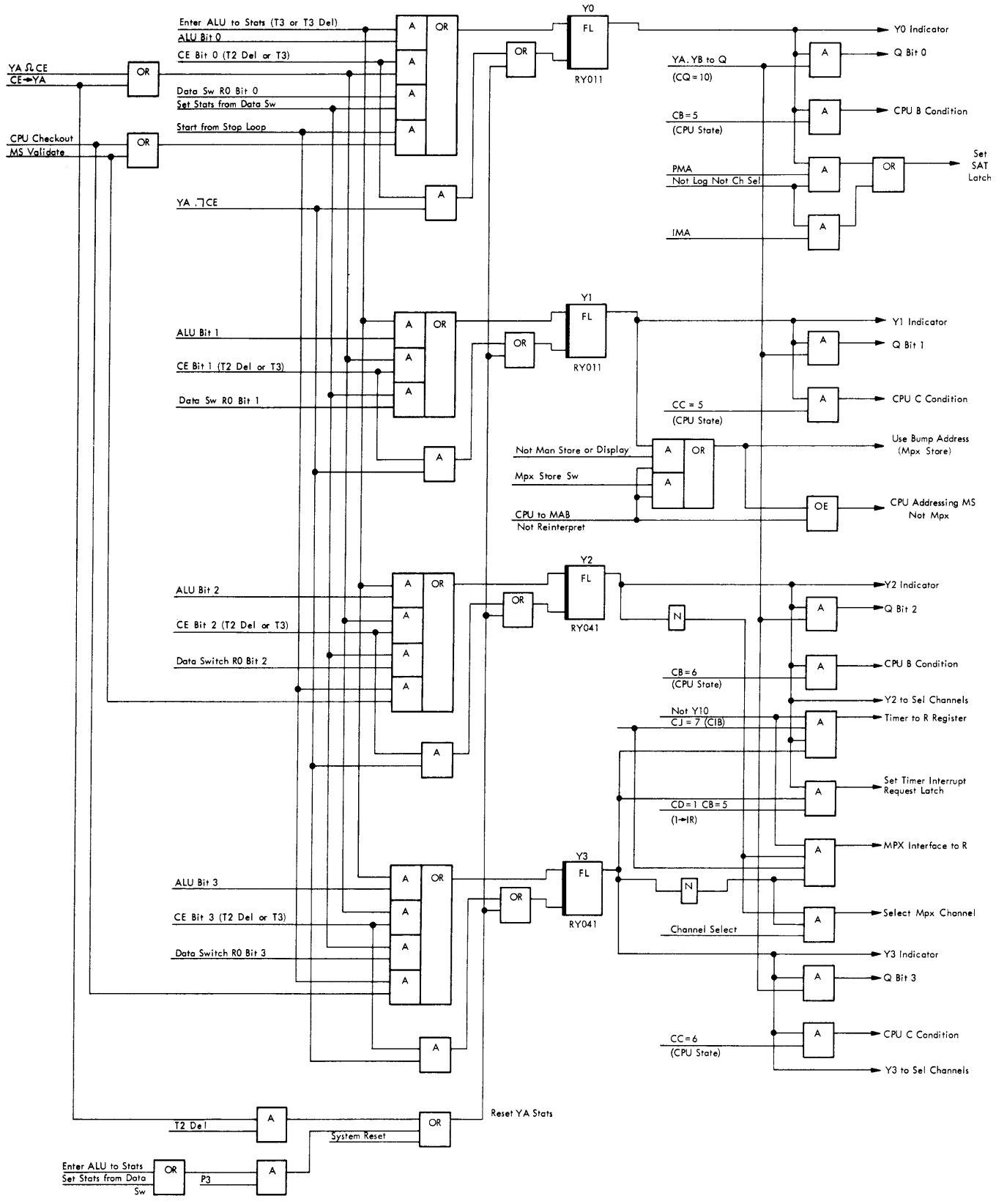


Figure 15. YA Stats

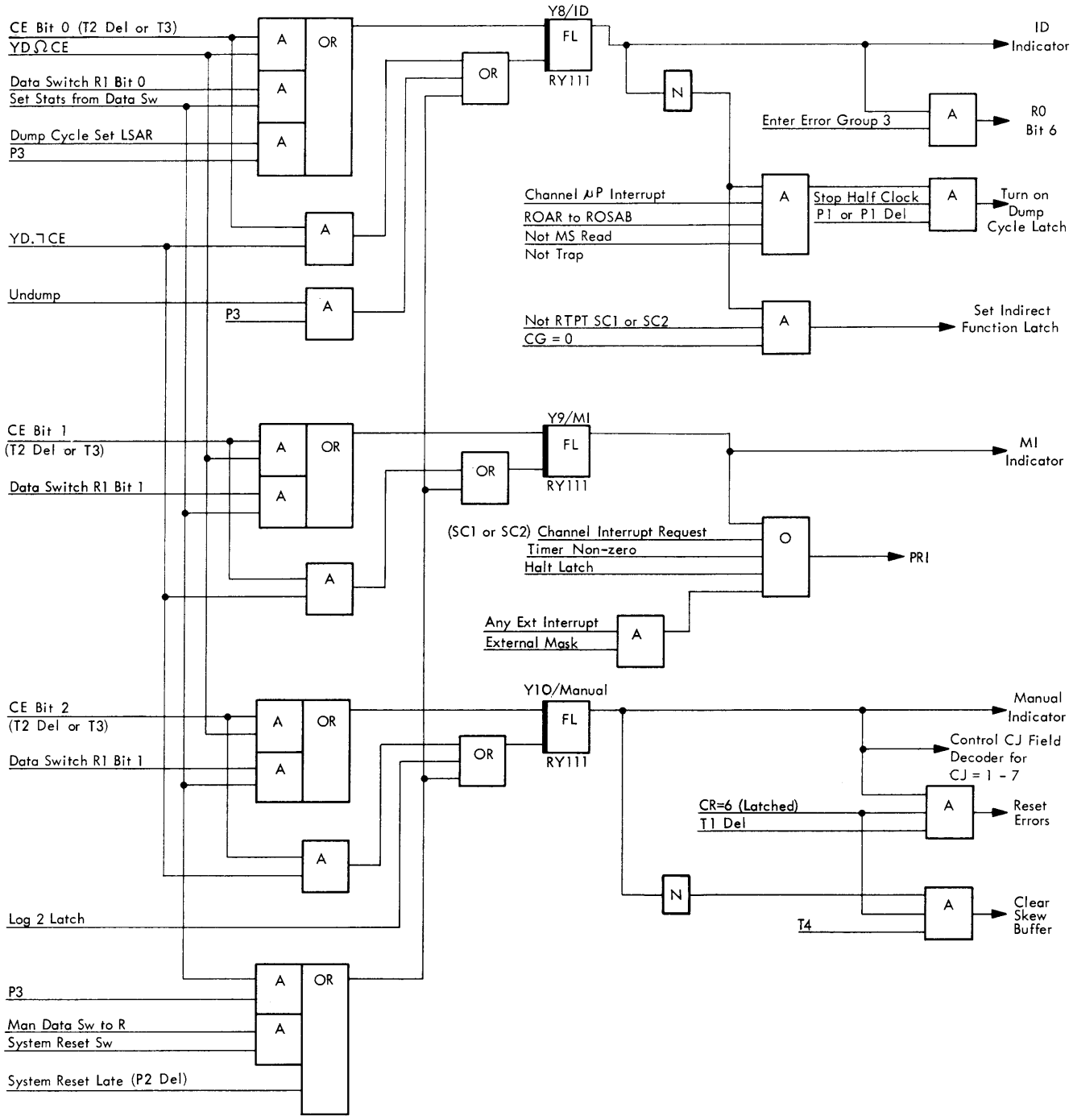


Figure 16. YD Stats

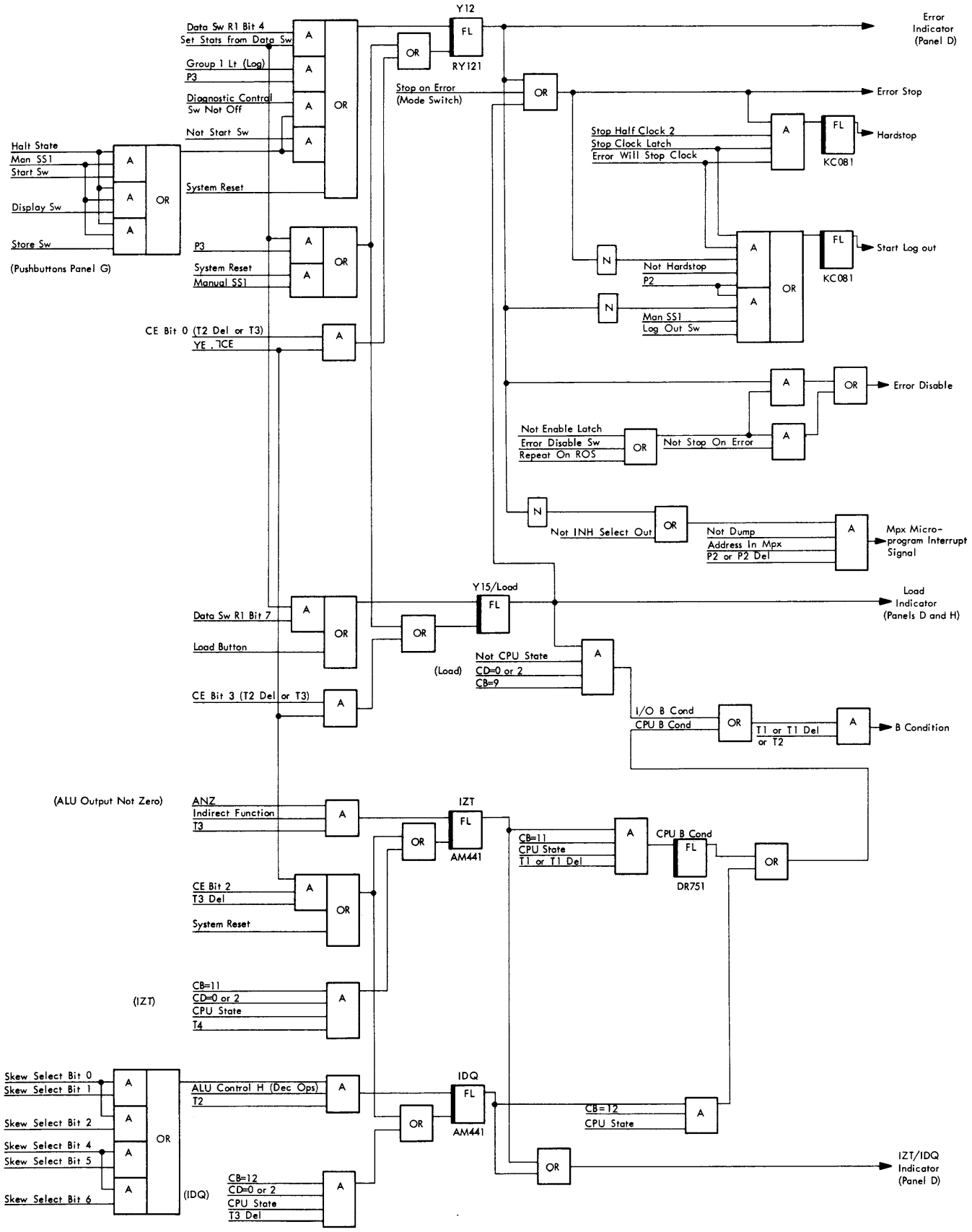


Figure 17. YE Stats

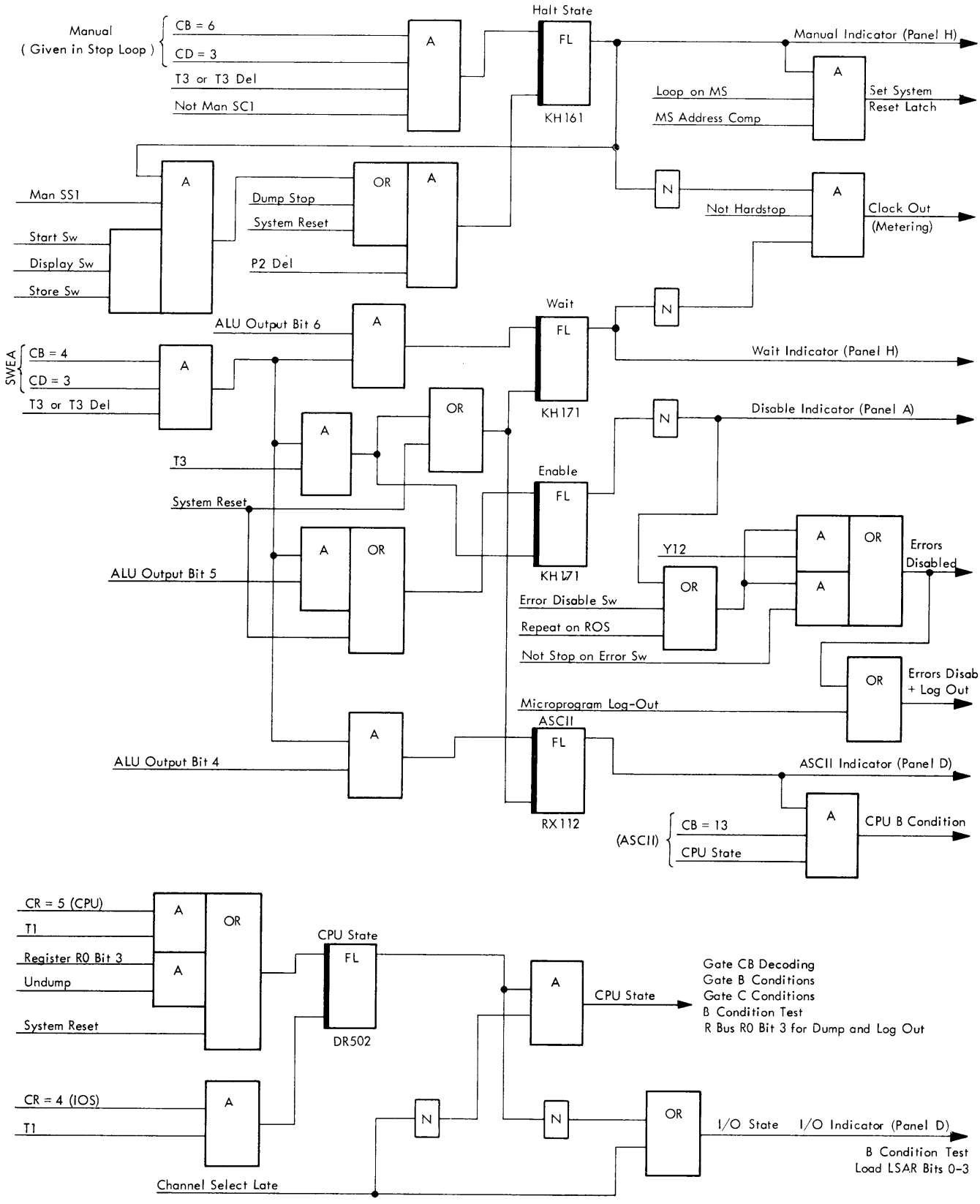


Figure 18. Stats (Halt State, Wait, Enable, ASCII, I/O)

Arithmetic and Logical Unit (ALU)

- **ALU performs arithmetic and logical functions on data held in data flow registers**
- **Basic registers are connected to ALU by an eight-bit data path**
- **ALU processes one data byte at a time**
- **ALU can perform direct and indirect functions**
  Direct functions — a TROS control word field directly controls ALU
  Indirect functions — a five-bit function register controls ALU
- **Three carry latches control carries: YC, YCD, and YCI**
- **P and Q are input registers for ALU**
- **The Q input of ALU can be skewed**
- **Separate circuits handle a possible carry into the extension bits of A and C registers (extended carry)**
- **Extension bits of A and C registers can also be gated to the P and Q buses respectively**
- **The ALU output to AX or CX is compressed**

Note: A glossary of the special terms used in reference to the ALU is given at the end of this section.

The arithmetic and logical unit (ALU) performs arithmetic and logical operations on data held in the data flow registers. Two input registers, P and Q, accept one byte each from gated sources, such as the registers, emit field, or stats.

Inputs to ALU are combined as directed by microprogram. The ALU output bus returns the result to the specified location.

An ALU operation is attempted for every microinstruction. If no operation is required by the program, zeros from the P and Q buses are AND'ed and no output destination is called. An ALU check could therefore occur even when no function is performed.

The unit treats all data as binary data. When a result is developed, any carry generated for each digit will occur after a count of 15 is reached. Additional logic is added to provide for the processing of decimal data. In this case, any generated carry for each digit should occur after a count of 9 is reached. The logic adds 6 to each decimal digit on the P bus during decimal addition to make this possible.

Figure 19 shows the ALU data path. The data on the P and Q buses is latched in the P and Q registers, respectively. If the microprogram calls skew, bits 4-7 of the Q register enter ALU as bits 0-3. The contents of the skew buffer enter ALU as bits 4-7. ALU performs the specified operation on the two bytes and places the result on the ALU output bus.

The ALU is completely controlled by microprogram. Inputs, operation, and gating of the output are all determined by decoding specific TROS fields. Control latches (a four-bit register) are set to give one of sixteen possible operations, eight logical (AND, OR, etc.), six arithmetic, and two shifts (left and right).

The control latch outputs are decoded to indicate one of 16 possible functions (four direct, 12 indirect).

Input registers P and Q are parity checked but the parity is not carried through ALU. The logic is in two-wire form; this means that every signal (and logical state) is represented by two lines of opposite polarity to allow checking throughout ALU. Parity is generated for the output before it is returned to the main data flow.

ALU Control

Input data and the destination of the ALU output are controlled by TROS control fields. The TROS word also controls the setting of the ALU control and function registers. These controls determine whether ALU works on decimal or binary operands and what type of arithmetic operation it performs.

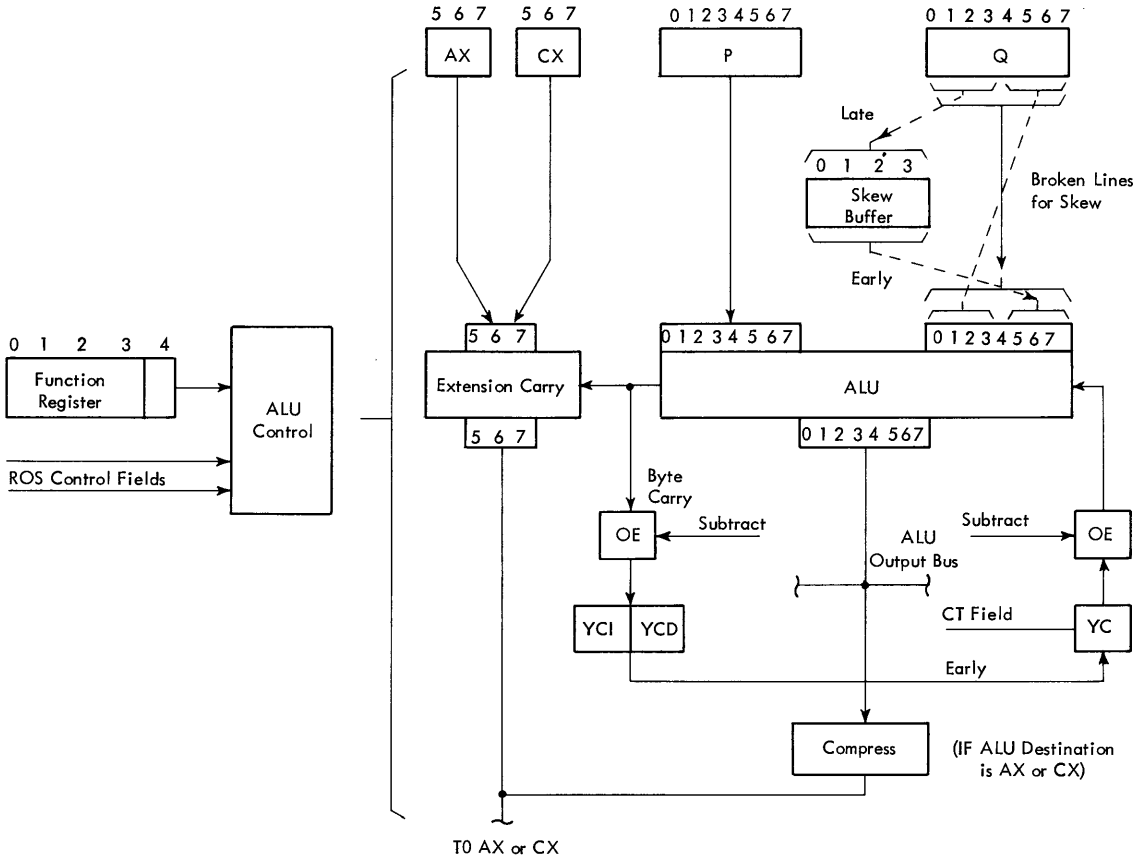


Figure 19. ALU Schematic

TROS Control

Three fields of the control word direct the inputs to the P and Q buses and the output back to the data flow. The CP and CQ fields gate the sources to the P and Q registers and the CM field controls the output destination.

Three other fields, the CG, CE and CN, are used to control the function or operation that the ALU performs during the microinstruction. These fields are decoded to raise a series of control signals.

The listing of functions and the control signals raised for specific functions are shown in Figure 20.

Three further fields CK, CR and CT determine the skew function, reset of the skew buffer, and the set and reset of the carry latches respectively.

The CE, CM and CN fields are latched to allow their output to be available during the whole cycle of the current microinstruction. The CP, CQ, and CN fields are decoder checked to detect an incorrect output. If no lines, or two lines, instead of one are active, the control check latch is set to indicate an error.

The correct bit pattern of the other control fields used in the ALU is checked by the ROS word data check. The outputs of the sense latches and control latches are XOR'ed with the CS field output. This is the parity of the TROS word and an even result gives a TROS data check.

Figure 21 shows the effect of the ROS fields upon the ALU operation.

CP Field

- **Three-bit field**
- **Controls the ALU input to the P bus (setting of the P register)**

CP=0
Symbolic – Z
Function – gate zeros with good parity to P register

	ALU Control Bits	Logical Operation		Active Control Signals											Connect Box Function (F)	Right Shift Function (G)	YC Stat Conditions	YCI or YCD Stat Conditions
		Abb	Meaning	K	L	M	N	H	J	S	W	X	Y	V	U				
*P	0 0000	OR	OR (PQ)	K	L	M									U	P or Q	Zero Output	No Effect on Bit 7	Not Affected
#	1 0001	AND	P . Q	K											U	P and Q	Zero Output	No Effect on Bit 7	Not Affected
	2 0010	DSQ	P - Q (Decimal)	K			N	H		S			Y	V	U	Equivalent P Q	Inverse of Q	YC Off Gives Carry to Bit 7	Set if No Carry Bit 0
*P	3 0011	SUQ	P - Q (Binary)	K			N			S			Y	V	U	Equivalent P Q	Inverse of Q	YC Off Gives Carry to Bit 7	Set if No Carry Bit 0
	4 0100	P	Pass P	K	L										U	Pass P	Zero Output	No Effect on Bit 7	Not Affected
*P	5 0101	AND	P . Q	K											U	P and Q	Zero Output	No Effect on Bit 7	Not Affected
P	6 0110	DSP	Q - P (Decimal)	K			N	H		S		X		V		Equivalent P Q	Q	YC Off Gives Carry to Bit 7	Set if No Carry Bit 0
	7 0111	SUP	Q - P (Binary)	K			N			S		X		V		Equivalent P Q	Q	YC Off Gives Carry to Bit 7	Set if No Carry Bit 0
	8 1000	PNQ	P . $\overline{Q}$		L											P . $\overline{Q}$	Zero Output	No Effect on Bit 7	Not Affected
P	9 1001	Q	Pass Q	K		M										Pass Q	Zero Output	No Effect on Bit 7	Not Affected
P	10 1010	XOR	Exclusive OR (P Q)		L	M										P Q Exclusive OR	Zero Output	No Effect on Bit 7	Not Affected
	11 1011	QNP	$\overline{P}$. Q			M										$\overline{P}$. Q	Zero Output	No Effect on Bit 7	Not Affected
P	12 1100	RSH*	1 Bit Rt Shift of Q								W	X		V	U	Zero Output	Q Shifted Two Positions Rt	YC On Gives Rt Shift Output Bit 1	Set if There is Carry Bit 0
	13 1101	LSH	1 Bit Left Shift of Q									X		V	U	Zero Output	Q	YC On Gives Carry to Bit 7	Set if There is Carry Bit 0
	14 1110	DAD	P + Q (Decimal)		L	M		H	J			X		V	U	P Q Exclusive OR	Q	YC On Gives Carry to Bit 7	Set if There is Carry Bit 0
*P	15 1111	ADD	P + Q (Binary)		L	M						X		V		P Q Exclusive OR	Q	YC On Gives Carry to Bit 7	Set if There is Carry Bit 0

* Direct Function
\# Gives Function Check

Figure 20. ALU Control Signals and Functions

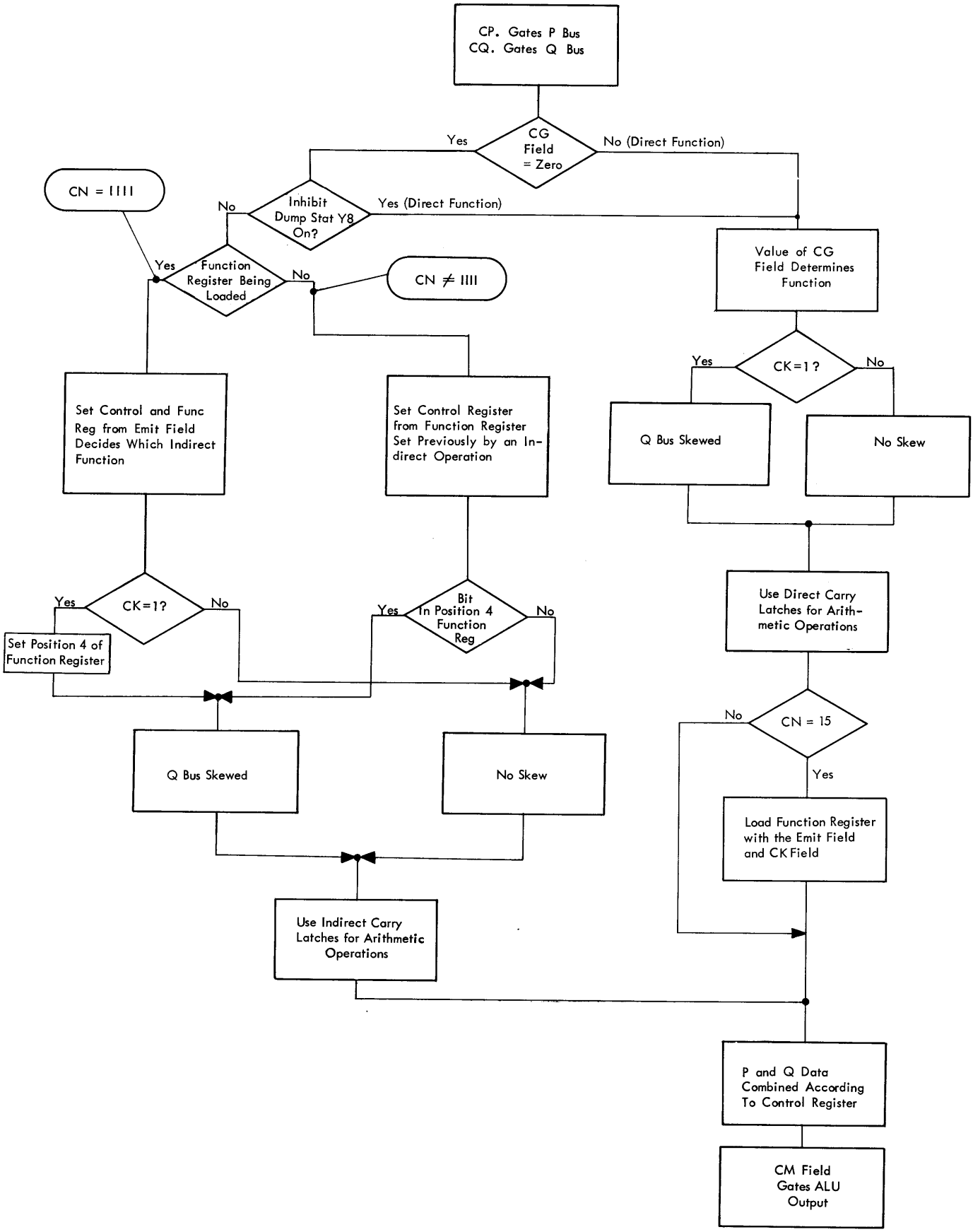


Figure 21. Effects of ROS Fields on ALU

CP=1
Symbolic – AX
Function – gate AX to P register bits 5-7, zeros to 0-4

CP=2
Symbolic – A0
Function – gate A0 to P register, AX to ALU extension

CP=3
Symbolic – A1
Function – gate A1 to P register

CP–4
Symbolic – B0
Function – gate B0 to P register

CP=5
Symbolic – B1
Function – gate B1 to P register

CP–6
Symbolic – E0
Function – gate emit field to P register bits 0-3, zeros to 4-7

CP=7
Symbolic – 0E
Function – gate emit field to P register bits 4-7, zeros to 0-3

NOTE: Zeros are gated to the ALU extension, except when CP=2 or CQ=4. If CP=2 AND CQ=4, CP=2 has higher priority and AX is gated to the ALU extension.

CQ Field

- **Four-bit field**
- **Controls the ALU input to the Q bus (setting of the Q register)**

CQ=0
Symbolic – Z
Function – gate zeros with good parity to Q register

CQ=1
Symbolic – B0
Function – gate B0 to Q register

CQ=2
Symbolic – B1
Function – gate B1 to Q register

CQ=3.
Symbolic – CX
Function – gate CX to Q register bits 5-7, zeros to 0-4

CQ=4
Symbolic – C0
Function – gate C0 to Q register, CX to ALU extension

CQ=5
Symbolic – C1
Function – gate C1 to Q register

CQ=6
Symbolic – D0
Function – gate D0 to Q register

CQ=7
Symbolic – D1
Function – gate D1 to Q register

CQ=8
Symbolic – E0
Function – gate emit field to Q register bit 0-3, zeros to 4-7

CQ=9
Symbolic – 0E
Function – gate emit field to Q register bit 4-7, zeros to 0-3

CQ=10
Symbolic – Y
Function – gate YA and YB stats to Q register

NOTE: Zeros are gated to the ALU extension, except when CP=2 or CQ=4. If CP=2 and CQ=4, CP=2 has higher priority and AX is gated to the ALU extension.

CM Field

- **Four-bit field**
- **Controls the destination of the ALU output**

CM=0
Symbolic – Z
Function – no destination

CM=1
Symbolic – AX
Function – gate ALU output to AX (compressed)

CM=2
Symbolic – A0
Function – gate ALU output to AX and A0

CM=3
Symbolic – A1
Function – gate ALU output to A1

CM=6
Symbolic – B0
Function – gate ALU output to B0

CM=7
Symbolic – B1
Function – gate ALU output to B1

CM=9
Symbolic – CX
Function – gate ALU output to CX (compressed)

CM=10
Symbolic – C0
Function – gate ALU output to CX and C0

CM=11
Symbolic – C1
Function – gate ALU output to C1

CM=12
Symbolic – Y
Function – gate ALU output to Y stats (Y0-Y7)

CM=14
Symbolic – D0
Function – gate ALU output to D0.

CM=15
Symbolic – D1
Function – gate ALU output to D1

NOTE: CM=2 or 10. Carry, if any, propagated through extension.

CG Field

- **Two-bit field**
- **Controls the ALU function**
- **For CG=0 the function performed depends on Y8 and REINT control**

CG=0 (Not REINT and Y8=0)
Symbolic – ?
Function – indirect ALU function

CG=0 (REINTor Y8=1)
Symbolic – Ω
Function – direct OR function

CG=1
Symbolic – .
Function – direct AND function

CG=2
Symbolic – –
Function – direct subtract function (P–Q) binary

CG=3
Symbolic – +
Function – direct add function, binary

CE Field and CN Field

- **Source of emit data**
- **Four bits wide, any number from 0-15 entered into data flow**
- **If CN field=15, ALU function register loaded from CE field**

For CN = 15 the emit field specifies the following indirect ALU functions:

CE=0
Symbolic – OR
Function – OR P and Q

CE=1
Symbolic – ACK
Function – ALU functional check

CE=2
Symbolic – DSQ
Function – P–Q decimal

CE=3
Symbolic – SUQ
Function – P–Q binary

CE=4
Symbolic – P
Function – pass P only

CE=5
Symbolic – AND
Function – AND P and Q

CE=6
Symbolic – DSP
Function – Q–P decimal

CE=7
Symbolic – SUP
Function – Q–P binary

CE=8
Symbolic – PNQ
Function – P AND'ed with NOT Q

CE=9
Symbolic – Q
Function – pass Q only

CE=10
Symbolic – XOR
Function – exclusive OR P and Q

CE=11
Symbolic – QNP
Function – NOT P AND'ed with Q

CE=12
Symbolic – RSH
Function – right shift Q by one bit

CE=13
Symbolic – LSH
Function – left shift Q by one bit

CE=14
Symbolic – DAD
Function – P+Q decimal

CE=15
Symbolic – ADD
Function – P+Q binary

CK Field

- **One-bit field**
- **Controls the ALU skew function**
- **If ALU function register is loaded in same microinstruction, the skew bit is stored in function register bit 4**

CK=0 – No function, Q data not skewed
CK=1 – Q input to ALU skewed

CR Field

- **Three-bit field**
- **Controls miscellaneous ALU functions**

CR=6 and Y10 off
Symbolic – 0→SK
Function – reset the ALU skew buffer to zero (after the current cycle)

CT Field

- **Two-bit field**
- **Controls set and reset of the ALU carry latches**

CT=2
Symbolic – 0
Function – the YC stat is reset at P1 of the current cycle

CT=3
Symbolic – 1
Function – the YC stat is set at P1 of the current cycle

Function Control

ALU functions can be either direct or indirect.

For direct functions, the ALU control circuits are determined by the TROS control word; for indirect functions the ALU performs the operation specified by the five-bit function register.

The function register is also under control of the TROS control word, but once it is set (from the CE field) it stores the information until it is reset.

Direct and indirect functions have separate associated carry latches, YCD and YCI respectively.

The existence of the two distinct types of ALU functions is justified as follows.

1. The ALU function to perform (add, subtract and shift, etc.) is specified in one microinstruction and set into the function register. This approach may save a number of TROS control words.

In routines where similar machine instructions are executed, (for example, RR fixed-point add and subtract) a number of microinstructions may be identical except for the ALU function. In this case a common routine can be shared by both instructions, the only difference being the entry to the common routine where the add instruction would set the function register to binary add and the subtract instruction would set the function register to binary subtract.

2. For microprogram control operations (main storage address modification, loop control for variable field lengths etc.), the ALU has to be available between actual data operations. Without the possibility of two different controls, a step to preserve ALU information (function and carry) would be necesary before the ALU could be used again.

Control Register

- **Decoded to determine ALU function**
- **Set from CG field or CE (emit) field or from function register**
- **Displayed on console**
- **Reset at T4 del**

Function Register

- **Used during indirect operations**
- **Positions 0-3 loaded from emit field**
- **Reset by reloading**
- **Position 4 loaded from CK field, determines skew operation**
- **Set when CN=15, at T4 del**

The CG field can directly determine the function or select the function register that is loaded from the emit field for an indirect operation. The bit pattern in the control register is decoded to raise the control signals for the particular operation.

Eleven control signals are used. Figure 22 shows the control register content, the signals and the functions. A twelfth signal, U, is generated to make the number of signals even or odd to match an even or odd number of bits in the control register (not counting control register parity).

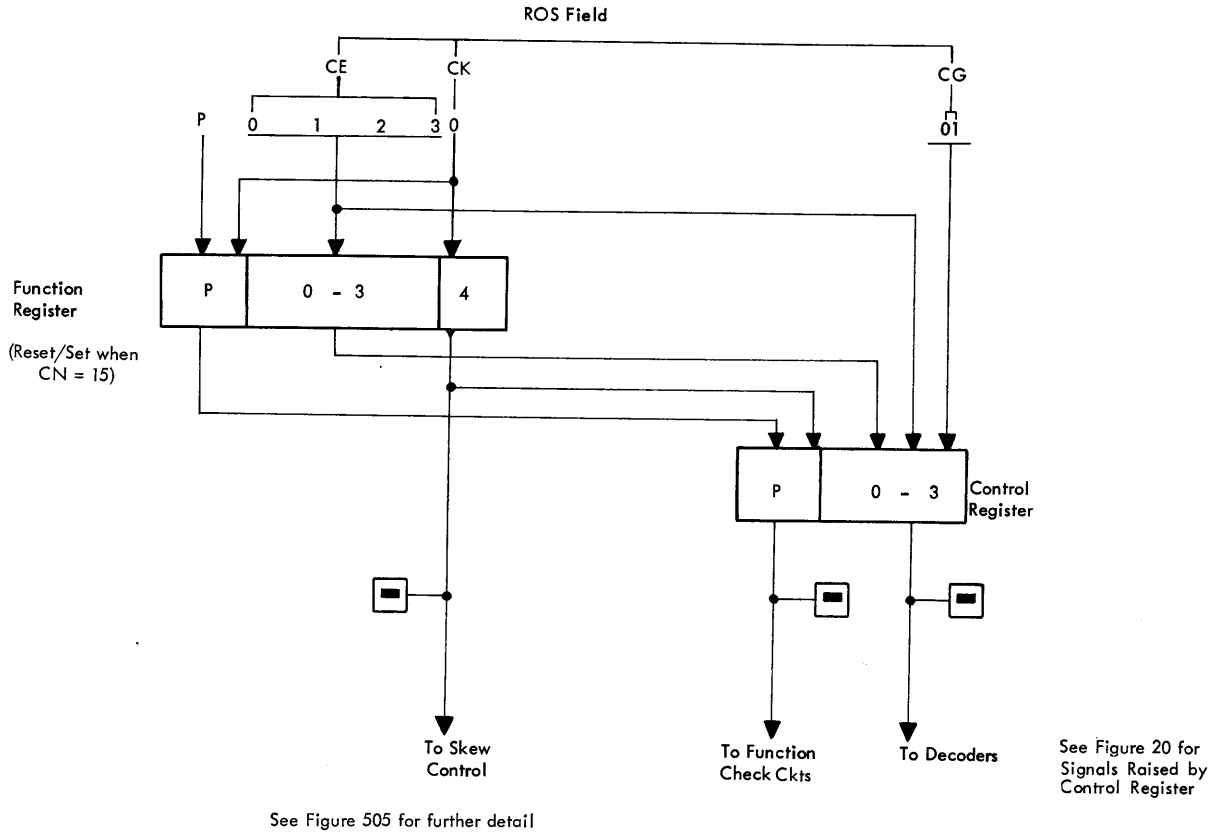


Figure 22. Function and Control Registers

Control Signals (Figure 20)

Consider bits 0-3 only of the control register and the control signals K, L, M, N, H, J, S, W, X, Y, V, U. In this sequence U is a pseudo control signal used for checking reasons. Controls are identified as follows:

1. An even number of control register bits raises an even number of control signals.

2. An odd number of control register bits raises an odd number of control signals.

3. K, L, M, N: determine the function of the connect box. Combinations of these signals are used to control the specified function or operation.

4. H: is active for all decimal operations; controls decimal correction and a subtract of 6 from decimal digit is performed if necessary. This is necessary if there is no carry from the high-order position.

5. J: is active for decimal addition.
   Hex 66 is added to P in the decimal filler.

6. S: is active for subtract; inverts the carry into bit position 7 for YC; inverts the carry out of bit position 0 into YCI/YCD; inverts the carry from bit position 0 for EX carry operations.

7. U: has no effect on ALU functions; is generated to check control decoding. A function check occurs if the decoding check is incorrect.

8. V: is active for arithmetic and shift operations; gates carry from bit position 0 into YCI/YCD. If V is not active, YCI/YCD regenerated (carry regenerated for nonarithmetic operations). Gates YC into bit 7 of ALU if *no* right shift. If right shift, then control W blocks this action. W is active for right shift (two-bit shift of Q data) blocks normal carry into bit 7 from YC stat; gates bit 7 of Q to bit 1 of right shift box.

9. X: if active, Q bits pass through right shift uninverted if no W. If W is present, then right shift 2 places as above. X is active for right shifts and all arithmetic operations except subtract Q from P.

10. Y: is active only for subtract Q from P operations. Inverts the skew select output (Q bus data) through the right shift unit.

Function Checks

Figure 23 shows the logic associated with the function checks and gives a table of the input conditions. In this circuit the control signals forming the circuit inputs are not all used in their positive-polarity form.

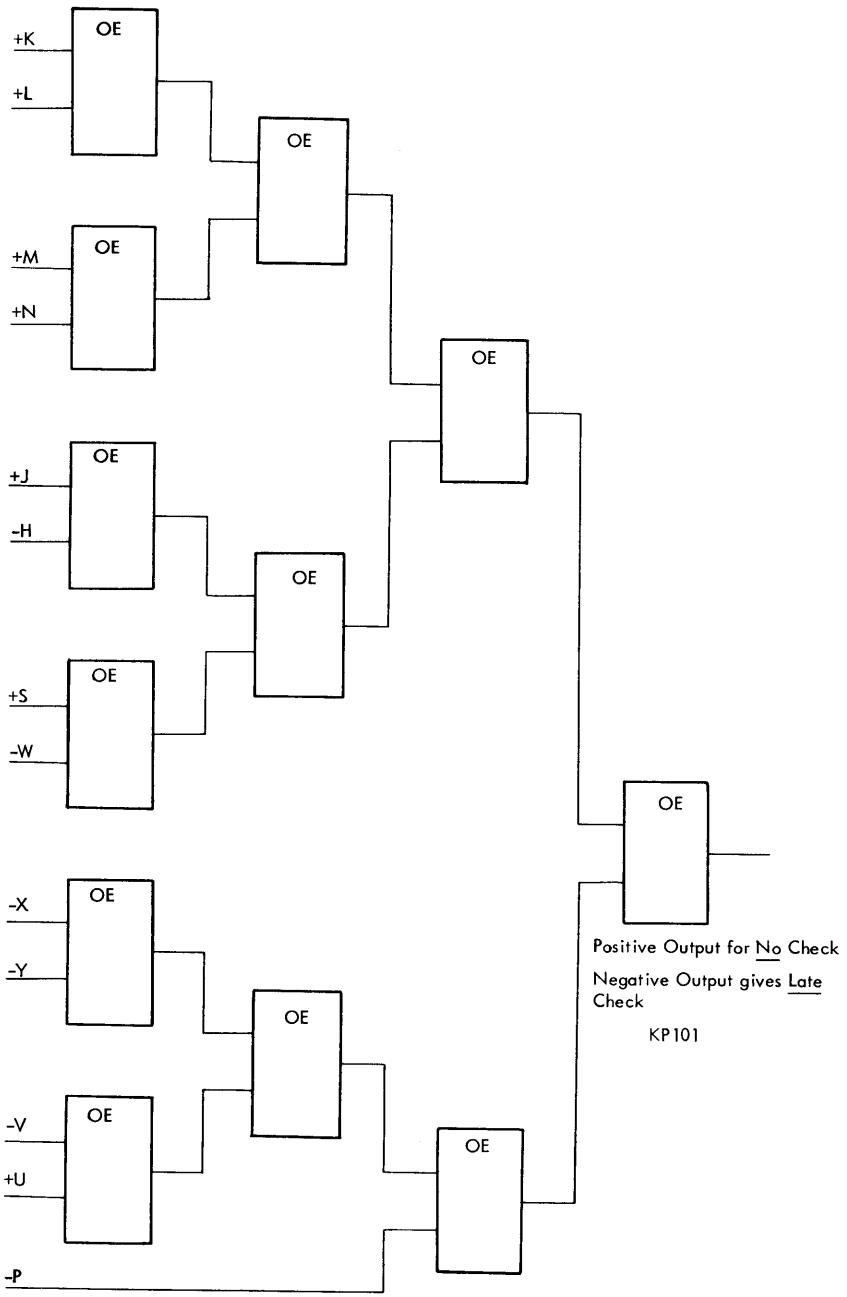


Figure 23: **Function Check Signals**

	CONTROL REGISTER VALUE					ACTIVE SIGNALS USED IN CHECK CIRCUIT											
HEX	BITS 0 - 3				P	K	L	M	N	$\overline{H}$	J	S	$\overline{W}$	$\overline{X}$	$\overline{Y}$	$\overline{V}$	U	$\overline{P}$
0	0	0	0	0	1	X	X	X		X			X	X	X	X	X	
≠ 1	0	0	0	1		X				X			X	X	X	X	X	X
2	0	0	1	0		X			X			X	X	X			X	X
3	0	0	1	1	1	X			X	X		X	X	X			X	
4	0	1	0	0		X	X			X			X	X	X	X	X	X
5	0	1	0	1	1	X				X			X	X	X	X	X	
6	0	1	1	0	1	X			X			X	X		X			
7	0	1	1	1		X			X	X		X	X		X			X
8	1	0	0	0			X			X			X	X	X	X		X
9	1	0	0	1	1	X		X		X			X	X	X	X		
A	1	0	1	0	1		X	X		X			X	X	X	X		
B	1	0	1	1				X		X			X	X	X	X		X
C	1	1	0	0	1					X					X		X	
D	1	1	0	1						X			X		X		X	X
E	1	1	1	0			X	X			X		X		X		X	X
F	1	1	1	1	1		X	X		X			X		X			

≠ Deliberate Function Check (even number of active control signals)

X Inputs which are positive for that control register value

A late check occurs if there is no output from the last exclusive-OR stage. This arises because the circuit fed from the last exclusive-OR stage requires a negative input.

An odd number of positive inputs produces a positive output. This is shown in Figure 23 together with a table of the inputs that are positive when the control register has the value shown. An odd number of positive inputs should always be present to give a final positive output and a 'no check' condition. When the control register has a value of 1, a deliberate check is raised by an even number of inputs. This facility is used to verify the operation of the check circuits.

Arithmetic

The ALU handles both binary and decimal arithmetic operands. The ALU operations can be understood by referring to ALU schematic data flow, Figure 19.

Binary Arithmetic

- **Carrys out of bit 0 are lost**
- **Overflow occurs if carrys out of bit 0 and bit 1 are not equal**
- **ALU adds or subtracts operands as directed**
- **A 1 bit in the most significant position indicates a negative binary number**
- **Negative binary numbers are processed through the ALU in 2's complement form**
- **Negative results are in 2's complement form and are returned to storage in the same form**

During addition, the connect box performs the function exclusive-OR to produce the sum output of the first half-add stage. The half-add carry is produced by the AND of one of the input bits with the absence of a half-sum. The second half-add carry is the AND of a carry input and the presence of a half-sum. The final sum is produced by performing the function exclusive-OR between the half-sum and the carry.

During subtractions, the connect box is set up to perform the function Equivalent. This has the effect of inverting the P bits. Thus the subtraction Q minus P (Q−P) is performed. To perform the subtraction P minus Q (P−Q) the output of the right-shift box is inverted. This inverts the Q bits.

When subtracting the first (i.e. least significant) byte, the incoming carry is arranged to be 1. The two's complement is the inverted data plus a one in the least-significant position. This one is generated by the carry in. See Figure 24 for examples of binary subtraction.

Decimal Arithmetic — True ADD

- **Correction of 6 added to first operand**
- **Second operand binary added**
- **If high order of packed digit generates a carry, the result is correct**
- **If no carry generated, result decreased by 6 (add 2's complement, 1010). Ignore carry from this operation**

Subtraction

+13	0.01101	Minuend	-13	1.10011
-(+11)	0.01011	Subtrahend	-(+11)	0.01011
	0.01101			1.10011
	0.10100			1.10100
	1			1
+2	0.00010	Difference	-24	1.01000
+13	0.01101		-13	1.10011
-(-11)	1.10101		-(-11)	1.10101
	0.01101			1.10011
	0.01010			0.01010
	1			1
+24	0.11000		-2	1.11110
+11	0.01011		-11	1.10101
-(+13)	0.01101		-(-13)	1.10011
	0.01011			1.10101
	1.10010			0.01100
	1			1
-2	1.11110		+2	0.00010

Rule

In all cases, add the 1's complement of subtrahend to minuend and add a carry into least-significant bit. Ignore carry out of sign position.

Note 1: Negative numbers are held in 2's complement form in storage.
Note 2: Only six binary positions are shown.

Figure 24. Examples of Fixed-Point Binary Subtraction

Decimal Arithmetic — Complement ADD

- No correction to first operand
- Add 2's complement of second operand
- If high order of packed digit generates a carry, the result is correct
- If no carry generated, result decreased by 6 as for true ADD
- If sign change occurs, the result has to be recomplemented

Treatment of Negative Decimal Numbers

- Signs of operands tested before an operation to determine true or complement ADD function
- Held in true form in storage with negative sign

ALU Operation

The ALU can perform logical or arithmetic operations as controlled by the TROS words.

Logical Operations

OPERATION	CONTROL REGISTER VALUE	CONTROL SIGNALS ACTIVE
OR	0000	K, L, M, U
*AND	0001	K, U
PASS P	0100	K, L, U
P AND Q	0101	K, U
NOT Q	1000	L,
PASS P	1001	K, M
X OR	1010	L, M
NOT P AND Q	1011	M

The following applies to all logical operations:
Right shift output is zero.
No carrys into bit 7 of ALU.
No inter-bit carry of ALU byte.
Connect box alone decides function.
*AND operation 0001 is used only to bring up function check.

Shifts

OPERATION	CONTROL REGISTER VALUE	CONTROL SIGNALS ACTIVE
Right Shift Q	1100	W, X, V, U
Left Shift Q	1101	X, V, U

Right Shift Q

- Shifts Q bus content only
- Connect box output zero
- Carry box causes one place left shift
- Q data shifted two places right to give final right shift of one place
- Data on P bus unimportant except that parity should be satisfied (connect output zero)
- Multiple byte shifts done byte by byte left to right
- YCI, YCD and YC stats allow bit to shift from byte to byte

Left Shift Q

- Connect box output zero
- Q data passed through to carry box which produces carry out for that position
- Result is an output for the next-higher bit position, or left shift of one position

Arithmetic Operations

OPERATION	CONTROL REGISTER VALUE	CONTROL SIGNALS ACTIVE
P−Q Decimal	0010	K, N, H, S, Y, V, U
P−Q Binary	0011	K, N, S, Y, V, U
Q−P Decimal	0110	K, N, H, S, X, V,
Q−P Binary	0111	K, N, S, X, V,
P+Q Decimal	1110	L, M, H, J, X, V, U
P+Q Binary	1111	L, M, X, V,

P − Q Binary (P minus Q)

- No decimal filling on P bus
- Q data inverted through right-shift box
- Meaning of carry stats inverted
- Effect of carry out of bit 0 inverted
- 1 added into least-significant bit of first byte only
- Decimal correction occurs if no carry from high-order position
- A result may be in 10's complement form
- Recomplement by further pass through ALU and subtract from zero

P − Q Decimal (P minus Q)

- Same as P−Q decimal except no decimal correction

Q − P Decimal (Q minus P)

- Same operation as for P−Q decimal except no inversion of Q data

Q − P Decimal (Q minus P)

- Same as for P−Q decimal except no decimal correction and no inversion of Q data

P+Q Decimal (P plus Q)

- Decimal fill on P bus data
- Decimal correction if no carry from high-order position

P+Q Binary (P plus Q)

- Straight binary addition

NOTE: The ALU Worksheet (as shown in Figure 25) can be used to prove any type of ALU operation.

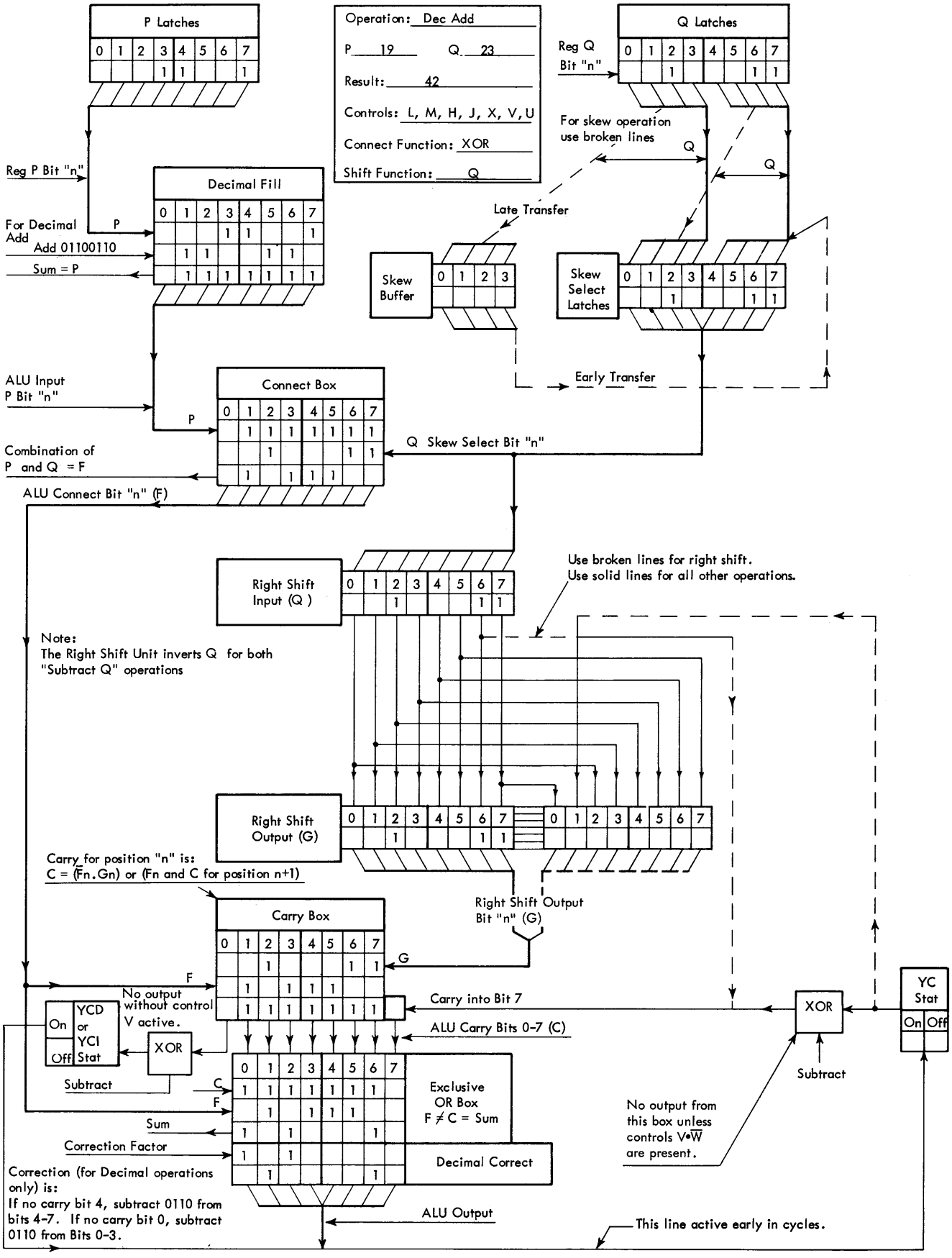


Figure 25. ALU Work Sheet (Sheet 2)

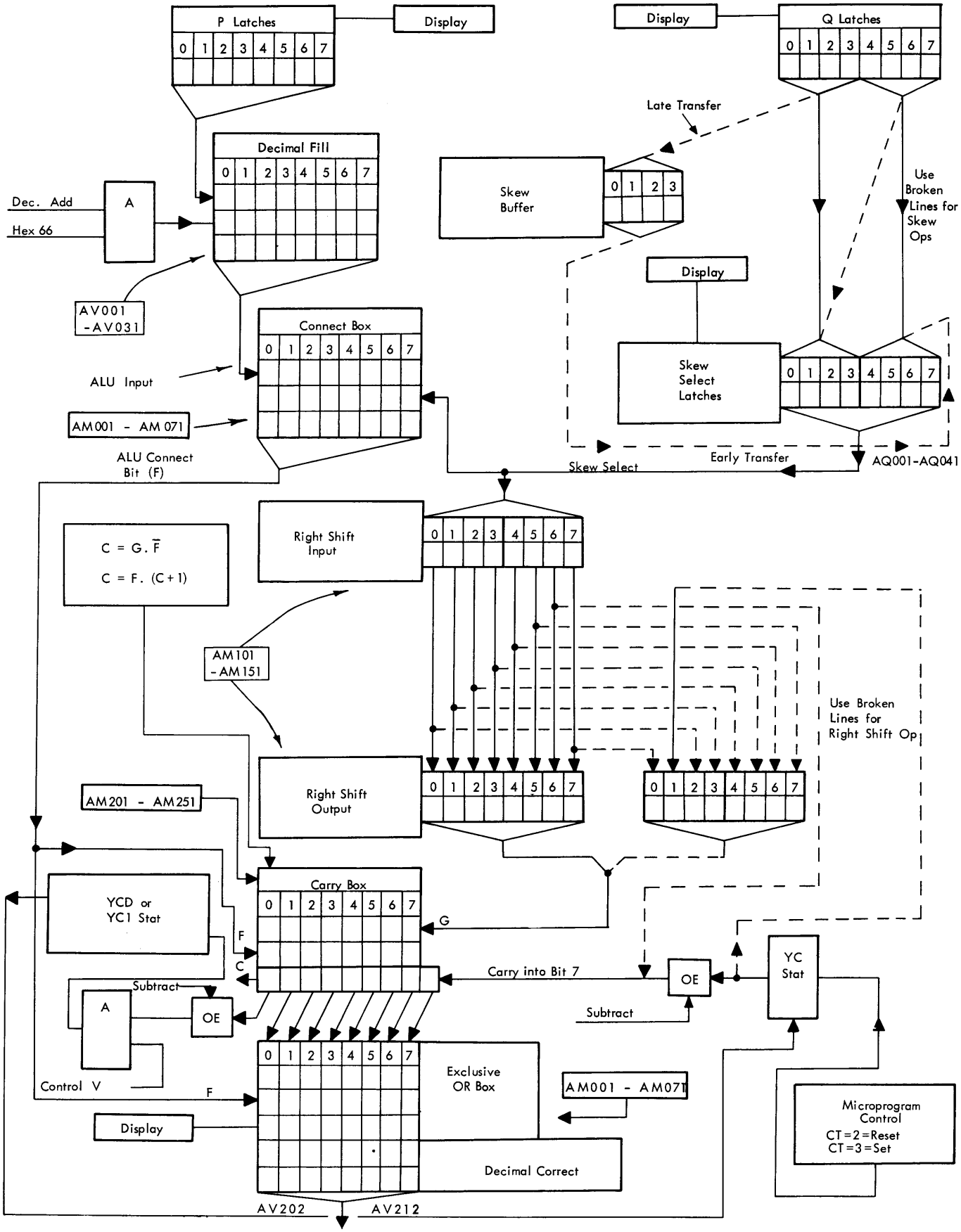


Figure 25. ALU Work Sheet (Sheet 1)

$C_n = (\bar{F}_n \cdot G_n) + (F_n \cdot C_n + 1)$

- **Carry stats checked after operation to determine true or complement result**
- **Correct sign generated by microprogram before result is stored**

Decimal data are stored as two decimal digits per byte. Each digit is in four-bit binary-coded form. The ALU normally adds four-bit digits naturally in the scale of 16, i.e. hexadecimal (hex).

For a decimal add operation, 6 is first added to each digit on the P bus by the decimal filler logic. Since each digit must have a value less than 10, the resulting sum will be less than 16 and no carry out of that digit position is generated. A normal hex addition then takes place.

If a carry out of that digit position is generated during the hex addition, the resulting sum is correct. If no carry is generated, it was not necessary to add 6 to give the correct sum. In this situation, 6 must be subtracted. The decimal correct box is controlled, by the absence of this digit carry, to subtract 6.

During decimal subtraction, 6 must be subtracted from the sum digit when a borrow is required from the next digit. Since the presence of a borrow is represented by the absence of a carry, the situation is exactly the same as for addition except that no filling is required at the input.

Carry Control

Carry Latches YCI and YCD

- **YCI is indirect function carry stat**
- **YCD is direct function carry stat**
- **Affected only by arithmetic or shift operations**
- **Receives carrys from bit position 0 late in ALU cycle**
- **Subtract operation inverts effect of carry from bit 0**

Carry Latch YC

- **Set by microprogram or from YCI or YCD stats**
- **Set early in ALU cycle**
- **If set, gives carry into bit position 7 for arithmetic and left-shift operations**
- **Inverts effect during subtract operation**

The ALU output carry (byte carry) is stored either in the YCD or YCI stat, depending on the current ALU function (direct or indirect). The YC stat is needed to give immediate carry availability to the ALU operation.

The YCI or YCD latch is on from previous operation and as soon as the function (direct or indirect) is determined, the YCI and YCD latch output is gated to YC at T1 or T1 del. The stat is reset every cycle at T1.

The YCI or YCD receives its condition at T3, late in the ALU cycle; this allows the microprogram to test the YCI and YCD conditions for possible branching to the next TROS address. Figure 26 shows the carry latches.

Either YCI or YCD is reset at T2 del every cycle, as the cycle must be an indirect or direct operation. If the function being performed is a logical function, the carry has not been used at T1 or T1 del. The latch has been reset, so that the carry must be regenerated to retain the original condition. If the YCI or YCD stat was on, the line carry stat YC will be active, and, depending on whether the function was indirect or direct the set carry stat YCI or YCD will be made active, and the appropriate stat is set again at the end of the ALU cycle.

The operation described above relates to CPU microprogram execution. When a selector channel interrupt occurs, the ALU direct carry (YCD) is preserved in an additional latch: the CPU carry latch. The line, not selector channel late, gates YCD to set this latch at T3 or T3 del for a selector channel interrupt. This is necessary because channel operations involving the ALU are direct operations which may require the use of the YCD carry latch.

The additional CPU carry latch is not reset during a channel operation as it is normally reset at T3 gated by direct and not channel select late. The original direct carry then remains available until the next CPU direct operation.

It is not necessary to store a direct carry produced by a channel if interrupted because significant carrys are not allowed to extend beyond a read/write cycle of main storage. Channel interrupts cannot occur during this time.

The carry stats are in two-wire form, thus one of the pair is always on.

Plus and minus signals (called carry condition) which set the YCI or YCD stats depending upon the function, are also the inputs to the two-wire checking circuits for these stats. If a check occurs, the ALU 2-wire check output carry line is active. The check for the YC stats is taken directly from the YC and NO YC output, and incorrect operation activates the line ALU 2-wire check input carry.

Both checks light the two-wire check lamp on the main console and give a machine late check. The internal CE panel distinguishes between a YCI/YCD or YC stat error.

During subtract operations, reset carry (RC) is called to add one to the low-order first byte to form the two's complement. Also, if there is a carry out during subtract, the stat is tested for the off condition (affect inverted).

The carry circuits are gated by an ALU control signal called subtract. Subtract is generated whenever a complement add of Q is called. The carry out and the carry in signals are XOR'ed with subtract. A block diagram of the carry latches is given in Figure 26.

Example:

True ADD operation; subtract is down. The byte carry is propagated into bit 7 of the next byte.
Complement ADD operation; subtract is up. YCD/YCI: not set with an output carry, set with no output carry. Carry to bit 7 of next byte: no carry if YC is on, carry if YC is off. In other words, the carry is still properly propagated to the next byte.

This arrangement has special significance only for the first byte of a complement addition. Remember how the two's complement of a binary number is obtained: invert every bit of the number and add a 1 to the lowest-order bit. This "add 1" for the first byte processed is automatically done by the previously explained arrangement.

Tests and Conditions

- **Results of ALU operations give testable conditions to determine part of next TROS address**
- **B condition tests, if successful, force bit 1 of ROAR to 1**
- **C conditions tests, if successful, force bit 0 of ROAR to 1**
- **Conditions not latched up; tested only in cycle in which they occur**
- **Latched conditions available until tested or changed by subsequent operation**

As a result of an ALU operation, several tests and conditions can force either or both of the two low-order bits of the next TROS address to the 1 state. These are known as the B condition test and C condition test and are taken as a result of the value in the CB and CC fields respectively of the current TROS word. A successful test for a B condition sets bit 1 of ROAR to 1 and for the C condition sets bits 0 of ROAR to 1.

B Condition Tests CPU State and CD = 0 or 2

CB=2
Symbolic – YCD
Function – 1 forced if YCD stat on.

CB=4
Symbolic – Ψ ALU ≠ 0
Function – 1 forced if ALU output non-zero.

CB=9
Symbolic – Ψ FXPTO
Function – 1 forced if fixed-point overflow.

CB=10
Symbolic – Ψ ALU 7
Function – 1 forced if ALU output bit 7 is 1.

CB=11
Symbolic – IZT
Function – 1 forced if IZT latch on.

CB=12
Symbolic – IDQ
Function – 1 forced if IDQ latch on.

Note: Ψ means tested only in cycle in which condition occurs (at T2 del).

C Condition Tests, CPU State

CC=2
Symbolic – YCD
Function – 1 forced if YCD stat on.

CC=4
Symbolic – Ψ ALU ≠ 0
Function – 1 forced if ALU output non zero.

CC=9
Symbolic – Ψ ALU 6
Function – 1 forced if ALU output bit 6 is 1.

CC=10
Symbolic – Ψ ALU 0
Function – 1 forced if ALU output bit 0 is 1.

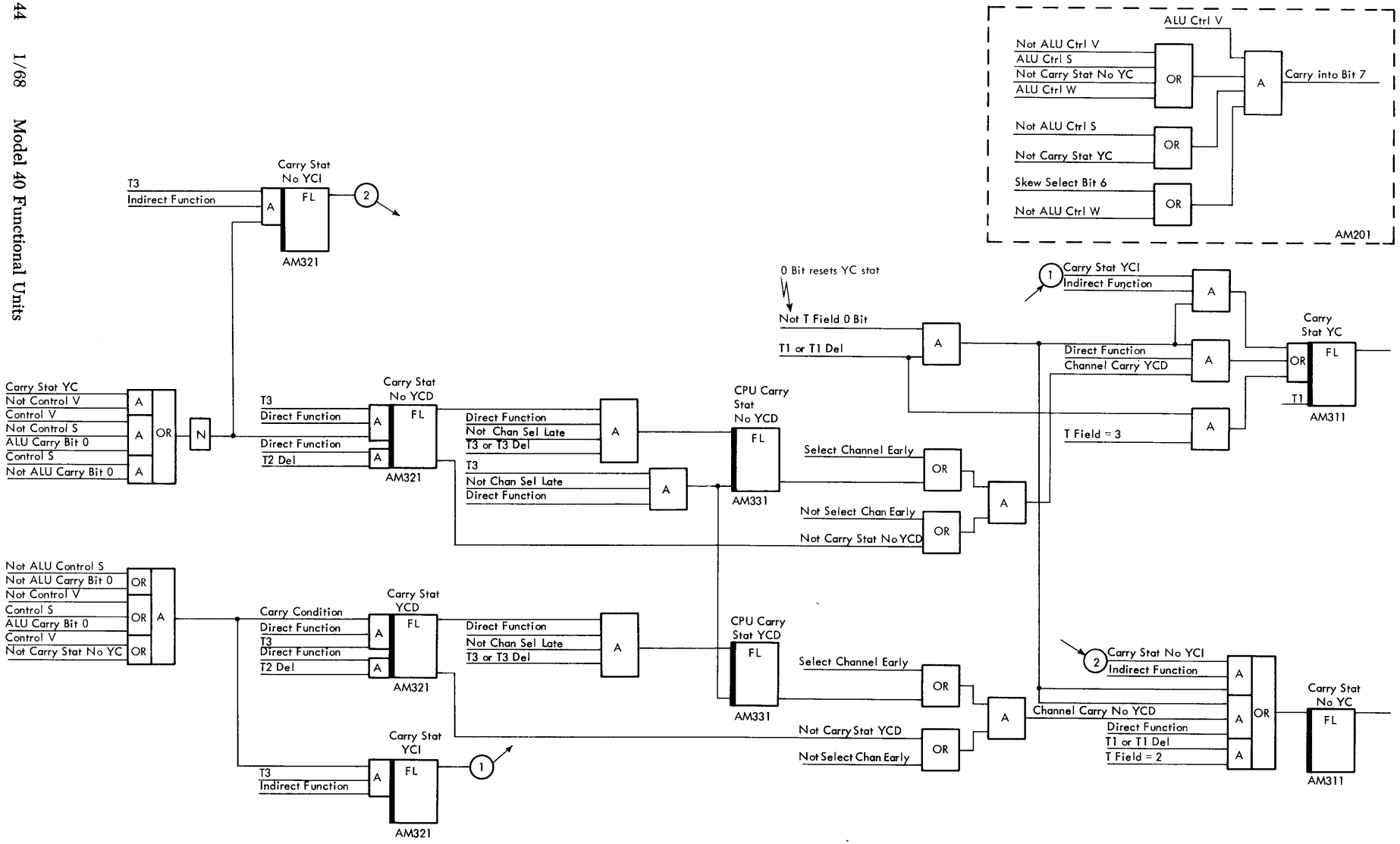
CC=11
Symbolic – Ψ Q0 ≠ 0
Function – 1 forced if Q register bits 0-3 non zero.

CC=13
Symbolic – YCI
Function – 1 forced if YCI stat on.

CC=15
Symbolic – Ψ QPTY
Function – 1 forced if bad parity in Q register.
Note: Ψ means tested only in cycle in which condition occurs (at T2 del).

Additional Conditions

The TROS section of this manual gives details of the formation of the next TROS address and the B and C condition tests.
The conditions that can be tested only in the cycle in which they occur are not latched up. The sampling time is T2 del.



Notes:
1. Control S is active for subtract operations.
2. Control V is active for arithmetic and shift operations
3. The ALU 2-wire system uses two latches for each stat function. When one of these latches is set, the other latch must be reset.
4. T field controls microprogram set and reset of the carry stat YC:
   CT = 2, reset YC stat (actually prevents the set of YC stat).
   CT = 3, set YC stat

Figure 26. ALU Carry Latches

The latched conditions IDQ (invalid decimal digit on the Q bus) and IZT (integrated zero test) remain set until tested or reset by microprogram. The YCI and YCD latches remain testable until changed by a future indirect or direct operation respectively.

All the B and C conditions can be changing in the cycle in which they are tested.

Summary of ALU Conditions

YCD Direct Carry Stat
Direct function and carry from bit 0.
Indirect function and YCD stat on.

ALU $\neq 0$
ALU output non-zero (ANZ).

FXPTO, Fixed Point Overflow
Carry bit zero and no carry bit one.
Carry bit one and no carry bit zero.

ALU 7
ALU output bit 7 equals one.

IZT, Integrated Zero Test (Y14)
Sets IZT/IDQ on main console.
Set if ALU output non zero on indirect function.
Reset when tested by microprogram.
Not reset by output of all zeros if previously set.
If tested in cycle and is changing, the new value is tested, then latch is reset.

IDQ, Invalid Decimal Digit on Q Bus (Y14).
Sets IZT/IDQ on main console.
Latch set if invalid decimal digit in skew select latches during decimal operation.
Reset when tested by microprogram.
If set, decimal operation with valid digits does not reset IDQ.
Invalid decimal digits are A through F.
If IZT/IDQ is tested in a cycle when it is changing, the new value is tested.

ALU 6
ALU output bit 6 equals one.

ALU 0
ALU output bit 0 equals one.

Q0 $\neq 0$
Bits 0-3 of Q bus non zero (QNZ).

YCI, Indirect Carry Stat
Indirect function and carry from bit 0. Direct function and YCI stat on.

QPTY, Bad Parity in Q Latches
Blocked if direct data, external or channel interrupts being gated to Q register.

Figure 25 shows the ALU worksheet. This can be used to prove any ALU operation.

ALU Circuits

Two-Wire Logic

- **Each logical state represented by two lines of opposite polarity**
- **Two-wire check for errors made on output and carry circuits**
- **Errors indicated by bit position on internal CE panel**
- **Over-all two-wire check indication on main console also causes late check**

The ALU logic is executed in two-wire form, i.e. each and every signal in the ALU is represented by two signals, one positive and one negative at all times. The following considerations are made in the ALU circuit design:

1. Each function generator in the pair must be independent of the other generator so failure in one generator will not be reflected in the pair.

2. Since the generators of a pair may have some common input signals, the generators must be designed so that a change in polarity of a single wire of an input pair cannot cause a change of output in both generators, i.e. if a single input wire changes from positive to negative it must not cause the output of one generator to change from positive to negative and the other generator output to change from negative to positive. This can be guaranteed by arranging that any path from input to output of one generator has exactly the same number of inversions as any path through the other generator of a pair.

Two-wire checking is made on the binary output and carry circuits. A fault gives a console indication and causes a late check. The specific bit position in error can be identified from the error lights on the internal CE console.

P and Q Input Latches

- **One byte of data set into both P and Q latches**
- **Seven possible sources for P latches**
- **P latch entry under microprogram CP field control**
- **14 possible sources for Q latches**
- **Q latch entry under microprogram CQ field control**
- **Latches set at beginning of ALU cycle**
- **Reset at end of ALU cycle**

Each bit position of the registers is a latch set from the source and gated by the signal appropriate to the decoded value of the controlling fields.

The setting of the registers is dependent on the TROS word sense latch output and the decoding of the fields. The setting is early in the ALU cycle so that the data can be available. Reset occurs at P4 or P4 del time, ALU timing, Figure 27.

Both the P and Q registers are parity checked and any error is indicated on the main console; the early check light is also lit and the early data check latch is set if a parity error occurs.

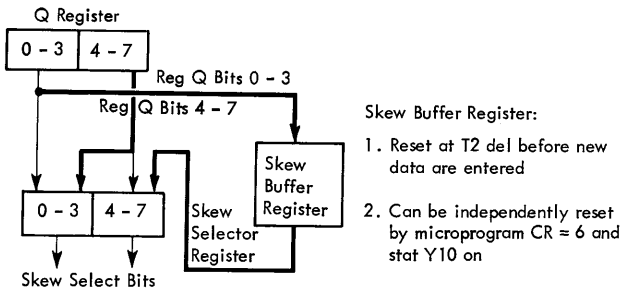


Figure 28. Skew Select Schematic

Skew Feature

- **Affects output of Q latches only**
- **Active when skew called; otherwise simple transfer path**
- **Four-bit left shift of data**
- **Bits 4-7 become bits 0-3**
- **Bits 0-3 enter a buffer stage; used next skew cycle**
- **Current buffer content becomes bits 4-7**

When a skew operation (Figure 28) is called by microprogram, a four-bit left shift is applied to the byte on the Q bus. The contents of a four-bit buffer are moved to the lower half of the incoming shifted byte and the most-significant half of the input byte is entered into the buffer. This buffer content is then available for the next skew cycle; this need not be the next ALU cycle. Bits 4-7 of the original byte become bits 0-3 in the skew select register.

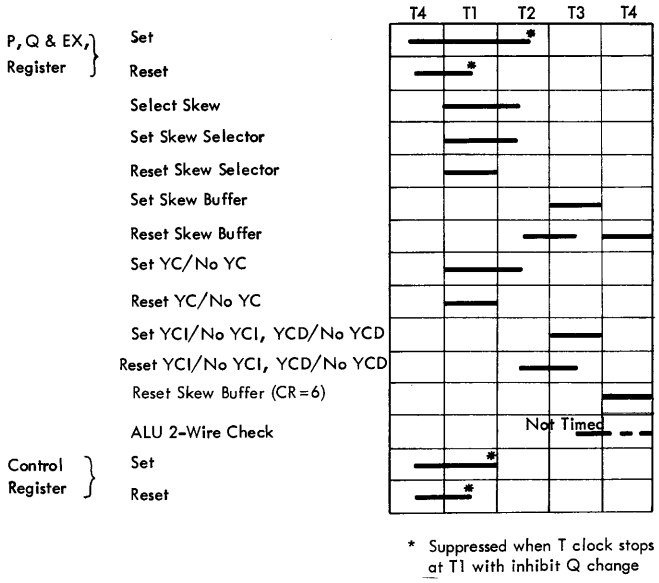


Figure 27. ALU Timing Chart

A parity bit is generated for the skew buffer as a result of bits 0-3 of the Q register being odd and skew called, or if the buffer is reset by microprogram or system reset. Microprogram reset and system reset put zeros in the buffer.

The skew select register also has a parity bit which is the parity of the Q register if no skew is called, or the XOR of bits 4-7 of the Q bus and skew buffer parity.

Incorrect parity in the skew select register raises a parity check indicated on the main console; it also gives a late check. The skew select register is displayed on the main console. It should be remembered that this is not necessarily the same data as would be in the Q register.

Decimal Filler

- **Input is from the P input latches**
- **Active in decimal add mode only, otherwise simple transfer path**
- **Modifies bits 0-3 and 4-7 by adding 6 to each group**
- **Effectively adds 0110 0110 (66) to whole byte**

This circuitry effectively adds six to each decimal digit (valid or invalid) entering from the P register. The reason for adding six has been explained earlier, in the arithmetic section, which describes the processing of decimal arithmetic in the ALU. Control signal J, active for the decimal add operation only, controls the logic to modify the input bit pattern. Under normal operation, the data pass through unaltered.

Right Shift Unit

- **Accepts Q bus data either skewed or unmodified**
- **Right shift operation, data shifted two positions to the right**

- **Arithmetic operations (except P minus Q, binary and decimal), no modification of data**
- **P minus Q operations, output is inverted**
- **For logical operations, output is zero**

The right shift unit is in the Q bus data path and the input is from the skew select register. For normal arithmetic operations (except P minus Q, both binary and decimal) the data pass unaltered.

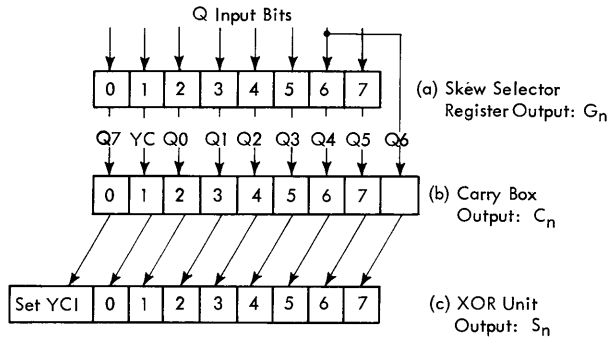
For P minus Q operations the logic inverts the input bit pattern and for logical operations the data path is blocked (output zero). When control signal W is active (for a right shift operation) the logic shifts the Q input two bit positions to the right.

Right shift is performed by setting up the ALU as for left shift (which is done in the carry box) and shifting the Q bus input to the carry box two places to the right. This is done to simplify control of entry of the leading bit into the carry stat.

The output of the right shift unit is given the symbol "G". A block diagram showing a right shift operation is in Figure 29.

