AD-A063 003 QUESTRON CORP SAN DIEGO CALIF F/G 9/2
RADIATION HARDENED MICROPROCESSOR VOLUME I.(U)
MAY 78 V V NICKEL, P A ROSENBERG F33615-77-C-1001
UNCLASSIFIED AFAL-TR-78-55-VOL-1 NL

1 OF 4
AD A063003

DDC FILE COPY AD A063003

(18) (19) LEVEL II (2)

AFAL-TR-78-55-VOL-1
Volume I

AD63004

4

RADIATION HARDENED MICROPROCESSOR
Volume I.

(10) V. V. /Nickel
P. A. /Rosenberg

QUESTRON CORPORATION
SAN DIEGO, CALIFORNIA 92108

DDC RECEIVED JAN 8 1979 F

(11) MAY 1978

(12) 372 p.

TECHNICAL REPORT AFAL-TR-78-55, Volume I
(9) Final Report for Period Oct[illegible]1976 – Dec[illegible]1977

(16) 6096

(15) F33615-77-C-1001

(17) 40

Approved for public release; distribution unlimited.

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

JUB

409 170 79 01 04 003

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Gary D. Gaugler
Project Officer

Stanley E. Wagner, Chief
Microelectronics Branch
Electronic Technology Division

FOR THE COMMANDER

William J. Edwards, Director
Electronic Technology Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAL/DHE *,W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE/56780/13 December 1978 — 75

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFAL-TR-78-55 Volume I	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) RADIATION HARDENED MICROPROCESSOR		5. TYPE OF REPORT & PERIOD COVERED Final Oct 76–Dec 77 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) V. V. Nickel, P. A. Rosenberg		8. CONTRACT OR GRANT NUMBER(s) F33615-77-C-1001
9. PERFORMING ORGANIZATION NAME AND ADDRESS Questron Corporation 2727 Camino Del Rio S., Suite 125 San Diego, California 92108		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS CDRL 003 60964004
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory AFAL/DHE-3 Air Force Systems Command, U.S. Air Force Wright-Patterson AFB, Ohio 45433		12. REPORT DATE May, 1978 13. NUMBER OF PAGES 362
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

8080 Microprocessor
Automatic Test Generation
CMOS/SOS
Emulation
General Processor Unit (GPU)
Low Cost, Functional Testing
LSI
LSI Testing
Microprogramming
Radiation Hardened Microprocessor
Universal Array (UA)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Functional capabilities of a CMOS/SOS LSI device family for emulation usage were assessed. The device family consists of an 8-bit processor bit slice called the general processor unit (GPU), a 1024-bit read only memory (ROM) and a Universal Array ~~(UA)~~ (GUA) device with approximately 300 gates. The devcies were analyzed in detail and deficiencies in their designs are itemized. Recommendations for changes are made to improve the emulation capabilities of the device family. Radiation hardening characteristics of the GPU,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

79 01 04 003

ROM and GUA are assessed in relation to their usage in emulating existing microcomputers and mini-computers. A complete functional specification is provided for the GPU. During this effort, the GPU design was verified utilizing low cost, functional testing techniques developed specially for LSI design validation. The key to low cost, functional testing is the automatic generation of test vectors. Several faults were detected in the GPU, and are itemized in the report. A top level throughput characterization was also performed on the GPU. To the emulation capabilities of the devices, a design of an 8080 microprocessor emulator, utilizing the CMOS/SOS device family, was completed during this effort. Hardware and firmware requirements for the emulator are determined to measure emulation efficiency and a throughput comparison is made between the emulator and the actual 8080.

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
A

Table of Contents Volume I

Introduction 1

1. GPU Functional Characterization 4

1.1 GPU Summary of Characteristics 5

1.2 GPU Architecture 8

1.2.1 Register File 8

1.2.2 Port Buffers 8

1.2.3 Data Type Selectors 10

1.2.4 Arithmetic/Logic Unit 10

1.2.5 Shift Select 12

1.2.6 Boundary, Connect Control 12

1.2.7 Destination Shift Select Control 13

1.2.8 Data Input/Output 13

1.2.9 Address Decoders, P1B and P2B Source Select Control 14

1.3 Control Descriptions 15

1.3.1 Source Select Control 15

1.3.2 Data Type Selectors 16

1.3.3 ALC Control 18

1.3.4 Destination Shift Selector Control 18

1.3.5 Boundary and Connect Control 18

1.3.6 Data Output Enable 22

1.3.7 Load Clock 22

1.4 Data Signal Descriptions 23

1.4.1 Data Input 23

1.4.2 Carry In, Carry Out 23

1.4.3 All Zero Detect Output 23

1.4.4 Multiplexer Shift Bits 24

1.4.5 Data Output 24

1.5 GPU Timing Restrictions 29

1.5.1 P2B Hold Time 29

1.5.2 P1B Hold Times 29

1.6 Cycle Time and Clock Characteristics 31

1.6.1 Load Clock 31

1.6.2 Data Path Delays 32

1.6.3 Set-up and Hold Times 33

2. Emulation Analysis 35

2.1 Design Completeness of the GPU 36

2.1.1 GPU Functional Design Deficiencies 36
2.1.1.1 GPU Pipelining 36
2.1.1.2 Missing Overflow Detection Logic 39
2.1.1.3 Carry-in Select 40
2.1.1.4 Control Function Sharing 42
2.1.1.5 Asynchronous Design 43
2.1.2 Emulation with the GPU 47
2.1.2.1 GPU Off-Chip Delays 48
2.1.2.2 The MQ Register and LSI 50
2.2 ROM Design Critique 51
2.3 GUA Design Critique 52
2.4 RAM Design Critique 53
2.5 Proposed GPU SOS COS/MOS Device Family 55
2.6 Radiation Hardening of CMOS/SOS Circuits 58
2.6.1 TCS077 Multiplier 58
2.6.1.1 Electrical Performance 59
2.6.1.2 Radiation Response 59
2.6.2 TCS072 RAM 60
2.6.2.1 Electrical Performance 60
2.6.2.2 Radiation Response 60
2.6.3 TCS091 Gate Universal Array 61
2.6.3.1 Radiation Response 64
2.6.4 Radiation Hardening CMOS/SOS IC's 64
2.6.4.1 Radiation Hardened Oxide Process 64
2.6.4.2 Radiation Hardened IC Design Rules 65
3. GPU Testing Results and Methodology 67
3.1 Testing Results 67
3.1.1 Summary 68
3.1.2 Fault Isolation and Diagnosis 69
3.1.2.1 Register File Access Problem 70
3.1.2.2 Simultaneous Activation of FETS on P1B Input Bus 81
3.1.2.3 Output Disable on Direct Input 83
3.1.2.4 R,T and S,A and M Race Conditions 86
3.1.2.5 Load Clock, P2B Loop Condition 91

Defense Documentation Center

CAMERON STATION, ALEXANDRIA, VIRGINIA 22314

STAFF BULLETIN

DDC-B | No. 11 | 16 March 1979

OFFICIAL

1. UNITED STATES AIR FORCE ASSISTANCE FUND. The 1979 Air Force Assistance Fund (AFAF) Campaign is now being conducted and will run through 31 Mar 79. The three charitable organizations recognized by the Air Force as eligible recipients for financial assistance from the annual AFAF Campaign are: Air Force Enlisted Men's Widows and Dependents Home; Air Force Aid Society, and the Air Force Village. These organizations are not supported through the Combined Federal Campaign. Col Tom O. Olofson, USAF, is the project officer for the Defense Logistics Agency. Barbara Thurman is the DDC representative for the AFAF. Anyone desiring information regarding the AFAF or wishing to make a tax deductible contribution is invited to call Mrs. Thurman on Ext. 46975. All contributions will be greatly appreciated for this most worthy cause- A COMMITMENT TO CARING. (DLA Bulletin, No. 10, 14 Mar 79)

2. EARLY CLOSING OF FINANCE CASHIER, FRIDAY, 16 MARCH 1979. The cashier in the DASC Accounting and Finance Office will close at 1130 hours, Friday, 16 March 1979, because of office excellence remodeling work. (DLA Bulletin, No. 10, 14 Mar 79)

BY ORDER OF THE ADMINISTRATOR

Rene' S. Lehman

RENE' S. LEHMAN
Director, Office of
Support Services

UNOFFICIAL

CARPOOL. Established carpool desires additional drivers. Bowie, Crofton, MD area (points north and Annapolis) to Cameron Station/AMC Building. Reserved parking between buildings 7 and 8. (DLA-LM, John Giddo, 47932)

AFGE LOCAL #2449. The regular meeting of AFGE Local #2449 will be held 26 Mar 79 in Room 8G285, at 1130. Please plan to attend. Guests are welcome. (Helen Atkinson, Secretary, AFGE 2449)

SOFTBALL PLAYERS. All interested personnel who wish to play for DDC's Softball Team, are invited to attend a pre-season meeting at 9:00 a.m., 21 Mar 79, in the DDC Conference Room. Tryouts will start 27 March after work and the season begins the third week of April (16 April). Gordon Willey, DDC-TII, x46877.

DEPARTURE: Roberta J. Massey, DDC-S, Computer Operator, 16 Mar 79.

NEW EMPLOYEE: Mary Powe, DDC-DFA, File Clerk, GS-1, 19 Mar 79.

CAMERON

SAFETY

BULLETIN NUMBER 385-03-79

COL. DONALD L. BURT, POST CDR.
LTC JAMES R. POACH, DEP CDR.
SGM J. B. QUALLS, POST SGM.
MR. RAY SIMPSON, SAFETY OFFICER

MARCH 1979

CAMERON STATION, ALEXANDRIA, VIRGINIA 22314 SAFETY OFFICE: 274-6507

PUBLISHED MONTHLY BY THE CAMERON STATION SAFETY MANAGEMENT OFFICE IN THE INTEREST OF PROMOTING A SAFE ENVIRONMENT, NOT ONLY IN THE WORKPLACE, BUT ALSO IN THE HOME. ARTICLES FOR PUBLICATION IN THE NEWS BULLETIN ARE WELCOME FROM ALL CAMERON STATION PERSONNEL. UNIT COMMANDERS AND ACTIVITY CHIEFS ARE REQUESTED TO DISPLAY THE CAMERON STATION SAFETY BULLETIN ON BULLETIN BOARDS TO PROVIDE ADEQUATE OPPORTUNITY FOR ALL PERSONNEL TO READ. THE COMPILATION OF MATERIAL IN THIS PUBLICATION IS DESIGNED TO BE INFORMATIONAL ONLY AND DOES NOT CONSTITUTE ENDORSEMENT, APPROVAL OR RECOMMENDATION. THE BULLETIN MAY BE REPRODUCED BY UNITS OR ACTIVITIES.

HOW TO MINIMIZE INJURY IF YOU FALL:

1. Try not to tense your muscles.

2. Let the fleshy parts of your body absorb the impact.

3. Roll or move with the direction of the fall.

IF YOU HAVE AN ACCIDENT...it means that something went wrong--either in the way you did the job or with the equipment you used.

HOW MUCH IS A HAND WORTH?

Just one moment--the twinkling of an eye and it will all be over! A missing finger or perhaps a hand, because the machine guard was not in place! OR, perhaps we feel that, because we have done it that way for many years, that it can't happen to use, always the other person, but not us! That kind of thinking is dangerous. Use all machine guarding, all the time. Lock out machinery or equipment for cleaning, adjustment or repairs. Wear protective equipment where required. Protect that hand!

NATIONAL POISON PREVENTION WEEK---MARCH 18-24, 1979.

AVERAGE STOPPING DISTANCES....

Reaction and braking distances vary with road conditions, even with weather conditions, speed and the motor vehicle, as well as with the driver, himself. Practice good driving habits by knowing what your motor vehicle will do.

A driver who tailgates does not give himself enough time to stop by not having enough distance between his car and the car ahead, to handle emergency situations. Running into the vehicle ahead happens often because many drivers do not allow a safe stopping distance. USE THE "TWO SECOND RULE". BE A SAFE DRIVER!

OPEN UP...

A mind is like an umbrella or a parachute---it only works when it's open!

D R I V I N G ---

S cream about inflation.....Then spend $1,000.00 to fix up your car because you wrecked it by trying to squeeze another 100 miles out of a worn set of tires!

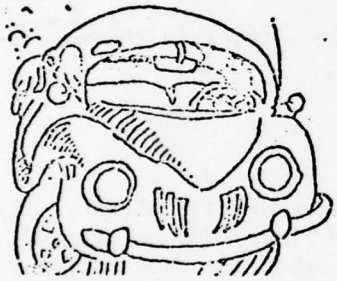
A sk for leave.....Then spend it in the hospital because you didn't feel you needed new brake linings!

F eel you're losing your rights because you have to wear seat belts....then carry a bad facial scar for life because you weren't wearing one!

E ase out into fast moving traffic....then speed through a residential area!

T ake a motorcycle safety course....Then wreck your bike trying to pop a "wheelie"!

Y ell at the other guy for not yielding the right-of-way....Then speed up when someone tries to pass you!



W H Y ? ? ? ? ? ? ? ? ?

Spring is coming.... When our thoughts had better be turning to safety. What are your plans for this spring and summer? Do they include avoiding injury while you are working out of doors?

it is written: "Bless the Lord, O My Soul, And All That Is Within Me, Bless His Holy Name!" PSALM 103:1.

COMING EVENTS. Special Services orders all tickets through the Pentagon Ticket Service. The Pentagon Ticket Service offers a convenient way for employees to order tickets for most Washington area leisure activities and events: concerts and plays at Kennedy Center, sports at Capital Centre, most of the local dinner theaters, discounts for theme park admissions, special attractions like the circus, Orioles baseball, discount Global Menu Club memberships--and NOW tickets and bus passes for travel throughout the nationwide Greyhound system. PTS obtains discounts for DOD personnel whenever it can, but often the major advantage it offers is a way around the long lines at entrance box offices. A 60¢ service charge is added to ticket prices to partly offset the cost of operation but NO SURCHARGE IS MADE ON BUS TICKETS.

KENNEDY CENTER.

EISENHOWER THEATRE.	Now thru 24 Mar	A BEDROOM FARCE
	28 Mar-1 April	ST. MARK'S GOSPEL (Tickets on sale Box Office only)
	4 Apr-12 May	THE GIN GAME (Tickets go on sale 18 Mar)
OPERA HOUSE.	Now thru 24 Mar	CARMELINA
	27 Mar-15 Apr	AMERICAN BALLET THEATRE

NATIONAL THEATRE.	Now thru 17 Mar	A CHORUS LINE
	21 Mar-26 May	THE WIZ

FORD'S THEATRE. 16 Mar-29 Apr FESTIVAL

ARENA STAGE. 30 Mar-6 May DON JUAN

CAPITAL CENTRE.	23 Mar 8 PM	KENNY ROGERS/DOTTIE WEST	$8.80/7.70
	28 Mar 8 PM	NAZARETH	$7.70/6.60
	21 Apr 2 & 8 PM	GOSPEL SHOW	$8.50/7.50
	28 Apr 2 & 8 PM	COUNTRY MUSIC CONCERT	$10/8

(DASC-I, Mrs. Turley, HOURS: 0900-1500)

COMMODORES

NAVY BAND'S "COMMODORES"

TO APPEAR 21 MARCH 1979

1130 - 1200 / 1215 - 1245

The Commodores, the United States Navy Band's jazz ensemble in Washington, D.C., will appear at THE AUDITORIUM BUILDING 14 CAMERON STATION.

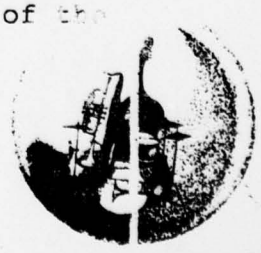
Combining jazz with popular and rock music, the Commodores present an entertainment package the whole family can enjoy.

The Commodores were formed in January 1969 and designated the official dance band of the United States Navy. It is composed of the top jazz and "Big Band" players in the Navy today, most of them career men with at least eight years of service. Included in this flexible organization are a vocalist and, of course, the various instrumentalists all of whom are bonafide soloists.

The music of the Commodores sets a high and festive mood, and the obvious personal pride shown by the members in their performances assures the audience the finest in popular jazz.

Since the group started performing, the Commodores have received high praise for their performances throughout the world and remain in constant demand for concert appearances.

In its almost 10 years of existence the band has traveled to Europe and South America. It is the only service band to play for the Newport Jazz Festival, and is a member of the National Association of Jazz Educators.