Connect Box

- **Combines output of P latches (via decimal filler) and Q latches (via skew)**
- **Four control signals used K, L, M, N**
- **9 of 16 possible combinations of control signals are used**
- **Output is zero for shift operation (left and right)**



Boolean expressions for the stages above:

	True	Complement (During Subt)	Right Shift	
(a) $G_n =$	$\overline{W}XQ_n$	+ $Y\overline{Q}_n$	+ WQ_{n-2}	Q bus input shifted two positions right
(b) $C_n =$	F_nC_{n+1}	+ $\overline{F}_nG_n$		No connect function; therefore, carry out for every R or S input
(c) $S_n =$	$F_n\overline{C}_{n+1}$	+ $\overline{F}_nC_{n+1}$		Effective left shift one position, carry in, and no connect function

Figure 29. Right Shift Box (Example Right Shift)

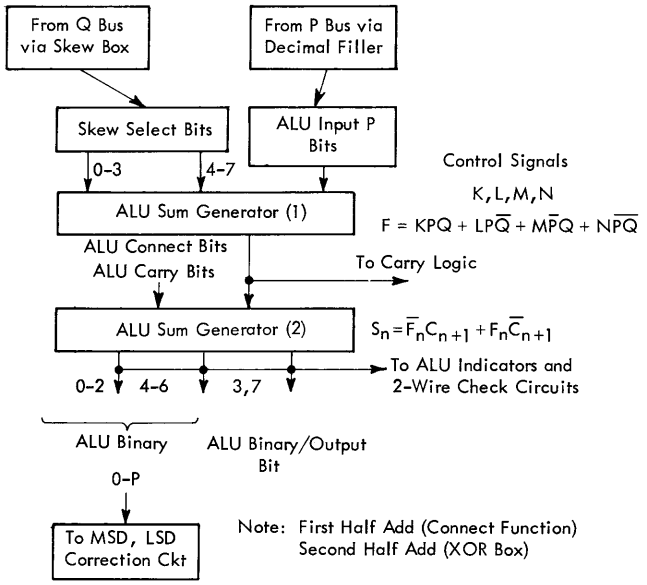


Figure 30. ALU Sum Generator

The connect box is the first stage of the sum generator, and is referred to as the first half add stage. It combines the P and Q data as determined by the four control signals. No inter-bit carrys are generated at this stage; this is decided later by considering the output of this circuit and the Q data. Figure 20 shows which signals are active for any particular function.

The simplified logic of this unit together with that for the carry and XOR (second half add stage) are shown in Figure 30. The operation of these additional stages is explained following this section.

The output from the connect box is called the connect function and is given the symbol "F".

The terms show that the logic is controlled by the signals K, L, M, and N which gate the AND combinations of the P and Q data both in the "bit" and "no bit" states.

Carry Box

- **Combines connect box and right-shift box output (Q bus data)**
- **Determines bit-to-bit carrys within the byte**
- **Accepts any carry into the low-order position**
- **Produces any carry out of the high-order position**

A carry out of a bit position occurs if there is an input from the right shift logic and no connect function for that position. A carry out is also obtained if there is a connect function bit for the position and a carry in from the adjacent low-order stage.

An inter-byte carry held in the main carry stats can enter the low-order position as required and any carry out from the high-order position may be routed to the carry stats. This represents the straightforward logic for obtaining a carry out of a bit position. To reduce logic delay through the ALU, "Look-ahead" is used in the carry path making the logic more involved.

For any bit position, by looking back to the connect function inputs to lower-order bits, the logic predicts a carry out state for that position. This is in addition to the normal carry logic.

The carry path over the first four stages (7-4) uses straightforward logic with one level of delay for each stage. (Since the input bits are staggered, there is nothing to be gained by look-ahead over these stages.)

The carry out of stage 3 (C3) looks ahead to C6. C0 and C1 both look ahead to C3. This eliminates four levels of delay to produce the final carry (C0) and the final sum output.

Figure 31. Carry Box and Look-ahead Equations (deleted)

XOR Box

- **Combines carry and connect box output**
- **Exclusive OR of these inputs for all ALU functions**
- **Output is ALU binary output. Modifications may be needed if in decimal mode**
- **Output displayed on main console**

The final sum is produced by performing the function exclusive OR between the first half sum (connect function) and the output of the adjacent low-order carry stage.

The final sum is the ALU binary output; if the operation was decimal, a further stage, the decimal correct logic, may modify this.

The console display of the ALU output is fed from this XOR stage and is not necessarily the final output that is routed back to the data flow, if in decimal mode. A two-wire check is also fed from the XOR output.

If an error is detected, a two-wire check occurs which causes a late check. Figure 30 shows the XOR box and its relation to the connect box and carry box.

Decimal Correction

- **Accepts ALU binary output**
- **Active in decimal mode only**
- **Modifies bits 0-3 (MSD) and 4-7 (LSD) by minus 6 if no carry from bits 0 and 4 respectively**
- **If either digit has value less than 6 then add 2. Exception is MSD value of 5 only**
- **No modification if carry is detected, from high-order digit**
- **Output is ALU output**

If a carry is sensed out of the high-order bit of either digit (ALU bit 4 and bit 0 for the LSD and MSD respectively), the resulting sum is not decimal corrected for both valid and invalid decimal digits. The carry is propagated into the high-order digit from the LSD or sets YCI (indirect carry stat) if from the MSD.

If no carry is sensed out of a high-order bit, the resulting sum must be decimal corrected. For both the MSD and LSD, all sums falling between 6 and 15 are decimal corrected by subtracting 6.

Values between 5 and 0 for the LSD and 4 and 0 for the MSD have 2 added to them. The odd case of MSD value equals 5 is corrected by subtracting 6. This special case is required by the microprogrammer and affects the sign handling of a particular routine.

A simple example as follows verifies that adding 2 gives the same result as subtracting 6 except for the affect on the high-order digit. As subtracting 6 from a value between 0 and 5 would produce an invalid decimal digit, adding 2 produces a valid similar result but with no high-order bit present. The circuitry modifies the data when the control signal H is active during decimal operations.

ALU Output Parity

- **Parity generated for ALU output**
- **Parity is true for all ALU functions, binary and decimal**
- **No check on parity in ALU; bad parity would be detected at output destination**
- **Parity bit displayed on main console**

Both the P and Q input registers are parity checked, but the parity of the data is not carried through the ALU operation because the logic is in two-wire form. When the ALU output is formed, a new over-all parity bit must be generated.

Parity is generated for bits 0-3 and 4-7, (as parity of whole byte is ODD, then if a parity bit exists for bits 0-3, there is no parity bit for bits 4-7 and vice-versa) and XOR'ed with not change MSD parity and not change LSD parity respectively. These two signals are generated as a result of any modification to the binary outputs during a decimal operation. This could affect the final parity bit.

The results of these combinations are also XOR'ed to produce the final ALU parity. This will be true parity for whatever function has been performed and is not the parity of the binary output alone. This bit is displayed on the main console.

NOTE: If the ALU is in decimal mode, the parity displayed need not correspond with the displayed binary output.

Figure 32 shows the parity logic in simplified form.

ALU Extended Carry

- **Extension (EX) bits 5, 6, 7 modified only if carry out of bit 0, ALU**
- **If adding (includes shifts) plus 1, change to EX 5, 6, 7 but EX 5 never changed 1 to 0**
- **If subtracting minus 1, change to EX 5, 6, 7 but EX 5 never changed 1 to 0**
- **No modification during logical operations**
- **Parity modified as necessary**

This special ALU circuit provides main-storage address updating within two microinstructions.

In all ALU operations where the high-order byte of A or C register (A0 or C0) is specified as ALU input, the corresponding extension bits (AX or CX) are automatically gated to the extended carry circuits.

If both A0 and C0 are ALU inputs, AX overrides CX.

The extended carry circuits update AX or CX contents with the possible byte carry from the main ALU function.

The output of the extended carry circuits is gated to AX or CX whenever the ALU output is gated to A0 or C0.

The following table shows how the extension bits are modified:

EXTENSION BITS				ADDITION/SHIFTS				SUBTRACTION			
P	5	6	7	P	5	6	7	P	5	6	7
1	0	0	0	0	0	0	1	0	1	1	1
0	0	0	1	0	0	1	0	1	0	0	0
0	0	1	0	1	0	1	1	0	0	0	1
1	0	1	1	0	1	0	0	0	0	1	0
0	1	0	0	1	1	0	1	1	1	1	1
1	1	0	1	1	1	1	0	0	1	0	0
1	1	1	0	0	1	1	1	1	1	0	1
0	1	1	1	0	1	0	0	1	1	1	0

The logic of the extended carry is in two-wire form consistent with the main ALU logic. A separate two-wire check light for EX is displayed on the console and the

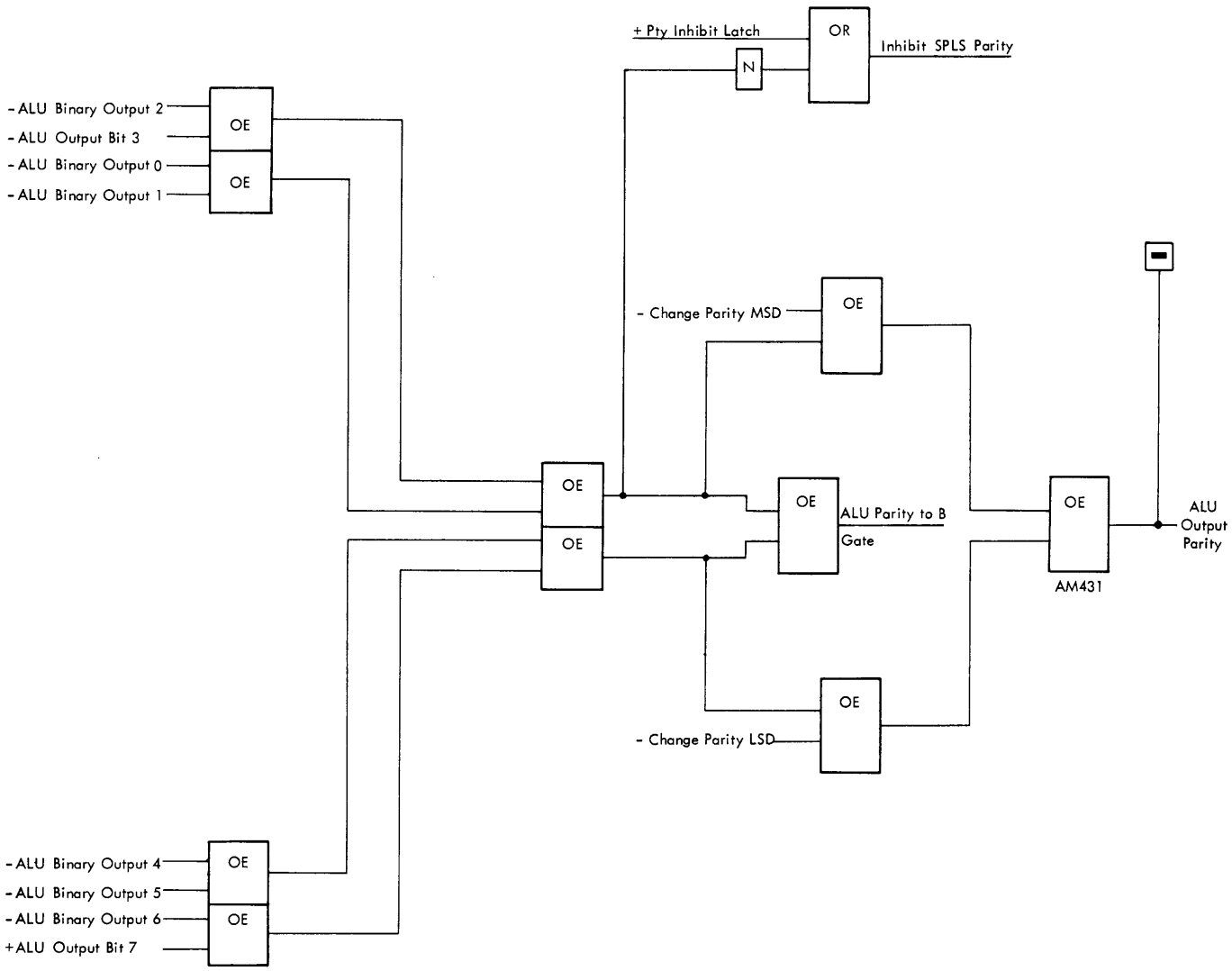


Figure 32. ALU Parity Generation

individual bit position in error can be located from the internal CE panel. The late check latch is also set.

Incoming parity of the extension bits is checked by XOR logic. If it is incorrect, a REG EX parity check occurs. This is displayed on the main console and also causes an early check. The extension output bits have new parity generated by examining bits 5, 6, and 7 of this output. No check of correct outgoing parity is made; parity is checked at destination.

Compress ALU Output

- **Produces compress bits 5 or 6, 7 and parity from full byte**
- **Compress bit 5 or 6 formed from OR of bits 0-5, or bits 0-6**
- **ALU bit 7 goes directly to AX or CX registers**
- **If applicable, ALU bit 6 goes directly to AX or CX registers**
- **Correct compress parity bit generated**
- **Output is available every ALU cycle**
- **Gate for the output is at registers AX or CX**

In all ALU operations where the ALU output is gated to AX or CX, the eight-bit result is compressed into three or two bits by OR'ing ALU output bits 0-5 or 0-6 into extension bit 5 or 6. This function ensures that any result bit of a 1 in positions 0-5 or 0-6 (which indicates a main-storage address outside the available storage size) is represented in the extension and is available for invalid address testing. The interpretation of extension bits varies with the storage size available.

Checking

The following is a summary of the checks in the ALU. All the checks are described under their separate section headings.

ALU Input Checks

- **P register parity check**
- **Q register parity check**
- **EX register parity check**
- **All give an early machine check**
- **All indicated on main console**

ALU Internal Checks

- **Function check of control signals**
- **Two-wire check for the following:**
  1. **Output carry (from signal carry condition)**
  2. **Input carry (from output of YC stats)**
  3. **All bit positions of ALU**
- **Skew selector register parity**
- **All give a late machine check**
- **Skew, function, and two-wire checks indicated on main console**
- **Two-wire check further identified from internal CE console**

ALU Output Checks

- **No output checks; parity generated but not checked**
- **Check would be at output destination**

ALU Timing

The principal timings of the ALU are shown in Figure 27.

Glossary for ALU

Use this glossary in conjunction with Figure 25.

Carry Box: inspects the connect function and the Q bus data and generates the inter-bit carrys.

Carry Stats: store the carrys between bytes or carrys introduced for an operation.

Control Register: four-bit register; the output is decoded to raise control signals for the specific operation.

Connect Box: the first half of the sum generator; input is P and Q data, and the output is connect function.

C8: carry into position 7 (low-order) of byte.

Decimal Filler: logic in the P data path that modifies the input during decimal add operations.

Function Register: set by microprogram; feeds control register to determine the ALU operation.

LSD: least significant digit (refers to decimal digit).

MSD: most significant digit (refers to decimal digit).

MSD-LSD Correct: logic that modifies the ALU output as required following a decimal operation.

Right Shift Unit: logic used basically to right shift the Q data before it enters the carry box. The output is given the letter G.

Skew Buffer: latches that contain the skewed or shifted most-significant four bits from the Q register.

Skew Select Register: the next stage of logic after the Q register and skew buffer; will contain Q bus data or shifted data.

Transformer Read Only Storage (TROS)

- **The TROS control word contains 56 bits, grouped into 18 fields; CA to CT**
- **TROS contains fixed, predetermined information**
- **TROS can only be read out**
- **TROS can be addressed by three registers: ROAR, ROSCAR 1 or ROSCAR 2**
- **TROS uses a 12-bit address for the 4K model TROS and a 13-bit address for the 6K or 8K models of TROS**

TROS contains fixed, predetermined information that can only be read out. The information is stored in sense latch amplifiers and is used to control data flow, control special circuitry, determine the next TROS address, and parity check the TROS word itself. The only method of changing the TROS information is to change physically the TROS control tapes. Depending on machine features, the capacity of TROS can vary to a maximum of 8,192 words.

ROAR or ROSCAR, containing 12 bits, can address a maximum of 4,096 words. The 13th bit is used on systems having the 1401 or 1410 compatibility feature installed.

Principles of Operation

- **TROS uses current transformer principle**
- **Drive lines link with transformer in positions where "one" bit output is required**
- **Drive line bypasses transformer in position where "zero" bit output is required**
- **Output of TROS controls machine functions**
- **Output word is called a micro-instruction or TROS control word**
- **Chain of micro-instructions is called a microprogram**

Figure 33 shows the principle of TROS.

The primary of the transformer is an addressed drive line. The secondary of the transformer forms the sense winding. When a drive line links with a transformer core a current pulse in this drive line induces a current pulse in the secondary winding. If the same drive line bypasses a transformer, no current pulse is induced in that particular sense winding.

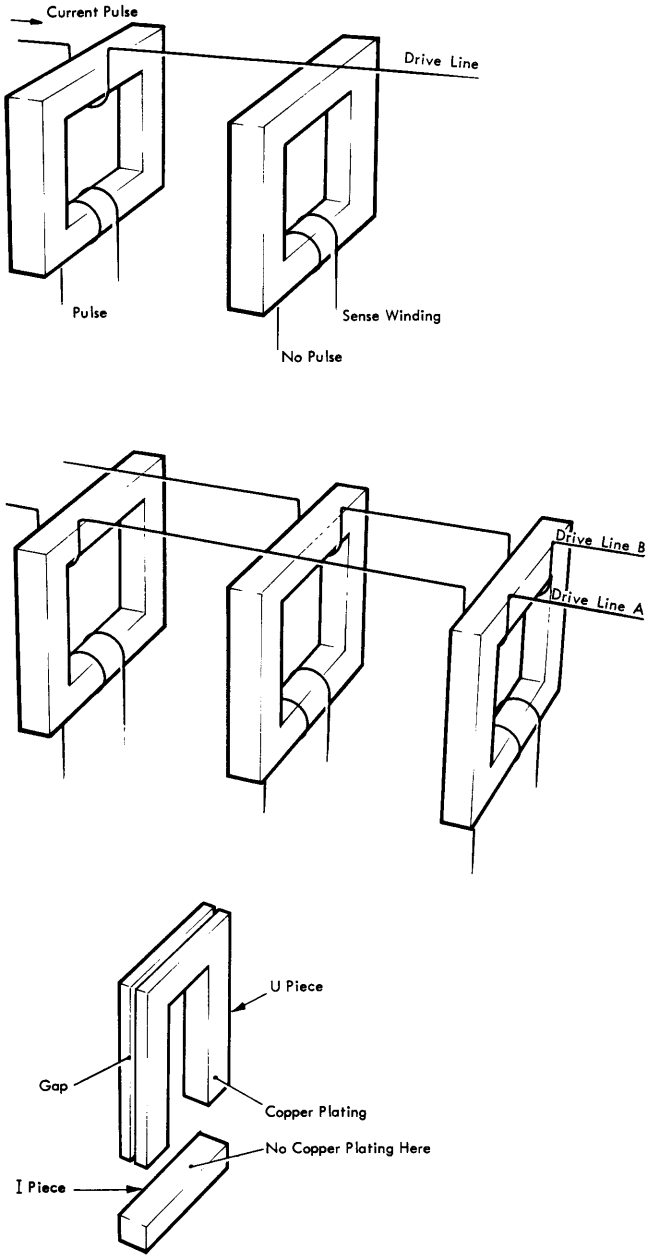


Figure 33. Principle of TROS

A pulse in the sense winding represents a "one" bit, no sense-winding output represents a "zero" bit. Additional drive lines could be used in a similar manner. In Figure 33, a current pulse in drive line A will give an output 101; a pulse in drive line B will give an output of 011.

Each transformer can give an output of a "one" or "zero" bit. Two drive lines for each TROS tape allow two different bit configurations per TROS tape.

The contents of any TROS word can be read out and latched in a register. This latched information can be decoded and used to control machine functions. Part of this output, which may be modified by machine condition bits, determines the address of the next TROS word.

TROS words are addressed in a particular sequence. Sequencing of TROS addresses is called a Microprogram. To perform any operation in the machine the various parts of the CPU (main storage, arithmetic and logic unit, registers and stats, etc.) are controlled by microprogram to perform certain functions in a specified sequence. Thus an operation is defined by a sequence of TROS words. This sequence is the microprogram for the operation.

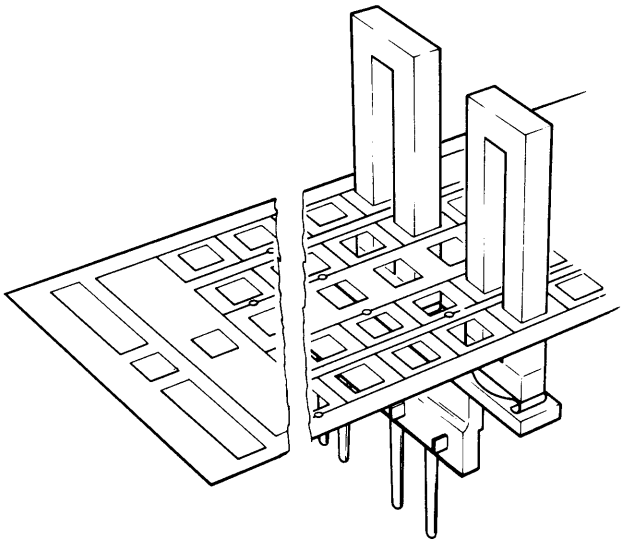


Figure 34. Tape with "U" and "I" Core

Physical Description

- **TROS can contain a maximum of 32 modular units of 256 words each**

TROS is built of modular units, each containing 256 addressable words, each word having a length of 56 bits. Depending upon feature requirements, up to 32 modules are available.

The Transformer (Figure 34)

- **TROS transformer consists of a U piece and an I piece**
- **Sense winding on I piece**

The core of a TROS transformer consists of two parts, a U piece, and an I piece. Both pieces are made of soft ferrite having a low retentivity.

To reduce flux leakage, the U and I pieces are first coated with an insulating material and are then copper plated. A sense winding of 35 turns is wound on the I piece, and the U cores are gapped around their outside face to prevent the plating from acting as a short-circuited turn.

Tapes (Figure 35)

- **Two drive lines on one flexible plastic tape**

TROS drive lines are etched in copper on flexible plastic tapes. On each tape, two drive lines are printed, both in the form of a ladder network. Holes are punched between the rungs of the ladder so that U cores can be inserted through the tapes to mate with the I cores.

Each leg of the U core, when inserted in the tape, is encircled by the sides of the ladder network and two of its rungs. By physically interrupting a side of the ladder, the conductor may either bypass or link with the U core.

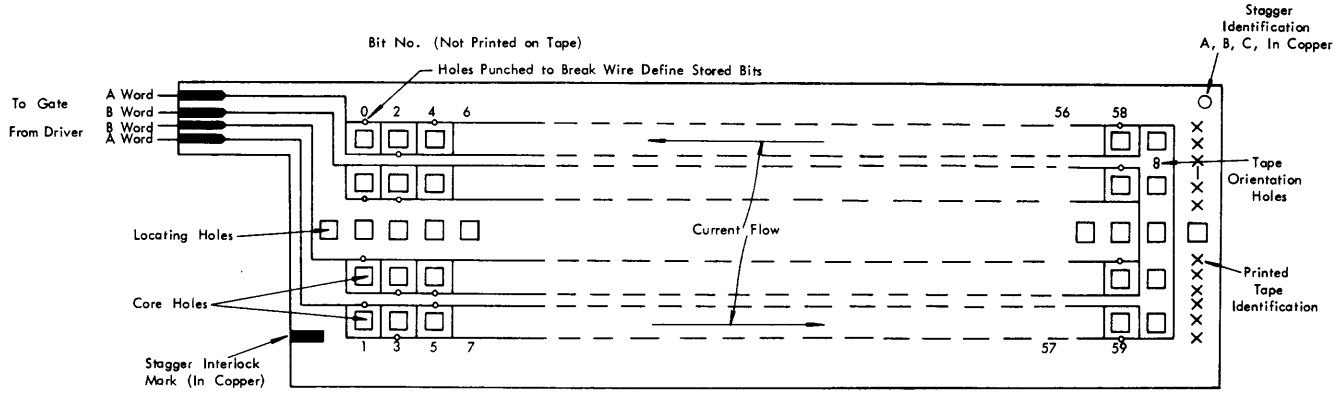


Figure 35. Tape Layout

The two drive lines terminate at the tab end of the tape, each line representing a word. The outer conductor loop is called an A word, the inner loop is the B word. Bit positions along the tape are numbered 0-59, bits 0 and 1 being at the tab end.

In Figure 35, the A word is punched to produce a logical 0 for bit position one and a logical 1 for bit position three on the sense line.

Tape Stagger

To reduce the capacitance between the drive wires on adjacent tapes three different types of tapes are used. On each type the conductor pattern is displaced from the holes by a different amount. These three types are labelled A, B and C and are arranged sequentially throughout the deck of tapes. This is called tape stagger (Figure 36).

Tape Numbering

The upper 64 tapes in the module are inverted with respect to the lower 64 tapes. This creates more space for the connection of the tape ends to the module end boards. The bottom of the module is defined as the side nearer to the I cores. The tapes in the lower half of the tape deck are numbered 0-63 from the bottom up; the tapes in the upper half of the tape deck are numbered 64-127 from the top down.

Resistance Tape

To dampen resonance caused by inter-tape capacity and flux leakage, a distributed loss is introduced. This is achieved by including in each module a plain plastic resistance tape on which single turns of resistance foil are etched. These insulated resistance loops encircle each limb of each U core, and have a resistance of about 0.6 ohms.

Tapes in the tape deck are divided equally into two sections, with end terminations passing on either side of the chassis. Since all tapes are similar in their basic construction, the bottom set of tapes is completely reversed with respect to the top set.

To prevent the wiring on tapes 63 and 127 from touching, the resistance tape is located between these two tapes.

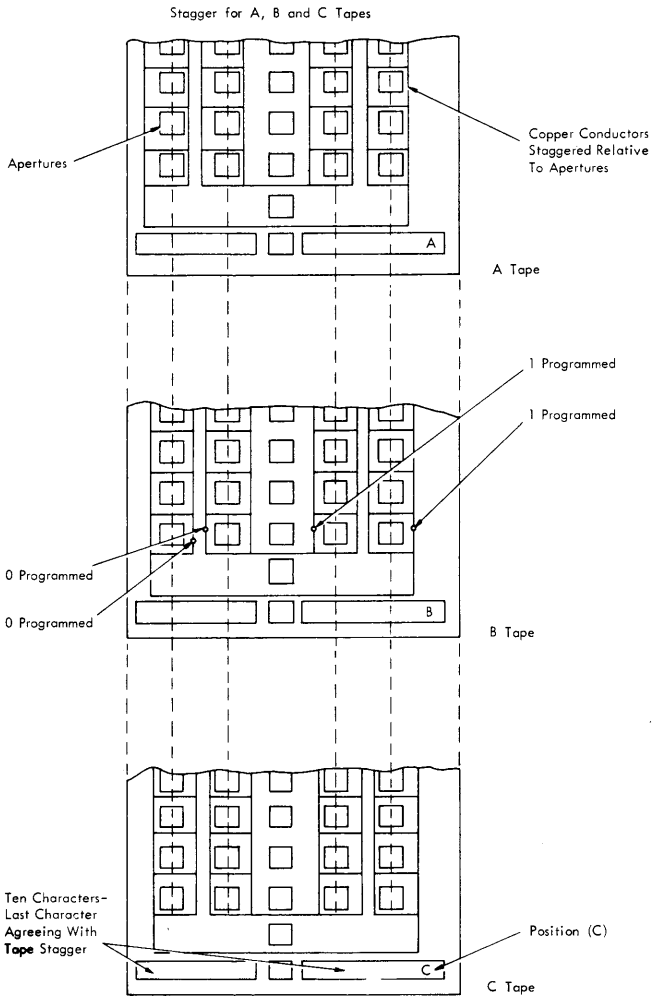


Figure 36. Tape Stagger

The Module

- **Module has 128 plastic tapes**
- **Two words per tape give 256 words per module**

Figure 37 shows an exploded view of a TROS module.

The tape deck (18) consists of 128 plastic tapes carrying the 256 words. Holes are punched between the rungs of the ladder network to accept the U cores (21) which pass through the tapes to mate with the I cores (4).

The I cores (4) are held in a core-carrier assembly consisting of parts (3), (5) and (6) and are wound with a 35-turn sense winding connected to the pins on part (6).

Tapes are lifted on and off the module by tray (19) and are located by means of aligning pins (20), screwed into blocks (2). The blocks also carry the two rods (1) supporting the core-carrier assemblies. There are thirty core carriers and one dummy core carrier.

The dummy core carrier (24) is next to core carrier (3) and is used only to support the tape deck. The support (8) and chassis (7) screw into blocks (2) and are spaced by the rails (9).

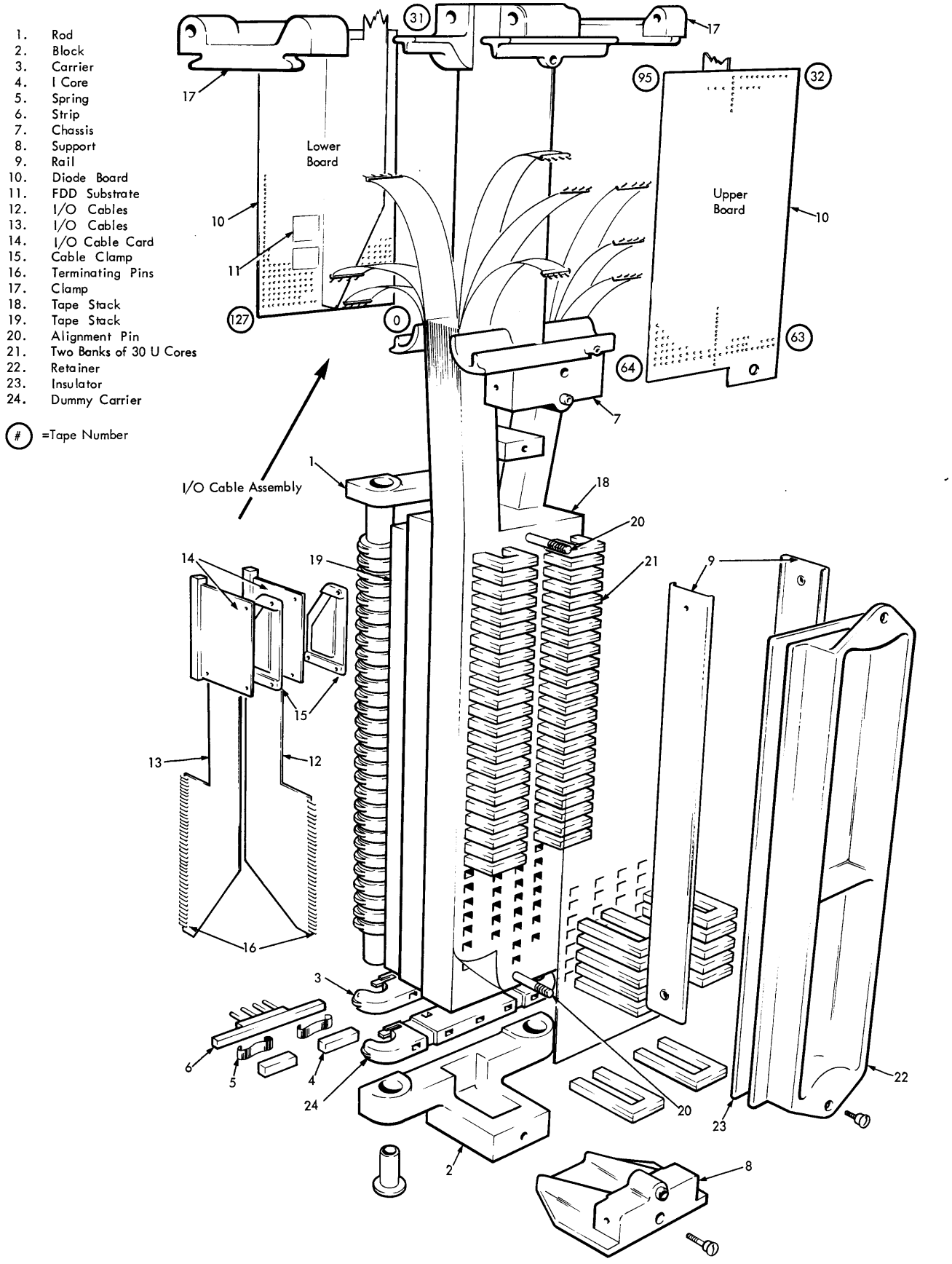


Figure 37. TROS Module (Exploded View)

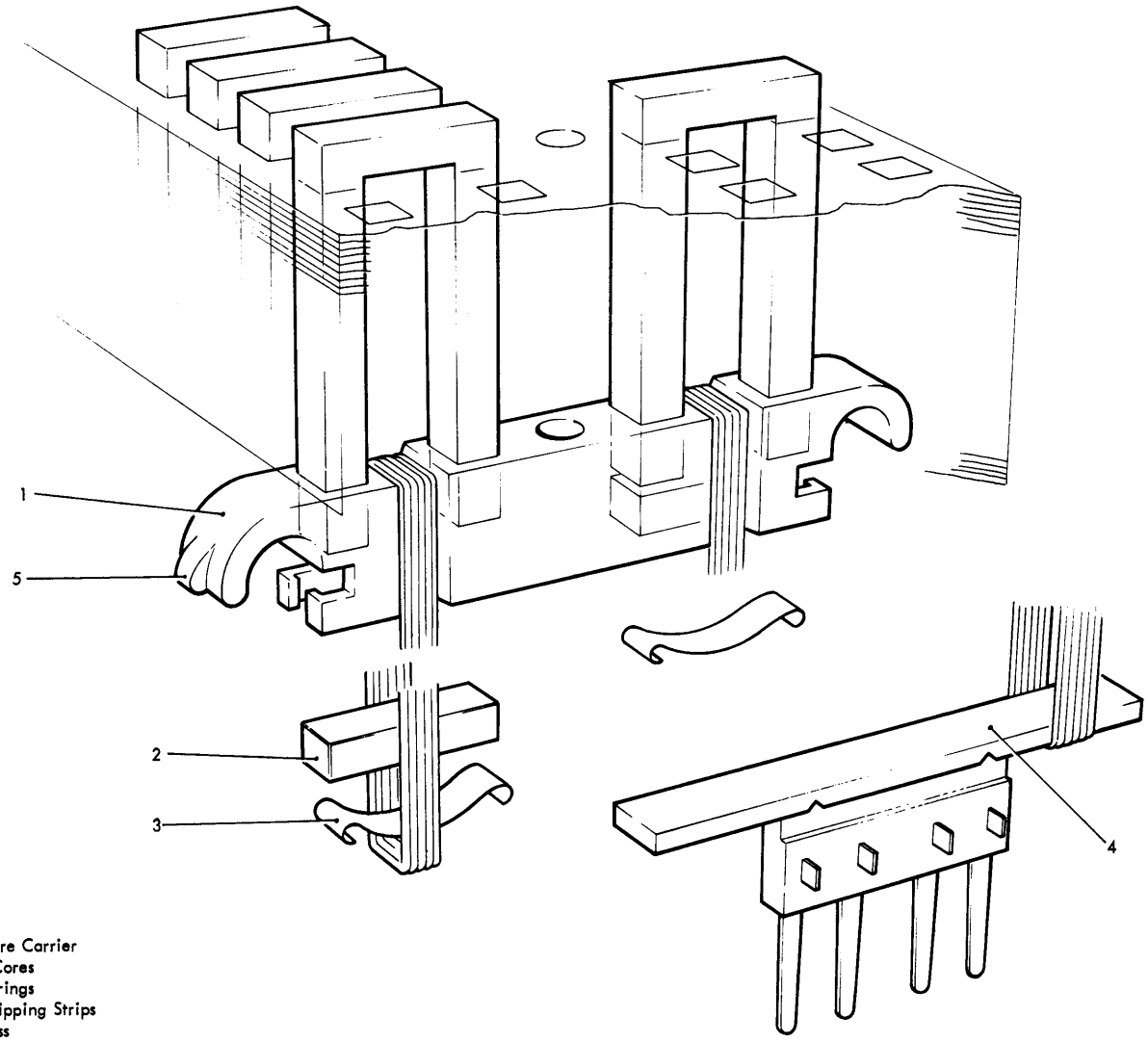
Connected to the chassis are the module end boards (10) to carry the diodes (11) used for TROS word addressing and connections to the tapes in the tape deck. Connections to the tape consist of pins placed in plated-through holes in the board on which the four tape-terminal connections are pressed.

Input/output connections to the module end boards are made by tapes (12) and (13). These tapes have pins (16) similar to those on the tapes in the tape deck, passing through plated holes in the boards (10) and soldered to the printed circuitry. The tapes are clamped to the chassis (7) by the clamps (17) so as to relieve any strain in the connections to boards (10). The flexible ends of the tapes are connected to cards (14) and are held with strain-relief clamps (15). U cores (21) are held in the module by the retainer (22) and insulator (23). The retainer screws into support (8) and chassis (7), forcing the U cores against the I cores (4) and compressing the spring (5).

Core-Carrier Assembly

- **Core carrier contains two I pieces**
- **Sense windings are wound around the core carrier with the I core inside**

The core-carrier assembly (Figure 38) consists of a core carrier (1) into which the I cores (2) are inserted. Springs (3) are placed behind the I cores to ensure proper contact with the U cores. Springs and I cores are held in position by clipping strips (4) onto the core carrier. Sense windings are wound around the core carrier, encircling the I pieces. Ends of the sense windings are connected to the pins on the strip. A boss (5) on one end of the core carrier enables correct visual orientation when placing the core carrier onto the rods. If assembled incorrectly, it would be impossible to connect the pins on the strip to the laminar buses.



1. Core Carrier
2. I Cores
3. Springs
4. Clipping Strips
5. Boss

Figure 38. Core-carrier Assembly

Laminar Bus

- **Laminar bus connects corresponding sense windings of each module to one sense amplifier**

When a number of modules have been assembled, the sense windings associated with each particular bit of a module are connected in parallel. For each bit, 16 sense windings (one in each module in a 16-module TROS) are connected in parallel by means of a laminar bus.

Each bus bar consists of four conductors printed on a strip of non-conducting material, with two conductors on each side of the strip. Each conductor has pins connected to it, coinciding with the pins of the core carrier. Pins of the core carriers are welded to those of the laminar bus bars.

The Array

- **Circuits are carried on two large cards**
- **Circuits are connected to array via four module-commoning boards**

The general layout of a 4K TROS is shown in Figure 39. Decoders, drivers, timing cards, sense amplifiers, and sense latches are on the two large cards to the left of the array. The connection between the modules and the circuit large card that carries the decoders, drivers, and timing cards, is via four module commoning boards.

All necessary module interconnection wiring for drivers and gates is on this board. I/O cards from each module are plugged into the module commoning boards that carry the gate circuit cards and the gate strobe cards. Each gate circuit carries four gate circuits. One gate circuit is required for each module.

Functional Description

- **Timing obtained by tapping CPU P clock pulse in delay line**
- **Constant-current drive to TROS switched by selected driver**
- **Sense amplifier has input threshold controlled by sense restore voltages**
- **Sense strobe times sense amplifiers**
- **Sense latch reset by sense reset circuit**

Figure 40 is a block diagram of the logic circuits associated with a 4K TROS. The timing of the circuitry is obtained by feeding a pulse from the CPU P clock into a delay line and tapping the delay line at different places. Timing for the gates and drivers is provided by the gate strobe and driver strobe circuits respectively.

The driver supply circuit produces the constant current drive for the TROS. This is the current to be switched by the selected driver.

The sense amplifier is designed so that before a sense pulse appears at the input, the input of the threshold stage is restored to a certain level by applying a constant sense restore voltage for a certain time. The restore circuit provides this restore voltage.

The sense strobe circuit produces the timing pulse for the sense amplifiers; the sense reset circuit produces a reset pulse for the sense latch.

TROS Timing

- **TROS can cycle continuously at a speed of 625 nanoseconds per cycle**
- **Input threshold at sense amplifiers must be restored before inputs are accepted**
- **No current in array until driver is selected**
- **Non-selected drivers isolated by reverse-biased diodes**
- **1 outputs last longer than 0 outputs**

The TROS can continually cycle at a speed of 625 nanoseconds. Figure 41 shows the internal and external timing of TROS.

TROS Addressing

- **12-bit address register (ROAR or ROSCAR) for 4K TROS**
- **ROAR or ROSCAR is gated to ROSAB (read only storage address bus)**
- **64 drivers and 64 gates to select one word in 4K TROS**
- **Diode for each word line at driver end to prevent back circuits**
- **ROSAB bits 11-6 address 64 gates; ROSAB bits 5-0 address 64 drivers**
- **Four gates for each of the 16 modules**
- **64 drivers common to all 16 modules**

Figure 42 shows the driving and gating principle used in addressing TROS. Addressing of TROS is via the read only storage address bus (ROSAB), which is fed from the ROAR or ROSCAR register.

When the CPU is in control, ROAR is gated to ROSAB. When selector channel 1 takes over control, ROSCAR 1 is gated onto ROSAB; if selector channel 2 is using the data flow, ROSCAR 2 is gated to ROSAB.

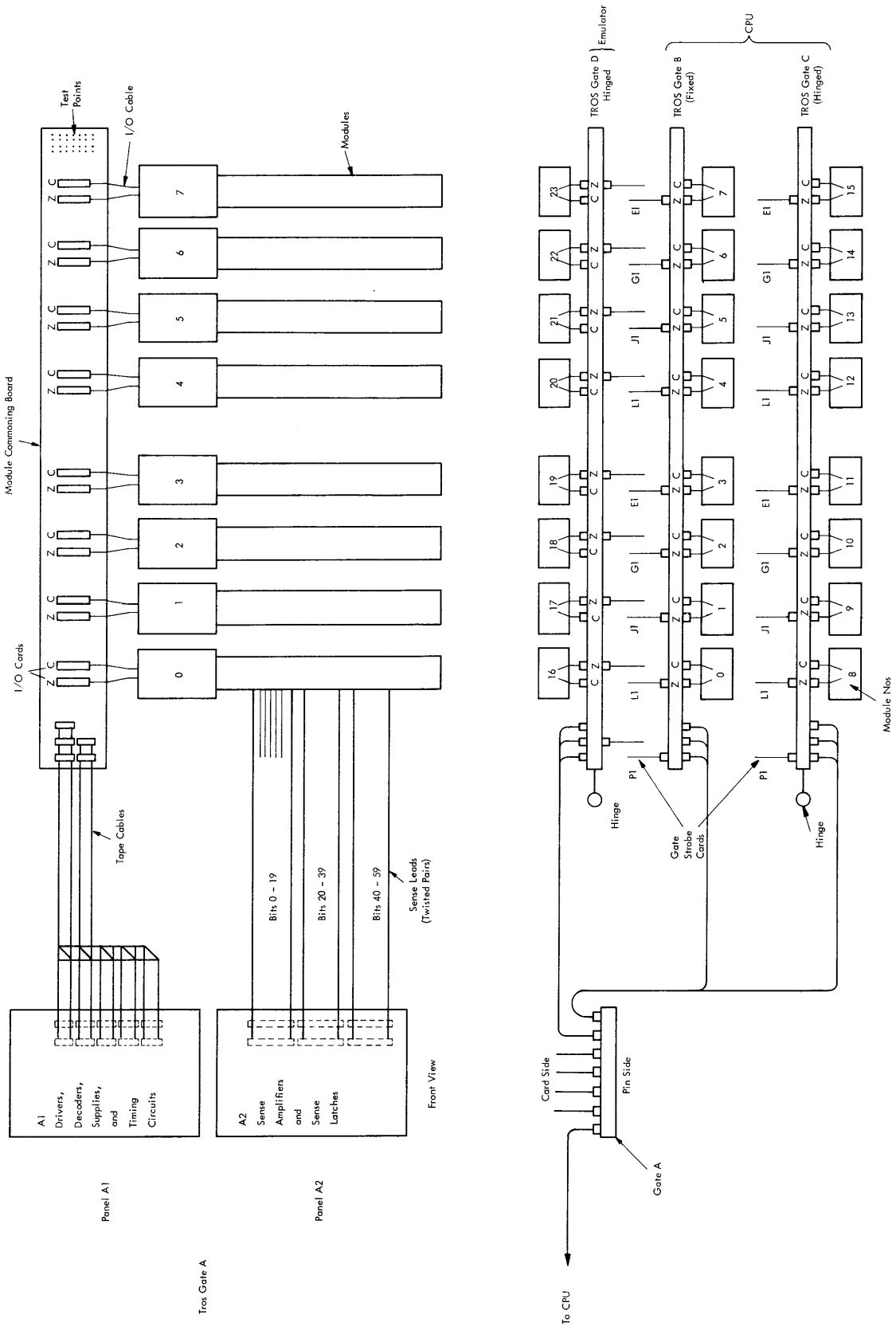


Figure 39. General Arrangement of TROS

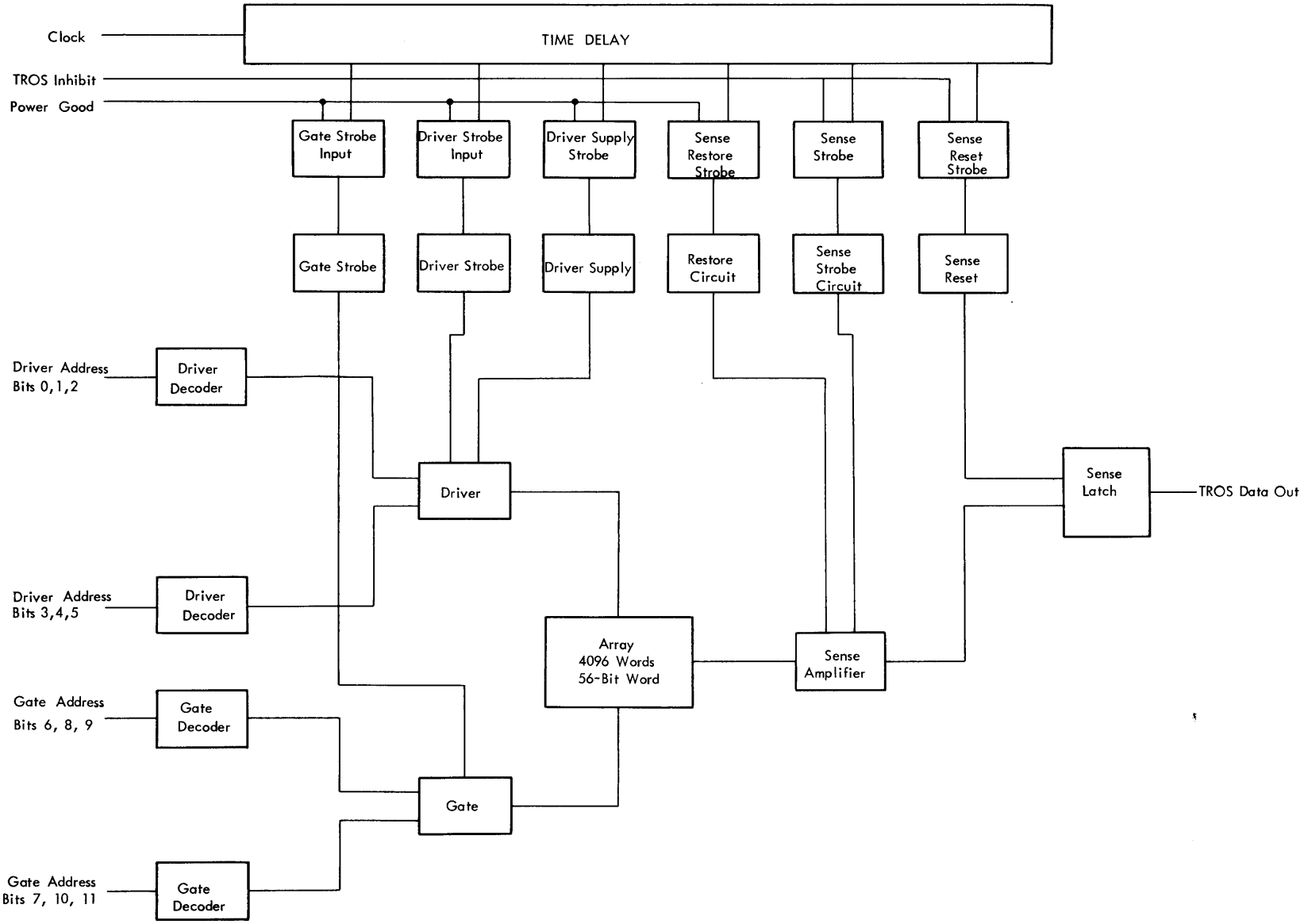
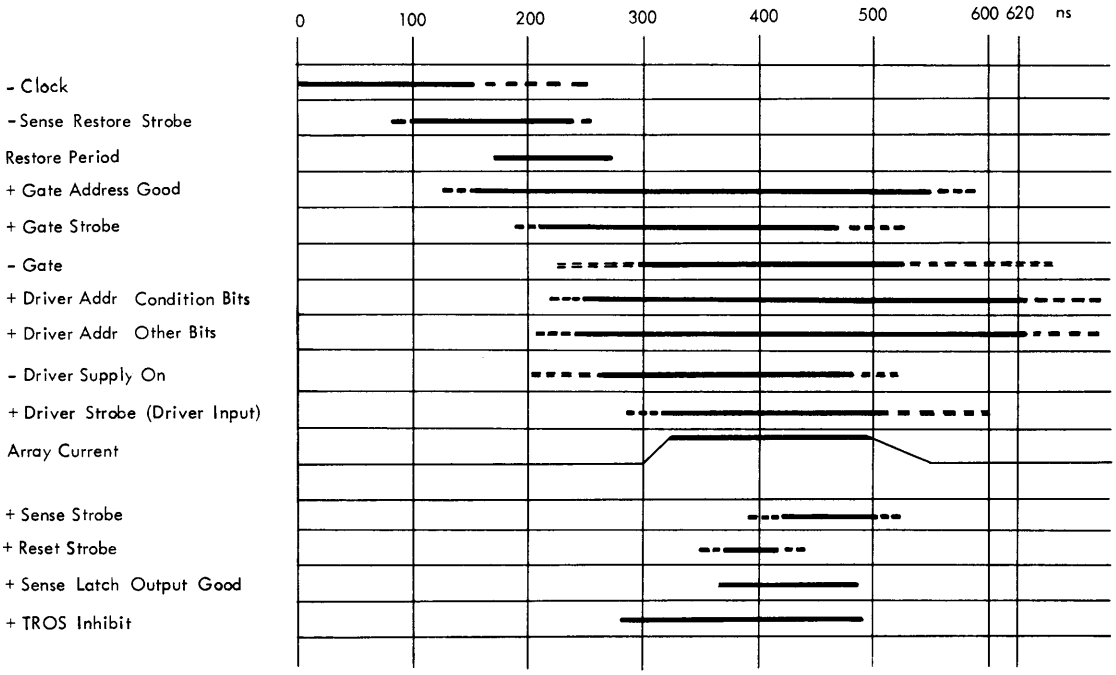


Figure 40. 4K TROS Block Diagram



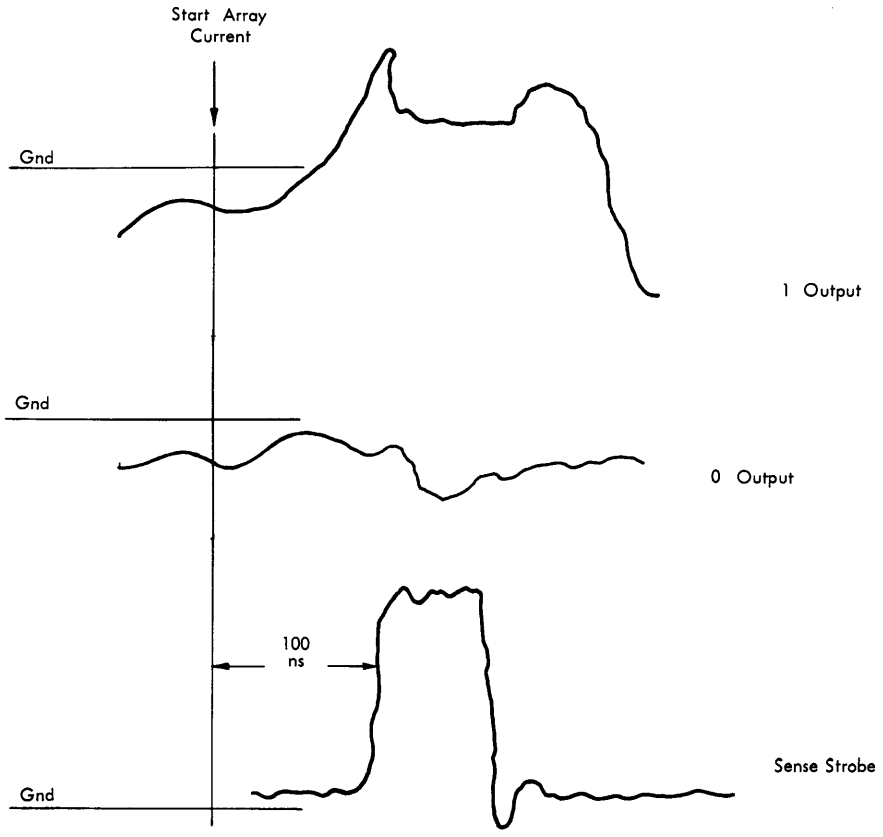
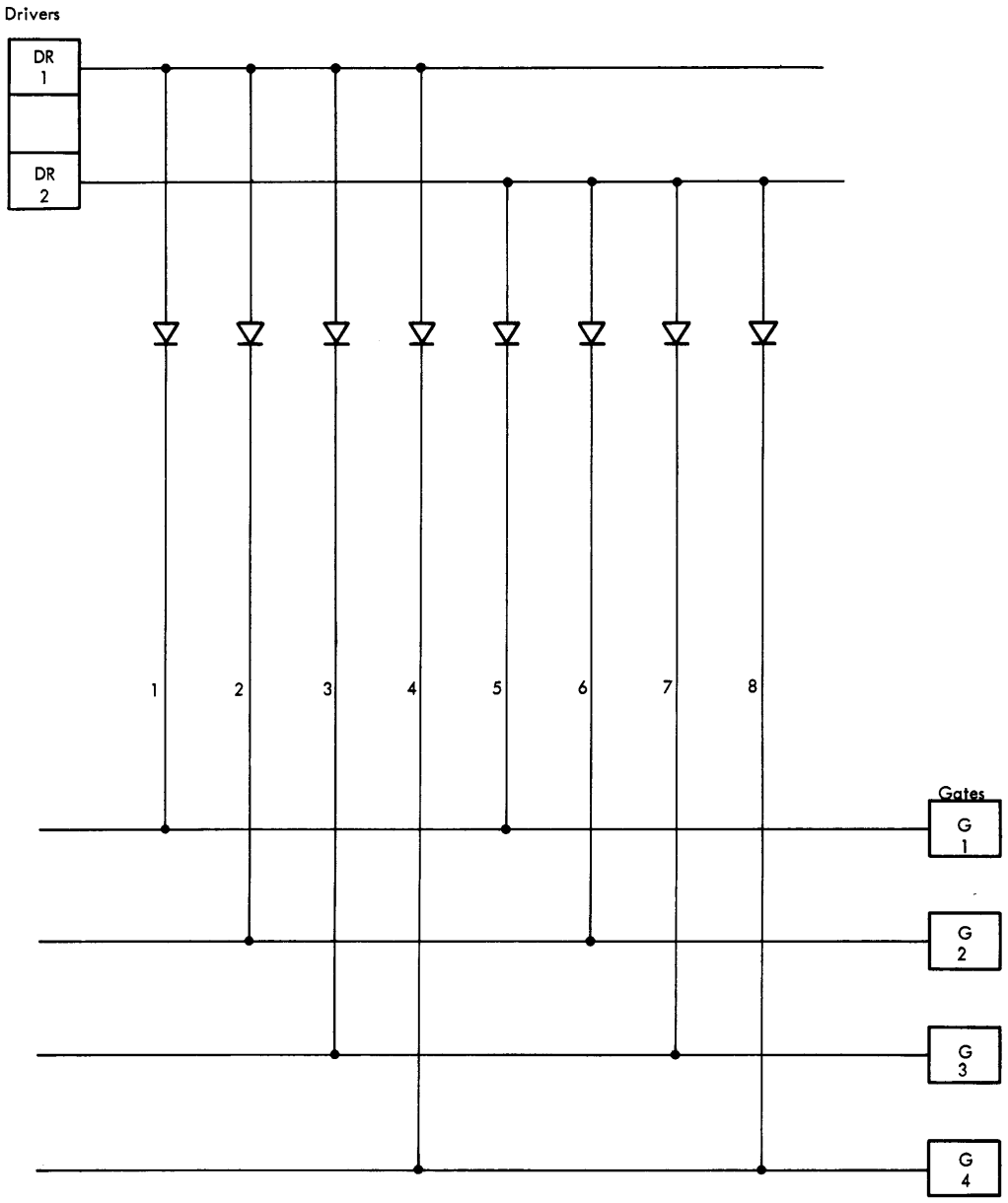


Figure 41. TROS Timing and Sense Current Waveforms



Any one of the eight word lines shown is selected by using two drivers and four gates, and selecting one of the two drivers and one of the four gates. For example, by activating driver 2 and gate 1, word line (TROS word) 5 is selected

Figure 42. Principle of Driving and Gating

To address any word line, a drive circuit is needed at one end of the line, and a gate circuit at the other end. A diode in series with each word line prevents back circuits.

To select any one of the 4,096 drive lines of a 4K TROS, 64 drivers, 64 gates and associated diodes are needed. The matrix of drivers and gates is controlled by the decoded output of the 12-bit read only storage address bus (ROSAB).

Figure 43 shows the decoding of ROAR. The control of the 64 gates is by bits 11 to 6 of ROSAB. The 64 drivers are controlled by bits 5 to 0 of ROSAB. There are four gates to each module, but the drivers are common to all modules. All storages have 64 drivers, but the number of gates is dependent upon the number of modules.

TROS Drive Scheme

- **128 tapes in module terminated on two module end boards**
- **Four word lines in each module commoned to one driver**
- **64 word lines in each module commoned to one gate**

The 128 tapes in a module are terminated on two module end boards or diode boards on which are also mounted the matrix diodes in the form of FDD (four double diodes) substrate blocks. The 256 words in the module are addressed by 64 drivers and 4 gates.

Drive and gate commoning of lines on the module end boards is described.

Drive Commoning

- **Four word lines per driver**

Because only 64 drivers are used for 256 word lines, four word lines from each module are commoned to one driver. The four word lines are the A and B lines on any particular tape together with the A and B lines on the corresponding tape in the other (upper or lower) half of the module. Leads to the drivers are numbered 0-63, starting from the bottom of the module. For example, A and B words on tape 95, together with A and B words on tape 32, are common to driver 32 via four diodes.

Gate Commoning

- **64 word lines per gate**

A module has 256 word lines and only four gates, with 64 word lines commoned to each gate. In the upper half of the module, all A lines are taken to one gate, all B lines to another gate. The lower half of the module is similar.

Leads to the gates are numbered 0-63. Gates 0-3 are connected to module 0. Gates 4-7 are connected to module 1. Gates 60-63 are connected to module 15. For example, B words of tapes 64-127 are commoned to gate 3; A words of tapes 64-127 are commoned to gate 2. In the lower deck all A words are commoned to gate 0; all B words to gate 1.

Diodes in series with the word lines of the driver end are in FDD (four double diodes) substrate blocks in the module end boards. Each board carries 16 substrate blocks each containing eight diodes.

Decoding of ROSAB (See Figure 43)

- **Bits 11-6 of ROSAB form gate address**
- **Bits 5-0 of ROSAB form driver address**

Gate Address Decoding

- **Bits 11, 10, and 7 decoded to produce eight decoder output lines numbered 00-70**
- **Bits 9, 8, and 6 decoded to produce eight decoder output lines numbered 0-7**
- **Bits 11-8 determine module number**
- **Bit 7 determines upper or lower half of module**
- **Bit 6 determines A or B word on tape**

Bits 11, 10, and 7 of ROSAB are fed into a decoder where each bit combination is decoded to activate one decoder output line. With three bits, eight bit combinattions are possible; therefore, eight decoder output lines, numbered 00-70, are provided.

Bits 9, 8, and 6 of ROSAB are fed into another decoder, the output lines of which are numbered 0-7. The output lines of both decoders are fed to the 64 gates that have the last decoding stage at their inputs. The four gates used in every module are shown in Figure 43. The result of this system of decoding is as follows:

ROSAB bits 11-8 determine the module number.

ROSAB bit 7 determines the position of the tape in the module: Bit 7 = 0 means lower half of module; bit 7 = 1 means upper half of module.

ROSAB bit 6 determines the position of the word on the tape (A or B): Bit 6 = 0 means A word; bit 6 = 1 means B word.

Driver Address Decoding

- **Bits 5, 4, and 3 decoded to produce eight output lines numbered 00-70**
- **Bits 2, 1, and 0 decoded to produce eight output lines numbered 0-7**
- **Driver address determines tape number in upper or lower half of module**

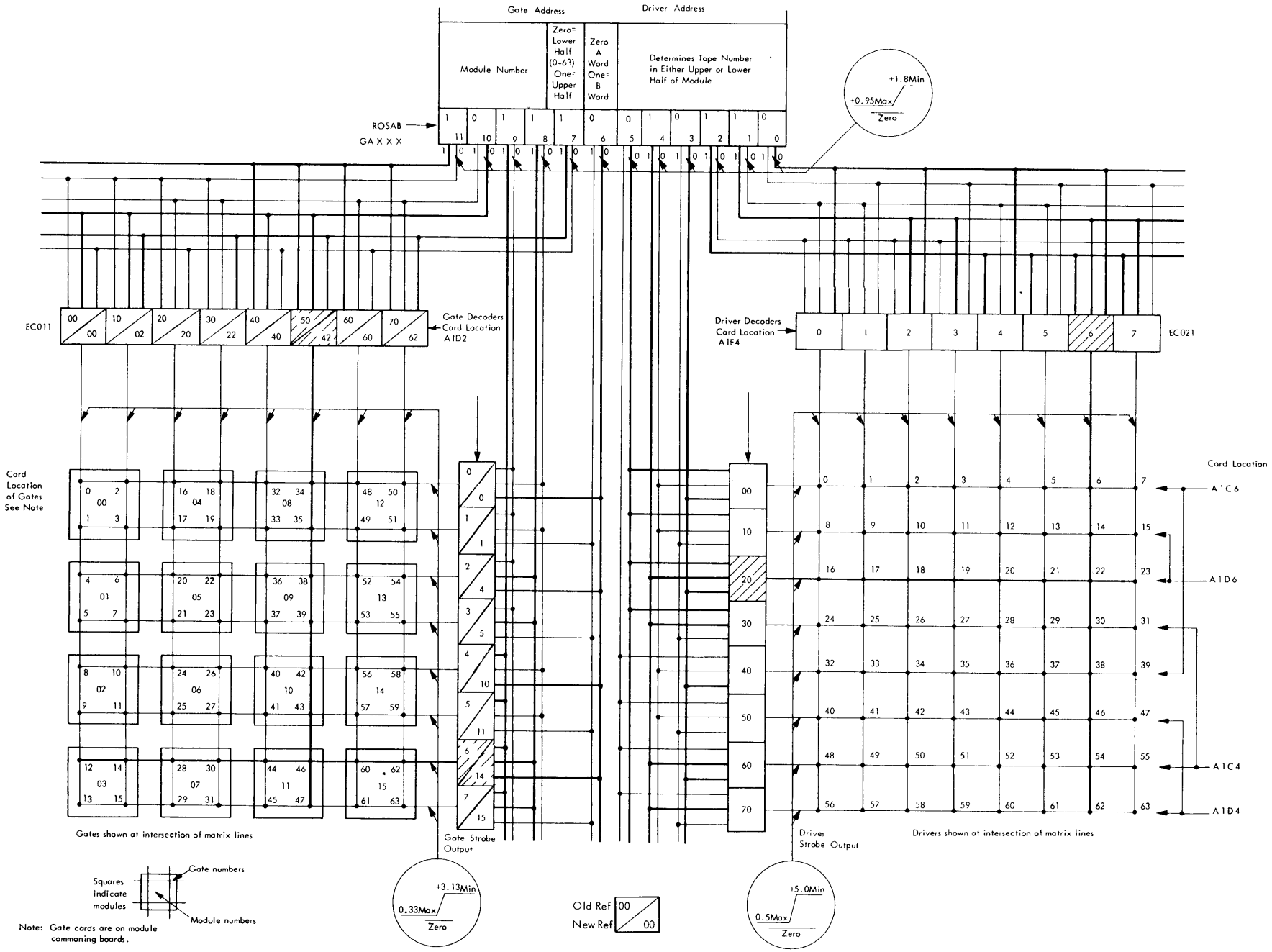


Figure 43. Decoding of ROAR

Bits 5, 4, and 3 of ROSAB are fed into a decoder where each bit combination is decoded to activate one decoder output line. With three bits, eight bit combinations are possible; therefore, eight decoder output lines, numbered 00-70, are provided.

Bits 2, 1, and 0 of ROSAB are fed into another decoder. Output lines of this decoder are numbered 0-7. Output lines of both decoders are fed to the 64 drivers that have the last decoding stage at their inputs. These 64 drivers are common to all modules. Driver address determines the tape number in either the lower or upper half of the module.

Example of ROSAB Decoding

ROSAB is assumed to have the following bit configuration: 101110010110. The gate address is 101110, and gate 46 is selected. The driver address is 010110, and driver 22 is selected.

To determine the location of the selected word: ROSAB bits 11-8 contain 1011. The reference 1011 means the word is located in module number 11. ROSAB bit 7 contains 1, which means the selected word is in the upper half of this module. ROSAB bit 6 contains 0, therefore the selected word is an A word. ROSAB bits 5-0 (driver address) are 22, which means that the selected word is on the 22nd tape in either the lower or the upper half of the module. It has already been determined that the selected word is in the upper half of the module, so the tape is 105 (i.e., 127 minus 22); therefore, the word selected by ROSAB address 101110010110 is word A on tape number 105 in module number 11.

A simplified method of determining the module, tape number, and tape-stagger class is shown below. Using the same example as before, the gate number was 46, the driver number was 22. Dividing the gate by 4 gives a module number of 11, with a remainder of 2. The remainder value will indicate an upper A tape. The tape number is determined by subtracting 22 from 127, result 105.

Module
4) Gate

Remainder

0	1	2	3
LA	LB	UA	UB

Tape number equals driver number.

Tape number equals 127 minus driver number.

TROS Control Word and Data Flow

To explain the detailed functions of the TROS control word, some data flow functional units must be understood. Figure 44 is a layout of the TROS control word, indicating the basic function of each field. The CJ and CP fields use only three bits for CPU control. Additional bits in positions 54 and 55 are used to expand these fields when the machine is equipped with the selector channel feature. Figure 45 shows the system data flow.

Data Registers

- **In data registers containing more than eight bits, the individual bytes are referred to as A0, B0, etc. for the high-order byte; A1, B1, etc. for the low-order byte; AX, CX, RX, for the bits in the extension**
- **Any one byte can be gated to or from the ALU**
- **All bits are gated to or from R register**
- **Registers are reset when new information is entered, except the P, Q, and R registers which are reset every machine cycle**
- **P, Q, and R registers are also referred to as P, Q, and R bus**

Emit Circuits

- **At various locations of the data flow, fixed data from the microprogram can be introduced (emit field)**
- **The emit field is four bits wide and is the CE field of the TROS control word**

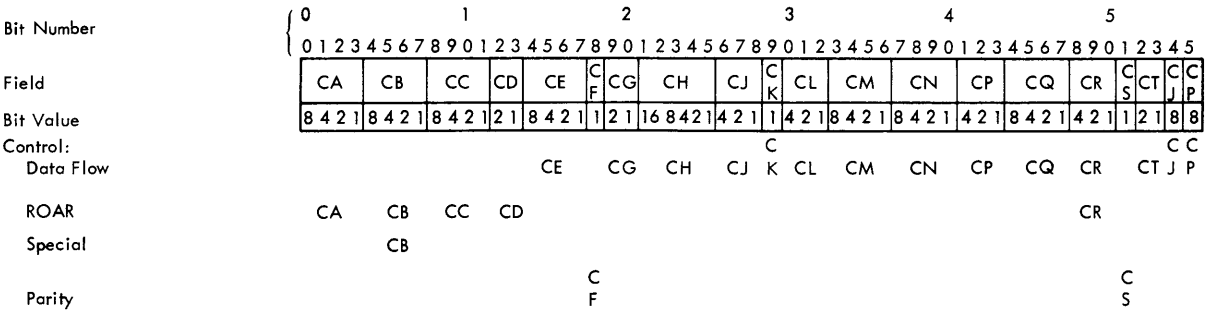


Figure 44. TROS Control Word and Basic Functions

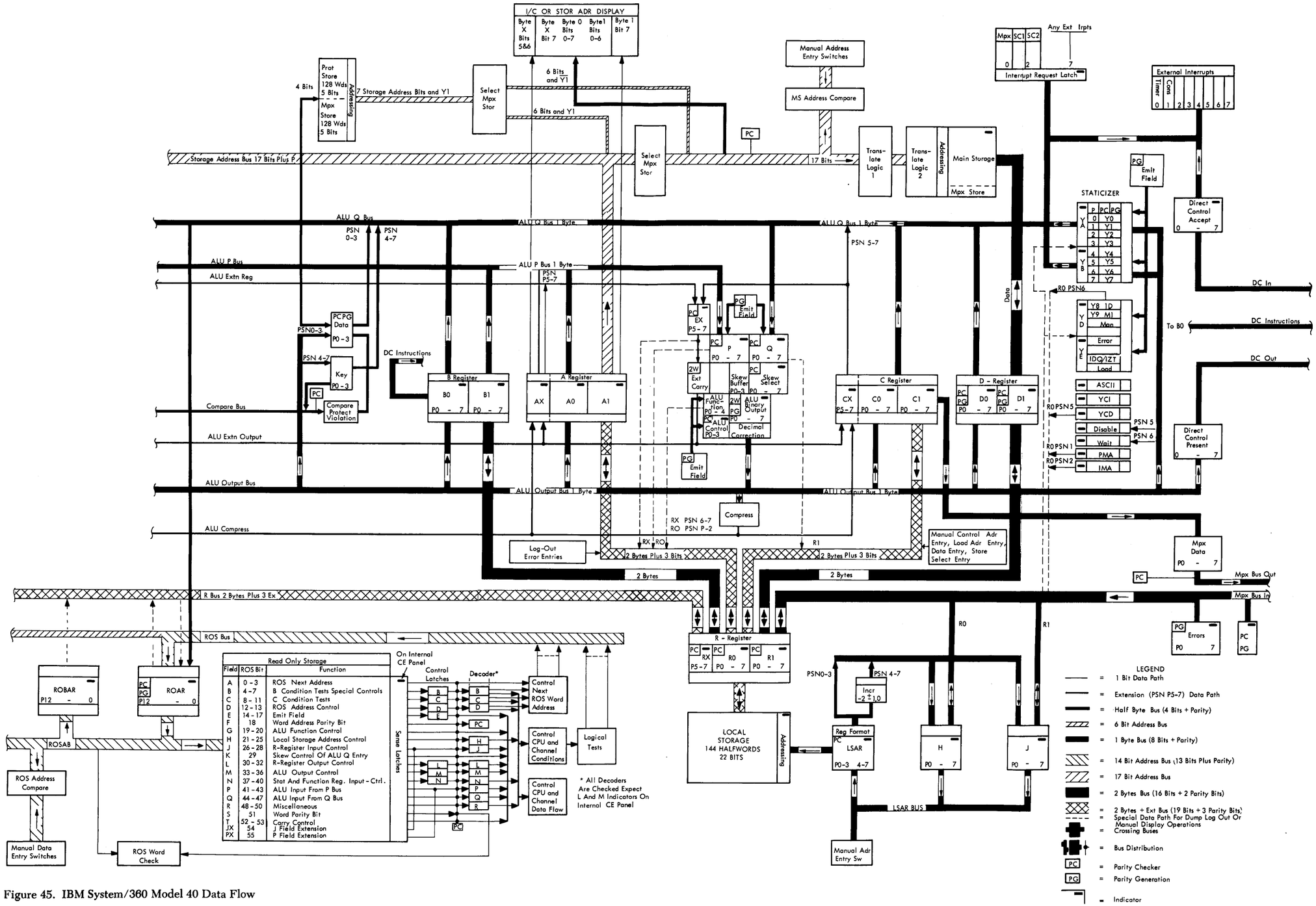


Figure 45. IBM System/360 Model 40 Data Flow

The emit field can be gated to one or more of the following:

1. Bits 0-3 or 4-7 of P
2. Bits 0-3 or 4-7 of Q
3. Bits 0-3 of ALU function register
4. Bits 0-3 or 4-7 of LSAR
5. YA, YB, YD or YE stats
6. Selector channel flag registers

TROS Control Word Parity Bits

- **The TROS control word contains two parity bits, CF and CS**

CF Bit

ROAR and ROSCAR contain a parity bit to make the over-all bit count odd. The CE bit is 1 if the TROS control word is stored at a TROS address where the address parity has to be on.

The CF bit is compared with the address parity bit. If they do not agree, a machine check is indicated. This is a check to ensure that the TROS control word actually addressed by ROAR or ROSCAR is read out of TROS and no TROS address decode malfunction is present.

CS Bit

This is the over-all parity bit of all 56 TROS control word bits. Over-all bit count is always odd. The output of the TROS sense latches is checked for an odd bit count.

Control Latches

- **Some control fields have control latches**

Control fields CB, CD, CE, CL, CM and CN have control latches set from the sense latches at T1 or T1 del time and are reset at T1 time in the following cycle. The purpose is to have the control field output available for a whole cycle (625 nanoseconds) because the sense latch output is only from T1 to T4 time.

Examples for Data Flow Control

Different TROS control words are shown and the functions performed are listed below. (Only data control fields are shown.)

1.

CB	CD	CE	CG	CH	CJ	CK	CL
1111	10	1111	01	10111	100	1	001

CM	CN	CP	CQ	CR	CT
0110	0101	110	0110	010	10

The CD field does not contain 1 or 3, therefore the CB field is not used to control special functions. ALU operations performed:

Direct AND operation (CG=1), YCD is reset (CT=2)
P input=11110000 (CP=6; CE=15)
Q input=D04, input to ALU is skewed (CQ=6; CK=1)
Result is gated to B0 (CM=6)
R bus operation performed:
D is gated to R (CJ=4)
R is stored in local storage (CL=1)
Local storage is addressed with the J register contents
LSAR is incremented by +1 and stored back to J (CH=23)
All YB stats are reset (CN=5; CE=15)
Main storage write is initiated (CR=2)

2.

CB	CD	CE	CG	CH	CJ	CK	CL
0001	00	1111	11	00100	110	1	101

CM	CN	CP	CQ	CR	CT
1010	0000	100	0010	110	10

CD=0 does not allow the CB field to be decoded as special function. ALU operations performed:

Direct ADD operation (CG=3), YCD is reset (CT=2)
P input=B0 (CP=4)
Q input=B1 input to ALU is skewed (CQ=2, CK=1)
Result is gated to C0 (CM=10)
After the operation the skew buffer is reset (CR=6, assume Y10 off)
R bus operation performed:
Local storage output is gated to R (CJ=6)
R is gated to B (CL=5)
Local storage is addressed with 01001111, this address is stored back to J (CH=4, CE=15)

3. Assume the machine is in I/O state, Y2 and Y3 are off (Mpx channel)

CB	CD	CE	CG	CH	CJ	CK	CL
0101	01	1000	01	10100	110	0	101

CM	CN	CP	CQ	CR	CT
0000	0000	0000	0000	0000	00

The interface control check is set (CD=1, CB=10)
ALU function is direct AND (CG=1) carry is not affected (CT=0)
Zeros to P (CP=0)
Zeros to Q, no skew (CQ=0, CK=0)
Output has no destination (CM=0)
R bus operation performed:
Local storage output to R (CJ=6)
R is gated to B (CL=5)
Local storage is addressed with 00101000; this address is incremented by 1 and stored back to J. (CH=20, CE=8, I/O state, Mpx channel operating).

Determining the Next TROS Address

- **Basically the 12 address bits are generated in four different formats, depending on the CB and CD fields**
- **Some address bits can be set according to certain machine conditions or data within the data flow**
- **The CA field is normally a direct part of the next TROS address**
- **Some of the old address bits are normally retained for the next address**
- **Certain TROS addresses are forced by logic circuits**

Basic formats are shown in Figure 46.

FNB Format

FNB (functional branch) is specified when the CB field is equal to 15.

ROS address bit generation for the next cycle is:

1. The CD field is set into bits 11 and 10.
2. The CA field is set into bits 9 to 6.
3. A zero is set into bit 5.
4. Bits 0-3 of the Q register are set into bits 4 to 1 of the address.
5. The condition specified in the CC field is tested: if the condition is present, 1 is entered into bit 0; if the condition is not present a 0 is entered into bit 0.

This format is often referred to as 16-way branch, because the four Q register bits can have any of 16 combinations. In practice, the C condition (CC field) expands the possibilities to 32 but in most cases the CC field specifies a fixed value of 0 or 1.

CD = 0, 1, 3 Format

The TROS address is set accordingly if the CD field contains 0, 1 or 3 and the CB field does not specify FNB.

Address bit generation for the next cycle is:

1. Bits 11-6 are not affected and are retained in the new address.
2. The CA field is set into bits 5-2 of the next address.
3. For CD=0, the condition specified in the field is tested: if present, 1 is set into address bit 1; if not present, 0 is set into address bit 1.
4. For CD=1 or 3, bit 1 is set to 0. The condition specified in the CC field is tested: if present, 1 is set into address bit 0; if not, 0 is set into bit 0.
5. When CD=3 and CB=14, UNDUMP is called; ROAR is restored from local storage.

This format basically offers a four-way branch (B and C condition). The high-order six bits are unchanged. It is the format most frequently used for sequential microinstructions and branches within short routines.

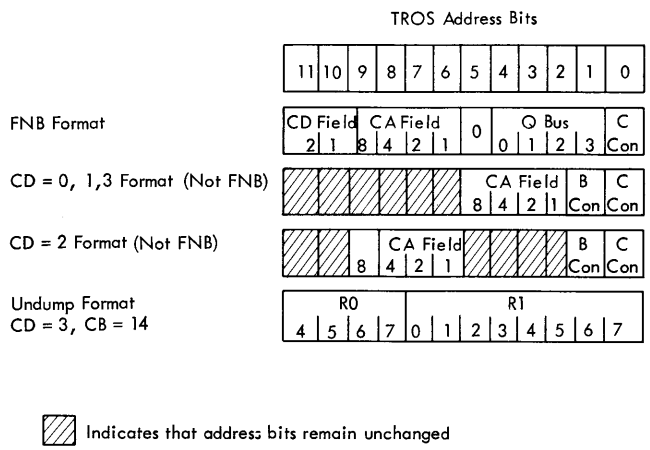


Figure 46. Basic TROS Address Bit Generation

CD = 2 Format

The next TROS address is set accordingly if the CD field contains 2 and the CB does not specify FNB.

Address bit generation for the next cycle is:

1. Bits 11, 10 and 5 to 2 are retained.
2. The CA field is set into bits 9-6 of the next address.
3. The condition specified in the CB field is tested: if present 1 is set into address bit 1; if not present, 0 is set into address bit 1.
4. The condition specified in the CC field is tested; if present, 1 is set into address bit 0; if not, 0 is set into bit 0.

The format is similar to the CD=0 format except for positioning of the CA field bits. The CA bits are set into more-significant TROS address positions; branches within a greater area are thus possible.

Undump Format

In the last undump cycle (hardware cycle) ROAR is set from the R register (ROAR address was dumped into local storage).

R0 bits 4-7 are set into ROAR bits 11-8
R1 bits 0-7 are set into ROAR bits 7-0

Special Formats

With CD=3, the CB field controls special data flow functions. Some of these functions are branch conditions. In these cases, the basic format is as described for CD=0; however, the B or C condition tests for setting address bits 1 and 0 may be replaced with special functions, or bits 1 and 0 may be set by logic circuits.
Figure 47 shows how the next TROS address is determined.