What you

should know

about

CREDIT UNION BANK ON US

- DIRECT DEPOSIT TO YOUR SHARE DRAFT ACCOUNT
- BILL PAYER SERVICES

FOR FURTHER INFORMATION CONTACT:

CAMERON STATION FEDERAL CREDIT UNION

BLDG. #2

274-7641

ANN. NO.	AREA OF CONSIDERATION	POSITION, TITLE, GRADE & SERIES	CLOSING DATE

PART I: Position Vacancies in Activities Serviced by the Civilian Personnel Division, Defense Logistics Agency, Administrative Support Center, Cameron Station, Alex., VA

52	(4)	PROGRAM ANALYST, GS-345-12, DLA-S	19 Mar
53	(2/4)	COMPUTER PROGRAMMER, GS-334-12, DASC-D	16 Mar
58	(4)	SUPPLY MANAGEMENT REP., GS-2003-12, (Pot. to GS-13), DCASO	23 Mar
59	(4)	SUPPLY MANAGEMENT REP., GS-2003-12, (Pot. to GS-13), DCSAO	23 Mar
61	(11)	MICROPHOTOGRAPHER, WI-4430-6, DDC	27 Mar
62	(10)	MANAGEMENT AN LYST, GS-343-5, (Pot. to GS-11), DDC	15 Mar
63	(10)	OFFICE ADMIN. ASSISTANT (TYPING), GS-301-5, (Pot. to GS-6), DLA-S	15 Mar
64	(10)	VOUCHER EXAMINER (TYPING), GS-540-3, (Pot. to GS-5), DASC-F	15 Mar
65	(10)	ADMIN. SERVICES ASSISTANT (TYPING), GS-301-4 (Pot. to GS-5), DFSC	16 Mar
66	(4)	PROCUREMENT AGENT, GS-1102-12, DFSC	30 Mar
67	(4)	SUPERVISORY INSPECTOR GENERAL REPRESENTATIVE, GS-301-14 (NTE 1 year, May be extended or may become permanent), DLA-I	26 Mar
68	(3)	SECRETARY (TYPING), GS-318-5, NPO	2 Apr
69	(11)*	COMPUTER AID, GS-335-3/4 (Pot. to GS-5), DASC-D	30 Mar
70	(2)	INVENTORY MANAGEMENT SPECIALIST, GS-2010-10, DFSC	2 Apr

*and reinstatement eligibles

KEY TO AREA OF CONSIDERATION:

(1) DAS (In Wash Metro Area)
(2) DLA-Wide
(3) DoD Wide (In Wash Metro Area)
(4) DoD Wide
(5) Federal Agencies
(6) All Sources (In Wash Metro Area)
(7) All Sources
(8) Upward Mobility
(9) DASC-Z activities and all organizations located at Cameron Stat.
(10) DASC-Z serviced activities
(11) Federal Agencies in Wash Metro Area
(12) Dept of Navy (In Wash Metro Area)

PART II: Position Vacancies in Defense Logistics Agency Field Activities:

DCASR LOS ANG	QUALITY ASSURANCE SPECIALIST (MECH.), GS-1910-9	12 Apr
DCASR LOS ANG	QUALITY ASSURANCE SPECIALIST (ELECTRONIC), GS-1910-9	29 May
DCASR LOS ANG	CONTRACT ADMINISTRATOR, GS-1102-12	6 Sep
DCASR LOS ANG	CONTRACT PRICE ANALYST, GS-1102-11	12 Apr
DGSC RICHMOND	MANAGEMENT ANALYST, GS-343-13, D/S Chicago, Illinois	27 Mar
DCASR BOSTON	ELECTRONICS ENGINEER, GS-855-11/12, D/S Owego, New York	28 Mar

3.1.2.6 Missing Overflow Detection Logic 93
3.1.2.7 Decoder Sneak Pulse 95
3.2 Throughput Analysis 99
3.2.1 Summary 99
3.2.1.1 Register to Register Operation 100
3.2.1.2 Register to Data Out, Carry Out and All Zero Out 103
3.3 Testing Methodology 108
3.3.1 Summary 108
3.3.1.1 GPU Summary of Characteristics 111
3.3.1.2 Software System Overview 114
3.3.2 The GPU Microprogram Assembler 116
3.3.3 GPU Simulator 120
3.3.4 GPU Exerciser Software 122
3.3.5 Testing Approach 125
3.3.5.1 Overview 125
3.3.5.2 Degree of Test Complexity 126
3.3.5.3 Test Structure 129
4. Q80 Design Description 134
4.1 Functional Description 135
4.1.1 Architecture 135
4.1.1.1 Register File, Arithmetic Logic Unit 137
4.1.1.2 Interface Unit 140
4.1.1.3 Microprogram Control Unit 140
4.1.2 Processor Cycle 146
4.1.3 Interface Description 146
4.1.4 Interface Timing Protocol 153
4.1.4.1 Memory, I/O Read Ready/Wait Handshake 153
4.1.4.2 Memory, I/O Write Ready/Wait Handshake 153
4.1.4.3 Instruction Fetch Transaction 162
4.1.4.4 Memory Read and Write Transactions 162
4.1.4.7 Hold Acknowledge Protocol 165
4.1.5 Instruction Set 167
4.1.5.1 Summary 167
4.1.5.2 Instruction and Data Formats 168
4.1.5.3 Addressing Modes 169
4.1.5.4 Symbols and Abbreviations 170

4.1.5.5 Instruction Descriptions 173
4.1.5.5.1 Data Transfer Group 173
4.1.5.5.2 Arithmetic Group 178
4.1.5.5.3 Logical Group 185
4.1.5.5.4 Branch Group 191
4.1.5.5.5 Stack, I/O, and Machine Control Group 196
4.1.6 Flags 201
4.1.7 Interrupts 203
4.1.8 State Diagram 205
4.2 Q80 Breadboard Logic Design Description 215
4.2.1 Architecture 215
4.2.2 LSI Partitioning 218
4.2.3 Condition Flag and Shift Control GUA 221
4.2.4 Decode GUA 225
4.2.5 Interface GUA 228
4.2.6 Q80 Microsequencer 230
4.3 Firmware 233
4.3.1 Overview 233
4.3.2 The Q-80 Instruction Cycle 233
4.3.3 Partitioning of the Microaddress Space 234
4.4 Packaging 358

List of Illustrations Volume I

1.1-1 GPU Data Paths 7
1.2-1 Register File Bit Slice 9
1.3-1 Use of MX-bits as Shift Carriers 20
1.3-2 Effect of Load Clock 23
1.4-1 GPU Dual In-Line Package 27
1.4-2 Bonding Diagram and Pin-out for TCS074 28
1.5-1 P2B Hold Times 30
1.5-2 P1B Hold Times 30
1.6-1 GPU Setup and Hold Times 33
1.6-1 GPU Setup and Hold Times 33
2.1-1 Example of Pipeline Operation in Control Processor 37
2.1-2 Pipelining in Floating Point Processor 38
2.1-3 Sharing of Data Paths for Fixed, Floating Point 41
2.1-4 Register File, P1B and P2B Master Slave Relationship 45
2.1-5 Separating Master-Slave and Multiple Source Functions 46
2.1-6 Carry-Out Propagation Delay for 32-Bit Processor 49
2.5-1 Bus Tranceiver 55
3.1-1 Register File Failure Symptom 71
3.1-2 Register File Dual Access Bit Slice Schematic 76
3.1-3 Failure Curve for Part AFAL No. 10 (7716F) 77
3.1-4 Failure Curve for Part AFAL No. 12 (7716E) 78
3.1-5 Failure Curve for Part AFAL No. 2 (7716D) 79
3.1-6 Failure Curve for Part AFAL No. 23 (7718A) 80
3.1-7 Karnaugh Map for P1B Input Bus FET Activation 81
3.1-8 Simultaneous Activation of Drivers 82
3.1-9 Example of Pipeline Operation in Control Processor 84
3.1-10 Pipelining in Floating Point Processor 85
3.1-11 P2B Race Condition Sequence 88
3.1-12 P1B Race Condition Sequence 89
3.1-13 DI into P1B Race Condition 90
3.1-14 Load Clock, P2B Loop Condition 92
3.1-15 Port 1 Direct in Decode Sneak Pulse 96
3.1-16 Command Sequence that Triggers Decoder Sneak Pulse 97
3.1-17 P1B Source State Function 98

List of Illustrations Volume I (Cont.)

3.2-1	Register to Register Delay Time Characteristics	102
3.3-1	MDS-800 Computer System Configuration	110
3.3-2	GPU Architecture	113
3.3-3	System Overview	115
3.3-4	Example of Higher Level Construct Usage	119
3.3-5	Sample of GPU Simulator Listing	121
3.3-6	Sample of GPU Exerciser Listing	124
3.3-7	Degree of Testing	128
3.3-8	Conceptualization of Test Vector Design Approach	127
3.3-9	Testware Organization	132
3.3-10	GPU Testing Strategy	133
4.1-1	Q-80 8080 Emulator	136
4.1-2	Q-80 8080 Emulator Data Paths Register File	139
4.1-3	Microprogram Control Unit	142
4.1-4	Effects of the Clock in the Q-80	147
4.1-5	8080 Emulator Interfaces	152
4.1-6	Memory, I/O Read – No Handshake	154
4.1-7	Memory, I/O Read Ready/Wait Hankshake	155
4.1-8	Memory, I/O Write-No Handshake	156
4.1-9	Memory, I/O Write READY/Wait Handshake	157
4.1-10	Instruction Fetch	160
4.1-11	Memory Read/Write	161
4.1-12	I/O Read/Write	163
4.1-13	Interrupt Protocol	164
4.1-14	Hold Protocol	166
4.1-15	Data Format	168
4.1-16	Instruction Formats	168
4.1-17	PSW Format	201
4.1-18	Q-80 State Diagram	214
4.2-1	Q-80 Breadboard/Firmware Development Station	216
4.2-2	Q-80 Emulator Block Diagram	219
4.2-3	Status Flag Shift Control GUA	220
4.2-4	PWS Format in the Q-80	222
4.2-5	Decode GUA	226
4.2-6	Interface GUA Block Diagram	229
4.2-7	Q-80 Microsequencer	231
4.3-1	Q-80 Instruction Cycle	235

List of Tables Volume I

1.1-1	GPU Summary of Features	6
1.3-1	Source Select Control	15
1.3-2	Data Type Select Control	17
1.3-3	ALC Control	18
1.3-4	Destination Shift Select Control	19
1.3-5	Boundary and Connect Control	21
1.4-1	GPU Control and Data Signals	25
1.4-2	GPU Pin Signal List	26
1.6-1	GPU Timing Data	31
1.6-2	GPU Data Path Time Delays	32
2.1-1	Source Select Control	42
2.5-1	Proposed GPU SOS COS/MOS Device Family	57
2.6-1	Chip Speed vs. Channel Length for TCS077	59
2.6-2	Device Sizes in TCS091 GUA	62
2.6-3	Optimum Ratio of Wn/Wp Register File Failure Data	62
3.1-1	Register File Failure Data	72
3.1-2	Register Access Failure Characteristics (Static Test Results)	73
3.2-1	Register to Register Delay Time	101
3.2-2	Register to Data Out Delays	104
3.2-3	Register to Carry Out Delays	105
3.2-4	Register to Carry Out Delay for Logical Operations	106
3.2-5	Register to All Zero Out Delay	107
3.3-1	GPU Summary of Characteristics	112
3.3-2	GPU Microprogram Assembler Language	117
3.3-3	Iterative Repetitions	120
3.3-4	Xrcise Subcommands	123
3.3-5	Testware Modules	130
4.1-1	Q80 Summary of Characteristics	143
4.1-2	Symbols and Abbreviations	170
4.1-3	Setting of Flags	202
4.1-4	Next State at End of Execution for HLT-instructions	209
4.1-5	State Transition at End of Instruction Execution (except HLT)	209
4.2-1	Control of the CY Input	222
4.2-2	Channel Assignments for the A-multiplexer	222
4.2-3	Control of the CI input to the GPU	223
4.2-4	Program Counter Load Control	223
4.2-5	End Condition Generation Control	224
4.2-6	I/O Decode Operations	227
4.2-7	Q80 Microaddress Configurations	232

Introduction

This Final Report was prepared by Questron Corporation, San Diego, California, under USAF Contract No. F33615-77-C-1001 and P00001. This work was performed for the Air Force Avionics Laboratory, Wright-Patterson Air Force Base, Dayton, Ohio, under the direction of Capt. T. Margraff and Mr. R. Conklin in the Air Force Program Office. This Final Report describes work performed from 1 October 1977 through 31 December 1977 by the Computer Sciences Department of Questron Corporation in El Segundo, California.

The work was directed by V. V. Nickel, Manager of Computer Sciences Department. The principal engineers contributing to this effort were P. A. Rosenberg, Senior Engineer and G. L. McCollum, Senior Engineer. The manuscript was released by the authors in January 1978.

The purpose of this effort was to evaluate the computer emulation effectiveness of an RCA microprocessor bit slice SOS COS/MOS device family. The device family consists of the following members:

- General Processor Unit (GPU) TCS-074
- Gate Universal Array (GUA) TCS-091
- Read Only Memory (ROM) TCS-075
- Random Access Memory (RAM) TCS-072

The heart of the microprocessor bit slice device family is the GPU. For this reason it is described in detail in Section 1.

The approach taken in the emulation analysis was two pronged. First, the effectiveness of the device family in emulating various computers from simple 4-bit microprocessors to high power 32-bit floating point processors was analyzed. Several deficiencies in the functional design of the GPU were found during this analysis and are now being corrected in a new design. More importantly, deficiencies in the breadth of the device family were found to exist and recommendations for additional part types for the family are proposed (refer to Section 2.5) to ameliorate the deficiencies. The emulation capabilities analysis results are presented in Section 2.

Second, a parallel effort was undertaken to define and then design, to the gate level, a particular computer for emulation utilizing the device family. This effort provided a real test of the true emulation capabilities of the device family. As a result of this effort, deficiencies in the devices were found—they are also reported on in Section 2. The computer selected for emulation was the 8080. This microprocessor is representative of the higher performance microprocessors on the market and, as such, was an excellent emulation analysis vehicle. The details of the breadboard design of the 8080 Emulator are presented in Section 4.

Since the GPU device itself was brand new, being quite literally "hot from the furnaces", Questron undertook a functional testing effort of the GPU to verify and validate its design. During the process, Questron conceived, developed and proved the concept of low cost functional testing for LSI design verification. This concept is explained in detail in Section 3. During this effort, Questron detected, isolated and diagnosed a total of seven design flaws in the GPU (refer to Section 3). The GPU is currently being reworked to correct the flaws.

To summarize, the major accomplishments of this program are as follows:

- A detailed functional specification of the GPU device (Section 1) that will enable potential users of the device to understand its features and also its nuances. Such a specification did not previously exist because of the newness of the device.

- An analysis of the emulation effectiveness of the RCA SOS COS/MOS device family (Section 2). This analysis has resulted in design modification being made to the GPU which will significantly enhance its capabilities. In addition, recommendations are made for additional devices for the parts family to upgrade the effectiveness and viability of the family in emulating computers.

- Development and proof of the concept of the low-cost functional testing technique for LSI design verification and validation (Section 3). This testing technique resulted in

the detection, isolation and diagnosis of seven functional design, logic design, electrical design, mask layout and process related flaws. These flaws are currently being ameliorated in a reworked GPU design.

- A detailed design of a breadboard 8080 Emulator was completed (Section 4). This design is in sufficient detail so that hardware can be procured and a working 8080 Emulator breadboard fabricated.

1. GPU Functional Characterization

This section presents a detailed functional description of the GPU. It should be noted that the GPU, as described in this section, is functionally quite different than described in Section 3. The reason is that this section describes the GPU as modified by the results of the GPU emulation analysis effort (reported in Section 2) and the GPU testing effort (reported in Section 3). The reworked GPU is functionally superior to the present GPU and should be available in the first quarter of 1978.

1.1 GPU Summary of Characteristics

The General Processor Unit (GPU) is an 8-bit central processor unit bit slice intended for use in CPU's, peripheral controllers, programmable microprocessors and dedicated controllers. It is a high speed, CMOS/SOS device which is cascadable to allow efficient emulation of any computer with word lengths from 8 bits on up.

As illustrated Figure 1.1-1 the GPU is an 8-bit processor bit slice consisting of; a 16 word by 8 bit dual access register file, two port buffers (P1B, P2B), extensive data type selection multiplexing, arithmetic/logic circuit (ALC) and powerful shifting capability. It has separate 8-bit data input and data output paths. Data can be input directly to the register file, P1B or P2B. Data can be output from P1B, P2B or from the ALC. Output drivers are designed to drive 30pf with a rise and fall time of 30ns at 10 volts. The data paths are designed to facilitate the implementation of complex algorithms. For example, two bit multiply data paths consisting of a right shift of one into the ALC and right shift of one or two into the register file are implemented on the chip. Although the device is not yet fully characterized, preliminary tests indicate it is fast – full cycle (accessing of two operands from the file, operating on them through the ALC and storing the result back to the register file) operations of up to 10 MHz have been observed. Table 1.1-1 summarizes the significant features of the GPU.

The GPU is implemented as a 48-pin device consisting of 2943 transistors of various sizes, in a 43.2×10^3 mil^2 area; this is an average of 14.7 mil^2 per transistor. The GPU is fabricated using the silicon-gate CMOS/SOS process employing only one epitaxial deposition. A single ion implant is used to dope the silicon islands. The channel oxide is formed by the conventional wet-HCL thermally grown SiO_2 method.

Table 1.1-1 GPU Summary of Features

- 8-bit processor bit slice.
- 16 dual access 8-bit general purpose registers.
- Single clock operation – ability to access two registers, operate on them and store away result in register file, all in one clock cycle.
- 8-bit parallel arithmetic logic circuit (ALC) with carry look ahead, all-zero detection and overflow indication.
- Data path and control provided for multiple step algorithms such as two-bit multiply, division and floating point.
- Expandable – can concatenate any number of GPU's for larger word size machines.
- Microprogram versatility – can independently control selection of sources, ALC operation and destination of data.
- CMOS/SOS – chip size 201 x 215. Low power, high speed, high noise immunity and radiation hard.
- Static operation. Can be run at any speed from single step to 10MHz.
- Single DC power source 4 Volts to 15 Volts.
- Separate data input and data output paths – output buffers are tri-state.
- Pipeline operation – can simultaneously output ALC results while directly loading into the register file from the data input.
- 8-bit on chip, carry lookahead for high speed arithmetic operations.
- Quotient and remainder adjustment hardware.