Forced. Addresses

Certain machine conditions force the entire address by logic circuits:

Console Operations	TROS ADDRESS FORCED (HEX)
MS Validate (Diagnostic Control Switch)	002
System Reset, Load, Power on	005
Dump (Diagnostic Control Switch)	006
MS pattern (Diagnostic Control Switch)	009
MS address (Diagnostic Control Switch)	00A
LS pattern (Diagnostic Control Switch)	00C
LS address (Diagnostic Control Switch)	00D
Start	400
Display	402
Store	403
Log out	00F
Dump (Microprogram Interrupt)	001
Trap (CR = 3)	
During main storage read	404
During main storage write	405

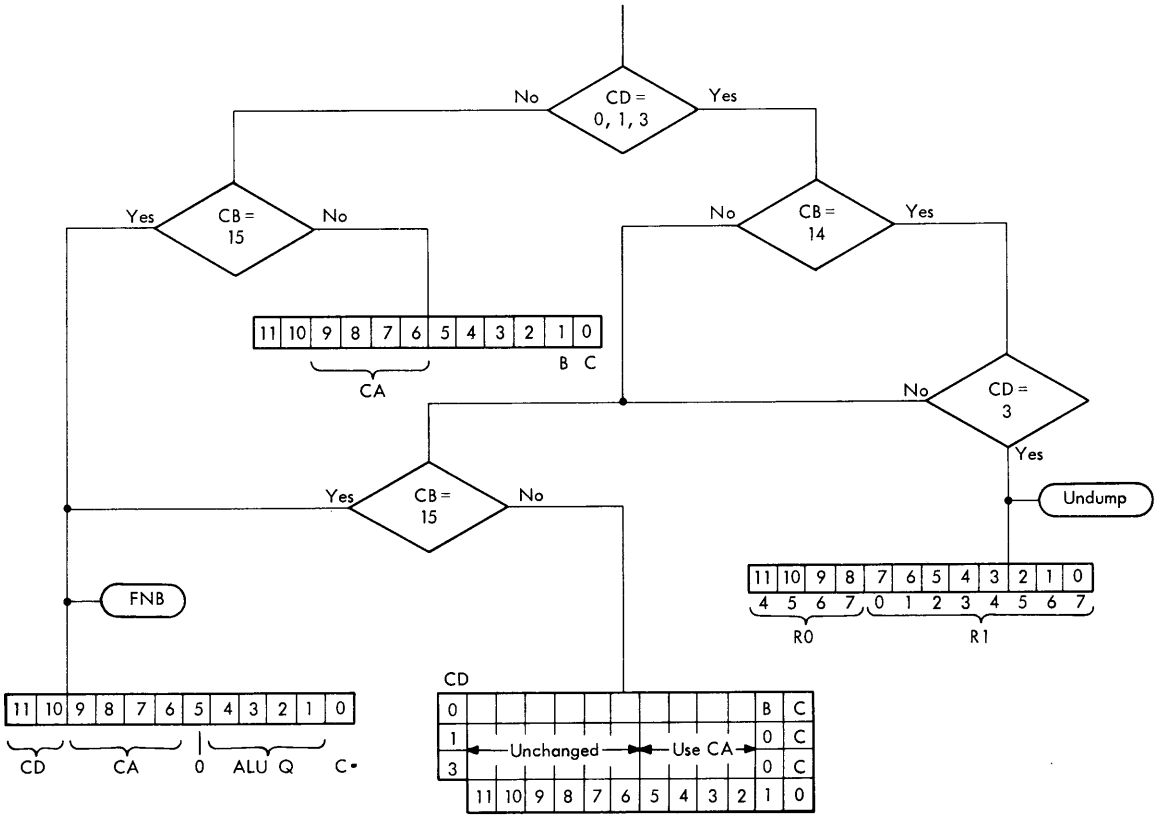


Figure 47. Determining Next TROS Address

Selector Channel Operations

For selector channel operations the following addresses can be forced into ROSCAR:

	TROS ADDRESS FORCED (HEX)
1 byte data service	700
2 byte data service	704
Terminal status in	708
Status after address in	710
Skip, count = 1	718
Skip, count > 1	71C

Examples For Generating Next TROS Address

1. CPU state, present TROS address 281 (hex)

CA	CB	CC	CD
1111	0000	1100	10

Not FNB, CD=2: basic format as in Figure 46

Present address:	
Bits retained:	0 0 0 0 0 0
CA field:	1 1 1 1
B condition:	0
C condition depends on PRI:	0/1
Next address:	0 0 1 1 1 1 0 0 0 0 0 X

Two way branch: if not PRI to 3C0
if PRI to 3C1

If the CC field has been equal to 1, the next address is always 3C1.

2. CPU state, present TROS address is 3C0; during the present cycle 00011010 is gated to the Q register (by the arithmetic statement; the data comes from within the data flow and may be an operation code of a machine language instruction).

CA	CB	CC	CD
0000	1111	00	10

CB = 15 = FNB: no bits of the present address are retained. Refer to Figure 46.

ROAR bits:	11 10 9 8 7 6 5 4 3 2 1 0
CD field:	1 0
CA field:	0 0 0 0
Forced zero:	0
Q register bits 0-3:	0 0 0 1
C condition:	0
Next TROS address	1 0 0 0 0 0 0 0 0 0 1 0 = 802 *hex*

With this instruction a 16-way branch is programmed. Depending on the Q register, which may have any value 0-15 in the high-order four bits, the 16 even hex addresses 800-81E may be generated.

3. Channel microprogram, I/O state, present TROS address is 646

CA	CB	CC	CD
1100	1011	1111	00

Not FNB, CD = 0, basic format as in Figure 46

Present address:	0 1 1 0 0 1 0 0 0 1 1 0
Bits retained:	0 1 1 0 0 1
CA field:	1 1 0 0
B condition depends on SVC-I:	0/1
C condition depends on OP-I:	0/1

Next TROS address: 0 1 1 0 0 1 1 1 0 0 X X
If neither condition is present, next address is 670.
If SVC-I is present but not OP-I, next address is 672.
If OP-I is present but not SVC-I, next address is 671.
If both conditions are present, next address is 673.

TROS Checking

ROBAR and ROBAR timing (4K TROS) are shown in Figure 48.

ROBAR

- **ROBAR holds the microinstruction address used during the previous cycle**

The read only storage backup address register (ROBAR) contains the contents of ROAR or ROSCAR used in the previous T cycle. It is loaded every T cycle at T4 del time.

When a machine check occurs, the output of any of the three check latches (control, early data, and late check) raises the line inhibit ROBAR or Q change. This inhibits the setting and resetting of ROBAR so that when the T clock is stopped, ROBAR contains the TROS address during which the error occurred. The contents of ROBAR are logged out. ROBAR is displayed on the main console.

TROS Address Check (Figure 49)

- **Detects internal addressing failure**
- **Causes a control check**
- **The TROS address and the control word with which the error occurred are retained**
- **Parity bit generated from address is compared with CF field; unequal result indicates error**

The output of ROSAB is XOR'ed to generate odd parity. The ROSAB parity bit latch is set at P4 time and is reset at P4 time in the next cycle. Therefore, it is still on in the next cycle when the control word appears at the sense latches. The sense latches are set at the end of T4. The output of the ROAR parity bit latch is XOR'ed with the CF field. An unequal compare sets the check latch at T1 time. The TROS address output sets the control check latch at T3 time.

Inhibit ROS is raised which inhibits the setting and resetting of the TROS sense latches. Inhibit ROBAR or Q change is raised, which inhibits the setting and resetting of ROBAR. ROBAR contains the address at which the error occurred. The T clock stops at the end of this cycle. The last T clock pulse is T4 del.

ROAR or ROSCAR may contain the next address, depending on what type of error occurred. TROS address check is indicated on the main console.

TROS Word Data Check (Figure 50)

- **Detects failure of TROS sense circuits, sense latches, and control latches**
- **Causes a control check**
- **The TROS address and the control word at which the error occurred are retained**
- **The complete control word is checked for odd parity**
- **CS field contains parity bit for the control word**

Each control word contains an odd number of bits. The CS field contains the word parity bit.

The output of the sense latches of control fields CA, CC, CF, CG, CH, CJ, CK, CP, CQ, CR and CT, and the output of the control latches of control fields CB, CD CE, CL, CM and CN are XOR'ed, together with the control word parity sense latch CS. An even result causes a TROS word data check. TROS word data check sets the control check latch at T3 time.

Once the control check latch is set, the sequence of events is the same as for the TROS address check. The TROS word data check is indicated on the main console.

Control Field Decoder Checks

- **Checks the decoders of control fields CB, CC, CD, CH, CJ, CN, CP, CQ and CR**
- **Causes a control check**
- **The TROS address and the control word at which the error occurred are retained**

The outputs of each control field decoder are XOR'ed; an even result causes a control check (Figure 50).

Note that the outputs of the CH field decoder are checked in three parts: LSAR destination, LSAR increment, and decode load LSAR. The decoder checks of control fields CP, CQ and the decode load LSAR error are latched up. The decoder checks of control fields CB, CC, CD, CJ, CN, CR, and LSAR destination and LSAR increment check are fed directly to the control check latch. Once the control check latch is set, the sequence of events is the same as for the TROS address check. Decoder checks are indicated on the internal CE panel.

ROAR Format Checks (Figure 51)

- **Check the correct setting of ROAR during some of the address modifications**
- **Cause a late check**
- **The erroneous ROAR contents are retained in ROBAR**

The data that have set the ROAR latches are XOR'ed with the output of the ROAR latches that have been set by

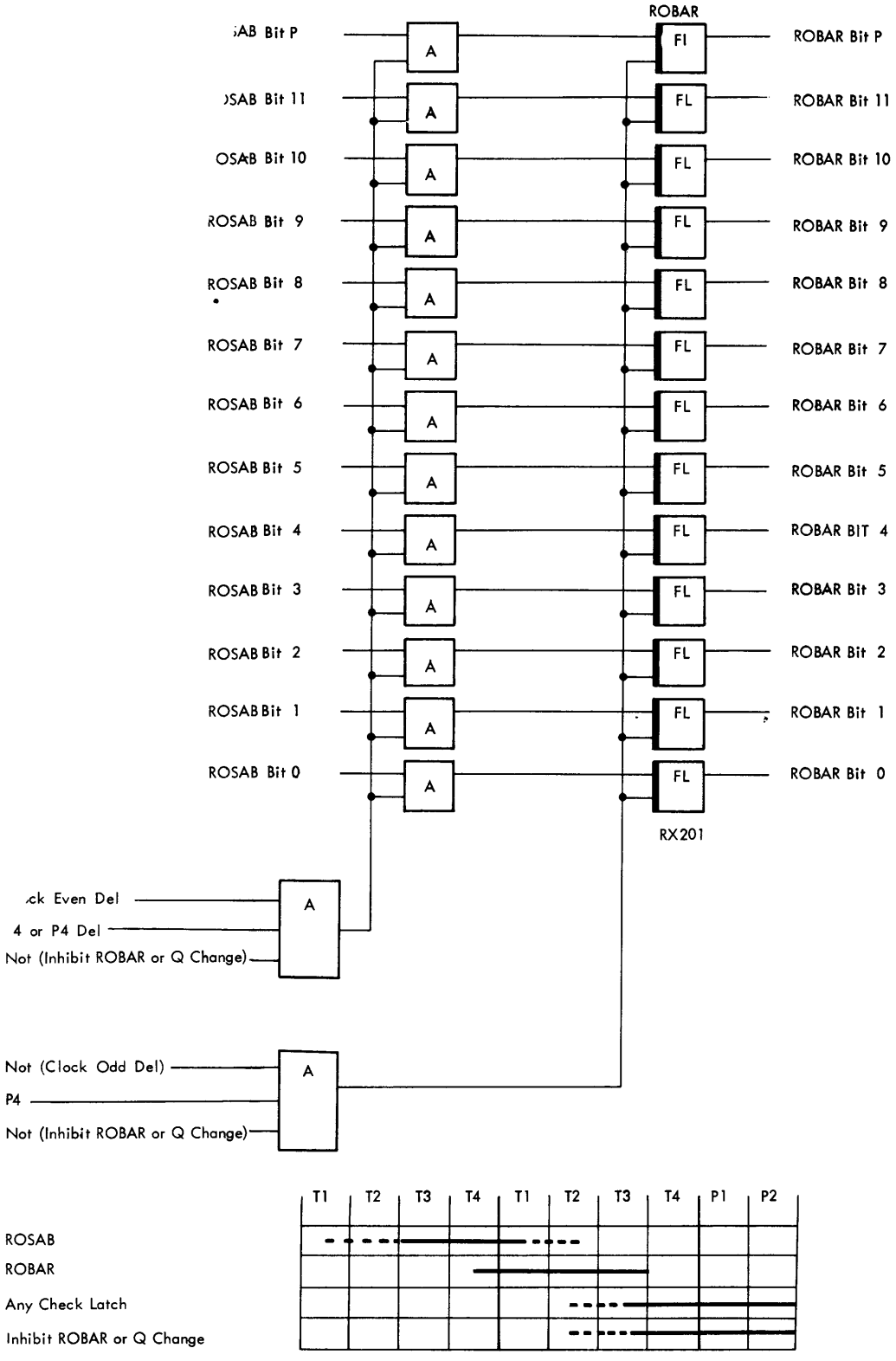


Figure 48. ROBAR and ROBAR Timing (4K TROS)

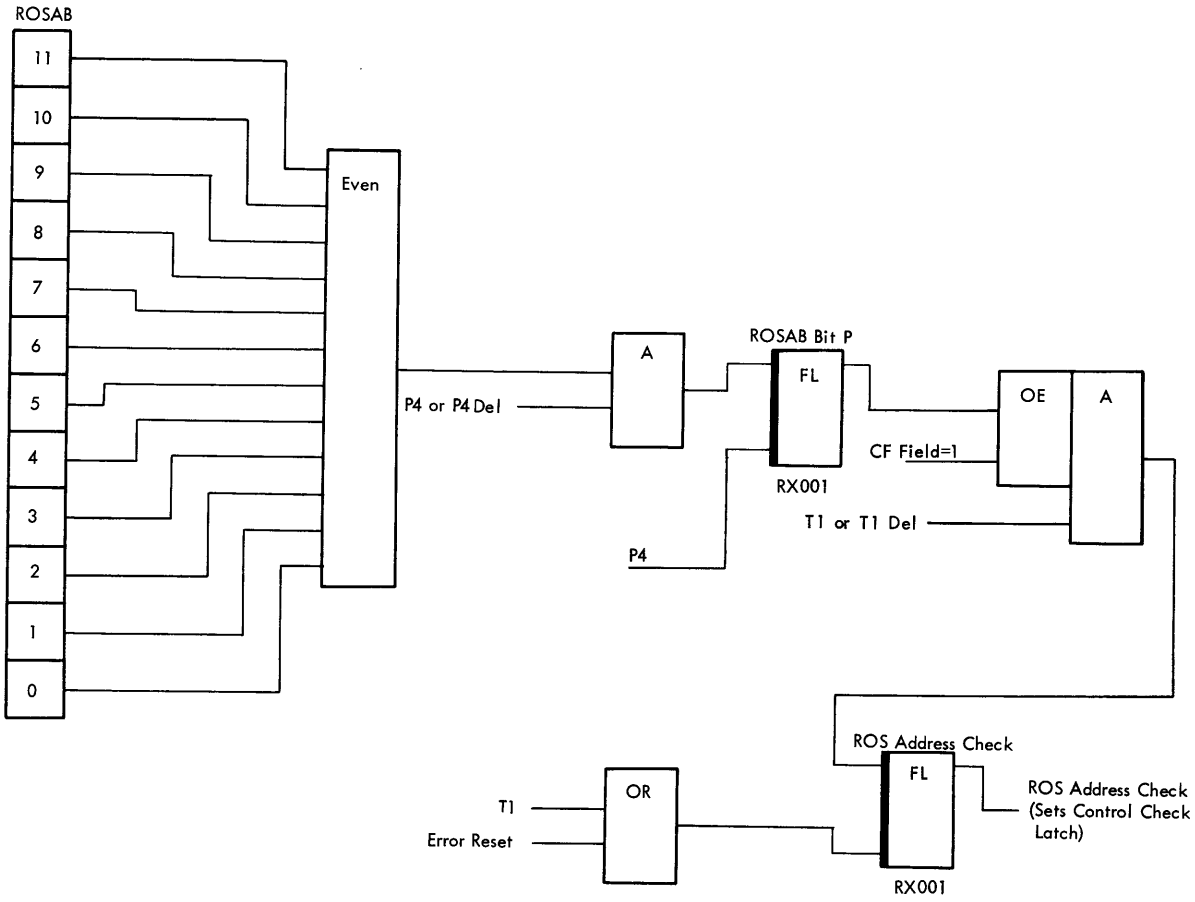
them. An odd comparison indicates an error and sets the late check latch at T3 del time. The transfers that are checked are:

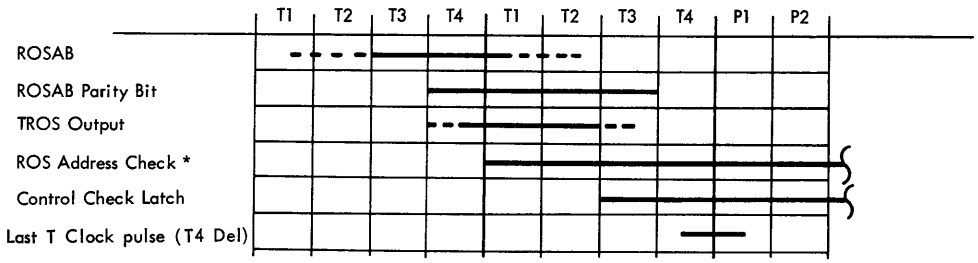
CD field to ROAR bits 11-10
CA field to ROAR bits 9-6
CA field to ROAR bits 5-2
Q bus bits 0-3 to ROAR bits 4-1.

The T clock is stopped at the end of the cycle. The last T clock pulse is T4 del.

The erroneous ROAR contents are retained in ROBAR.

The ROAR format check is displayed only on the main console by means of the late check indicator.





* Set when unequal comparison between ROSAB bit P and CF field bit

Figure 49. TROS Parity Bit Generation and TROS Parity Check (4K TROS)

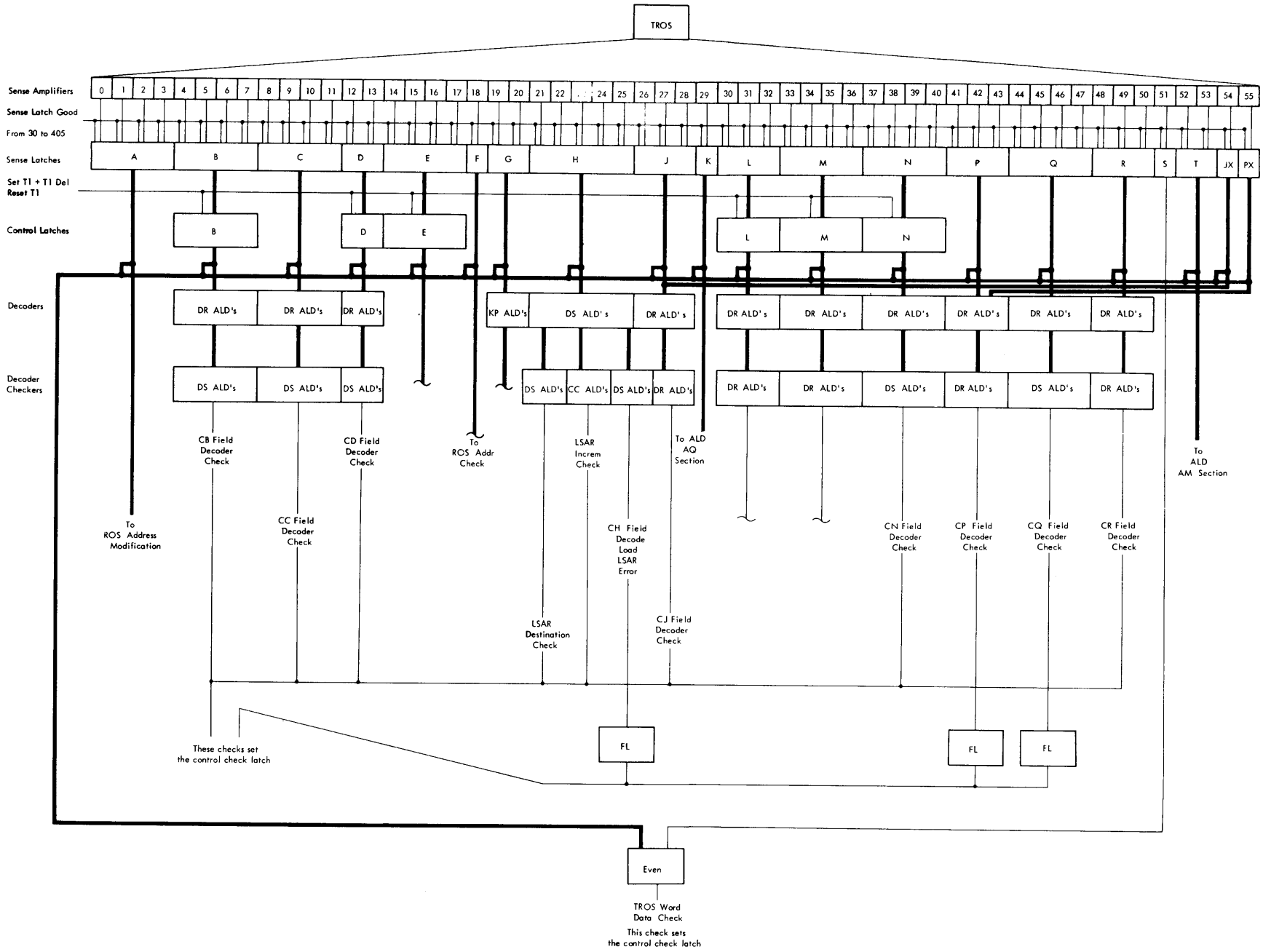


Figure 50. Data Check and Control Field Decoder Checks

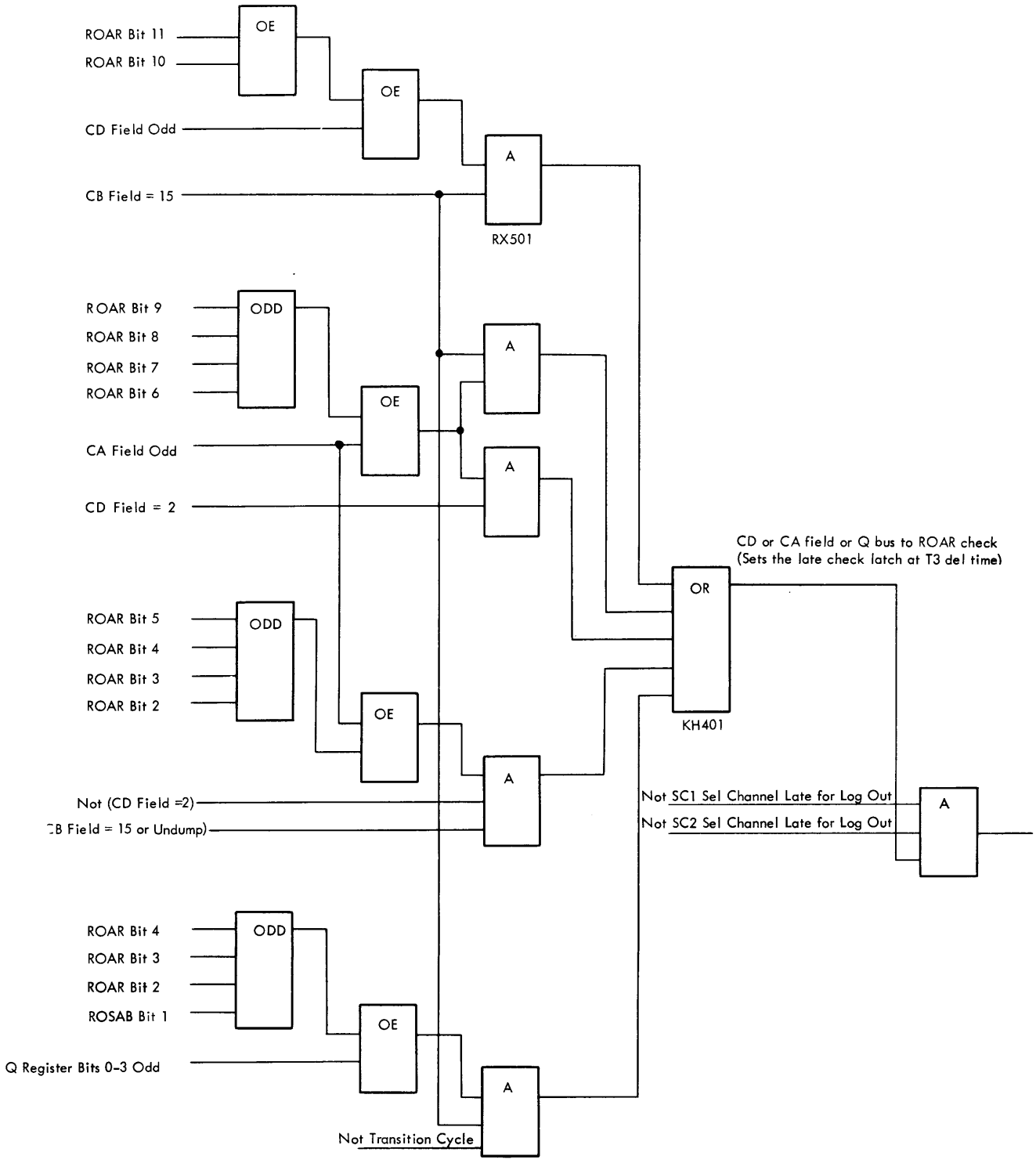


Figure 51. DAQX to ROAR Transfer Check

Overall Microprogram Block Diagram

- **All individual microroutines held in TROS form the microprogram that is responsible for over-all system operation**
- **Connection of the individual routines, by microprogram and logic circuits, defines the over-all function of the control**

All individual microroutines held in TROS constitute the microprogram responsible for all system operations. This microprogram is the center of the control section.

Figure 52 is an over-all block diagram where all routines necessary for system operations (diagnostic routines and special console operations are not shown) are interconnected, either by microprogram branches or with the interconnecting logic circuit functions. All basic functions of the CPU are shown; channel functions are indicated only in their relation to the CPU microprograms.

Individual Microroutines

The following is a list in alphabetical order of the basic operation of the routines shown on Figure 52.

CPU and Channel Checkout: Functional check of most of the basic circuits, especially circuits not permanently checked by logic circuitry.

Dump and Undump: Routine that stores the CPU data flow into local storage dump area and enters the Mpx channel microprogram for handling the microprogram interrupt (data or status service).

This routine can be initiated between any two microinstructions, provided that the dump is not inhibited. [Between main storage (MS) read and write, dump also may not take place]

Execution: One routine for every machine-language instruction (currently 144 with all special features).

I-Fetch: Reads the machine instruction to be executed from main storage (address as indicated) in the instruction counter (IC) in the current (PSW), decodes the operation code, branches to the appropriate microroutine, and updates IC.

IPL (Initial Program Load): Loads 24 bytes of data from the I/O device selected by the load-unit switches on the console into the main storage 0-23 (0-17 hex) (initial PSW + CCW + CCW) and command chains to the CCW in location 8.

Load PSW from MS into LSTOR: Loads the PSW from the main storage location corresponding to the new PSW of the particular interrupt (or location 0 after IPL) into local storage. If bit 14 of this PSW is 1, the system stays in the wait loop of this routine.

Load PSW into Data Flow: Loads certain PSW information from local storage into data flow circuits (protection key, system mark, etc.).

Logout: Transfers the contents of the entire data flow and the status of CPU and channel control latches into main storage starting at location 80 hex.

SC1 or SC2 Buffer Service: Break-in to transfer selector channel data to or from MS. This can occur at any time when main storage is free.

Store CSW: Stores all CSW information from the data flow into main storage location 40 hex.

Store-Display: Microroutine for manual store and display operations. This routine is entered also in its display function before the system goes into the stopped state (stop loop).

Store PSW from LSTOR into MS: Store the current PSW from local storage into main storage in the old PSW location.

Stop Loop: Single microinstruction which is cycled if the machine is said to be in manual stop condition.

System Reset: Validates the general and floating-point, registers, validates the current PSW, and clears the dump area into local storage. If system reset is entered from IPL, main storage is validated: (Validate —read out storage with possible errors ignored and write the same information back with correct parity).

Terminal Status and Status after Request In: Break-in to tell the CPU that status information is available on a selector channel I/O device. This can also occur at any time main storage is free. The SIR (set interrupt request) allows the microprogram to go into a program interrupt routine (next time PRI is tested). This is necessary to allow programmed testing of the status received from an I/O device. Note that a program interrupt results in a PSW change.

Update PSW into Local Storage: Stores PSW information held in logic circuits back into the current PSW in local storage.

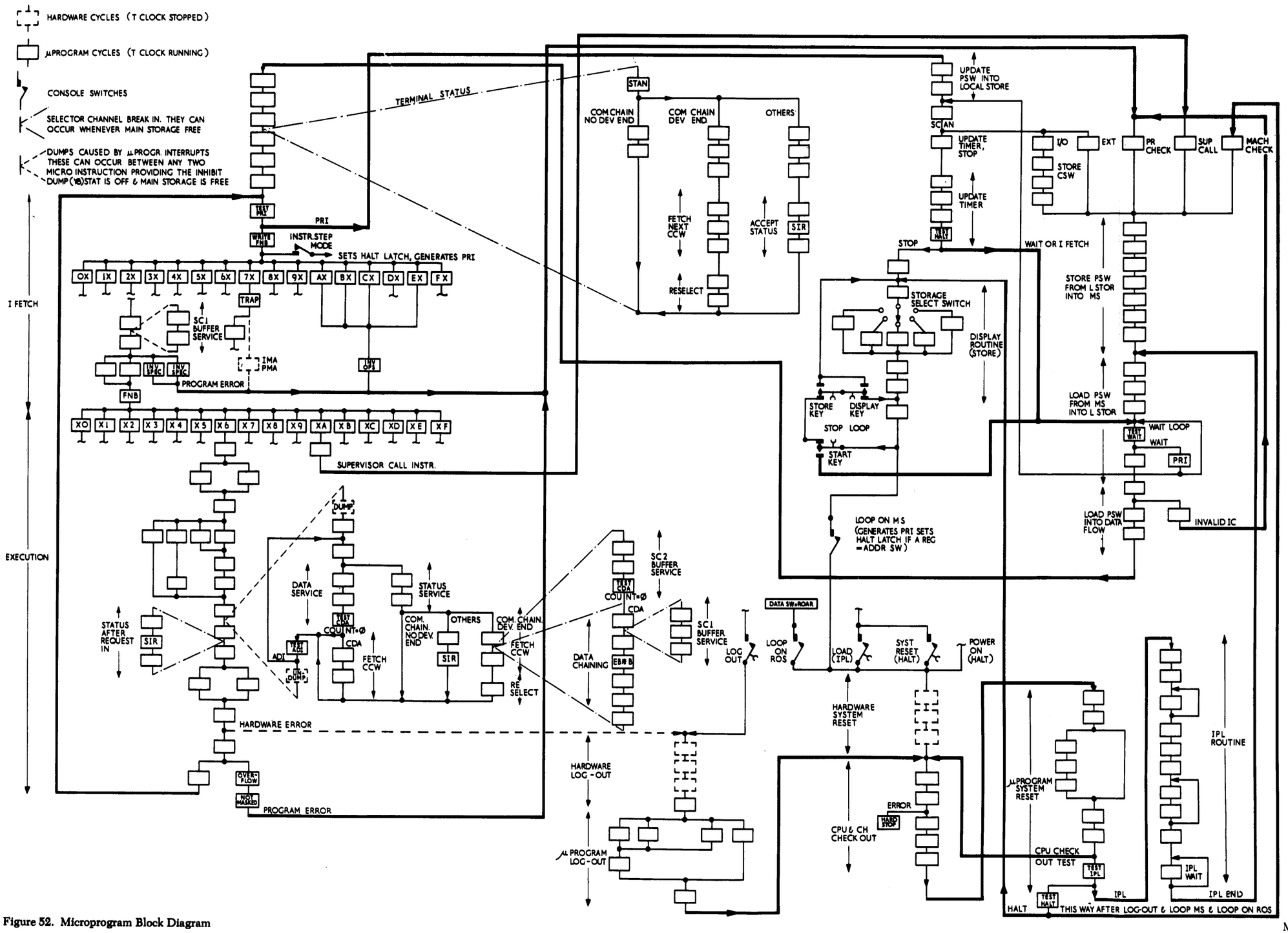


Figure 52. Microprogram Block Diagram

Circuit Functions

Following is a list of the basic purposes of the circuit functions shown in Figure 52. Circuit functions are listed in alphabetical order:

Display Key: Console key, generates display conditions and the start address of the store-display micro-routine.

Dump: Single hardware cycle initiated by microprogram interrupt. Stores ROAR into local storage and generates address of first microprogram dump cycle.

Hardstop: Machine errors detected in the CPU check-out, log out, system reset or IPL routines stop the T clock (hardstop condition).

Hardware Logout: Any machine error detected in other routines than these listed above under hardstop, stops the T clock immediately and initiates hardware cycles to store part of the data flow unchanged into local storage. Forces address of the first microprogram log out cycle into ROAR and starts T clock.

Hardware System Reset: The T clock is stopped, hardware cycles are initiated and all data flow registers, stats and other latched conditions are reset. The halt latch is set and the address of the first CPU and channel check-out microinstruction is forced into ROAR. The T clock is started again.

Instruction Step Mode: The PRI condition is generated every machine instruction and the halt latch is set on.

Load Key (Initial Program Load): Console push button, initiates hardware system reset and sets the IPL latch.

Logout Key: Initiates a log out when pressed, simulating exactly the occurrence of a circuit error.

Loop on Main Storage: Whenever the machine comes into the stop loop, the hardware system reset is entered.

Loop on ROS: Whenever the content of ROAR equals the setting of the data switches on the console, the hardware system reset is entered.

Power On Key: After machine power is switched on, hardware system reset is initiated and the halt latch is set.

PRI (Program Interrupt): Condition set by microprogram (channel programs) or logic circuits (stop key, external interrupts) if the corresponding interrupts occur and are not masked off (PSW bits 0-7). Tested by microprogram.

Program Errors: Program errors detected when the microprogram tests for them, result in a branch to the program check interrupt routine.

Start Key: Console push button, generates the ROAR address of the first instruction from the load PSW into data flow routine to leave the stop loop.

Store Key: Console key, generates store conditions and start address of the store-display loop.

Undump: Single hardware cycle at the end of the undump routine to restore dumped ROAR from local storage.

System Operation

- **During normal job execution the system is not hardstopped and is under control of the microprogram**
- **In the stop state, the microprogram is cycling in the stop loop which can be entered only by manual controls**
- **In the wait state the machine is cycling in the wait loop and is ready to accept interrupts (Depending on system mask)**
- **In normal CPU instruction execution, a closed loop of I-Fetch and instruction execution is performed**
- **PRI-generating program interrupts are taken in I-Fetch or in the wait loop; function performed: exchange of PSW's**
- **Microprogram interrupts caused by I/O operations are taken between two microinstructions**
- **The only malfunctions which hardstop the system are solid machine errors detected in CPU check-out, log out, system reset or IPL routines**
- **For diagnostic purposes some other modes of operation are possible**

During normal job execution, the system is never hardstopped and is always under control of the microprogram. To illustrate this statement, the sequence of events to run one simple program is given. Assume that no supervisor program is in charge of over-all system control.

Program and detail cards are in the card reader, the program calls for reading cards and printing group totals on a printer. Refer to Figure 52.

1. Power-on key pressed: After power is brought up, hardware system reset takes place, followed by CPU check-out and microprogram system reset. Since the halt latch was set by power on, the microprogram branches to the stop loop.
2. Reader and printer are made ready.
3. The address of the reader is set into the load unit switches of the console; the load button is pressed.

The hardware system reset, CPU and channel check-out, and microprogram system reset routines are again executed. This time, because the load (IPL) stat is on, the exit from microprogram system reset is into the IPL routine.

The reader is started by a command to read 24 bytes from the first card into main storage 0-17 hex; the microprogram stays in the IPL wait loop until the channel program reaches the end of the I/O operation.

During this time the IPL wait loop is interrupted several times by microprogram interrupts caused by the I/O operation. With IPL, command chaining to the next CCW is generated.

It depends now entirely on the information punched into the first card, where in main storage the following program cards are stored, and whether a short load routine is entered first, or whether all program cards are read by command chaining. As soon as command chaining is no longer specified, and channel end is generated, IPL end is indicated.

The IPL wait loop is now left and the first PSW (punched in the first card, columns 1-8) is loaded into local storage from main storage location zero. With the wait bit off, PSW information is loaded into the data flow and I-Fetch is started from the instruction count (IC) specified in this first PSW.

Subsequently, machine instructions are executed. The system is now in a loop of I-Fetch and execution phases.

One of the machine instructions will be a start I/O for the card reader to read a detail card. After this instruction, the program has to wait until the information is available in main storage. Several possible ways exist in which this waiting period may be programmed:

1. Loop on a test I/O instruction until the channel is found to be free again, indicating that the read operation has finished.
2. Load a PSW with the wait bit on and the system mask on (load PSW instruction). The system will then stay in the wait loop until the channel-end interrupt is generated.
3. A supervisor call instruction and a pre-stored new SVC PSW with the wait bit on and the system mask on "allow."

When the system waits in the wait loop for a channel-end interrupt, the wait loop is left with a new PSW in control of the system; I-Fetch starts from the location indicated by the new IC.

The program now stays in a loop to read cards and to accumulate their contents.

In a subsequent instruction, when it is found that a total has to be printed, the printer will be started. At the end of the print operation an interrupt will be generated and the current PSW will be replaced by the new I/O PSW. If the interrupt occurs during instruction execution, it will be taken during I-Fetch after the current instruction is completed.

The programmer must remember all interrupts possible in his program and must make the necessary provisions for the different interrupts to be properly handled, i.e., masking off interrupts he does not want to accept during certain program segments, and designing the program introduced by the new PSW's accordingly.

The philosophy of a processor which is operating normally (processing instructions or waiting, but not stopped) is important in real-time and teleprocessing systems. In the operating state, the timer is updated; interrupts by inquiries from transmission lines etc. are accepted.

Error Handling

Program Errors (detected by microprogramming) generate an immediate branch condition to the program check interrupt routine. Consequently, the new program check PSW controls further operations, and it is up to the user how the particular condition should be handled.

Machine Errors (if enabled by PSW bit 13) stop the T clock at the end of the cycle in which the error was detected. The hardware logout sequence, followed by microprogram logout, CPU check-out and system reset, is executed.

If no further check is indicated by these routines (an intermittent error is assumed), the machine check interrupt is performed and again the system is under control of a PSW pointing to a special program segment where the programmer can analyze what corrective actions are necessary.

Manual Operations

Pressing the console stop key generates PRI and the microprogram branches from I-Fetch to the stop loop. This is the necessary machine state for most console operations and can be entered only manually.

In addition to store and display, a number of other console-controlled operations are implemented for diagnostic purposes. The system can be cycled in a single-instruction-step and single-microinstruction-step modes, and machine errors can be immediately hard-stopped.

CAS Logic Diagrams

- **CAS is the abbreviation for Control Automated System**
- **CAS logic diagrams (CLD's) represent the various microroutines in a symbolic notation**
- **On CLD's the individual microinstructions are represented by blocks which define symbolically the functions performed by the TROS control word. Blocks are connected to each other to represent the actual instruction sequence**

All microroutines held in TROS are symbolically documented on CAS logic diagrams (CLD's) which form part of the diagrams necessary to maintain the system; in a similar form ALD's are used to document hardware circuitry.

Microroutines are designed by microprogrammers in symbolic language. This information is then punched into IBM cards and is fed into CAS. CAS (control automated system) is a number of computer programs which translate the symbolic microroutines into information for production of TROS tapes and which print the CLD's.

On CLD's the individual microinstruction is represented by a block which defines the actual TROS control word in symbolic notation. Symbols used are the symbolic statements introduced for the various control fields (see TROS Control Word and Detailed Data Flow).

Individual microinstructions are connected to each other, representing the actual instruction sequence.

Outside the CLD blocks, comments are added to relate the particular microinstruction to the object function of the routine (incrementing main storage address of operand 1; test for wrong length record; set the condition code into local storage; etc.).

Interpretation of a Single CAS Block

- **All TROS controls fields necessary to define the function of the block are represented in symbolic notation. Interpretation of the symbols is given in the control field specifications**
- **Statements in the CAS block are grouped according to their functions**

Each CAS block is printed on a grid measuring 15 x 8 print positions. No matter what the actual function of the block, a frame printing is always present. For example:

Print line 1:

positions 2-6 = branch leg identifiers
positions 7-12 = frame symbols
positions 13-15 = TROS address of the present microinstruction (3 hex digits)

Print line 8:

positions 1-2 = two-character co-ordinate to identify the block position on the CLD sheet positions 3-5 = frame symbols
positions 6-10 = branch leg selector
positions 11-13 = frame symbols
positions 14-15 = block serial number. This number is originally in relation to the block co-ordinate, but remains unchanged if the CAS blocks on a CLD are re-arranged.

Edge characters in print positions 1 and 15:

Edge characters specify the TROS control information on the corresponding print lines

E = Emit statement
A = Arithmetic statement (8-bit ALU data path)
L = Local storage address statement (LSAR loop)
D = Data transfer statement (16-bit R register path)
C = Special controls
S = Main storage controls
R = ROS address control
I = Frame symbol

The edge characters appear in the framework of the block only if there is a data line to go with them. Unused locations for floating edge characters are filled with the frame symbol.

One data line can contain a maximum of two statements. Right-hand edge characters (position 15) are printed only if the second statement on the line is of a type different from that indicated by the left edge character.

The Emit Statement (Line 2, first statement)

The controls of the CE, CN and CK field are combined in one statement. Three types of emit statement are possible:

Type 1: A simple statement for the binary value of the CE field. This format is used in the absence of one of the following two types (for example, if the emit field is used only to introduce data to the Q register).

Type 2: A statement that combines CE and CN fields to set and reset stats.

Examples (refer to CN field symbols)

1100 ➔ YA
YA Ω 1010
YA • ⌝0010

Type 3: A statement that controls the setting of the ALU function register (CN = 15).

Examples (refer to CN and CK field symbols)

0000, OR	Set function register bits 0-3 to perform OR
0011, SUQ	Set function register bits 0-3 to perform P−Q binary
0011, SUQ*	Set function register bits 0-3 to perform P−Q binary and set bit 4 to skew Q input

The statement in Figure 53 specifies:

1111, ADD*	Set function register 0-4; P + Q binary, Q skewed

The Arithmetic Statement (Line 3)

Controls of the CG, CK, CM, CP and CQ fields are combined in one statement.

The statement is of the general form

(Specified by)	P	operation	Q	➔	result
(Field)	CP	CG	CQ	CK	CM

Examples (refer to appropriate symbols):

A1 + 0E ➔ C1:	direct add A1 + 0000 and emit field, gate the ALU output to C1.
B0 + B1* ➔ B0:	direct add B0 + B1 skewed, gate the ALU output to B0.

The statement in Figure 53 specifies

B0 ? C1* ➔ Y:	indirect ALU function (the function register is set in the same instruction to P + Q skewed). Actual function: indirect add B0 + C1 skewed, gate the ALU output to Y0 − Y7.

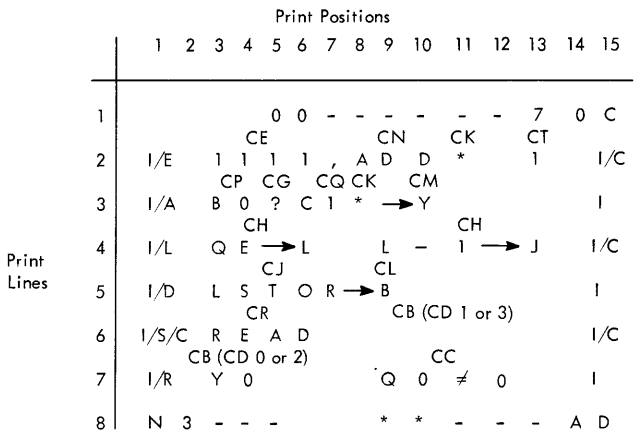


Figure 53. CAS Block Frame

The Local Storage Address Statement (Line 4)

Controls of the CH field are specified. The function is written as two statements: load LSAR and store LSAR. Examples (refer to CH field symbols)

J ➔ L L + 1 ➔ J Load LSAR with the J register, use LSAR to address local storage. Store LSAR back to J incremented by +1.

Figure 53 QE ➔ L L − 1 ➔ J
Load LSAR with emit field bits 0-2, Q register bits 0-3 emit field bit 3 (in this example; 1 1 1 $C1_0$ $C1_1$ $C1_2$ $C1_3$ 1) use LSAR to address local storage. Store LSAR back to J decremented by 1 (in this example 1 1 1 $C1_0$ $C1_1$ $C1_2$ $C1_3$ 0)

The Data Transfer Statement (Line 5)

Controls of the CJ and CL fields are combined in one statement. The statement is of the general form:

Input to R register (CJ field) ➔
Destination of R register (CL field)

Examples (refer to CJ and CL field symbols)

D ➔ A : transfer D register to A register
C ➔ LSTOR : call local storage write and store C

Figure 53: LSTOR ➔ B:
call local storage read and gate read data to B

The Main Storage Control Statement (Line 6)

If a main storage statement is present, it is printed as the first statement of line 6, identified by edge character S.

The only symbols possible:
CR = 1; Read; call main storage read.
CR = 2; Write; call main storage write.

The Control Statements

There are four types of control statements, all of which can occur simultaneously.

Type 1 (Second Statement Line 2): Controls of the CT field.

0=reset the YC carry stat before ALU operation.
1=set the YC carry stat before ALU operation.

Example (Figure 53): set YCI

Type 2 (Second Statement Line 4): Special controls of the CH field for selector channels (INT and REST).

Type 3 (First Statement Line 6): Identified by the edge character C; the remaining CR field controls are indicated here (the main storage controls of the CR field are indicated with edge character S). Refer to CR field symbols.

Type 4 (Second Statement Line 6): Identified by the edge character C; this statement specifies all the special controls of the CB field if CD = 1 or 3. (Refer to CB field symbols for CD = 1 or 3.)

The TROS Address Control Statements (Line 7)

The first statement on line 7 is the B condition test specified in the CB field if CD = 0 or 2 (or FNB if CB = 15). Bit 1 of the next TROS address is forced according to the result of the test (refer to CB field symbols for CD = 0 or 2).

This statement and the control statement type 4 are mutually exclusive. Note, however, that the special branch conditions CB = 0 and CB = 8 with CD = 3 (STAN and EDIT) are printed as a control statement on line 6.

Example (Figure 53): Set ROAR bit 1 for next TROS address if Y0 is on.

The second statement on line 7 is the C-condition test specified in the CC field.

Bit 0 of the next TROS address is forced according to the result of the test (refer to CC field symbols).

Example (Figure 53): Set ROAR bit 0 for next TROS address if the Q register bits 0-3 are non zero (in the example: if high-order four bits of C1 are not zero).

Determination of Microinstruction Sequence

- **The microinstruction sequence is determined by the next TROS address generated in the present CAS block**
- **Sequencing is indicated by printed connections**
- **Branch leg selector and branch leg identifier are provided**

The information contained within one CAS block is not sufficient to construct the next TROS address, since CA and CD fields are not specified.

Determination of the next CAS block, however, is simplified, as the individual blocks are connected to each other in their logical sequence (Figure 54).

To determine the next block for microprogram branches, a branch leg selector and a branch leg identifier are given in every block.

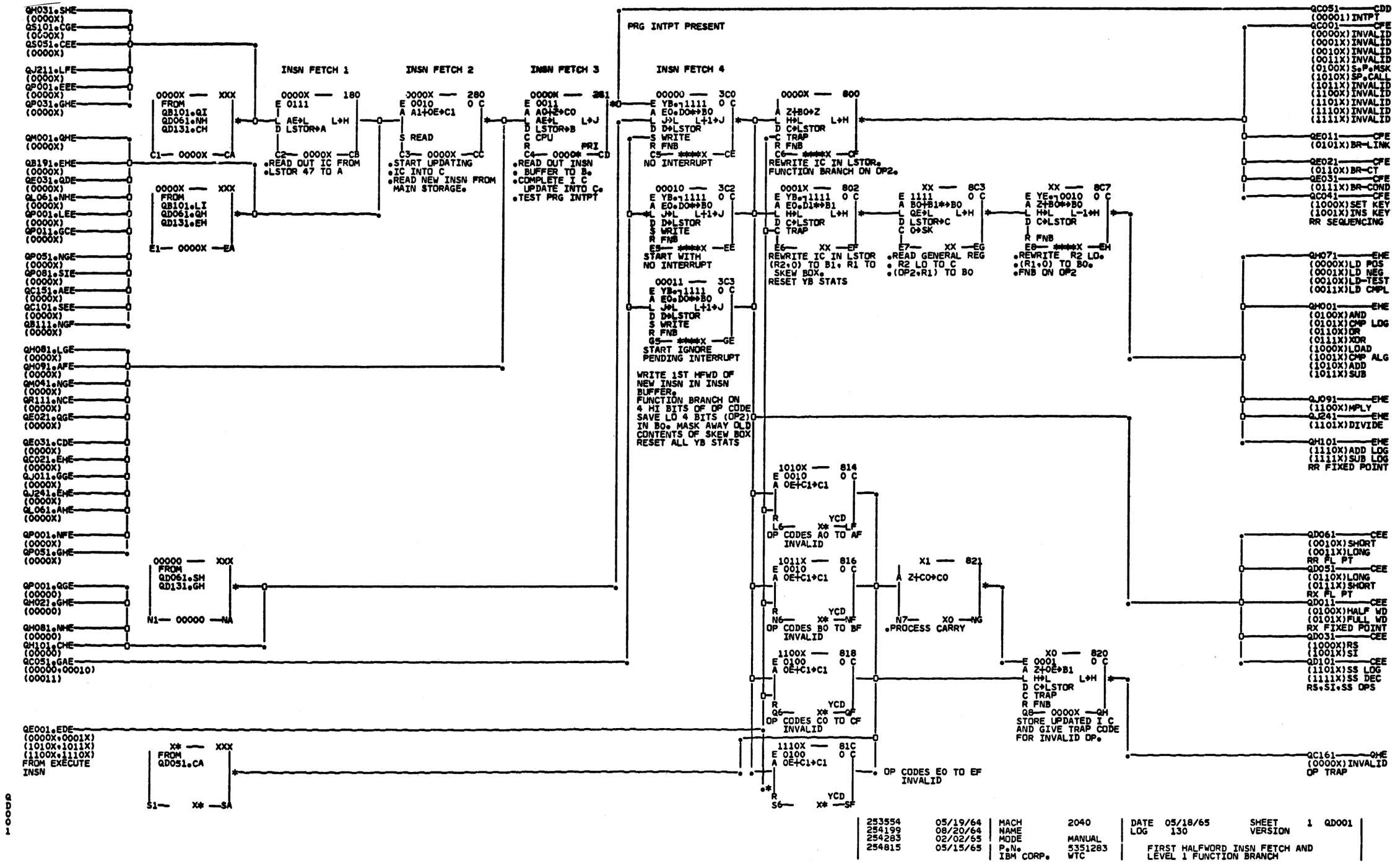


Figure 54. CLD Sample

Branch Leg Selector (Figure 53, Line 8, Positions 6-10)

Basically this is a five-bit binary number, specifying low-order five bits (4-0) of the next TROS address. Five characters are printed only if in the CAS block FNB is specified. Without FNB, positions 6-8 are blank.

The following characters may appear:

X – don't care bit, i.e. the block connection determines the next block, the B and C condition are coded accordingly but not shown in the R statement.
0 – The corresponding bit is forced to zero.
1 – The corresponding bit is forced to one.
* – The corresponding bit is determined by the test result of the R statement.
EXAMPLES:
XX – the next block is defined by the block connection only.
X* – two-way branch on the C-condition bit 0 of ROAR is determined by the result of the C-condition test. The block connection leads to two blocks, the proper block is defined by the branch leg identifier.
*0 – two-way branch on the B-condition, C-condition is forced to 0. This form is used if the block connections are drawn for a four-way branch and the present block is an additional input to only two of the four blocks.
** – four-way branch on B and C condition.
****X – 16-way functional branch, the C-condition is a "don't care" bit (either 0 or 1 but not specified in the R statement).

Branch Leg Identifier (Figure 53, Line 1, Positions 2-6)

Basically a five-bit binary number, specifying the low-order five bits (4-0) of the current TROS address. Five positions are printed only if the current block is one of the 32 legs of a preceding FNB. Without this condition, positions 2-4 are blank.

The following characters may appear:

X – "don't care" bit, i.e. the block connection determines how the previous block is connected to the present one.
0 – the corresponding bit of the present address is 0.
1 – the corresponding bit of the present address is 1.
EXAMPLES:
XX – the present block is defined by the block connection.
X0 – the present block is one leg of a two way C-condition branch, it will be addressed if the C-condition test of the previous block is not satisfied.
0110X – the present block is one leg of a 16-way functional branch (the C-condition in the previous block is not specified).

Actual Sequence

The CAS blocks on Figure 54 are discussed as an example of sequence interpretation.

1. Locate each block on the CLD with its two-character co-ordinate (Line 8, positions 1 and 2).

2. Start in block C2.

The block connection shows that only one possible next block exists: C3. The branch leg selector specifies XX.

The actual TROS addresses are not of importance in this sequence. Note however, that C2 and C3 do not have sequential addresses (line 1, positions 13-15, 180 and 280 respectively).

Also CA and CD field are not shown in the CAS block, they can be defined by investigating the two addresses:

Address of C2: 180 hex = 0 0 0 1 1 0 0 0 0 0 0 0
Address of C3: 280 hex = 0 0 1 0 1 0 0 0 0 0 0 0

C2 does not specify FNB, but bits 9 and 8 of ROAR have to be altered: CD has to be =2 to insert the CA field into positions 9-6 and CA will be 1010.

3. Next block to C3 is C4, again defined by block connection.

4. C4 connections are drawn as a four-way branch; the leg selector of C4 however is 0*, i.e. B-condition is forced to 0 and only C-condition is tested. This two-way branch can lead only to the blocks with branch leg identifier 00 or 01.

The C-condition of C4 specifies PRI. Depending on the status of this latch, the next block is either C5 (PRI off) or a block on CLD QC051 (PRI on) as indicated by the off-page reference.

Blocks E5 and G5 can be addressed only from the incoming connections.

5. Assume PRI is off, the next block is C5. C5 specifies FNB but no C-condition test. The branch leg selector is ****X, a 16-way branch is performed and the block connections show a number of possible next blocks as well as off-page references.

The four asterisks of the branch leg selector actually specify bits 4-1 of the next TROS address. With FNB these bits are set according to Q register bit 0-3.

To replace the asterisks with a 4-bit number, the Q register contents must be known.

The arithmetic statement specifies D0 to Q. To determine what value is currently in D0 the previous blocks must be investigated to find a statement that puts information into D0. (In this case, block C3, main storage read.)

For the moment the function of this entire routine is not important; assume D0 contains 10101111. The branch leg selector thus becomes 1010X, the next block is L6 (scan all connected blocks and off-page references for a branch leg identifier 1010X).

6. L6 – two-way branch on YCD. If YCD is off, next block is Q8; if YCD is on, N7 is inserted between L6 and Q8.

The arithmetic statement of L6 specifies updating of C1; N7 is obviously a block to propagate a possible carry to C0. This block however is executed only if in L6 a carry has actually been generated.

7. Q8 – specifies FNB but zeros are forced to the high-order Q register bits and the C-condition is ignored.

This indicates that there is only one possible next block, properly indicated with a branch leg selector that contains no asterisks but the actual zero address bits of the next block. The off-page reference points to the CLD where this block will be found.

The type of coding used is necessary, since the present TROS address has to be altered in the two most-significant bits (bits 11 and 10) which is possible only with FNB and an appropriate CD field.

A summary of considerations for proper interpretation of CLD's is:

1. Timing considerations.
2. Priority considerations.
3. Miscellaneous considerations.

Timing Considerations

- **Over-all timing is basically as shown in Figure 1**
- **Essential timings for interpretation of CLD's are given in Figure 55**

The basic timing relationships as shown in Figure 55 are sufficient to interpret properly the functions performed by the CAS block statements.

For interpretation of special TROS field decoding, timings of CR field functions and Y stats are essential (CPU I/O state, skew buffer, Y8, Y10).

Determination of the next TROS address requires actual timings of ROAR or ROSCAR set up (Figure 55).

If the CR field specifies set or reset of the CPU stat, decoding of TROS control fields affected by the CPU stat is already in accordance with the new setting in the same cycle.

TROS field decoding depending on Y stats (Y8, Y10) is not affected if the stats are altered in the current cycle.

B-condition and C-condition tests normally reflect results of the current cycle; testing of Y stats however is related to the old setting. (Not true for YCD and YCI which do reflect the result of the current cycle.)

When the ALU function register is loaded, and an indirect function is specified in the same cycle, the new indirect function is already executed.

The skew buffer is reset at the end of the cycle in which it is specified; the contents used in this cycle are the old contents. (True for both CK=1 or CR=6.)

Priority Considerations

- **If arithmetic and data-transfer statements specify the same destination, the arithmetic statement overrides the data transfer.**
- **For data to the D register, data transfer statement overrides main storage read data**
- **The AX bits override the CX bits if A0 and C0 are specified as ALU entry**

The arithmetic statement overrides the data transfer statement.

EXAMPLE:
A0 + B0 → B0
and C → B
Result:
B0 contains the result of A0 + B0
B1 contains C1

Data transfer statement overrides main storage read data.

EXAMPLE:
A0 + B0 → D0
and A → D
and the previous block calls main storage read
Result:
D0 contains the result of A0 + B0
D1 contains A1
Main storage data are lost

Note: Arithmetic statement alone to override one byte of MS read data is not allowed (invalid microprogram statement).

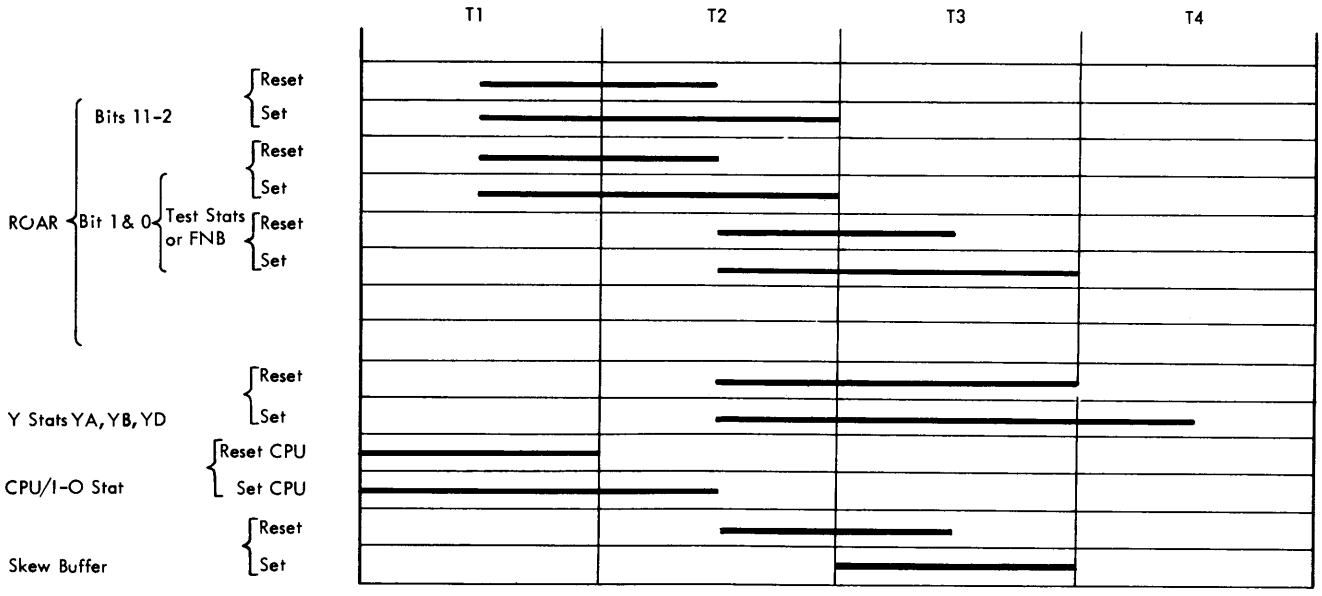


Figure 55. Essential Timings for Interpretation of CLD's

AX overrides CX on the ALU extension.

EXAMPLE:

A0 + C0 → C0

Result on ALU entry:

AX to extension
A0 to P
C0 to Q
CX is lost

Miscellaneous Considerations

Direct ALU functions affect the YCD stat.

Indirect ALU functions affect the YCI stat.

Logic ALU functions do not affect YCD or YCI.

If neither A0 or C0 is specified on ALU input, zeros are gated to the ALU extension.

If A0 or C0 is the destination for ALU output, the possible carry is indicated by YCD or YCI, but is also included in the extension output and gated to AX or CX.

Incrementing or decrementing LSAR affects only bits 4-7; a possible carry out of bit 4 is not propagated into bit 3.

If no arithmetic statement is specified in the CAS block, the TROS control word specifies the function Z • Z → ZZ.

If no local storage address statement is specified in the CAS block, the TROS control word specifies BE → L.

If no data transfer statement is specified in the CAS block, the TROS control word specifies Z → Z.

Additional Information on CLD'S

- **Actual TROS addresses are printed in every CAS block. Cross-reference between TROS address and CLD is provided by means of address lists**
- **Off-page references are included**
- **Individual CAS blocks are normally commented**
- **The CLD heading provides proper identification of the individual CLD page**

Every CAS block contains the actual TROS address as a three-digit hexadecimal number (line 1, positions 13-15).

The address is used to locate a TROS tape physically within TROS, or to find the CLD location of a CAS block if only its address is known (example: hardstop with ROBAR pointing to a certain TROS address).

TROS addresses and CLD's are cross referenced in the address lists which are part of the system diagrams. The address lists also show the actual TROS control word coding (for all fields) and give additional information like stagger class, driver gates affected, etc.

Lines leaving the CLD page are referenced with a net specification from which block on the page they originate (block co-ordinate) and the destination page to which they are going.

Lines entering the CLD page are referenced with the page and net specification from which they are coming.

Off-page references for special versions (microprograms which are in TROS only if certain special features are installed such as floating point, or storage protect) are shown in pseudo CAS blocks labelled "From" and "Go To" blocks. These blocks are shown on standard CLD's although the particular version is not installed and allow a quick distinction between standard and version-connections.

Most CAS blocks are commented. The function of the individual block is related to the over-all objective of the routine and is identified or explained in terms of this function.

For proper identification of CLD's, a heading is provided which contains data such as:

CLD page number	QD001
Title	First syllable I-Fetch
Machine type	2040
Part number of CLD	5351283
EC level	253554 dated 19-5-64

CLD Example of I-Fetch

- **Read machine language instruction from main storage**
- **Interpret the operation code and branch to the appropriate microroutine**
- **Test for program interrupt**
- **Test for invalid operation codes and invalid specification of operands**
- **Update instruction count (IC)**

Tie-in of I-Fetch routine is shown on Figure 52. The routine is entered either from the program segment that loads a new PSW into the data flow or the system is already executing machine-language instructions, looping I-Fetch and execution phases.

The updated instruction count (IC) is held in local storage, indicating the main storage addresses from which the next machine-language instruction has to be fetched. The operations involved in I-Fetch are as follows:

1. Read machine-language instruction from main storage. If the main storage location specified is found to be invalid, a microprogram branch to the program check trap occurs.
2. Interpret the operation code and branch to the appropriate routine. If the Op code or the specification of operands is found to be invalid, branch to the program check trap.
3. Update IC.

If during I-Fetch, PRI is on, i.e. a program interrupt is requested (PRI is set by I/O or external interrupt requests or by pressing the stop key), the microprogram branches to the interrupt routine.

Entry To I-Fetch

Several entries to the I-Fetch routine are possible in order to save execution time for this constantly used microinstruction sequence. It is possible that at the end of the an execution phase, one to three steps of I-Fetch can already be executed because reference to main storage is no longer necessary and the required TROS control fields are available (steps during which test and set of the condition code are programmed for example).

A further entry is provided for the execute instruction. Objective here is actually an I-Fetch operation of a single instruction outside the normal sequence.

Exits of I-Fetch

The exits of I-Fetch are branches to the individual instruction routines specified by the operation codes or branches to special routines for exceptional conditions.

Depending on the instruction format (RR, RX, RS, SI, SS), one to three halfwords have to be read out of main storage and IC has to be updated accordingly. The corresponding I-Fetch segments are therefore called:

First Halfword I-Fetch
Second Halfword I-Fetch
Third Halfword I-Fetch

Exits to the individual instruction routines depend on the instruction format and can be after first, second or third halfword I-Fetch.

First Halfword I-Fetch (Figure 54)

An instruction halfword is sometimes referred to as a 'syllable' hence the title of this CLD sheet (Figure 54).

Example:
IC, held in local storage indicates a main storage address where the machine instruction RR fixed point add, general register 7 + general register 8, is specified. Format of instruction:

OP	CODE	OPERANDS	
Op 1	Op 2	R1	R2
0001	1010	0111	1000

Sequence Of I-Fetch Using CLD Example

Block C2

Function: IC is read out from local storage into A register. Local storage is addressed with 0100 0111 (address of IC, 47 hex). This address is stored back into H.
Data Flow at End of Cycle:

A – IC
H – 47
Other data not relevant.

Block C3

Function: A1 (low-order byte of IC) is updated by 2, the result is stored in C1. Main storage is called to read.

Data Flow at End of Cycle:

A – IC
C1 – IC + 2
H – 47

Block C4

Function: A possible carry from the previous addition is propagated to the high-order byte of IC; the result is stored in C0 (and CX). The instruction buffer (IB, local storage location where the first halfword of the current instruction is stored for further reference) is cleared, the old contents are gated to B, but no longer used. LSAR is addressed with 0100 0011 (address of IB, 43 hex) the address is stored back to J. The CPU I/O stat is set to CPU. The C-condition test looks for PRI.

Data Flow at End of Cycle:

A – IC
B – IB
C – IC + 2
D – Op 1, Op 2, R1, R2 (main storage read data, 0001 1010 0111 1000 in this example)
H – 47
J – 43
If PRI is on, I-Fetch is abandoned and a branch to the interrupt scan routine takes place.
If PRI is off, next block is C5.

Block C5

Function: Function branch on Op 1 (0001). Main storage is called to write back (restore) the machine instruction; the instruction is also stored in local storage IB. Local storage is addressed with J (IB address). This address is up-dated by +1 and is stored back to J (which now indicates local storage where other PSW information is stored. ALU function separates Op 1 and Op 2. Op 1 enters the skew buffer, Op 2 0000 is stored into B0. All YB stats are reset (Y4-Y7).

Data Flow at End of Cycle:

A – IC
B0 – Op 2 0000 (1010 0000)
B1 – rest of old IB
C – IC + 2
D – Op 1, Op 2, R1, R2
Skew buffer : Op 1
H – 47
J – 44
Y4-Y7 : off
The next block is 0001X, E6

Block E6

Function: Store updated IC into local storage. Local storage is addressed by H register which still contains address of IC. ALU function separates R1 and R2. R1 enters the skew buffer (Op 1 is now lost) R2 0000 is stored into B1. The YB stats are again reset.

Data Flow at End of Cycle:

A – IC
B0 – Op 2 0000
B1 – R2 0000 (1000 0000)
C – IC + 2
D – Op 1, Op 2, R1, R2
Skew buffer : R1
H – 47
J – 44
Y4-Y7 : off
If the main storage operation addressed an invalid storage area, a branch to the program check trap routine takes place (TRAP) and I-Fetch is abandoned, (only test for IMA since Y0 = Off).

Block E7

Function: The low-order halfword of general register R2 (register 8) is read out of local storage and set into C. Local storage is addressed with 1111 0001, F1 hex (low-order halfword of general register 8); this address is stored back to H. ALU function combines Op 2 and R1, R2 enters the skew buffer but is reset at the end of the cycle. Op 2 R1 is stored into B0.

Data Flow at End of Cycle:

A – IC
B0 – Op 2 R1 (10100111)
B1 – R2 0000
C – R2 low-order halfword
D – Op 1, Op 2, R1, R2
Skew buffer : 0
H – F1 (low-order halfword address of R2, general register 8)
J – 44
Y4-Y7 : off

Block E8

Function: Function branch on Op 2. The low-order halfword of R2 is written back to local storage (restored). Local storage is addressed with H and stored back decremented by 1; H now contains the address of the high-order halfword of R2 (high-order bits of general register 8). ALU function separates Op 2 and R1; Op 2 enters the skew buffer; R1 0000 is stored back to B0; IZT and IDQ are reset (Y14). They may be set on by ALU circuits in the previous cycles but for further operations they have to be off.

Data Flow at End of Cycle:

A – IC
B0 – R1 0000 (01110000)
B1 – R2 0000 (1000 0000)
C – R2 low-order halfword
D – Op 1, Op 2, R1, R2
Skew buffer : Op 2
H – F0 (high-order address of R2, general register 8)
J – 44 (address of system mask of PSW)
Y4-Y7 : off
IZT and IDQ : off
Next block is 1010X on CLD QH001 the first address of the RR fixed-point add routine.

Summary

For RR fixed-point operations, first halfword I-Fetch is left with the following essential information in the data flow:

1. Updated IC in local storage
2. Register numbers R1 R2 in B0 B1, respectively
3. Low-order halfword of R2 in C
4. Address of high-order halfword R2 in H
5. YB stats reset
6. ALU conditions reset

Other Microinstructions on Figure 54

Block C6

Similar block to E6; it is entered from the first-level function branch if the operation code specifies RR sequencing operations. For these operations no common operand has to be fetched from local storage. This block immediately performs the second-level function branch to enter the proper instruction routine.

Blocks L6, N6, Q6, S6

Branches from the first-level function branch for invalid operation codes lead to one of these blocks. The function of these blocks is identical. Together with N7, IC is again updated by 2 (according to the program specifications the IC points to the address of the invalid operation code +4).

Block Q8

Complete restoring of IC, testing for protected storage address (TRAP) and branch to the program check trap routine for invalid operation codes.

Blocks E5 and G5

Identical blocks to C4, are executed if entry to I-Fetch comes after pressing the start key. The PRI test in I-Fetch is ignored for the first instruction.

Machine Instructions — CLFC's

Machine operation during execution of individual machine instructions is defined by microprogram routines. All microprogram routines are represented on CLD's.

For I-Fetch and a few instructions, additional explanation is given in this section as examples of how to interpret flow charts.

The following microprogram routines are described:
First-level I-Fetch.
Second-level I-Fetch RX half and full fixed-point.
Second-level I-Fetch RX floating-point.
Second-level I-Fetch RS and SI operations.
Second-level I-Fetch SS logical operations.
Second-level I-Fetch SS decimal operations.
Branch and link.
Branch on count.
Branch on condition.
Convert to binary.
Convert to decimal.

Fixed-point halfword add and subtract.
Fixed-point multiply.
Floating-point multiply.
Shift instructions.

Instruction Fetch Microprogram

Figure 601 is the Instruction Fetch CLFC. The instruction fetch (I-Fetch) microprogram is entered when the instruction counter of the PSW from local storage 47 (hex) is transferred to the A register and a read main storage call is given in the microprogram. The read call causes main storage to be addressed by the A register contents and the information from main storage transfers to the D register.

The D register thus contains the first halfword of the next machine instruction to be executed. If this machine instruction is an RR type, the complete instruction is held in the D register. The instruction counter is updated by +2 and is transferred to the C register.

The microprogram next tests the PRI condition. If it is present, the program interrupt routine is entered which first writes back to main storage the contents of the D register.

With no PRI condition present the D register is set into the instruction buffer location of local storage (43 hex) and a function branch is called using the most-significant four bits of the operation code from register D0.

The most-significant four bits of register D0 are Op 1 and the next four bits, Op 2. Prior to the function branch, Op 1 is set into the skew buffer. This function branch gives a possible 16 exits of which four are invalid. The four invalid exits are the invalid operation codes:

A0 to AF
B0 to BF
C0 to CF
E0 to EF

Decoding one of the foregoing operation codes causes an invalid operation trap.

The remaining 12 valid exits define:

0	RR	Sequencing Operations
1	RR	Fixed-Point Operations
2	RR	Floating-Point Operations, Double Precision
3	RR	Floating-Point Operations, Single Precision
4	RX	Fixed-Point Operations, Half Word
5	RX	Fixed-Point Operations, Full Word
6	RX	Floating-Point Operations, Double Precision
7	RX	Floating-Point Operations, Single Precision
8	RS	Control and Shift Operations
9	SI	Logical and Input-Output Operations
D	SS	Logical Operations
F	SS	Decimal Operations

Note: if an RR instruction is decoded, the complete instruction is in the D register, but if an RX, RS or SI instruction is decoded, the second halfword of the instructions is read out from main storage to the D register. Similarly, decoding an ss instruction initiates two further main-storage read/write cycles in which the second and third halfwords of the instruction are read out.

The function branch that uses the four bits of Op 1 is called a "first-level function branch" while the function branch that follows uses the four bits of Op 2 and is called a "second-level function branch." The first-level function branch defines only the type of operation as shown in the previous table. The second-level function branch defines the specific instruction within the operation type.

For an RR instruction, a second-level function branch is called, using Op 2 from register B0. The exit from the function branch is to the particular microprogram defined by the operation code of the instruction. The instruction-fetch routine for an RR instruction is complete at this point and the execute phase begins.

For other instruction types (RX, RS, SI and SS), the complete instruction is first read out from main storage before calling a second-level function branch. Thus for all instructions when the I-Fetch routine is complete (after the second-level function branch) the full instruction is in the machine data flow and local storage before the execute phase begins

Second-Level Instruction Fetch: RX Half and Full Word Fixed Point

Figure 602 is the RX Fixed-Point Second-Level Instruction Fetch CLFC.

The purpose of this microprogram is to read out from main storage the second halfword of the RX instruction, form the address of operand 2 from the contents of X2, B2 and D2 and then execute a second-level function branch using the lower four bits of the operation code (Op 2). The contents of all registers after completion of the first-level function branch are shown in Figure 602.

Initially, stat Y7 is set to signify a halfword operation and the instruction counter (IC) is updated by a +2 and transferred to register C prior to storing in local storage 47 hex. R1 and X2 from register D1 transfer to the skew buffer and B1 register respectively before the next read call is given to read out the second halfword of the machine instruction.

Two tests now take place to determine if X2 and/or B2 are zero. If both are zero, the address of operand 2 is given by the 12 bits of D2. If X2 only is zero, operand 2 address is given by (B2) + D2; if B2 is zero, operand 2 address is given by (X2) + 42; if neither B2 or X2 is zero the operand 2 address is given by (X2) + (B2) + D2.

Note that (X2) or (B2) means the contents of these addresses in local storage.

The operand 2 address is checked for validity and, if invalid, the trap or program check interrupt routine is entered. With a valid operand 2 address, a function branch is called, using the lower four bits of the operation code from register C0 together with stat Y7 to determine either a half or full word operation. Thus, there are 32 possible exits from the second-level branch microinstruction as shown in Figure 602.

At the end of the I-Fetch routine, R1 is in register B0 and operand 2 address in the A register.

Second-Level Instruction Fetch: RX Floating Point

Figure 603 is the RX Floating-Point Second-Level Instruction Fetch CLFC.

The purpose of this microprogram is to form the effective address of operand 2 in register A, check it for validity and, if valid, preform a function branch on the lower four bits of the operation code.

On entry, stat Y6 is set to denote a double-precision operation or is reset to denote a single-precision operation. The updated IC transfers to local storage 47 hex and the second halfword of the instruction is read out from main storage to the D register.

A series of tests checks for X2 and B2 zero to determine the formation of operand 2 address. R1 is also checked to ensure that an even pair of general registers is specified. If R1 is odd, the program check interrupt routine is entered because of invalid address specification.

In a single-precision operation the least-significant two bits of operand 2 address are checked for zero. In a double-precision operation, the least-significant three bits of operand 2 address are checked for zero. If not zero, the program check interrupt routine is entered.

Finally, a function branch is called on the lower four bits of the operation code. The function branch gives a possible 16 exits of which seven are invalid.

Note that for the same instructions both single and double precision operations use the same exit to the execute phase. The distinction between single and double operations is then made during the execute phase for the particular operation.

Second-Level Instruction Fetch: RS and SI Operations

Figure 604 is the RS and SI Operations Second-Level Instruction Fetch CLFC.

The purpose of this second-level I-Fetch microprogram is to read out the second halfword of the machine instruction, to form the operand 2 address from B2 + D2 (RS format) and enter the execute phase by performing a function branch on the lower four bits of the operation code.

For an SI format machine instruction, the operand 1 address is formed by adding B1 + D1.

On entry to this routine, stat Y6 is set if the instruction format is RS.

The IC is incremented by +2 to give the next instruction address and stored into local storage 47 hex. A read call fetches B2, D2 (RS) or B1, D1 (SI) from storage. The contents of B2 or B1 are then read out from the general-purpose register area or local storage.

A four-way branch occurs next in the microprogram by testing Y6 (RS or SI) and QNZ (B2 or B1 = 0). With B1 or B2 equal to zero, the operand 2 address is formed from D1 (SI) or D2 (RS). If the contents of the base register are not zero these contents are added to the displacement to form the operand 2 address for an RS instruction or operand 1 for an SI instruction.

Finally, a function branch is called on the lower four bits of the operation code to provide 16 exits into the execute phase for the RS instruction.

At the completion of a second-level I-Fetch for an RS instruction, the operand 2 address (B2 + D2) is in the A register and the B0 register contains R1, R3. For an SI instruction, the operand 1 address (B1 + D1) is in the A register and the B0 register contains the immediate data I2.

Second-Level Instruction Fetch: SS Logical

Figure 605 is the SS Logical Second-Level Instruction Fetch CLFC.

This microprogram fetches the second and third halfwords of the instruction from main storage, forms the addresses of operand 1 and operand 2, and finally performs a function branch on the lower four bits of the operation code. The exits from the function branch are to the execute phases of the SS logical instructions.

Since the SS instruction consists of three halfwords, the IC is updated by +4 and restored into logical storage 47 hex. Thus, the IC now indicates the next sequential machine instruction.

Stat Y5 is set to signify a logical operation and read call fetches the second halfword of the instruction from main storage to register D. The four bits of B1 are checked for zero and the operand 1 address is formed in register C from the contents of B1 + D1 or D1.

Another read call transfers the third halfword of the instruction from main storage to register D. The four bits of B2 are checked for zero and the operand 2 address is formed in register A from the contents of (B2) + D2 or D2.

A function branch is called on the four lower bits of the operation code, and the execute phase is entered. At the completion of the second-level I-Fetch routine, the required information is contained in registers as follows: Operand 1 address is contained in register C. Operand 2 address is contained in register A. The L field (bits 8-15 of the instruction) is in register B0.

Second-Level Instruction Fetch: SS Decimal

Figure 606 is the ss Decimal Second-Level Instruction Fetch CLFC.

This microprogram fetches the second and third halfwords of the machine instruction from main storage, forms the addresses of operand 1 and operand 2 and then performs a function branch on the lower four bits of the operation code. In a decimal operation there are 16 possible exits from the second-level I-Fetch routine. Seven of these exits are invalid operations causing a program trap.

Entry is from the first-level function branch of the I-Fetch routine. Before the second halfword of the instruction is read out, the instruction counter is updated by +4 to the address of the next instruction and is stored into local storage 47 hex.

The four bits of B1 are checked for zero, and, if zero, the address in register A is updated by +2 and the third halfword of the instruction is read out from main storage. The four bits of L1 are added to the 12 bits of D1 to form the operand 1 address which is transferred to the C register.

If B1 is not zero, the contents of the general-purpose register specified are read out from local storage and added to the 12 bits of D1 in register B. The four bits of L1 are next added to the B register contents to form the operand 1 address in register C.

The contents of the general-purpose register specified by B2 are read out from local storage and the four bits of B2 are checked for zero. If zero, the general-purpose register contents are not used in forming the operand 2 address and, in this case, the four bits of L2 are added to the 12 bits of D2 to give the operand 2 address in register A.

If the four bits of B2 are not zero, the specified general-purpose register is read out and added to the 12 bits of D2 and the four bits of L2. This forms the operand 2 address which is set into the A register. The operand 1 address in C is stored in local storage 41 hex before the function branch on Op 2 is executed. Thus at the end of the operation:
Operand 1 address is in local storage 41 hex and register C.
Operand 2 address is in the A register.

Branch and Link

Figure 608 is the Branch and Link CLFC.

This machine instruction can be in either RR or RX format with operation 05 or 45 respectively.

RR Format

The rightmost 32 bits of the updated PSW are stored as link information in the general-purpose register specified by R1. The branch address or next instruction address is obtained from the contents of the general-purpose register specified by R2.

If R2 is zero, no branching takes place and the next instruction address is given by the instruction counter. However, the rightmost 32 bits of the PSW are always stored in the general register specified by R1.

RX Format

The rightmost 32 bits of the updated PSW are stored as link information in the general-purpose register specified by R1. The branch address is formed from the contents of X2 + B2 + D2 and replaces the next instruction address.

RR Format Operation

On entry, the instruction counter from local storage 47 hex is transferred to register C, and R2 is checked for zero. If R2 is zero, register C is transferred to register A and this is the next instruction address. With R2 non zero, the contents of the general register specified by R2 are read out from local storage and are set into the A register to form the next instruction address.

The lower halfword of the general register specified by R1 is cleared and the instruction counter from register C is stored there. A local storage read from location 46 hex to register D causes the ILC, CC and system mask to be set in D0 and the high IC is set into D1.

The ILC and CC are cleared from D0 and an ILC of 01 with the CC setting of Y2, Y3 replace the most significant four bits of register D0. Thus, D0 now contains the true ILC and CC together with the system mask and D1 contains the high IC. The high halfword of the general register specified by R1 is cleared and register D is stored at this location.

The general register specified by R1 now contains the rightmost 32 bits of the updated PSW. The A register contains the branch or next instruction address. This can be the contents of the general register specified by R2 or the IC from the PSW if R2 was zero.

A main storage read causes the A register to address main storage to obtain the next instruction. At the same time, the A register contents are incremented by two and are transferred to the C register. Thus the next I-Fetch routine is entered.

RX Format Operation

On entry to the branch and link execute phase of the microprogram the A register already contains the sum X2 + B2 + D2. The A register was set previously during the second-level I-Fetch microprogram and now represents the branch address.

The RX entry to the branch and link microprogram first transfers the IC to register C and then stores register C in the lower half-word of the general register specified by R1. The ILC is set to binary 10 to signify

a two-halfword instruction and the remainder of the microprogram is common with the RR operations described previously.

The ILC, CC and system mask are formed in register D0 and the high IC in D1. The contents of the D register are then stored into the high halfword of the general register specified by R1. The exit is to I-Fetch using the A register to address main storage and obtain the first halfword of the next instruction.

Branch on Count

Figure 612 is the Branch on Count. CLFC.

The branch on count instruction causes the contents of the general register specified by operand 1 to be algebraically reduced by one and, if the result is not zero, causes the IC to be replaced by the branch address.

The branch address is obtained from the general register specified by R2 in an RR format instruction or in an RX format instruction from X2 + B2 + D2. If the result of the subtraction is zero, the IC is not replaced and the normal instruction sequencing proceeds. Thus, if the contents of the general register specified by R1 have a value other than one, branching will take place unless R2 is zero.

The branch address is formed before the contents of the general register specified by R1 are checked for zero even though the branch address may not be required. The condition code is not affected and the subtraction proceeds as in fixed-point arithmetic with the sign and 31 bits of the general register participating in the operation.

In an RR operation, operand 2 (R2) is initially checked for zero and, if zero, the IC is transferred to register A. This ensures that the branch address, which is always held in register A, is in fact the IC and so no branching will take place even though the R1 contents are not zero after subtraction. When R2 is not zero, the branch address from the general register specified is set into register A.

In an RX operation, the branch address is in register A from the previous second-level I-Fetch routine. The two types of operation now use the same microprogram to complete the operation.

Operand 1 reads out from local storage into the B and D registers and one is subtracted from the least significant byte in B1 register. Each of the four bytes of operand 1 are checked for zero and if all are zero the IC is set into register A and the I-Fetch routine entered. This represents a no-branch situation because the original contents of operand 1 were one.

After subtraction, any bit in any of the four bytes of operand 1 signifies that a branch must occur. In this case, the branch address, already in register A, is used to address main storage to fetch the next instruction.

Branch on Condition

Figure 618 is the Branch on Condition CLFC.

The branch on condition instruction causes the IC to be replaced by the branch address when the condition code matches the four-bit mask in the R1 field of the instruction.

The branch address is obtained from the general register specified by R2 (RR format) or from X2 + B2 + D2 for an RX format instruction. Matching of the condition code with the four instruction bits of R1 occurs as shown in the following table:

CONDITION CODE	INSTRUCTION BIT NO. = 1
00	8
01	9
10	10
11	11

A match will occur and a branch take place when the condition code is 01 and the four-bit field of R1 is 0100. No branch can occur when all four mask bits of R1 are zero or when the R2 field of an RR instruction is zero. An unconditional branch occurs when all four mask bits of R1 are ones.

In an RR operation, the IC is transferred from local storage 47 hex to register A and R2 is checked for zero. If R2 is zero, the next instruction fetch routine is entered, using register A to read out the next sequential instruction. With a non-zero R2 field, the microprogram tests the condition code in stats Y2 and Y3. This results in a four-way branch and each leg from the branch tests the R1 mask against the emitted four bits that represent the condition code setting.

If the mask and the condition code do not match (result of the AND function is zero) the next sequential instruction is fetched from main storage. If the R1 mask and the condition code match, the branch address is read out to register A from the local storage location defined by R2. The next instruction fetch routine is then entered using the branch address.

In an RX operation, the A register, on entry, contains the branch address. This is replaced by the IC only if the result of the "R1 mask – condition code" match is zero.

Convert to Binary

Figure 610 is the Convert to Binary CLFC.

The main storage address defined by (X2) + (B2) + D2 contains a 64-bit double word in packed-decimal format. This double word is converted to a 31-bit-plus-sign binary word and is stored in the general register specified by R1. Any invalid decimal digit or invalid sign in the second operand causes a program check interruption.

A program check interruption is also initiated when a decimal number greater than 2,147,483,647 or less than −2,147,483,648 is converted. In this case the conversion is completed to fill the 32-bit general register and the program check interruption occurs with the interruption code set to nine (fixed-point divide).

The principle employed in the conversion is as follows. Multiply the most-significant decimal digit by ten and add the result to the next most-significant decimal digit. The partial result is again multiplied by ten and the next decimal digit added. This process is repeated until the last digit is detected.

The last digit (which is always a left digit because the right digit is the sign) is added to the result to form the final binary conversion. The times-ten multiplication is done by adding the two and eight multiples of the partial sum. The two's multiple is obtained by a one-place left shift. The eight's multiple is obtained by a one place right shift of the partial sum plus a four-place left shift obtained by skew.

This is equivalent to a three-place left shift or 'times eight.' The partial sum is contained in the D register and carries are propagated to register C. Thus, the final result is contained in registers C and D, register D containing the least-significant halfword.

The following example illustrates the principle of converting decimal to binary.

Example: Decimal number for conversion:

DIGIT 1	DIGIT 2	DIGIT 3		
0001	1001	0011 (193)		
				IN SKEW REGISTER
Digit 1:		0000	0001	
× 2		0000	0010	
RSH (÷ 2)		0000	0000	1000
Skew		0000	1000	
Add (× 2) + (× 8)		0000	1010	
Add digit 2		0000	1001	
Partial sum		0001	0011	
× 2		0010	0110	
RSH (÷ 2)		0000	1001	1000
Skew		1001	1000	
Add (× 2) + (× 8)		1011	1110	
Add digit 3		0000	0011	
		1100	0001	Final result

Note that right shift moves a significant low-order bit into the skew register so that the bit is not lost when skew is called.

Entry to the convert-to-binary microprogram is from the RX second-level I-Fetch routine. Initially the 16 digits of operand 2 are transferred from main storage to local storage locations 07 to 00. The most-significant two digits enter local storage 07 and the least significant digit alone enters 00.

Only two digits are stored in each local storage location with the exception of 00. Register J is used to read out each pair of digits when required and J=00 is used to signify the last digit. This is detected when the microprogram test "LS4" indicates that the lower four bits of LSAR are zero. If the decimal sign is negative, stat Y7 is set to enable the two's complement of the result to be formed upon completion of the operation.

Starting from location 07, the digit pairs are read out from local storage and checked for bit content to eliminate the leading decimal digit zeros. When the digit pair with significance is found, the search is halted and the left digit of the pair tested for zero.

If the left digit is zero, the right digit is set into the partial-sum register (D).

The loop is now entered in which register D is multiplied by ten and the next-significant digit is added to the sum. The multiplication is performed as previously explained by adding the two and eight multiples of the partial sum. Carries from register D are inserted in register C which becomes the high halfword of the result.

If the left digit is added to the partial sum, stat Y0 is set to indicate that the next digit to be added, after multiplication, is the right digit. Conversely, if the right digit is added, stat Y0 is reset to indicate "add left digit next."

After a left digit is added, and before the tens-multiply routine is entered, a test is made for the last digit. The last digit will always be a left digit, and is detected by the LS4 test when the last digit is read out from local storage with J = 00.

Thus, the final digit has been added to the partial sum and an exit is called. At this point, stat Y7 is checked to determine the original decimal sign. If it was negative, the result is two's complemented. The result transfers from registers C and D to the general register specified by R1 and if no overflow condition is present the next instruction-fetch routine is entered.

If either a 31-bit overflow or a 32-bit overflow has occurred during conversion, the program check interrupt routine is entered with the interruption code in the PSW signifying a fixed-point divide check (code 9).

Convert to Decimal

Figure 613 is the Convert to Decimal CLFC.

The binary 32-bit signed integer in the general register, specified by R1, is converted to a decimal 64-bit double word and stored at the main storage location of (X2) + (B2) + D2. The stored result is in packed-decimal format and consists of 15 decimal digits plus sign.

Since the maximum decimal value of a 31-bit binary field is 2,147,483,648, the maximum number of significant digits in the stored result is 10. The decimal sign occupies the rightmost four bits of the result and the leftmost 20 bits (5 digits) are made zero.

The following example illustrates the basic principle of binary to decimal conversion.

Example: Binary number for conversion:

0000 1001 0110 1011 (= 2,411 decimal)

Equivalent hexadecimal digit number:

0 9 6 B

Multiply by digit radix (16) and add product to next least significant digit	0
	(×) 16
	0
	(+) 9
	(×) 16
	144
	(+) 6
	150
	(×) 16
	2400
	(+) 11
Decimal conversion	2411

This decimal value appears in the machine as:

0010 0100 0001 0001

Since the machine operates with eight bits at a time, the radix used is 256 and not 16. Thus, to multiply a binary byte by 256 the byte is left shifted eight places. This is accomplished in the microprogram by binary adding the byte to itself eight times with carrys inserted into another, previously zeroed, register. Decimal conversion takes place in this second register in parallel with the binary doubling occurring in the first register.

This is accomplished by decimal doubling in the second register whenever the first register is binary doubled. Binary carries from the first register are inserted into the second where they have a decimal significance. Decimal doubling is accomplished by decimal adding the register to itself and any carry is inserted after doubling is completed.

Since the act of binary doubling clears the first register, carries out of the decimal register are inserted into the first register. Thus, after eight binary and decimal double operations, the original binary byte is converted to a four-digit decimal number. The following example illustrates this description.

Example: Transfer a number by binary doubling and convert by decimal doubling, inserting decimal carries into binary register.

Binary number for conversion:

1001 0110 (=150 decimal)

Registers:

	A DECIMAL DOUBLE TO CONVERT	B BINARY DOUBLE TO TRANSFER
Original register contents	0000 0000	1001 0110
First double	0000 0001	0010 1100
Second	0000 0010	0101 1000
Third	0000 0100	1011 0000
Fourth	0000 1001	0110 0000
Fifth	0001 1000	1100 0000
Sixth	0011 0111	1000 0000
Seventh	0111 0101	0000 0000
Eighth	0101 0000	0000 0001
	Decimal carry	

The true result is given by registers B, A; i.e. 0150.

For binary values of more than one byte, the process is repeated for the next most-significant binary byte using another register for the binary doubling, but using registers A and B with their contents (partial result) for decimal doubling.

There are four conversion loops in the microprogram for converting each binary byte of the 32-bit register specified by R1. However, all four loops are used only when the most-significant byte of R1 is not zero after the true value is obtained. The true value is the positive binary value of the 31 bits of the register specified by R1.

If operand 1 is negative, it is two's complemented to obtain the equivalent positive binary value. The leftmost byte is checked for zero and, if zero, the next byte is checked and so on, until a byte containing bits is discovered. Register H is set to reflect the number of byte conversions or loops required.

The first loop is entered in which the binary addition occurs in register C0 and the decimal addition in register B1. The decimal partial result on exit from this loop is contained in C0 and B1 register. The second loop (loop 3) is entered provided that register H is not zero.

In this loop, the binary adidtion is done in C1 register and the decimal addition in C0 and B1. The decimal partial result on exit is now in C1, C0, B1 registers. The third loop (loop 4) uses D0 as the binary addition register and the decimal addition occurs in C1, C0 and B1.

The final loop (loop 5) uses D1 as the binary register and on exit the ten-digit result is contained in D1, D0, C1, C0 and B1. On exit from loop 5 there is the possibility that D1 contains a binary ten (1010) owing to the conversion of a large binary value. Register D1 is decimally added to itself before the result is stored.

Therefore, in the singular case, when D1 is ten, the register is set directly to 0010 0000 (decimal 20) and the carry, if any, inserted. No decimal addition takes place prior to storing the result in this case. The binary value of D1 cannot be greater than 1010 and any value less than 1010 is decimal added to itself to give the true two most-significant decimal digits of the result.

The four-bit sign is inserted to the right of the result and the result is stored as D1, D0, C1, C0, B1, sign, in the main-storage location given by operand 2. The leftmost 20 bits of this location are made zero. When certain loops are not used in the conversion, their associated binary addition register (C1, D0, D1,) is made zero so that the result in main storage in these areas is zero.

Fixed-Point Halfword Add and Subtract

Figure 616 is the Fixed Point Halfword Add and Subtract CLFC.

The fixed-point halfword add and subtract instructions use the same microprogram. The difference in the two instructions occurs on initial entry when the appropriate indirect function is set into the function register. Thus, in a subtract instruction, the bytes are processed using the indirect function 'subtract Q.' Similarly in an add instruction, the bytes are processed using the indirect function 'add.'

The halfword operand 2 is added to or subtracted from the fullword operand 1 and the result replaces operand 1 in local storage. Before addition or subtraction, the halfword operand 2 is expanded to a full word by propagating the sign bit through the 16 high-order bits.

On initial entry, a constant of all ones is set into register C0 and is used later when operand 2 is negative and a resultant carry from the lower halfword addition or subtraction requires propagation into the high halfword. Stat Y6 is used to signify a negative operand 2 sign.

After processing byte 3 and byte 2 of operands 1 and 2, the microprogram branches one of four ways as determined by the indirect carry stat YCI and the sign of operand 2.

If there is no carry from the lower halfword addition or subtraction, the operand 1 high halfword is stored as the result in the high halfword. If there is a carry for a propagation to the high halfword, bytes 1 and 0 are processed with an all-zero operand 2 if positive sign, or an all-ones (constant) operand 2 if negative sign.

If the result of the lower halfword addition or subtraction produces no carry, a no overflow condition is possible. In this case, the sign and significance of the result are checked and the condition code set accordingly.

When a carry is propagated through the high halfword of the result, an overflow is possible. When an overflow occurs, the condition code is set to binary 11 and a program check interrupt occurs if the fixed-point overflow mask in the PSW is one. With no overflow present, the condition code is set according to the sign and significance of the result.

Fixed-Point Multiply

Figures 620, 621, and 622 are the Fixed-Point Multiply CLFC, associated notes, and loop detail respectively.

The fixed-point multiply instruction can be in RR or RX format. R1 specifies the even register of a pair of general registers in local storage. The odd register of the pair contains the 32-bit multiplicand. The multiplier is contained in the 32-bit general register specified by R2 or occupies a word or halfword in main storage location X2 + B2 + D2 (RX format). The 64-bit product occupies the pair of general registers specified by R1. Thus, the lower word of the product replaces the multiplicand and an overflow cannot occur.

The sign of the multiplicand is initially checked and, if negative, the two's complement of the multiplicand is formed and set into local storage. The two, three, and six multiples of the multiplicand are then formed and set into local storage. These multiples will all be in two's complement form if the original sign of the multiplicand is negative. Each multiple occupies two halfwords in local storage with addresses from 02 to 0F.

The lower halfword of the multiplier is transferred to register D and each hexadecimal digit of the multiplier is analyzed, using a function branch, to determine which of the multiples are to be added or subtracted in accordance with the table shown in Figure 621. The order of processing each hexadecimal digit is also shown in Figure 621.

In the table, the hexadecimal digits 8 to F are considered to produce a carry into the next higher hexadecimal digit. Therefore, when a hexadecimal digit is analyzed, the high-order bit of the hexadecimal digit to the right is checked and, if one, the multiples used are those in the 'carry in' column. Thus, a 32-way branch occurs in the microprogram for each of the eight hexadecimal digits analyzed.

The multiples, as decided by the multiplier digit, are then added or subtracted with skew (four-bit shift) for every digit in the multiplier. The sequence is the order of processing as shown in Figure 621. The partial product obtained after processing the first halfword of the multiplier is contained in register A and, in the pair of general registers, specified by R1.

The final product occupies the 64 bits of the pair of general registers in local storage specified by R1.

The sign bit of the multiplier is treated as a data bit. This ensures that a negative product, due to a negative multiplier and positive multiplicand, is produced automatically in two's complement form. After the result is obtained, the original multiplicand sign is tested and, if negative, the result is two's complemented to give the true result before storing in the product location.

Following are five examples to illustrate the multiply principle.

Example 1:

		DIGIT 2	DIGIT 1
Multiplicand	+ 7	0000	0111
Multiplier	+ 33	0010	0001

The multiples of the multiplicand are:

×1	0000	0111
×2	0000	1110
×3	0001	0101
×6	0010	1010

Reference to the table and the digit 2 of the multiplier shows that the ×2 multiple is to be added; that is 0000 1110 0000.

Siminarly, analysis of digit 1 of the multiplier shows that the ×1 multiple is to be added.

	0000	1110	0000		
+	0000	0000	0111		
Result	0000	1110	0111	=	+ 231

Example 2:

Multiplier = 2089
Multiplicand = 7001
Multiplier = 0000 1000 0010 1001
Multiplicand = 0001 1011 0101 1001
The multiples of the multiplicand are:

	×1 = 0000	0001	1011	0101	1001
(Left Shift ×1)	×2 = 0000	0011	0110	1011	0010
(Add ×1 + ×2)	×3 = 0000	0101	0010	0000	1011
(Left Shift ×3)	×6 = 0000	1010	0100	0001	0110

Analyze multiplier hexadecimal digits using table and sequence in Figure 621.

Hex Digit 2 +(×3) Shift 1 Hex digits.
Hex Digit 1 −(×1 and ×6), Shift 0 Hex digits.
Hex Digit 4 +(×1) Shift 3 Hex digits.
Hex Digit 3 −(×2 and ×6), Shift 2 Hex digits.

Start	0000	0000	0000	0000	0000	0000	0000	0000
+×3	0000	0000	0000	0101	0010	0000	1011	0000
−×1	1111	1111	1111	1111	1110	0100	1010	0111
−×6	1111	1111	1111	1111	0101	1011	1110	1010
+×1	0000	0001	1011	0101	1001	0000	0000	0000
−×2	1111	1111	1100	1001	0100	1110	0000	0000
−×6	1111	1111	0101	1011	1110	1010	0000	0000
	0000	0000	1101	1111	0010	1001	0100	0001

or 14,625,089

Example 3: This example has a negative multiplier and shows that handling the sign of the multipler in the same way as a data bit produces the correct negative result and sign.

		Digit 3	Digit 2	Digit 1
Multiplicand	+ 12	0000	0000	1100
Multiplier	− 15	1111	1111	0001

The multiples of the multiplicand are:

×1	0000	1100
×2	0001	1000
×3	0010	0100
×6	0100	1000

Analysis of multiplier digit 2 gives − (×1) Shift 1 Hex digit.
Analysis of multiplier digit 1 gives + (×1) Shift 0 Hex digit.

−(×1)	1111	1111	0100	0000		
+(×1)	0000	0000	0000	1100		
Result	1111	1111	0100	1100	=	− 180

Example 4:

Multiplicand	− 7	1111	1001
Multiplier	+ 6	0000	0110

Since the multiplicand is negative, it is two's complemented before the multiples are formed. The multiples of the multiplicand are:

×1	0000	0000	0111
×2	0000	0000	1110
×3	0000	0001	0101
×6	0000	0010	1010

Analysis of multiplier digit 2 gives no operation (all zeros).
Analysis of multiplier digit 1 gives + (×6).

No Op	0000	0000	0000
+(×6)	0000	0010	1010
Result	0000	0010	1010

This result is two's complemented since the original multiplicand is negative.
Final product: 1111 1101 0110 = − 42.

Example 5:

Multiplicand	− 7	1111	1001
Multiplier	− 6	1111	1010

The multiples of the multiplicand are the same as in Example 4.
Analysis of multiplier digit 2 gives: No operation (all zeros)
Analysis of multiplier digit 1 gives: − (×6)
Analysis of multiplier digit 4 gives: No operation
Analysis of multiplier digit 3 gives: No operation

No Op	0000	0000	0000
−(×6)	1111	1101	0110
Result	1111	1101	0110

Since the original multiplicand sign is negative the result is two's complemented.
Final product: 0000 0010 1010 = + 42.

When the multiplicand is the most negative value (-2^{31}), the principle of multiplication is different. The hexadecimal digits of the multiplier are not analyzed and the ×2, ×3, and ×6 multiples of the multiplicand are not formed. Only the ×1 multiple of the multiplicand is formed and complemented.

In this case, the multiplier is multiplied by 2^{31} which means shifting the multiplier 31 places left. The microprogram accomplishes this by shifting the multiplier one place right and setting the result into the 32 high-order bits of the product field.

The carry out from the right shift goes into the most-significant bit of the 32 low-order bits of the product field. The 64-bit result is then two's complemented to give the true product value.

Floating-Point Multiply

Figures 628 and 629 are the Floating-Point Multiply/Divide Initialization and Floating-Point Multiply CLFC's respectively.

The normalized product of the multiplier and multiplicand is set into the operand 1 location. The microprogram uses operand 1 as the multiplier and operand 2 as the multiplicand.

In double-precision operations, both the multiplier and multiplicand are double words. In this case, the product is formed in five halfword locations (the product field plus register A) and shifted right as the multiply operation progresses. Thus, the least-significant part of the product is lost and the result is contained in a 64-bit double word.

In single-precision operations the multiplier and multiplicand are single words and the product a double word, so that truncation of the product does not occur.

During the operation an exponent underflow or overflow can occur.

Exponent overflow occurs if the final product characteristic exceeds 127. The operation is completed and

a program interrupt occurs. The fraction is normalized and correct; the sign is correct; the characteristic is 128 smaller than the correct characteristic. If the final characteristic is brought within range because of normalization, the overflow exception does not occur for an intermediate product characteristic exceeding 127.

Exponent underflow occurs if the final product characteristic is less than zero. If the corresponding mask bit is one, a program interrupt occurs. The fraction is normalized and correct, the sign is correct, and the characteristic is 128 larger than the correct characteristic. If the corresponding mask bit is zero, the result is made a true zero. Underflow is not signalled when an operand's characteristic becomes less than zero during prenormalization, and the correct characteristic and fraction value are used in the multiplication.

Other conditions arising that can cause a program check interruption are: operation, addressing (RX format), and specification.

The microprogram examines the multiplicand fraction and, if the fraction is unnormalized, shifts the fraction left one hexadecimal digit at a time until it is normalized. The multiplicand characteristic is decreased by one for every four-bit left shift of the fraction. A zero multiplicand fraction is detected by an IZT test after shifting the whole multiplicand one hexadecimal digit left during normalization. In this case, the result is made a true zero before the next instruction is fetched.

When the multiplicand fraction is normalized and not zero, the sign of the result is formed by examining the multiplier and multiplicand signs. Like signs means a positive result; unlike signs means a negative result. Normalization of the multiplier fraction and adjustment of the characteristic now take place and again a zero fraction is detected by the IZT test resulting in a true zero result.

When both operands are normalized, their characteristics are added and then decreased by 64 to give the intermediate characteristic of the result. At this point, exponent underflow is detected, if present, causing an exponent underflow interrupt if PSW bit 38 is a one. If PSW bit 38 is a zero, a zero result is stored. When exponent underflow does not occur, the multiples of the multiplicand are formed. These are the one, two, three, and six multiples and are stored into the working area of local storage.

For single-precision operations, the high halfword of the multiplier is stored into the most-significant halfword result location in local storage. The multiplier high halfword consists of the eight-bit intermediate characteristic result and the eight high bits of the multiplier. The lower halfword of the multiplier transfers to register D and a 32-way branch is performed on the next to lowest hexadecimal digit.

This 32-way branch is executed using the four bits of the hexadecimal digit and the most-significant bit of the hexadecimal digit to the right. Each multiplier digit is analyzed in a similar manner to determine which multiples of the multiplicand are to be added or subtracted to form the product.

This is the same method as is used in fixed-point multiply and the microprogram in this area is common. That is, the microprogram for the analyzing of multiplier digits and adding and subtracting of multiples is common to both fixed and floating-point multiply.

In double-precision operations, the three most-significant halfwords of the multiplier are stored in local storage locations 40, 41, and 42 hex. The most-significant eight bits of local storage 40 contain the intermediate characteristic result. The storing of these three halfwords in 40 to 42 clears the floating-point register, enabling the product to be developed there. The multiplier lower half-word then transfers to register D and the multiply loop is entered by analysis of each multiplier digit in turn. This completes the initialization phase of floating-point multiply.

The partial product of the addition or subtraction of multiples is formed in register A and the result floating-point register. In a double-precision operation, the partial product is truncated as the multiplication of the fraction progresses so that the final product is a 64-bit double word.

Exit from the multiply loop occurs when all multiplier digits have been analyzed. This is detected when stats Y3 and Y6 are both on. Y6 indicates that the last (most-significant) multiplier halfword is being processed and Y3 indicates the last multiplier digit in the current halfword.

The product is checked for a normalized result by testing the most-significant fraction digit for zero. If zero, the intermediate characteristic is reduced by one and the entire 56-bit fraction shifted one hexadecimal digit left.

The four-bit left shift is accomplished in the microprogram by setting an 'add and skew' function in the function register and calling an indirect function when normalization is required. The final product, fraction characteristic and sign are then stored, in that order, into the floating-point register initially occupied by the multiplier (operand 1).

Shift Instructions

Figure 635 is the Shift Instructions CLFC.

There are eight shift instructions with operation codes from 88 to 8F which are all in RS format. They cover single-word, double-word, fixed-point arithmetic and logical shifts to the right or left.

Operand 1 of the instruction designates the word or double word to be shifted. The lower six bits of the operand 2 address (B2 + D2) are used as the binary value of the number of places to be shifted. At the completion of an arithmetic shift operation, the condition code is set before the instruction-fetch routine is entered. The condition code is not set in logical operations.

Shifting to the left causes zeros to fill the rightmost vacated bit positions and in an arithmetic operation an overflow occurs when a bit unlike the sign is shifted out of bit position 1. When shifting right, the low-order bits are lost and the original sign bits are filled in the leftmost vacated bit positions.

Since six bits specify the number of bit positions to be shifted, the maximum shift is 63 places and is used when shifting double words. When shifting a single word, the binary value of the six-bit count will not exceed 31. The presence of the most significant bit in the six-bit count defines a shift count between 32 and 63 places.

Similarly, the next most-significant bit in the six-bit count defines a shift count between 16 and 31 places. The four least-significant bits define a shift count between zero and 15. Thus, the microprogram initially tests the two high bits of the shift count to determine if 16 or a multiple of 16 places have to be shifted.

The shifting of 16 places at a time is done by a register transfer in loop 5 for a 16-bit left shift or in loop 2 for a 16-bit right shift. The six-bit shift count is then decremented by 16 and the two high bits are checked for zero. Further shifting and decrementing by 16 takes place until the two high bits of the shift count are zero.

Shifts of one to 15 places are executed using a maximum of two loops. The second loop, if required, is always a one-place shift to the left or right. Entry into the first loop is determined in the microprogram by a function branch using the lower four bits of the shift count.

The desired shift is obtained in the ALU by use of the indirect function register and setting this register to left shift, skew, skew and left shift, pass Q, or pass Q and skew, etc. A shift of eight places is obtained directly by moving the byte to another register while shifting.

In right-shift operations, the lower-four bits of the shift count are two's complemented before performing the function branch. This facilitates the testing of the lower two bits of the shift count to determine if a second loop is necessary to shift one further place.

Thus, for either a right or left shift the lower two bits of the shift count define whether a second loop is entered, and if entered, whether a one place left shift or a one place right shift is executed. The lower two bits of the shift count are set into stats Y1 and Y0; entry to the second loop being indicated when Y1 is one. When a shift operation is decoded, the stats Y4, Y5 and Y6 are set or reset as follows:

Y4 = 0 Right shift
Y4 = 1 Left shift
Y5 = 0 Single word shift
Y5 = 1 Double word shift
Y6 = 0 Logical shift
Y6 = 1 Arithmetic shift

SHIFT TABLE FOR 1-16 PLACES

SHIFT LEFT COUNT	SHIFT LEFT	SHIFT RIGHT
0000	0	0
0001	1 (1L)	15 (1L. 16R)
0010	2 (1L) + (1L)	14 (1L. 16R + (1L)
0011	3 (4L) + (1R)	13 (4L. 16R) + (1R)
0100	4 (4L)	12 (4L. 16R)
0101	5 (4L. 1L)	11 (4L. 1L. 16R)
0110	6 (4L. 1L) + (1L)	10 (4L. 1L. 16R) + (1L)
0111	7 (8L) + (1R)	9 (8R) + (1R)
1000	8 (8L)	8 (8R)
1001	9 (1L. 8L)	7 (1L. 8R)
1010	10 (1L. 8L) + (1L)	6 (1L. 8R) + (1L)
1011	11 (4L. 8L) + (1R)	5 (4L. 8R) + (1R)
1100	12 (4L. 8L)	4 (4L. 8R)
1101	13 (4L. 1L. 8L)	3 (4L. 1L. 8R)
1110	14 (4L. 1L. 8L) + (1L)	2 (4L. 1L. 8R) + (1L)
1111	15 (16L) + (1R)	1 (1R)

The preceding table shows how any shift, left or right, from 1 to 15 places is accomplished. The table also shows the significance of the lower two bits of the shift count in determining the entry to the second loop and also, if on entering the second loop, whether the one-place shift is to the left or right.

For example, when a right shift of 10 is decoded the first loop (loop 2) performs a left shift of 5 and a right shift of 16. This is accomplished by calling an indirect function and setting left shift in the function register (IL) and also calling 'Skew' to obtain the four-place left shift. The right shift of 16 is obtained by a register transfer.

Thus, on exit from the first loop an 11-place right shift is completed. The shift count contains 0110 and Y1 being on denotes that a second loop is necessary. Y0 being off defines the second loop as the one-place left-shift loop (loop 1). This is shown in Figure 635. Loop 2 and loop 1 are used for a right-shift-of-10 operation.

Exit from any one of the six loops is to the instruction-fetch routine. When an arithmetic shift operation has been performed, the condition code is set before the instruction-fetch routine is entered. Stat Y6 = 1 defines an arithmetic shift operation.

Ferrite Cores

- **Small, doughnut-shaped, ferromagnetic ring.**
- **High resistivity.**

A ferrite core is a small, doughnut-shaped ring of ferromagnetic material prepared from iron oxide, zinc, manganese, and nickel. These materials are ground to a fine powder, mixed with a binder, and pressed into shape. After it is fired in a kiln, the ferrite core is hard and brittle. The high resistivity of the ferrite material makes eddy-current losses and shielding requirements negligible, which results in efficient magnetic cores.

Magnetic Properties

- **Can be magnetized in either of two states.**
- **Polarity changed by current in drive lines.**
- **Output sensed when 1 is changing to 0.**
- **Change 0 to 1 by reversing direction of drive current.**

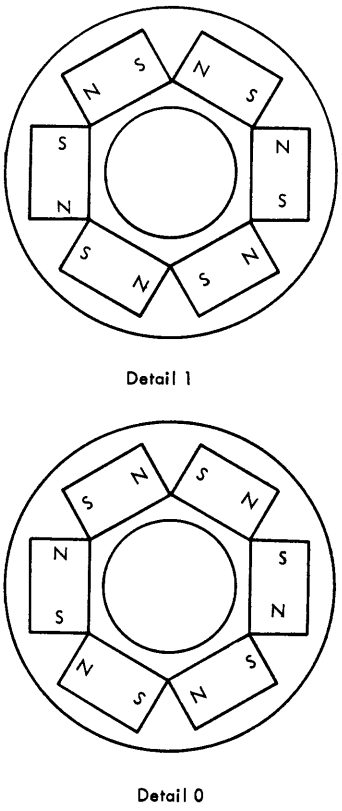


Figure 56. Two Stable States of Magnetic Core

The ferromagnetic character of the ferrite core permits it to be magnetized by any convenient magnetomotive force (mmf). After being magnetized, the core retains its magnetic polarity even though the mmf is removed. This makes the core useful as a storage device.

Electric current carried by a wire through the hole in the core can be used to generate the mmf. Thus, the polarity and strength of the mmf and the polarity of the core can be controlled by varying the direction and amplitude of the electric current.

Under static conditions (no current), the ferrite core is magnetized in one of two possible stable states. Details 1 and 0 in Figure 56 show a schematic representation of these two stable states In either state, the magnetic flux is contained entirely within the ferrite material.

The actions of a core changing state occur in sequence. Drive current through the horizontal wire builds up a magnetic field in the direction shown by the outer curved arrows in Figure 57.

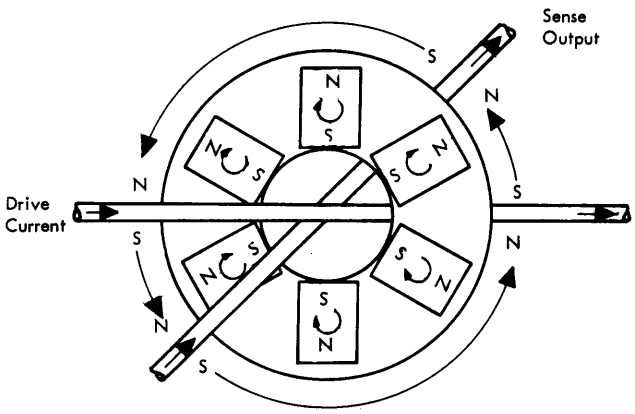


Figure 57. 1 changing to 0

When the current generates an opposing field of sufficient strength, the magnets break from each other and spin about as shown by the small curved arrows in Figure 57 to align themselves in the opposite direction. Once flipped, the magnets retain their new position. During the re-alignment of the magnetism in the core, magnetic lines of force extend beyond the ferrite material as shown schematically in Figure 58.

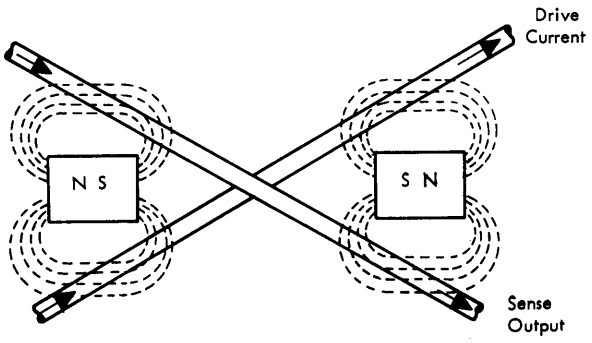


Figure 58. Core Field While Changing States

The external lines occur only while the magnets are changing direction; they recede into the ferrite when the change is complete. The lines of force, cutting the sense output line, enable the sensing of a reversal of core magnetic state. A certain field strength must be exceeded to cause the core to change state; neither a lower level nor a greater level has much effect.

Since this field strength or mmf is developed by an electric current, that current which is more than adequate to flip the core can be called I. I/2 is then not sufficient to flip the core. It can then be assumed that a current −I, can be made to flow in the opposite direction for 0 status in the core.

All the drive current through the core does not have to be carried by one wire. If two wires go through the core in the same direction, I/2 in each has the same effect as I in one wire, as the fields developed by the currents are added together algebraically.

Hysteresis Loop

The magnetic characteristics of ferrite material can be shown graphically by a hysteresis loop (Figure 59, Detail A). The curve represents the magnetic state of the material, plotted against mmf (H) on the horizontal axis, and flux direction and density (B) on the vertical axis.

As the current goes from 0 to −I peak, the magnetic state of the core follows the upper left portion of the loop, assuming an initial magnetic state of 1. Little change occurs as the minus current builds up to −I/2. However, the increase from −I/2 to −I completely switches the magnetizations to the 0 state. The core flips to 1 between +I/2 and +I as the current changes from 0 to peak to +I peak.

The principal features of core operation are:

1. Current in a direction that would magnetize a core to a specific state has little effect on the core magnetism if the core is already in that state.

2. One-half current in a direction that reverses the magnetic state of a core has little effect on its magnetism.

3. Full current in a direction that reverses the magnetic state of a core reverses its state.

4. Practical use of ferrite cores for information storage requires them to have a nearly square hysteresis loop.

The preceding description applies to a ferrite core having inherent characteristics desirable for information storage and operating under normal conditions. Abnormal conditions, particularly high temperatures, can distort the normal hysteresis loop so that the core is no longer an efficient storage device.

Internal heat reduces the core's ability to retain a specific state, causing the hysteresis loop to distort (Figure 59, Detail B).

In the sloped condition, a set core is unable to resist a half-select current and does not retain full magnetization when no drive current is flowing (no mmf present).

A core may become hot and lose its storage ability because temperature control on the core environment has been lost, or because one core location has been repeatedly driven from one to zero and back to one again many times in rapid succession. The latter condition is known as 'beating' a core location.

Defects in a core (cracks, breaks, voids) make it more temperature sensitive.

Controlling the Core

To be able to use a core as a storage device it must be possible to write either a one or a zero in a core and to read either a one or a zero out of a core.

Whether information is written into, or read out of a core is determined entirely by the direction of current in the X and Y wires that pass through the core.

If it is assumed that to read information out of a core both the X and Y wires must carry a current of −I/2 then to write information into a core the X and Y wires must carry a current of +I/2.

Reading Information out of a Core

To allow information to be read out of a core, a third wire is passed through the core. This wire is called the sense wire.

During the read operation the current through both the X and the Y wires is −I/2.

When the core was in the one state (a one stored in the core), it is flipped to the zero state (a zero stored in the core). This induces a pulse in the sense wire which indicates that a one was stored in the core.

When the core was in the zero state (a zero stored in the core), the −I/2 currents through the X and Y wires have little effect on its state. No significant pulse is induced in the sense wire which indicates that a zero was stored in the core.

Note that the core is always in the zero state after it is read out.

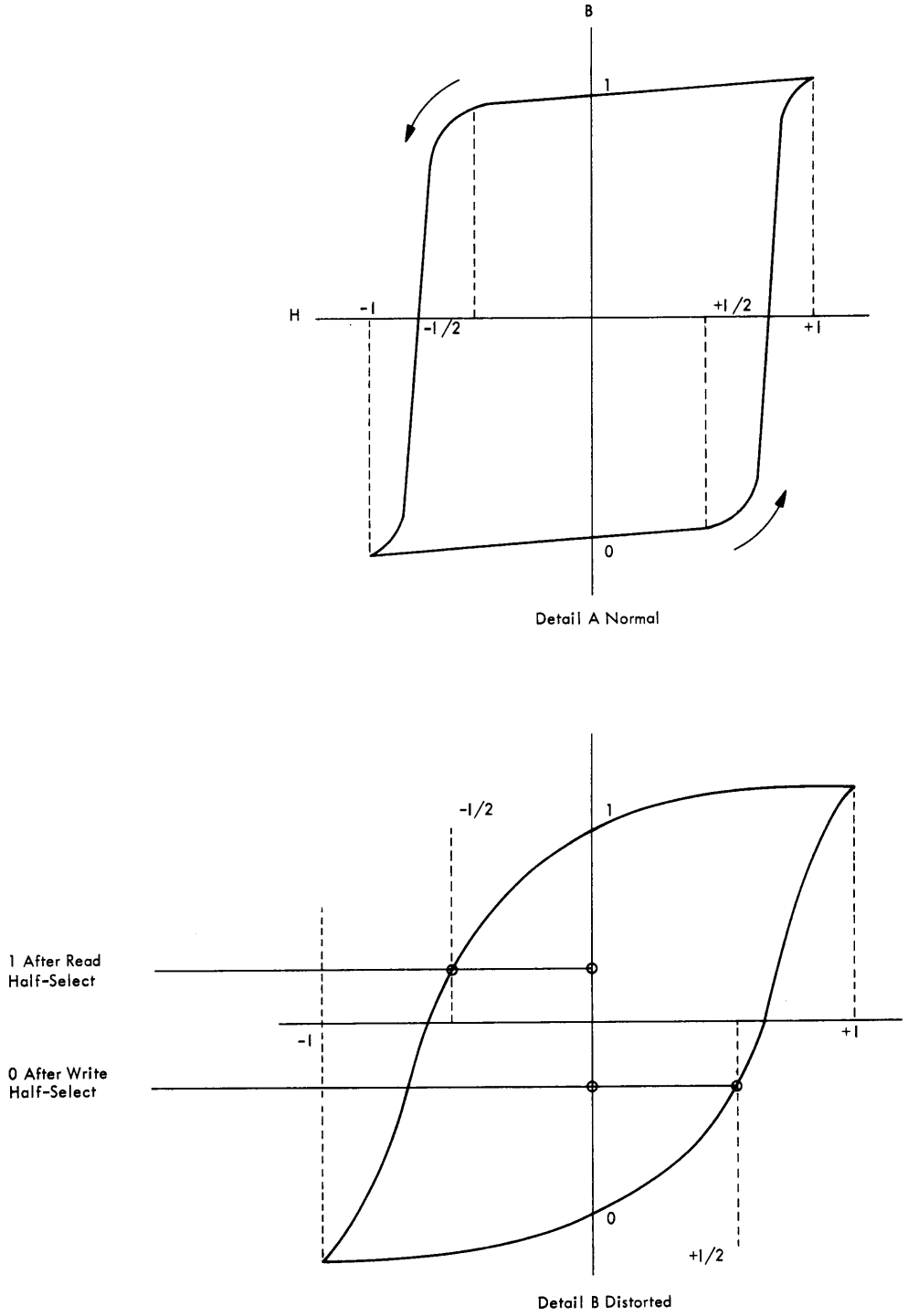


Figure 59. Normal and Distorted Hysteresis Loops

Writing Information Into a Core

To control the writing of a zero or a one, a fourth wire is passed through the core. This wire is parallel to the Y wire and is called the inhibit or Z wire.

Before a write operation, the core must be in the zero state, therefore, a write operation must always be preceded by a read operation.

During the write operation, the current through both the X and the Y wires is +I/2. This flips the core to the one state and a one is stored in the core at the end of the write operation.

To write a zero into a core, it must be prevented from flipping to the one state. This is done by sending a current of −I/2 through the inhibit (Z) wire which cancels the effect of one of the +I/2 currents. The remaining +I/2 current is not sufficient to flip the core to the one state, and a zero is stored in the core at the end of the operation.

During the write cycle, the flipping of the core to the one state induces a pulse in the sense wire but this sense pulse is disregarded.

The following table summarizes all the combinations of X, Y, and Z half select currents and their effects on the core.

	X	Y	Z	EFFECT
Read operation	−I/2	0	0	None
	0	−I/2	0	None
	−I/2	−I/2	0	Flip to zero state
	+I/2	0	0	None
	0	+I/2	0	None
	+I/2	+I/2	0	Flip to one state
Write operation				
	+I/2	0	−I/2	None
	0	+I/2	−I/2	None
	+I/2	+I/2	−I/2	None

Four and Three Wire Systems

To select and control a core, an X wire, a Y wire, an inhibit (Z) wire, and a sense wire are needed. The system of controlling the core with these four wires is called the four-wire system.

The main storage of the 2040 uses a three-wire system in which one wire performs the combined functions of sense during a read operation and inhibit during a write operation.

Coincident Current Addressing

- **Two drive lines, one in each dimension, select each core in a plane.**
- **Half current is carried by each selected drive line.**
- **A core is selected where two selected drive lines intersect.**
- **Half current does not flip cores.**
- **Coincidence of two half-currents flips cores.**

An 8 × 8 core plane, using the three-wire system as shown in Figure 60, illustrates coincident current addressing. This plane is capable of storing 64 bits of information. By pulsing the X and Y drive lines intersecting in a core, each with I/2, any core in the plane can be addressed.

On a read cycle, the status of the selected (dark) core is to be tested for the presence of a one (1) bit by pulsing drive lines X3 and Y3 each with a −I/2; these two lines intersect at the selected core. A one bit in the core flips to zero and induces a voltage in the sense winding. If the addressed core contains a zero, no significant change in its magnetic state occurs and very few magnetic lines of force cut the sense winding. The other 14 cores on drive lines X3 and Y3 each receive only −I/2 which is not sufficient to change their state.

On a write cycle, a one may be written into the selected core by pulsing the same two drive lines in the opposite direction (+I/2). If the inhibit winding carries no current, the X and Y currents (+I/2 each) cause the selected core to flip to a one. The zero state of the selected core can be retained on the write cycle by passing −I/2 through the sense/inhibit winding in a direction opposite to the +I/2 in the X drive line. This cancels the effect of the +I/2 in the X drive line and the +I/2 in the Y drive line is not sufficient to flip the core. The I/2 inhibit current is sent through the sense/inhibit line when a zero has to be set into the selected core on a write cycle.

To minimize unwanted sensing caused by the many half-selected cores (cores with only −I/2 or +I/2 flowing through them), the sense/inhibit winding is offset in such a way that an equal number of cores on any X line are on each leg. This reduces the effect of the X half-select noise by placing equal noise on each side of the sense amplifier.

To read or write a storage word (more than one core selected at the same time), the X and Y wires that are used to select that storage word intersect in all cores used for that word. Each core (bit) of the word has a separate sense and inhibit wire which it shares with the corresponding core (bit) in the other storage words.

The physical location of a storage word in an array depends on the winding of the X and Y wires through the planes in the array.

Split-Cycle Operation

Split-cycle operation means that there may be a gap of an indefinite period between a read and write cycle. For each storage word, a write cycle must always be preceded by a read cycle, but different storage words may be read out, one after another, and written back later in any sequence or number.

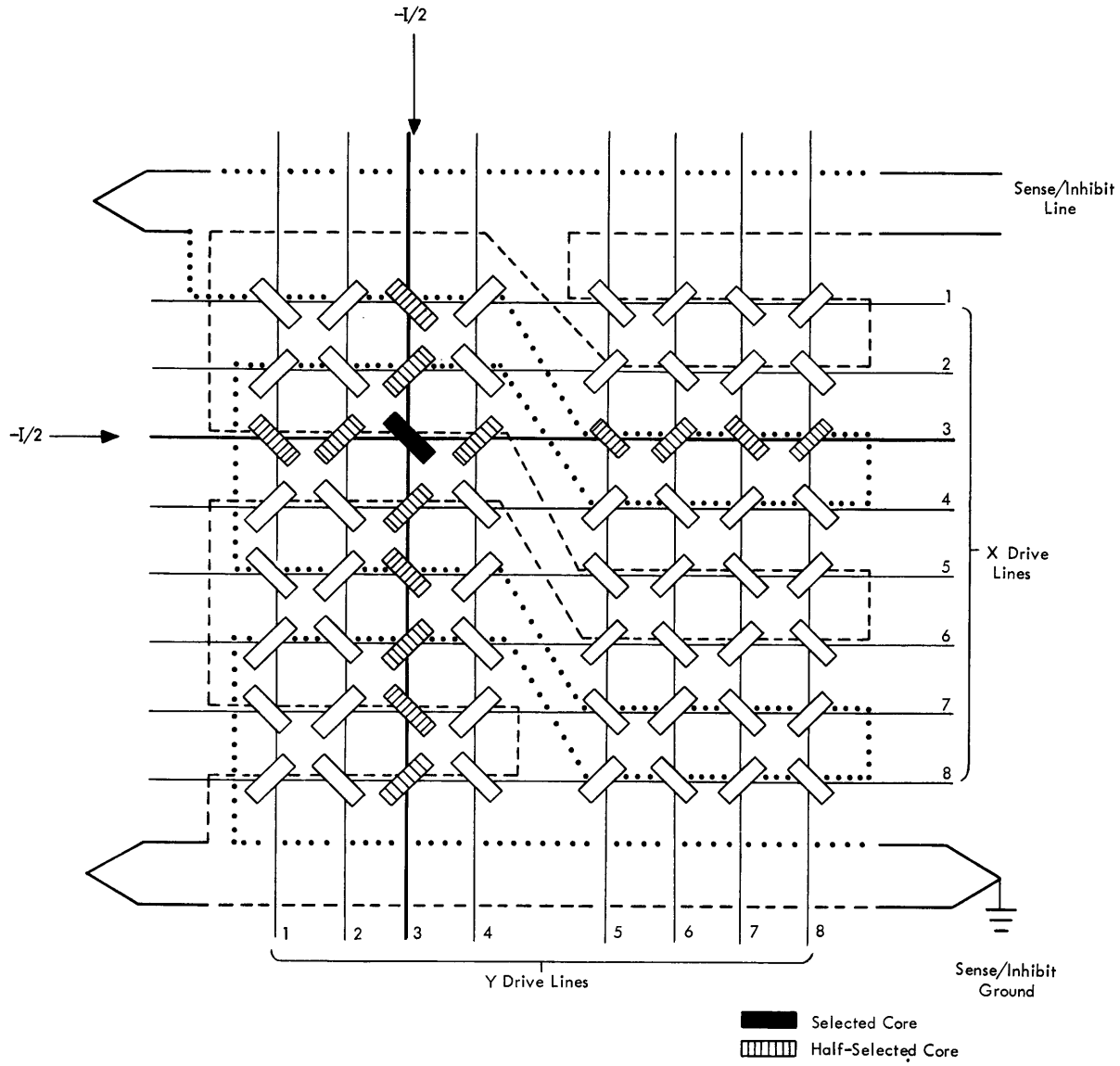


Figure 60. Coincident Current Addressing

Phase Reversal

Phase reversal is a system whereby two cores can be driven for each pair of drive lines in each plane by reversing the direction of flow or drive current in one of the drive lines. The advantage of phase reversal is that twice as many addresses in one dimension can be driven by a given number of drive circuits.

A simplified core plane of the type used in the storage unit of the IBM 2040 is shown in Figure 61.

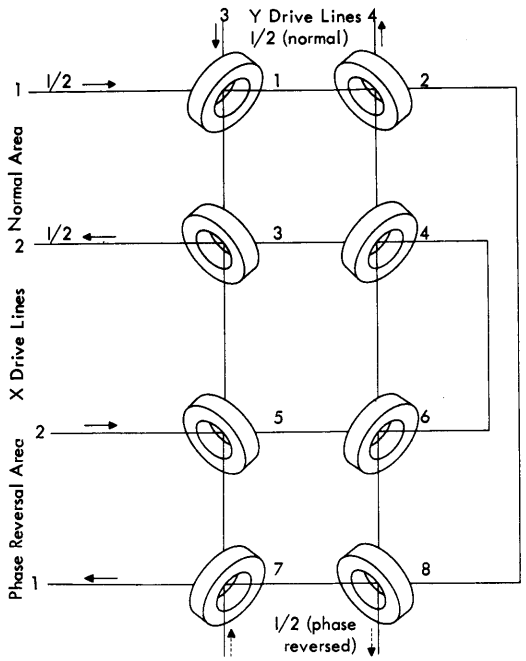


Figure 61. Cores Involved in Phase Reversal

X drive line 1 passes through cores 1, 2, 7 and 8, and Y drive line 3 passes through cores 1, 3, 5, and 7. If half-select currents (I/2) are pulsed on X drive line 1 and Y drive line 3 at the same time, in the direction shown by the solid arrows, they are in phase at core 1 and fully select it.

Cores 2, 3, 5, and 8 have only half-select current flowing through them and are unaffected by it.

Core 7 has both half-select currents flowing through it, but they are in opposite directions and, therefore cancel each other.

This is a method of selecting (addressing) one core out of the total matrix of eight. Similarly, core 2 can be selected by lines 1 and 4; core 3 by lines 2 and 3, and core 4 by lines 2 and 4 with half-select currents of the same phase as required for core 1.

However, in order to select cores 5, 6, 7 and 8, it is necessary to reverse the direction of one of the drive currents. With the Y drive current reversed (broken arrow) and the X drive current in the same phase as before (solid arrow), the drive currents are in phase and add at core 7 and are out of phase and cancel at core 1. Cores 5, 6, and 8 can be selected by similar phase reversal of the Y drive current.

If the direction of current that has been used to select these eight cores, one at a time, is called the read-direction current and the opposite direction, the write-direction current, it can be shown that these eight cores can again be selected, one by one, by driving both half-select currents in the opposite directions. In this way, any core in a plane can be selected for either a read or a write cycle.

External Machine Circuitry

Local Storage Address Register (LSAR)

- **Holds local storage address.**
- **Eight bits plus parity.**
- **Loaded under microprogram or logic control.**
- **Output to address loop via incrementer.**

LSAR contains the address for local storage. It consists of eight bits plus a parity bit. An address can be placed in LSAR from registers H or J, the emit field, Q register bits 0-3, and the console switches. An address can also be forced into LSAR by logic circuits.

LSAR can be read out via an incrementer (a four-position adder that adds ±1, −2 or 0 to the LSAR output) to registers H and J. The data path from these registers to LSAR and back is called the local storage address loop.

Local Storage Address Loop (Figure 62)

- **Address loop control in fixed sequence.**
- **Address loop is two half-bytes under separate control.**

The sequence in the address loop is as follows:

Load LSAR
Address local storage
Load one of the registers in the address loop from LSAR via the incrementer

The two half bytes of LSAR (bits 0-3 and bits 4-7) are controlled by separate circuits.

Only bits 4-7 can be read out of LSAR via the incrementer and any incrementer carry out of bit 4 is not propagated to bit 3.

Registers in the Address Loop

- **Registers H and J used to store local storage addresses.**
- **Registers H and J have a direct path to the 16-bit data flow.**

Registers H and J handle eight bits plus parity. Their purpose is to hold local storage addresses while forced addresses may be used in LSAR. An address can be used in LSAR, updated by the incrementer, and stored in the H or J register until it has to be used in LSAR again.

In addition, the H and J registers can be loaded from or set onto the R bus.

Incrementer

- **Used to increment LSAR bits 4-7.**

The incrementer modifies the local storage address by ±1, −2, or 0 during the transfer from LSAR to H or J. Only LSAR bits 4-7 pass through the incrementer; bits 0-3 bypass it. The carry from bit 4 is not propagated to bit 3; therefore any local storage starting address can specify a maximum field of 16 consecutive local storage addresses.

Address Loop Controls

- **Control by microprogram (CH field) or logic circuits.**
- **Three types of control:**
  1. **Source control.**
  2. **Incrementer control.**
  3. **Destination control.**

Control of the address loop can be either by microprogram or by logic circuits, depending on the operations. The address loop is normally under microprogram control. The control is the CH field. The CH field contains five bits, therefore, 32 different bit configurations are possible.

Control of the data flow from and into LSAR is divided into three groups:

1. Source control controls the loading of LSAR.
2. Incrementer control causes the incrementer to perform a ±1, −2, or 0 modification of the local storage address during transfer from LSAR.
3. Destination control; controls the destination of LSAR to H or J Register.

Under microprogram control, the decoding of the CH field brings up signals for source, destination, and incrementer controls.

Figure 63 shows for each address loop transfer:

1. The value in the CH field from which the control signals are derived.
2. The control line name for source, destination, and increment controls and the ALD page where the CH decoding circuits are to be found.
3. The TROS control field mnemonic.

Note: The TROS control field mnemonics do not correspond with the line names in the ALD's.

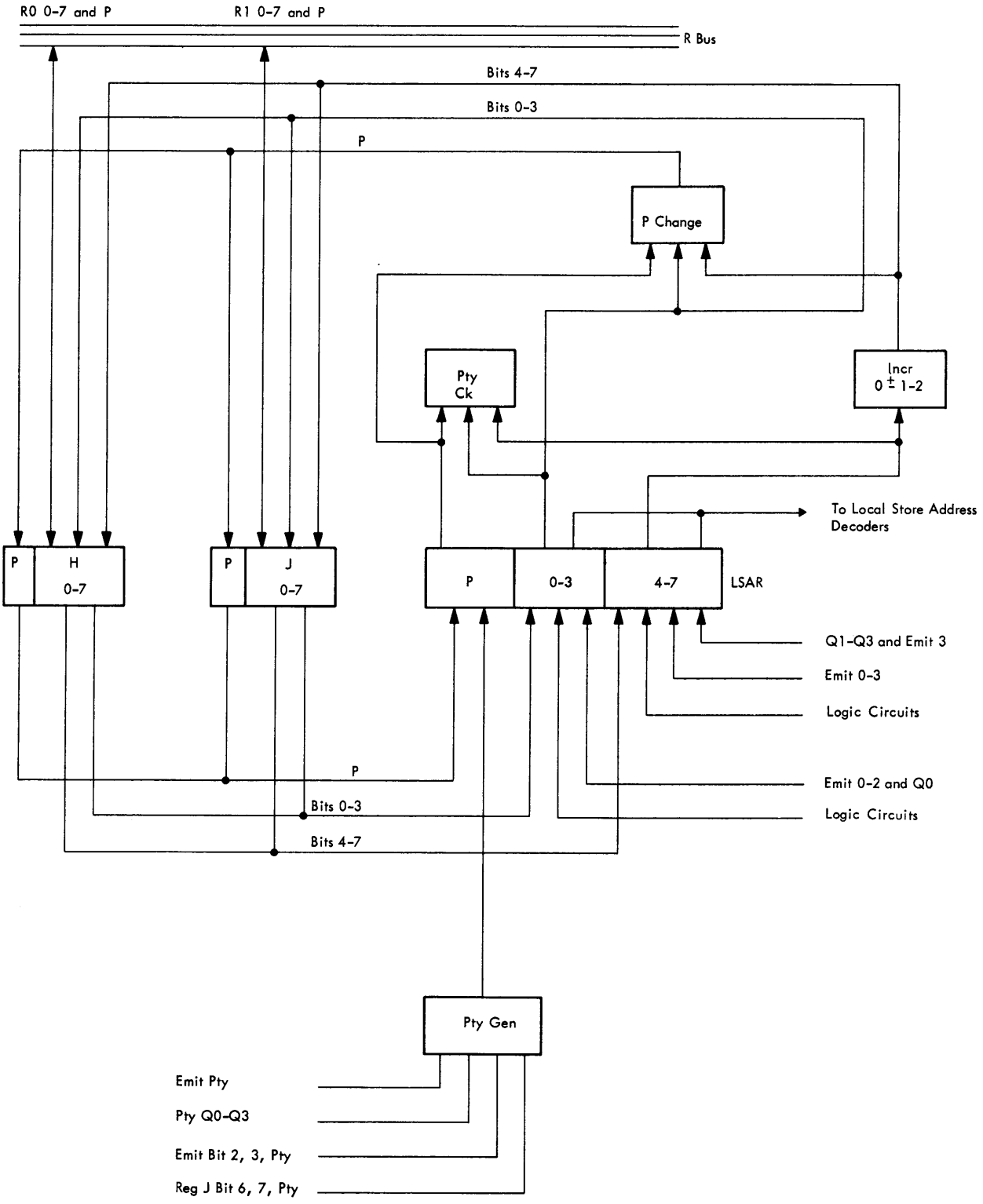


Figure 62. Local Storage Address Loop

	Control Field CH	CONTROL LINE NAME						ROS MNEMONIC (Control Signal Specs)	
		Source Control		Increment Control		Destination Control			
			ALD Page Number		ALD Page Number		ALD Page Number		
0	00000	LSAR Load LB Bits 0-3 LSAR Load LA or LB Bits 4-7	DS021 DS021		CC001		DS051	BE→L	
1	00001							*	
2	00010	LSAR Load H Bits 0-3 Plus Parity LSAR Load H Bits 4-7	DS021 DS021			LSAR To Register H		H→L	L→H
3	00011							*	
4	00100	LSAR Load LA Bits 0-3 LSAR Load LA or LB Bits 4-7	DS031 DS021			LSAR To Register J		AE→L	L→J
5	00101	LSAR Load LB Bits 0-3 LSAR Load LA or LB Bits 4-7	DS021 DS021			LSAR To Register J		BE→L	L→J
6	00110	LSAR Load LQ Bits 0-3 Plus Parity LSAR Load LQ Bits 4-7	DS031 DS031			LSAR To Register J		QE→L	L→J
7	00111	LSAR Load J or LJ Bits 0-3 LSAR Load J Bits 4-7	DS031 DS031			LSAR To Register J		J→L	L→J
8	01000	LSAR Load LB Bits 0-3 LSAR Load LA or LB Bits 4-7 H Field Bit 1 to B Gate	DS021	Increment - 1	CC001			BE→L	INT
9	01001			Increment - 1				*	
10	01010	LSAR Load H Bits 0-3 Plus Parity LSAR Load Bits 4-7	DS021	Increment - 1		LSAR To Register H		H→L	L - 1 → H
11	01011			Increment - 1				*	
12	01100	LSAR Load LA Bits 0-3 LSAR Load LA or LB Bits 4-7	DS031 DS021	Increment - 1		LSAR To Register J		AE→L	L - 1 → J
13	01101	LSAR Load LB Bits 0-3 LSAR Load LA or LB Bits 4-7	DS021 DS021	Increment - 1		LSAR To Register J		BE→L	L - 1 → J
14	01110	LSAR Load LQ Bits 0-3 Plus Parity LSAR Load LQ Bits 4-7	DS031 DS031	Increment - 1		LSAR To Register J		QE→L	L - 1 → J
15	01111	LSAR Load J or LJ Bits 0-3 LSAR Load J Bits 4-7	DS031 DS031	Increment - 1		LSAR To Register J		J→L	L - 1 → J
16	10000	LSAR Load LB Bits 0-3 LSAR Load LA or LB Bits 4-7 Reset Reinterpret Latch	DS021	Increment + 1	CC001			BE→L	REST
17	10001			Increment + 1				*	
18	10010	LSAR Load H Bits 0-3 Plus Parity LSAR Load H Bits 4-7	DS021	Increment + 1		LSAR To Register H		H→L	L + 1 → H
19	10011			Increment + 1				*	
20	10100	LSAR Load LA Bits 0-3 LSAR Load LA or LB Bits 4-7	DS031 DS021	Increment + 1		LSAR To Register J		AE→L	L + 1 → J
21	10101	LSAR Load LB Bits 0-3 LSAR Load LA or LB Bits 4-7	DS021 DS021	Increment + 1		LSAR To Register J		BE→L	L + 1 → J
22	10110	LSAR Load LQ Bits 0-3 Plus Parity LSAR Load LQ Bits 4-7	DS031 DS031	Increment + 1		LSAR To Register J		QE→L	L + 1 → J
23	10111	LSAR Load J or LJ Bits 0-3 LSAR Load J Bits 4-7	DS031 DS031	Increment + 1		LSAR To Register J		J→L	L + 1 → J
24	11000	LSAR Load J or LJ Bits 0-3 LSAR Load LJ Bits 4-7	DS031 DS041		CC001	LSAR To Register J		JE→L	L → J
25	11001	LSAR Load J or LJ Bits 0-3 LSAR Load LJ Bits 4-7	DS031 DS041					JE→L	
26	11010	LSAR Load LA Bits 0-3 LSAR Load LA or LB Bits 4-7	DS031 DS021					AE→L	
27	11011							*	
28	11100	LSAR Load LA Bits 0-3 LSAR Load LA or LB Bits 4-7	DS031 DS021			LSAR To Register H		AE→L	L→H
29	11101	LSAR Load LB Bits 0-3 LSAR Load LA or LB Bits 4-7	DS021 DS021			LSAR To Register H		BE→L	L→H
30	11110	LSAR Load LQ Bits 0-3 Plus Parity LSAR Load LQ Bits 4-7'	DS031 DS031			LSAR To Register H		QE→L	L→H
31	11111	LSAR Load J or LJ Bits 0-3 LSAR Load J Bits 4-7	DS031 DS031	Increment - 2	CC001	LSAR To Register J	DS051	J→L	L - 2 → J

* Not Used.

Figure 63. CH Field Control Chart

Example Source Code:

```
ALD        TROS CONTROL FIELD
LA  =          AE
LB  =          BE
LQ  =          QE
LJ             JE
```

Under control of logic circuits, source, destination, and increment are determined by control lines coming directly from the CPU circuits.

Set and Reset of LSAR

- **When the T clock is running (microinstructions are being executed), LSAR is reset at T1 in every machine cycle.**
- **Set of LSAR is controlled by the source control lines of CH field decoding or by logic circuits.**

In normal machine operations with the T clock running, LSAR is reset at the beginning of every machine cycle at T1. During logic cycles and manual operations, LSAR is reset by various conditions (Figure 64).

The setting of LSAR is controlled by the source control signals developed by CH field decoding as in Figure 63.

The possible source formats are shown in Figure 65.

The four high-order bits of LSAR (bits 0-3) and the four low-order bits of LSAR (bits 4-7) are controlled independently.

LSAR parity is generated according to the source specification.

Logic-controlled LSAR formats are also shown in Figure 65. For dump, or in certain log out cycles, fixed values are forced into LSAR. Any address can be entered from the console address switches.

Simplified circuits for loading LSAR are shown in Figure 64.

Example: The CH field equals 21 (10101); emit field in this microinstruction = 8 (1000).

Figure 63 gives the source control lines as follows:

LSAR load LB bits 0-3

LSAR load LA or LB bits 4-7,

Consider the machine in I/O state with the selector channel 2 microprogram being performed.

Figure 65 gives the following address loading in LSAR.

0 0 1 1 1 0 0 0

Figure 63 gives for CH = 21 an increment control of +1 and the J register as the destination control. As a result, at the end of this microinstruction the J register will contain 00111001.

Incrementer

- **Incrementing is controlled by CH field bits 0 and 1.**
- **LSAR bits 4-7 are fed through the incrementer.**

The incrementer is under control of CH field bits 0 and 1. These bits are decoded to give the following control lines:

CH field bits 0	1	
0	0	no line, increment 0
0	1	increment −1
1	0	increment +1
1	1	no line, increment 0.

A CH field of all ones (11111) produces the control signal increment −2. Figure 66 shows the incrementer and incrementer control circuits.

LSAR bits 4-7 are fed through the incrementer and modified according to the control lines active.

A possible carry out of bit 4 is not propagated to bit 3.

After incrementing, LSAR parity is corrected by the circuit shown in Figure 67.

LSAR Destination

- **Destination of LSAR is controlled by the destination control lines of CH field decoding or by logic currents (Figure 68).**

The destination of LSAR can be either the H or J register, or no destination may be specified.

Figure 63 shows the destination control lines developed for the various CH field values and also indicates the ALD page where the circuits can be found.

Figure 68 shows the set and reset of the H and J registers. These registers can also be set from the R bus.

Manual setting of H and J is via LSAR. The destination control includes these logic signals.

During microprogram log out, LSAR logic is gated to the H register by logic circuits.

Address Loop Checking

- **LSAR checked for odd parity.**
- **Parity of registers H and J loaded directly into LSAR.**
- **CH decoders are checked for source, increment, and destination decoding errors.**

Odd parity is maintained in LSAR for all conditions where the timing ring is running and also for displays and store. This means that parity is generated for all cases except when loading from the H or J registers, where the parity of LSAR is also loaded from the registers.

The parity in the local storage address loop is checked at the output of LSAR. This means that a parity error in any of the registers in the address loop will be detected in LSAR. The circuit for this is shown in Figure 67.

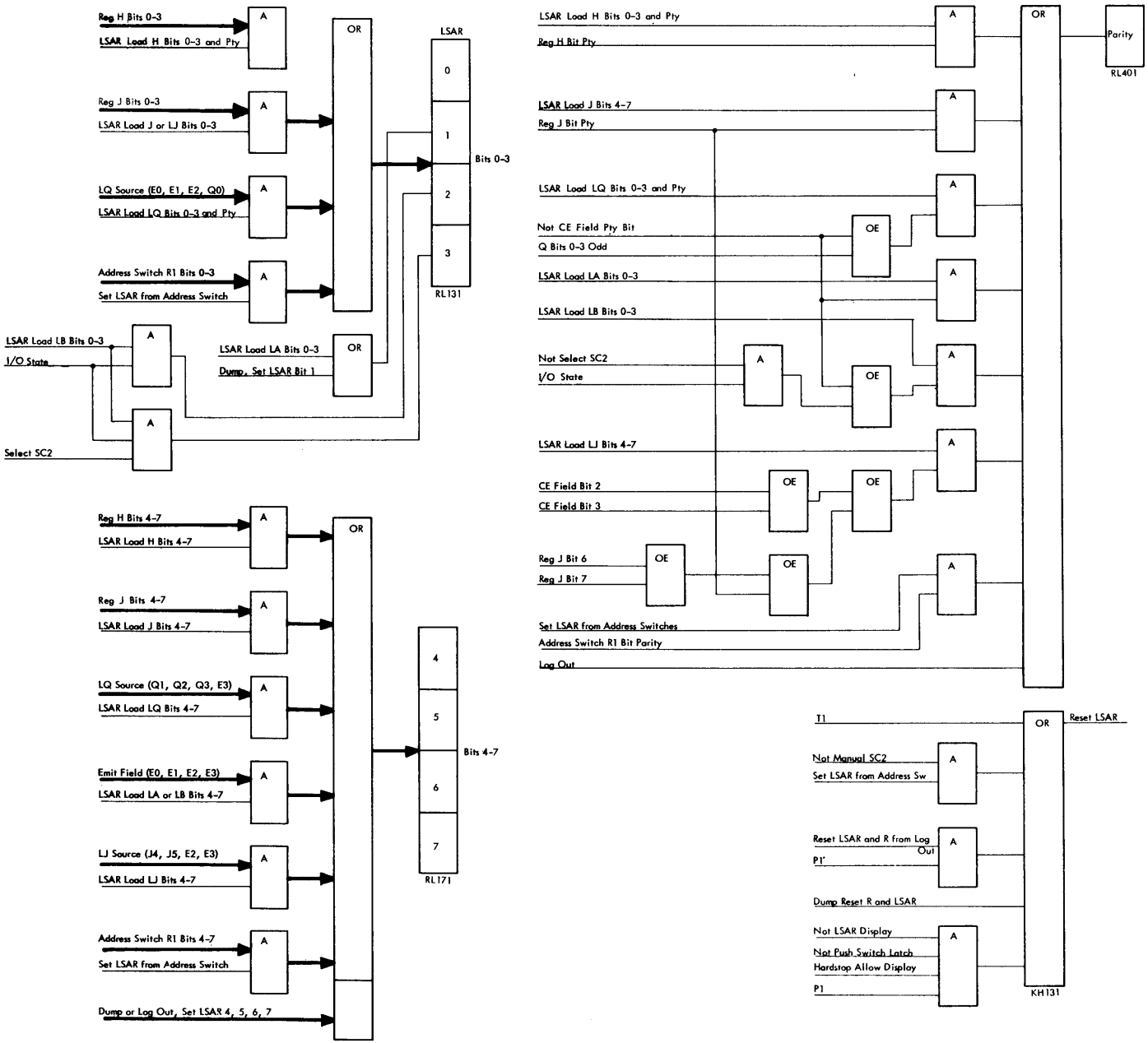


Figure 64. Set and Reset LSAR

Source Code	0	1	2	3	4	5	6	7
H	H0	H1	H2	H3	H4	H5	H6	H7
J	J0	J1	J2	J3	J4	J5	J6	J7
LA (=AE)	0	1	0	0	E0	E1	E2	E3
LB (=BE) (When in CPU State)	0	0	0	0	E0	E1	E2	E3
LB (=BE) (When in I/O State Mpx or SC1)	0	0	1	0	E0	E1	E2	E3
LB (=BE) (When in I/O State, SC2)	0	0	1	1	E0	E1	E2	E3
LQ (=QE)	E0	E1	E2	Q0	Q1	Q2	Q3	E3
LJ (=JE)	J0	J1	J2	J3	J4	J5	E2	E3
Logic Circuit Forced Addresses								
Dump	0	1	0	0	1	1	1	1
Log-out	0	0	0	0	1	1	1	1
Address Switches	X	X	X	X	X	X	X	X

H = H register bits
J = J register bits
E = Emit field bits
Q = Q bus bits

Figure 65. LSAR Bits

Decoder

- **CH field decoder checked in three parts:**
  1. **Source control check.**
  2. **Destination control check.**
  3. **Incrementer control check.**

Source Control Line

Source control lines are exclusive OR'ed together. An even result indicates a decoder error and sets the H-field decode load LSAR error latch. (ALD DS081.)

The output of this latch sets the control check latch.

Destination Control Line

The 'CH field bits 234' decoder outputs that are not used to generate a destination control signal are OR'ed together to set the destination 0 latch. The output of this latch is exclusive OR'ed with the destination control lines. An even result sets the LSAR destination check latch (ALD DS061). Setting the LSAR destination check latch results in a control check.

Incrementer Control Line

The three incrementer control lines are tested for one line active or no lines active. See Figure 66. If two lines are active an error is indicated and the control check latch is set.

Timing

Over-all timing of the local storage address loop is shown in Figure 69.

Local Storage Data Register

- **R register is LS data register.**
- **LS read/write is controlled by CJ and CL fields.**

The data register for local storage is the R register. The R register control fields are used to call local storage read or write. With the CJ field = 6, the R bus source is local storage. With the same control, local storage is called to read.

With the CL field = 1, the R bus destination is local storage. The same control also initiates a local storage write cycle.

In addition to the normal parity checking, R register parity is sampled at the end of a local storage read cycle at time P4.

In the case of a local storage read parity check, the T clock is stopped at the end of the next machine cycle, during which the early check latch is set (because of the later checking time, it is not possible to stop in the same cycle).

The contents of the R register and the address in LSAR are lost (both registers are reset in every machine cycle), but the LS read parity check is latched up and indicated on the console.

Note: Because of close timing conditions, local storage read is not checked in the last microinstruction prior to a dump cycle.