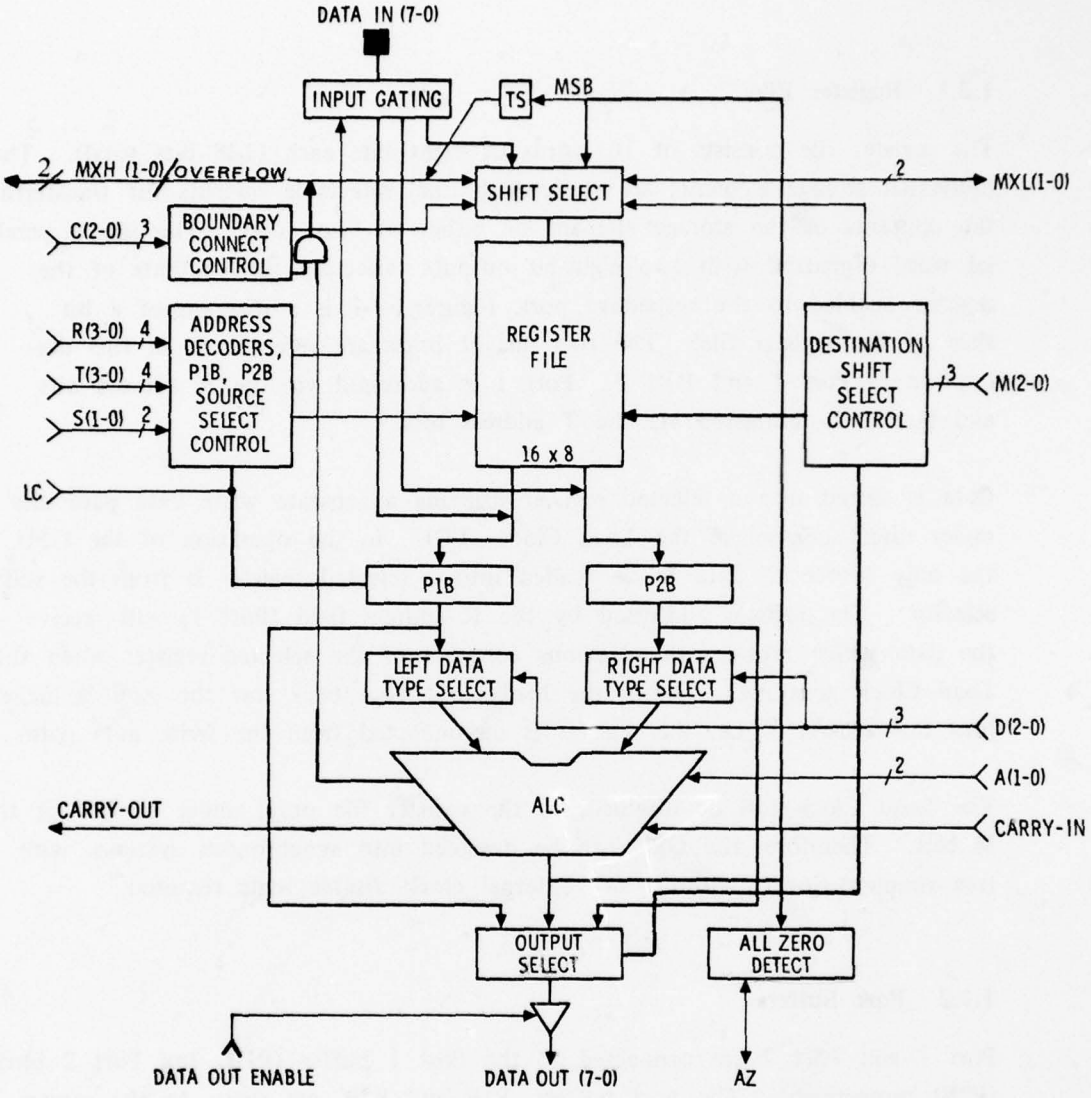


Figure 1.1-1 GPU Data Paths

1.2 GPU Architecture

For ease in understanding and to facilitate explaination, the GPU is decomposed into nine subfunctions and each is explained as a separate entity in the following paragraphs.

1.2.1 Register File

The register file consists of 16 words of eight bits each (128 bits total). The individual storage elements are static with dual selectable outputs for transferring the contents of the storage element on either of two ports. The file is parallel word organized with two eight-bit outputs reflecting the contents of the register enabled to the respective port. Figure 1.2-1 is a diagram of a bit slice of the register file. The two output ports are referred to in this description as Port 1 and Port 2. Port 1 is addressed via the R address bits and Port 2 is addressed via the T address bits.

Data is stored into a selected register utilizing a separate write data path and under direct control of the Load Clock (LC). In the operation of the GPU, the only source of data to be loaded into a selected register is from the shift selector. The register addressed by the R address field (Port 1) will receive the data which replaces the previous contents of the selected register when the Load Clock goes high. When the load clock goes back low the data is locked into the register – i.e. the register is disconnected from the write data path.

The Load Clock can be disabled, to the register file only, under control of the M bits. Therefore, the GPU can be designed into synchronous systems, with free running clocks, without any external clock disable logic required.

1.2.2 Port Buffers

Port 1 and Port 2 are connected to the Port 1 buffer (P1B) and Port 2 buffer (P2B) respectively. The port buffers, P1B and P2B, are slaves to the master register file. When LC is low they follow the contents of the addressed register in the register file (except when explicitly disabled by control decoding). When LC is high the register file is electrically disconnected from the port buffers, thus the buffers retain their value while information is being written

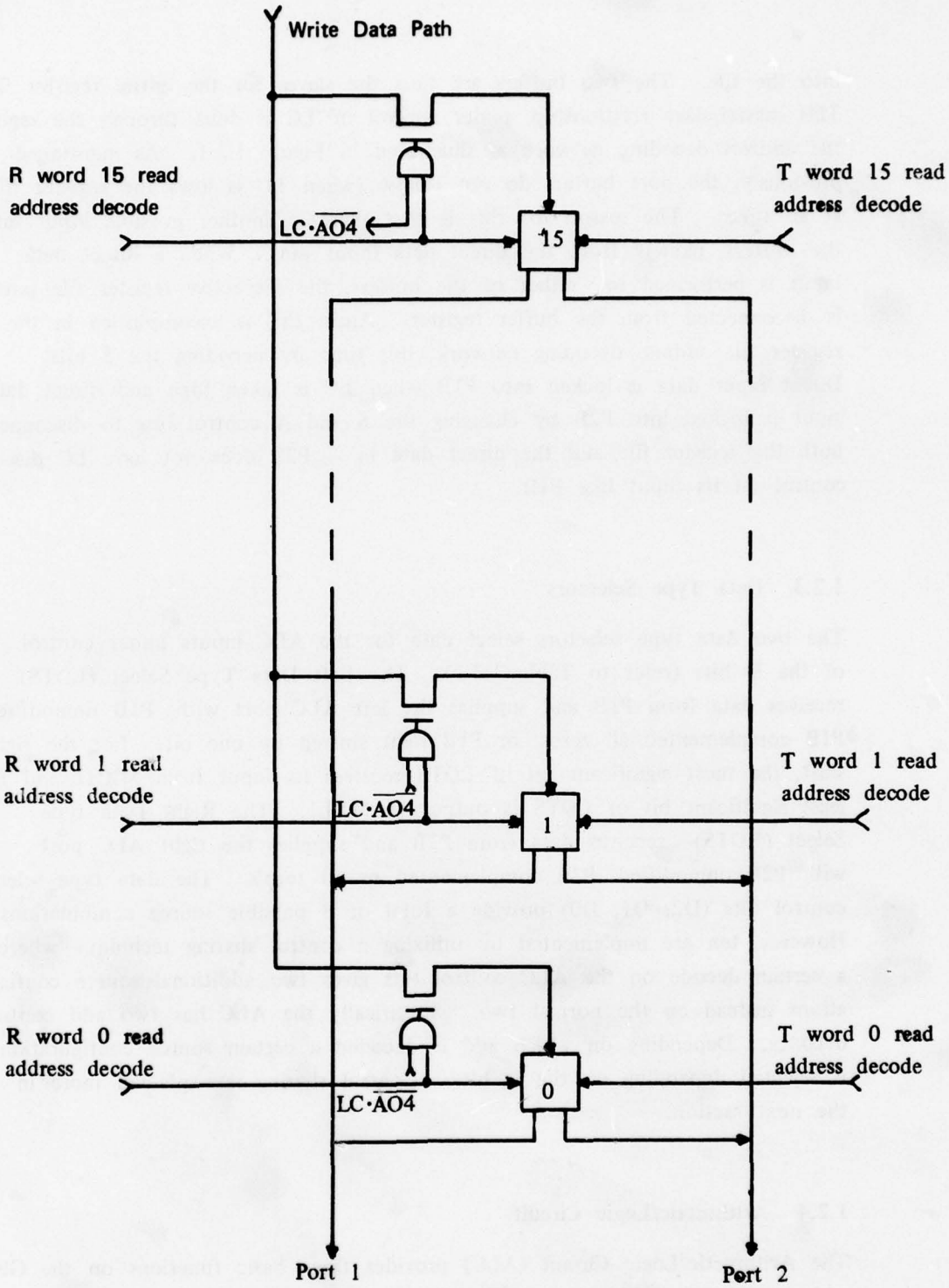


Figure 1.2-1 Register File Bit Slice

into the file. The two buffers are thus the slaves for the entire register file. This master/slave relationship, under control of LC is done through the register file address decoding network as illustrated in Figure 1.2-1. As mentioned previously, the port buffers do not follow (when LC is low) the register file at all times. The reason for this is that there is another possible input into the buffers, namely from the direct data input pins. When a direct data input is performed to either of the buffers, the respective register file port is disconnected from the buffer register. Again this is accomplished in the register file address decoding network; this time by decoding the S bits. Direct input data is locked into P1B when LC is taken high and direct data input is locked into P2B by changing the S and A control bits to disconnect both the register file and the direct data in – P2B does not have LC disable control on its input like P1B.

1.2.3 Data Type Selectors

The two data type selectors select data for the ALC inputs under control of the D bits (refer to Table 1.3-2). The Left Data Type Select (LDTS) receives data from P1B and supplies the left ALC port with; P1B unmodified, P1B complemented, all zeros, or P1B right shifted by one bit. For the right shift, the most significant bit of LDTS receives its input from MXH1 and the least significant bit of LDTS is output to MXL1. The Right Data type Select (RDTS) receives data from P2B and supplies the right ALC port with P2B unmodified, P2B complemented or all zero's. The data type selectors control bits (D2, D1, D0) provide a total of 8 possible source combinations. However, ten are implemented by utilizing a control sharing technique where a certain decode on the ALC control bits gives two additional source configurations instead on the normal two. Specifically the ALC has two add control decodes. Depending on which add is decoded a certain source configuration is selected depending on the D bits. Control sharing is explained more in the next section.

1.2.4 Arithmetic/Logic Circuit

The Arithmetic/Logic Circuit (ALC) provides three basic functions on the GPU: add, logical OR and logical AND. There is one carry-input to the least significant bit, and one carry-output from the most significant bit. A group look-ahead carry circuit is incorporated into the ALC. Each GPU in a con-

catenated set simultaneously establishes its own internal group propagate (GP) and group generate (CG) status.

$$GP = P_0\ P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7$$

$$GG = C_i\ P_{(0\text{-}7)} + G_{(0)}\ P_{(1\text{-}7)} + G_{(1)}\ P_{(2\text{-}7)} + \cdots + G_{(6)}\ P_{(7)} + G_{(7)}$$

The group propagate term is unique to a group independent of the carry-in and mutually exclusive with the generate terms. Therefore, if GP is true on a particular GPU, the carry-in to that particular GPU can be gated immediately to the carry out. If GP is false, the GPU can output a Group Generate One (GG1) or Group Generate Zero (GG0) as required, independent of the carry-in.

$$GG0 = (\overline{GG})\ (\overline{P})$$

$$GG1 = (GG)\ (\overline{P})$$

An overflow indication is provided on the GPU for detecting boundary conditions for addition and subtraction. Overflow is defined simply as a change in the sign bit when performing addition or subtraction; for example if the sign bit goes to a one state (negative) during the addition of two positive numbers, overflow has occurred. Overflow is easily detected on the GPU by taking the exclusive OR of the carry into and the carry out of the most significant bit:

$$\text{Overflow} \equiv C_7 \oplus C_6$$

The overflow signal output is timeshared with shift data on the MXH1 pin under control of the C bits (refer to next section).

Detection of an all-zero output (AZ0) of the ALC is also provided. The GPU generates its own all-zero status. A group level propagate/generate all zero detect circuit provides a fast group status with few delays. A not

all-zero status, in the GPU, is represented by pulling the AZ0 down to zero from its high impedance state. In a system of more than one GPU, the AZ0 outputs are tied all together to form a wire-or. An all -zero group status is represented by a logical ONE, on the bussed AZ0 and not-all-zeros by a logical zero. An external pull-up resistor is required for the AZ0 output.

It should be noted that for logical operations the carry-out is set as follows:

AND CO ≡ 1

OR CO ≡ 0

1.2.5 Shift Select

The Shift Select provides powerful left-right shifting capability on the output of the ALC before it is stored back into the register file. The Shift Select is capable of shifting the ALC output one bit position right, two bit positions right, one bit position left or straight through (no shift). The destination of the Shift Select data is always the register specified by the Port 1 address (R address). Direct data input to the register file from the data input pins also goes through the Shift Select (un-shifted). Again the data is written into the register specified by the Port 1 address. Shifting is controlled by the M-bits directly, to determine the shift and by the Boundary Connect Control (C bits) indirectly, to determine shift input and output connections (refer to Section 1.2.6). The M bits are simply decoded to select the shift input to the register file.

1.2.6 Boundary, Connect Control

The Boundary Connect Control determines what is input and output on the MXH1, MXH0 most significant bi-directional input /output shift bits and MXL1, MXL0 least significant shift bits – MXL0 is bidirectional, MXL1 is a tri-state output only. The Boundary Connect Control state is determined by the C bits (C2, C1, C0). The C bits provide three general classes of states for the four MX shift pins and the Shift Selector. The first class of states configures the GPU for normal inter-circuit shift operations in a multiple GPU machine. The second class of states causes the overflow status indicator to be output on MXH1 while zero or one is shifted into the Shift Selector (if a shift is commanded). The third class of states causes MSB

extension for right shifts and special outputs to be connected to the MXH bits for left shifts. For example, TS and P2B bit 7 can be output on MXH1. TS is also loaded, under certain decodes of the (Bits, from the most significant output bit os ALC. This is useful for saving sign bit information required for remainder and quotient correction in division.

1.2.7 Destination Shift Select Control

The Destination Shift Select Control is configured by the M bits (M2, M1, M0) to control the Shift Select. In addition, the Destination Shift Select Control can disable the Load Clock to the register file and select what is to be output on the data output pins.

1.2.8 Data Input/Output

There are eight Data In lines to input external data to the GPU. This data input can be directed to one of three destinations. The external input data is available to both port buffer registers (P1B or P2B) or the register file as specified by the Source Select Control (the S bits). Data input via Port 2 *is always stored in* P2B, for input to the adder. Data input via Port 1 can either be stored into P1B for input to the adder or stored direct into the register file by specifying so via the Destination Select Control and issueing a Load Clock. When data are stored direct into the file, the previous data in P1B are not changed.

There are eight Data Out lines to output data external to the GPU. Multiplexed into the Data Out lines are three sources of output data; the ALC output, P1B, and P2B. The Destination Shift Select Control specifies the output. The ALC is the default source; i.e. the ALC is selected at all times except when P1B or P2B are explicitly selected. The Data Out pins are tri-state, under control of the Data Output Enable (DOE) signal. This signal must be active to output data, otherwise the data output pins remain in the high impedance state.

1.2.9 Address Decoders, P1B and P2B Source Select Control

This block of logic decodes the R (Port 1) and T (Port 2) address input fields and provides two separate one of sixteen register enable decoders for the two ports – the R and T fields are four bits each. The two decoders enable one of sixteen registers to be connected to P1B and P2B; the R decoder enables data to P1B and the T decoder enables data to P2B. Either of the two decoders can be disabled when a direct input to P1B or P2B (specified by the P1B, P2B Source Select Control - S bits) is being performed - thus electrically disconnecting the register file from the port buffer register while the Data Input pins are connected. Both decoders are disabled when LC goes high, thus preserving data in P1B and P2B when writting into the register file.

The R decoder also determines the register to be writen into with the shifted or unshifted result of the ALC (specified by the M bits) or the direct data input (also specified by the M bits). Under source select control, the R+1 register is accessed when the original R address is even.

1.3 Control Descriptions

The following sections define the functions of all GPU control input fields. All control input signals are active high as illustrated in the following tables.

1.3.1 Source Select Control

There are two source select control inputs, S1 and S0, called the S-bits. The S-bits control the selection of data sources into P1B and P2B. The following table details the S-bit control decoding.

Table 1.3-1 Source Select Control

(Please note timing restrictions presented in Section 1.5)

Reference	Inputs S1 S0	FUNCTION P1B Source	FUNCTION P2B Source	Modifier	Comment
SS0	0 0	(R)	(T)	$\overline{ADI}$	$\overline{ADI}$ is A_1A_0 = 00, 10 or 11; ADD, AND, OR.
		(R)	(P2B)	ADI	ADI is A_1A_0 = 01; ADD
SS1	0 1	DI	(T)	$\overline{ADI}$	If direct input to register file desired, LC must be issued and M bits set to zero (AO0), Otherwise P1B ← DI.
		DI	(P2B)	ADI	
SS2	1 0	(R)	DI	$\overline{ADI}$	Data locked into P1B as long as load clock high for all control combinations.
		(R)	(T)	ADI	
SS3	1 1	(R+1)	DI	$\overline{ADI}$	Original R address must be even, otherwise no change in R address input will occur.
		(R+1)	(T)	ADI	

Notice that there are eight possible control decodes for the S-bits. This is achieved by using the ALC control input as a modifier, thus achieving an additional control configuration for each S-bit decode state. The only restriction is that for all AD1 source select control configurations, the ALC is in the add state; that is, logical operations cannot be performed. P1B receives data from port 1 of the register file (the contents of register address by the R field, signified by (R)) and from the data input pins DI. If SS3 is decoded and the R address is even, R+1 will be accessed. If R is originally odd then the same address, namely R, will be accessed. P2B receives data from Port 2 of the register file (that is, the contents of the register addressed by the T field) and from DI. Data can be retained in P2B with the control combinations SS0 · AD1 and SS1 · AD1. With these control settings P2B will no longer follow (or be a slave of) the register file. Please note that there is a timing restriction on the S-bits (refer to Section 1.5). The S-bits must be changed before the R and T address fields to avoid race conditions. If a direct input to the register file is required SS1 must be set along with AO0.

1.3.2 Data Type Selectors

There are three Data Type Select control bits (D2, D1 and D0) called the D-bits which control the Left and Right Type Selectors. The following table details the D-bit control decoding.

Table 1.3-2 Data Type Selector Control

Reference	Inputs D2	D1	D0	FUNCTION P1B, Left ALC In	P2B, Right ALC In	Modifier	Comment
DS0	0	0	0	Zero	False		$\overline{AD1}$ is A1 A0 = 00, 10, 11, add, And, Or AD1 is A1 A0 = 01; Add
DS1	0	0	1	True	False		
DS2	0	1	0	P1B/2	False	AD1	(1) Right Shift P1B by one, MXH1 gets shifted in and lease significant bit is shifted out through MXL1 independent of M and C control states.
DS2	0	1	0	False	False	$\overline{AD1}$	
DS3	0	1	1	False	Zero		
DS4	1	0	0	Zero	True		
DS5	1	0	1	True	True		
DS6	1	1	0	P1B/2	True	AD1	See note (1) above.
DS6	1	1	0	False	True	$\overline{AD1}$	
DS7	1	1	1	True	Zero		

Ten possible data type combinations are implemented by modifying two of the eight possible D-bit combinations with the ALC control bits, thus obtaining two additional states. These two additional states are used to provide P1B/2 (right shift of one) to the left port of the ALC while either adding or subtracting P2B (no logical operations can be performed on P1B/2). The shift into the most significant bit is taken from MXH1 and the least significant bit of P2B is shifted out through MXL0.

The other D-bit combinations select P1B, P1B complemented, P2B, P2B complemented, and zero for input to the ALC. The significant capability that the Data Type Selector provides is to subtract from either port; i.e. P1B–P2B or P2B–P1B.

1.3.3 ALC Control

Two ALC control bits, A1 and A0, called the A-bits, configure the ALC to perform arithmetic and logical operations. The following table details ALC operation decodes.

Table 1.3-3 ALC Control

Reference	Inputs A1	A0	Function	Modifier	Comments
AD0	0	0	ADD		Left + Right + CI
AD1	0	1	ADD		Left + Right + CI
AD2	1	0	AND		Left AND Right:carry out forced to "1".
AD3	1	1	OR		Left OR Right: Carry out forced to "0".

There are two add states, AD0 and ADI. AD1 is utilized, in many cases, as a modifier to extend the control decoding combinations of other functions. The add operations add the left and right inputs to the ALC with the Carry In (CI). AND simply and the left and right ALC inputs while setting Carry Out (CO) to the one state. OR simply ors the left and right ALC inputs while setting CO to zero. All control settings for the ALC other than AD1 are refered to as $\overline{AD1}$ in the other sections; they are, AD0, AD2 and AD3.

1.3.4 Destination Shift Selector Control

This group of three bits (M2, M1 and M0), called the M-bits, has control of the Shift Selector, LC enable into the registers file, and the output multiplexer source selection. The following table details M-bit control decoding.

Table 1.3-4 Destination Shift Select Control

Reference	Inputs M2	M1	M0	Description Function	Modifier	Comment
A00	0	0	0	Register* ← DI File DO ← ALC	SS1	If SS1 is not present garbage is loaded into the register file.
A01	0	0	1	Register ← ALC Left 1 File DO ← ALC	MX bit states dependent on C bits	AO0 through A05 is ALC to circuit output.
A02	0	1	0	Register ← ALC Right 1 File DO ← ALC	MX bit states dependent on C bits	
A03	0	1	1	Register ← ALC Right 2 File DO ← ALC	MX bit states dependent on C bits	
A04	1	0	0	Register ← Register File File DO ← ALC		LC disabled to register file, "no load".
A05	1	0	1	Register ← ALC File DO ← ALC		
A06	1	1	0	Register ← ALC File DO ← P2B		P2B to circuit output.
A07	1	1	1	Register ← ALC File DO ← P1B		P1B to circuit output.

*All references to the register file as a destination assume LC is generated to perform the load. If LC is not generated all operations will be performed (as dictated by the control decoding) except the final load into the register file.

As noted in the comments, all M-bit control states select ALC to be output on DO, except A06 and A07, which select P2B and P1B, respectively, to be output. In addition the M-bits control what gets written to the register file, through the Shift Selector and they enable the LC into the register file for writing results. If AO0 is selected for a direct input from the data input pins, SS1 also must be selected; otherwise, garbage will be loaded into the addressed register. During this direct in to the register file, P1B retains its previous contents so pipelining of the GPU is possible. That is, simultaneous with the direct in, the results of the ALC operation appear on the output (if enabled). When the LC goes back down, P1B will then follow the register file.

For left shifts of one, the shift input comes into the least significant bit through MXLO and MXHO outputs the most significnat bit of the ALC (dependent on the state of the C-bits). The directions are reversed for a right shift of one; the input comes into the most significant bit from MXHO and the least significant bit is output on MXLO. For a right shift of two, MXH1 inputs to the most significant bit and MXHO to the next most significant bit (bit position 6). MXLO outputs the least significant bit and MXL1 outputs the next least significant bit (bit 1). The following schematics illustrate shift connections for the MX bits when the C-bits are set for normal shifts (RF1).

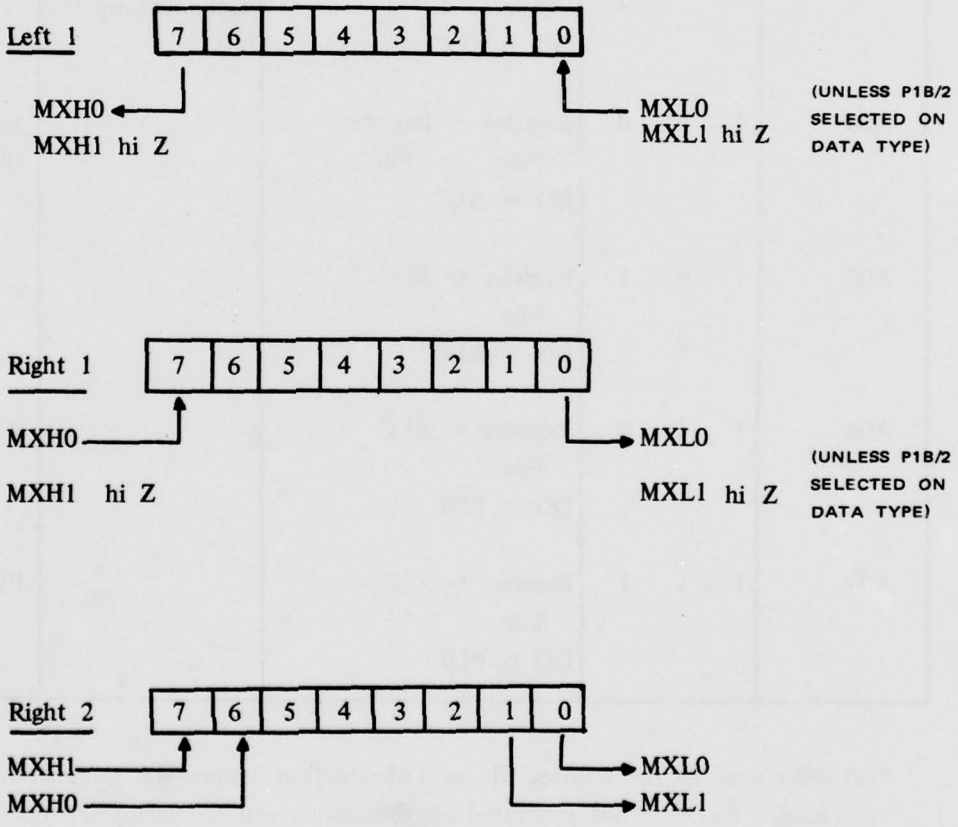


Figure 1.3-1 Use of MX-bits as shift carries.

The C-bits control what actually gets shifted into the GPU. Certain decodes of the C bits disconnect the MX lines and input zeros, ones or extend the sign (refer to next section).

A04 disables the clock to the register file. Therefore, it is possible to design the GPU into synchronous, free running, clocked systems without any external clock gating.

1.3.5 Boundary and Connect Control

This group of 3-bits, C2, C1 and C0, called the C-bits controls the enabling of MX bit drivers and determine what data is shifted in and out of the GPU. The following table details C-bit decoding states.

Table 1.3-5 Boundary and Connect Control

Reference	Inputs C2	C1	C0	Function	Comment
RF0	0	0	0	OFF	MXH0, MXH1 in Hi Z state MXL0, MXL1 states dependent on M-bit decoding.
RF1	0	0	1	Normal intercircuit shift connections	M-bits are a modifier for all C-bits decodes, except RF5, RF6, RF7.
RF2	0	1	0	One shift in, OVF → MXH1	
RF3	0	1	1	One shift in, OVF → MXH1	
RF4	1	0	0	MSB extend for Right shift; otherwise, ALC_7 → MXH0, ALC_6 → MHX1	
RF5	1	0	1	ALC_7 → MXH0, TS → MXH1	
RF6	1	1	0	ALC_7 → MXH0, $P2B_7$ → MXH1	
RF7	1	1	1	ALC_7 → MXH0, $P2B_7$ → MXH1 ALC_7 → TS	

The C-bits condition the data paths of the MX bits and Shift Selector for shift operations; if a shift is not specified by the M-bits, then the C-bit decoding does not affect the data entering the Shift Selector. RF0 turns off the MXH bits irregardless of the M-bit control, however, the MXL bits are not affected by RF0. RF1 is the control decode for normal intercircuit shifting (refer to previous section for details). RF2 causes a zero to be shifted in to the least significant bit if a left shift is specified or into the most significant bits if a right shift of one or two is specified. Also, the overflow status signal (OVF) is output on MXH1 irregardless of the M-bit decoding. RF3 is exactly the same as RF2, except that it causes a logical one to be shifted in. RF4 causes the most significant bit to be extended for right shifts (A02 or A03) only; otherwise ALC_7 and ALC_6 appear on MXH0 and MXH1, respectively. RF5 through RF7 output status information of various data paths on the MXH pins. *RF5 through RF7 should never be selected simultaneous with a shift command on the M-bits* (in some cases the result will be indeterminant).

1.3.6 Data Output Enable

Data Output Enable is a single control input that turns on the Data Output tri-state drivers on the GPU. This signal is active high; a zero state places the drivers in the high impedance state.

1.3.7 Load Clock

All data path transactions and microprogram control sequencing are controlled by a single forcing function on the GPU. It is called the Load Clock (LC) and it directly controls the loading of data into the register file, P1B and P2B. While LC is low, P1B (except in the case of a direct load into the register file) and P2B (except in case of P2B = P2B) follow the register file port outputs. When LC goes high, both P1B and P2B are disconnected from the file (locking in their data) and the results of the data path operation are stored away into the register file.

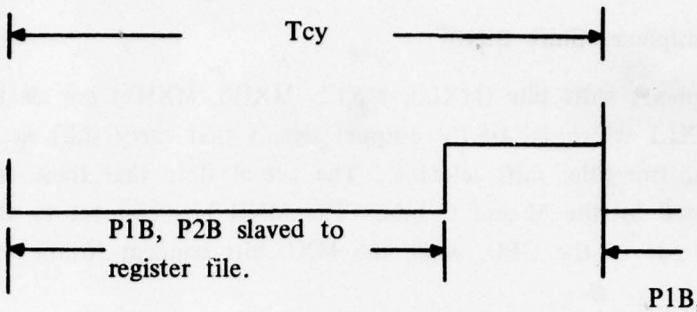


Figure 1.3-2 Effect of the load clock.

1.4 Data Signal Descriptions

The following paragraphs describe GPU data input and output signals.

1.4.1 Data Input

There are eight data input pins, DI_0 through DI_7. DI_7 is the most significant bit. Under control of the S and M bits, data can be directly input to the register file, P1B or P2B. All data input is active high; i.e. high voltage is logical 1.

1.4.2 Carry In, Carry Out

Carry In (CI) is an active high input to the GPU ALC. Carry Out (CO) is the active high carry out from bit 7 of the ALC.

1.4.3 All Zero Detect Output

The all zero detect output (AZO) remains in the high impedance state when the ALC output is all zero. When the ALC output *is not zero* the AZO output is pulled down to zero. In a multi-GPU system, all AZO outputs can simply be bussed together to effect a wire-OR.

1.4.4 Multiplexer Shift Bits

The multiplexer shift bits (MXL0, MXL1, MXH0, MXH1) are all bidirectional (except MXL1 which is strictly output) signals that carry shift in and shift out data to/from the shift selector. The actual state that these signals take is determined by the M and C bits. The MXH bits connect to the most significant side of the GPU, while the MXL bits connect to the least significant side.

1.4.5 Data Output

There are eight data input pins, DO_0 through DO_7. DO_7 is the most significant bit. Under control of the S and M bits, data can be directly output to the register file, P1B or P2B. All data input is active high; i.e., high voltage is logical 1.

Table 1.4-1 GPU Control and Data Signals

GPU CONTROL

PIN NO.			
41	A0	ALC Control Bit 0	Input
42	A1	ALC Control Bit 1	Input
33	R0	Port 1 Register Select Address Bit 0	Input
30	R1	Port 1 Register Select Address Bit 1	Input
26	R2	Port 1 Register Select Address Bit 2	Input
27	R3	Port 1 Register Select Address Bit 3	Input
32	T0	Port 2 Register Select Address Bit 0	Input
31	T1	Port 2 Register Select Address Bit 1	Input
28	T2	Port 2 Register Select Address Bit 2	Input
29	T3	Port 2 Register Select Address Bit 3	Input
24	S0	Source Select Control Bit 0	Input
23	S1	Source Select Control Bit 1	Input
45	D0	Data Type Select Control Bit 0	Input
44	D1	Data Type Select Control Bit 1	Input
40	D2	Data Type Select Control Bit 2	Input
35	M0	Destination Select Control Bit 0	Input
34	M1	Destination Select Control Bit 1	Input
36	M2	Destination Select Control Bit 2	Input
39	LC	Load Clock	Input
12	C0	Boundary, Connect Control Bit 0	Input
13	C1	Boundary, Connect Control Bit 1	Input
14	C2	Boundary, Connect Control Bit 2	Input
8	DOE	Data Output Enable	Input

PIN NO. 1 V_{SS}

PIN NO. 25 V_{DD}

GPU DATA

PIN NO.			
22	D10	Data Input Bit 0 LSB	Input
21	D11	Data Input Bit 1	Input
20	D12	Data Input Bit 2	Input
19	D13	Data Input Bit 3	Input
18	D14	Data Input Bit 4	Input
17	D15	Data Input Bit 5	Input
16	D16	Data Input Bit 6	Input
15	D17	Data Input Bit 7 MSB	Input
43	C1	Carry In	Input
37	MXL0	Multiplexer Shift Low Bit 0	TS*Input/Output
38	MXL1	Multiplexer Shift Low Bit 1	TS Output
10	MXH0	Multiplexer Shift High Bit 0	TS Input/Output
11	MXH1	Multiplexer Shift High Bit 1	TS Input/Output
46	DO0	Data Output Bit 0 LSB	TS Output
47	DO1	Data Output Bit 1	TS Output
48	DO2	Data Output Bit 2	TS Output
2	DO3	Data Output Bit 3	TS Output
3	DO4	Data Output Bit 4	TS Output
4	DO5	Data Output Bit 5	TS Output
5	DO6	Data Output Bit 6	TS Output
6	DO7	Data Output Bit 7 MSB	TS Output
7	CO	Carry Out	Output
9	AZO	All Zero Detect Output	TS Output[

*TS = Tri-State

Table 1.4-2 GPU Pin Signal List

Pin No.	Mnemonic	Description	Pin No.	Mnemonic	Description
1	V_{SS}	OV, Ground Reference	25	V_{DD}	4V to 13V
2	DO3	Data Output bit 3	26	R2	Port 1 Register Select bit 2
3	DO4	Data Output bit 4	27	R3	Port 1 Register Select bit 3
4	DO5	Data Output bit 5	28	T2	Port 2 Register Select bit 2
5	DO6	Data Output bit 6	29	T3	Port 2 Register Select bit 3
6	DO7	Data Output bit 7	30	R1	Port 1 Register Select bit 1
7	CO	Carry Out	31	T1	Port 2 Register Select bit 1
8	AZI	All Zero Detect In	32	T0	Port 2 Register Select bit 0
9	AZO	All Zero Detect Out	33	R0	Port 1 Register Select bit 0
10	MXH0	Multiplexer Shift High bit 0	34	M1	Destination Select Control bit 1
11	MXH1	Multiplexer Shift High bit 1	35	M0	Destination Select Control bit 0
12	C0	Boundary, Connect Control bit 0	36	M2	Destination Select Control bit 2
13	C1	Boundary, Connect Control bit 1	37	MXL0	Multiplexer Shift Low bit 0
14	C2	Boundary, Connect Control bit 2	38	MXL1	Multiplexer Shift Low bit 1
15	DI7	Data Input bit 7	39	LC	Latch Clock
16	DI6	Data Input bit 6	40	D2	Data Type Select Control bit 2
17	DI5	Data Input bit 5	41	A0	Adder Logic Circuit Control bit 0
18	DI4	Data Input bit 4	42	A1	Adder Logic Circuit Control bit 1
19	DI3	Data Input bit 3	43	CI	Carry In
20	DI2	Data Input bit 2	44	D1	Data Type Select Control bit 1
21	DI1	Data Input bit 1	45	D0	Data Type Select Control bit 0
22	DI0	Data Input bit 0	46	DO0	Data Output bit 0
23	S1	Source Select Control bit 1	47	DO1	Data Output bit 1
24	S0	Source Select Control bit 0	48	DO2	Data Output bit 2

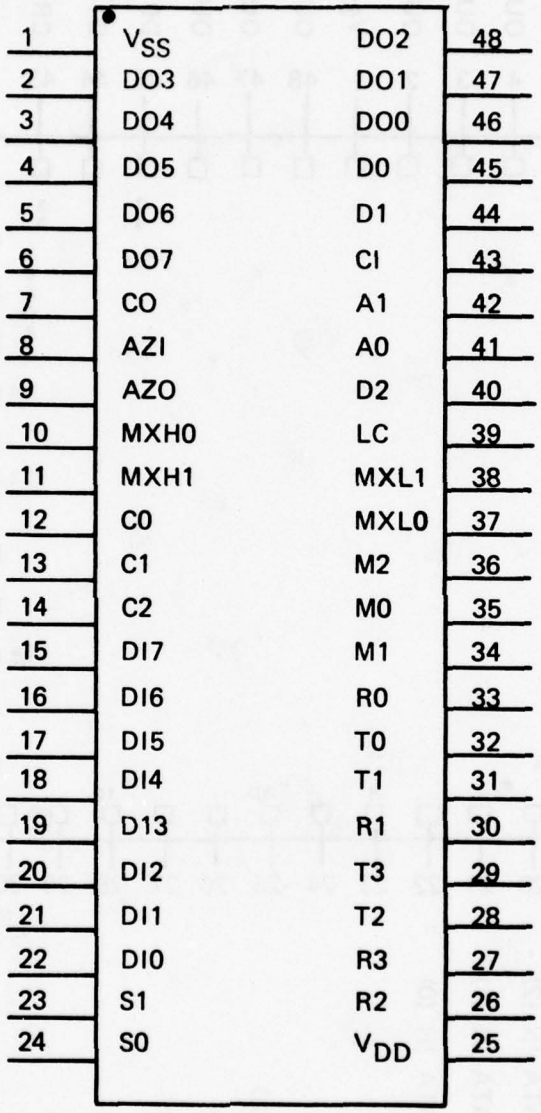


Figure 1.4-1 GPU Dual In-line Package (top view)