Local Storage Unit

- **Small high-speed core-storage unit.**
- **Four-wire system.**
- **Size: 144 words of 24 bits.**
- **Split-cycle operation.**
- **Cycle time of 625 nanoseconds for read or write.**
- **Access time of 350 nanoseconds.**
- **Two core planes mounted on pluggable cards.**
- **Local storage is temperature-compensated.**

Local storage (LS) is a small, high-speed, four-wire, core-storage unit used to provide fixed-point registers, floating-point registers, temporary storage for channel operations, CPU working space during error conditions and I/O microprogram interrupts, and general working storage. Only the fixed-point and floating-point stor-

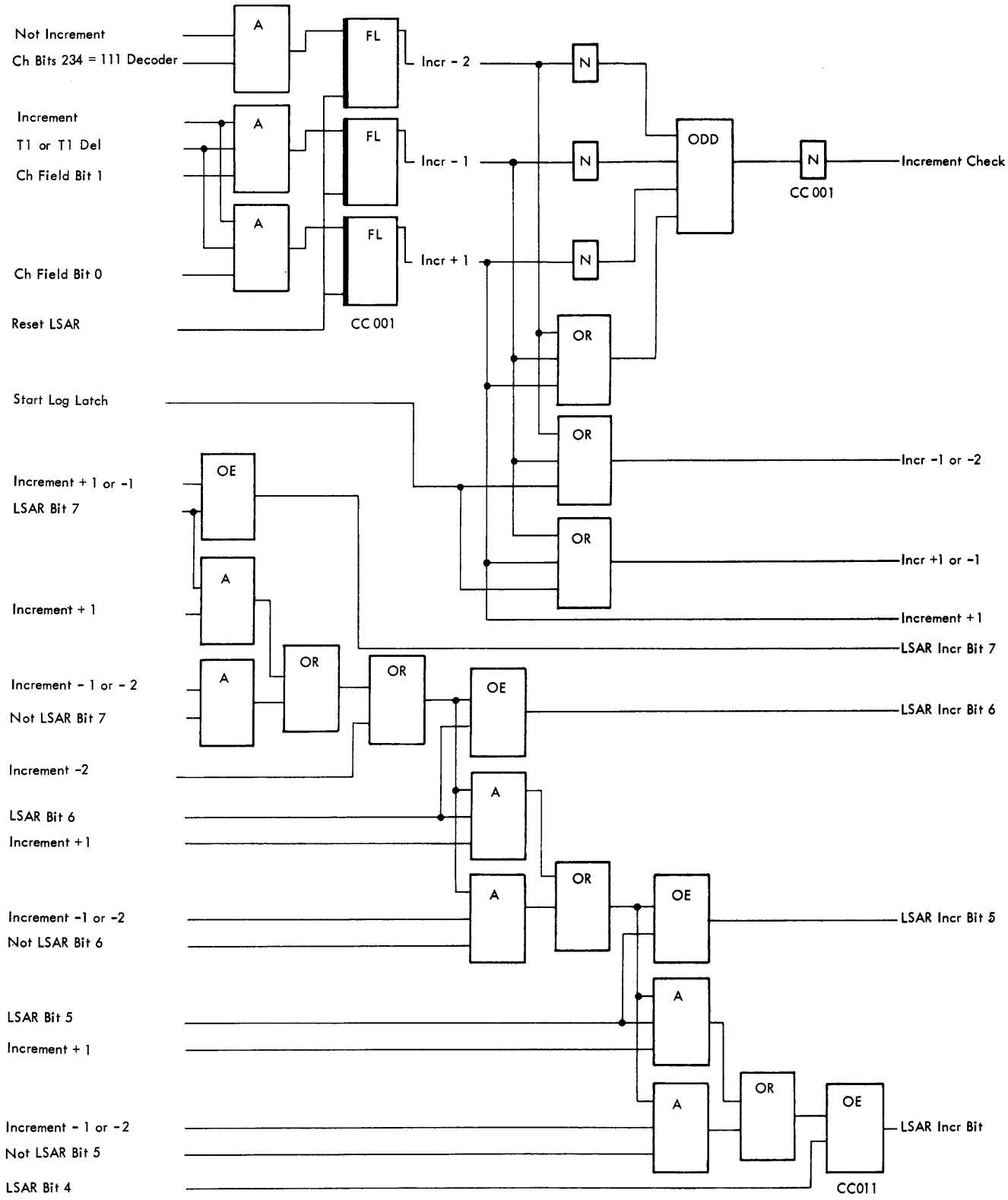


Figure 66. Incrementer and Incrementer Control

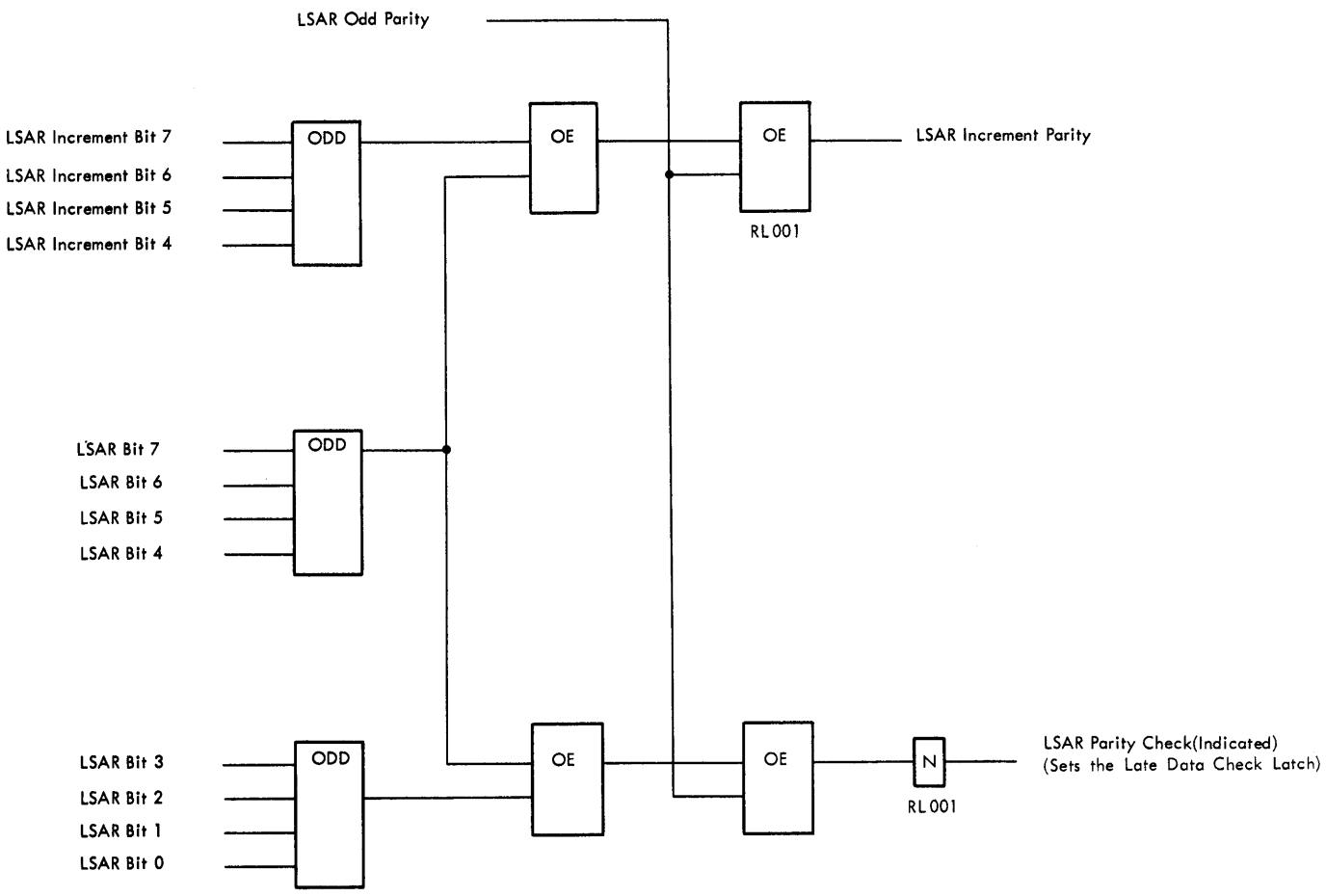


Figure 67. Parity Check/Change

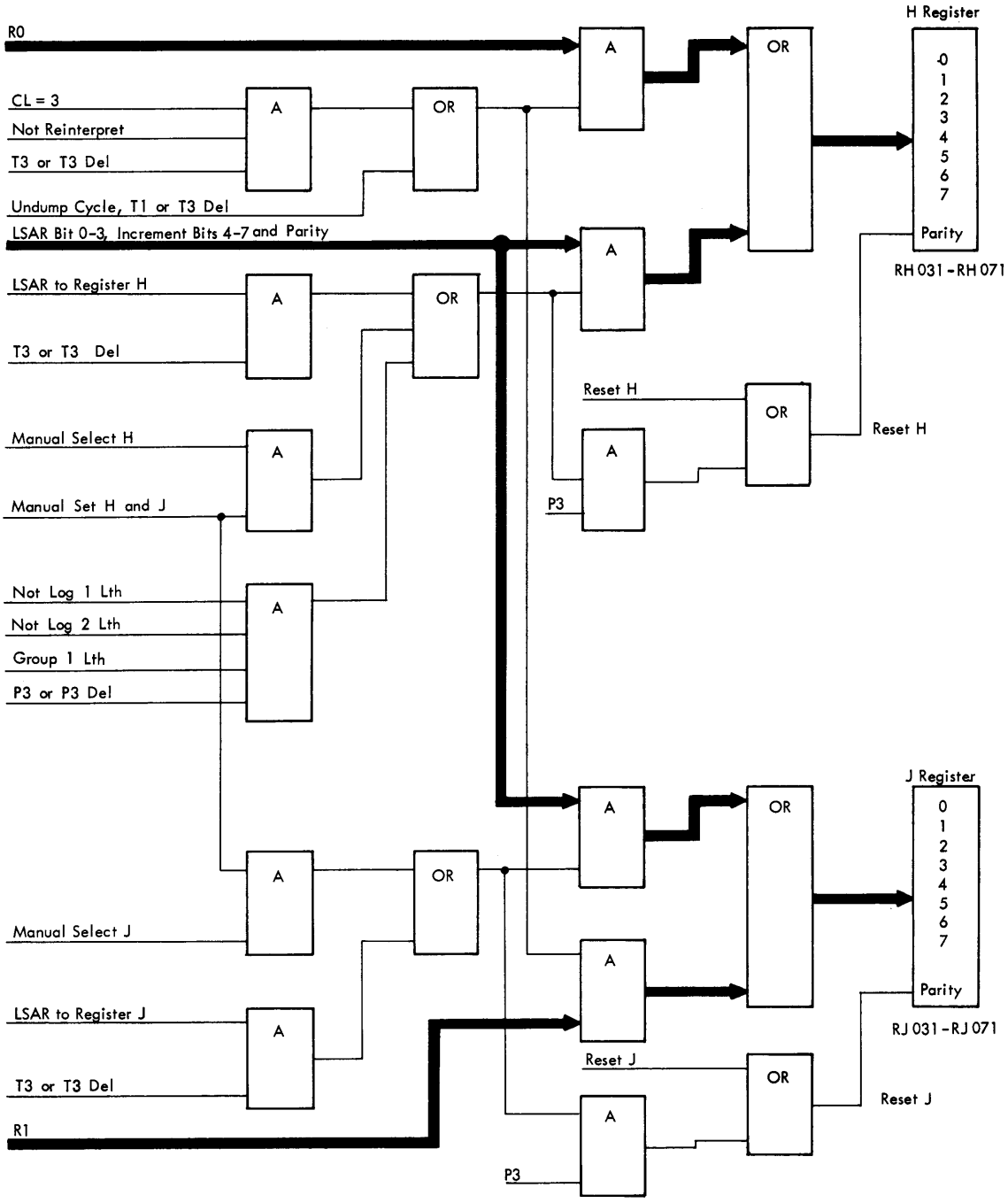


Figure 68. LSAR Destination, H and J Register

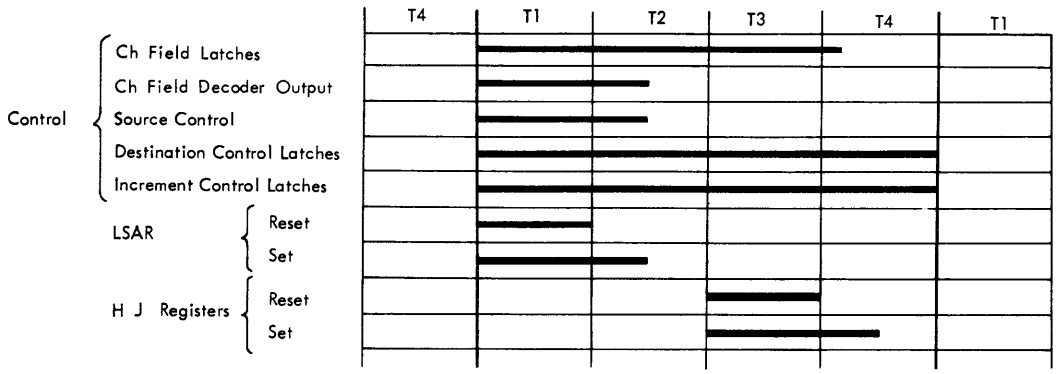


Figure 69. Local Storage Address Loop Timing

age areas are addressable by the machine language program.

The local storage has a capacity of 144 words of 24 bits. A local storage word has to store a complete main storage address in one location. Twenty-two bits, including three parity bits, are used assuming a main storage of 128K bytes.

The local storage is of SLT construction and is capable of split-cycle operation.

A read or write cycle takes 625 nanoseconds. The access time is 350 nanoseconds. Access time is defined as the interval between the time when the select pulse reaches the input connector and the time when the word leaves the unit on its way to the R register.

The local storage consists of two core planes each mounted on a 4-96 SLT card. All connections to the core planes are made by printed wiring on the SLT small cards. Also mounted on these cards are the matrix diodes, connecting the X and Y lines in the core planes, and the temperature-sensing thermistor. This thermistor is the sensing element for a circuit that keeps the drive current at its nominal value with temperature variations.

The storage unit operates within a temperature range of +10° to 40°C (+50° to 104°F) and within a relative humidity range of 10 to 80 percent.

Figure 70 is a block diagram of the circuitry associated with local storage. The number on each line indicates the number of lines coming from one block.

The timing of the circuitry is obtained by feeding a storage select pulse, timed by the CPU P clock, into a delay line and tapping the delay line at 50-nanosecond intervals.

The timing for the write gates and drivers is provided by the read gate timing circuit via the read control amplifier.

The inhibit drivers are timed with the write gate. The sense strobe, which strobes the sense amplifiers during a read cycle, is provided by the sense strobe circuit.

The two current source circuits provide the drive current for the X and Y drivers, which is approximately 380 ma. This is the current to be switched by a selected driver.

The V reference generator circuit provides a reference voltage to the current sources to define the current produced. To vary the current, V reference can be varied over a certain range by a potentiometer located on the CE panel. V reference is also controlled by a thermistor, located near the array, to compensate changes of drive current due to temperature variations.

A special power supply provides a temperature controlled voltage −Vm to the inhibit drivers. The threshold voltage for the sense amplifiers is provided by the VSL generator.

Read Operation

The X and Y drive line combination unique to a word is selected. The drive current is in the opposite direction to that during the write cycle. Cores that are in the one-state flip to the zero-state and a sense output pulse is induced in the sense line for that bit position.

Cores in a zero condition experience no significant change of state (remain in the zero-state), and no output is sensed.

Write Operation

- **Read must precede write to clear storage location.**

To write a particular bit configuration into any word, the X and Y drive lines for that word are selected. This attempts to put all one's into that word. Where a zero is required an inhibit line is raised for that bit position, preventing the position setting to the one-state by nullifying the effect of the Y half-select current.

A read cycle must be taken before a write cycle in order to clear the storage location; this is handled by microprogram.

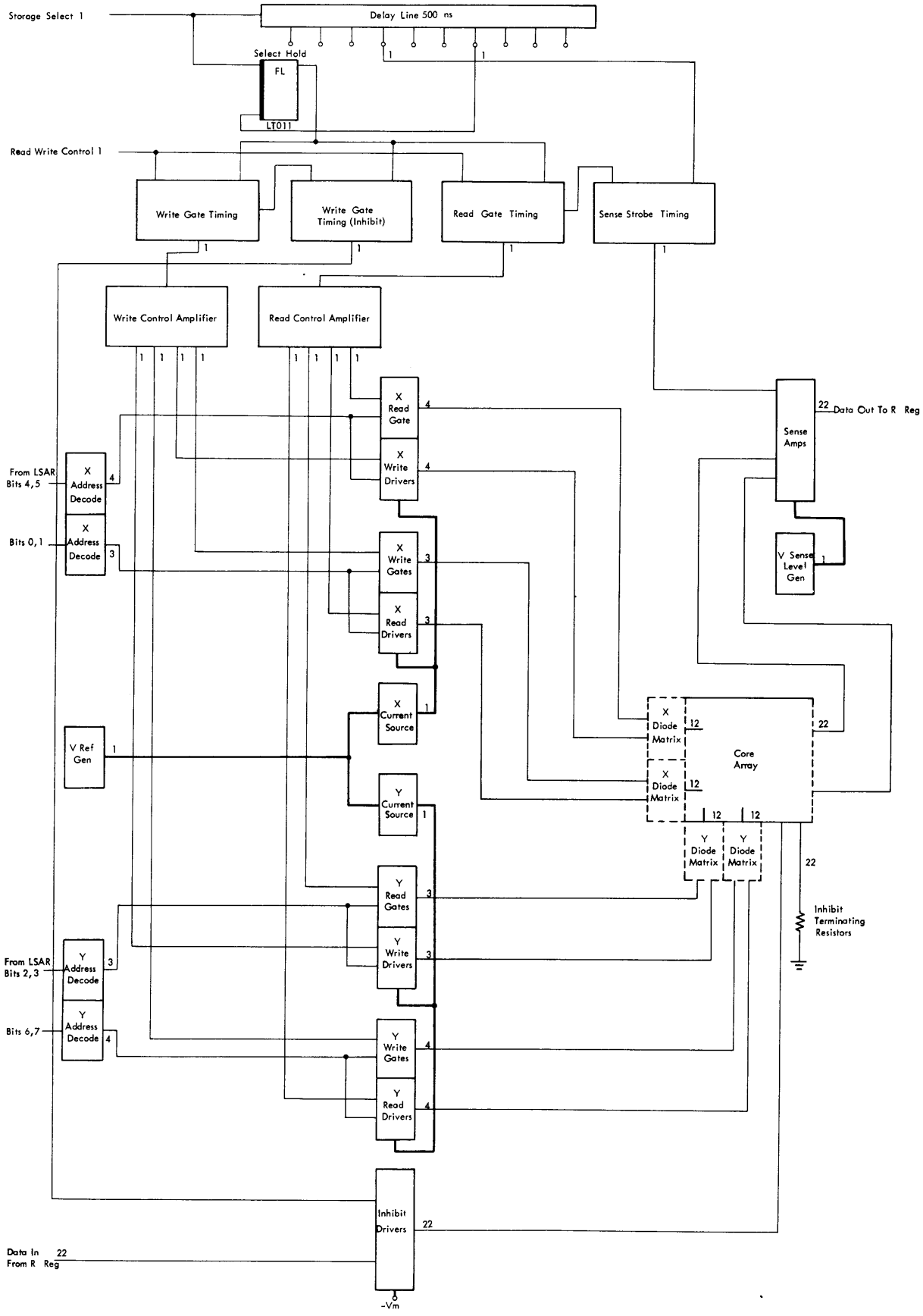


Figure 70. Local Storage Block Diagram

Addressing

- **Local storage addressed by LSAR.**
- **LSAR bits 0, 1, 4 and 5 form X address.**
- **LSAR bits 2, 3, 6 and 7 form Y address.**

The local storage is addressed by the local storage address register (LSAR), which has eight bits plus a parity bit.

LSAR bits 0, 1, 4 and 5 are decoded to address the X lines.

LSAR bits 2, 3, 6, and 7 are decoded to address the Y lines (Figure 71).

The X decode of LSAR bits 0 and 1 together with the Y decode of LSAR bits 2 and 3 addresses local storage in blocks of 16 words each.

Because only 144 words (9 blocks of 16 words) are needed, some bit configurations are invalid:

X decode LSAR bits 0, 1 = 10 is invalid and is automatically decoded into bits 0, 1 = 11.

Y decode LSAR bits 2, 3 = 01 is invalid and is automatically decoded into bits 2, 3 = 00.

The words in each 16-word block are addressed by LSAR bits 4 and 5 (X) and LSAR bits 6 and 7 (Y).

Example:

LSAR contains 00101101
LSAR bits 0, 1 contain 00
LSAR bits 2, 3 contain 10
Therefore the block containing words 32 through 47 is addressed.
LSAR bits 4, 5 contain 11
LSAR bits 6, 7 contain 01
Word 13 of the block, i.e. Word 45 is addressed, (Figure 71).

Drive Scheme

When entering the LS ALD pages, the LSAR bit lines change name as follows:

LSAR bits	0	1	2	3	4	5	6	7
become SAR line	128	64	32	16	8	4	2	1

The 144-word storage has 12 X lines and 12 Y lines.

The X lines are labelled X0 through X11 at one end and X0 prime (′ or P) through X11 prime at the other. The Y lines are labelled Y0 through Y11 at one end and Y0 prime through Y11 prime at the other end.

Physical layout of the core planes is shown in Figure 72.

X Line Driving and Gating

- **12 X lines.**
- **Four write drivers, four read gates at one end.**
- **Three write gates, three read drivers at other end.**

A matrix drives and gates the 12 X lines in the read and write directions. The matrix consists of four write drivers and four read gates at one end of the lines, and three write gates and three read drivers at the other end.

Local storage address register bits 4 and 5 are decoded and each bit combination addresses a write driver and a read gate.

The final selection between write driver and read gate is made by either the line labeled 'read gate (T)' being active during a read cycle, or the line labeled 'write gate (T)' being active during a write cycle.

Local storage address register bits 0 and 1 are decoded and each bit-combination addresses a write gate and a read driver.

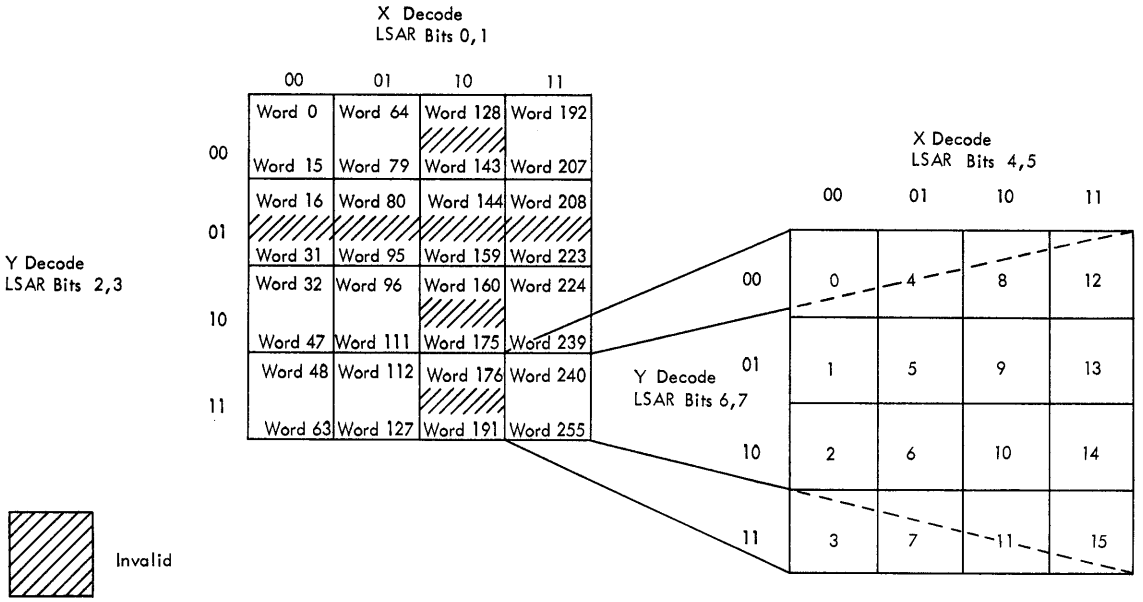


Figure 71. Address Decode Diagram

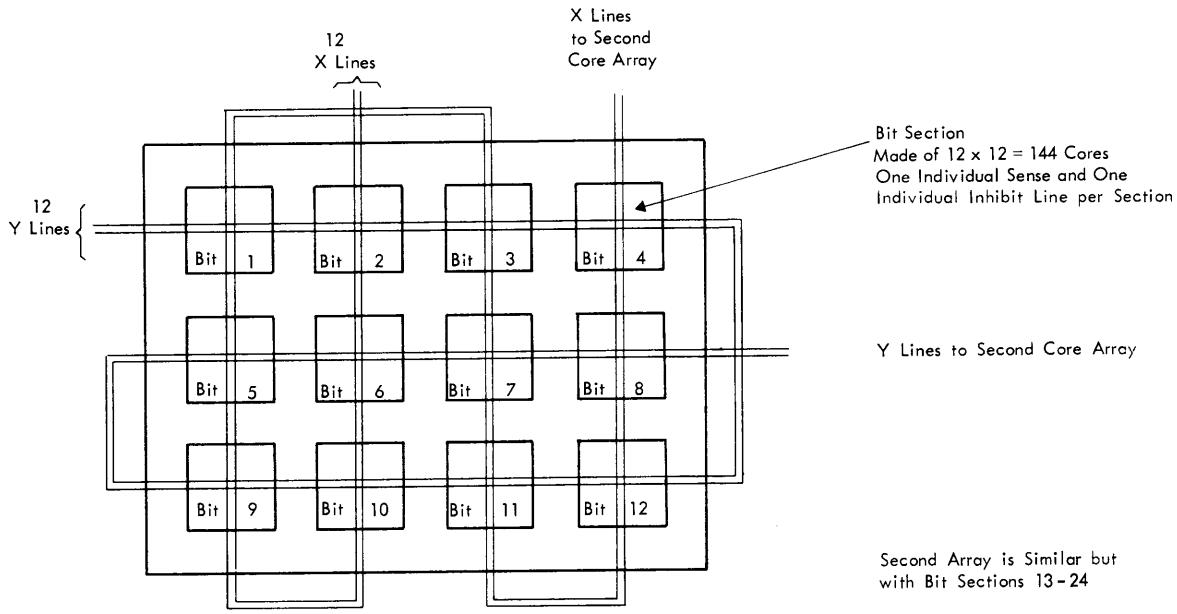


Figure 72. Layout of One Core Plane

The final selection between write gate and read driver is again made by either the line labeled 'read gate (T)' being active during a read cycle and the line labeled 'write gate (T)' being active during a write cycle.

The diodes in series with the X lines isolate a selected X line from not-selected X lines.

The numbers over the blocks indicate the labels of the X line ends connected to these blocks. 'P' stands for 'Prime'.

Example (Figure 73):

LSAR bits 0 and 1 contain 10
LSAR bits 4 and 5 contain 01
Local storage is to perform a read cycle.

Local storage address register bits 0, 1 = 10 is invalid and is automatically decoded as LSAR bits 0, 1 = 11. This, together with read gate (T) will activate the read driver.

The X line ends 8P, 10P, 9 and 11 are connected to this driver.

LSAR bit 4, 5 = 01, together with read gate (T) will activate the read gate.

The X line ends 1P, 5P and 9P are connected to this gate.

The labels 9 and 9P are common to one X line and this line will be selected.

Y Line Driving and Gating

The Y drive scheme is shown diagrammatically in Figure 74.

The Y line driving and gating is similar to the X line driving and gating. The drivers and gates are addressed by LSAR bits 2, 3, and 6 and 7.

Inhibit

- **One inhibit line for each bit position of the R register.**
- **Inhibit line activated during write cycle when corresponding R register bit is zero.**

A local storage word has a maximum length of 24 bits, of which only 22 bits are used assuming a 128K byte main storage. The local storage data register is the R register.

The inhibit line goes through every core within a bit section for a particular bit position and runs parallel with the Y lines.

The 22 drivers that are used are activated by the R register output during a write cycle (Figure 75). A driver is activated when the corresponding R register bit position contains a zero (0).

Sense

- **One sense line for each bit position of the R register.**

The sense operation is shown in Figure 76.

Twenty-two of the 24 sense lines are used, assuming a 128K byte main storage.

The sense line goes through every core within a 'bit section' for a particular bit position and runs parallel with the X lines.

Both ends of each sense line are connected to a differential amplifier.

When a pulse is sensed during a read cycle (sense strobe line active), the corresponding bit in the R register is set to 1.

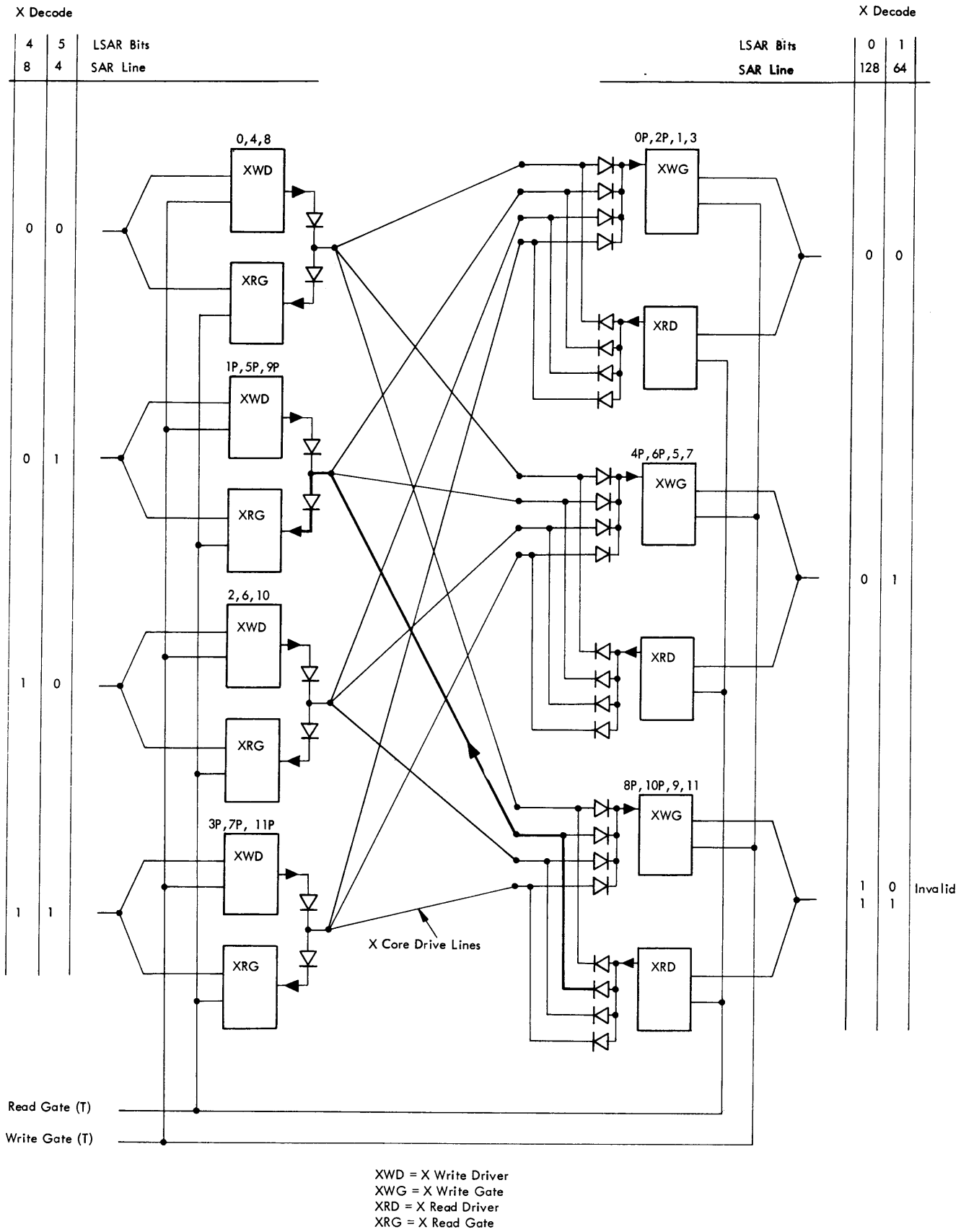


Figure 73. X Drive Diagram

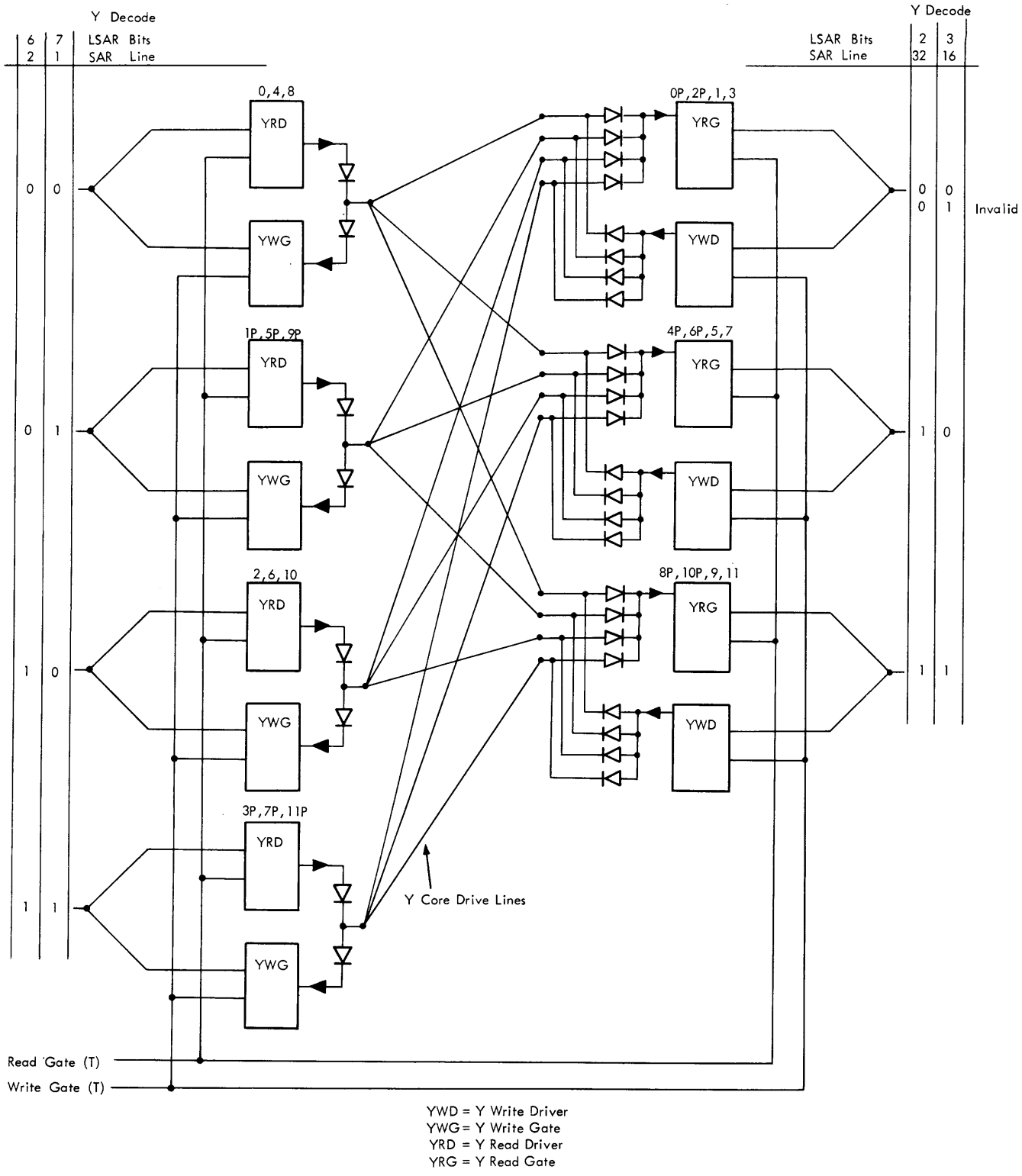
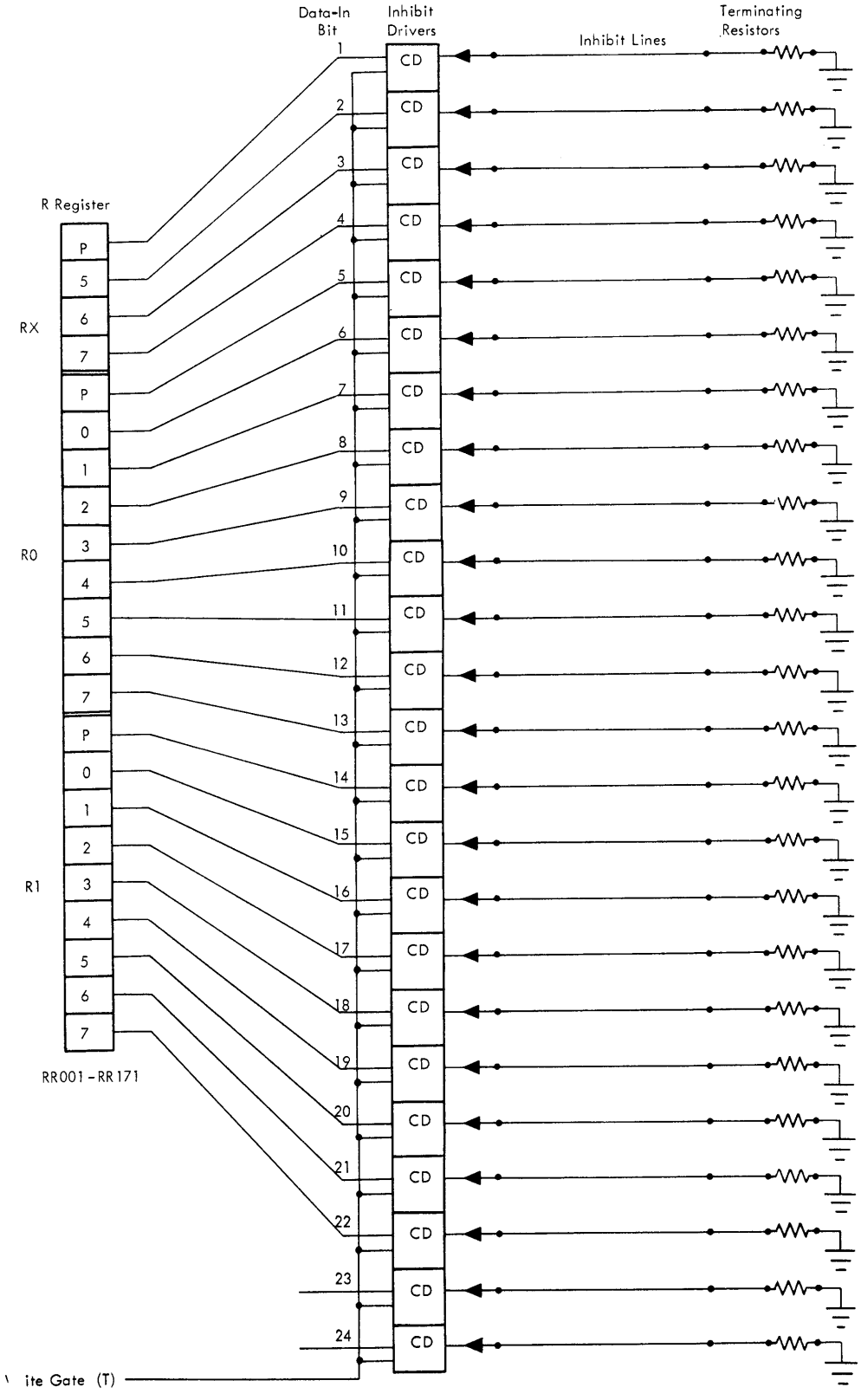
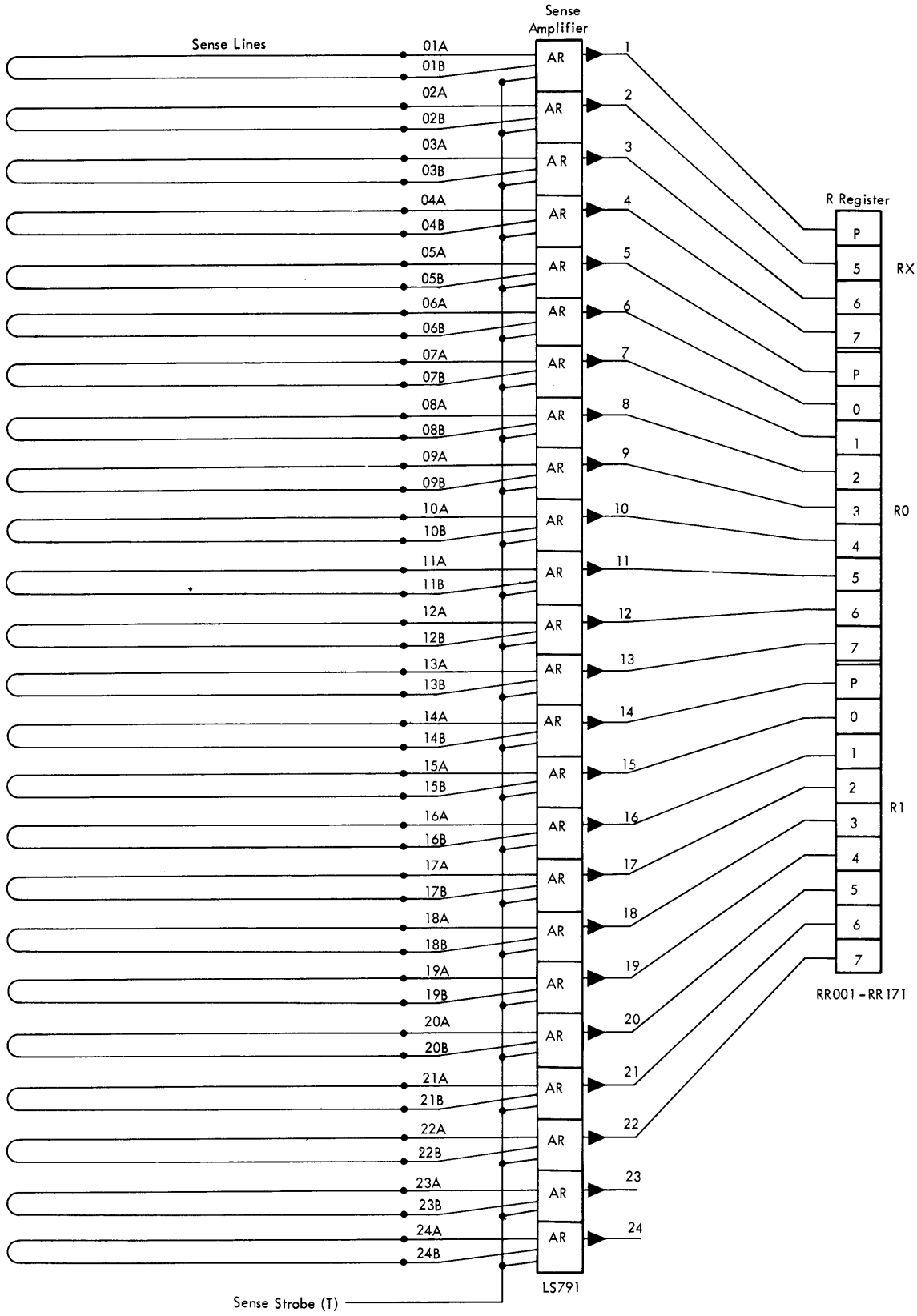


Figure 74. Y Drive Diagram



Note: Inhibit Lines Run Parallel With Y Lines In Each Bit Section
CD = Core Driver

Figure 75. Inhibit Drive Diagram



Note: Sense Lines Run Parallel With X Lines In Each Bit Section

Figure 76. Sense Diagram

Read and Write Controls

- **Local storage call initiates storage cycle.**
- **Read control determines read or write cycle.**

The local storage read and write controls are shown in Figure 77.

Two signals are generated to control local storage read or write:

Local storage call initiates the storage cycle.

Read control determines whether a read or write cycle is to be performed.

To initiate a read cycle, local storage call is timed by P1 del and extended by write control. To initiate a write cycle, local storage call is timed by clock even at P2.

Read control is activated when a local storage read cycle is required. Therefore, local storage call and read control both active will cause the local storage to perform read cycle; local storage call active and read control not active will cause the local storage to perform a write cycle.

Timing

- **Timing pulses provided by delay line clock.**
- **Outputs of delay line provides timing pulses.**
- **Timing asynchronous with respect to CPU timing.**

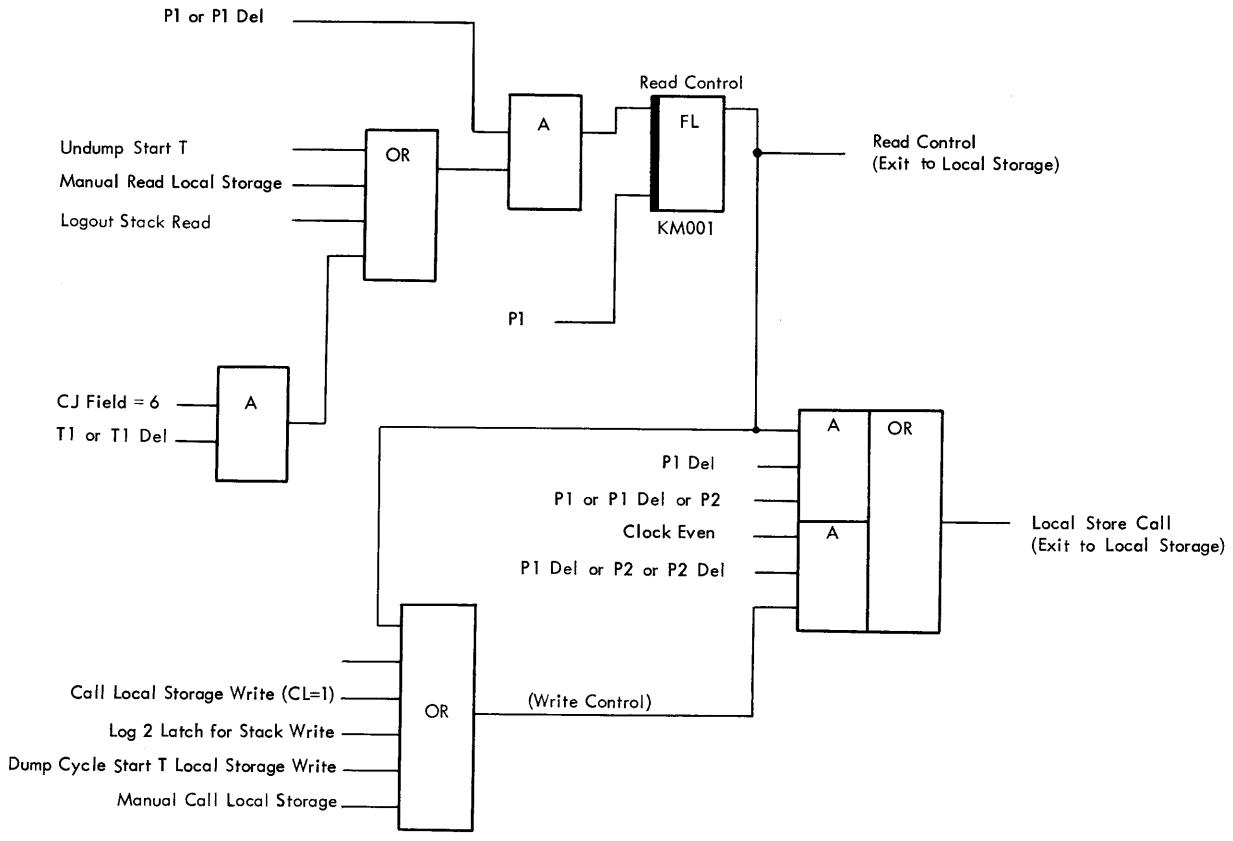
Figure 78 shows the local storage timing circuit.

Figure 79 shows the timing for LS including the data-in and data-out times.

The timing circuitry provides the timing for the local storage drivers and gates, the inhibit drivers, and sense amplifiers. It is activated by the storage select and read/write control signals.

To initiate the storage cycle, the storage select pulse is fed into a delay line and also sets a timing latch.

The delay line, providing a maximum delay of 500



Note: EC255278 changes width of local store call from 157ns to 220ns on read cycle.

Figure 77. Local Storage Read and Write Controls

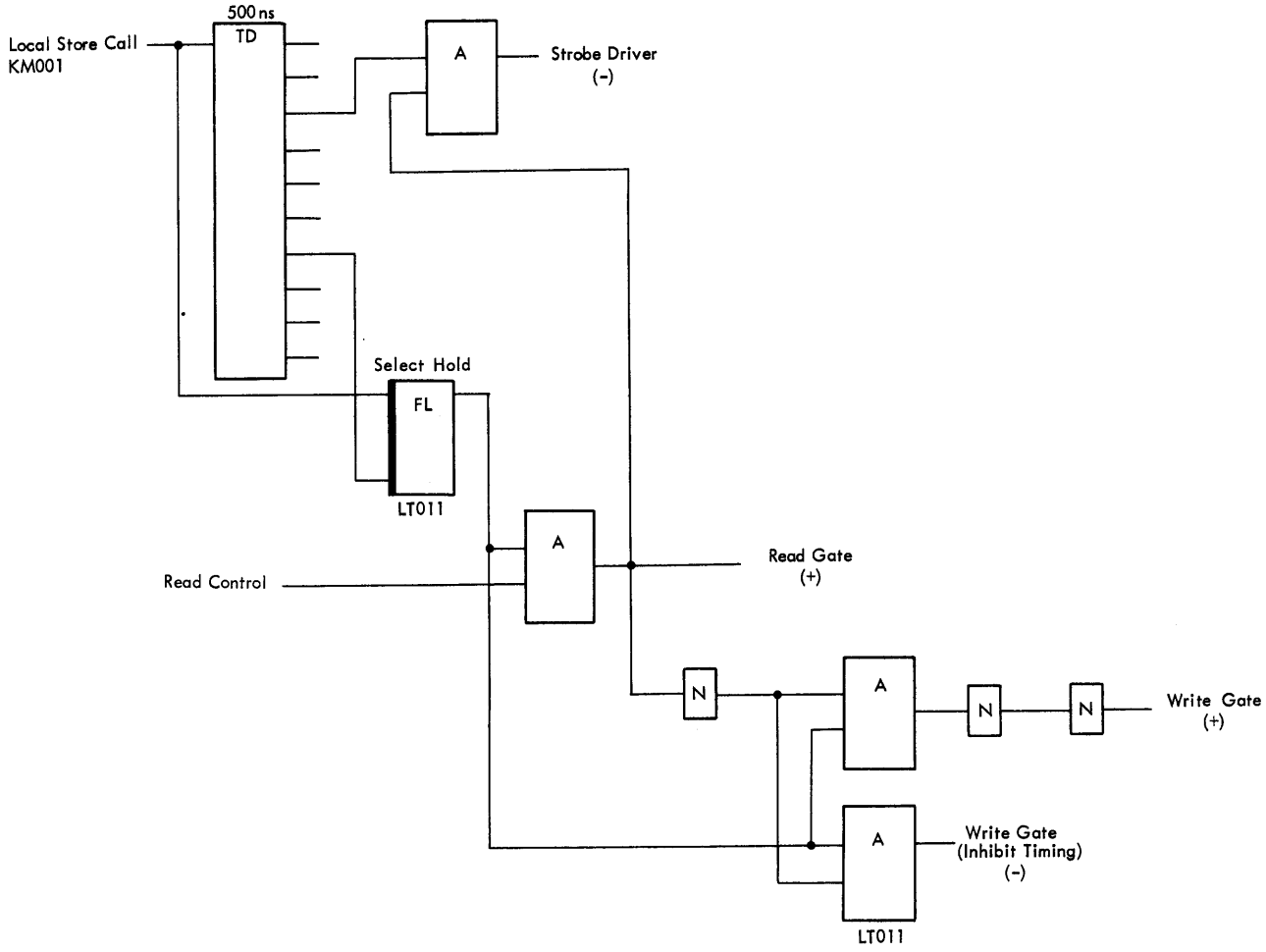


Figure 78. Local Storage Timing Circuit

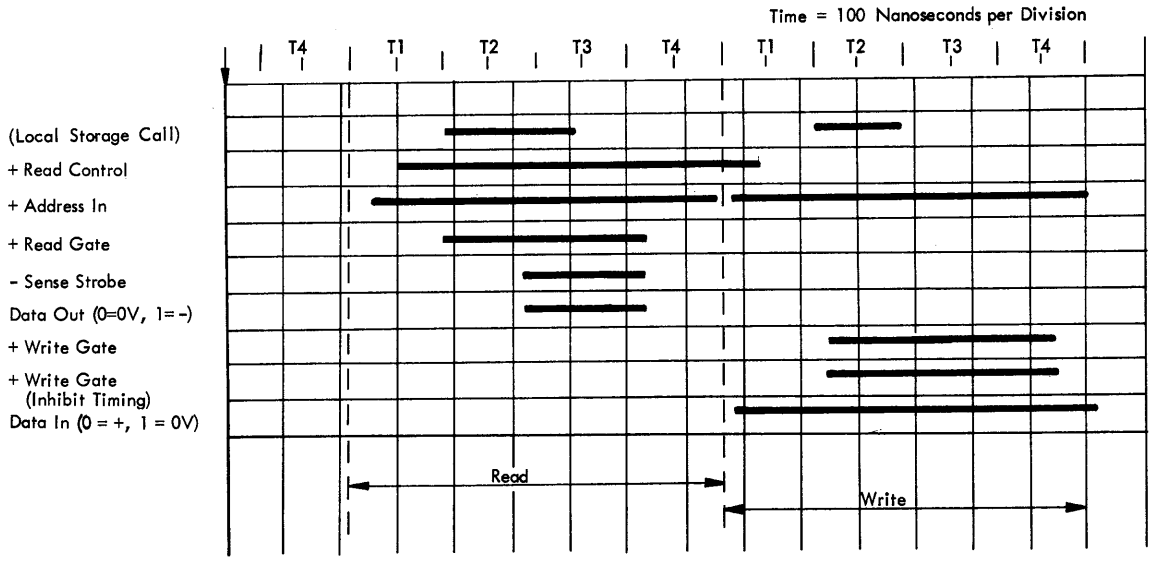


Figure 79. Local Storage Timing Diagram

nanoseconds, is tapped at 50-nanosecond intervals. After 350 nanoseconds, the delayed storage select pulse resets the timing latch.

The output of the timing latch is AND'ed with read/write control. When read/write control is active the AND condition is satisfied and read gate is activated to time the read drivers and gates. When read/write control is not active, write gate is raised to time the write drivers and gates and inhibit drivers.

Because of the delay introduced in the form of three inverters, the write gate timing for the drivers and gates comes later than for the inhibit drivers.

During read gate time, a 150-nanoseconds-delayed storage select pulse provides the sense strobe for the sense amplifiers.

Read Cycle

When read control is positive, a local store call pulse initiates a read cycle at P1 del time (Figure 79). At that time the address is already available in LSAR and the proper X and Y drivers and gates are address-selected. The read gate timing of 350 nanoseconds activates the address-selected X and Y read drivers and gates via the read control amplifier.

The X and Y drive currents start to flow in the read direction and flip addressed cores that were in the one-state to the zero-state, inducing a sense pulse in the corresponding sense lines.

About 150 nanoseconds after raising the read gate timing pulse, the strobe pulse, which is actually the delayed storage select pulse, allows the sense amplifiers to accept the sense pulse from the sense lines. A one appears at the sense amplifier output as a negative-going pulse.

Write Cycle

Read/write control drops as soon as the local storage read latch is reset. See Figure 79. The next storage select pulse initiates the write cycle. The address is available in LSAR and the proper X and Y drivers and gates are address-selected. The data to be written are available in the R register.

Bit positions that contain a zero will select the corresponding inhibit driver. The write gate timing of about 350 nanoseconds activates the address-selected X and Y write drivers and gates via the write control amplifier. The X and Y drive currents start to flow in the write direction.

The inhibit timing activates the data-selected inhibit drivers, starting before the write gate timing and ending at the same time or slightly later.

The X and Y half-select currents try to flip all addressed cores to the one state. The inhibit half current flows in the opposite direction through the cores that are to be kept in the zero-state, preventing them from flipping to the one-state.

Circuit Description

The following circuit descriptions are located in the Appendix of the *System/360 Model 40 Power Supplies, Features, and Appendix* (Form 223-2845):

Special Power Supply (−Vm)
Reference Voltage Generator (V Ref)
Sense Level Generator (VSL)
Read Control Amplifier
Write Control Amplifier
Current Source Circuit
Decoder
Gate and Driver Circuits
Inhibit Driver
Sense Amplifier
Strobe Driver

- **Optional feature.**
- **Protects main storage from destruction by unauthorized programs.**
- **Principal significance in multiprogramming.**
- **Multiple blocks need not be consecutive.**
- **Four-bit storage key associated with each block.**
- **Four-bit protection key associated with each CPU or channel program.**
- **Every main storage access automatically evaluates the protection key.**

The storage protect feature protects defined areas of main storage from inadvertent destruction by unauthorized programs.

In multiprogramming this feature protects individual programs from the other programs. Accidental destruction as a result of programing errors or machine malfunctions are avoided. Even with a single object program there is at least one other program, the supervisor, in main storage. The supervisor program must not be destroyed by an improper object program.

For protection purposes, main storage is divided into blocks of 2047 (2K) bytes. Up to 16 different areas of main storage can be independently protected, each area consisting of a number of 2K blocks. These blocks need not be consecutive.

The four-bit storage keys associated with each 2K main storage block are held in a separate core-storage unit, similar to local storage, called SPLS (storage protect local storage). Storage keys are inserted into SPLS by the instruction set storage key, a privileged instruction only available in supervisor state.

When main storage is accessed, in a CPU operation, the storage key of the addressed location is read out of SPLS and compared with the four-bit protect key supplied by the PSW.

During I/O operations, the protect key is supplied by the UCW where it is initially set up by the CAW.

For selector channels, the protect keys are held in the channel key registers. For the multiplex channel, the protect keys for all subchannels are held in the multiplex (Mpx) area of SPLS.

During every main-storage read cycle, the storage key is compared with the protect key. An unequal comparison is latched up and prevents the D register data from changing before it is written back into storage.

In addition, the unequal compare condition is tested by microprogram during all instruction routines which enter new data into main storage (store, move, logic operations in the ss format, decimal operations, and read type I/O operations).

Detection of unequal compare (protect violation) results in a program check interrupt or, during I/O operations the operation is terminated and the program check bit set into the CSW.

A protect key of 0000 always gives equal compare, and therefore ignores any protection set up by the storage keys.

Figure 80 shows a symbolic representation of the storage protect feature. In the following description, the terminology used corresponds to the symbology in the figure.

Every 2K block of main storage is protected by a padlock (storage key). Main-storage information can be destroyed only by somebody in possession of the key (protect key) that opens the padlock.

Sixteen different locks and keys are available as four bit numbers; key 0 is a master key that opens every lock. The lock 0, however, can be opened only by the master key.

The padlocks used during system operation are held in SPLS in boxes that correspond to the 2K blocks of main storage. Loading of padlocks into SPLS takes place before actual job execution starts. Any number of locks of the same type can be used and no particular sequence of lock assignment has to be observed. Padlocks are loaded by the instruction set storage key.

Functional Unit Circuits

Storage Protect Address Bus

- **Storage protect address bus contains eight bits and is labeled L MAR 1 to L MAR 128 (Figure 81).**
- **Feed through from main storage address depending on Mpx stat or Mpx console switch (Figure 83).**

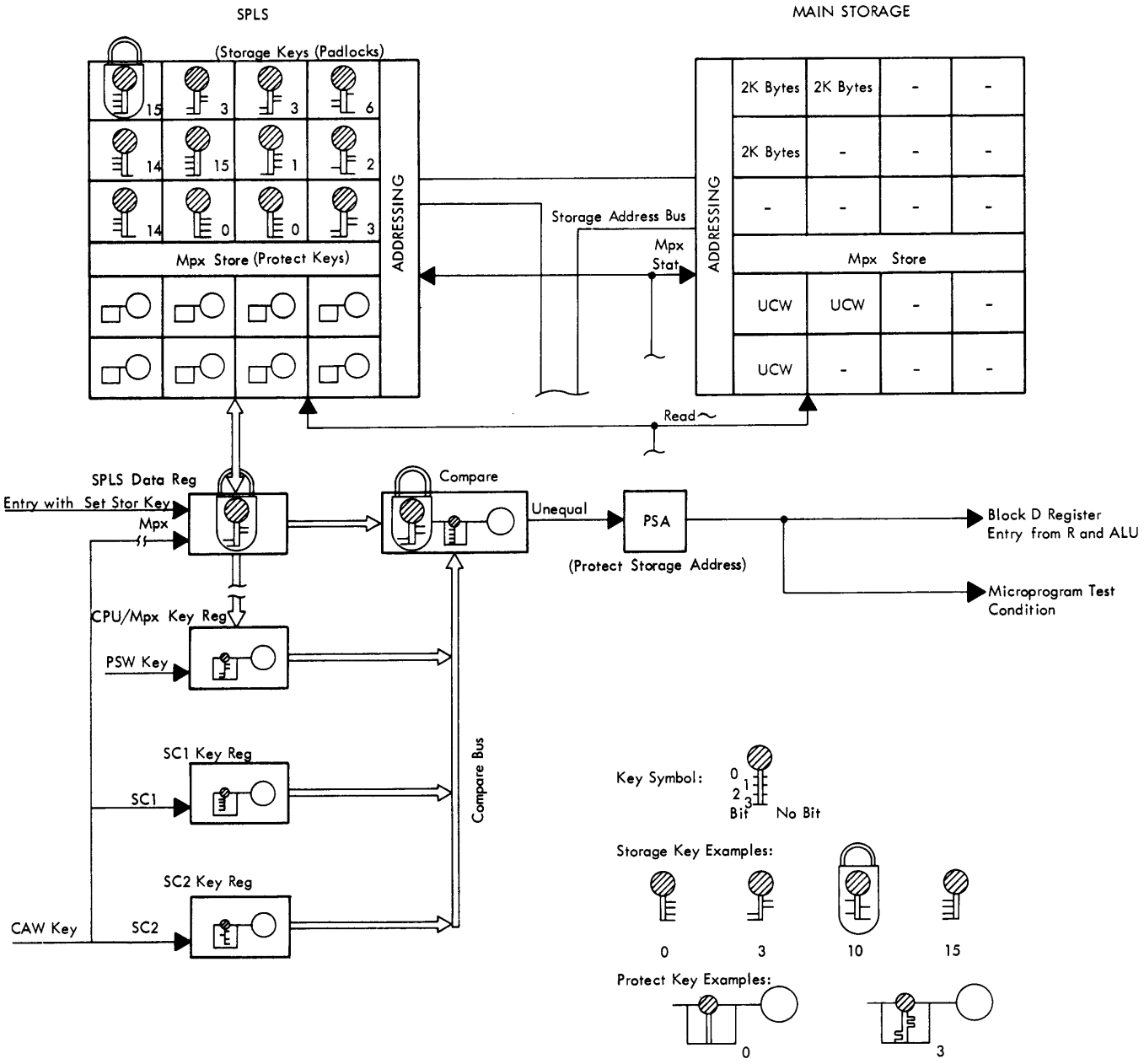


Figure 80. Storage Protection Symbolically

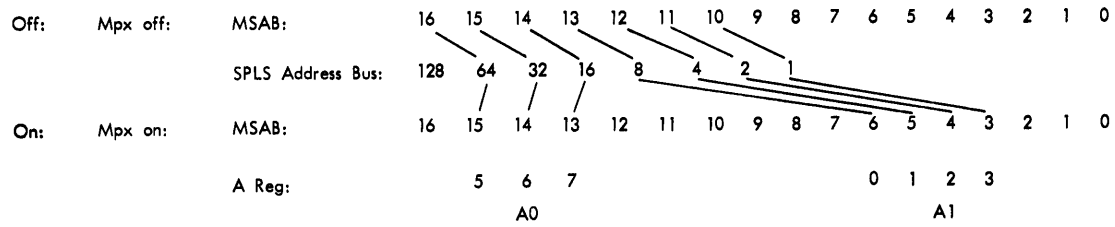


Figure 81. SPLS Addressing Scheme

SPLS Data Register

- **Input/output register for SPLS.**

The data register is four bits wide plus one parity bit. It can be loaded from:

1. Storage protect local storage. Good parity always generated.
2. Q bus bits 0-3.

See Figures 82, 84 and 85.

CPU/Mpx Channel Key Register

- **Holds the CPU or multiplex channel protect key.**

The CPU/Mpx channel key register is four bits wide plus one parity bit.

It can be loaded only from ALU bus bits 4 through 7 (the parity bit is generated).

The output is gated to the compare bus and Q bits 4 through 7 (Figures 84 and 85).

Selector Channel Key Registers

- **Holds the selector channel protect keys.**

The selector channel key registers are four bits wide plus a parity bit. They can be loaded only from ALU output bits 4 through 7. The parity bit is generated.

The output is gated to the compare bus during log out to R bus.

Parity Checking

The SPLS data register and the compare bus are checked for odd parity. A parity error forces an early check (Figure 85).

Assignment of Storage

- **Storage keys are set using the privileged instruction, SSK.**

Storage keys are loaded into SPLS locations 0 through 127 by the set storage key instruction, SSK. Bits 8 through 31 of the register specified by R2 address the storage block for which the key is to be set.

The key occupies bits 24 through 27 of the register specified by R1. Bits 28 through 31 are set to zero; bits 0 through 23 are unchanged. The keys are loaded into the SPLS data register from the ALU bus bits 0 through 3. The SPLS data register is loaded into SPLS in the location specified by the main storage address.

Examining the Storage Keys in SPLS

- **Storage keys in SPLS locations 0 through 127 can be examined by the instruction insert storage key (ISK).**

The storage keys in SPLS locations 0 through 127 can be examined by the insert storage key instruction. Bits 8 through 31 of the register specified by R2 address the storage block for which the key is to be obtained.

The key is inserted in bits 24 through 27 of the register specified by R1. The keys are read out of SPLS and loaded into the SPLS data register. The SPLS data register is read out to the Q bus bits 0 through 3. See Figure 82.

Storage Protect for CPU Operation

- **Key register loaded when loading PSW.**

During a load new PSW routine, the protect key in the PSW bits 8-11 is loaded into the CPU/Mpx channel key register from the ALU bits 4-7. The key is kept in this register as long as the PSW is controlling the CPU. When main storage is addressed, the main storage address bus bits 16-10 address a storage key in SPLS.

The addressed key is read out and loaded into the SPLS data register. In the second cycle of read, comparison between SPLS data register and CPU/Mpx channel key register takes place. If a protect violation is sensed, the PSA latch is set.

Storage Protect for Selector Channel

- **During start I/O routine, protect key associated with CAW is set into SC1 or SC2 key register.**

During the start I/O routine on a selector channel, the protect key from the CAW is set into the appropriate selector channel key register from ALU bits 4-7.

The key is held in the channel key register for the entire I/O operation. When main storage is accessed by the channel, the corresponding storage key is read out into the SPLS data register.

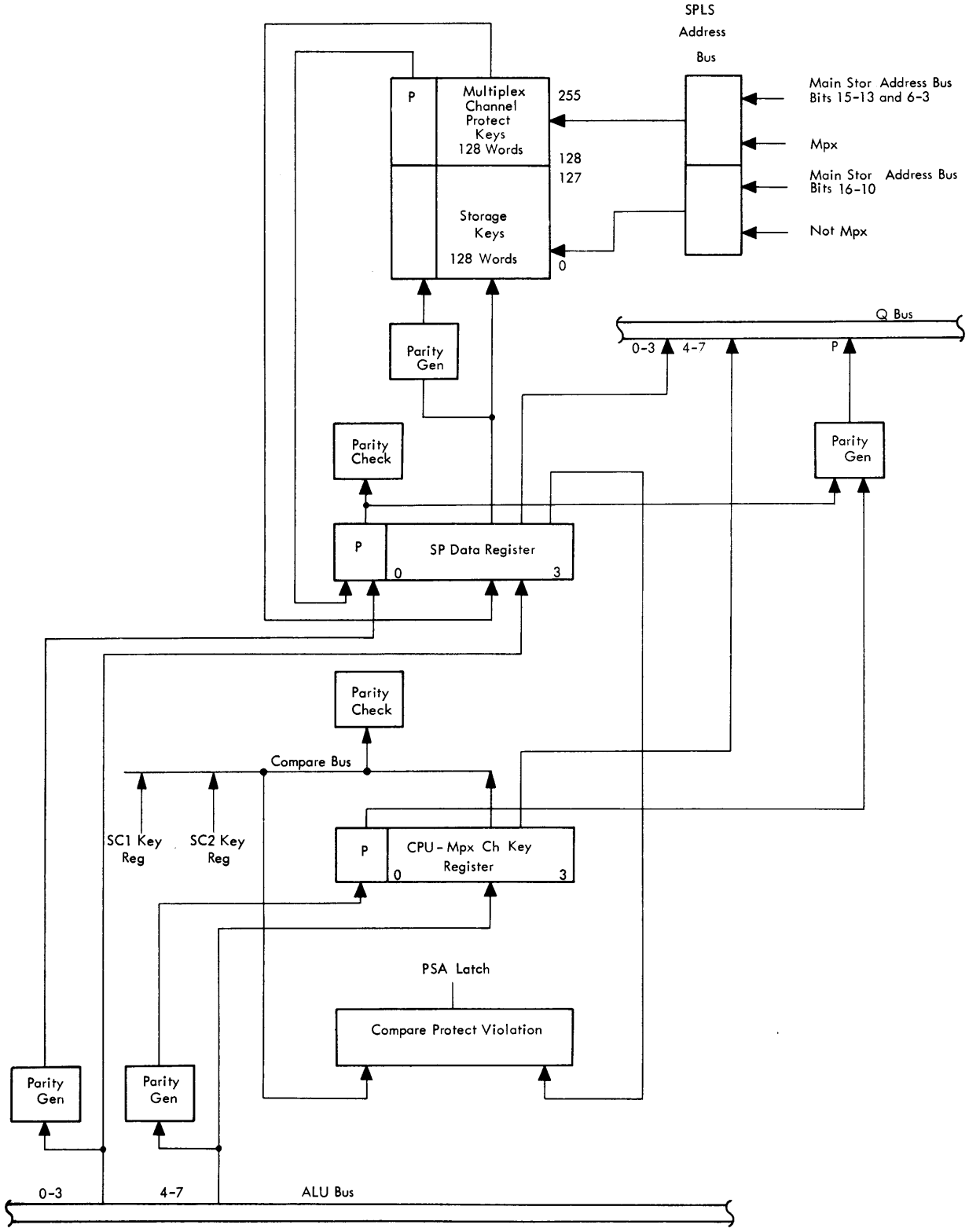


Figure 82. Storage Protect Data Flow

In the second read cycle, comparison between SPLS data register and channel key register takes place. If a protect violation is sensed, the PSA latch is set.

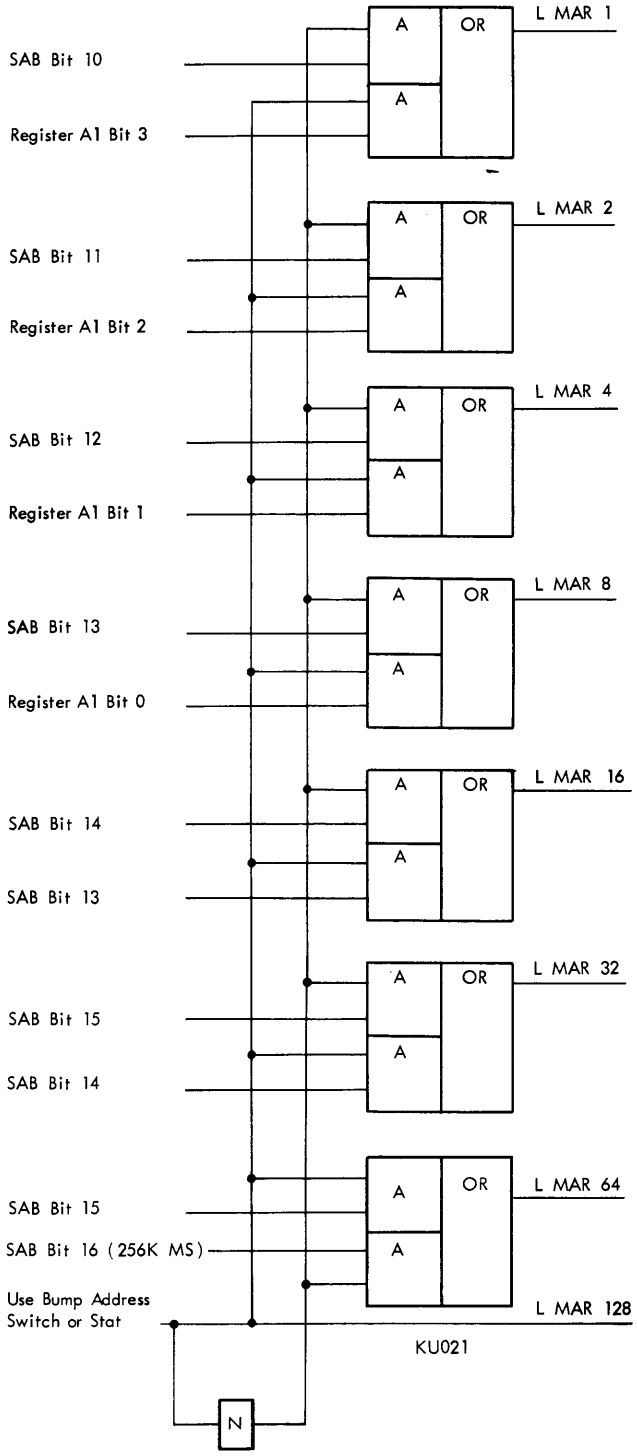


Figure 83. Storage Protect Address Bus

Storage Protect for Multiplex Channel

- **When UCW is stored in multiplex storage, associated protect key is stored in SPLS.**
- **During dump, CPU protect key is saved in local storage.**
- **When UCW is addressed, protect key is read out to SPLS data register and transferred to CPU key register.**

During the start I/O routine, when a UCW is stored in multiplex storage, the protect key that belongs to the UCW is stored in the corresponding location of storage protect local storage. During dump (Mpx channel requires service) the CPU protect key in the CPU key register is placed in local storage. When the UCW is read out of multiplex storage, the protect key belonging to the UCW is read out of SPLS and loaded into the SPLS data register.

The SPLS data register is fed to the Q bus (bits 0-3), circulated twice through the ALU, and skewed each time. In the first cycle, the key enters the skew buffer; in the second cycle, skew select bits 4-7 enter the skew buffer. The key appears at the ALU bus bits 4-7 and is loaded into the CPU/Mpx channel key register.

When main storage is accessed to handle a byte of data from the channel, the corresponding storage key is read out of SPLS and loaded into the SPLS data register. In the second cycle of read, comparison between SPLS data register and CPU/Mpx key register takes place. If a protect violation is sensed, the PSA latch is set.

Protect Storage Address (PSA) Detection

- **Mismatch between storage key and protect key generates the line invalid tag, if the protect key is other than zero.**
- **Invalid key generates PSA in the second machine cycle after main storage read is called, provided that no parity error is detected in either SP data register or on the compare bus.**
- **PSA directly preserves the D register contents by turning on PSA/ISA latch (refer to D register).**
- **PSA is latched in the CPU or in one of the selector channels to provide an indication to the program:**
  1. **In CPU or multiplex channel operations, SAT is generated if Y0 is on.**
  2. **In selector channel operations, the protection check and program check bits are set and the I/O operation is terminated.**
- **During multiplex channel break-in, the CPU/PSA latch is dumped.**
- **PSA detection circuit is shown in Figure 86.**

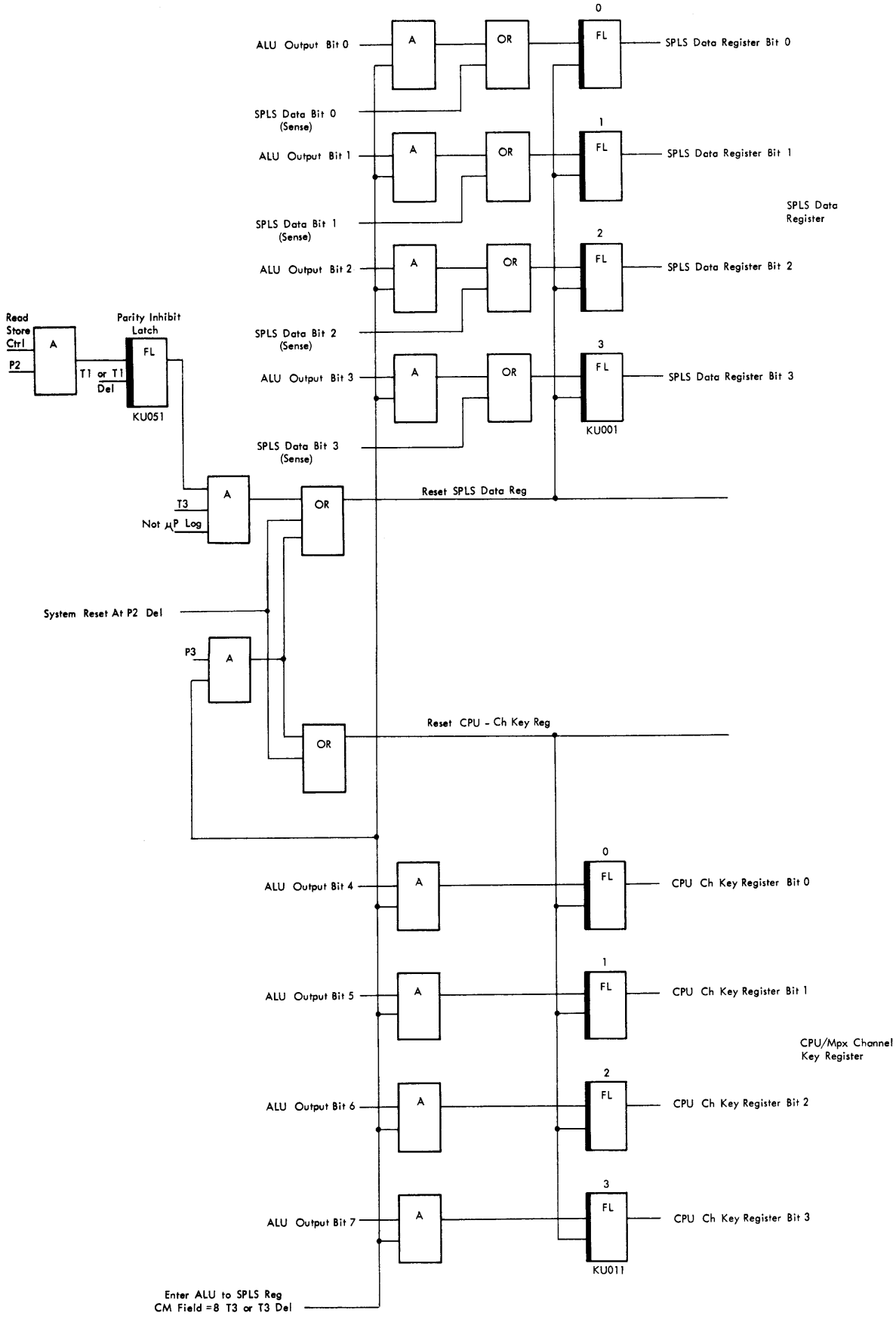


Figure 84. Storage Protect Local Storage Data Register, CPU/Mpx Channel Key Register

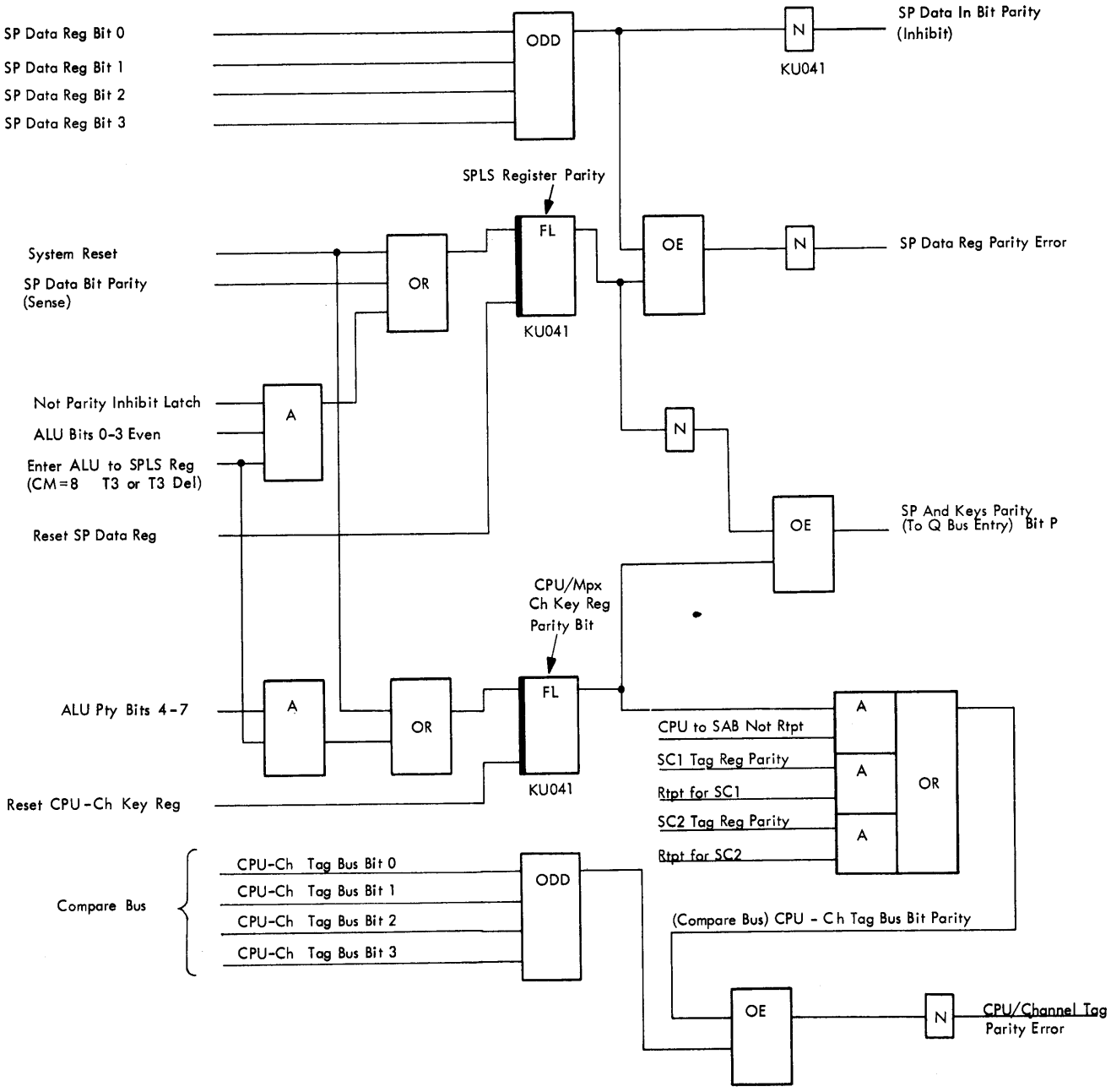


Figure 85. Parity Inhibit, Parity Generation, Parity Check

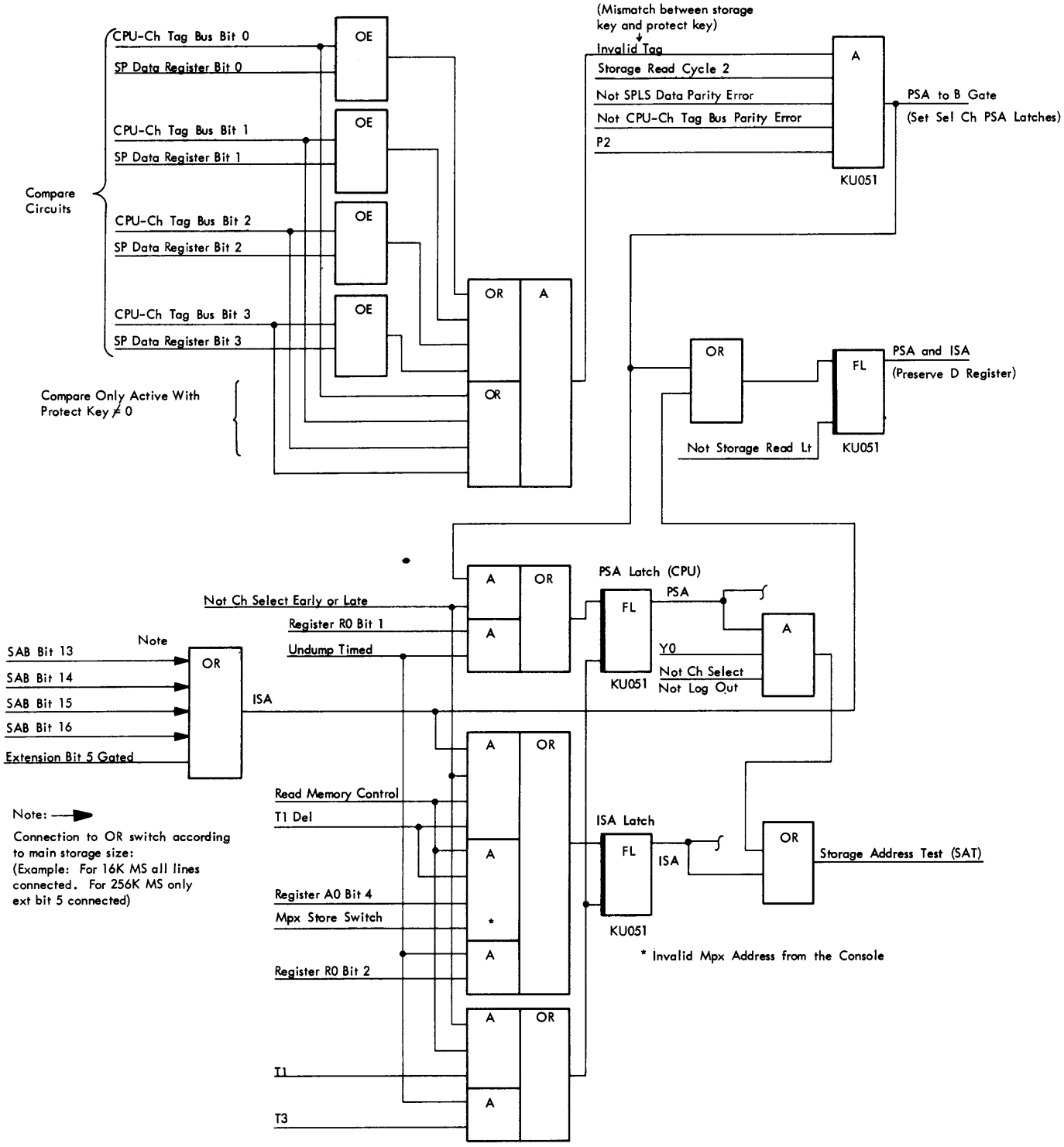


Figure 86. PSA and ISA Detection

Invalid Storage Address (ISA) Detection

- ISA detection is a standard machine feature and always active although the storage protect special feature may not be installed.
- ISA is generated whenever a main storage address outside the available storage size is called.
- ISA directly preserves the D register contents by turning on PSA/ISA latch (refer to D register).
- ISA is latched up in the CPU or in one of the selector channels to provide an indication to the program:
  1. In CPU or multiplex channel operations, SAT is generated.
  2. In selector channel operations, protection check and program check bits are set and the I/O operation is terminated.
- During multiplex channel break-in, the CPU/ISA latch is dumped.
- ISA detection circuit is shown in Figure 86.

Storage Address Tests (SAT and TRAP)

- Microprogram tests which are used only in CPU or multiplex channel microprograms after access to main storage to ensure proper addressing by the program.
- SAT is a C-condition test.
- TRAP is a microprogram branch to a forced TROS address if the SAT condition is present.

The SAT and TRAP circuits are shown in Figure 87.

SAT

The SAT condition can be tested with CC field = 14. Bit 0 of the next TROS address is set to 1 if SAT is present.

Note that the microprogram can distinguish between PSA and ISA by means of Y0. If Y0 is off, only ISA generates SAT; with Y0 on, either condition may be responsible for SAT.

TRAP

TRAP, CR field = 3, is another method of testing SAT, but in this case the next TROS address is completely forced by logic circuits if SAT is present. TRAP is used to branch from any microroutine to the program check interrupt microprogram if main-storage addressing is not properly specified.

TRAP sequence: (Refer to Figure 87 and Figure 3, T Clock Start-Stop):

1. With the micro-order TRAP, the trap latch is set at T2 del.
2. Without the SAT condition present, the trap latch is reset at T4 of the same cycle and nothing happens. However, if SAT is present, the trap latch stays on and stops the T clock at the end of the current cycle.
3. A CPU cycle takes place in which the trap force TROS address latch is set at P1 del. (This latch generates the signals necessary to force ROAR to either 404 or 405 depending on the main storage read latch. If TRAP is called after MS read, the program check interrupt routine has first to write back the D register in order to preserve the contents of the MS location. After write this is not necessary.)
4. At P3 in the CPU cycle, the trap latch is reset and the T clock can start at T1 of the next cycle.
5. The trap force TROS address latch is reset at P4 del of the CPU cycle.

Storage Protect Local Storage

- Small, four-wire core storage unit.
- Capacity, 256 six-bit words, 2040 uses five-bit words.
- SLT construction, one core plane.
- Alternate read and write cycles in parallel to main storage.
- Access time is 300 nanoseconds.
- Temperature compensated.

NOTE: The words storage and memory are used interchangeably between this instruction manual and the ALD's. For instance, the line storage select in this manual may become memory select in the ALD's. Beware of this when going from figures in this manual to the ALD's, and vice versa.

The storage protect local storage unit (SPLS) is a small, four-wire core storage unit used to store storage keys for every 2K block of main storage. It also holds the protect keys for the multiplex sub-channels.

The capacity of the unit is 256 words of six bits. In the IBM 2040 only 5-bit words are used. The SPLS uses SLT construction and consists of one core plane. Alternate read and write cycles are initiated in parallel with main storage R/W cycles. The access time is approximately 300 nanoseconds.

The core plane is mounted on a 4-96 SLT card. There are six bit sections in the core plane. See Figure 88. The temperature-sensing thermistor and the matrix diodes connected to the X and Y lines are mounted on the same SLT card as the core plane. All connections to

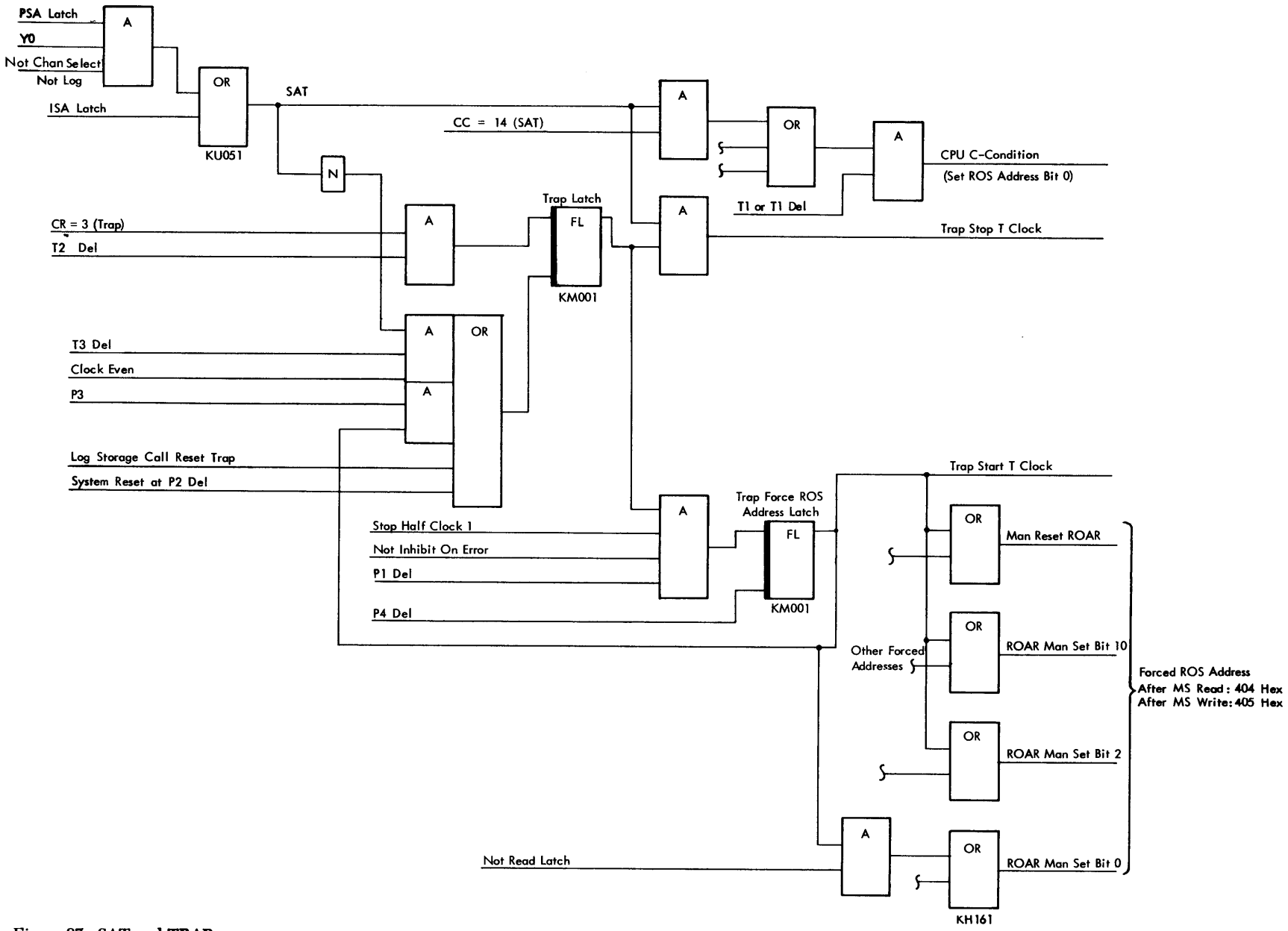


Figure 87. SAT and TRAP

the core plane are made by printed wiring on the SLT card.

General Write Scheme

- **X and Y drive lines selected for word addressed.**
- **Inhibit lines selected for those bit positions with zero-state.**

To write a particular bit configuration into any word, the X and Y drive lines for that word are selected. This attempts to put all ones (1's) into that word. When a zero is required, an inhibit line is raised for that bit position. This prevents that position setting to the one state by nullifying the effect of the Y half-select current. A read cycle must be taken before a write cycle to clear the storage word. This is controlled by CPU.

Reading Out a Location

- **X and Y drive lines selected for word addressed.**
- **Direction of current opposite to that in a write operation.**
- **Outputs from one-bit positions induced in sense lines.**

The X and Y drive line combination unique to a word is selected as for a write operation.

The drive current is in the opposite direction to that during the write phase. Cores that are in the one state flip to the zero state and a sense pulse is induced in the sense line for that bit position. Cores in a zero condition experience no significant change of state and no output is sensed. They remain in the zero state.

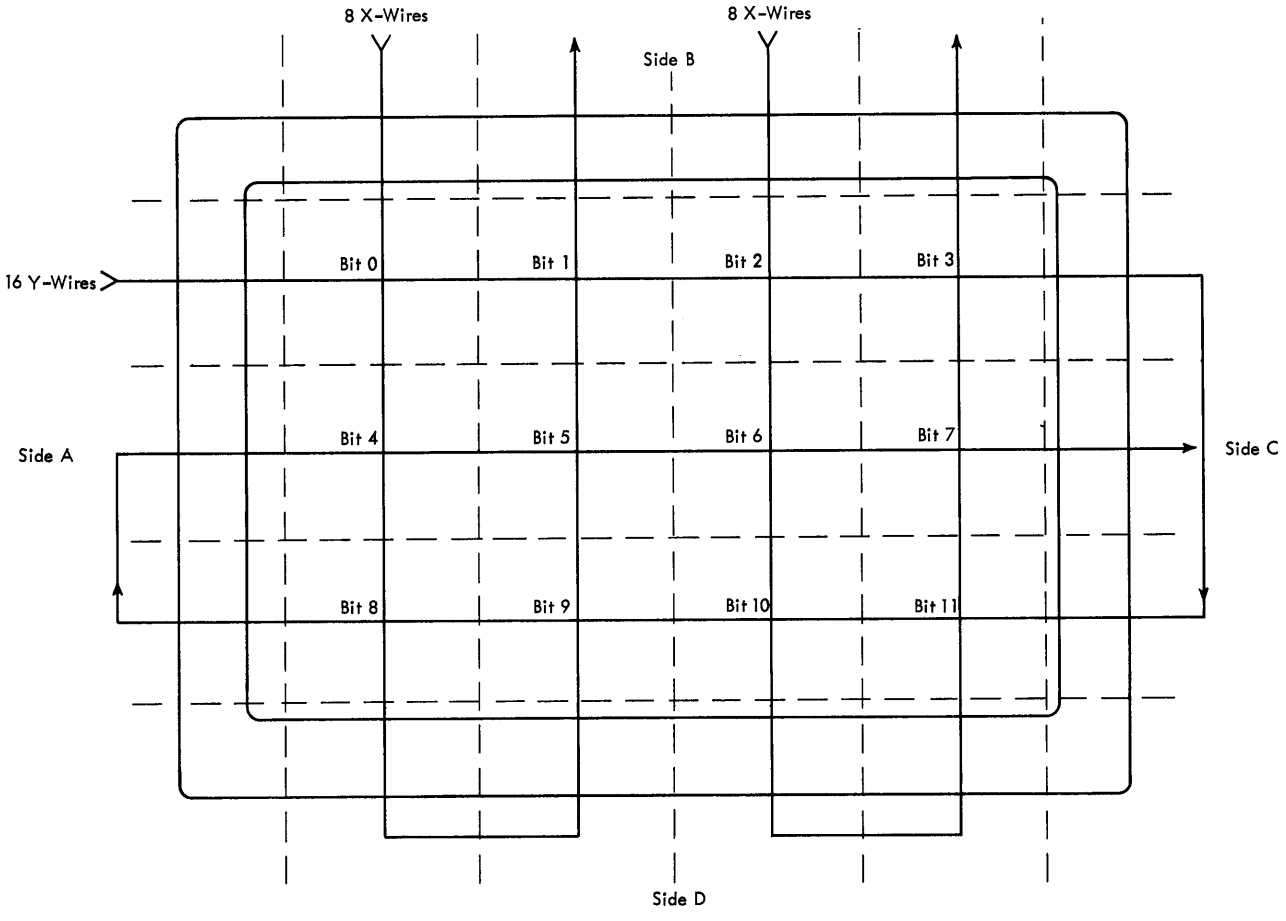


Figure 88. Storage Protect Core Plane Bit Sections

Addressing

- **SPLS addressed by SP address bus.**
- **SP address bus is eight bits wide.**
- **Bits 0 through 3 determine X address.**
- **Bits 4 through 7 determine Y address.**

The storage protect unit is addressed by the SP address bus which address bus has eight bits labeled L MAR 1-128. Figures 81 and 89 show the decoding.

Bits 0, 1, 2, and 3 are decoded to address the X lines; bits 4, 5, 6, and 7 are decoded to address the Y lines.

X decode address bits 0 and 1, together with Y decode address bits 4 and 5 address storage protect in 16 blocks of 16 words each. The words in each 16-word block are addressed by address bits 2 and 3 (X) and bits 6 and 7 (Y).

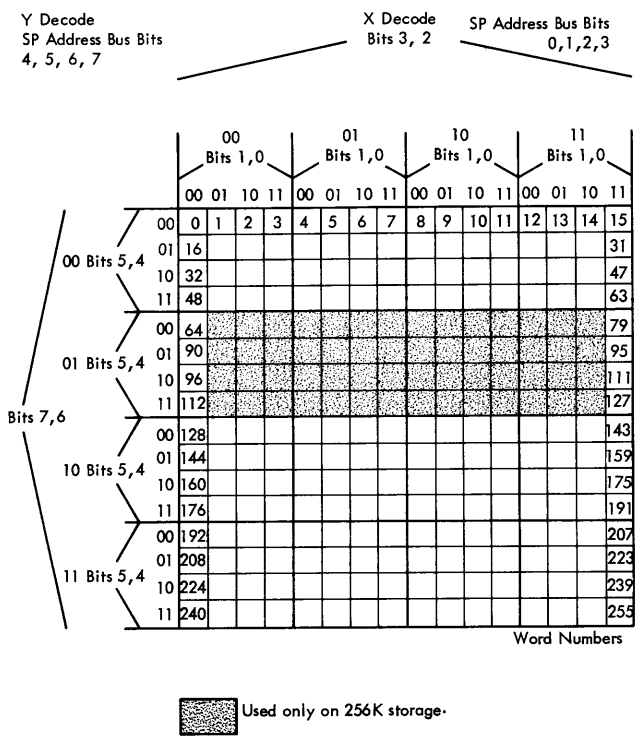


Figure 89. Storage Protect Address Decode

X-Y Drive

- **16 X lines and 16 Y lines.**

The 256-word storage protect unit has 16 X lines and 16 Y lines. The X lines are labeled X0 through X15 at one end and X0P (prime) through X15P at the other. The Y lines are labeled Y0 through Y15 at one end and Y0P through Y15P at the other.

X-Line Driving and Gating

- **16 X lines driven and gated in read and write directions by a matrix which has four write drivers and four read gates at one end and four write gates and four read drivers at the other end.**
- **SPLS address bus bits 2 and 3 define one write driver and one read gate.**
- **SPLS address bus bits 0 and 1 define one write gate and one read driver.**

Figure 90 shows the X-line driving and gating circuits.

To drive and gate the 16 X lines, both in the read and the write directions, a matrix is provided which has four write drivers and four read gates at one end of the lines and four write gates and four read drivers at the other end.

SP address bus bits 2 and 3 are decoded and each bit combination addresses a write driver and a read gate. The final selection between write driver and read gate is made by either the line labeled 'X write driver control' being active during a write cycle, or the line labeled 'X read gate control' being active during a read cycle.

SP address bus bits 0 and 1 are decoded and each bit-combination addresses a write gate and a read driver. The final selection between write gate and read driver is made by either the line labeled 'X write gate control' being active during a write cycle, or the line labeled 'X read driver control' being active during a read cycle.

The diodes in series with the X lines isolate a selected X line from the not-selected lines. The numbers over the blocks indicate the labels of the X line ends connected to these blocks.

Example: SPLS is to perform a read cycle. Address bus bits 0 and 1 contain 10.

Address bus bits 2 and 3 contain 01.

Address bus bits 0 and 7 = 10, together with X read driver control activate the read driver. The X line ends 4P, 5, 12P, and 13 are connected to this driver.

Address bus bits 2 and 3 = 01, together with 'X read gate control' activate the read gate. The X line ends 8, 10, 12 and 14 are connected to this gate.

The labels 12P and 12 are common to one X line and this line is selected as shown in Figure 90.

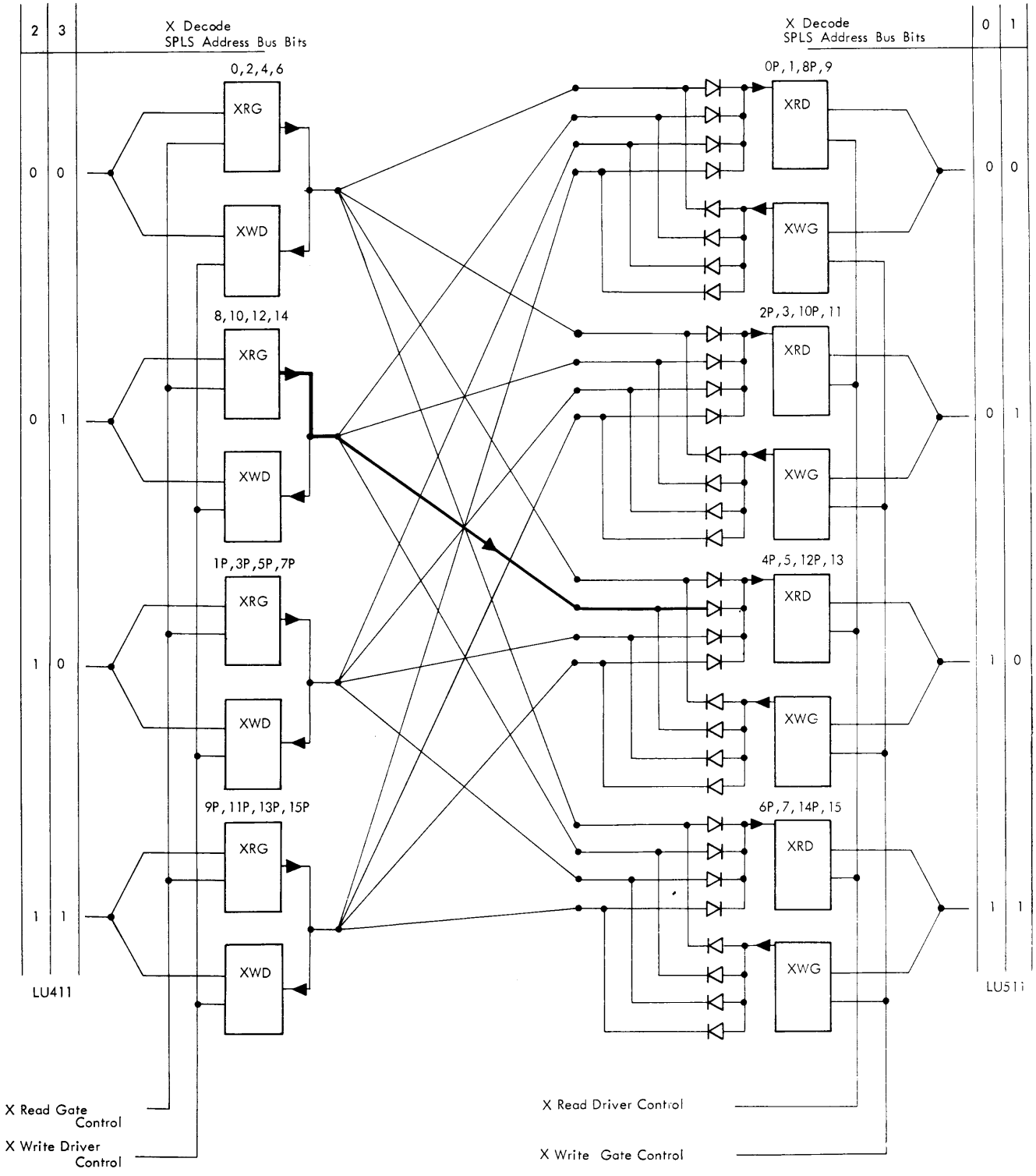


Figure 90. X Line Driving and Gating

Y Line Driving and Gating

Figure 91 is a diagram of the Y line driving and gating circuits.

The Y line driving and gating is similar to the X line driving and gating. The drivers and gates are addressed by SP address bus bits 4, 5, 6 and 7.

Sense

- **Five sense lines used.**
- **Sense lines are physically parallel to X lines in array.**

Figure 92 shows the sense circuit.

Five of the six available sense lines are used. The sense line for a particular bit position goes through every core within the bit section for that particular bit. The sense lines run parallel to the X lines. Both ends of each sense line are connected to a differential amplifier. When a pulse is sensed during a read cycle (sense gate line active), the corresponding bit in the SP data register is set to a one.

Inhibit

- **Five inhibit lines plus drivers and terminator resistors.**
- **Inhibit lines are physically parallel to Y lines in array.**

Figure 93 shows the inhibit circuit.

There are six 'bit sections' in the SP core plane of which only five are used.

Because an inhibit line is needed for each bit position, five inhibit lines plus drivers and terminator resistors are used.

The inhibit line for a particular bit position goes through every core within the 'bit section' for that bit position. The inhibit lines run parallel to the Y lines. The inhibit drivers are activated by the SP data register output during a write cycle ('inhibit timing' line active). A driver is activated when the corresponding SP data register bit position contains a zero (0).

Read and Write Controls

- **For every main storage read command, there is a corresponding SP read command.**
- **For every main storage write command, there is a corresponding SP write command.**
- **Two control signals: storage protect select and storage read control.**

The external controls for the storage protect unit must accomplish the following.

With every main storage read command there must be an SP read command. With every main storage write command there must be an SP write command.

To accomplish this, two control signals are generated: storage protect select and storage protect read control.

Storage protect select initiates the storage protect cycle set. It is active when the MS write latch is set or when the MS read latch is set. It is timed with P1 del. Not log 1 latch inhibits the selection of storage protect during log out.

Storage protect read control determines whether a read or write cycle is to be performed. It is active when the MS write latch is not on, i.e. when the MS read latch is on. When entering the storage protect ALD pages, 'storage protect read control' changes name to 'read-write control'.

Together the two control lines achieve the following:

When an MS read is called, an SP read cycle is also called (storage protect select active, storage protect read control active). When an MS write is called, an SP write cycle is also called (storage protect select active, storage protect read control not active).

Timing Logic

- **Provides timing for SP drivers and gates, inhibit drivers and sense amplifiers.**
- **Activated by two signals: storage select and read-write control.**

The SP timing circuit is shown in Figure 94.

The timing circuit provides the timing for the SP drivers and gates, the inhibit drivers and the sense amplifiers.

The timing circuitry is activated by two signals: storage select and read/write control.

The storage select pulse initiates the storage cycle. It is fed into a delay line and also sets the timing latch. The delay line provides a maximum delay of 500 nanoseconds. It is tapped at 50-nanosecond intervals along its length.

The timing latch is reset after 500 nanoseconds by the delayed storage select pulse.

The output of the timing latch is AND'ed with read-write control. When read-write control is active the AND condition is satisfied and read gate is activated, which times the read drivers and gates.

When read-write control is not active, write gate and inhibit timing are raised. Because of the delay introduced in the form of three inverters, write gate comes later than inhibit timing. Write gate times the write drivers and gates. Inhibit timing times the inhibit

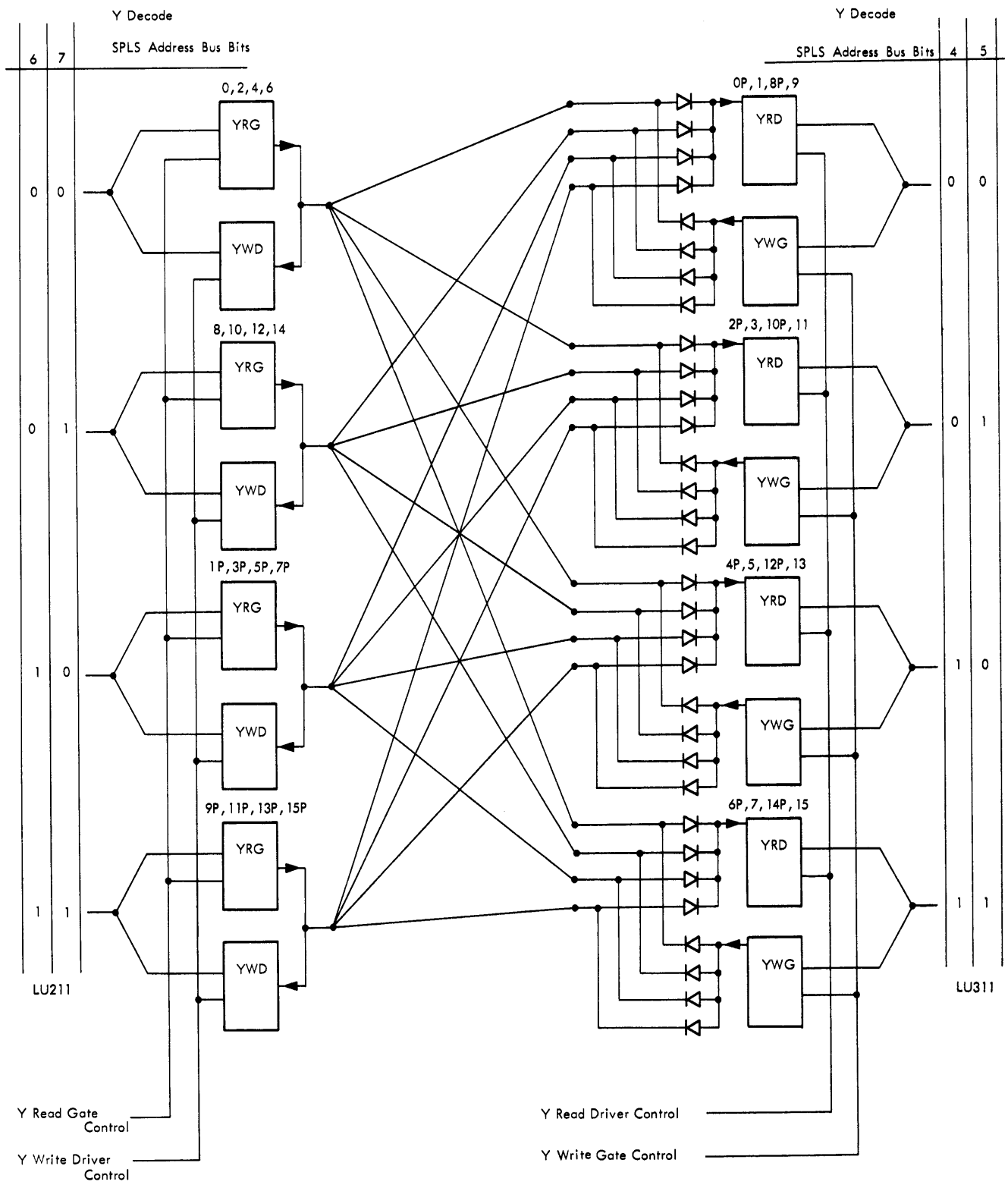
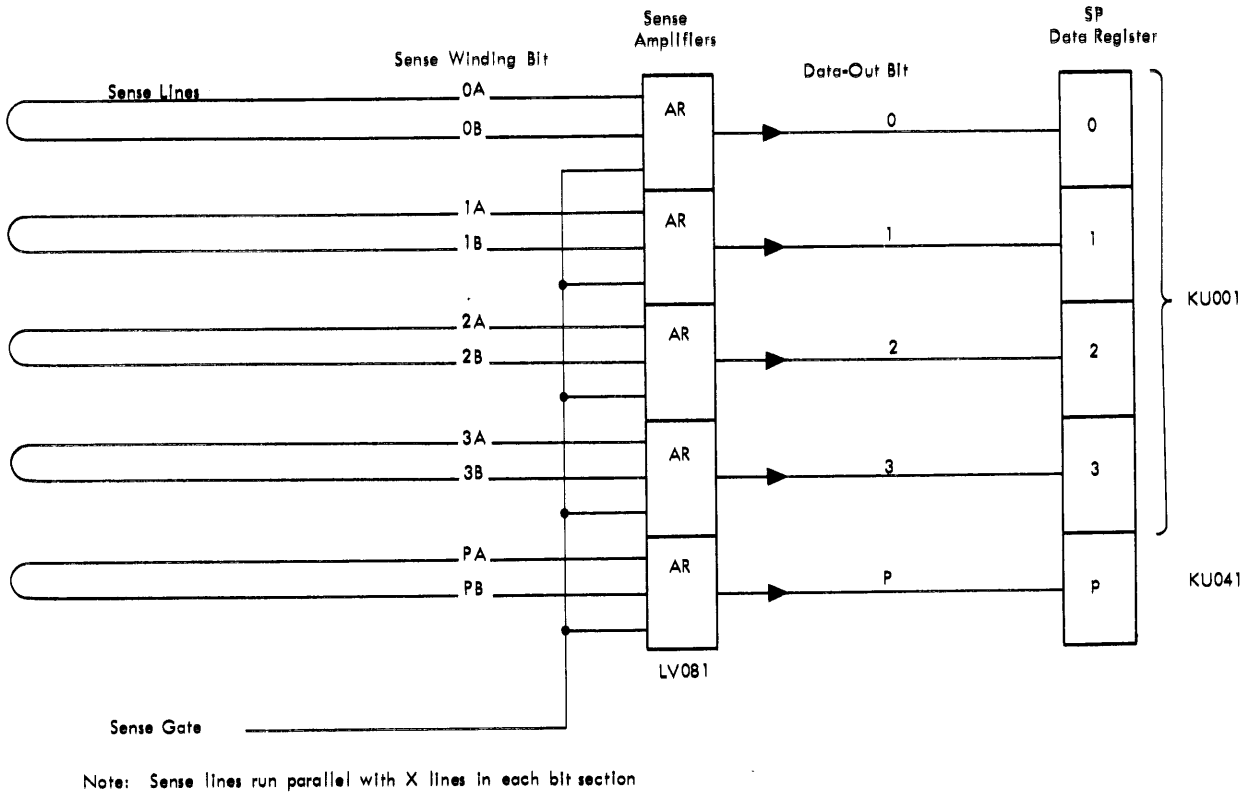


Figure 91. Y Line Driving and Gating



Note: Sense lines run parallel with X lines in each bit section

Figure 92. Sense Schematic

drivers. During read gate, a 200-nanosecond-delayed storage select pulse provides the sense strobe for the sense amplifier.

Read Cycle

- **Initiated by storage select pulse at P1 del.**

A read cycle is initiated by a storage select pulse at P1 del when read-write control is positive. At that time, the address is already available on the SP address bus and the proper X and Y drivers and gates are selected.

The read gate timing of 500 nanoseconds activates the driver-gate control circuits which, in turn, activate the address-selected read drivers and gates.

The X and Y drive currents start to flow in the read direction and flip addressed cores that were in the one-state to the zero-state. This induces a sense pulse in the corresponding sense lines. About 200 nanoseconds after the rise of the read gate timing pulse the strobe pulse, which is actually the delayed storage select pulse, allows the sense amplifiers to accept the sense pulse from the sense lines. A one appears at the sense amplifier output as a negative-going pulse.

Write Cycle

- **Read-write control drops when main storage write is called.**
- **Write cycle initiated by next storage select pulse.**

Read-write control drops as soon as main storage write is called. The next storage select pulse will initiate the write cycle. The address is available on the SP address bus and the proper X and Y drivers and gates are selected. The data to be written are available in the SP data register. Bit positions that contain a zero select the corresponding inhibit driver.

The write gate timing of 500 nanoseconds activates the driver-gate control circuits which, in turn, activate the address-selected write drivers and gates. The X and Y drive currents start to flow in the write direction.

Inhibit timing activates the data-selected inhibit drivers. Inhibit timing starts before write gate timing and ends at the same time as write or slightly later.

The X and Y currents try to flip all addressed cores to the one state. The inhibit half-current flows in the opposite direction through the cores that are to be

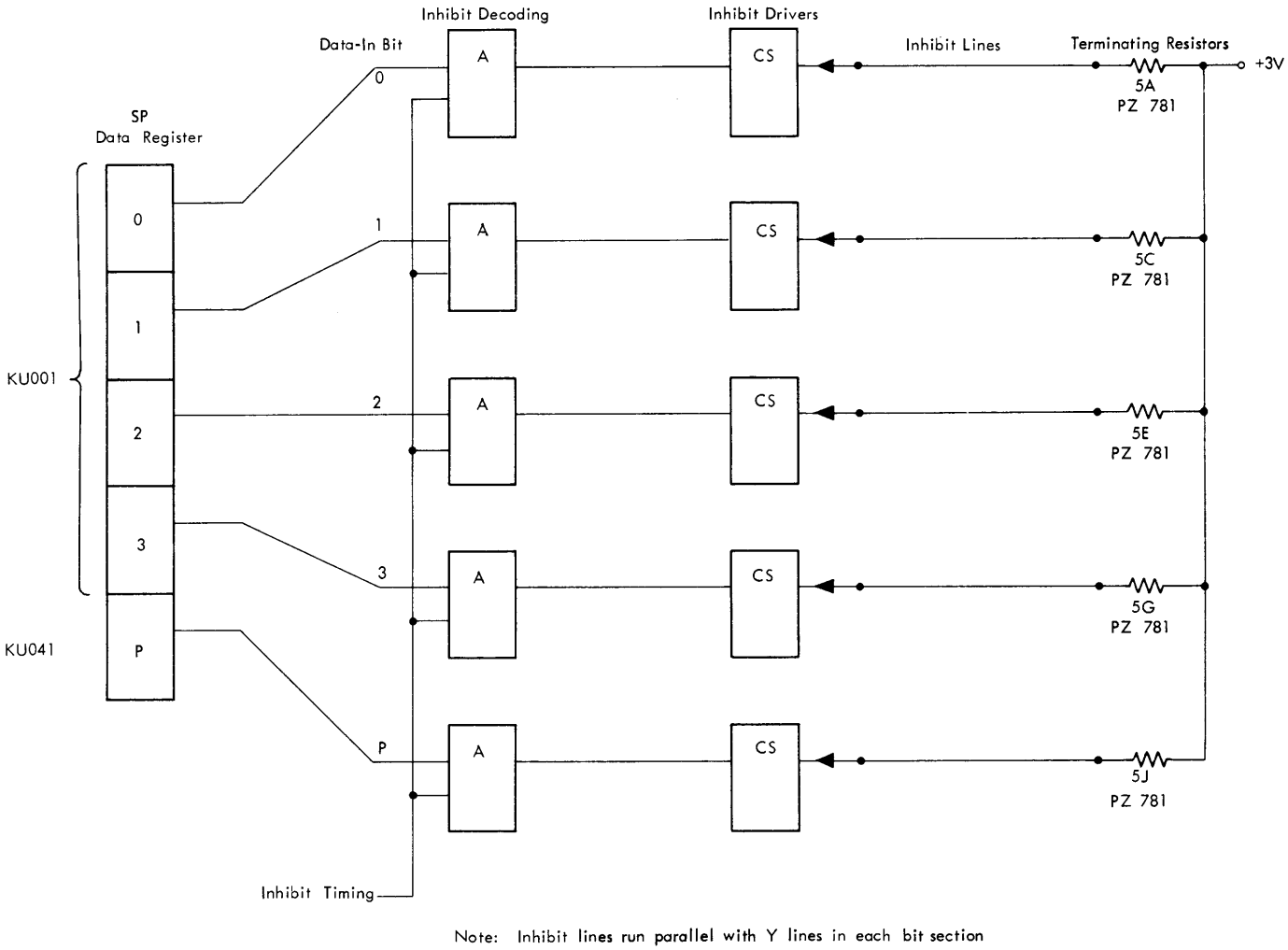


Figure 93. Inhibit Drive Schematic

kept in the zero state and prevents these cores from flipping to the one state.

Storage Protect Block Diagram

- **Timing controlled by feeding a storage select pulse through a tapped delay line.**
- **Tapping at 50 nanosecond intervals.**
- **Inhibit timing active during write cycle only.**
- **Sense strobe timing active during read cycle only.**
- **Write and read current amplitudes controlled by V reference.**
- **Value of V reference controlled by thermistor.**

Figure 95 is a block diagram of the circuitry associated with storage protect. The number on each line indicates the number of lines coming from one block.

The timing of the circuitry is obtained by feeding a storage select pulse, which is timed by the CPU P clock, into a delay line and tapping the delay line at different places. Each tap provides a delay of 50 nanoseconds.

The timing for the write gates and drivers is provided by the write gate timing circuit via the write gate and driver control circuits. The timing for the read gates and drivers is provided by the read gate timing circuit via the read gate and driver control circuits.

The line read-write control determines whether a read cycle or a write cycle is to be performed, by activating either the read gate timing circuit or the write gate timing circuit.

The inhibit timing circuit provides the timing for the inhibit drivers and is active during a write cycle only.

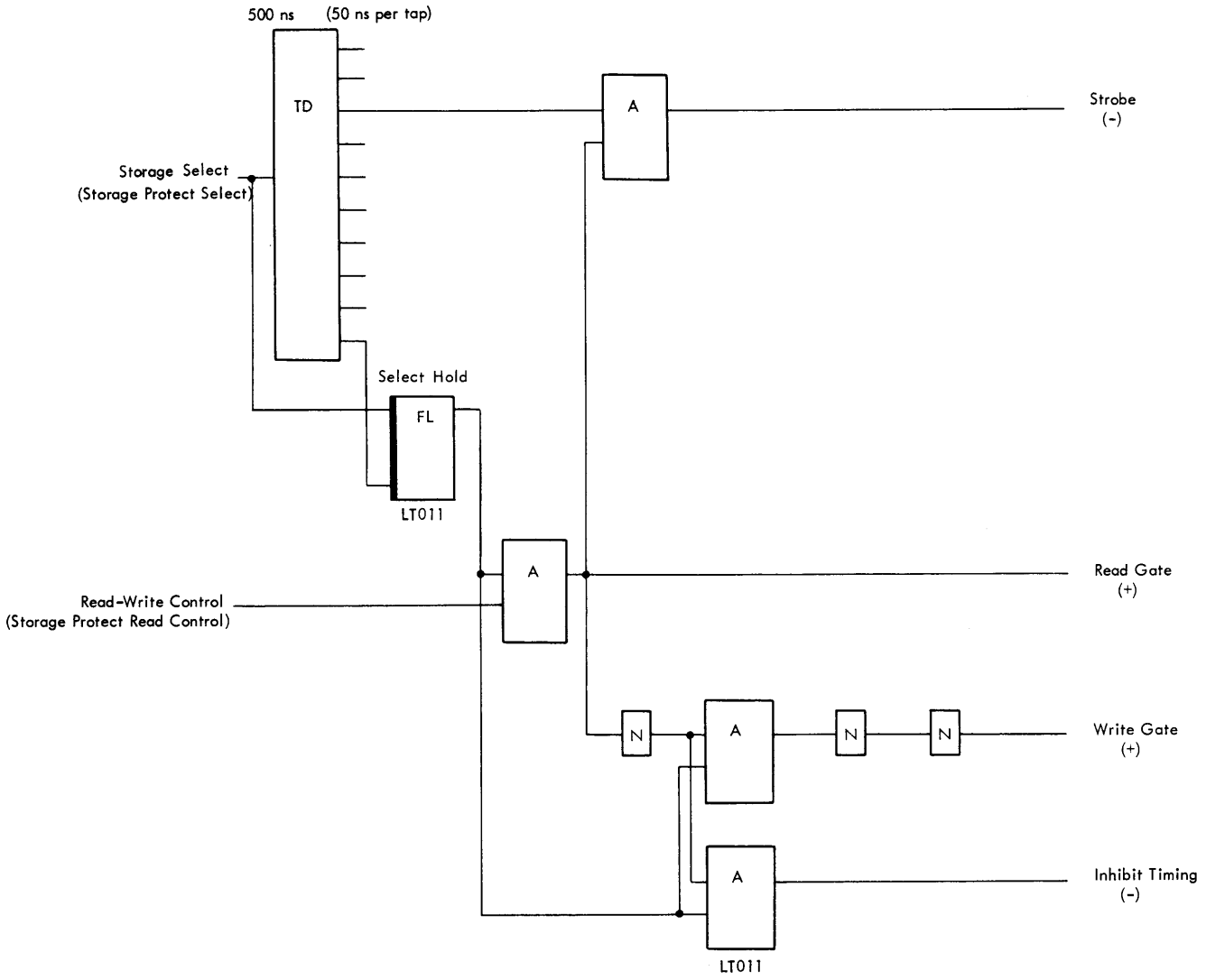


Figure 94. Storage Protect Timing Circuit

The sense strobe circuit provides the sense gate for the sense amplifier and is active during a read cycle only.

The X and Y write gate and driver control circuits are supplied with the write gate timing and also with a reference voltage, V ref.

The outputs write gate control and write driver control provide timing to the write gates and drivers.

The amplitudes of the currents in these lines are controlled by V reference and regulate the X, Y drive current supplied by the write drivers and gates to the drive lines. In this way, the X, Y drive currents are defined by the reference voltage V reference and have a nominal value of 240 ma.

The X and Y read gate and driver control circuits are supplied with the read gate timing and also with V reference. The output lines read gate control and read driver control are fed to the read gates and drivers respectively. Their function is the same as that of write gate control and write driver control.

The reference voltage is provided by the V reference generator. To adjust the X, Y drive currents, V reference can be varied within certain limits by a potentiometer on the CE panel.

V reference is also controlled by a thermistor, located near the core planes, to compensate changes of drive

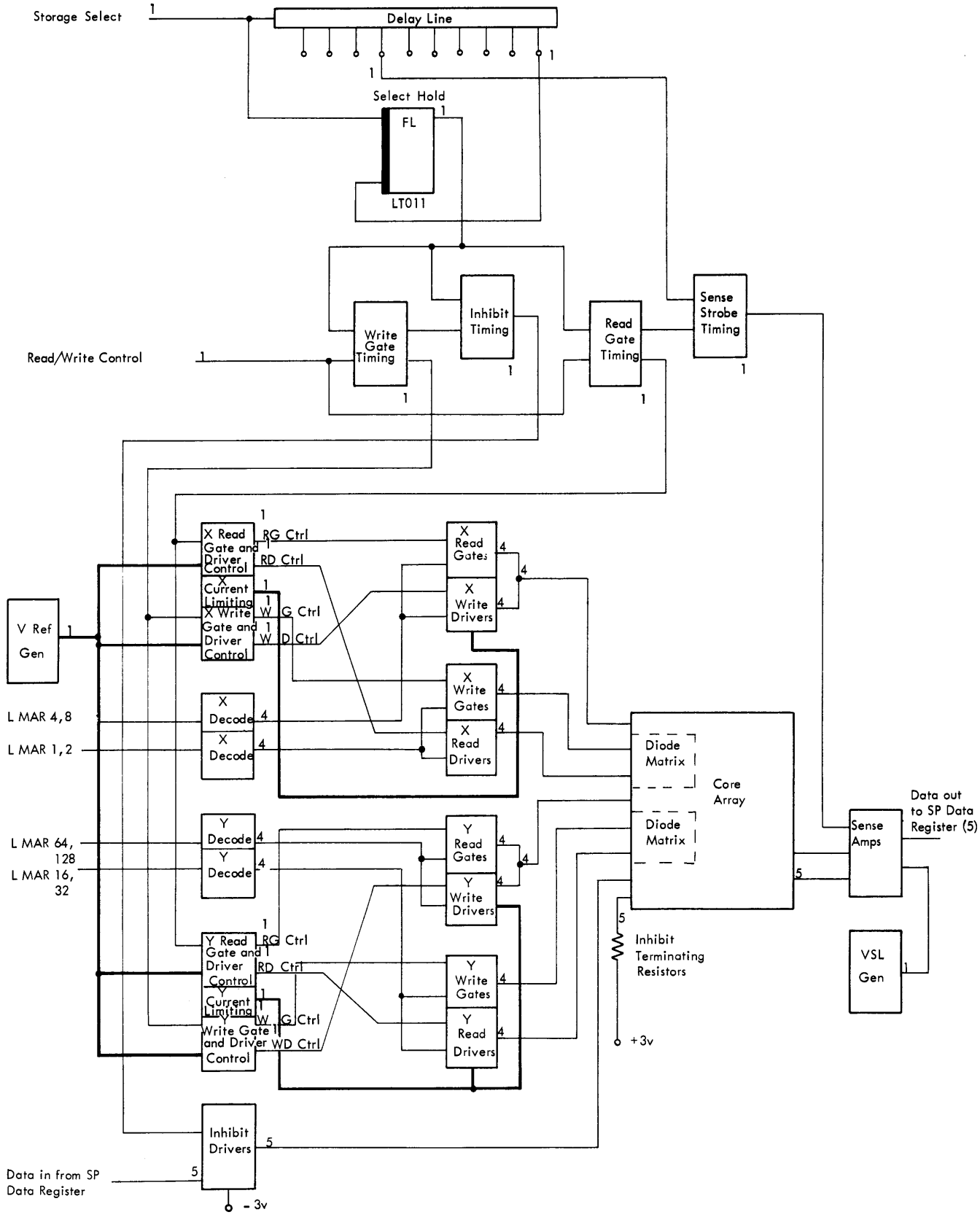


Figure 95. Storage Protect

- **Core storage unit.**
- **2.5 microsecond read-write cycle.**
- **Split read-write cycle.**
- **Halfword (two bytes) operation.**
- **Random access.**
- **SLT circuits and packaging.**
- **Three-wire system.**

NOTE: The words storage and memory are used interchangeably between this instruction manual and the ALD's. For instance, the line storage select in this manual may become memory select in the ALD's. Beware of this when going from figures in this manual to the ALD's, and vice versa.

The main storage unit for the IBM 2040 is a 2.5 microsecond, random access, magnetic core storage unit.

Main storage accesses for a read or a write operation are completely independent and must be called individually by microprogram. This is known as a split read-write cycle.

The storage unit will accept and retain data input from the central processing unit (CPU) or from input/output units (I/O units) and deliver data to the CPU or the I/O units. The storage unit handles all input and output as data, making no distinction between program instructions, control words, or numbers. Each reference to storage either stores or reads out one halfword (two bytes).

The storage unit operates within a temperature range of 15°C to 32°C (60°F to 90°F) and within a humidity range of 20 per cent to 80 per cent.

The inhibit and the sense functions are performed physically by the same wire used as an inhibit line on a write and as a sense line on a read operation. This system is known as the three-wire system, the X drive, the Y drive, and the inhibit or sense line.

Capacity

- **Six sizes — 16, 32, 64, 128, 192, 256 K bytes**
- **1K bytes means 1,024 bytes.**
- **Multiplex storage additional for multiplexor channel control words.**

Main storage consists of two sections:
1. The actual main storage, and
2. An additional section called multiplex storage.

Multiplex storage is accessible only by the microprogram and is used to hold the unit control information for the multiplexor channel. Multiplex storage is not accessible by the machine-language program.

A main storage unit contains two arrays, each array capable of storing a maximum of 64K bytes.

Main storage is available in five sizes. Multiplex storage size increases with main storage size to a maximum of 2,048 bytes.

	MAIN STORAGE		MULTIPLEX STORAGE
MODEL	BYTES	HALFWORDS	BYTES
D	16,384	8,192	256
E	32,768	16,384	512
F	65,536	32,768	1,024
G	131,072	65,536	2,048
GF	196,608	98,304	2,048
H	262,144	131,072	2,048

In order to provide the maximum main storage size of 262,144 bytes, two storage units, each of 131,072 bytes are used.

In either a read or a write cycle, two bytes (18 bits) are handled simultaneously. Figure 96 shows that any storage word can be immediately located (random access) by its co-ordinates (address).

Speed

- **1.4 microseconds — read.**
- **1.1 microseconds — write.**
- **Split-cycle operation with a write cycle always preceded by a read cycle.**
- **Access time 1 microsecond.**

Cycle time is the time required for reading out or writing a halfword.

In a read cycle, a halfword is read out of storage. One read cycle takes 1.4 microseconds. In a write cycle a halfword is written into storage. One write cycle takes 1.1 microseconds.

There may be a gap of an indefinite period following either a read or a write cycle, but a write cycle must always be preceded by a read cycle. The read-write sequence is not automatic, but is controlled by microprogram.

The maximum time between the storage select pulse reaching the main-storage logic and the data leaving the storage unit is called access time. The access time is 1 microsecond.

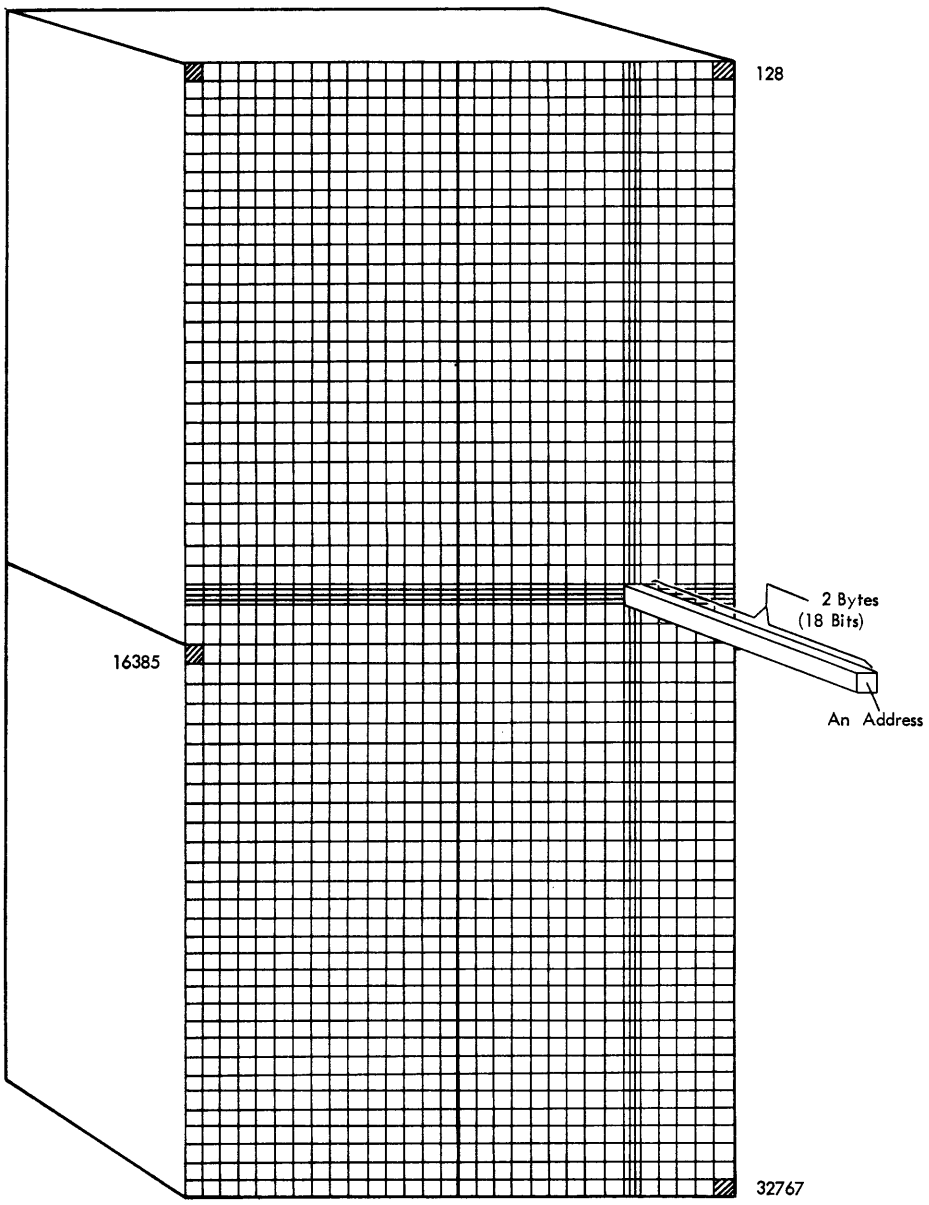


Figure 96. Core Storage Locations

Storage Cycle

- **System specifies read or write main-storage operation.**
- **Asynchronous operation.**
- **System provides start signal.**
- **System provides address.**
- **System data register provides data to be stored and accepts data read out.**

Main-storage logic circuits control the reading or writing of data. The system provides the main-storage logic circuits with a command (read or write) and a start signal. Once started, the storage cycle proceeds under control of the storage unit clock independently of the system timing.

The address provided by the system is decoded to determine the co-ordinates, called X and Y, of the addressed location. Together, X and Y completely identify the specified location.

In a read cycle, the contents of the addressed locations are withdrawn from storage and placed into the system's data register. This leaves the addressed location empty (all zero). In a write cycle, the data in the system's data register are stored in the specified location which was previously set to zero by the read operation.

At the end of a read operation the status of the storage unit stays in read until a write operation is started. At the end of a write operation the status of the storage unit is reset to read.

In the following sections only the main storage Model G will be discussed; other models operate similarly.

64K Halfword Main Storage

Address Registers

- **Main storage addressed via storage address bus (SAB).**
- **Registers A, S1 or S2 gated to SAB.**
- **16 bits are needed on SAB to address 64K halfwords.**
- **17 bits are necessary in the address register to address 128K bytes.**
- **Multiplex storage is addressed when the Mpx stat (Y1) or the 'Mpx store switch' is on.**
- **Translate storage address lines are used for the 1401-1410 emulator feature.**

The storage address bus is used to generate the X and Y lines for main-storage addressing. Information on the SAB can be obtained from the A register during CPU operations.

During CPU operations the lines "CPU to SAB not RTPT" and "not use bump address" are active.

Note that bit 6 of the AX register is not used for addressing a 64K halfword main storage. Additionally, as bit 7 of the address register forms the units position of the address, it is never used on the SAB. This bit is used for byte determination and main storage is addressed by halfwords. Therefore, a 17-bit output from the A register is reduced to 16 bits on the storage address bus for a 64K halfword main storage.

The Mpx stat must be on before multiplex storage can be addressed. If the multiplex storage is to be addressed from the console the multiplex store switch must be on.

The bits from the A register are gated to SAB with a change in pattern. As SAB is parity checked, all bits of the A register, with the exception of the units bit, must be gated onto SAB (see section on Parity Checking). This is the case even when all bits are not required as when Mpx storage is addressed.

A special line called address Mpx store is made active to enable the multiplex storage to be addressed.

The S1 register is gated on to SAB when selector channel 1 is being handled under microprogram control. In this case, the line RTPT for SC1 is active. Similarly the S2 register is gated on to SAB for selector channel 2 microprogram operations.

Note that the line CPU to SAB not RTPT when RTPT (reinterpret) is active during a dump or log out routine. This is necessary because the A register must always be in control of main-storage access if a dump or log out occurs while the function RTPT is active.

For channel data service the channel SC1 or SC2 select early conditions are present, activating the gating of the A register to SAB even if the channel breaks in during a dump operation. The A register is forced back into control only during log out because the line 'CPU to MAB not RTPT' cannot be deactivated during log out. This is more fully described in the section dealing with selector channel operations in the *System/360 Model 40 Theory of Operations Manual,* Form 223-2844.

Note: In the ALD's, translate storage address can be found entering the main-storage addressing logic. These lines are used for the 1401-1410 emulator feature.

available in each array. To select the remaining 16,384 storage words, phase reversal is used.

The 64 X wires in segment A are connected to the 64 X wires in segment B and the 64 X wires in segment C are connected to the 64 X wires in segment D. See Figure 101. Figure 102 shows that the direction of the current through the selected Y line determines whether a word in section A-C or section B-D is addressed as described in the following two paragraphs.

Read Cycle

- **X current always in normal read direction.**
- **Y current in read direction for segment A-C.**
- **Y current reversed for segment B-D.**

The X current is always in the normal read direction. When selecting cores in segment A-C, the Y current is in the normal read direction. When selecting cores in segment B-D, the direction of the Y current is reversed, i.e., is in the normal write direction.

Write Cycle

- **X current always in normal write direction.**
- **Y current in normal direction for segment A-C.**
- **Y current reversed for segment B-D.**

The X current is always in the normal write direction. When selecting cores in segment A-C, the Y current is in the normal write direction. When selecting cores in segment B-D the direction of the Y current is reversed.

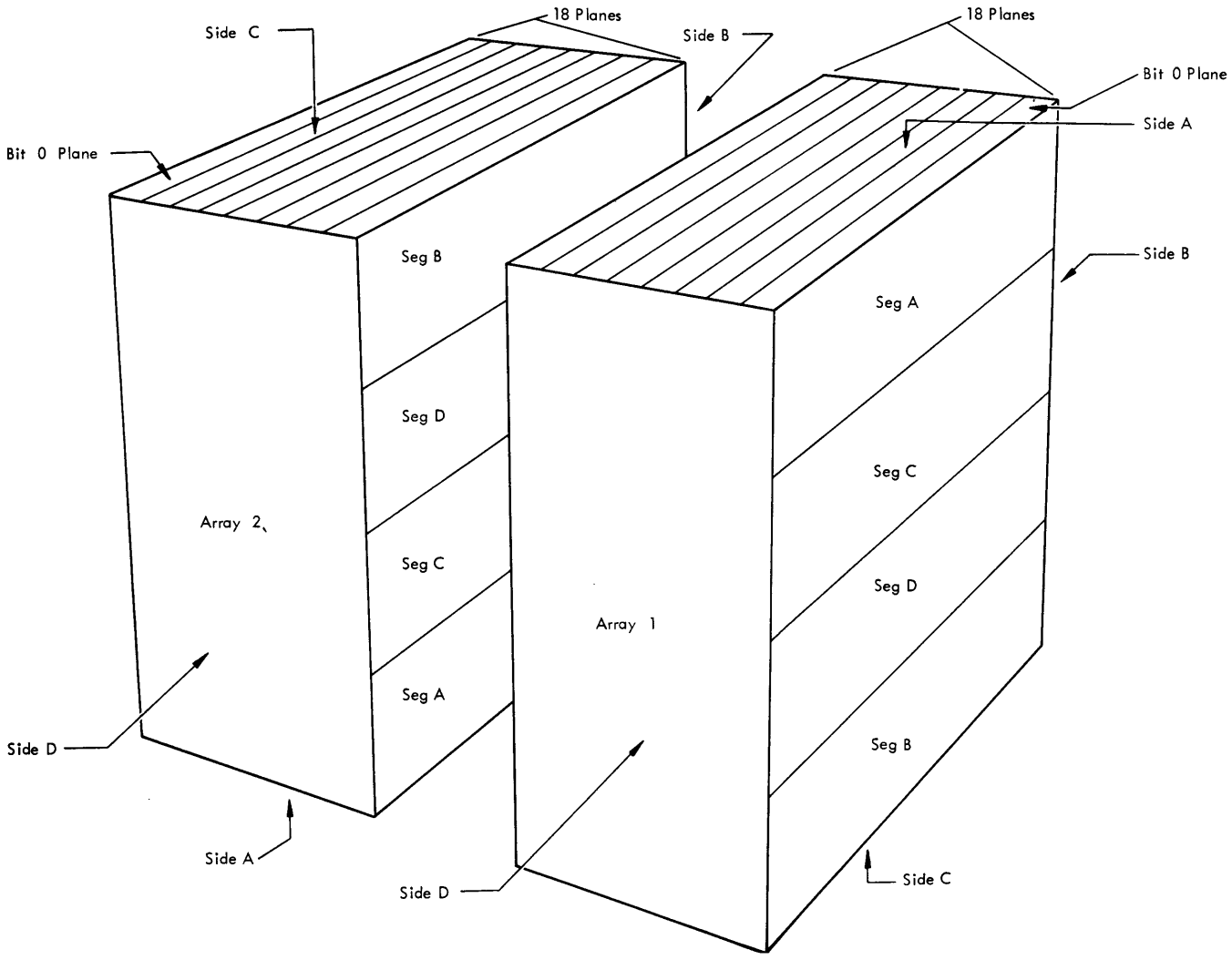


Figure 98. Main Storage Arrays

To achieve this, the direction of the Y current is determined not only by the read-write control but also by the main storage address bus bit 6 as follows:

Read cycle
- Bit 6 = 0 Y current in normal read direction
- Bit 6 = 1 Y current in reversed read direction.

Write cycle
- Bit 6 = 0 Y current in normal write direction
- Bit 6 = 1 Y current in reversed write direction.

Figure 103 shows the main storage address bus bit allocations.

Drive Scheme

- **Four basic circuits.**
- **Four drive-line terminating resistors.**
- **One of four terminator gate circuits switches a terminating resistor to end of selected drive line.**
- **Terminator gate turned on first, off last.**
- **Read-write driver turns array current on and off.**
- **Gates act as matrix to select proper line for each address.**
- **Direct drive (one transistor to drive each line).**
- **All lines charged to +60v when not driven.**

The storage unit drive scheme is built of four basic circuits, the terminator gate, the gate, gate decoder, and the driver.

The relationship of the four circuits and the drive line is shown in Figure 104. The terminator gate functions as a high-speed switch to connect the terminating resistor to the array line. This resistor determines the final value of the drive current. The gate and gate decoder work together as a unit to select one array drive line. The driver functions as a current switch to turn array current on or off through the drive line.

In the undriven state, the array line is charged to +60v via the resistor in parallel with the terminator gate transistor. The terminator gate is turned on before the driver. When the driver is turned on, the gate is also switched on and drive current starts to flow.

The driver is always the first circuit to be switched off. The terminator gate remains on until the current in the terminating resistor has decreased to zero and the voltage of the line has returned to +60v. Figure 104 also shows the basic timing of the circuits.

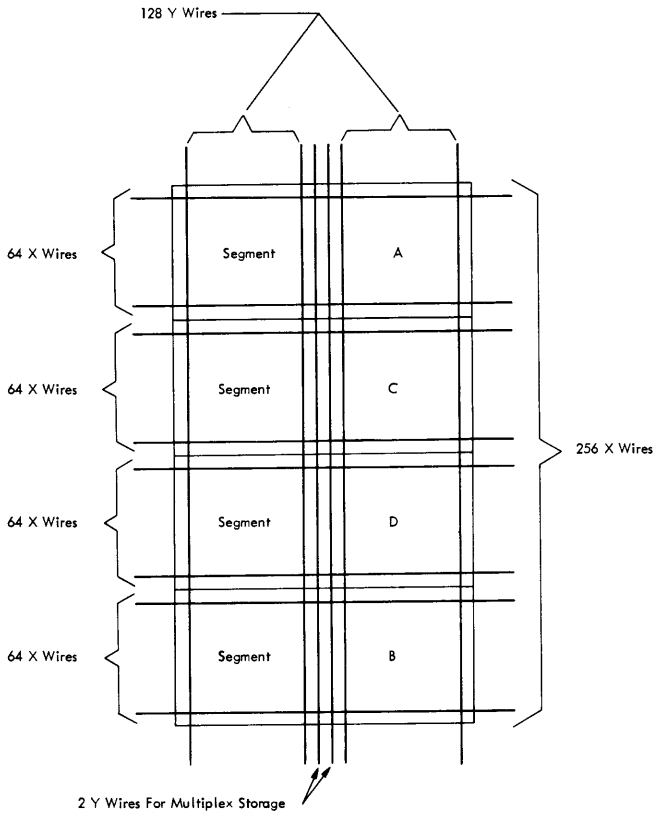


Figure 99. Plane Layout

Selection of One Driver Line (Figure 105)

- **One driver line driven with half-select current of 360 ma.**
- **Drive is bi-directional.**
- **Line individually driven by one diode and one gate.**
- **Diode driven in 16 groups of 16.**
- **Gates divided into 16 groups of 16 write gates and 16 groups of 16 read gates.**
- **All gates driven by only 16 gate decoders.**

The basic selection of one X and one Y drive line is described in only one dimension. One of 256 individual lines is selected and driven with a half-select current of approximately 360 ma. Each drive line passes through 256 cores in each of 18 planes, presenting a character-

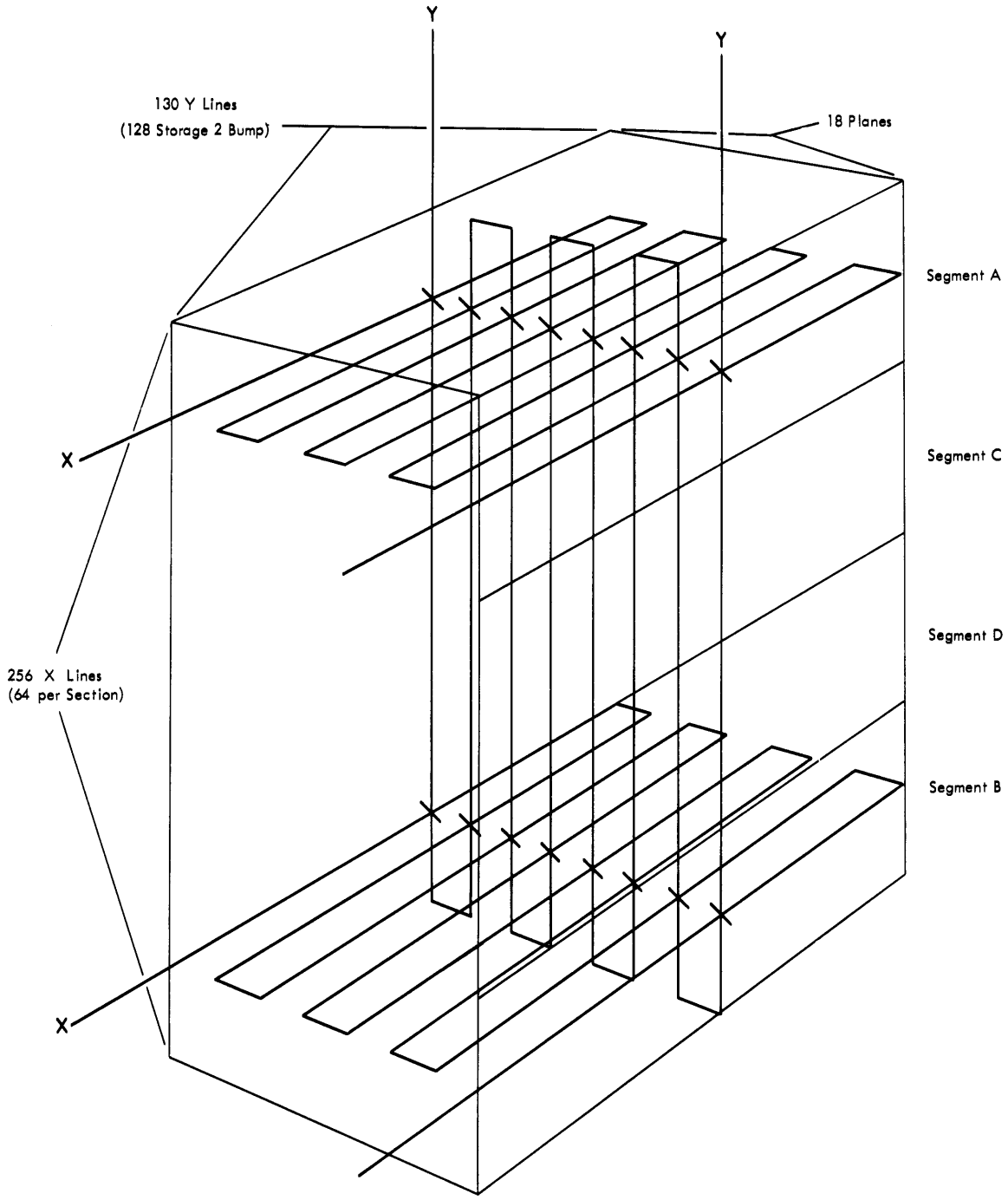


Figure 100. Routing of X and Y Wires

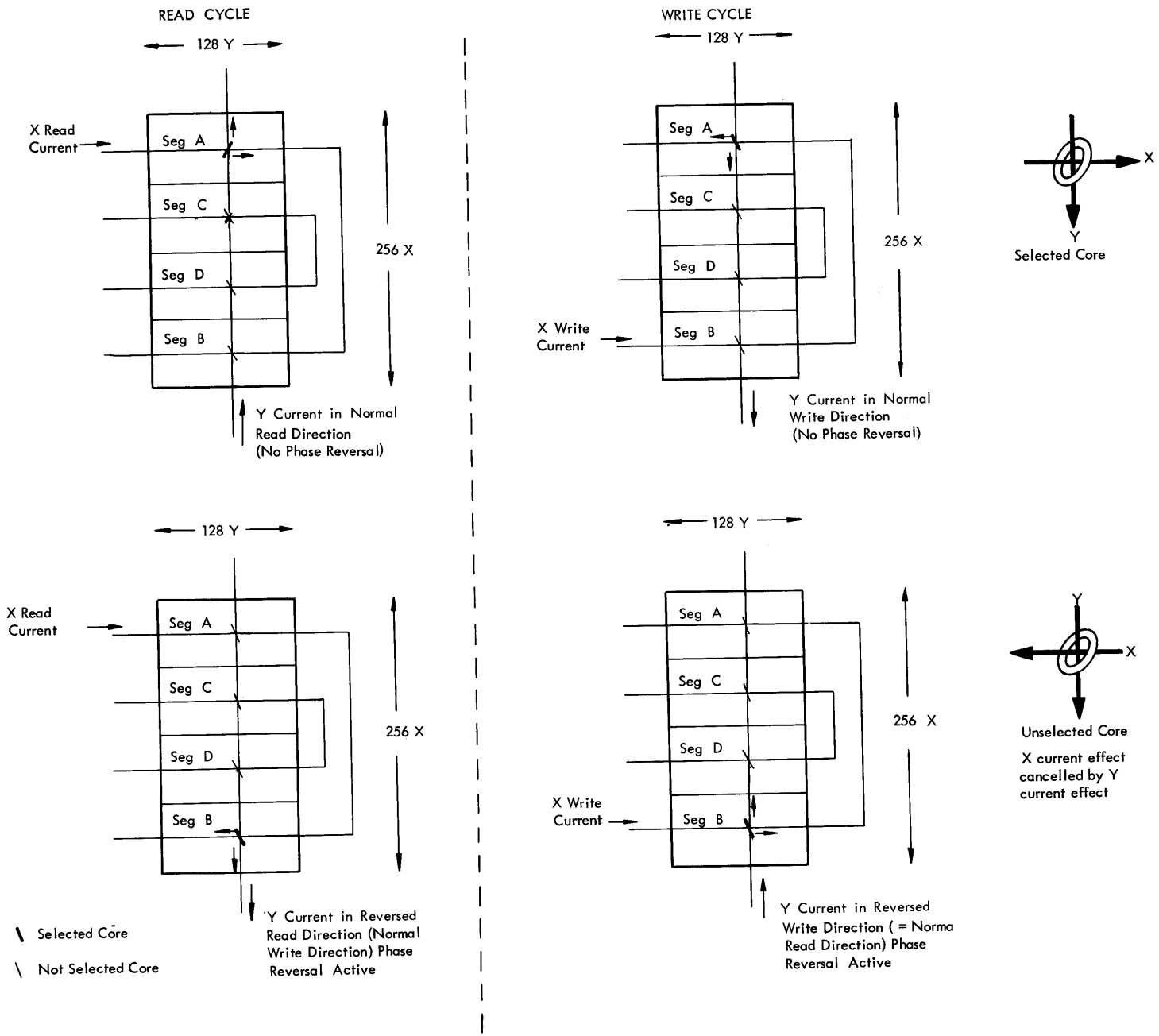


Figure 102. Phase Reversal

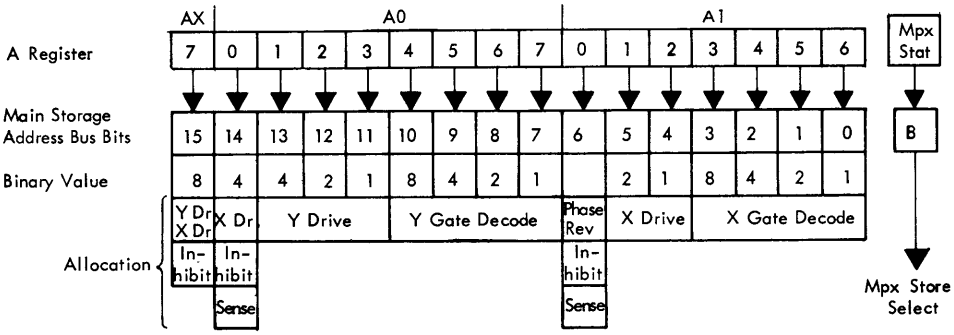


Figure 103. Storage Address Bus Bit Allocation

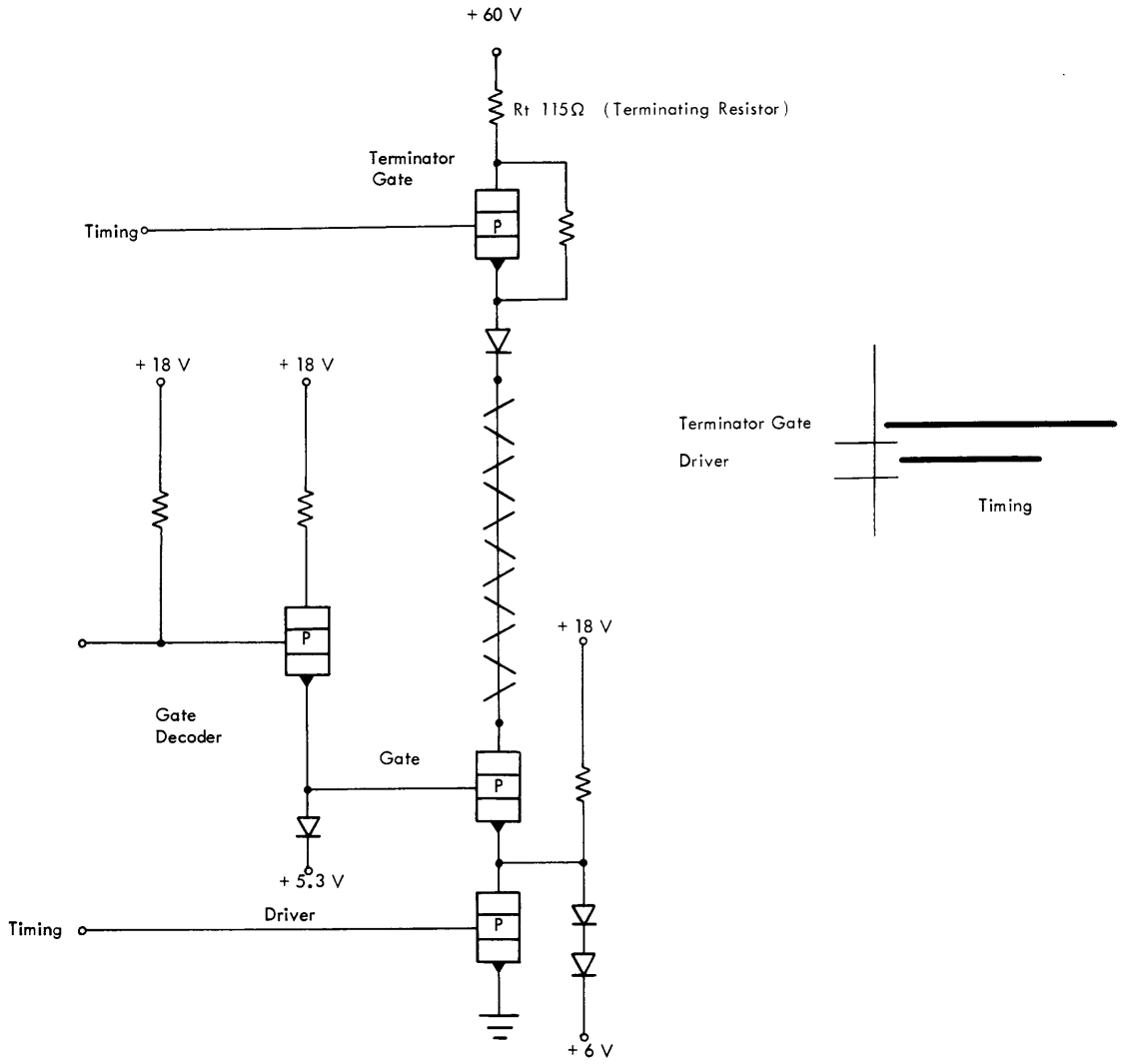


Figure 104. Connection of Four Basic Drive Circuits

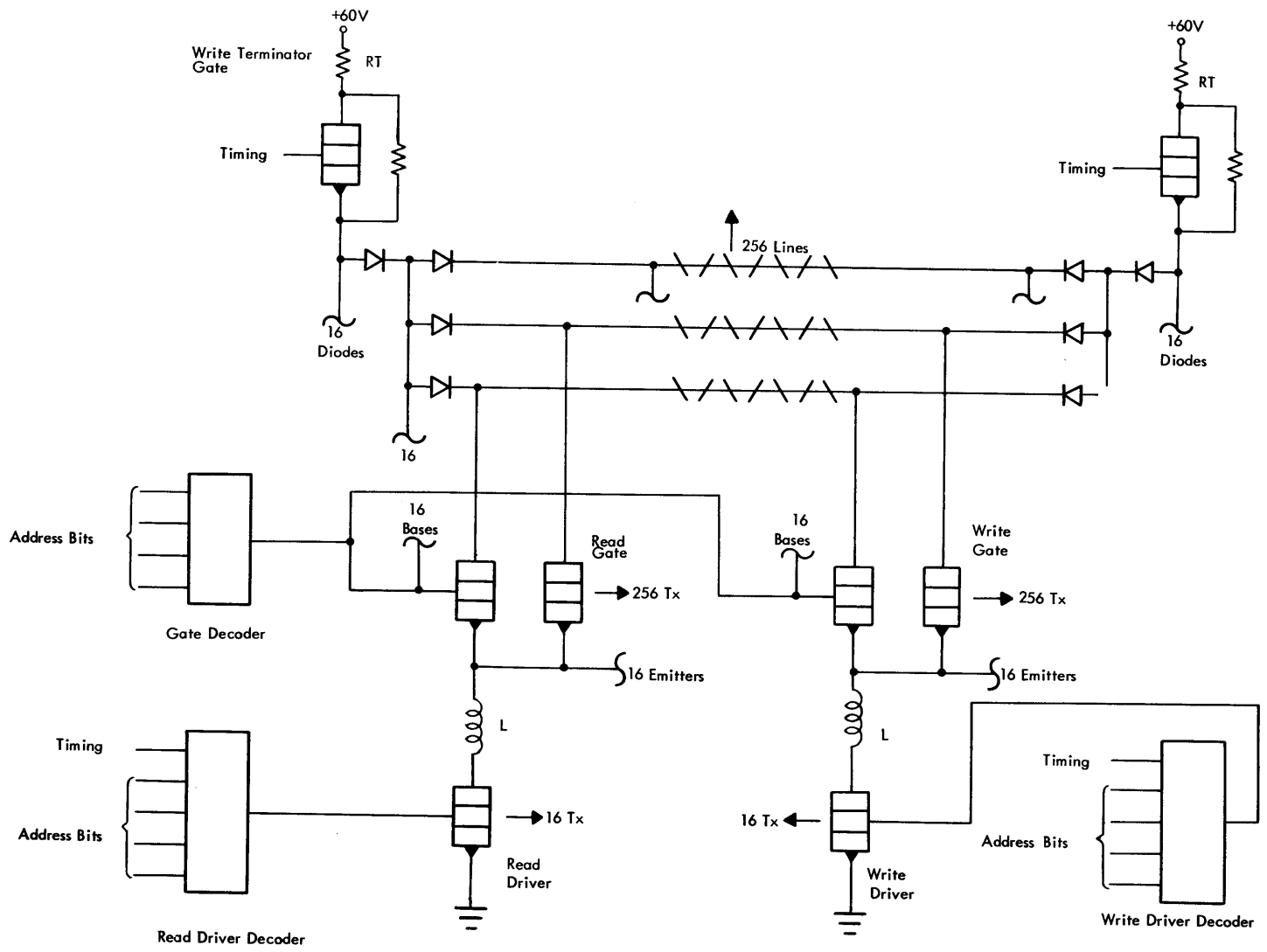


Figure 105. Main Storage Unit Drive

istic impedance of approximately 180 ohms and a transmission delay of 45 nanoseconds. The drive is bi-directional to provide read and write current at each address.