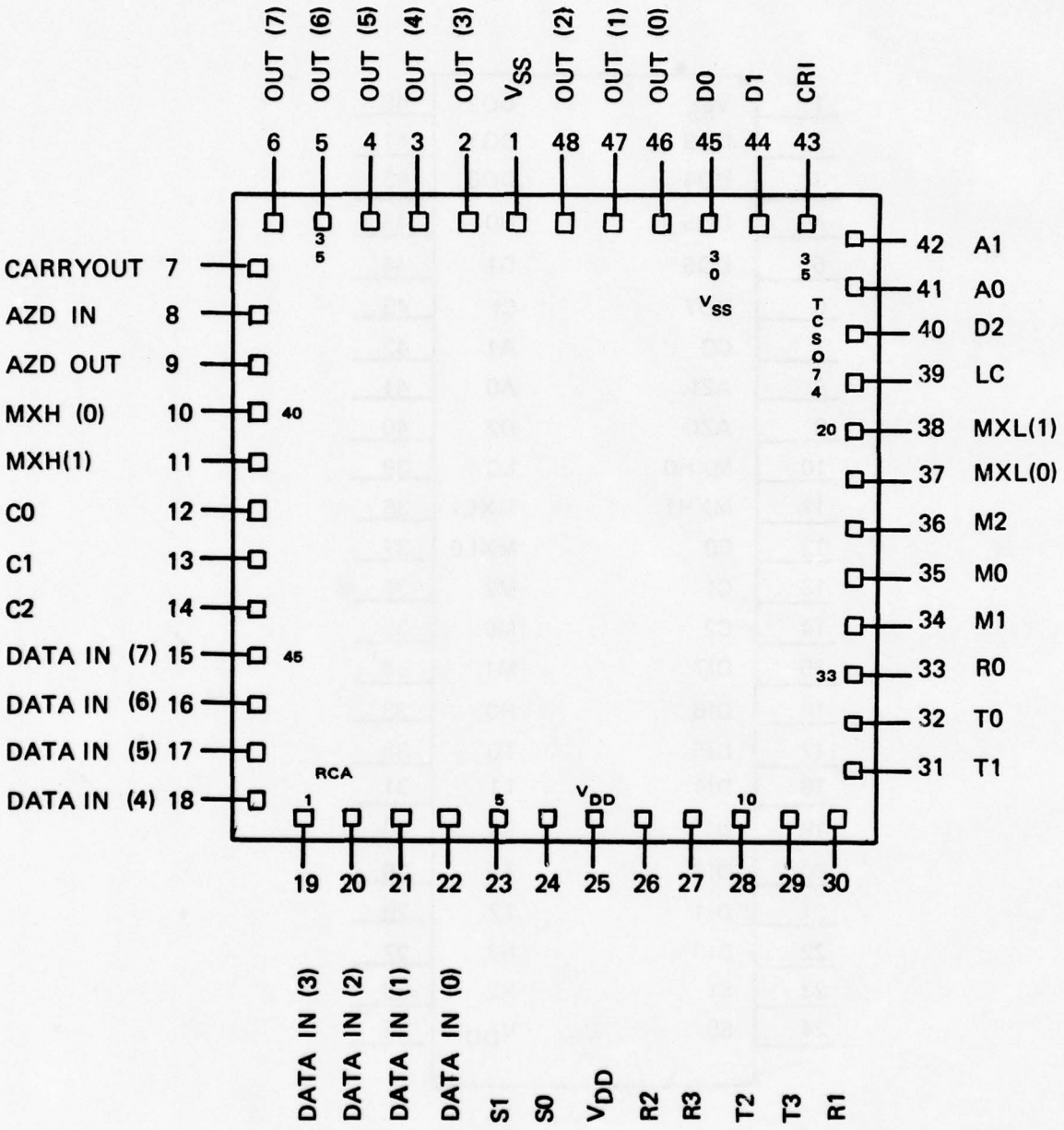


Figure 1.4-2 BONDING DIAGRAM AND PIN OUT TCS074

1.5 GPU Timing Restrictions

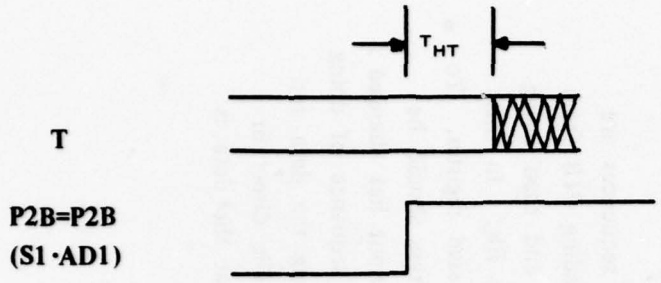
As mentioned previously there are several timing restrictions on GPU control signals that are the result of race conditions on P1B and P2B (refer to Section 3.1). The following sections present rules of operation that avoid the race conditions. No actual times are specified since the device has not been characterized. However, these times should be specified by the manufacturer when the device is characterized.

1.5.1 P2B Hold Time

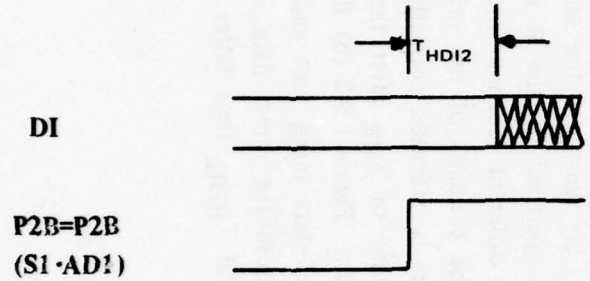
There are two data sources into P2B, DI and Port 2 of the register file as specified by the T address. When controls are set up such that P2B receives data from one of these sources, that data must remain stable for a hold time when switching to P2B = P2B (S1 · AD1). As illustrated in Figure 1.5-1 (a) the Port 2 address, T, must remain unchanged for a time of T_{HT} after P2B controls are set to P2B = P2B. The same situation is true when loading P2B from DI (refer to Figure 1.5-1 (b).) The data input must remain stable on the DI pins for T_{HDI2} after P2B controls are set to P2B = P2B.

1.5.2 P1B Hold Times

P1B hold time situations are similar to P2B except the control sequences are slightly different. Figure 1.5-2 (a) illustrates the sequence of loading P1B from Port 1 of the register file (as specified by the R address) first and then doing a direct load from DI into a different location into the register file. In this situation, P1B should retain the contents of the originally addressed register. To insure this operational sequence is executed correctly the R address should be delayed from changing by a time of T_{HR} after the command input has changed to the register file direct input. Figure 1.5-2 (b) illustrates the sequence of doing a load of P1B from the direct data input pins and then changing the data and loading into the register file from the direct data input pins. The direct in data must not be changed until T_{HDI1} time latter to insure that the data is retained in P1B.

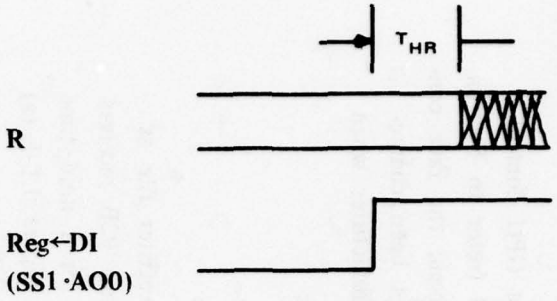


(a) Switching from P2B←Reg (T);
P2B←P2B

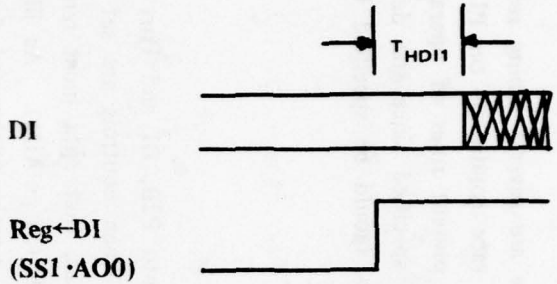


(b) Switching from P2B←DI;
P2B←P2B

FIGURE 1.5-1 P2B HOLD TIMES



(a) Switching from P1B←Reg (R);
Reg(X)←DI



(b) Switching from P1B←DI;
Reg(X)←DI

FIGURE 1.5-2 P1B HOLD TIMES

1.6 Cycle Time and Clock Characteristics

The following sections outline timing specifications that should be determined by the manufacturer in the course of characterizing its performance; this information is required by designers in utilizing the device.

1.6.1 Load Clock

The GPU is guaranteed to function correctly at the times specified below over the temperature range.

Table 1.6-1 GPU Timing Data

TIME	5V	10V	15V
Read-Modify-Write (Time from R&T change to end of Write Cycle)			
Maximum Clock Freq. for getting valid data out on longest delay path, AZO			
Minimum Clock Period			
Minimum High Time			
Minimum Low Time			

1.6.2 Data Path Delays

The GPU is guaranteed to function correctly at the times specified in Table 1.6-1 over the temperature range.

Table 1.6-2 GPU data path time delays.

From \ To	DO			CO			AZO			OVF			TS			MXH$_X$			MXL$_X$		
V_{DD} (VOLTS)	5	10	15	5	10	15	5	10	15	5	10	15	5	10	15	5	10	15	5	10	15
R, T			[1]			[2]															
DI			[3]			[4]															
CI						[5]															
S (P1B, P2B Source)																					
D (Data Type Select)																					
A (ALC Control)																					
M (Shift Select)																					
C (Connect Control)																					

Notes:

1. Through P1B to output and P2B to output.
2. Worst case propogation delay with CI=0.
3. Through P1B to DO
4. Adding DI+P2B with CI=0
5. Worst case carry propogation delay.

1.6.3 Set-up and Hold Times

Set-up and hold times are defined relative to the clock low-to-high transition. Inputs (both data and control) must be steady at all times from set-up time prior to the clock until the hold time after the clock. The set-up times allow sufficient time to perform the correct operation, on the correct data, so that the correct ALU data can be written into the register file correctly. Hold times for P1B and P2B are explained in Section 1.5.

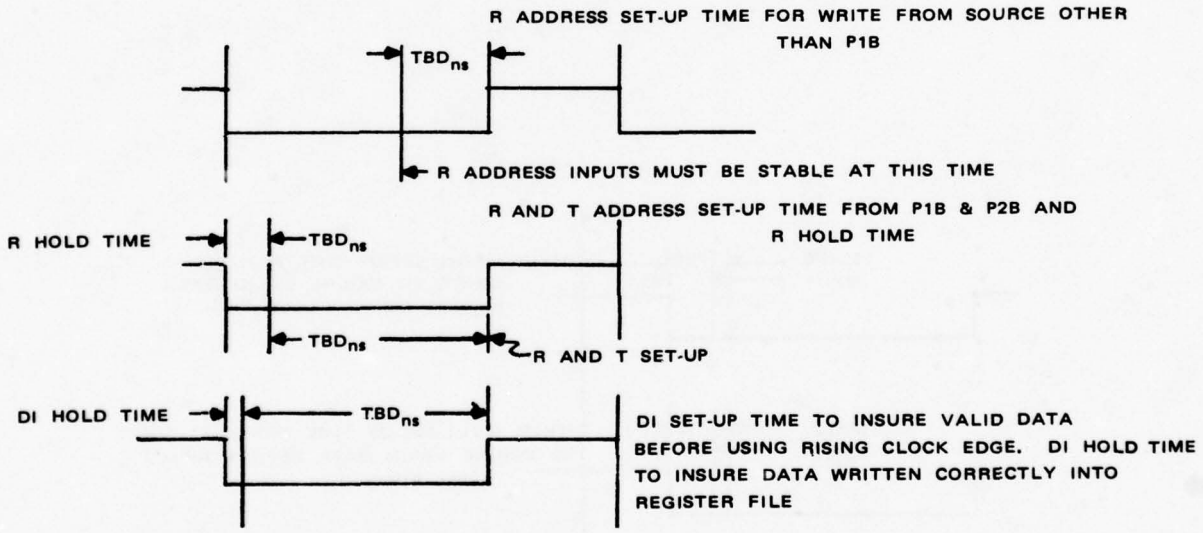


Figure 1.6-1 GPU setup and hold times. (Continued next page)

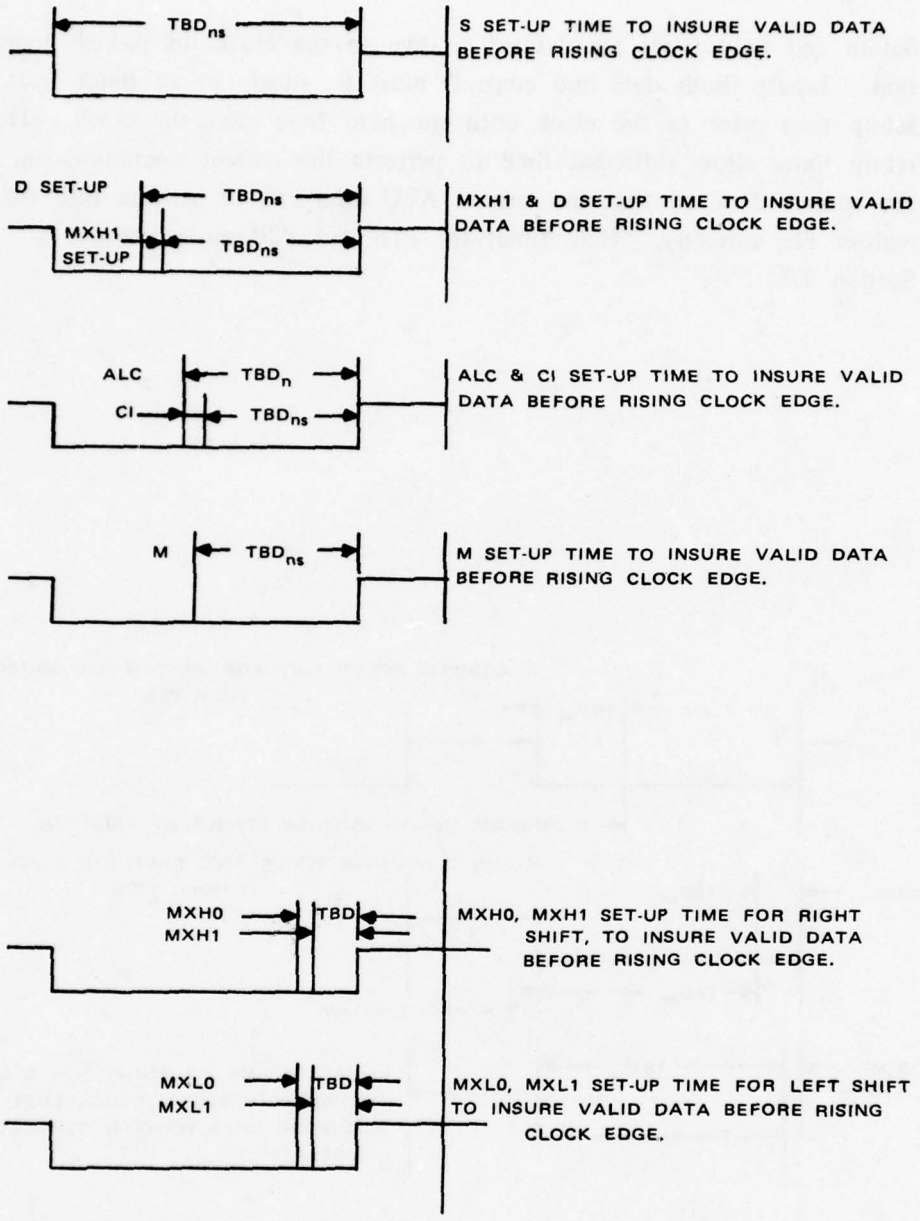


Figure 1.6-1 (Cont.) GPU setup and hold times.

2. Emulation Analysis

The concept of hardware emulation is not new considering the impressive example of the Amdahl 470V computer which can execute all of the IBM 360/370 software without modification at approximately one-half the IBM system cost. This investigation resulted in an application study of a family of devices that can be used to emulate a variety of computers. Microprocessor bit slice families are uniquely qualified for emulation of other computers because they do not restrict the designer, but rather enable him to tailor his machine to match another architecture so that the original software will run on either machine. Such an approach will be valuable to the Air FOrce as it will provide, at decreased life cycle cost, significant increases in the reliability of systems, while presenving expensive software investments. This approach will also provide the AIr Force with a unique growth capability in systems, not obtainable with today's fixed architectures.

The approach taken in the emulation analysis was two-pronged. First, the effectiveness of the device family in emulating various computers from simple 4-bit microprocessors to high power 32-bit floating point processors was analyzed. Several deficiencies in the functional design of the GPU were found during this analysis and are now being corrected in a new design. More importantly, deficiencies in the breadth of the device family were found to exist and recommendations for additional part types for the family are proposed (refer to **Section 2.5**) to ameliorate the deficiencies. The emulation capabilities analysis results are presented in Section 2. Second, a parallel effort was undertaken to define and then design, to the gate level, a particular computer for emulation utilizing the device family. This effort provided a real test of the true emulation capabilities of the device family. As a result of this effort, deficiencies in the devices were found—they are also reported on in Section 2. The computer selected for emulation was the 8080. This microprocessor is representative of the higher performance microprocessors on the market and, as such, was an excellent emulation analysis vehicle. The details of the breadboard design of the 8080 Emulator are presented in Section 4.

2.1 Design Completeness of the GPU

The following sections present the results of Questron's detailed analysis of the GPU. First, the GPU's functional capabilities were analyzed based on the original design specification and design documents. In addition, it was compared to such industry standards as the AMD2901 to determine if there were any glaring deficiencies in its functional design. The part was rigorously tested (refer to section 3.1) to determine all its capabilities (many functional capabilities of the GPU are not documented). Second, the GPU's emulation capabilities were examined. Requirements for 4, 8, 16, 24 and 32-bit machines were analyzed to determine if any deficiencies exist in the GPU relative to emulation usage. Design modifications have been recommended for the GPU and some are being incorporated into a reworked GPU.

Although, many points are raised about the GPU design completeness in the following sections, in all, it is considered an extremely powerful device. In fact, there is nothing on the market, today, that can match its capabilities. With the modifications being made (recommended by Questron), the viability of the device will be further enhanced.

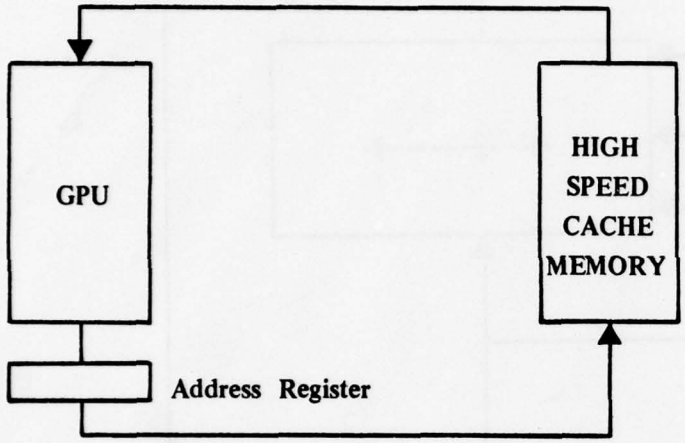
2.1.1 GPU Functional Design Deficiencies

This section itemizes functional deficiencies in the GPU. Some have been ameliorated in a design for a reworked GPU – others have not because of schedule constraints (each section denotes whether the deficiency has been corrected in the new design).

2.1.1.1 GPU Pipelining

Direct data input to the GPU can be directed to three destinations; P1B, P2B and directly into the register file. For direct input operations to the register file, the intent was that the output drivers be turned on with the ALC selected for output thus allowing pipeline operation of the GPU in system configurations. Currently the output drivers are disabled. Pipeline operation of the GPU has significant system throughput impact across the entire spectrum of applications;

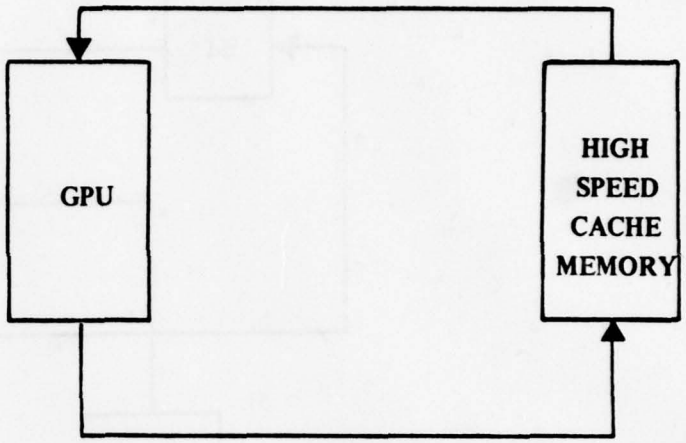
Current Implementation



Steps

1. Form effective address, output to Address Register.
2. Store operand into Register File.

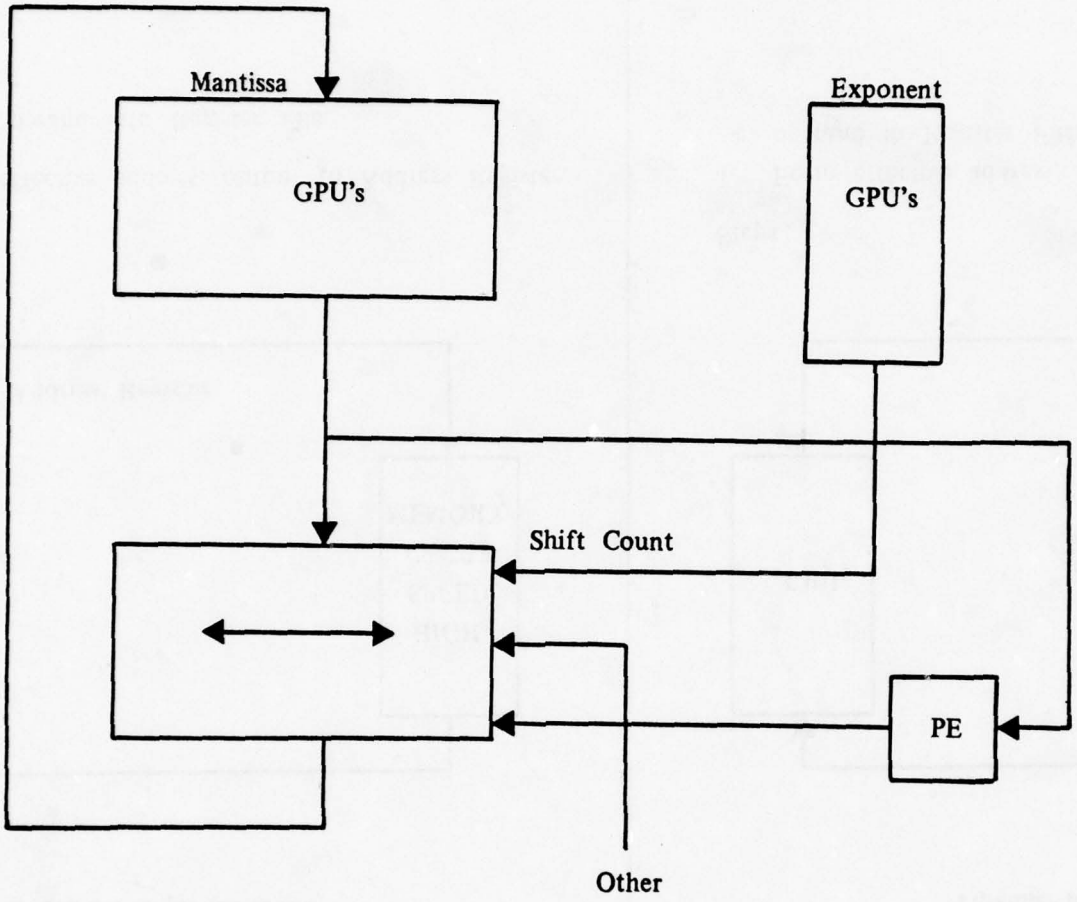
Pipeline Implementation



Steps

1. Form effective address, output to memory, store operand in Register File.

Figure 2.1-1 Example of Pipeline Operation in Control Processor



Steps

1. Weigh exponents and align Mantissa.
2. Perform dyadic and post normalize.

Figure 2.1-2 Pipelining in Floating Point Processor

from 8080 Emulator to signal processor. It enables one step execution of operations that now require two steps. If pipeline operations are nested in critical processor control cycles (cycles that are continuously performed) such as instruction and operand fectching, throughput can be considerably enhanced. GPU pipeline capability also has inherent hardware advantages – holding registers for data and addresses are not required. For example, the GPU can output an address, and hold it, and receive accessed data back in one control state. This is illustrated in Figure 2.1-1. Figure 2.1-2 illustrates GPU pipelining in a floating point processor. The floating point operation is performed in two steps. The exponents are weighed by the exponent GPU's a shift count generated and the mantissa aligned in the first step. This requires simultaneous direct in and direct output to and from the register file. In the second step the dyadic is performed and the result is post normalized. To perform this operation, in one step, also requires pipeline operation of the GPU. The result is output, fed through a priority encoder (PE), to generate a shift count, and normalized through the shifter then stored back into the register file; all in one step. The examples in both figures illustrate the requirement to be able to simultaneously output ALC results while doing direct input to the register file; in Figure 2.1-1 for effective address calculation and in Figure 2.1-2 for post normalization. This deficiency is being corrected in the reworked GPU design.

2.1.1.2 Missing Overflow Detection Logic

Static testing has determined that there is no overflow detection logic implemented on the GPU. The Tracor Inc. GPU design specification dictates that the overflow signal and the carry out signal are to share the same pin for output. For control decode $\overline{C2} \cdot \overline{C1} \cdot C0$ the overflow detection logic signal was to be output on the carry out pin. Overflow is defined as simply a change in the sign bit when performing addition or subtraction; for example, if the sign bit goes to a one state (negative) during the addition of two positive numbers, overflow as occurred. Overflow can be easily detected by taking the exclusive OR of the carry out of the most significant bit:

$$\text{Overflow} \equiv C_7 \oplus C_6$$

On examining the logic diagrams it appears that, not only, the exclusive OR function but the control decode of $\overline{C2} \cdot \overline{C1} \cdot C0$ and overflow, carry out selection must be added to the GPU.

For the reworked GPU it was originally proposed that the overflow signal be time shared on the MXH output pin; this was a good idea since now the Carry out and overflow signals were available simultaneously. However, in performing simulations, Questron discovered a basic conflict in this implementation with other control functions. It is Questron's recommendation that MXH1 be utilized to bring out the overflow signal.

2.1.1.3 Carry-in Select

This capability is functionally simple in concept, yet because of control pin limitations it is not currently being considered for the reworked GPU. The idea is that the carry signal applied to the carry input would be disabled from entering the GPU carry network under specific control conditions. This capability saves external gating of the carry signals required in many floating–point processors. Consider, for example, a 32-bit floating point processor that shares its data paths between fixed and floating-point operations. Typical floating-point processors of this kind utilize data formats as illustrated in Figure An 8-bit exponent is operated on in the most significant portion of the data paths and the 24-bit mantissa is operated on in the least significant portion. As illustrated in the figure, external gating is required between the exponent and mantissa GPU's. The reason being that for fixed point operations the most significant GPU gets a carry input from the next GPU. However, for floating point operations this carry needs to be disabled and the carry input received from another source (it could be hardwired to zero). Having select capability on carry-in also would be useful on the least significant GPU. Here, on virtually every computer considered for emulation, four carry input sources must be selected. They are as follows:

- one (1)
- zero (0)
- carry out (from most significant GPU)
- $\overline{\text{carry out}}$ (from most significant GPU)

Therefore, the proposal would be to provide a selector to select between four possible Carry-in sources. Two control lines would be required and two carry in pins would be needed (one and zero carry in's would be hardwired on the part). As stated previously, this proposed fix is not being considered because of pin limitations.

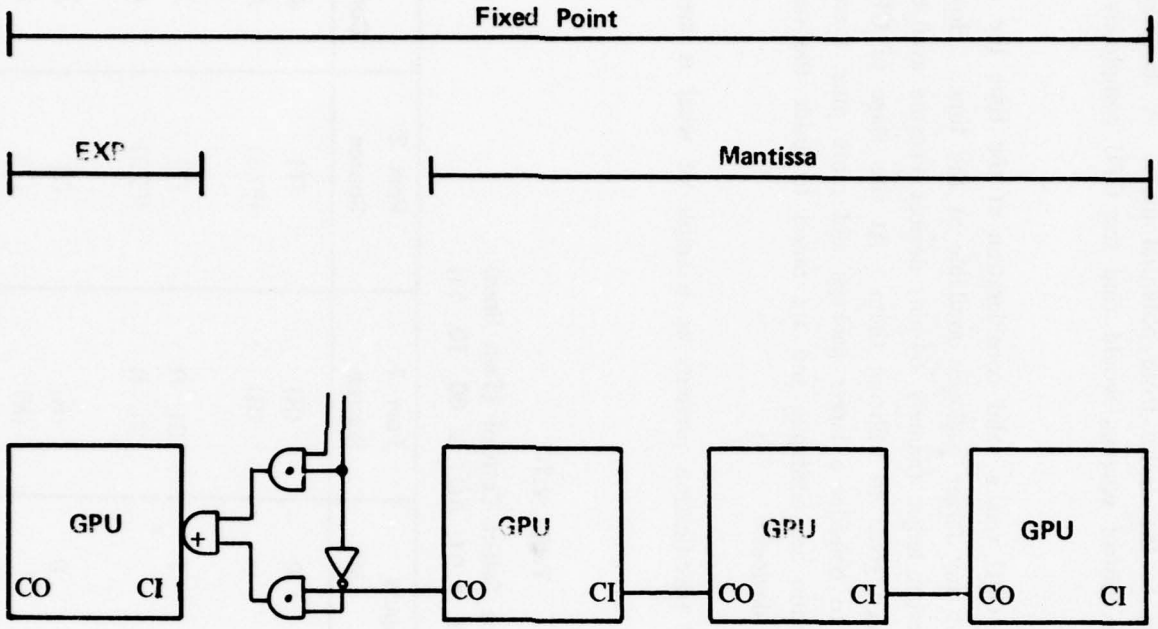


Figure 2.1-3 Sharing of Data Paths for Fixed, Floating Point

2.1.1.4 Control Function Sharing

Because of a policy of keeping the pin-out requirements of the GPU at 48 or less, some functional capabilities were sacrificed. To gain back some of the scarificed functional capability, control function sharing was incorporated into the design. This technique (control function sharing) of control makes the GPU an extremely difficult device to understand. Because of this, Questron has received some negative reaction to the device from potential users. A designer familiar with the 2901 type control scheme, would find the GPU completely foreign.

Limiting the pin-out of the GPU was a valid consideration at the time the design was initiated; there were not any larger packages available at the time. However, during the course of the design, larger (namely 64-pin) devices became available and consideration should have been given to utilizing them. At this stage of GPU development, it is too late to consider a larger package and more pins; therefore the points made in this section are academic and are raised to guide the design of future processor bit-slice devices.

The following control signal specification presents an example of what is meant by control function sharing:

Table 2.1-1

Source Select Control (Two lines)
(AD1 ≡ 01; $\overline{AD1}$ ≡ 00, 10, 11)

Reference	Inputs S1	Inputs S0	Port 1 Source	Port 2 Source	Condition
SS0	0	0	(R)	(T)	$\overline{AD1}$
			(R)	(P2B)	AD1
SS1	0	1	DI, R	(T)	$\overline{AD1}$
			DI, R	(P2B)	AD1
SS2	1	0	(R)	DI	$\overline{AD1}$
			(R)	(T)	AD1
SS3	1	1	(R + 1)	DI	$\overline{AD1}$
			(R + 1)	(T)	AD1

Here, the control inputs S1 and S0 determine the data sources for the two port buffers, P1B and P2B. Notice, however, that there are two possible states for each decode of S1, S0. These two states are controlled by the Adder/Logic Circuit (ALC) control function input. AD1 is an addition control decode; $\overline{AD1}$ is a control decode for addition, AND or OR. So, the ALC control function determines P1B, P2B data sources. The problem is that now the designer must consider what type of arithmetic/logical operation he is performing to select sources into P1B and P2B. The problem is further compounded by the fact that the ALC control decoding also determines the data type to be entered into the ALC circuit. Now the designer must keep in mind the functional control on two other subfunctions while writing control equations – an extremely difficult task. Control function sharing also has a deleterious impact on support software (such as microprogram assemblers) developed for the device. The microprogram assembler developed by Questron is 25% to 30% longer because of control function sharing. More difficult to quantize is the complexity of the microassembler. Instead of being a simple string translator, it is almost a compiler in that it must recognize the context that phrases are used in (refer to Section 4.5).

The source select control for P1B and P2B, presented in the previous paragraph, is just one example of control sharing – there are others. If pins were available the control signals for P1B and P2B should be increased to three, thus providing the desired data source selection capability independent of the other functions on the device. But, as stated previously, it is too late in the GPU development cycle to consider such changes. It is felt that the functional capabilities of the device will over shadow the difficulties in using the device. However, it is recommended that control function sharing not be used in future Air Force processor bit slice designs where at all possible.

2.1.1.5 Asynchronous Design

A fundamental aspect of microprogram controlled machines is the simultaneous (on a clock edge) switching of control signals with each fetch of a micro command. Because of race conditions between R,T and S,A and M on the GPU, certain operations between the register file and the port buffers (P1B, P2B) cannot be performed under microprogram control. This not only severly limits the microprogram capability of the device but makes its operation much to esoteric for the average user because certain sequences of normally valid microcommands would have to be declared illegal. Questron considers this to be a serious flaw in the device. Refer to Section 3.1 for details.

The reason for the race conditions is that, both, P1B and P2B are essentially non-clocked storage elements. As a result, when control signals are simultaneously changed, internal decoding races cause garbage to be loaded into P1B or P2B. This problem may have been the result of gradual evolution in the GPU design. Originally, P1B and P2B were complete slaves to the master register file, with no other data sources (refer to Figure 2.1-4). Then as more requirements for the GPU were developed, it was decided to provide other data sources into the register file slave elements, P1B and P2B (refer to Figure 2.1-5a). This is where the heart of the race condition problem lies. By combining the functions of P1B and P2B (being both slaves to the register file and destinations for other data sources) the race conditions were created. P1B and P2B should have been left as slaves to the register file and another tier of buffer registers added to act as sinks for multiple data sources (refer to Figure 2.1-5b).

Because the addition of a second tier of registers is such a drastic hardware change, the race conditions will be left in the reworked GPU. The device documentation should emphatically warn against the race conditions.

While characterizing the race conditions, another device race condition was discovered. Again the root of this race condition is the fact that P2B is a non-clocked storage element. The problem occurs when the register accessed through the two ports is the same (R=T) and an operation is performed that changes the original value of the contents of the register. If the load clock remains on too long, the result of the operation stored in the register file will race through P2B, be operated on again and a new result stored in the register file, ad infinitum. This places a limitation on the length of the load clock high state, thus limiting the flexibility of the GPU and again imposing another design rule on the user. This problem, in conjunction with the previous, are significant enough for Questron to recommend that P2B be redesigned and be made a clocked storage element or that P2B deselected from the register file while data is being written into the file (during Load Clock high). This race condition will be fixed in the reworked GPU.