The 256 lines are driven individually with a diode and a gate transistor connected to each end of each line. The diodes are driven in 16 groups of 16 by additional diodes to reduce the capacitance seen by any one line when that line is driven. The diodes are connected to the terminator gate circuits that switch a terminating resistor to the appropriate end of the driven line.

The gates are divided into 16 groups of 16 write gates and 16 groups of 16 read gates. The emitters within a read or write group are connected and are driven by one read or write driver. One of the 16 read or write drivers is selected by decoding four bits of the address.

The base of the first transistor in each group of read gates and the base of the first transistor in each group of write gates are connected. Corresponding transistors in each group have their bases similarly connected. Each of these strings of 32 transistors is driven by one gate decoder. Thus, all 512 read and write gates in either the X or the Y dimension are driven with only 16 gate decoders.

A total of 32 driver circuits is required for each dimension.

The gate decoder translates four bits of the address into 16 separate lines. Each line in turn selects one group of gate transistors. One of the gate transistors of the selected group is turned on by a read or write driver.

Current Path

To drive current in the read direction, the following actions occur.

1. The gate decoder decodes four bits of the address to select a group of read gates.
2. The read driver decoder decodes four bits of the address to select a driver.
3. The read terminator gate is turned on.
4. The selected read driver is turned on.

The current path is to the +60v supply through the terminating resistor, terminator gate, two diodes, the array drive line, a read gate, and through a read driver from ground. See Figure 105.

In the undriven state, all drive lines are charged to +60v. The resistors in parallel with the terminator gate transistors provide leakage current for the read and write gates to keep the lines charged. This is necessary to avoid forward-biasing diodes connected to unselected lines when a terminator gate is turned on.

When a line is driven, the voltage in that line drops, reverse biasing the diodes of the 15 lines of the same group and also the diodes connected to the terminator gates for the other groups of lines. In this way, the load capacitance formed by the not-selected lines is isolated from the selected line, resulting in a faster rise of the array current.

Termination

The terminating resistor is approximately 115 ohms. This value is a compromise between the requirements of obtaining a good wave-shape and keeping the supply voltage as low as possible to avoid breakdown.

The effect of this termination is to limit the initial rise of current to about 85 percent of its final value. Initial rise time is 100 nanoseconds, determined by the inductance in the collector circuit of the driver. The routing of the connecting leads between the gate emitters and the driver collectors has been adjusted to provide the correct inductance.

X Drive

- **Address bits 0 through 3 determine gate.**
- **Address bits 4, 5, 14, and 15 select one read driver with X read or one write driver with X write.**

The X drive scheme with typical routing of X drive lines through the arrays is shown in Figure 106. Address bits 0-3 are fed to the gate decoder. Address bits 4, 5, 14 and 15 are used to select one of the 16 read drivers when X read control is active and one of the 16 write drivers when X write control is active. Note that address bit 15 determines whether array 1 or array 2 is used.

Y Drive

- **Address bits 7-10 determine gate.**
- **Address bits 11, 12, 13, and 15 select one read driver with Y read or one write driver with Y write.**

The Y drive scheme is shown in Figure 107. Address bits 7-10 are fed to the gate decoder. Address bits 11, 12, 13, and 15 are used to select one of the 16 read drivers when Y read control is active and one of the 16 write drivers when Y write control is active.

Note that Y read or Y write depends not only on whether a read or a write cycle is to be performed but also on the state of main storage address bit 6. Note that address bit 15 determines whether array 1 or array 2 is used.

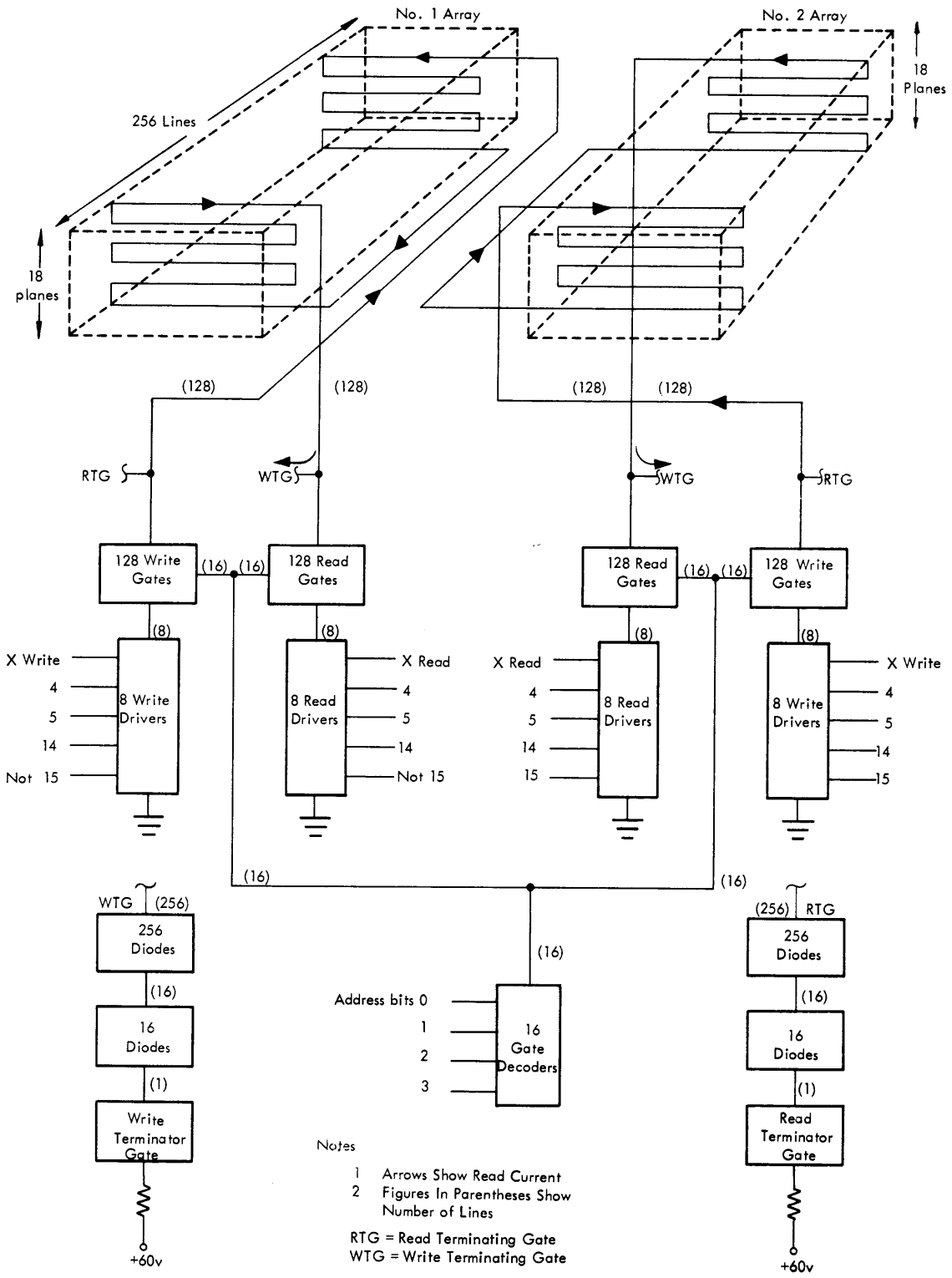


Figure 106. Storage X Dimension Drive

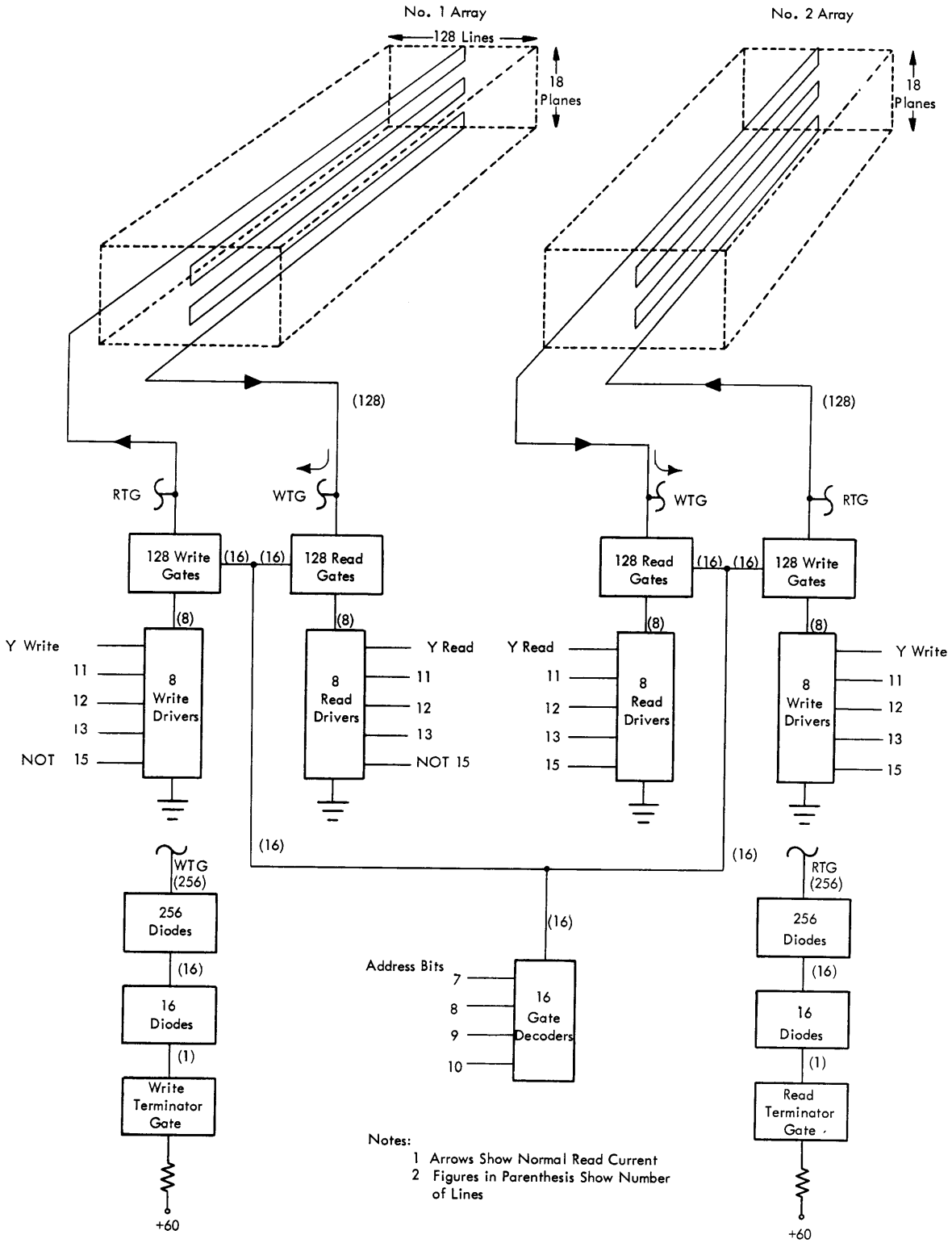


Figure 107. Main Storage Y Dimension Drive

Common Sense/Inhibit Line

- **Main storage uses three-wire system.**
- **Sense/inhibit line is a pair of wires parallel to X lines.**
- **Both legs grounded at one end.**
- **Other ends feed both inhibit driver and sense amplifier.**
- **Inhibit drivers activated on write when corresponding data bit is zero.**

In main storage, each core is controlled by three wires, one X, one Y and one wire to perform the combined functions of sense during the read cycle and inhibit during the write cycle. See Figure 108.

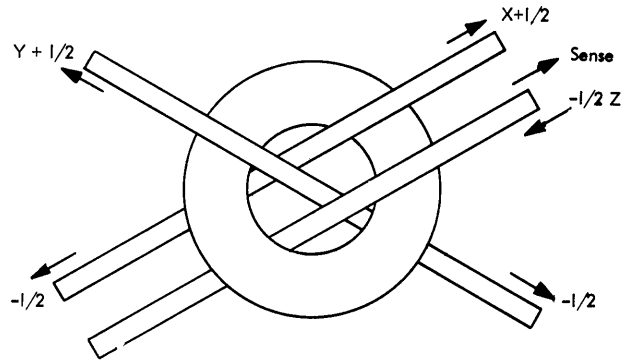


Figure 108. Core Wires (Three-Wire System)

In each segment of each plane, the sense inhibit line is a pair of wires threaded in a zigzag pattern parallel to the X lines so that there are 4160 cores on each leg (64 per leg for multiplex storage). At one end of the line, both legs are grounded; the other ends are connected to both the inhibit driver and the sense amplifier.

For each segment in each plane there is an inhibit driver and a sense amplifier. Eight inhibit drivers and eight sense amplifiers are allocated to the same bit, four of each per plane for array 1 and four of each per plane for array 2. See Figure 109. Thus in the 64K halfword storage there are 144 inhibit (Z) drivers and 144 sense amplifiers ($2 \times 4 \times 18 = 144$).

The inhibit drivers allocated to one bit can be activated when the corresponding bit in the data register (D register) contains a zero during a write cycle. However, only the 18 inhibit drivers of the segment in which the selected word is located are allowed to be activated.

This is governed by bits 15, 14, and 6 of the main storage address as follows:

MAIN STORAGE ADDRESS BITS			SEGMENT FOR WHICH THE INHIBIT DRIVERS ARE ALLOWED TO BE ACTIVATED	
15	14	6		
0	0	0	A	Array 1
0	0	1	B	
0	1	0	C	
0	1	1	D	
1	0	0	A′	Array 2
1	0	1	B′	
1	1	0	C′	
1	1	1	D′	

Example (See Figure 103):

Bit 15 = 0, bit 14 = 0, bit 6 = 0.
Bit 15 = 0 selects array 1.
Bit 14 = 0 means that X lines 0 through 63 only can be selected.
Bit 6 = 0 means no phase reversal.

With this, segment A is completely defined. Only the inhibit drivers for segment A are allowed to be active.

Outputs of the eight sense amplifiers allocated to one bit are OR'ed together to set the corresponding bit in the data register (D register) during a read cycle. Like the inhibit drivers the sense amplifiers are governed by the main storage address as follows:

MAIN STORAGE ADDRESS BITS		SEGMENTS FOR WHICH THE SENSE AMPLIFIERS ARE ALLOWED TO BE ACTIVATED
14	6	
0	0	A, A′
0	1	B, B′
1	0	C, C′
1	1	D, D′

Note that the sense amplifiers for the corresponding segments in array 1 and array 2 are selected together, regardless of the addressed word being in array 1 or array 2.

Inhibit

- **Half-current in X and Y wires would write all 1's at selected address.**
- **In normal data, some bits are 0.**
- **Inhibit drive stops 1 being written in specified bit position.**

An addressed core-storage location has all cores flipped to 0 in the read cycle. In a write cycle, half-select current in both the X and Y dimensions attempts to flip all cores at the selected address to 1's. However, normal data consists of 1's and 0's, so that some cores must remain at 0.

To retain the zero state in a plane at the write selected address, inhibit (Z) current flows in the common sense/inhibit wire in a direction opposite to the X drive current and cancels the effect of the X drive current for that plane.

Inhibit is per plane so that each position of an addressed location can be controlled separately. Contents of corresponding positions of the data register in CPU

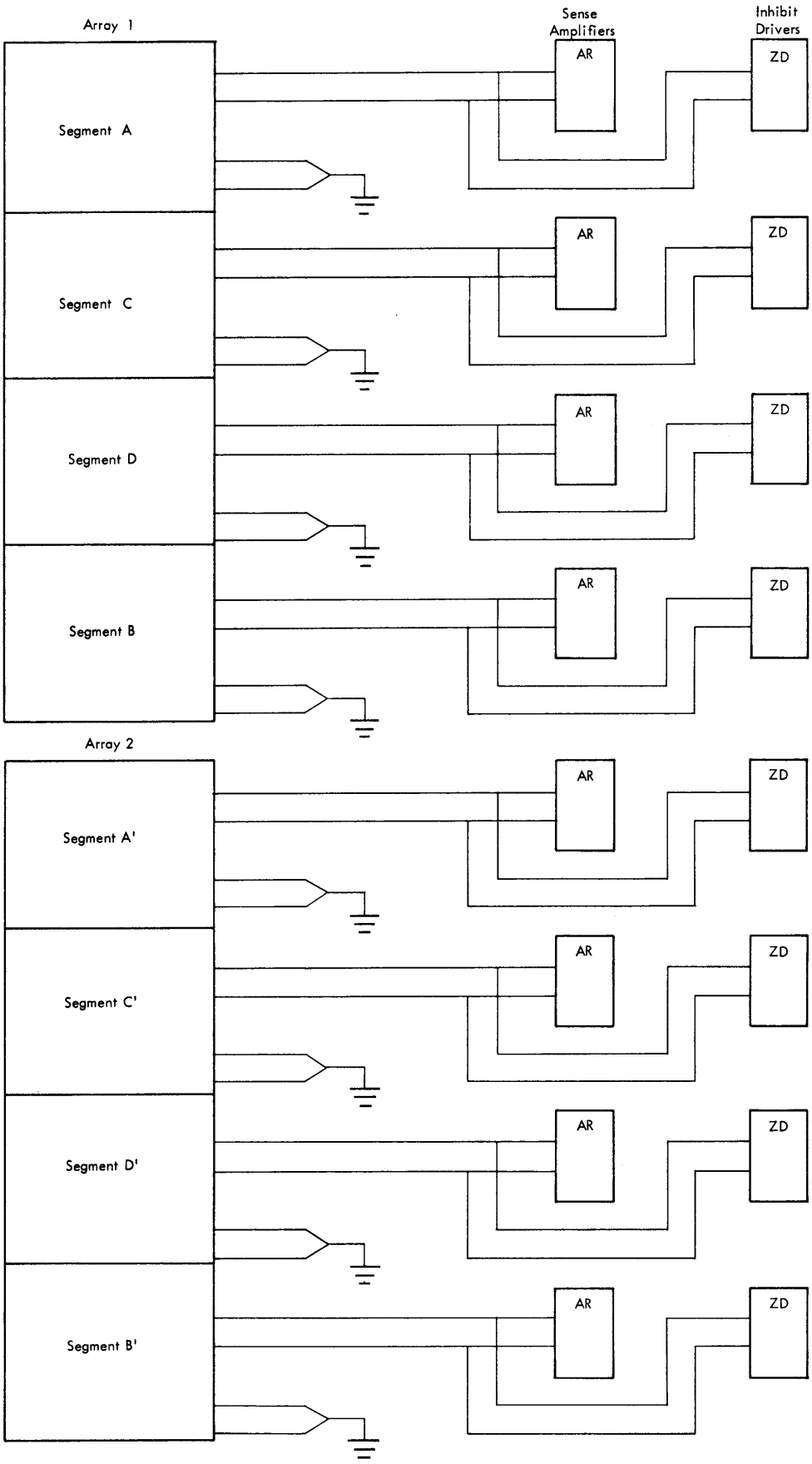


Figure 109. Main Storage Sense Amplifier and Inhibit Driver Allocation

determine whether or not a position of the selected address is to be inhibited.

Figure 110 shows a diagram of the inhibit driver arrangement for one bit plane. The output of the corresponding D register bit position is fed to all eight associated inhibit drivers; bit 7 of D1 is fed to the inhibit driver for phase 0, bit 0 of D0 is fed to the inhibit driver for plane 17.

One inhibit driver is selected by the contents of main storage address bus bit positions 15, 14, and 6, and is activated when the D register bit position contains a zero during a write cycle. Because only one of the eight inhibit drivers is selected and consequently only one sense inhibit line is driven, only one terminating resistor, Rt, is necessary for the eight drivers. This means that only 18 inhibit terminating resistors are required for the whole storage unit.

A simplified diagram of an inhibit driver and the associated sense/inhibit line is shown in Figure 111.

In a write cycle, transistor T1 is turned on by a positive pulse at its base. Current is drawn through the primary of the coupling transformer, inducing a pulse in the secondary which turns on T2. Current flows through the terminating resistor, Rt, transistor T2, the primary of the output transformer and one leg of the sense/inhibit line from ground.

When current is drawn through the primary of the output transformer, point A becomes positive with respect to point B. The effect of this on the secondary is that point D becomes positive with respect to point C and current flows from ground, through the second leg of the sense/inhibit line and the diode, to point D. The output transformer has a 1:1 turns ratio; therefore the current in both legs is the same, so that both legs of the sense/inhibit line are actually driven in series.

Sense

- **Core flipping from 1 to 0 during read operation induces pulse in sense/inhibit line.**
- **Cores that were originally 0 produce no significant output.**
- **Sense outputs fed to pre-amplifiers.**
- **Pre-amplifier outputs fed to final amplifiers.**
- **Final amplifier outputs fed to corresponding positions in D register.**

An addressed core-storage location has all cores flipped to 0 in a read cycle. The flipping of cores that were in the 1 state to the 0 state induces a pulse in the corresponding sense/inhibit line. Cores that were in the 0 state are not flipped and consequently no significant pulse is induced.

Figure 112 shows a diagram of the sense amplifier arrangement for one bit plane. Both legs of a sense/inhibit line are fed to a pre-amplifier. The pre-amplifier has a noise-limiting circuit at its input to isolate the pre-amplifier from the sense/inhibit line when inhibit current is present.

During a read cycle two pre-amplifiers for each bit plane are selected by the contents of main storage address bit positions 14 and 6, one in array 1 and one in array 2.

Only one of the two selected pre-amplifiers will receive a sense pulse from its sense/inhibit line. The outputs of four pre-amplifiers are connected to two final amplifiers; the outputs of the two final amplifiers are OR'ed together and fed to the corresponding bit position in the D register; the outputs of the sense amplifiers of plane 0 are fed to bit 7 of D1; the outputs of the sense amplifiers of plane 17 are fed to bit 0 of D0.

Control and Timing Logic

- **Four control signals from system control main storage:**
  1. **Storage select.**
  2. **Gate select.**
  3. **Write control.**
  4. **Power on reset.**
- **Independent circuits control main storage operation.**

Signals from the system control the logic associated with main storage. See Figure 113. These control signals are:

1. Storage Select –
   A CPU P clock pulse occurring every CPU cycle.
2. Gate Select –
   Gates storage select when a storage cycle is required by the system.
3. Write Control –
   Specifies whether a write or a read cycle has to be performed. Write control active means write. Write control inactive means read.
4. Power On Reset (Not Power Good) –
   Resets the main storage logic when power is switched on in the system.

The logic associated with main storage contains several circuits which control and time main storage opera-

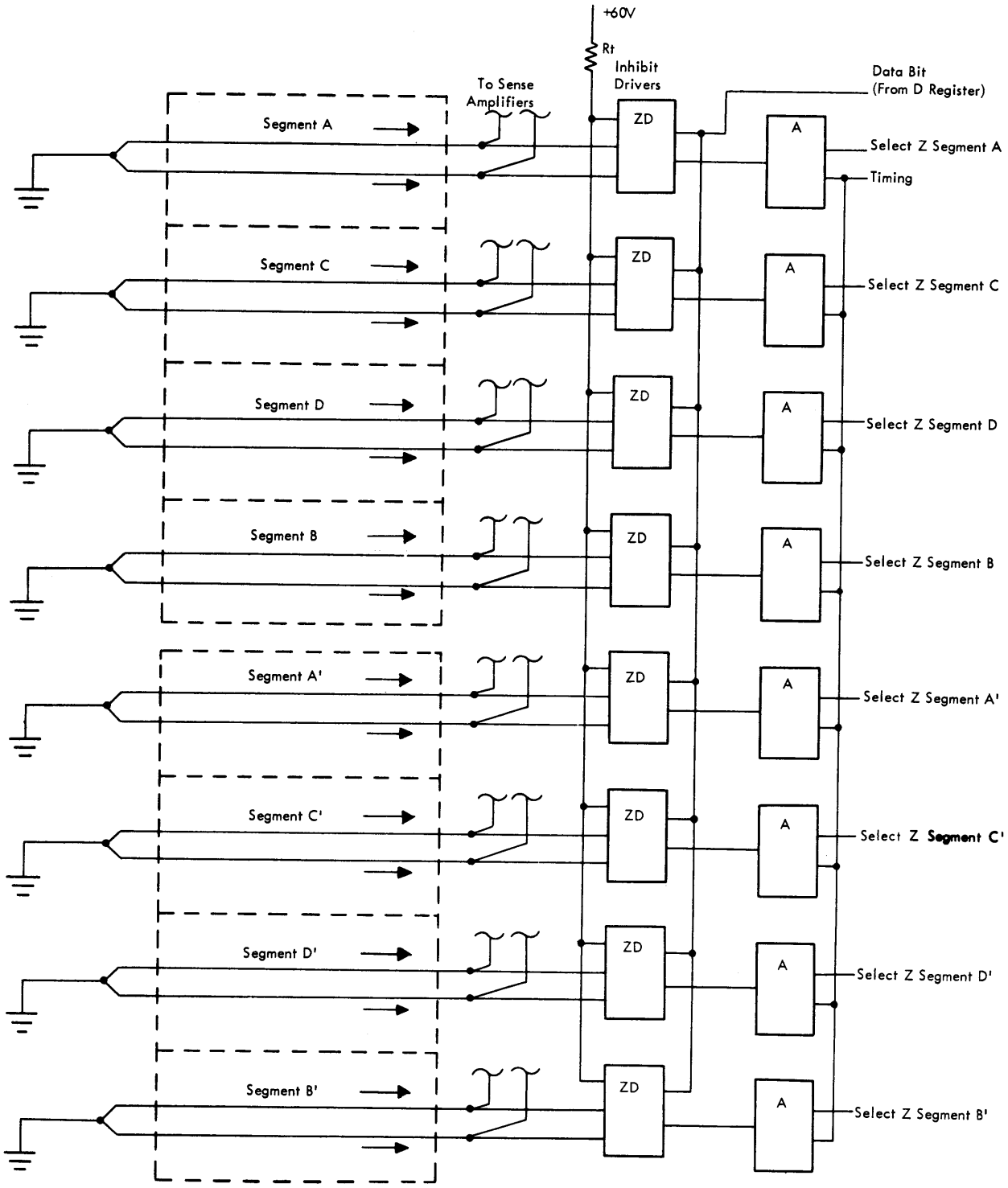


Figure 110. Inhibit Drive

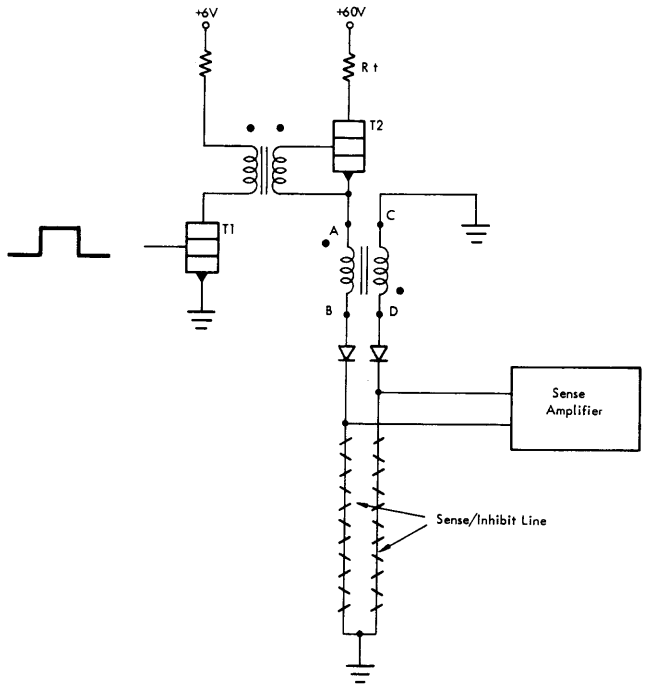


Figure 111. Principle of Inhibit Drive

tions. See Figure 504. These control and timing circuits are:

180-nanosecond select singleshot.
Delay line clock.
Storage busy.
Read-write control.
X read-write control.
Y read-write control.
X read-write driver timing.
Y read-write driver timing.
Multiplex storage driver timing.
600-nanoseconds interlock singleshot.
X read-write terminator gate timing.
Y read-write terminator gate timing.
Strobe A and B.
Z (inhibit) timing.

180-Nanosecond Select Singleshot

The select singleshot provides a 180-nanosecond pulse. This pulse is used as the input to the delay line clock and also as a timing pulse.

Storage select, gate select, and allow select are AND'ed together to fire the select singleshot. Allow select is active when the X read-write driver timing latch is off i.e. when the storage is ready to be selected by the system.

Delay Line Clock

Internal timing pulses are generated by a delay line in 250-nanosecond sections. Each section is tapped along its length at 25-nanosecond intervals. At the start of a storage cycle the 180-nanosecond select pulse is sent along the delay line.

The output of the taps is used to time the setting and resetting of the control and the timing latches. The outputs of the taps of the first three 250-nanosecond sections time the setting of the control and timing latches; the outputs of the taps of the last three 250-nanosecond sections time the resetting of the control and timing latches.

The master reset latch connects the first three sections with the last three. It is set with the output of the 525-nanosecond tap. After 50 nanoseconds, the delayed latch output attempts to reset the latch but because the set overrides the reset the latch will be on for the duration of the 180-nanosecond set pulse.

The purpose of the latch is to reshape the output of the 525-nanosecond tap before sending it into the last three delay line sections.

The 425-nanosecond tap is connected to a delay line which has a delay of 5 nanoseconds per tap. The outputs of this delay line are used to time the setting of the sense strobe latches.

Storage Busy

The storage busy latch provides an interlocking signal to the system indicating that the storage is performing a storage cycle.

Read-Write Control

The read-write latch controls the setting and resetting of the timing latches during a write or a read cycle. Write control, storage select, and gate select are AND'ed to set the read-write latch at the beginning of a write cycle. Nominally 875 nanoseconds after the rise of select the latch is reset to read. It is left reset for a a read operation.

X Read-Write Control

The X read-write latch is used to gate the X read or write terminator gates and the X read or write drivers. On a write cycle, the X read-write latch is turned on by storage select, gate select and write control and gates the X write terminator gate and the X write drivers. Nominally, 900 nanoseconds after the rise of select, the latch is reset to read. On a read cycle, the X read-write latch is left reset (read condition) and gates the X read terminator gate and the X read drivers.

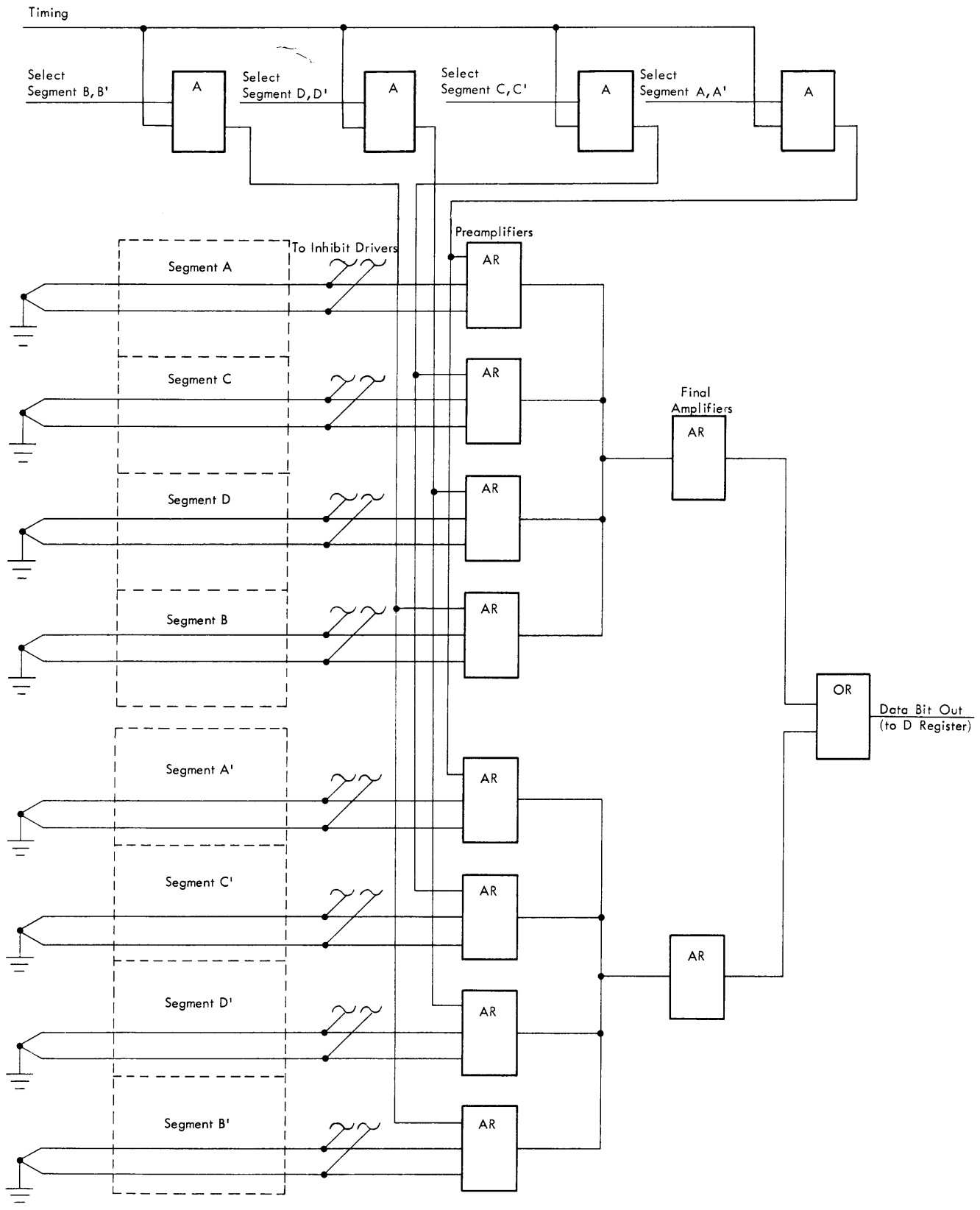


Figure 112. Main Storage Sense

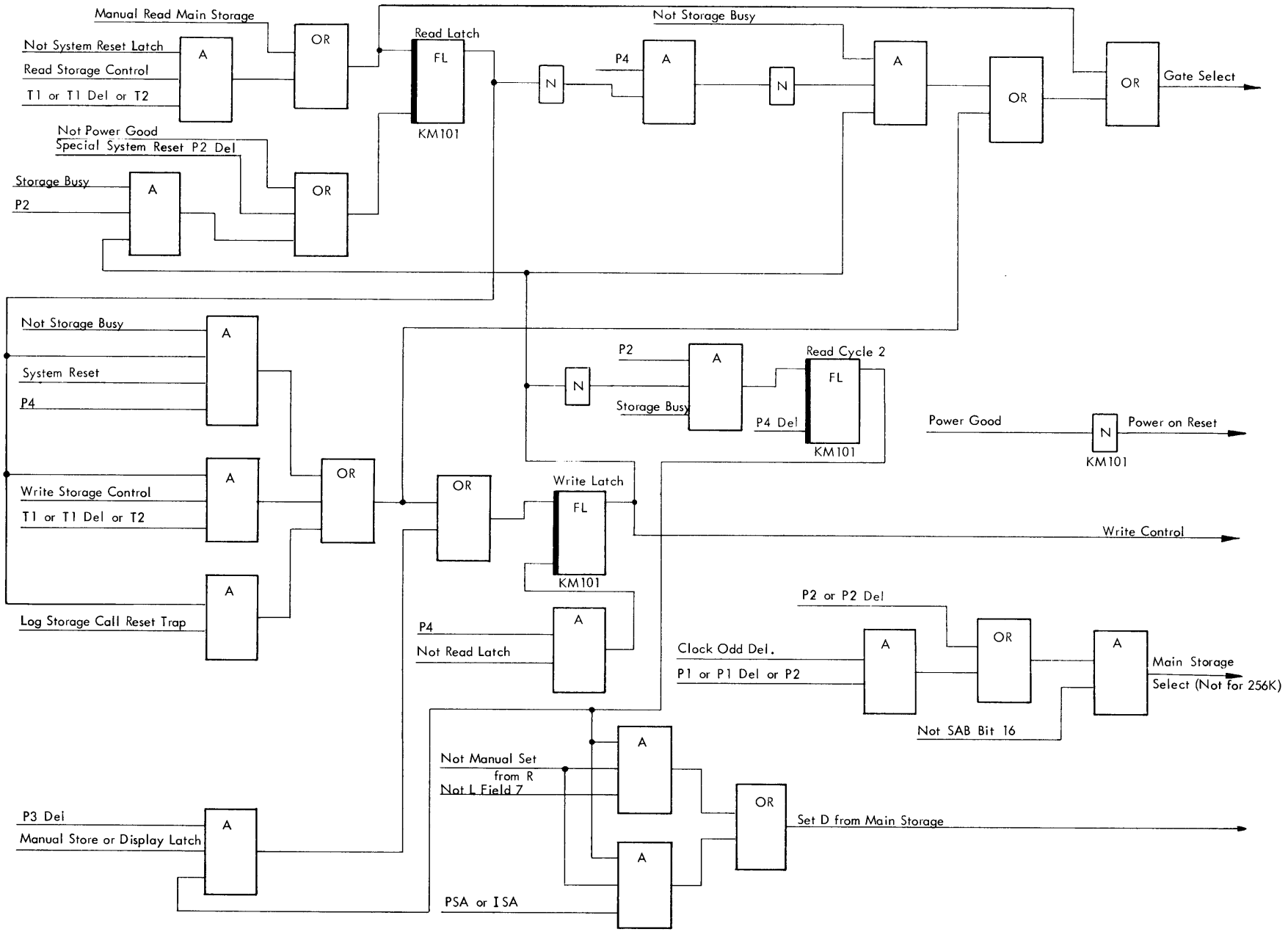


Figure 113. Storage Select and Gate Select Pulses

Y Read-Write Control

The Y read-write latch gates the Y read or write terminator gates and the Y read or write drivers similarly to the X read-write latch. Because phase reversal is used in the Y dimension, the setting of the latch depends not only on read-write control but also on the state of storage address bus bit 6.

On a write cycle with no phase reversal the Y read-write latch must be set to write. The AND circuit with select, write control and address decode not bit 6 sets the latch.

On a write cycle with phase reversal the Y read-write latch must be left reset (Y read condition); no setting is required.

On a read cycle with no phase reversal the latch must be left reset.

On a read cycle with phase reversal, the latch must be set to write. Address decode bit 6 together with select and not write control sets the latch.

Nominally, the Y read-write latch is reset 875 nanoseconds after the rise of select on a write cycle and 1,025 nanoseconds after the rise of select on a read cycle.

X Read-Write Driver Timing

The X read-write driver timing latch provides a timing pulse for all X drivers. It times a selected driver to generate an X driver pulse in the read or write direction.

Y Read-Write Driver Timing

The Y read-write driver timing latch provides a timing pulse for all Y drivers similarly to the X read-write driver timing latch. The latch can be set only when MS address bus bit B is zero, i.e., when multiplex storage is not used.

Multiplex Storage Driver Timing

The multiplex storage driver timing latch can be set only when MS address bus bit B is one.

600-Nanosecond Interlock Singleshot

With storage operating at maximum speed, a pulse started through the delay line during a write cycle could reset latches during the first 300 nanoseconds of a following read cycle because the read and write cycles are of different duration and the common delay line clock is of fixed duration. Therefore, the 600-nanosecond singleshot inhibits any read reset pulse during the first part of a read cycle.

The singleshot is fired when the read-write latch changes from a write to a read status, and write clock pulses are allowed to run out without interfering with the read cycle.

X Read-Write Terminator Gate Timing

The X read-write terminator gate timing latch provides timing for the X read terminator gate and the X write terminator gate. X read-write control selects either the read or the write terminator gate.

On a read cycle, the reset of the X read-write terminator gate timing latch is held off as long as the interlock singleshot is active.

Y Read-Write Terminator Gate Timing

The Y read-write terminator gate timing latch provides timing for the Y read terminator gate and the Y write terminator gate. Y read-write control selects either the read or the write terminator gate. On a read cycle, the reset of the Y read-write terminator gate timing latch is held off as long as the interlock singleshot is active.

Strobe A and B

The strobe A and B latches provide timing for the sense amplifiers. Strobes A and B are fed to different groups of sense amplifiers as follows:

Strobe A strobes the sense amplifiers for:
Byte 0 in segment B — B′
Byte 0 in segment D — D′
Byte 1 in segment A — A′
Byte 1 in segment C — C′
Strobe B strobes the sense amplifiers for:
Byte 0 in segment A — A′
Byte 0 in segment C — C′
Byte 1 in segment B — B′
Byte 1 in segment D — D′
Remember that main storage address bits 14 and 6 select one segment in array 1 and one segment in array 2 at a time.

Example: Segments A — A′ are selected by bits 14 and 6. Strobe A times the sense amplifiers for byte 1 and strobe B times the sense amplifiers for byte 0 in these segments.

Z Timing

The Z (inhibit) timing latch provides timing for all Z drivers.

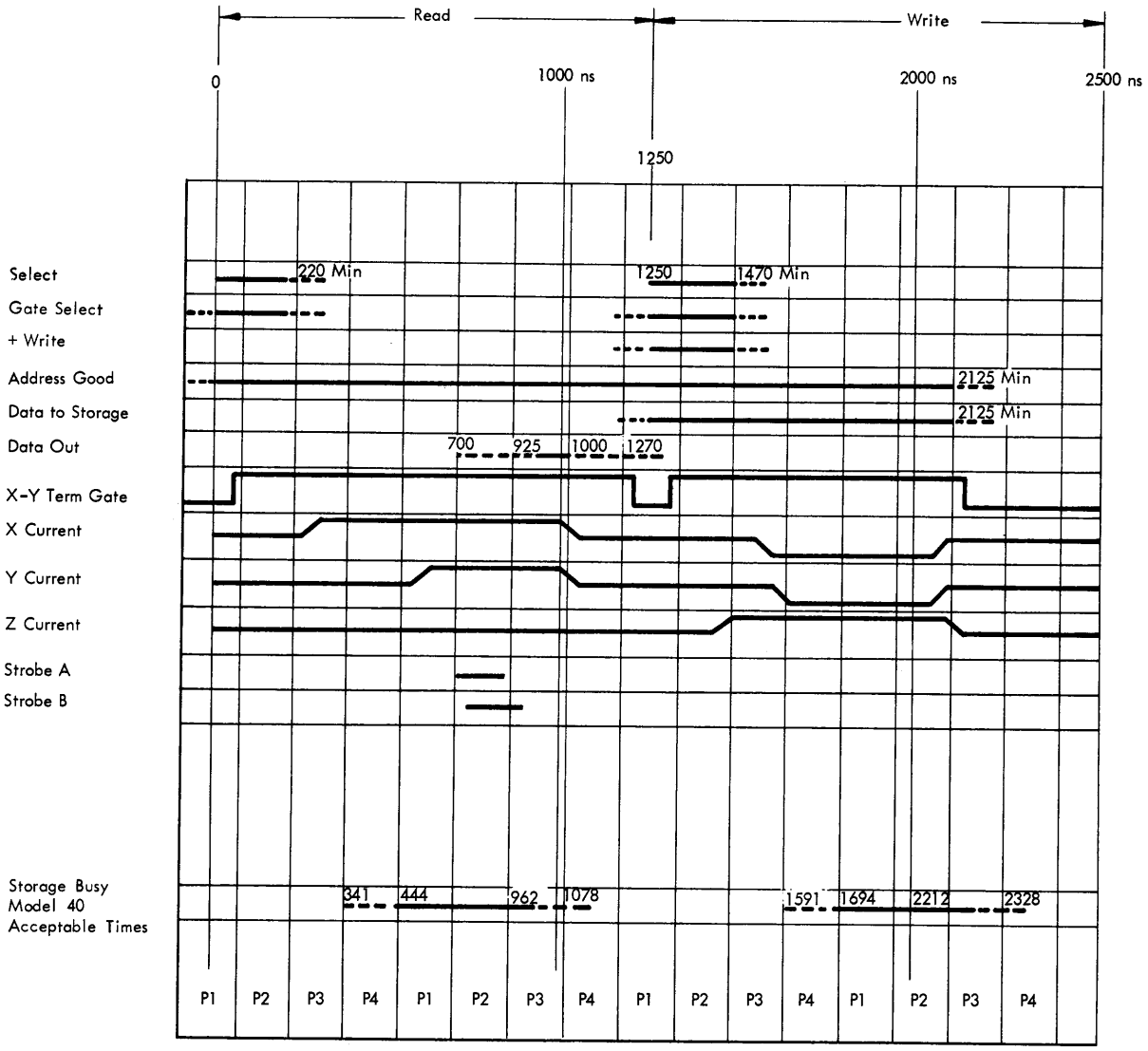
Block Diagram

Figure 101 is a block diagram of the 64K halfword main storage and the 1,024 halfword multiplex storage with associated circuitry.

The numbers on the lines indicate the number of lines.

Timing

Figure 114 shows the control and timing pulses for a read and a write cycle. Although there may be a gap



Notes: 1. All times measured from rise of select
2. Timings approximate unless detailed

Figure 114. Main Storage Timing Chart

of indefinite period following either a read or a write cycle, a read cycle is shown to be followed immediately by a write cycle.

Read Cycle

- **Address set in A register in second half of previous cycle.**
- **XY terminator gates turned on first.**
- **Read driver turns on later.**
- **X drive current turned on before Y drive current.**
- **Sense output is strobed to time 1-bit signal output.**

In the second half of the previous CPU cycle, the A register is set with the address. The address is gated onto SAB and decoded by the gate decoders. The bases of the selected gates are up 140 nanoseconds later.

At time zero of the read cycle the X and Y terminator gate timing and the X and Y read-write driver timing are up. The XY terminator gates are turned on first and the address-selected read driver turns on later because of delay in the circuits. Because the sense line is parallel with the X line, the X driver is turned on and X drive current noise is allowed to settle before the Y drive current is turned on.

After the X drive noise has settled, the Y read-write driver timing pulse turns the address-selected Y driver on. The leading edges of sense strobe A and sense strobe B are adjusted to coincide with the leading edge of a one-bit signal from the corresponding planes.

The X and Y drivers are turned off before the terminator gates. The terminator gates turn off after the current in the terminating resistors has decreased to zero and the voltage of the line has returned to +60v.

Write Cycle

Since an output is not being sensed on the write cycle, the problem of noise does not exist. Therefore, X and Y currents can be turned on within a shorter time interval. The inhibit drivers are turned on before the X and Y drivers and turned off after the X and Y drivers.

External Controls

- **To control main storage, control signals are generated in the CPU.**
- **These control signals determine:**
  1. **If a storage cycle must be initiated (gate select)**
  2. **The CPU time at which a storage cycle is to be started (storage select)**
  3. **Whether a read or a write cycle is to be performed (write control)**
  4. **The reset to the internal MS circuitry when power is turned on (power on reset).**

The main-storage external control circuitry is shown in Figure 113. The following paragraphs detail the external control circuitry.

Gate Select

Gate select is activated by the set pulse of the read latch or with the write latch output AND'ed with not storage busy.

Storage Select

Storage select is a P1 del or P2 or P2 del pulse that occurs every cycle. When gate select is active, the first storage select pulse initiates a storage cycle provided that the storage is ready.

Write Control

Write control is the output of the write latch. When the write latch is on, the storage will perform a write cycle when it receives select. When the write latch is off, the storage will perform a read cycle when it receives select.

Power On Reset

Power on reset resets the internal circuitry of main storage when power is switched on in the machine.

Control Timing (Figures 113 and 114)

When a read command is given by the microprogram (CR field = 001), the read latch is set with a T1 or T1 del or T2 pulse. This set pulse is also fed as gate select to the main storage. The storage select pulse is timed with P1 del or P2 or P2 del.

Gate select and storage select together initiate the storage cycle, provided that allow select is active. Allow select is generated internally in the MS circuitry and indicates that the storage is ready to be started.

Because the write latch is not on, a read cycle is now started. About 350 nanoseconds after the rise of storage select, the line storage busy from main storage is activated, indicating that a storage cycle is being performed.

Not write latch, storage busy, and time P2 set the read cycle 2 latch. This latch is reset at P4 del time. During the second cycle of read, it is invalid for the microprogram to give a MS write command.

When the write command (CR field = 010) is given after the second cycle of read, the write latch is set at T1 or T1 del or T2 time provided that the read latch is on. This means that a write command is valid only after a read cycle has been performed. The gate select comes up again and lasts until the rise of storage busy. Storage select will now initiate a write cycle.

The read latch is reset with write latch, storage busy, and P2 time. When the read latch is off, the next P4 pulse will turn off the write latch.

Data Register (D Register)

The signal, set D register from main storage (Figure 113), gates the data from main storage into the D register during storage read cycle 2. Main storage data out in bits 00 is connected to D1 bit 7, and main storage data out bit 17 to D0 bit 0.

Transfer of data from the R register to the D register, either controlled by microprogram (CL field = 7) or under manual control, has priority over the data transfer from main storage. R register to D register transfers are overridden by ALU to D0, D1 transfers. Storage address test, indicating an addressing error, cancels the priority of the microprogram-controlled transfers from R and from the ALU. This allows the data from the erroneously addressed MS halfword to be preserved in the D register. Refer to the Storage Protect section in this manual.

Checking Storage Address Bus

The SAB is checked for odd parity.

A parity error sets the early data check latch at P1 del time.

Figure 115 shows how the A, S1 or S2 registers' parity bit is fed onto the SAB.

Checking D Register

Figure 116 shows the D register parity generation and parity checking.

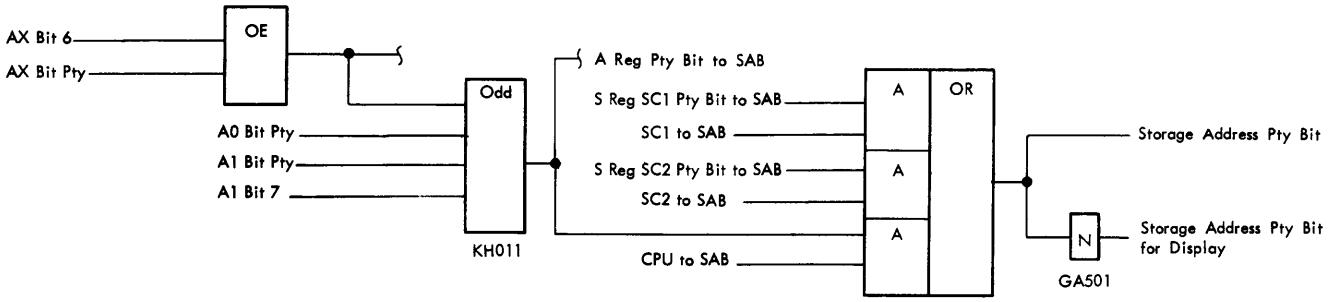
The D register is checked for odd parity. A parity error sets the early data check latch at P1 del time.

Good (odd) parity is always sent to main storage by design. A D register parity error, occurring when the D register contents are to be used as main storage data, does not prevent the starting of a main storage cycle.

Circuit Description

The following circuit descriptions are located in the Appendix of the *System/360 Model 40, Power Supplies, Features, and Appendix*, Form 223-2845.

Read-Write Driver
Terminator Gate
Gates
Gate Decoder
Array Diodes
Inhibit Driver and Clamp (Z Driver and Clamp)
Sense Pre-Amplifier
Strobe Emitter Follower
Final Amplifier



SAB Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Mpx Store
S Reg Bit	S1 Bit 6	S1 Bit 5	S1 Bit 4	S1 Bit 3	S1 Bit 2	S1 Bit 1	S1 Bit 0	S0 Bit 7	S0 Bit 6	S0 Bit 5	S0 Bit 4	S0 Bit 3	S0 Bit 2	S0 Bit 1	S0 Bit 0	SX Bit 7	Use Mpx Store
A Reg Bit X Y1	A1 Bit 6	A1 Bit 5	A1 Bit 4	A1 Bit 3	A1 Bit 2	A1 Bit 1	A1 Bit 0	A0 Bit 1	A0 Bit 0	AX Bit 7	A0 Bit 4	A0 Bit 3	A0 Bit 2	A0 Bit 7	A0 Bit 6	A0 Bit 5	Add
A Reg Bit X Not Y1	"	"	"	"	"	"	"	A0 Bit 7	A0 Bit 6	A0 Bit 5	"	"	"	A0 Bit 1	A0 Bit 0	AX Bit 7	Switch or Set

Figure 115. Main Storage Addressing Parity

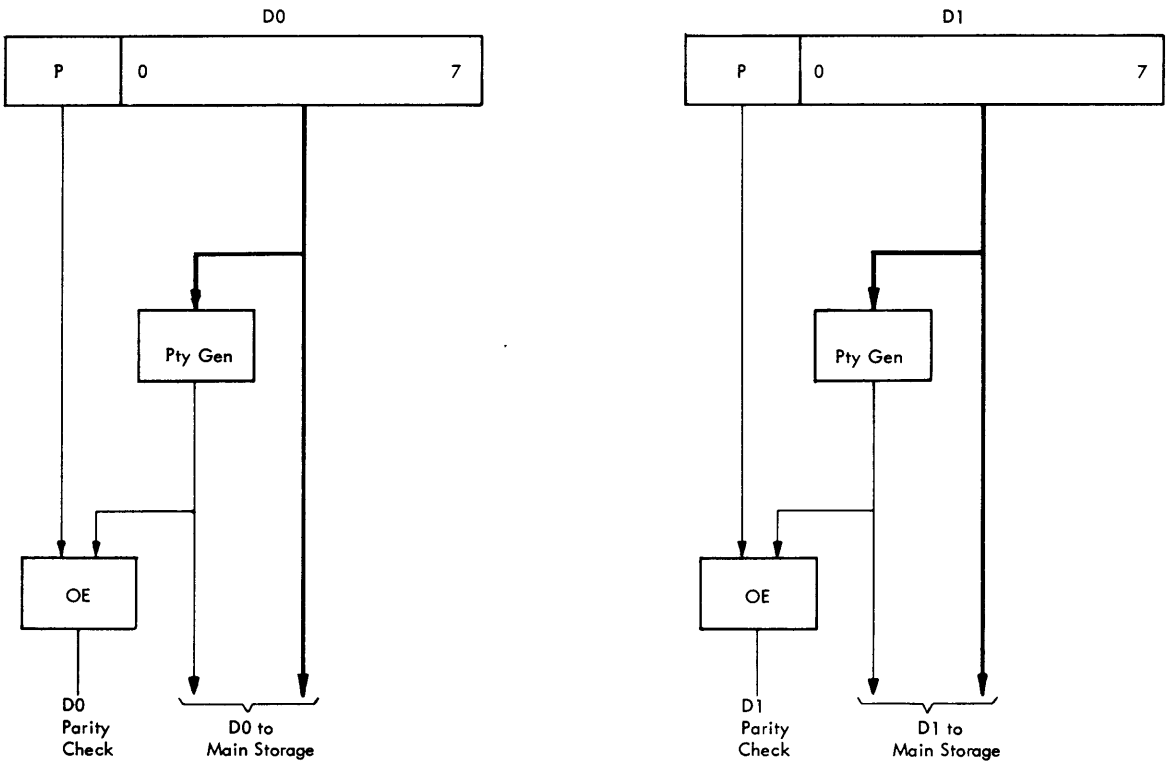


Figure 116. D Register Parity Generation and Parity Check

1,024K Halfword Multiplex Storage

- **Incorporated in main storage unit.**
- **Stores multiplex channel unit control words (UCW's).**
- **Size increases proportionately with main storage size.**
- **External control circuits in common with main storage.**
- **Separate Y drive scheme.**

The multiplex storage is contained in the main storage unit. It is used to store the unit control words for the multiplexor channel and is only accessible by the microprogram.

Multiplex storage size increases with main storage size as follows:

MODEL	MAIN STORAGE HALFWORDS	MULTIPLEX STORAGE HALFWORDS
D	8,192	128
E	16,384	256
F	32,768	512
G	65,536	1,024
GF	98,304	1,024
H	131,072	1,024

Note that Models G, GF, and H have the same size multiplex storage.

Multiplex storage shares with main storage the external control circuits, the internal control and timing circuits, the sense inhibit system, the X drive scheme and the Y terminator gates plus terminating resistors.

A separate multiplex store Y drive scheme is used to select and drive any one of the four multiplex storage Y lines (two per array) although it shares the terminator gates and terminating resistors used by the Y dimensions of main storage.

Only the multiplex storage associated with the 64K halfword main storage will be discussed in the following sections; other models operate similarly.

Multiplex Storage Address

- **With Mpx stat on, A0 bits 5, 6 and 7, form bits 15, 14 and 13 of address bus.**
- **A1 bit 7 determines which byte of D Register is to be processed.**
- **Mpx stat output forms bit B of address bus.**

Figure 117 shows the generation of the multiplex storage address from the A register output. When the Mpx stat is on, A0 bits 5, 6, and 7, form bits 15, 14, and 13 of the address bus. A1 bit 7 is not used in the address but determines which byte of the two-byte data register is to be processed by the system. AX bit 7 and A0 bits 0-4 are not used to address multiplex storage but are fed to the storage address bus because the SAB is parity checked.

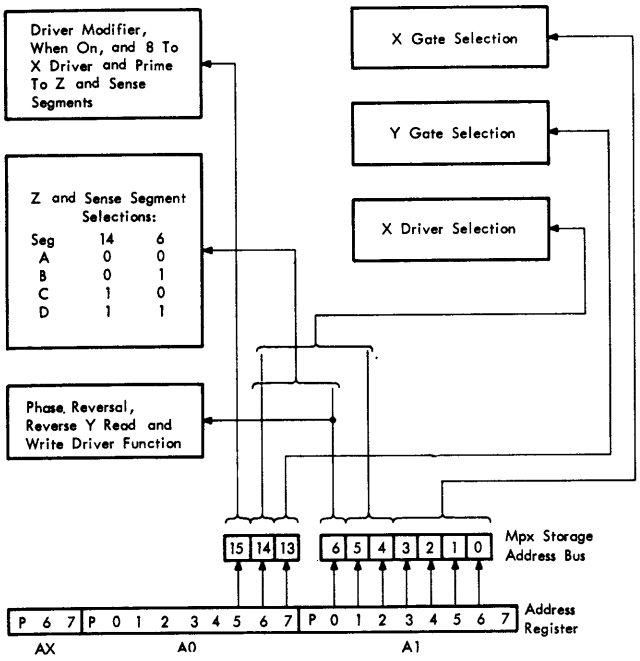


Figure 117. Storage Address Bit Allocation for Mpx Storage

The output of the Mpx stat forms bit B of the storage address. When the Mpx stat is on, the Y dimension of the main storage drive scheme is held off by suppressing the timing for the Y drivers, and the drive scheme of the multiplex storage Y dimension is activated by activating the timing circuit for the multiplex storage Y drivers. See Figure 504.

Because the X dimension drive scheme is common to main storage and multiplex storage, the address bits for the X dimension are the same as for main storage. Bit 13 is used to address the multiplex storage Y lines. Bit 15 is used for both X and Y and selects one of the two arrays of the storage. Address bits 12 through 7 are not used.

Phase reversal and inhibit/sense segment selection are common to main and multiplex storage.

Selection of One Y Line

- **Four gates for read or write.**
- **Four gate decoders decode address bits 13 and 15 to select a gate.**
- **Terminator gates and terminating resistors common with main storage.**

The principle of the multiplex storage X and Y drive scheme is the same as the main storage drive scheme. Figure 118 shows the basic selection of one Y line.

Because there are only four lines in the Y dimension, only four gates are needed either for read or write. Four gate-decoders decode the address bits 13 and 15 to select a read or a write gate. The emitters of the read gates are connected to one read driver. The emitters of the write gates are connected to one write driver.

The Y read and write terminator gates and terminating resistors are common to multiplex and main storage.

Multiplex Storage Y Drive

- **Write driver activated by Y read AND'ed with bump driver timing.**
- **Read or write depends on cycle control and on address bit 6 (phase reversal).**
- **Address bit 15 determines whether array 1 or array 2 is addressed.**

The multiplex storage Y drive scheme is shown in Figure 119.

Address bits 13 and 15 are fed to the gate decoders.

The read driver is activated with Y read AND'ed with bump driver timing. The write driver is activated with Y write AND'ed with bump driver timing. Y read or Y write depends not only on whether a read or a write cycle is to be performed, but also on the setting of phase reversal address bit 6. Note that address bit 15 determines whether array 1 or 2 is used.

Circuit Description

The following circuit descriptions are located in the Appendix of the *System/360 Model 40 Power Supplies, Features, and Appendix,* Form 223-2845.

Read-Write Driver
Terminator Gate
Gates
Gate Decoder
Array Diodes
Inhibit Driver and Z Clamp (Z Driver and Clamp)
Sense Pre-Amplifier
Strobe Emitter Follower
Final Amplifier

Main Storage Power Supplies

- **Standard SLT voltages −3, +3, and +6.**
- **Special voltages +18, +60.**

Power is supplied to the storage unit by the system from standard supplies which provide −3, +3, and +6 volts, and from special supplies for main storage

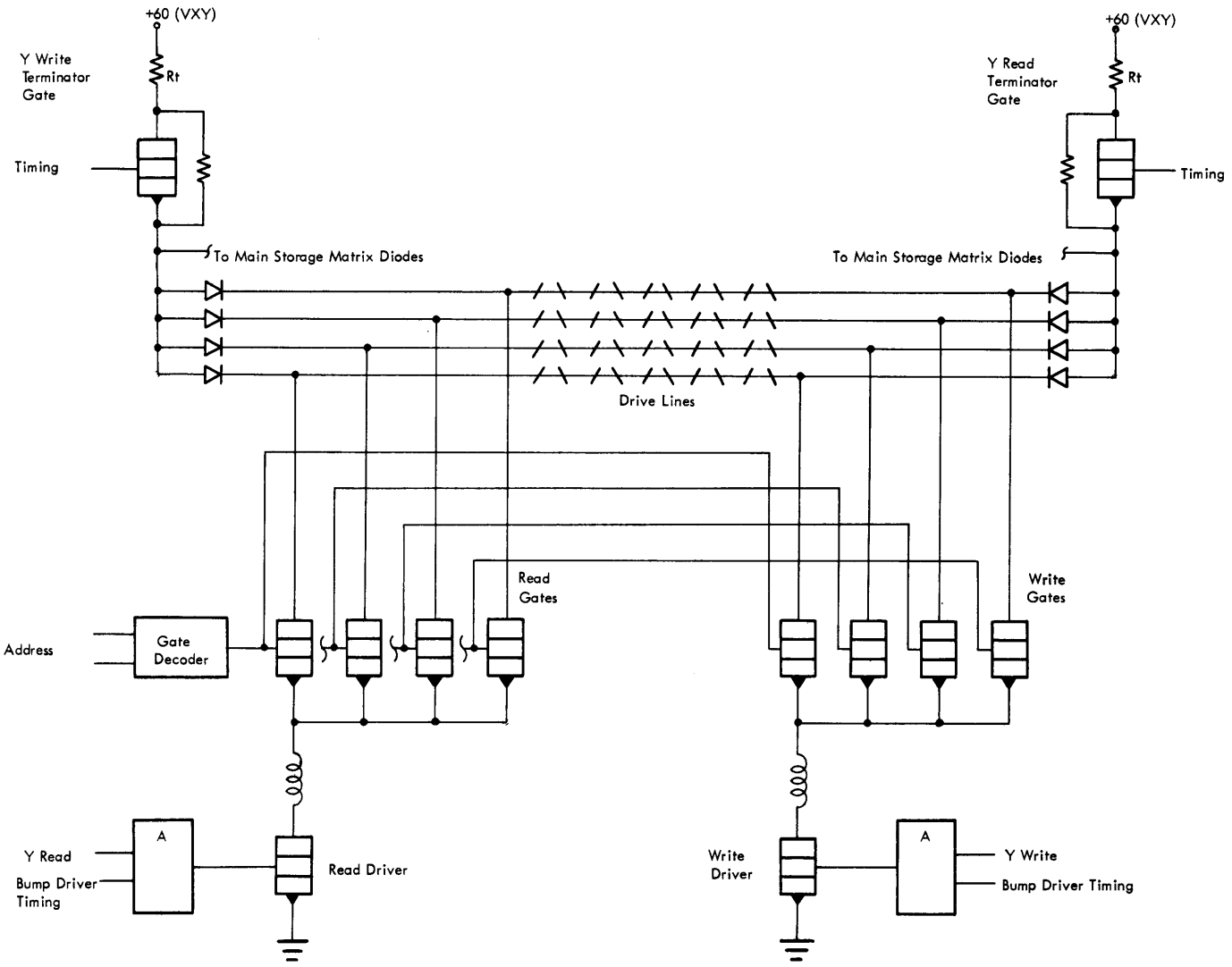


Figure 118. Multiplex Y Drive

which provide +18 volts, +60 volts for the XY drive and +60 volts for inhibit drive.

SUPPLY	MAX AV CURRENT	UNDERVOLTAGE PROTECTION	OVERVOLTAGE PROTECTION
+3	3.0	None	4.5
−3	10.0	None	−4.5
+6	10.0	5.2 ± 0.2	8.0
+60Z	4.0	None	70.0
+60XY	1.0	None	70.0
+18	4.0	None	22.0

Physical Structure of Main Storage

Figure 120 is a diagram of the main storage unit.

The 64K halfword main storage consists of a core array 1 with array end board D1 and jumper board D2, core array 2 with array end board D3 and jumper board D4, large boards A1 and B1, socket panel C1 and terminating resistor board A2.

A single-fan unit is on the bottom of each array and a blower unit with three fans is below panels A1, B1 and A2. Air filters are on the bottom of each fan unit and blower unit.

The core arrays are hung on hinges on the storage unit frame. Both arrays of the 64K halfword storage are identical but array 2 is inverted with respect to array 1 so that the hinges and the array end board are on the outside.

The 2-24 SLT cards containing the gates and array diodes are plugged on the array end boards D1 and D3.

On jumper boards D2 and D4, the X lines of segments A-B and C-D are jumpered together. Also, one end of each of the 128 Y lines and two multiplex Y lines are connected to these boards to be returned to the gates and array diodes on the array end boards D1 and D3.

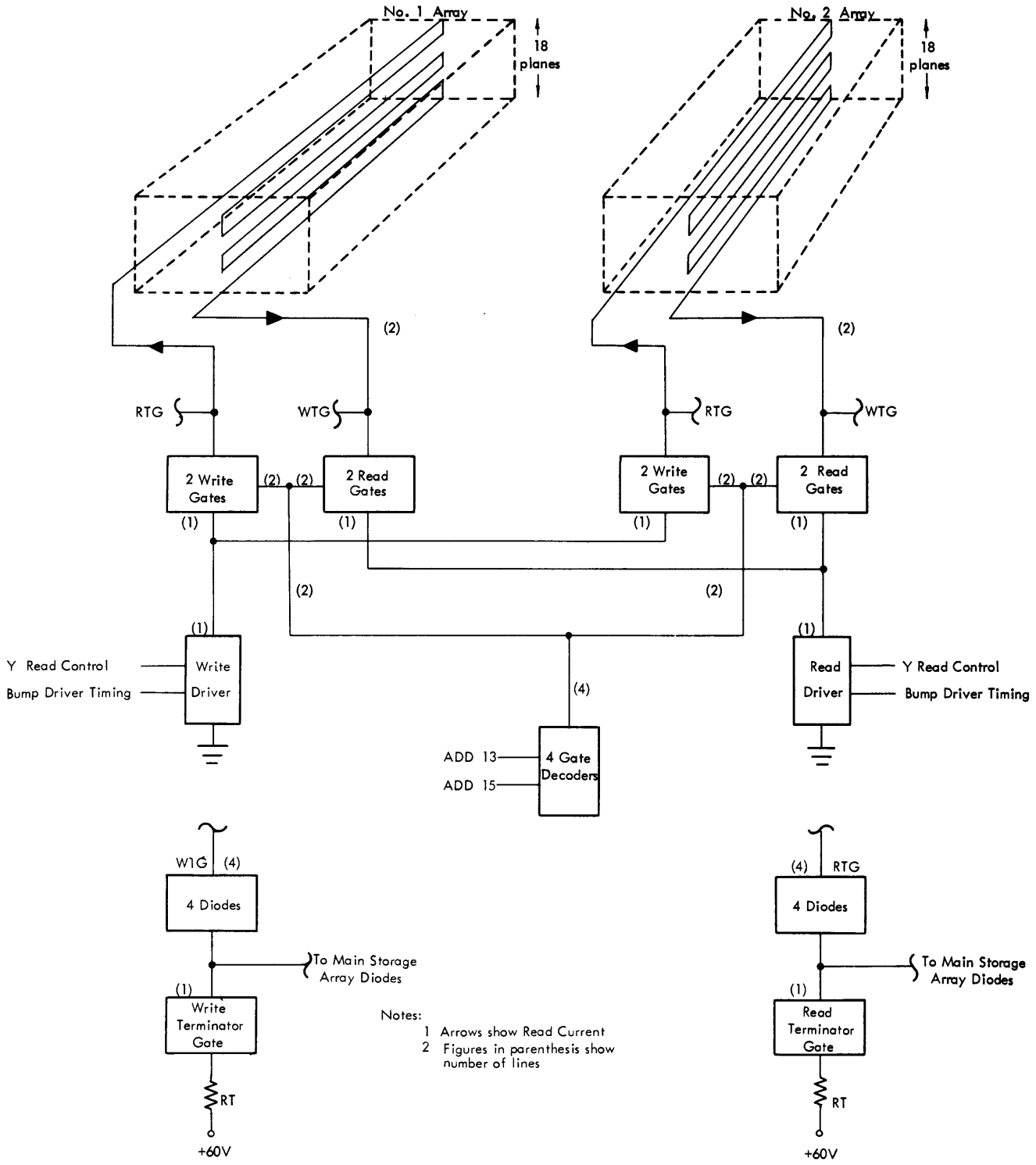


Figure 119. Multiplex Storage Y Dimension

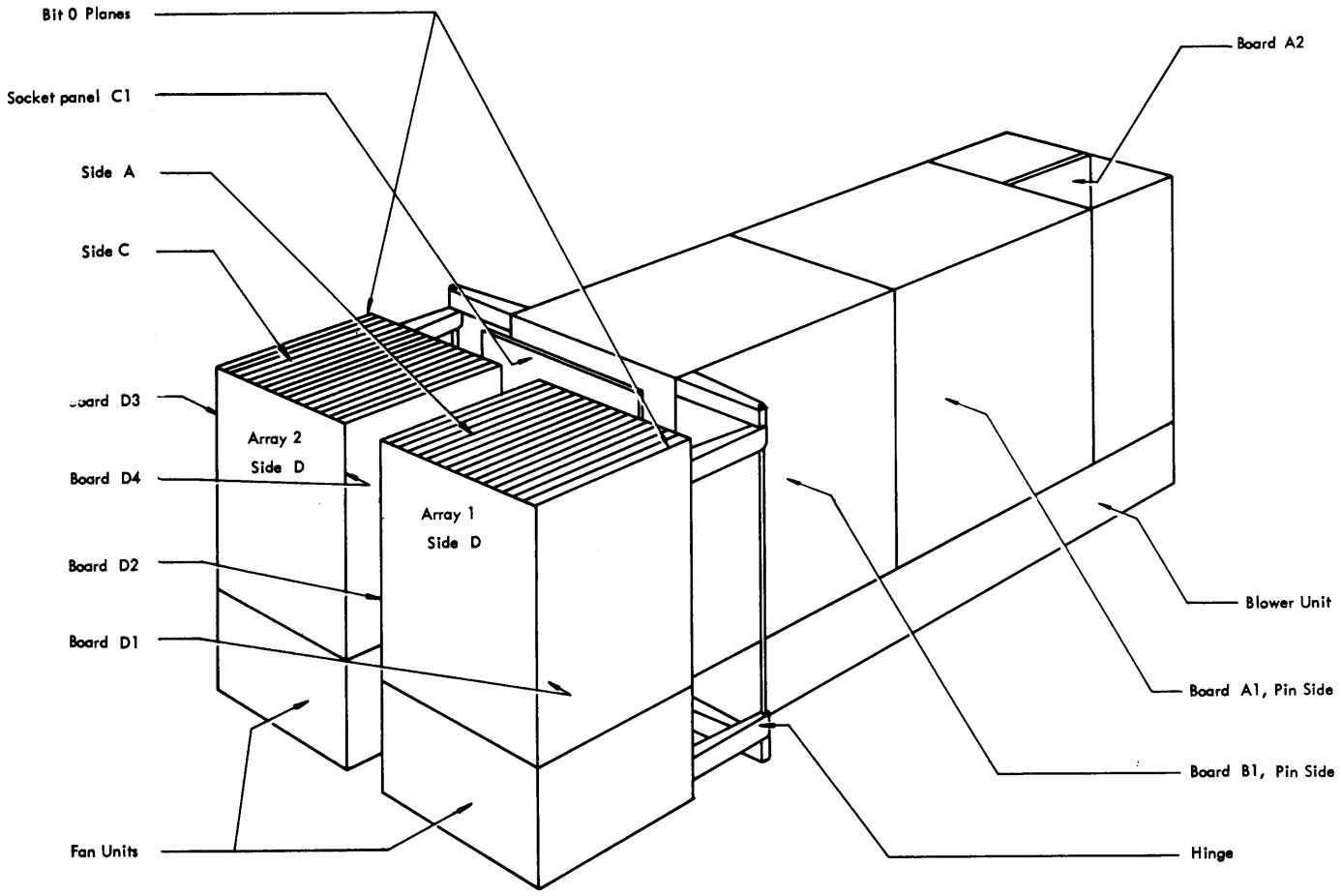


Figure 120. 64K Halfword Storage Unit

With the exception of the gates and array diodes, all storage circuitry is packaged on 2-36 SLT cards that are plugged on the large boards A1 and B1. Connections between the arrays and the circuitry on large boards A1 and B1 are made via socket panel C1. Cables from boards A1 and B1 are wire wrapped to one side of panel C1. The connections to the two arrays are separately plugged to the array side of the panel via flexible cables.

Board A2 carries the four XY terminating resistors and the 18 inhibit terminating resistors.

Arrays

Array planes are assembled with two planes facing each other. Each pair of planes is separated from the next pair by a sheet of insulating material. For each 18 plane array, 9 pairs of planes are assembled between an end board, D1 or D3, and a jumper board, D2 or D4. See Figure 121.

Both arrays of the 64K halfword storage unit are identical, but array 2 is inverted with respect to array 1 so that the hinges and the array end board are on the outside. Cable receptacles on panel C1 are also inverted for array 2 so that cables are not crossed.

Core Planes

There are 33,280 cores arranged in each core plane shown in Figure 122. Individual cores are held in place by the three wires that pass through them. The wires are welded to pins in the frame assembly. Cores are arranged so that their magnetic fields are at 90° to each other to eliminate interference between them.

The edge connector pins are moulded in a plastic frame fastened to an aluminum frame. The frame assembly is about 10¾ x 6¾ x ⅛ inches (27.3 x 17.2 x 0.3 cm) when assembled.