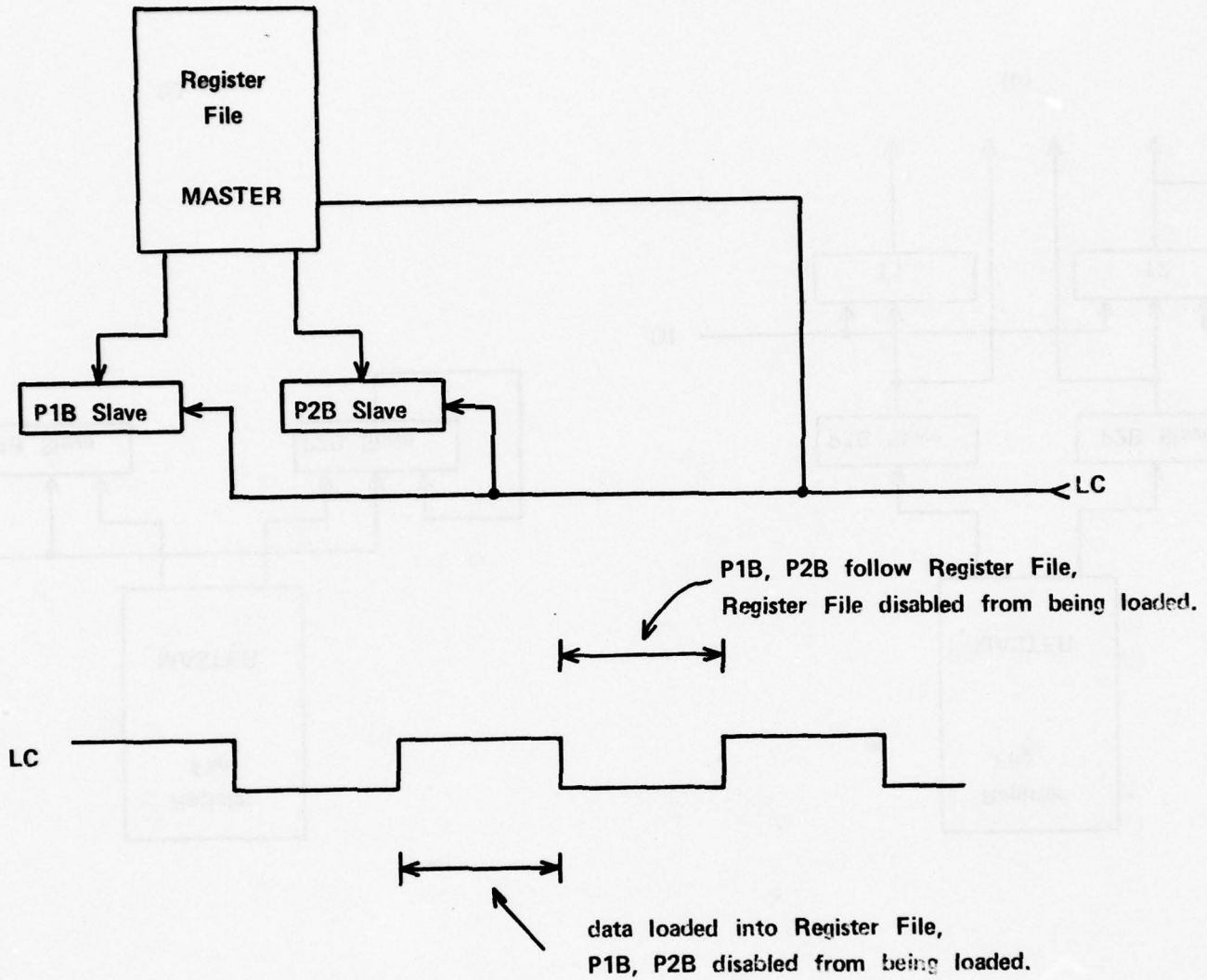


Figure 2.1-4 Register File, P1B and P2B Master Slave Relationship

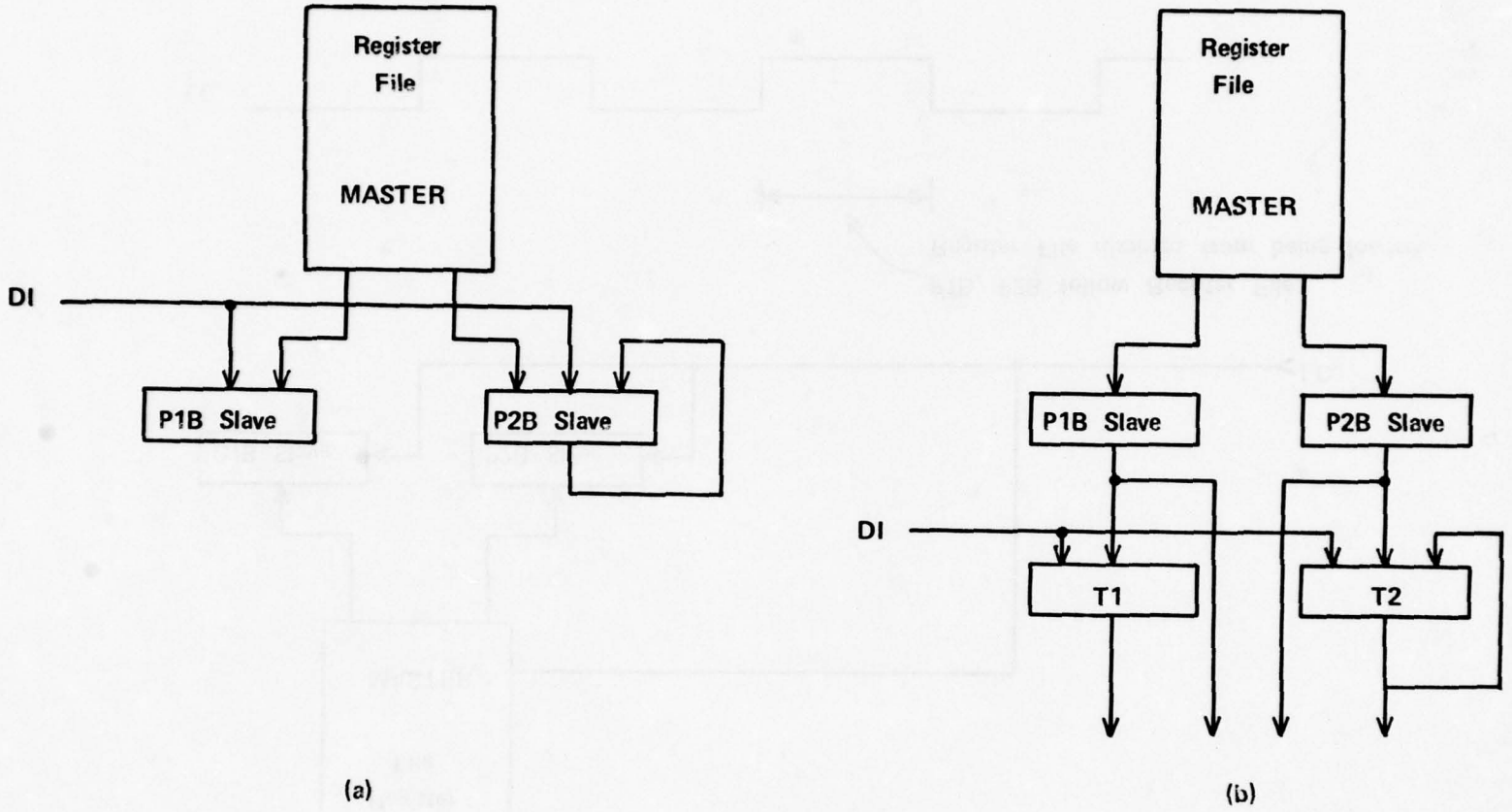


Figure 2.1-5 Separating Master-Slave and Multiple Source Functions

2.1.2 Emulation with the GPU

No major deficiencies in the functional emulation capability of the GPU were discovered during this analysis. The GPU architecture is well thought-out and is well suited for the implementation of computer central processor units (CPU's) across the entire applications spectrum. It should be stated that this analysis did not consider dedicated, special purpose control applications; however it is our feeling that the GPU could satisfy the requirements of a large segment of that market.

Early in this analysis it was decided that Questron would study, in detail, the emulation capabilities of the GPU in the lower end of the emulation applications spectrum—this lower end is made up of 16, 8 and 4-bit microprocessors. The reason for Questron's concentration on the emulation of microprocessors is that a parallel effort at a contractor was examining GPU emulation capabilities in the mid and high ends of the applications spectrum. So, Questron decided to look into the 8080 microprocessor as an evaluation vehicle. The utilization of the 8080 microprocessor as an evaluation baseline has several advantages. The 8080 microprocessor is representative of high performance microprocessors and, as such, its use as an evaluation baseline will constitute a true test of GPU emulation capability. In addition, a design for a high speed, low power, radiation hardened emulation of the 8080 microprocessor would have wide appeal. The paper design, resulting from this study, is owned by the Air Force so hardware could be procurred to the design.

As stated previously, no major deficiencies in the *functional* emulation capabilities of the GPU were found in this analysis (outside of those presented in Section 2.1.1). *However, there is a major problem in the electrical performance (refer to Section 3.2) of the GPU.* Questron feels the problem is so severe it could jeopordize the future of the GPU, and for that matter, the entire CMOS/SOS device family. Details are presented in the next section (2.1.2.1). In subsequent sections the capability of the GPU in implementing 4, 8, 16 and 32 bit machines is examined. In addition, Section 2.1.2.2 examines a point often brought up by GPU detractors—that it does not contain an MQ register for multiply and divide operations.

2.1.2.1 GPU Off Chip Delays

Questron feels that there is a significant throughput problem with the GPU (refer to section 3.2). If left unchanged, this throughput problem could seriously degrade the GPU's effectiveness in implementing larger word length computers and, for that matter, jeopordize the future of the CMOS/SOS parts family as a viable technology. The specific problem is that the delay for coming off the GPU with data is more than two times the delay for doing register to register operations. For example, at 10 volts, register to register operations can be done in less than 80 ns while the mean for register to data out operations is 182 ns (zero to one) and 186 ns (one to zero). The reason for this large delay is unclear. In fact, the worst case carry out delay time is shorter than a straight data out at 10 volts and higher; the inverse should be the case. Questron has in its possession a CMOS/SOS multiplier device (TCS-057) that was tested for propagation delays. It exhibits delays of less then 30 ns from command input to data output. Therefore, there should be no reason for the large GPU delays. Carry-out propogation delays are also excessive – a mean of 171 ns (zero to one) and 134 (one to zero) for the parts tested. If a 32-bit processor were to be implemented with four GPU's (refer to Figure 2.1-6), the total carry propogation would be greater than 800 ns. There are four carry propogation stages required and each takes approximately 200 ns (mean plus 1 sigma). An 800 ns add time for a 32-bit processor is extremely slow. In fact, only an extremely small segment of the emulation applications spectrum would tolerate such slow speed in order to derive some other benefit the GPU might offer. The same can be said for 24-bit and 16-bit applications; the GPU is just too slow. Even the 8080 emulator will suffer a serious degradation in expected performance. At 10 volts it was expected that the Emulator would have a cycle time (refer to Section 4.1.2) of approximately 120 ns or better. At this cycle time we expected the Emulator to be 5 times faster than the n-channel 8080. It appears that we'll have to settle for a cycle time of approximately 220 ns which means the Emulator will be barely 2 times faster. Even though the GPU is significantly fast for register to register operations, two 2901 parts (implementing an 8-bit processor) would be much faster on a system basis.

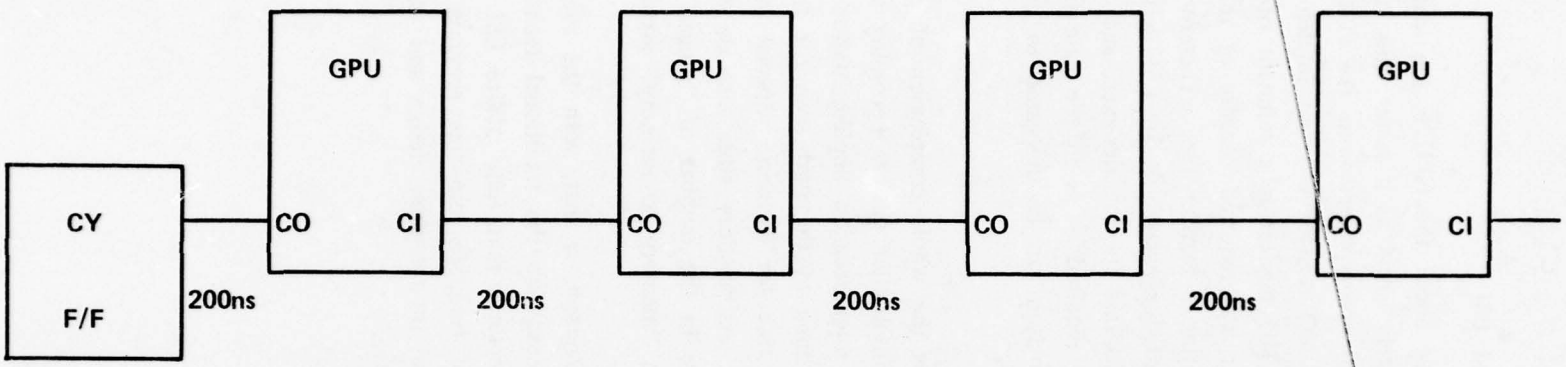


Figure 2.1-6 Carry Out Propogation Delay for 32-bit Processor

2.1.2.2 The MQ Register and LSI

The first thing designers notice about the GPU is the lack of the traditional MQ (Multiplier/Quotient) register. And as a result, this lack of the MQ register, turns out to be the major complaint about the GPU (In fact, it is the only one we've heard). The MQ register was not implemented in the GPU for very good reasons. The MQ register is a remnant of MSI design. In MSI design, a separate register was required because of the lack of effective memory devices for implementing a register file. Therefore, with MSI design, the MQ register was implemented separately. In LSI design, the functions of the register file can be specified by the designer, and if correctly defined, a separate MQ register is not required – as in the case of the GPU. With the GPU, the MQ register function can be delegated to one of the registers in the file.

Effective usage of LSI changes the whole complexion of processor design. For example, with more LSI multiplier devices becoming available and with the inevitable drop in prices, microprogram implementation of multiply operations is going to become a thing of the past even for the simplest of machines. The same will be true for division. Special purpose division LSI devices will become available. As memory sizes increase and as access times tumble, we will see a decrease in the number of "general purpose register" designs and see an increase in "memory to memory" architectures.

The point of the previous paragraph, is that, with the availability of LSI technology, designers should reexamine the traditional techniques of computer design and consider architectures that fully utilize LSI. The MQ register argument is an example of tradition impeding progress. LSI and VLSI are going to cause a revolution in computer design and the traditionalists will soon find tradition obsolate.

2.2 ROM Design Critique

The ROM (TCS-075) suffers several functional design flaws that severly limit its utility as a microprogram memory device. Questron strongly recommends that the ROM be redesigned to incorporate the changes described in the following paragraphs. It's inclusion in the CMOS/SOS GPU parts family as the *only* ROM device, will seriously degrade the viability of the family. The primary problem with the device is its limited capacity. This, coupled with poor organization and marginal features make the device obsolete for microprogram memory type applications.

The design requirement that the ROM output should be able to drive a 50pf capacitance and a TTL interface (capable of sinking 1.6 ma at 0.4V) is *totally incompatible* with its supposed purpose; that of being used as a microprogram ROM for the GPU CMOS/SOS parts family. There is absolutly no requirement for TTL compatibility in a GPU microprogram ROM. It should have the capability to drive a CMOS/SOS load with a reasonable rise time at 30pf – nothing more.

The ROM is organized very poorly for microprogram applications where the driving parameter is word length (i.e. the number of bits in each word). This fact is evidenced in the commercial arena where virtually all recently announced MOS ROM's and PROM's have a word length of 8-bits. A 512 x 2 configuration is useless and a 256 x 4 configuration is of marginal value. A 128 x 8 organization would be most useful. Questron strongly recommends that an 8-bit word length be adopted for the GPU CMOS/SOS parts family ROM.

The limited capacity (1024 bits) of the TCS-075 is the most serious problem with the device. It is, frankly, obsolete. For example, RCA has produced a 4096-bit CMOS/SOS ROM (TCS-078) organized as 512 words by 8 bits; however, its drawback is its large chip size (252 x 257 mils). Questron strongly recommends that the GPU CMOS/SOS parts family ROM have a capacity of *at least* 2048 bits organized as 256 words by 8-bits.

The ROM does have one important feature that should be retained. This feature is the pipeline register. This register is required in microprogram designs that pipeline (or overlap) microprogram ROM accessing with command execution; which is the case for just about every application we've examined.

2.3 GUA Design Critique

No major deficiencies were found in the design of the TCS-091 Gate Universal Array (GUA). It is our feeling, however, that more than one GUA type should be included in the GPU SOS COS/MOS parts family. TCS-091 adequately covers the mid-range of the applications spectrum providing up to 300 gates for use. GUA devices covering the low end and the high ends of the applications spectrum should also be included in the parts family. The TCS-090 (a 182 gate GUA) is perfect for the low end of the applications spectrum and the TCS-093 (a 632 gate GUA) is well suited for the higher end of the applications spectrum; both these devices should be included in the GPU SOS COS/MOS parts family.

The major problem with the TCS-091 is not with the device itself but with its support software and with its documentation. The documentation that exists, namely, "SOS COS/MOS Universal Array Users' Manual", by G. Skorup dated May 1, 1977, is not really oriented to a serious user of the GUA. Such critical information as; detailed propogation delays, fan-out effect on propogation delays, power consumption, use of low Z drivers, etc., is not included in this document. This type of information is critical to the design engineer. In addition, applications examples should be published for the TCS-091; the applications examples should be actual GUA's that have been fabricated. Design automation support software for GUA's is inadequate. Ideally, RCA should have available, on a time-sharing service, design automation programs for unrestricted use by GUA designers. Automatic placement and routing is, by far, the most glaring deficiency in RCA's repetoire of design aids. Computer aided design aids, such as automatic placement and routing, are standard items for most all LSI manufacturers. It is surprizing that RCA does not provide such aides to GUA customers. *For the GPU SOS COS/MOS device family to be truely viable, RCA must provide computer aided, automatic design aids for the GUA's.*

Some difficulties were experienced in utilizing the TCS-091 in the Emulator design which are worth mentioning. Because of the lack of automatic design aides and because of the desire to develop low risk, conservative designs, gate usage was limited to 70% (this percentage was recommended by RCA as a low risk design rule). At this low gate utilization the TCS-091

is in essence a 180 logic gate array as opposed to a 256 logic gate array. Thus, some functions had to be allocated to two GUA's, instead of one, which not only resulted in more part types (not to mention more total parts) but also resulted in a degradation in throughput because of off-chip delays in communicating between the two chips through driver, receiver pairs. It should be stated again; we were compelled to adopt this 70% gate utilization design rule to produce a low risk design because there is no computer aided design software available to the user for use with the GUA's.