The 130 Y drive lines run parallel to the long dimension, each one passing through 256 cores. 256 X drive

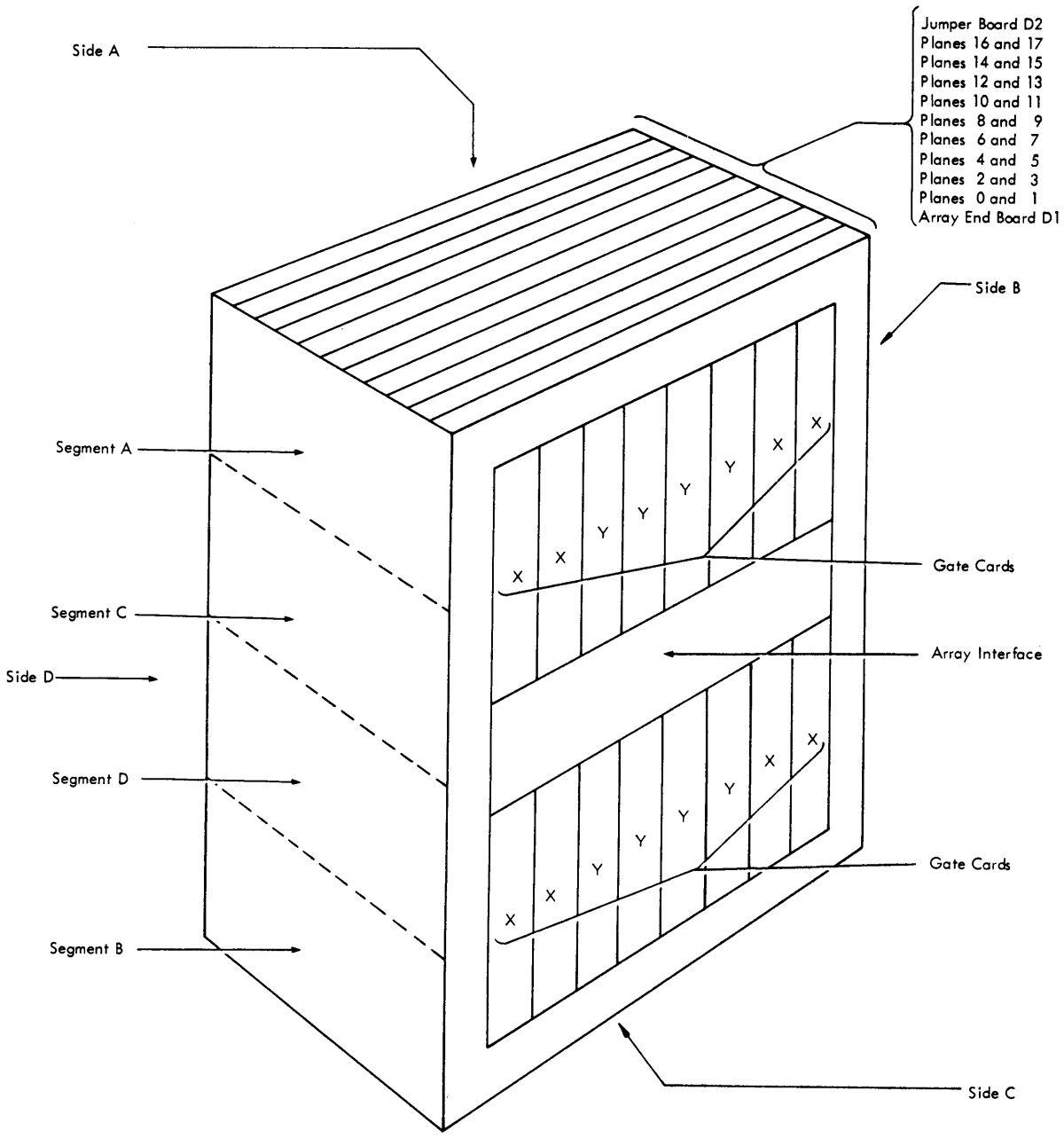


Figure 121. Array No. 1

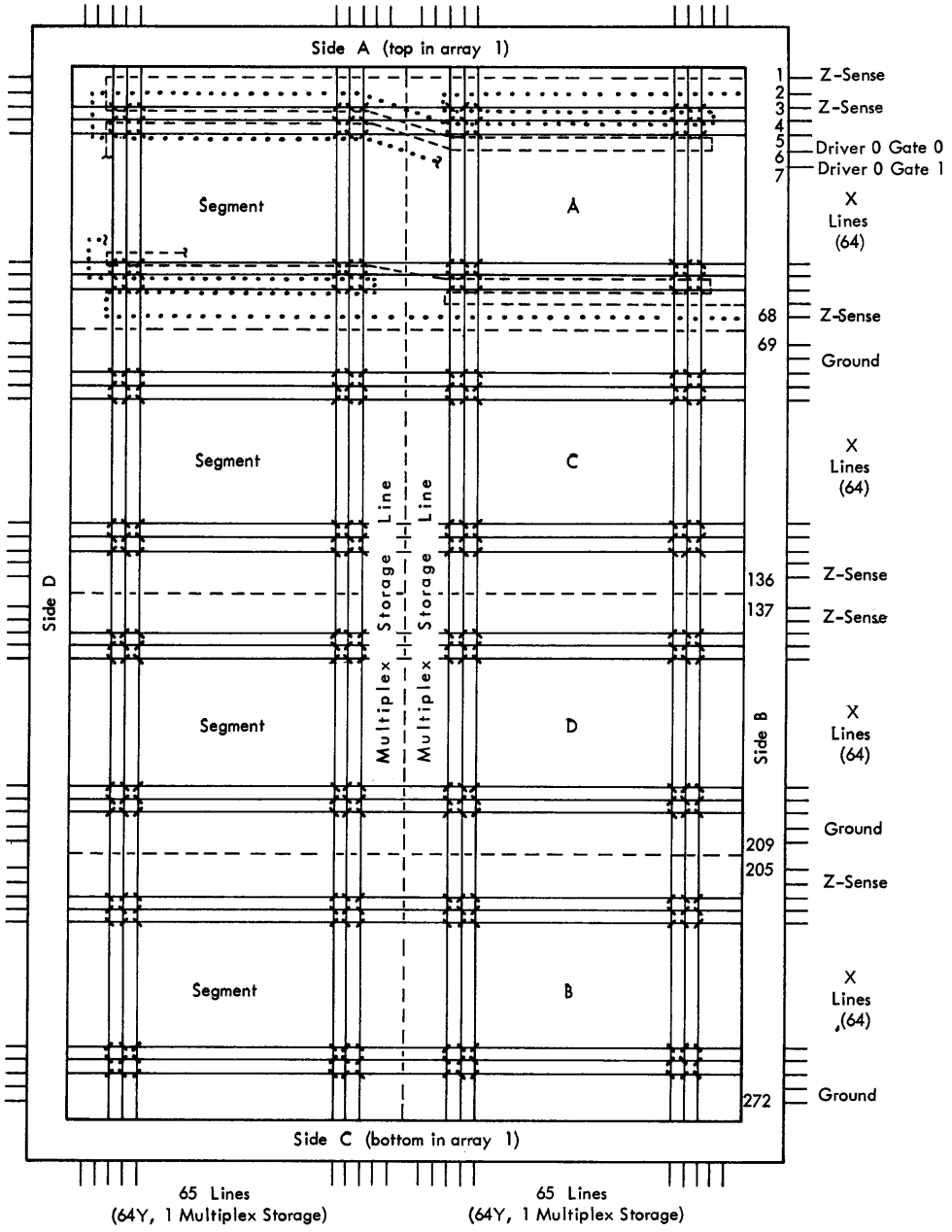


Figure 122. Plane Arrangement (Array 1, Invert for Array 2)

lines run parallel to the smaller dimension, each one passing through 128 cores.

In addition, the Y drive lines are divided into two groups of 65 each and the X drive lines are divided into four groups of 64 lines each. This division of the X and Y lines into groups causes the cores to be arranged in eight mats of 65 x 64 cores.

Two of these mats which are side by side along the smaller dimension, contain the cores for a sense/inhibit segment so that there are four sense/inhibit segments. From side A to side C, these segments are labeled: segment A, segment C, segment D and segment B.

The Y lines are connected between side A and side C to plane pins 3 to 67 and 70 through 134 counting from side B.

The X lines are connected between side B and side D to pins 3 to 66, 71 to 134, 138 to 202 and 207 to 270 counting from side A.

The sense/inhibit lines run parallel to the X lines and are connected along the B and D sides to pins that are not used for the X lines. These pins are: 1, 2, 67, 68, 69, 70, 135, 136, 137, 138, 203, 204, 205, 206, 271, and 272.

Both on side B and side D, pins 1, 2, 67, and 68 are used for the sense/inhibit line of segment A, pins 69, 70, 135 and 136 for the sense/inhibit line of segment C, pins 137, 138, 203 and 204 for the sense/inhibit line of segment D, and pins 205, 206, 271 and 272 for the sense/inhibit line of segment B.

The sense/inhibit line for each segment consists of two separate wires that are connected together at one end and that end connected to ground. The other two ends feed to the sense amplifier and the inhibit driver.

The two wires of the sense/inhibit line are staggered in such a way that an equal number of cores on any X line is on each leg of the sense/inhibit line (Figure 123). This reduces the effect of X half select noise by placing equal noise on each of the sense pre-amplifier inputs.

Figure 123 shows how the pins on the plane frame are connected together to complete a sense/inhibit

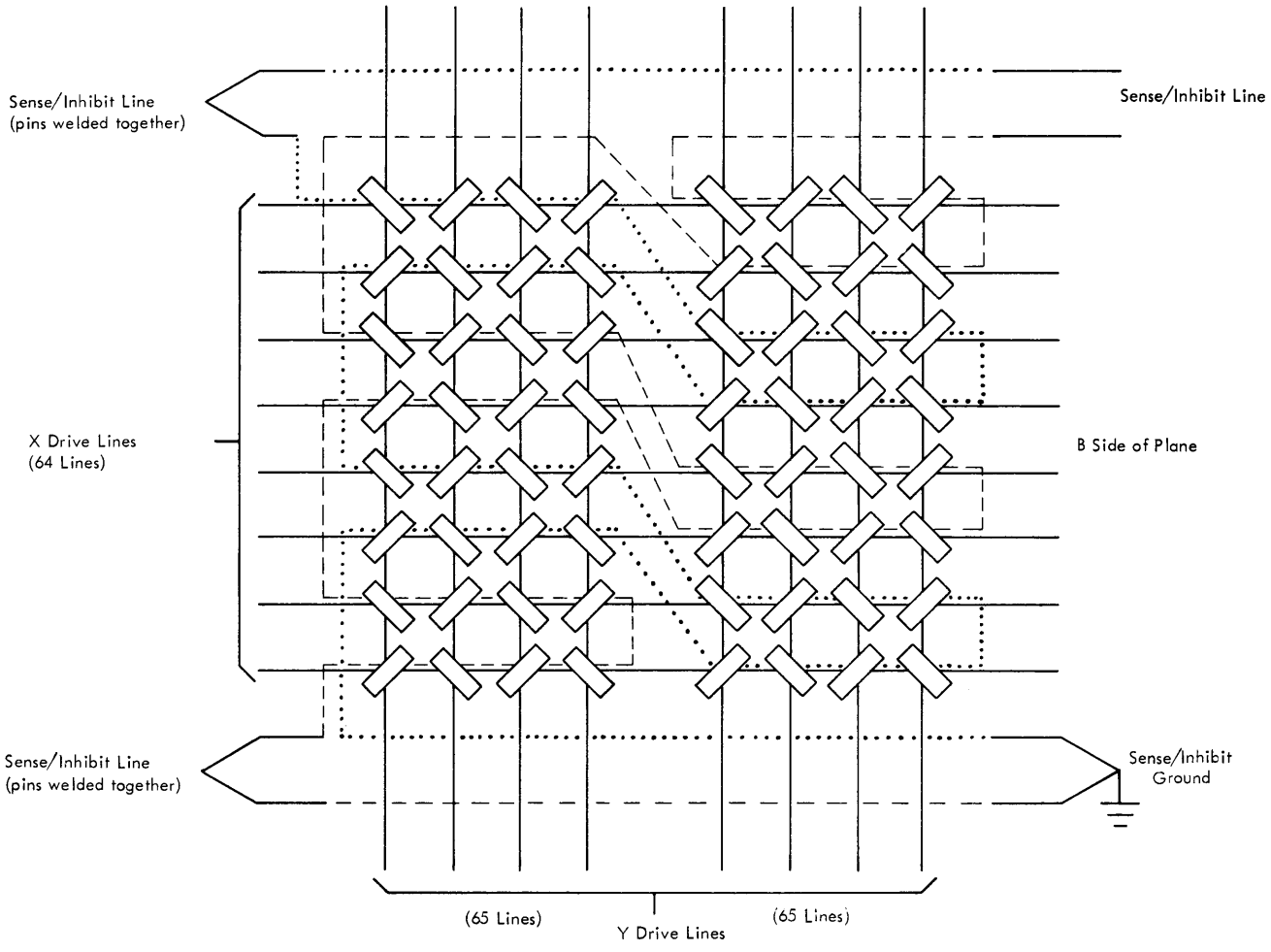


Figure 123. Sense/Inhibit Wire Routing

line. On the B side, the pins that go to ground are connected together and connected to the common sense/inhibit ground. The other two pins on side B are connected to wires that lead to the sense preamplifier and the inhibit driver.

In the 64K halfword storage, the pins on side B are connected as follows:

SEGMENT	GROUND PINS	SENSE/INHIBIT PINS
A	1 and 2	67 and 68
C	69 and 70	135 and 136
D	203 and 204	137 and 138
B	271 and 272	205 and 206

On side D the corresponding pins are welded together to connect the two legs of each sense/inhibit line, and allow the sense/inhibit cable to be connected to only one side of the array. The sense/inhibit pins on side D are welded together as follows:

SEGMENT	PINS WELDED TOGETHER
A	1 to 2, 67 to 68
C	69 to 70, 135 to 136
D	137 to 138, 203 to 204
B	205 to 206, 271 to 272

The Y drive line is wound between the X drive line and the sense/inhibit line to reduce coupling. See Figure 124.

X Drive Line Routing

X drive line connections are made from array end board D1 to plane 0 in array 1 and from array end board D3 to plane 0 in array 2.

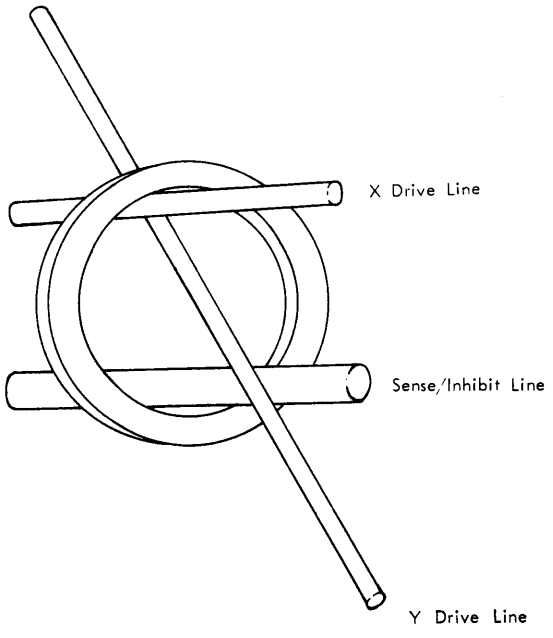


Figure 124. Core Winding Three Wire

One end of each of the 128 X lines in segments A and C of plane 0 is connected to the corresponding gate and array diode on the array end board.

Even numbered X lines in plane 0 have their ends at side B connected to the end board; odd numbered X lines in plane 0 have their ends at side D connected to the end board. See Figure 125. The other ends of the 128 X lines in segments A and C are connected to the corresponding X lines with next plane by welding their pins together so that all 128 X lines run through the segments A and C of all 18 planes. The X line ends at plane 17 are connected to the jumper board (D1 for array 1 and D4 for array 2).

On the jumper boards the X lines in segment A are connected to the lines in segment B and the lines in segment C to the lines in segment D. The 128 X lines run again through all planes and are again connected to the corresponding gates and array diodes on the array end board.

Y Drive Line Routing

One end of each of the 130 Y lines in plane 0 is connected to the corresponding gate and array diode on the array end board. The odd-numbered Y lines in plane 0 have their ends at side A connected to the end board. The even-numbered Y lines in plane 0 have their ends at side C connected to the end board. See Figure 126.

The other ends of the Y lines are connected to the corresponding Y lines in the next plane by welding their pins together so that all 130 Y lines run through all 18 planes. The Y line ends of plane 17 are connected to the jumper board and are returned via two flexible cables to the interface area on the array end board and from there to the appropriate gates and array diodes.

Sense/Inhibit Line Connections

The sense/inhibit lines with associated grounds are connected directly from the core plane pins on the B side of the array to SLT plugs, which are plugged into socket panel C1.

Array End Boards D1 and D3

A layout of array end board D1 is shown in Figure 127. The layout of array end board D3 is identical.

The array end board contains the gates and array diodes for each array. The interface area is shown in detail in Figure 128. The Y-line returns from the jumper board are connected to the interface area, and from there to the appropriate gates and drivers. All connections for the X and Y drive scheme between the

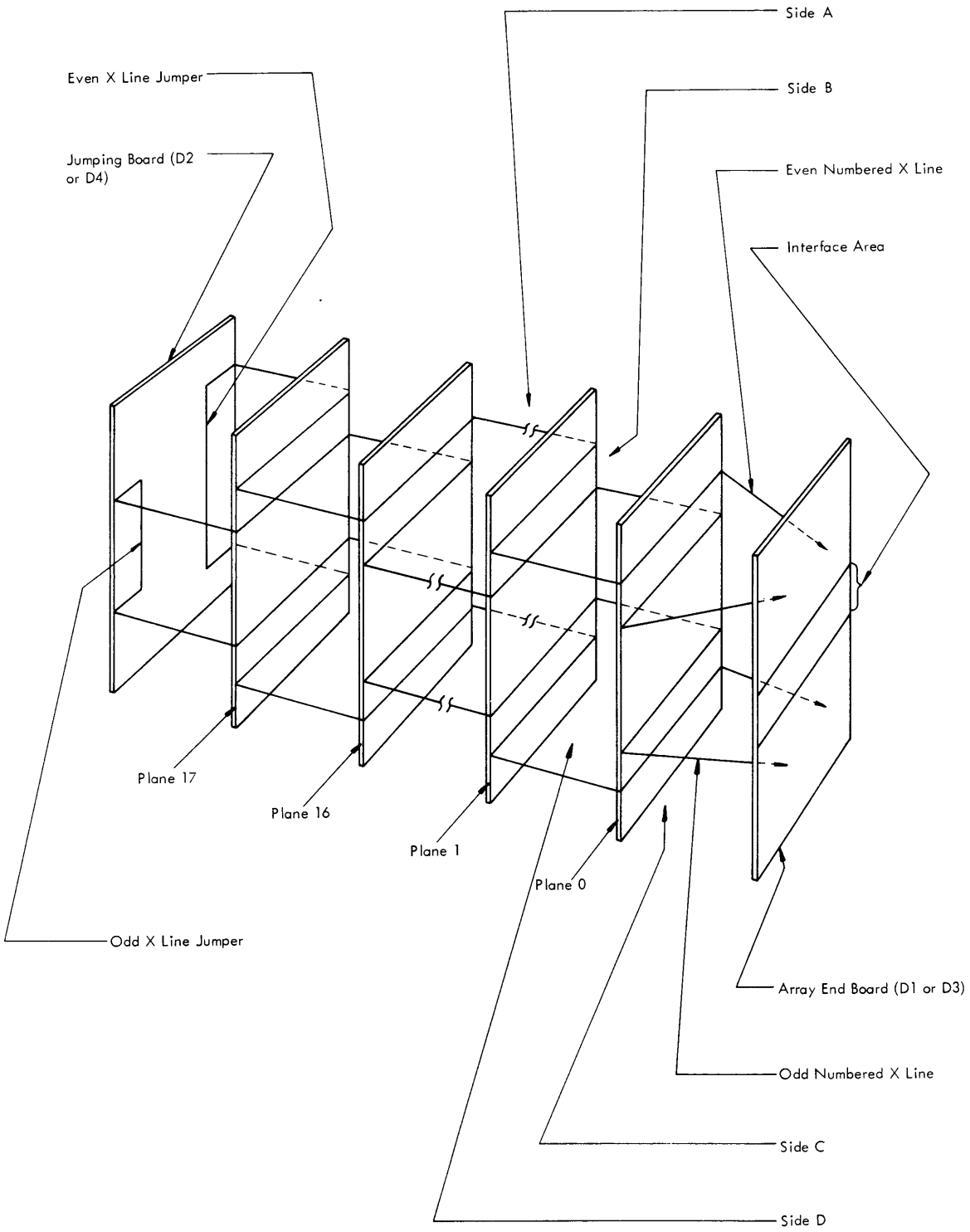


Figure 125. X Line Routing

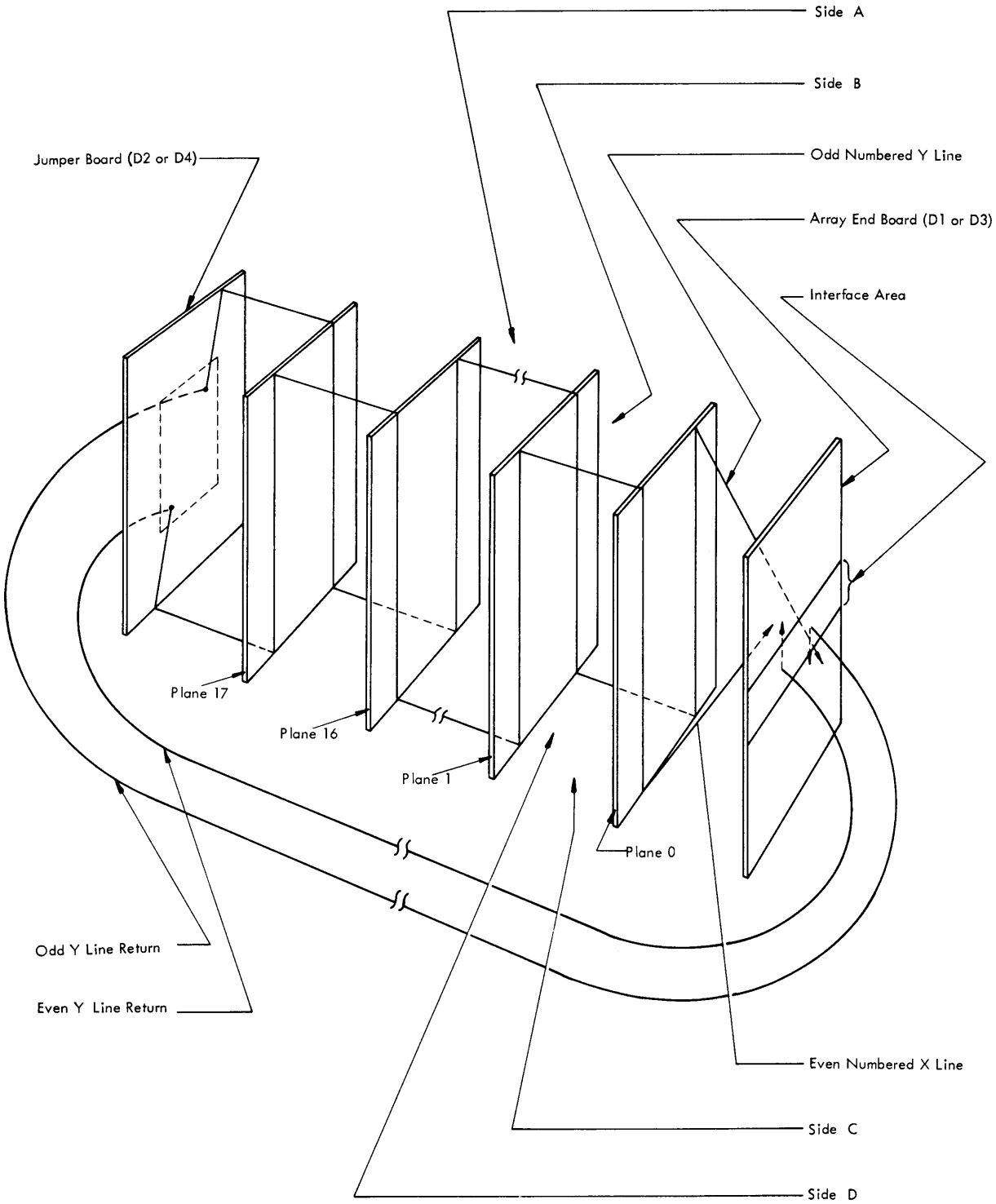


Figure 126. Y Line Routing

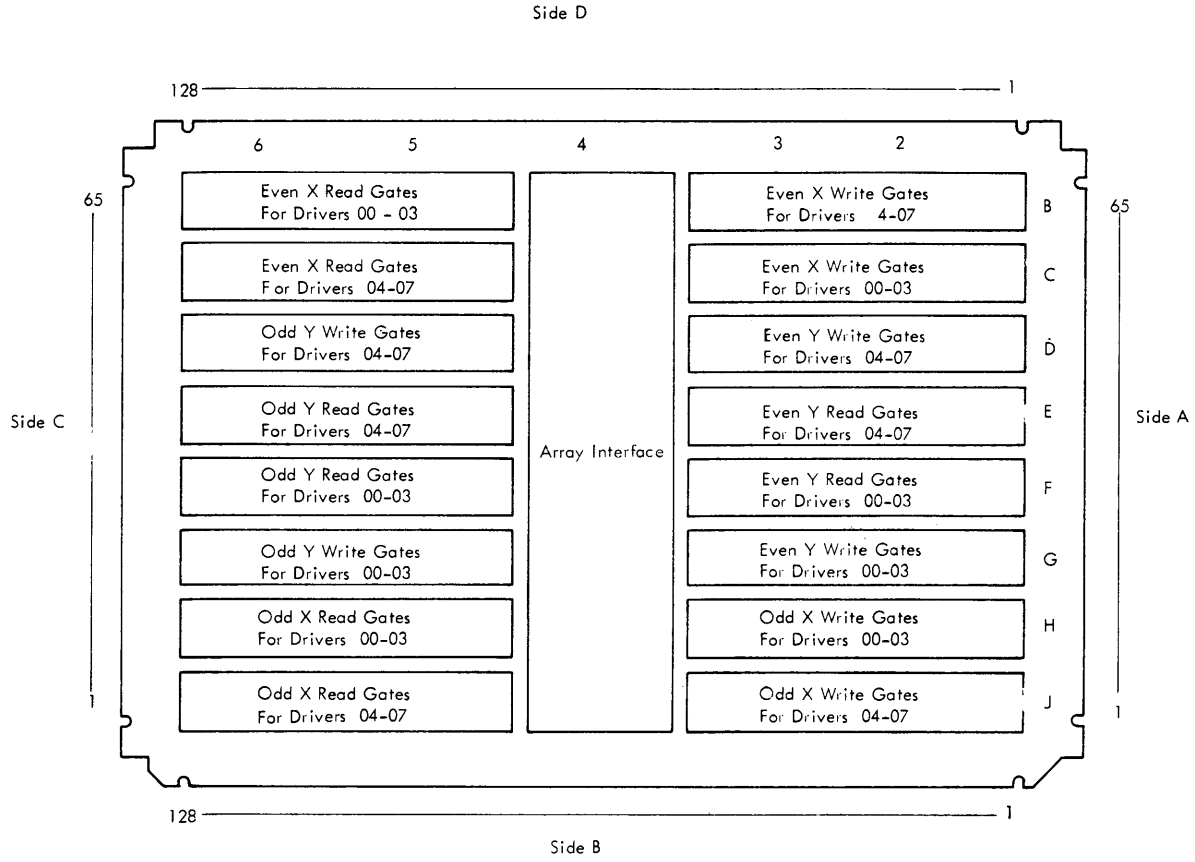


Figure 127. Array End Board D1

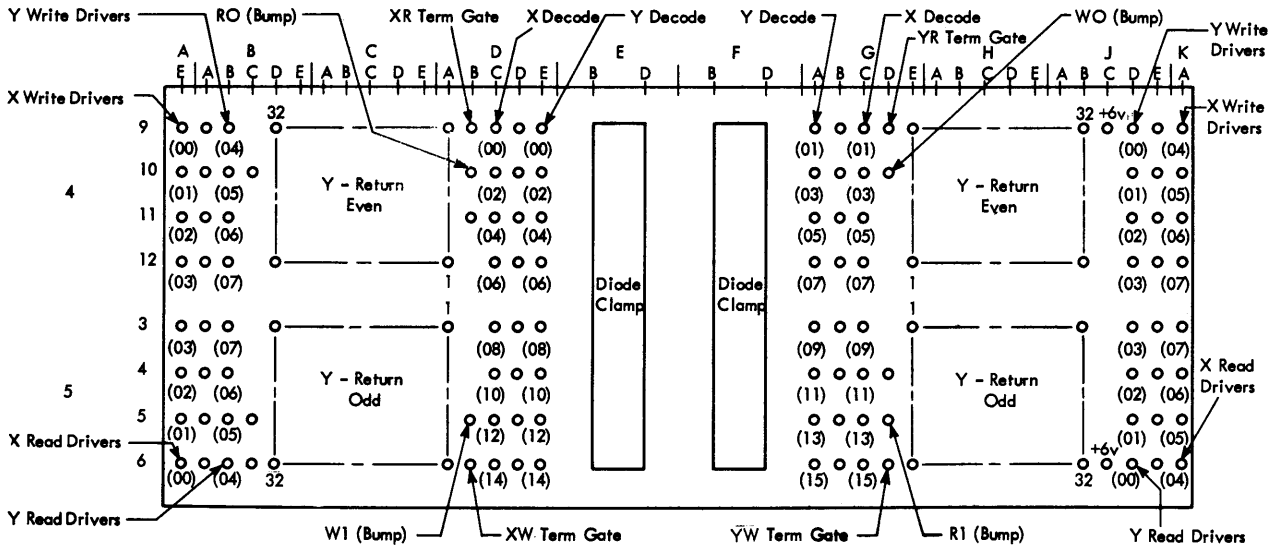


Figure 128. Interface Area of Board D1

large boards A1 and B1 and the array are made via the interface area. They are for each array:

Driver outputs (8 X read, 8 X write, 8 Y read, 8 Y write)
Gate decoder outputs (16 X, 16 Y)
Terminator gate outputs (X read, X write, Y read, Y write)
+6 volts to gate decoder clamps
Connections to the two multiplex Y lines (4).

Jumper Boards D2 and D4

A layout of jumper board D2 is shown in Figure 129. The layout of jumper board D4 is identical.

The jumper board contains the cross-over connections for the X lines and an area for connecting the Y drive line returns. On side B, the 32 even X lines of segment A are jumpered to the 32 even X lines of segment B and the 32 even X lines of segment C are jumpered to the 32 even X lines of segment D. On side D, the odd X lines are similarly jumpered. On side A, the 65 odd Y lines are connected from plane 17 of the array. On side C the 65 even Y lines are connected from plane 17.

Two flexible cables return the Y lines from the pins on the jumper board to the array end board.

Socket Panel C1

Figure 130 shows the layout of socket panel C1.

Cables from the arrays are plugged into the side shown and are clamped in place with phenolic strips. Cable wires from boards A1 and B1 are connected to the pin side of C1. Cables are color coded as follows:

blue, white, black } sense/inhibit lines
grey; four wires to terminator gates
black and brown; gate decode lines
black and orange; read-write drive outputs
purple; multiplex storage connections.

Board A1

Figure 131 shows the layout of board A1.

On board A1 are plugged the 2-36 SLT cards which contain the delay line clock, the control and timing circuits (except the bump driver timing latch), and the sense/pre-amplifier and Z drivers for the following segments:

SEGMENT	PLANE
A′	02-17
B′	02-17
C′	00-17
D′	00-17
C	16-17
D	16-17

The final amplifiers and strobe emitter followers for planes 00 through 17 are in array 2.

Board B1

Figure 132 shows the layout of board B1.

On board B1 are plugged the 2-36 SLT cards which contain the Z timing distribution, the Z clamps, the data-in and data-out powering, the address bit powering, the X and Y drivers, the X and Y gate decoders, the X and Y terminator gates, and the multiplex storage drivers, gates, gate decoders, array diodes, the bump driver timing latch, plus the sense preamplifiers and Z drivers for the following segments:

SEGMENT	PLANE
A	00-17
B	00-17
C	00-15
D	00-15
A′	00-01
B′	00-01

The final amplifiers and strobe emitter followers for planes 00 through 17 are in array 1.

Board A2

Figures 133 shows the layout of board A2. It carries the X, Y, and Z terminating resistors.

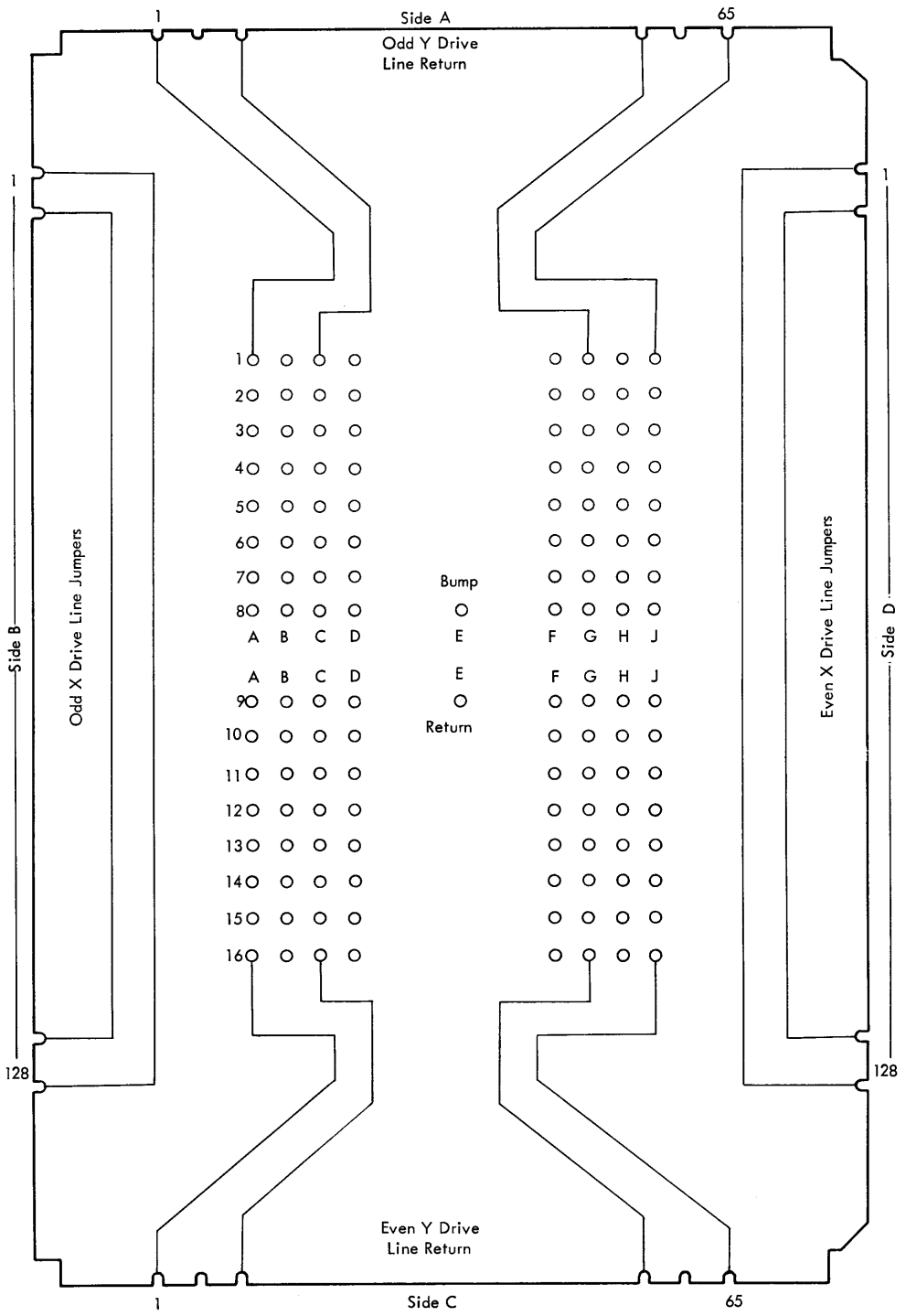


Figure 129. Jumper Board D2

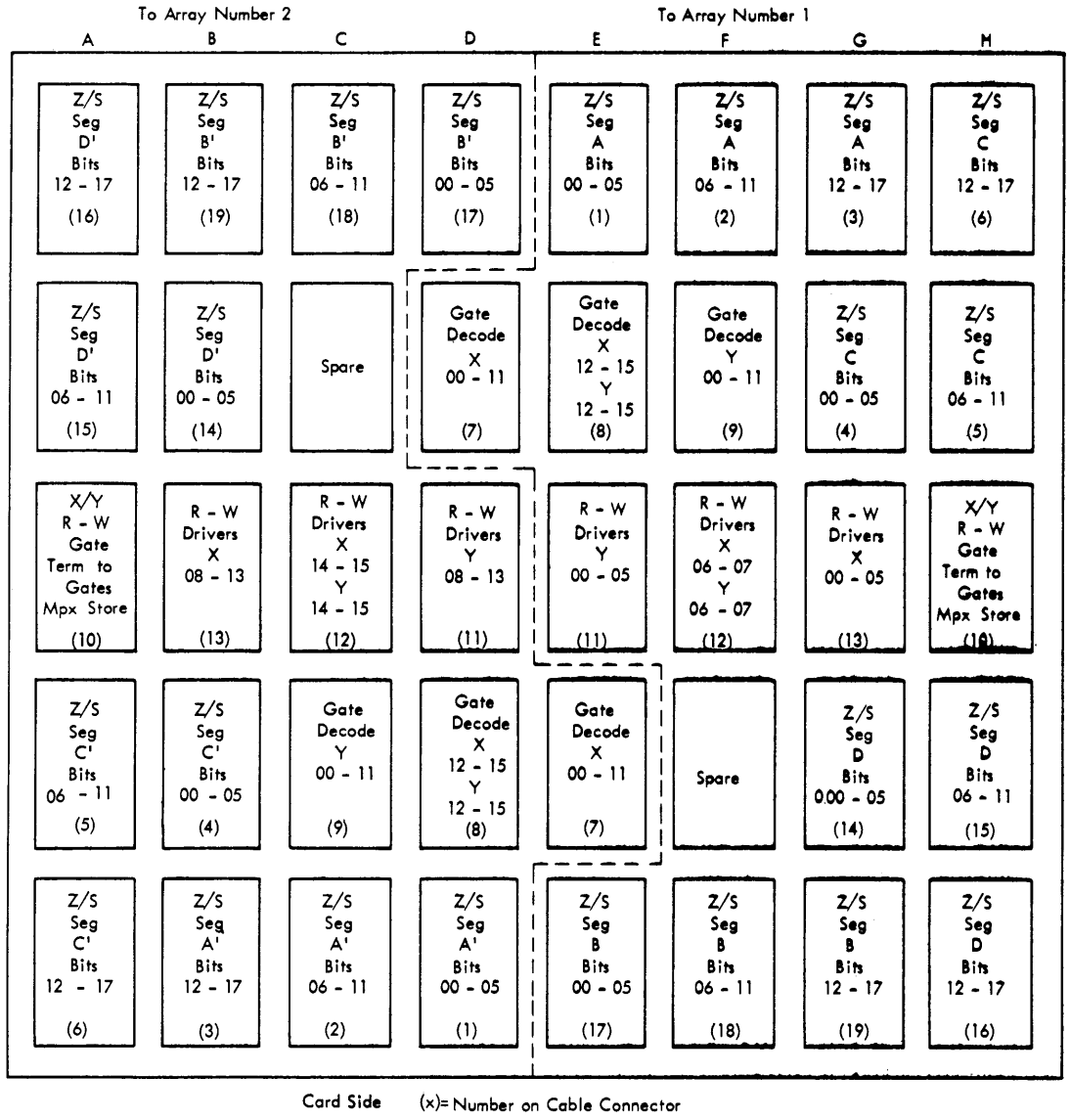


Figure 130. Socket Panel C1

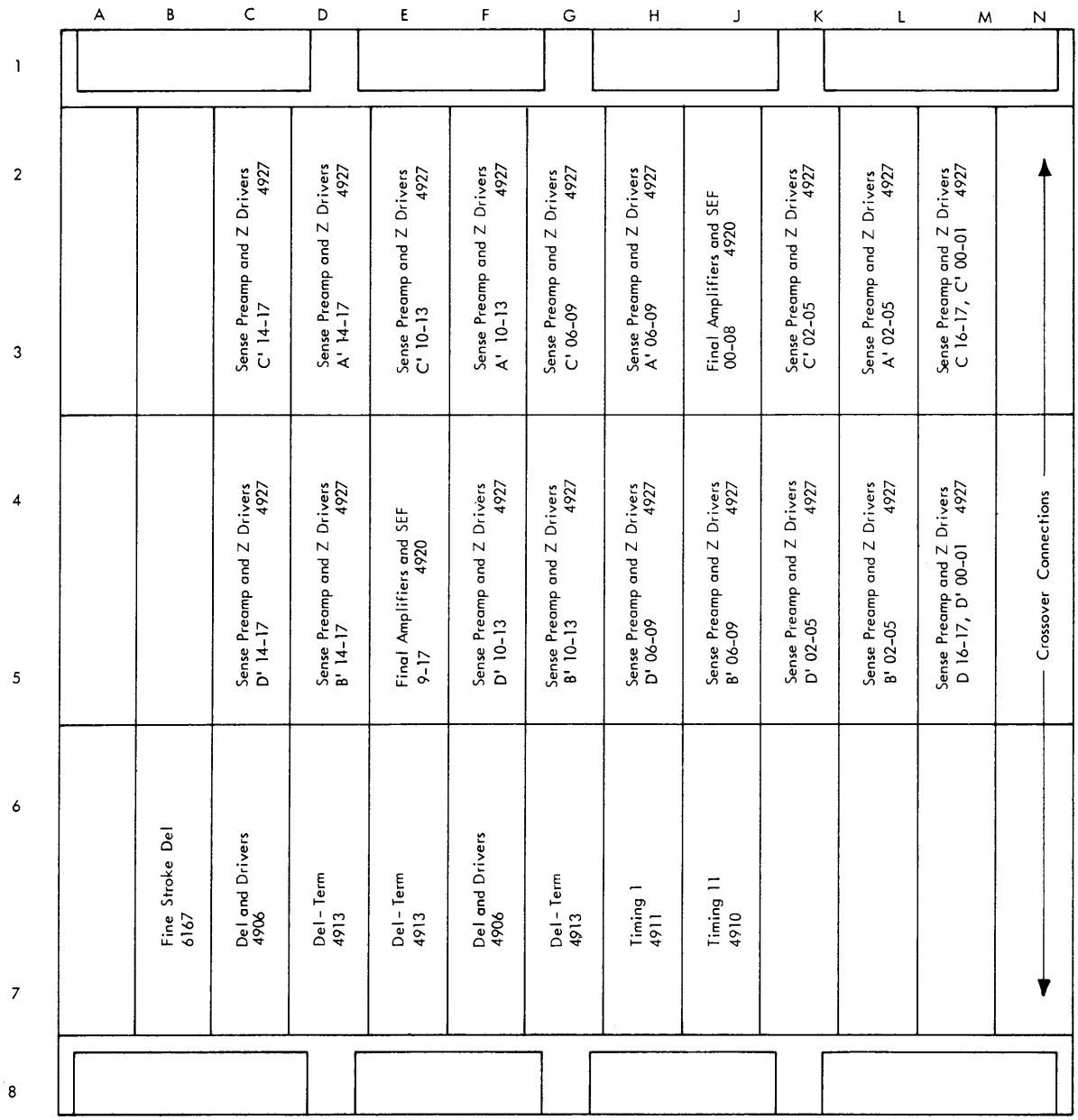
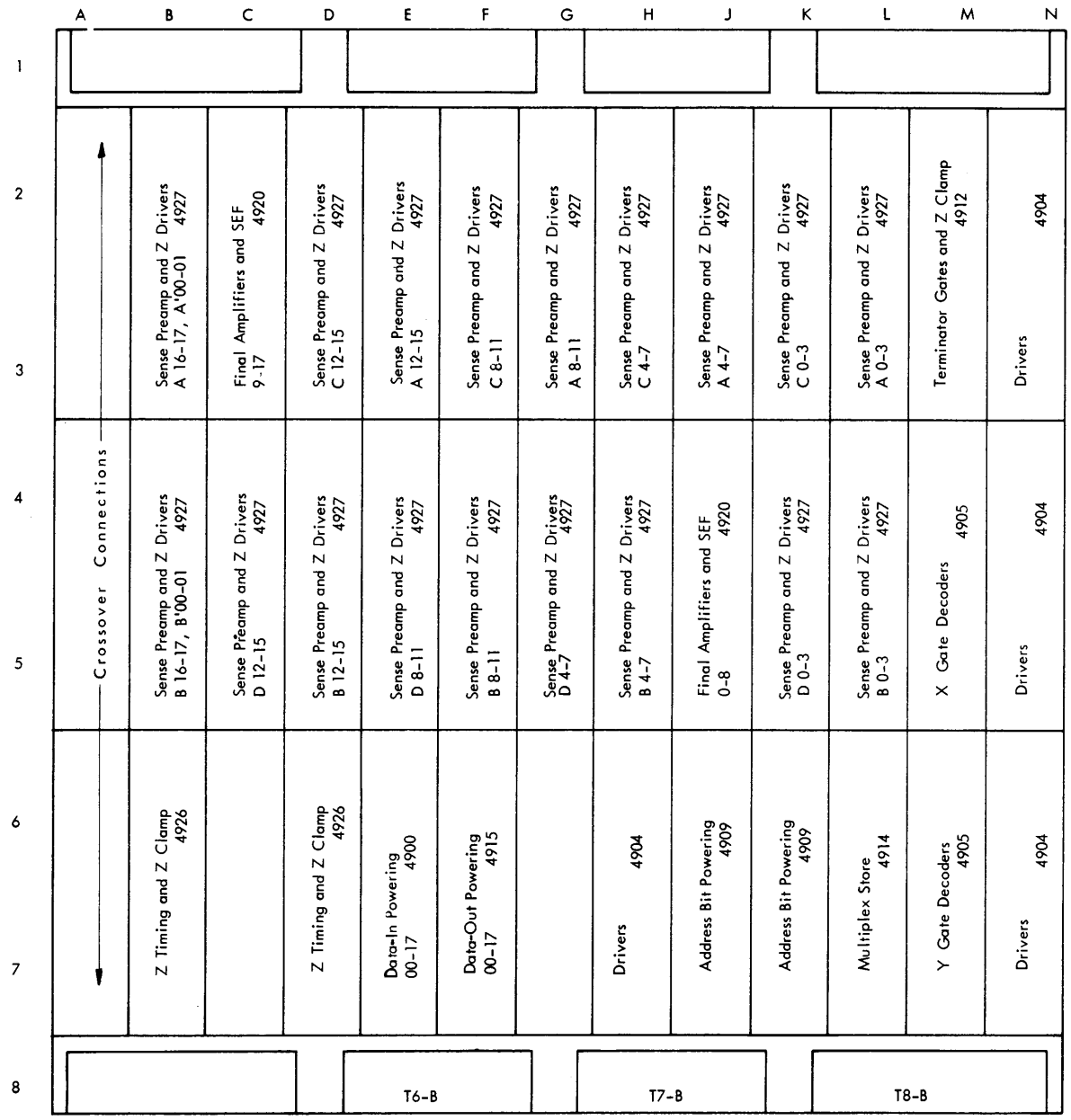


Figure 131. Board A1



Board B1, Card Side

Figure 132. Board B1 Layout

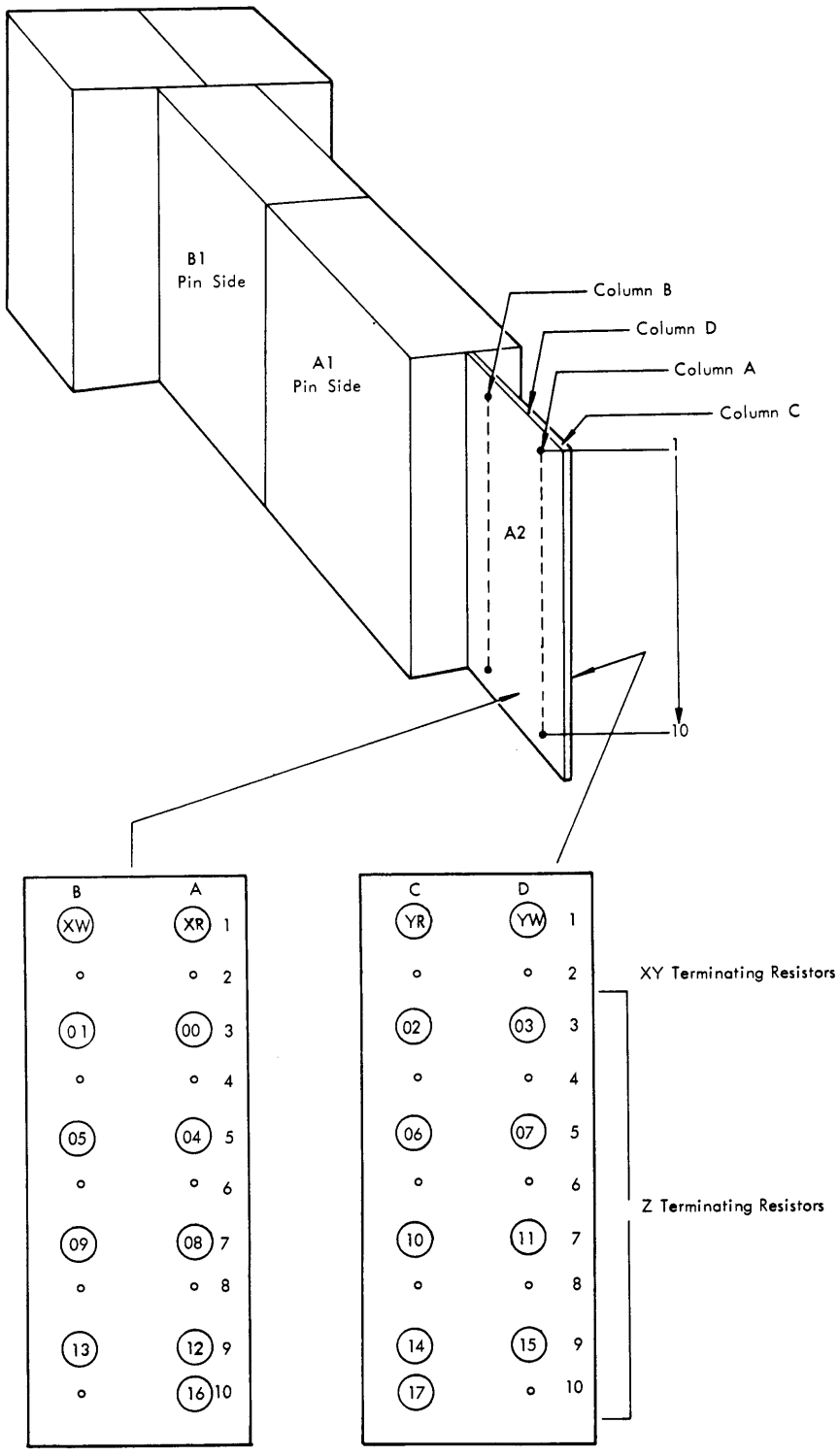


Figure 133. Board A2 Layout

Checking

- **Checking is incorporated in the IBM 2040 for two reasons:**
  1. To assure the user that results obtained from the system have a high degree of reliability
  2. To pinpoint most machine malfunctions to a certain defined area, thus facilitating fault location by customer engineers.

This section is concerned with system operation in case of machine malfunctions detected by logic circuits or microprogram tests during normal job execution.

For diagnostic procedures and fault-finding techniques refer to the appropriate sections of the *System/360 Model 40, 2040 Processing Unit, Field Engineering Maintenance Manual*, Form 223-2841. The function and logic of individual checking circuits are explained in this manual.

Simplified logic diagrams of all logic circuit checks may be found on ECAD's (Error Check Analysis Diagrams) in the ALD's.

Error Detection

- **Two types of machine malfunction detection are:**
  1. Logic.
  2. Microprogram.

Logic Detected Errors

Each check is individually indicated on the console panel or internal CE panel and where applicable, an over-all check light is also provided (e.g., two-wire checks). To aid the customer engineer, the individual error checks are OR'ed in groups by timing and function, and are latched up in three over-all check latches: control check, early check, late check. These check latches are also individually indicated on the console.

Overload and over-temperature sensing circuits are included in the power supply logic. Detection of either condition initiates the power off to sequence to protect the system from operation under marginal conditions and possible extensive damage to components not responsible for the initial malfunction.

Microprogram Detected Errors

Three micro-orders are available to terminate the current micro-routine when machine malfunctions are detected:

1. Start Log-Out; CB = 2 with CD = 3, CAS symbol: LOG
2. Stop T Clock; CB = 5 with CD = 3, CAS symbol: STPCL
3. Set Interface Control Check; CB = 10 with CD = 1, CAS symbol: ICC

Start Log Out

LOG is used in all invalid ROS control words (ROS control word not used, but TROS tape installed) to show that an unused word has been addressed or an invalid branch taken.

Invalid branches also contain (where possible) as next address, the address of the branch which most likely should have been taken. This allows the machine to carry on processing if the system is in error disable mode.

When LOG is given in disable mode, this condition is latched up as a control check.

Stop T Clock

STPCL is provided to generate the line stop T clock and is used only in diagnostic microprograms to hardstop the machine if errors are enabled.

Set Interface Control Check

ICC is used in channel programs to set the interface control check latch. Microprogram time-out loops are employed to detect failure of a control unit to respond with the correct tag within the allowed time.

The interface control check can also be set by hardware and forces a log out.

System Operation After Error Detection

- **With errors enabled, detection of a machine malfunction freezes the system as close to the malfunction as possible.**
- **For diagnostic purposes, the error environment is logged out or the machine is hard-stopped to allow analysis from the console.**
- **In normal processing mode, log out, CPU check out, and system reset are executed and the machine check interrupt routine is entered.**
- **An intermittent machine malfunction does not stop the system.**

Normal Processing Mode, Errors Enabled

A failure (Figure 134) detected by the overload or thermal circuits directly switches system power off.

Detection of any other machine malfunction stops the T clock at the end of the cycle in which one of the over-all check latches has been set (control, early, late).

Inhibit lines are raised to stop the system as close to the error as possible.

Example:
At the end of a T cycle ROS is already read out to provide the controls for the next cycle. By inhibiting the sense latch reset, the controls of the cycle in error are displayed. ALU operation: ALU control register and ALU entry P and Q are reset at T4 del, i.e. with the T clock stopped at the end of T4 del the old contents would be wiped out. With these resets inhibited, the ALU operation can be investigated.

With Y12 on, or console switch on error stop, the machine is now hardstopped (T clock stopped, hardstop latch on).

Without error stop a number of hardware cycles take place, during which various machine conditions and error data are written into local storage (log out).

The T clock is started and by microprogram the entire data flow (including the areas just set up in local storage), is written into permanent main storage locations, starting with address 80 hex (microprogram log out; for detailed log out operation refer to the *System/360 Model 40, 2040 Processing Unit, Field Engineering Maintenance Manual*, Form 223-2841).

Note that at the beginning of that routine, Y12 is set, which means that any machine malfunction detected in CPU check out and the subsequent routines will hardstop the system. During log out only control checks are enabled. CPU and channel check out takes place; only the channel that caused an error is checked (and reset). In this routine various microprogram tests are made. Unsuccessful results branch to microinstructions with the STPCL statement and consequently bring the system in hardstop.

Microprogram system reset takes place (for details see Maintenance Manual), Form 223-2841.

The machine check interrupt routine is entered, PSW's are exchanged. The routine that loads the machine check new PSW also resets Y12.

The system is again under control of a stored program.

From the above sequence, it can be seen that for an intermittent machine malfunction system, operation is not halted. It is assumed that a permanent failure which will require intervention by the customer engineer also becomes apparent during one of the routines executed with Y12 on and, therefore, does stop the system.

Normal Processing Mode, Errors Disabled

All check circuits are still active. Detection of any check sets its corresponding over-all latch but for early or late checks no further action is taken. Normal processing continues, the check latch remains on.

As soon as a new PSW is loaded which contains bit 13 = 1 the sequence for enabled errors takes place.

Power or thermal checks cannot be disabled.

Checks in Relation to Timing (Figure 135)

- **The T clock is stopped at the end of the cycle in which one of the over-all latches is set (control, early, late).**
- **The over-all check latches are normally set in the cycle in which an error is detected (exception: LS read check).**
- **The cycle in which an error is detected is not necessarily the cycle in which the error occurred.**
- **Reset of ROS sense latches is inhibited by control check and early check only; reset of ROBAR and ALU is inhibited by all three checks.**

For every machine malfunction three distinct machine cycles can be defined:

1. The cycle in which the error occurred.
2. The cycle in which the error is detected.
3. The cycle in which either control check, early check or late check is set and at the end of which the T clock is stopped.

Cycles 2 and 3, the cycle of error detection and the cycle of setting the check latch, coincide for all checks but one; local storage read check is detected at T4 and latched up. Due to close timing conditions, it is not possible to stop the system at the end of this cycle and to raise the various inhibits. The LS read check latch therefore is used to carry the error condition into the next cycle to set early check.

Depending on the checking circuit and the type of machine malfunction, cycles 1 and 2 can coincide, i.e. the error is detected in the same cycle in which it occurred. Examples of decoder checks: R, P, Q bus parity checks, ALU two-wire checks, etc.

There are, however, machine malfunctions possible which are detected only in the cycle after the one in which they occurred or even several cycles later.

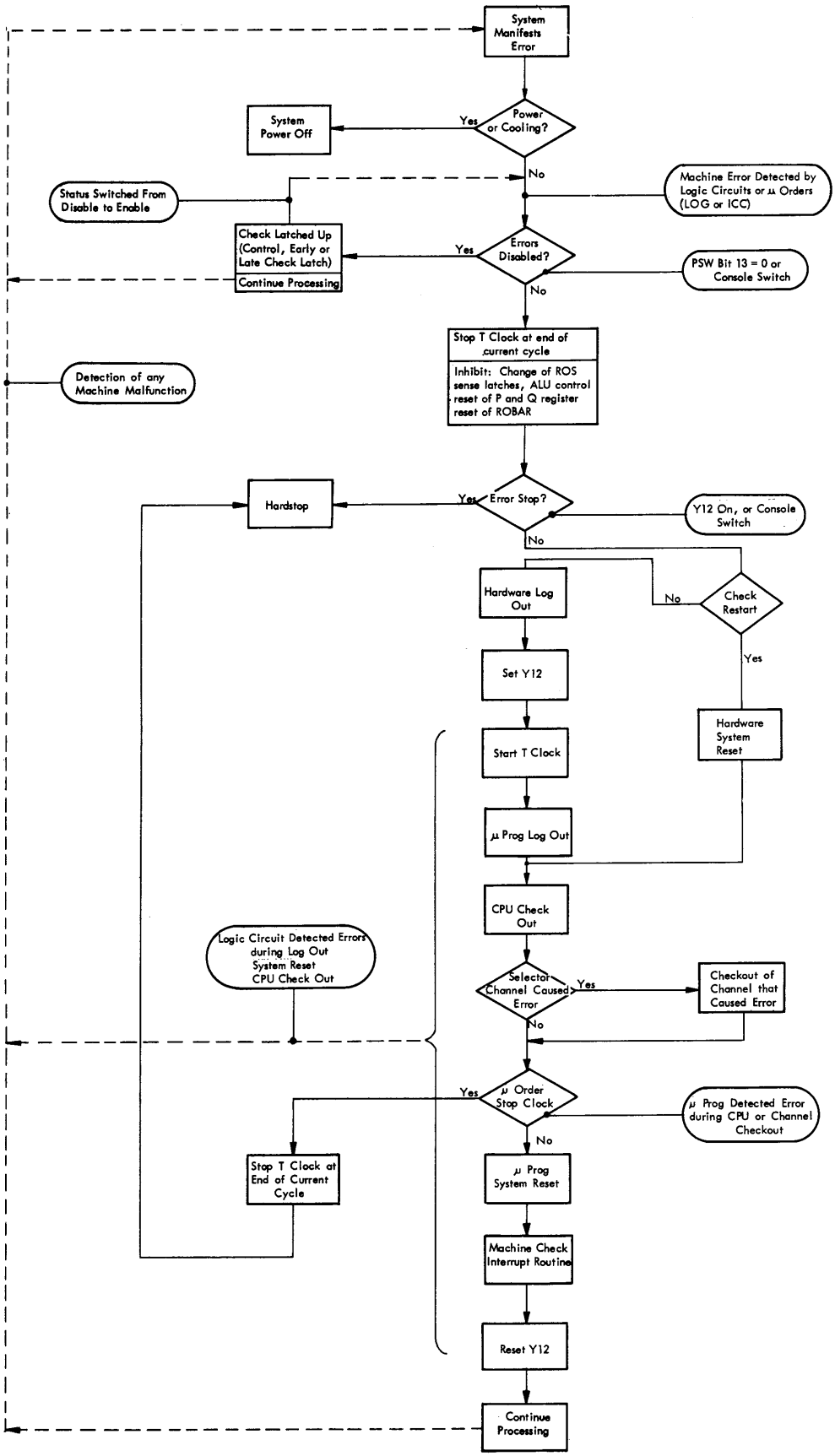


Figure 134. System Operation after Error Detection

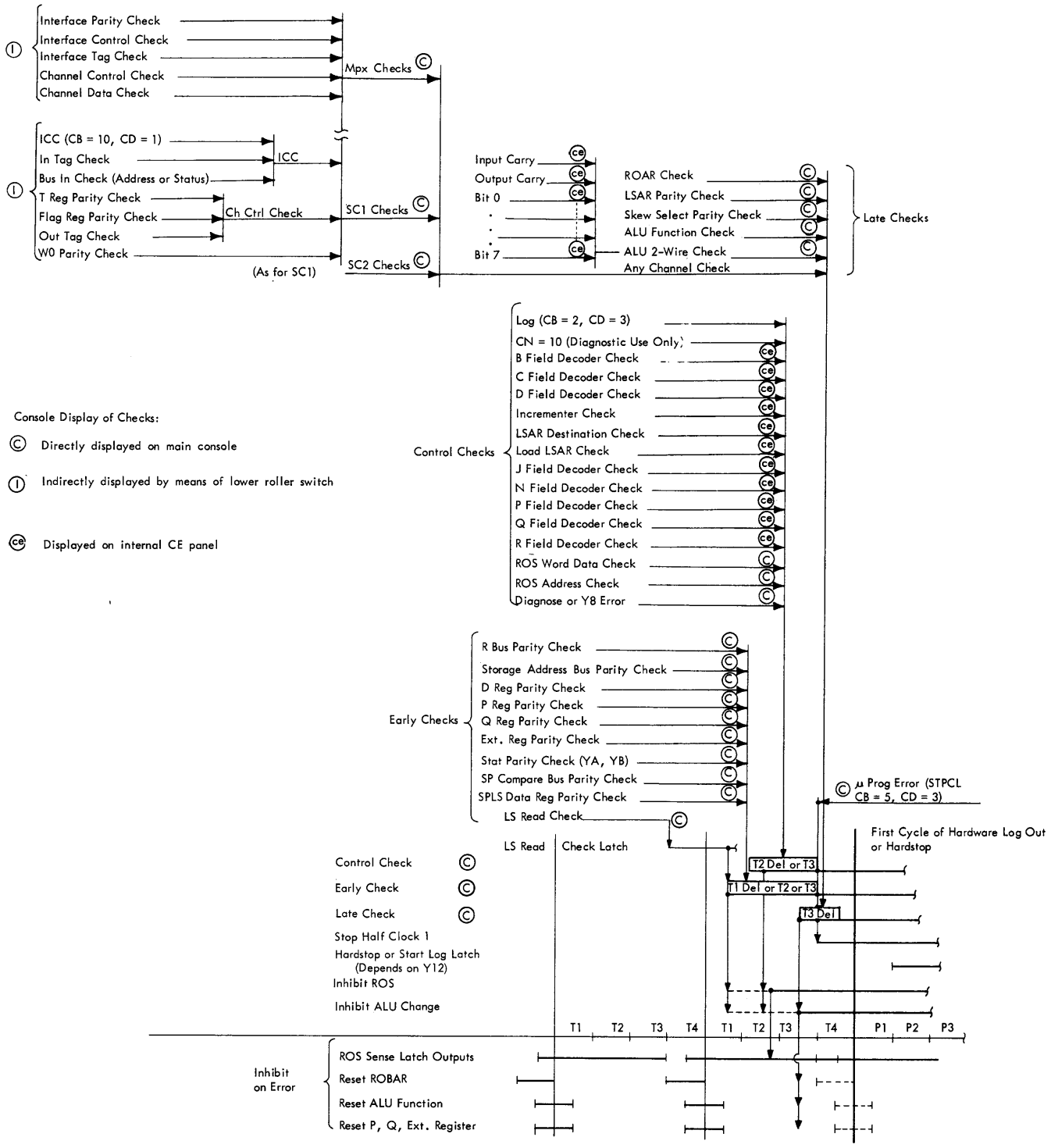


Figure 135. Checks in Relation to Timing

Examples:

1. D register parity, stat parity, etc., set the early check latch at time T1 del. Loading of these registers is normally at T3 or later. Consequently, if one of the latches fails to set properly, early check is set only in the next cycle.
2. All units which are not directly checked (e.g. A, B, C, H, J registers) lead to a check several cycles later when the erroneous information they may contain is gated to a checked unit.

While analyzing error conditions from the console it is also important to note that the stopping of the system is not identical for all checks. Only control and early checks prevent resetting of the ROS sense latches at T4 del before the system is stopped.

Reset of ROBAR, P, Q and ALU function however is inhibited by all three checks.

Other Machine Malfunctions

Machine malfunctions not detected by check circuits and microprogram checks described so far do not necessarily produce wrong results. Two major groups of errors belong to this category.

Microprogram Looping

These are malfunctions which lead to infinite looping of one or any number of microinstructions without producing a machine check. This type of error detection may be introduced deliberately because of lack of other possibilities for indication, e.g. in the IPL routine.

Machine Malfunctions Indicated as Program Check

In general, program checks cannot be caused by machine malfunctions (and machine checks cannot be caused by program errors). In I/O operations, however, definition of the two types may overlap.

Examples:

Reading of an invalid card code is signalled to the system as a program check (in the status byte), but this indication may actually be caused by a machine error in the card reader.

Channel overrun is assumed to be caused by programming I/O operations close to the channel capacity, but it may also be the result of a machine malfunction in logic circuitry.

Index

A clock generation ... 7
A register ... 11
  circuits ... 11
Add fixed-point halfword ... 92
ALU ... 30
  arithmetic operations ... 40
  binary arithmetic ... 39
  checking ... 51
  circuits ... 45
  compress ... 50
  conditions ... 45
  control ... 30
  decimal arithmetic ... 39
  extended carry ... 49
  glossary ... 51
  logical operations ... 40
  output parity ... 49
  registers ... 18
  shifts ... 40
  worksheet ... 41
ALU checking ... 51
ALU control ... 30
ALU extended carry ... 49
ALU function control ... 36
ALU operation ... 40
ALU output parity ... 49
Arithmetic
  binary ... 39
  decimal ... 39
  operations ... 40
ASCII stat ... 22

B clock generation ... 9
B condition tests ... 43
B register ... 14
Binary arithmetic ... 39
Block diagrams
  data check ... 72
  local storage ... 113
  microprogram ... 75
  4K TROS ... 59

Branch and link instruction
  RR format ... 89
  RR format operation ... 89
  RX format ... 89
  RX format operation ... 89
Branch leg identifier ... 82
Branch leg selector ... 82
Branch on condition ... 90
Branch on count ... 90

C condition tests ... 43
C register ... 14
Carry box ... 48
Carry control ... 42
  YCD ... 42
  YCI ... 42
Carry latches
  YCD ... 42
  YCI ... 42
CAS ... 77
  interpretation ... 78
  priority ... 83
  sequence ... 79
  timing ... 83
CAS block symbols
  arithmetic statement ... 78
  data transfer statement ... 79
  emit statement ... 78
  local storage address statement ... 79
  main control statement ... 79
  TROS address control statement ... 79
CE ROS field ... 35
CG ROS field ... 35
Charts and tables
  ALU control signals and functions ... 32
  CH field control chart ... 105
  main storage addressing parity ... 167
  phase reversal control ... 146
  R bus entry ... 21
  registers ... 12
  staticizers ... 25
  TROS control word and function ... 64
Checks
  errors disabled ... 187
  errors enabled ... 187
  program checks ... 190
  timing relationship ... 187
CK ROS field ... 36
Clock
  A clock generation ... 7
  B clock generation ... 9
  purpose ... 7
  T clock ... 9
CM ROS field ... 34
Complement add ... 40
Compress ALU output ... 50
Condition tests
  B (CPU state) ... 43
  C (CPU state) ... 43
Connect box ... 47
Console operations-forced addresses ... 67
Control latches ... 66
Convert to binary ... 90
Convert to decimal ... 91
CP ROS field ... 32
CPU state stat ... 22
CQ ROS field ... 34
CR ROS field ... 36
CT ROS field ... 36

D register ... 14
Data register (local storage) ... 108
Data registers ... 64
Dataflows
  storage protect ... 126
  system ... 65
Decimal arithmetic ... 39
Decimal correction ... 49
Decimal filler ... 46
Decoding of ROSAB ... 62
Determining next TROS address ... 79
  branch leg identifier ... 82
  branch leg selector ... 82
  sequence example ... 82
Drive lines
  main storage ... 148, 154
  main storage routing ... 173, 176
  multiplex storage ... 169
  storage protect ... 131-136

Dump forced addresses ... 67

Emit circuits ... 64
Error detection
  IF control check ... 186
  logic errors ... 186
  microprogram errors ... 186
  start log out ... 186
  stop T clock ... 186
Error handling
  errors disabled ... 187
  errors enabled ... 187
  microprogram looping ... 187
  program checks ... 190
  timing relationship ... 187

Ferrite cores ... 97
Flow charts
  determining next TROS address ... 68
  effects of ROS fields on ALU ... 33
  system operation after error detection ... 188
FNB format ... 66
Forced addresses ... 67
  console operations ... 67
  dump ... 67
  selector channel operations ... 68
  TRAP ... 67

H register ... 103
Halt stat ... 22
Hysteresis loop ... 98

I-fetch ... 84
  entry to I-fetch ... 85
  exits of I-fetch ... 85
  first halfword I-fetch ... 85
  I-fetch microprogram ... 87
  second level I-fetch RS and SI operations ... 88
  second level I-fetch RX fixed-point ... 87
  second level I-fetch RX floating-point ... 88
  second level I-fetch SS decimal ... 89
  second level I-fetch SS logical ... 88
  sequence of I-fetch ... 85
I-fetch entry ... 85
I-fetch exit ... 85
I/O mode stat ... 22
Initial program load microroutine ... 74
ISA detection ... 131
ISA stat ... 22

J register ... 103

Laminar bus ... 58
Latches
  ASCII ... 29
  CPU B cond ... 28
  CPU state ... 29
  enable ... 29
  halt state ... 29
  hardstop ... 28
  IDQ ... 28
  incr +1, −1, −2 ... 109
  IZT ... 28
  PSA, ISA ... 132
  read ... 163
  read control ... 121
  read cycle 2 ... 163
  ROBAR ... 70
  ROS address check ... 71
  ROSAB bit P ... 71
  select hold ... 122, 140, 141
  SPLS parity ... 129
  start log out ... 28
  stat parity bit ... 23
  stat parity generation ... 23
  TRAP ... 132
Latches (cont'd)
  TRAP – force ROS address ... 132
  wait ... 29
  write ... 163
  YC ... 44
  YCD ... 44
  YCI ... 44
  Y0, Y1, Y2, Y3 ... 26
  Y10/manual ... 27
  Y12 ... 28
  Y15/load ... 28
  Y8/ID ... 27
  Y9/MI ... 27
Load PSW from MS into LSTOR microroutine ... 74
Load PSW microroutine ... 74
Local storage
  address loop ... 103
  address loop checking ... 106
  address loop controls ... 103
  address register ... 103
  addressing ... 114
  data register ... 108
  decoder ... 108
  destination control line ... 108
  drive scheme ... 114
  incrementer ... 103
  incrementer control line ... 108
  inhibit ... 115
  local storage unit ... 108
  LSAR destination ... 106
  read controls ... 120
  read cycle ... 122
  read operation ... 112
  registers in address loop ... 103
  source control line ... 108
  sense ... 115
  set and reset of LSAR ... 105
  timing ... 120
  write controls ... 120
  write cycle ... 122
  write operation ... 112
  X line driving and gating ... 114
  Y line driving and gating ... 115
Local storage timing ... 120
Local storage unit ... 108
Log out microroutine ... 74

Machine instructions
  branch and link ... 89
  branch on condition ... 90
  branch on count ... 90
  convert to binary ... 90
  convert to decimal ... 91
  fixed-point halfword add and subtract ... 92
  fixed-point multiply ... 93
  floating-point multiply ... 94
  shift instructions ... 95
Magnetic core storage ... 97
  coincident current addressing ... 100
  controlling the core ... 100
  ferrite cores ... 97
  hysteresis loop ... 98
  magnetic properties ... 97
  phase reversal ... 102
  reading out a core ... 100
  split-cycle operation ... 102
  writing into a core ... 100
Main storage
  arrays ... 147
  capacity ... 143
  core locations ... 144
  general ... 143
  plane layout ... 148

Main storage (cont'd)
power supplies ... 169
speed ... 143
storage cycle ... 145
three wire system ... 176
64K storage unit ... 172
Main storage physical layout
array layout ... 172
jumper boards ... 179
physical structure ... 170
sense/inhibit line connections ... 176
X drive line routing ... 176
Y drive line routing ... 176
Main storage-64K halfword
address registers ... 145
addressing ... 146
control ... 159, 161, 164
D register ... 167
data register ... 167
drive current ... 148
drive line selection ... 148
drive scheme ... 148, 154
external controls ... 166
inhibit ... 157
Mpx timing ... 164
multiplex storage ... 168
read cycle ... 147, 166
read/write control ... 161, 164
sense ... 159
sense/inhibit line ... 157
storage address bus (SAB) ... 167
strobe A and B ... 164
termination ... 148
terminator gate timing ... 164
timing ... 159, 161, 164, 166
write cycle ... 147, 166
Microprogramming
block diagram ... 74
error handling ... 77
Microroutines ... 74
initial program load ... 74
load PSW ... 74
load PSW from MS into LS ... 74
log out ... 74
SC1 buffer ... 74
stop loop ... 74
store CSW ... 74
store PSW from LS into MS ... 74
store/display ... 74
system reset ... 74
update PSW ... 74
Multiplex storage
circuit description ... 169
Mpx addressing ... 168
Y line drive ... 169
Y line selection ... 169
Multiply fixed-point ... 93
Multiply floating-point ... 94
P clock pulses
pulse generation ... 7
P input latches ... 45
P minus Q ... 40
P plus Q ... 40
P register ... 18
Paraphase amplifier ... 7
PSA detection ... 127
PSA stat ... 22
Q input latches ... 45
Q register ... 18
R register ... 15
Reading out a core ... 100
Registers ... 11
A register ... 11
ALU control ... 18
ALU function ... 18
B register ... 14
C register ... 14
channel check ... 20
channel flags ... 20
channel key ... 20
CPU/Mpx key ... 125
D register ... 14
extension ... 18
H register ... 103
J register ... 103
local storage address loop ... 103
local storage address registers ... 103
local storage data registers ... 108
main storage address ... 145
main storage data ... 167
multiplex (data) ... 19
multiplex (error) ... 19
P register ... 18
Q register ... 18
R register ... 15
ROAR ... 20
ROBAR ... 21
ROSCAR ... 20
S register ... 20
selector channel key ... 125
skew buffer ... 19
skew select ... 19
status ... 20
storage protect (data) ... 19
storage protect (key) ... 19
storage protect data ... 125
T register ... 20
W register ... 20
Right shift unit ... 46
ROAR ... 20, 69
ROBAR ... 21, 69
ROS fields
CE ... 35
CG ... 35
CK ... 36
CM ... 34
CP ... 32
CQ ... 34
CR ... 36
CT ... 36
ROSCAR ... 20
RS operation I-fetch ... 88
RX fixed-point I-fetch ... 87
RX floating-point I-fetch ... 88
S register ... 20
SAT ... 131
SC1 buffer microroutine ... 74
Selector channel forced addresses ... 68
Selector channel registers
check ... 20
key ... 20
S register ... 20
status ... 20
T register ... 20
W register ... 20
Set and reset of LSAR ... 106
Shift instructions ... 95
Shifts ... 40
SI operation I-fetch ... 88
Simplified logic diagrams
A register ... 13
ALU ... 31
ALU carry ... 44

Simplified logic diagrams (cont'd)
ALU parity generation ... 49
ALU sum generator ... 47
D register parity ... 167
DAQX to ROAR check ... 73
function and control registers ... 37
function check signals ... 38
hardware entry to Q bus ... 15
inhibit drive ... 118
inhibit, SPLS ... 136
key registers ... 128
local storage address loop ... 104
local store R/W controls ... 121
local store timing circuit ... 122
LSAR destination, H and J regs ... 111
Mpx Y drive ... 169
MS sense ... 162
MS sense amplifiers, inhibit ... 158
MS unit drive ... 153
MS X drive ... 155
MS Y drive ... 156
MS, inhibit drive ... 160
parity check/change—LSAR ... 110
PSA and ISA ... 130
ROBAR ... 70
SAT and TRAP ... 132
sense ... 119
sense, SPLS ... 138
set and reset LSAR ... 107
set and reset of ROBAR ... 21
skew select ... 46
staticizer parity bit generation ... 23
stats (halt, wait, enable, ASCII, I/O) ... 29
storage protect ... 141
storage protect address diagram ... 127
storage protect timing ... 140
storage protect parity ... 129
storage select ... 163
TROS parity ... 71
X drive ... 116
X line drive, SPLS ... 135
Y drive ... 117
Y line drive, SPLS ... 137
YA and YB parity change ... 24
YA stats ... 26
YD stats ... 27
YE stats ... 28
Skew feature ... 46
Special formats ... 67
SPLS address bus
Microprogram looping ... 190
SS decimal I-fetch ... 89
SS logical I-fetch ... 88
Staticizers (stats)
ASCII ... 22
CPU state ... 22
enable ... 22
halt ... 22
I/O mode ... 22
ISA ... 22
PSA ... 22
wait ... 22
YA ... 22
YB ... 22
YCD ... 22
YCI ... 22
YD ... 22
YE ... 22
Stop loop microroutine ... 74
Storage protect
address bus ... 123
addressing ... 134
Storage protect (cont'd)
block diagram ... 139
CPU/Mpx key register ... 125
data register ... 123
functional unit circuits ... 123
general ... 123
inhibit ... 136
ISA detection ... 131
operation ... 125
PSA detection ... 127
purpose ... 123
read/write controls ... 136
reading ... 138
SAT and TRAP ... 131
selector chan key register ... 125
sense ... 136
storage assignment ... 125
storage protect local storage ... 131
timing ... 136
writing ... 133, 138
X-line drive ... 133, 134
Store CSW microroutine ... 74
Store PSW from LSTOR into MS microroutine ... 74
Store/display microroutine ... 74
Subtract fixed-point halfword ... 92
System reset microroutine ... 74
T clock ... 9
T clock pulses
pulse generation ... 7
T register ... 20
Timing charts
ALU timing chart ... 46
basic clock pulses ... 7
CAS timing ... 83
error checks ... 187
local storage ... 112, 121
main storage ... 165
ROBAR timing ... 70
T clock start/stop controls ... 10
TROS parity bit set ... 71
TROS timing ... 60
Timings
error handling ... 187
main storage ... 159, 164, 166
storage protect ... 136
TRAP ... 131
TRAP forced addresses ... 68
TROS
addressing ... 58, 66
CE field ... 35
CG field ... 35
checking ... 69
CK field ... 36
CM field ... 34
control field ... 32
control word ... 64
core-carrier ... 56
CP field ... 32
CQ field ... 34
CR field ... 36
CT field ... 36
data flow ... 64
decoding ... 62
drive scheme ... 62
latches ... 66
module ... 54
parity bits ... 66
physical construction ... 53-57
physical description ... 53
principle (figure) ... 52
principles of operation ... 52
tapes ... 53

TROS (cont'd)
timing ... 58
transformer ... 53
TROS address registers
ROAR ... 20
ROBAR ... 21
ROSCAR ... 20
TROS addressing ... 58
TROS array ... 58
TROS control word ... 64
TROS drive scheme ... 62
TROS I core ... 56
TROS module ... 54
TROS physical description ... 53
TROS resistance tape ... 54
TROS tape numbering ... 54
TROS tape stagger ... 54
TROS tapes ... 53
TROS timing ... 58
TROS transformer ... 53
True add ... 39
Two wire checking ... 45

Undump format ... 67
Update PSW microroutine ... 74

W register ... 20
Wait stat ... 22
Writing into a core ... 100

XOR box ... 48

YA stat ... 22
YB stat ... 22
YCD stat ... 22
YCI stat ... 22
YD stat ... 22
YE stat ... 22

READER'S COMMENT FORM

IBM System/360 Model 40 Functional Units, FEMI SY22-2843-1

- How did you use this publication?

  As a reference source ☐
  As a classroom text ☐
  As ..... ☐

- Based on your own experience, rate this publication...

	..... Very Good	..... Good	..... Fair	..... Poor	..... Very Poor
As a reference source:					
As a text:					

- What is your occupation? .....

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

CUT ALONG THIS LINE

YOUR COMMENTS, PLEASE . . . . . .

Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 419
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . . . . .

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

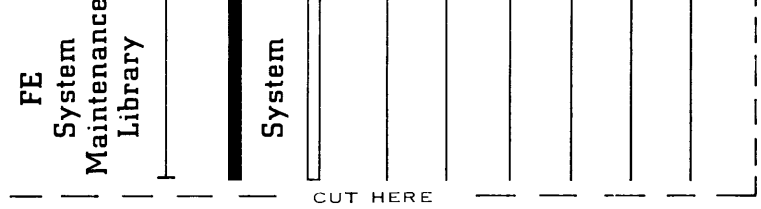
ATTENTION: FE MANUALS, DEPT. B96

fold fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]



SY22-2843-1

Printed in U.S.A. SY22-2843-1

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]