A problem with the GUA itself concerns the use of the low Z drivers in the tri-state mode. In this mode there are two signal inputs to the low Z driver; one for the N device and one for the P device. Because of impedance mismatch an I/O all must be utilized to drive the low Z driver – therefore two I/O cells are required for each low Z driver operated in the tri-state mode. This cuts into the number of I/O cells available for input and output signals. Ideally, scalling cells for the low Z driver should be provided so that I/O cells are not required to perform the scaling function.

2.4 RAM Design Critique

The TCS-072, 256 Word by 4-bit, read-write random access memory (RAM) is an excellent choice as a RAM for the GPU SOS COS/MOS device family. This device is pin-for-pin and functionally compatible with the Intel 2101, AMD9101, AMS 7101 and RCA 5101. The TCS-091 has the following features:

- 256 x 4 organization meets the needs for small systems memories
- Single power form
- Tri-state outputs with TTL drive capability
- Static operation-no refresh clocks required
- Simple memory expansion provided with two chip selects

- Fast cycle time - 100ns typical @ 10V.
- Separate output disable provided for ease of use in common data bus systems.
- Data retention down to 1.5 volts for standby power applications.
- Low operating power - 20mW typical; and low standby power - 3mW typical.
- Standard 22-pin package.

The TCS-072 is well suited for smaller system memory applications such as read/write scratchpad, local CPU storage and cache memories. It is, however, inadequate for larger main memory applications. Therefore, another RAM device should be included in the GPU SOS COS/MOS device family to satisfy this need. An excellent candidate for this is a SOS COS/MOS RAM, pin-for-pin and functionally compatible with the Intel 2114. The Intel 2114 is a 1024 word by 4-bit Static RAM. It has the following features that make it well suited for larger main memory applications:

- 1024 x 4 organization satisfies need for large main memory applications.
- Single power supply.
- Tri-state outputs with TTL drive capability.
- Static operation - no refresh clocks required.
- High density 18-pin package.
- High speed - up to 200ns access time.

The viability of the GPU SOS COS/MOS device family would be greatly enhanced with this 4K RAM.

2.5 Proposed GPU SOS COS/MOS Device Family

In the previous sections, references were made concerning the expansion of the GPU SOS COS/MOS device family to include devices not currently in existence in order to increase its viability. This section summarizes our thoughts on additional parts that should be included in the device family. Figure lists the proposed additional part types (the entries with a star are devices yet to be added to the family tree). Some of the proposed new devices may already exist or there may be devices functionally equivalent, and therefore, can be used to fill the need. RCA should examine Figure and determine if they have fabricated/parts that may be functionally identical to the parts listed. This proposed list is the result of the 8080 Emulator breadboard design (which clearly exposed deficiencies in the family) and the emulation analysis study reported on earlier.

The requirements for a 4K ROM, 4K RAM, 182 gate GUA and 600 gate GUA have been presented in previous sections. The other parts listed are commonly found in computers across the entire applications spectrum. This is especially true of the proposed Bus Transceiver device; this type of device is required in each and every computer we've examined – from the lowly microprocessor to the super high power floating point processor. The Bus Transceiver device transforms a bi-directional bus into separate buses as illustrated below:

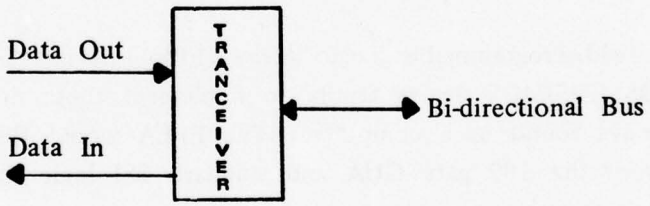


Figure 2.5-1 Bus Tranceiver

The 8080 Emulator design illustrates the need for a transceiver – a GUA is proposed that performs the functions of the above illustrated transceiver.

The Proposed Shift Network is a device required for high end floating point processors where the bulk of the processing task is centered on the

normalization of operands. In fact, without a shift network, floating point computation via hardware approaches the point of diminishing returns. The proposed shift network would; allow a shift of any amount to be made, be concatenateable for large word length machines, provide the logic for sign extension, rotate, arithmetic and logical shifts.

In addition to the Shift Network, a separate Vectored Priority Encoder is required for floating point processing. A Priority Encoder is required to provide the normalization shift count to the Shift Network. In addition, the Vectored Priority Encoder could be used to implement the priority interrupt structure of a computer. The Vectored Priority Encoder receives, as input, a data word and provides, as output, a binary code corresponding to the most significant bit position that is set. Ideally, for universal application, this device should contain a mask and enable register and it should be concatenateable.

The microprogram sequencer devices, by themselves, are not sufficient to implement the control section of a computer. In addition, flag buffering and multiplexing logic is required to control microprogram sequencing. This function would be handled by the proposed Microprogram Flag Handler device. This device performs the functions of buffering flags, selectively masking them, grouping them into branch flag sets and selecting a particular flag set to branch on.

A field Programmable Logic Array (FPLA) is included in the proposed GPU SOS COS/MOS device family to implement those miscellaneous logic functions always found in a computer. The FPLA would fill the gap that exists between the 182 gate GUA and standard SSI logic functions not implemented on saphire.

The last device is a Program Sequence Unit that implements the microprogram sequence control functions of a computer. It automatically generates addresses and updates the program counter for instruction fetch cycles, branch cycles, and subroutine call and return. It has built in condition code logic for program branching, a subroutine linkage stack and an adder for relative addressing.

Table 2.5-1 Proposed GPU SOS COS/MOS Device Family

	■ General Processor Unit (GPU)	TCS-074
	■ Read-Only-Memory (ROM) 1K	TCS-075
	■ State Universal Array (GUA) 300 gate	TCS-091
	■ 8 x 8 Multiplier with 2's complement	TCS-077
	■ Random Access Memory (RAM 1K)	TCS-072
	■ 2910 Controller	
☆	■ ROM 4K	
☆	■ RAM 4K	
☆	■ GUA 182 gate	TCS-090
☆	■ GUA 600 gate	TCS-093
☆	■ Bus Transceiver	
☆	■ Shift Network	
☆	■ Microprogram Flag Handles	
☆	■ Vectored Priority Encoder	
☆	■ FPLA	
☆	■ Program Sequence Unit	

☆ Not currently in device family

2.6 Radiation Hardening of CMOS/SOS Circuits

The following is a summary dealing with the members of the CMOS/SOS parts family to be developed and the radiation hardening considerations for these parts.

A description of some of these circuits follows:

2.6.1 TCS077 - Multiplier

The TCS077 array is an 8-bit expandable 2's complement type multiplier. It uses 2216 devices in a chip 181 mils x 175 mils. It multiplies two 8-bit operands (7 bits plus sign), a and b, and adds to this product an 8-bit word B. The a and b input operands are latched on the multiplier array to enable multiplication to proceed during periods when the input operands are no longer valid. The multiplier output is a 15-bit word (14 bits plus sign) which is represented algebraically as

$$C = a \times b + B$$

The chip has been built successfully in a 64 pin package using a non-hardened process and without the use of radiation hardening design rules. The layout rules in effect when it was first laid out called for channel lengths of 0.3 mils. The result is a circuit which performs an 8 x 8 multiplication in 280 ns with a dynamic power dissipation, measured at 10 volts V_{DD} and with an 18 pf. load, of 72 mw/MHz (this implies an effective chip capacitance for power dissipation, calculated from $P = CV^2f$, of

$$C_{EFF} = 720 \text{ pf}$$

2.6.1.1 Electrical Performance

RCA expects to use a somewhat shorter channel process under the AFML program than was used previously on this device (0.25 mils vs 0.3 mils). This should speed up the chip. Simulations have not been performed for L = 0.25 mils, but they have been done for L = 0.22 mils and for L = 0.20 mils. Chip speed vs L is shown in Table 3.

Table 2.6-1 Chip Speed vs. Channel Length for TCS077

Channel Length (mils)	Chip Speed (ns)
.3	280
.22	120
.20	105

Interpolation in this data for L = .25 mils gives a speed prediction of ~ 200 ns.

The present design of this device has no tri-state outputs. Redesign would be necessary to include these. Otherwise the present design appears acceptable (from an electrical performance viewpoint). It has been tested in a 16 x 16 multiplier and worked successfully (ro other chips are needed beyond four TCS077 chips).

2.6.1.2 Radiation Response

No radiation test data exists on the TCS077 multiplier. RCA bases their prediction that the device will meet the AFML spec of 5×10^4 rads(Si) total dose, and 1×10^9 rads(Si)/sec dose rate for logic upset on the measured performance of the GPU, along with improvements expected from the hardened process to be used [Boeing found that the GPU had a transient upset level of 6.4×10^9 rads(Si)/sec, and when built with a non-hardened process, had a total dose failure level of 5×10^3 rads(Si)]. It is not at all clear that this hardness level for the GPU means that the multiplier can meet the AFML goals. Simulation of this circuit, or of key portions of it, with a circuit simulation program would reduce the risk level substantially.

2.6.2 TCS072 - RAM

The TCS072 is a 256 word x 4 bit silicon gate CMOS/SOS Random Access Memory. It is fully static, and operates over the range from 3V to 15V. It is pin compatible with the Intel 2101, RCA 5101, and RCA 1822, and is packaged in a 22-pin DIP. Tri-state output drivers are provided for use in single or multiple bus systems. The memory output drivers are TTL compatible.

2.6.2.1 Electrical Performance

The TCS072 chip is 156 mils x 185 mils in size. It utilizes a five transistor memory cell, using a single N device as the X decode gate. Typical access time at 10V and driving an 85 pf. load is 110 ns. At 5V, the access time is 270 ns for the same load conditions. Average dynamic power dissipation at 10V is 6.5 mw/MHz. Leakage current at 10V is approximately 30μa (pre-rad).

2.6.2.2 Radiation Response

The TCS072 uses a 5 transistor memory cell (rather than a 6 transistor cell) for high bit density. 5T cells are more vulnerable to radiation, however, than the 6T cells. Back channel leakage in the X decode gate is a serious problem. Even with hard oxide processing, this device must be regarded as representing a high risk for achieving 50K rads(Si). A hardened version of the 1K RAM was presented as having a higher probability of meeting the AFML total dose goal. This device would employ a 6 transistor memory cell, with P-channel read and write line devices. Unfortunately, it is not planned to redesign the TCS072 to attempt to achieve these goals.

2.6.3 TCS091 - Gate Universal Array

The TCS091 Gate Universal Array is a 300 gate CMOS/SOS chip, 175 mils x 175 mils in size, which consists of a fixed placement of p devices, n devices, and tunnels in a repetitive ordered structure. All nodes of these devices (gates, drains, sources, and tunnel ends) are accessible, and all mask definition levels (except the metal mask) are fixed. The metal definition mask uniquely defines the device interconnect metal for each application.

The 300 gates are of the folloiwng types.

- 256 internal cells, consisting of two p and two n transistors comprising a two input gate.
- 24 I/O cells or 48 I/O devices. Each of the 48 pads has associated with it a gated diode protective circuit and an n and p device for use as an input inverter, output inverter, or transmission gate. The transfer characteristic of the inverter is close to 50% of the supply voltage, providing high noise immunity.
- 4 low Z cells for off chip drive.
- 8 high Z cells for active pull up.
- 8 custom binary cells connected to form two high speed binary dividers.

The sizes of these devices are as indicated in Table 2.6-2. This table brings out one of the shortcomings of GUA, viz. each gate of a given type has the same W/ℓ ratio. Palkuti and Pryor, in their paper on radiation hardened CMOS/SOS integrated circuits[1], point out that device geometries should be tailored to equalize rise and fall times. In this way one achieves optimum radiation hardness. Thus, cells designed with the Wm/Wp ratios given in Table 2.6-3 would be harder than the GUA cells since the GUA cells all have Wn/Wp ratios of 1.0.

Table 2.6-2 Device Sizes in TCS091 GUA

Cell Type	Device	L(mils)	W(mils)
Internal cells	p-device	0.25	1.9
	n-device	0.25	1.9
I/O cells	p-device	0.25	6.8
	n-device	0.25	4.3
Low Z cells	p-device	2.8	0.3
	n-device	5.3	0.3

Table 2.6-3 Optimum ratio of Wn/Wp

Cell Type	Wn/Wp
Inverter	0.7
2-NOR	0.3
3-NOR	0.2
2-NAND	1.4
3-NAND	2.0

RCA has considerable experience with GUA's. They have found that chip densities attainable with the three main approaches to LSI are as follows:

- GUA – 23 to 25 mil^2 per device
- Standard cell – 16 to 18 mil^2 per device
- Custom design – 14 mil^2 per device (with present SSTC design rules. Advanced design rules can give densities as high as 6 mil^2 per device.)

These densities would be affected by an unknown amount if p - island tie-downs were required as a radiation hardening ground rule. (Tie downs are not presently used on the TCS091)

Transmission gates are not used in internal gates in the TCS091, although the option does exist of using them in I/O cells (cell #2939T,2957T, and 2956T). It has been found that, while transmission gates require fewer devices than do alternate implementations using logic gates, the simple and efficient interconnection scheme possible with logic gates requires less chip area than transmission gates.(2)

For many potential applications, a 300 gate UA is not big enough to be useful, especially since once rarely uses more than 70 to 80% of the gates in any given application. For this reason, RCA was asked about their device TCS093, which is a 632 gate UA. This device has been designed, but never fabricated. They are presently designing a version of the 093 for an in-house user, and expect to begin fabrication in about six months. They expect to have data on the producibility of this large array in about nine months.

One of the advantages claimed for GUA's is that of quick turn around time. This advantage is not achieved in practice. RCA estimates five months as a reasonable time to go from logic inputs to chips out. If one used a standard cell approach to chip design, one could expect to get a design occupying less chip area in about the same time (5 months). This unexpected result comes about because of the fact

that GUA metal masks are laid out by hand (the software necessary to lay out GUA's with APAR is just now being developed by RCA under an IR and D program). Standard cell designs are laid out by machine, and can proceed faster.

2.6.3.1 Radiation Response

RCA has done radiation testing on 3 circuits built with 300 gate UA's processed with their so-called "radiation hardened oxide process". Results are presented in detail in reference 3, and summarized here. These circuits were built with ion implanted source and drain regions, dry oxide gate insulator, and N+ doped polysilicon gates (a somewhat different process than RCA expects to use on the AFML part family). They were found to have a total dose failure level which was dose rate dependent. Survival up to levels in excess of 1×10^6 rads(Si) was observed when irradiated at dose rates of 4×10^3 rads(Si)/hr. However, the degradation of prop. delay for exposure to these megarad doses will limit the safe design dose to $\leqslant 5 \times 10^5$ rads because of system speed requirements.[3]

2.6.4 Radiation Hardening of CMOS/SOS I.C.'s

2.6.4.1 Radiation Hardened Oxide Process

RCA proposes to spend the first nine months of the AFML program defining a radiation hardened oxide process. However, a preliminary outline of the process which will be used as a baseline is available. It is a variation of the double implant enhancement mode process referred to as the $2I^2$ (P/N) process. It uses 0.5 μm thick intrinsic silicon grown epitaxially on sapphire. By using separate implants for P and N islands, higher threshold voltages can be achieved than would be the case in a single implant process [I^2(N/N)]. This permits larger N-channel threshold voltage shifts before depletion mode operation results, thereby achieving greater hardness at the price of reduced pre-rad speed.

The gate dielectric of both the P-channel and the N-channel devices will be approximately 750 Å thick, and will be grown first in a pyrogenic (steam) environment, followed by dry oxygen. RCA would

not divulge the times and temperatures to be used, but indicated that low oxidation temperatures are required. The polysilicon film will be doped N+ during deposition to avoid exposing the gate oxide to a high temperature diffusion or to an ion implantation of the gate. Next, N+ and P+ implants for sources and drains are made. A thick Silox film (6000Å) is then deposited as a field oxide, and the implanted impurities activated at 850°C. Contact openings are then made, and aluminum interconnections are deposited by induction or filament heating Aℓ. Finally, another Silox film is deposited over the Aℓ, and bonding pads opened.

2.6.4.2 Radiation Hardened I.C. Design Rules

RCA did not use radiation hardened I.C. design rules in the design of any of the AFML parts. For these parts to achieve the 5×10^5 rads(Si) total dose level which RCA thinks their process is capable of, a number of such design rules would have to be imposed, and chip redesigns performed to implement them. The proposed design rules, and the impact of imposing them, are described below.

A. Proposed radiation hardened I.C. design rules:

- No transmission gates.
- Tie down all P islands not connected to V_{DD} with new low capacity tie down.
- No fan-in greater than 3.
- Equalize rise and fall times to the extent possible by geometry optimization.
- Design for 2.5V minimum internal noise immunity at 10V, 1×10^6 rads(Si).
- N island tie downs are not required.
- Use 6 transistor memory cell in RAM with P-channel read and write bus interface devices.

B. Impact of using these rad. hard design rules:

- All AFML chips would have to be redesigned.
- Chip area of all devices would increase.
- The multiplier (TCS077) would increase by about 15% in device count (along with the area required to accomodate these additional

devices), plus a 5% increase in for the devices already there. By a rough estimate, the chip would increase from 181 mils x 175 mils to 200 mils x 185 mils.

- The RAM might increase as much as 20% to 25%. This would result in a chip of the order of 190 mils x 200 mils (vs. 156 x 185 at present).
- A slight increase in size would result for the GUA.
- Since the GPU makes extensive use of transmission gates, its area increase would be large. (It is presently 215 x 201). There is a chance that it would be bigger than the 250 mils x 250 mils currently regarded as the maximum feasible chip size. However, RCA expects to accept delivery shortly on a new electron beam mask making facility which will remove this size limitation.
- The 2910 controller, to be designed and built under the AFML program is expected to be substantially bigger than the GPU, even without radiation hardened I.C. design rules. If such rules were imposed on this design, a chip would result larger than that permitted with present mask making facilities.
- Redesigns of these chips are expected to cost about $200K each.

3. GPU Testing Results and Methodology

3.1 Testing Results

This section itemizes the faults discovered during the course of Questron's testing of the GPU. It should be remembered that the functional characteristics of the GPU, as presented in this section, have been modified significantly for the reworked GPU. These modifications are reflected in the functional characterization of the GPU which is presented in Section 1.

3.1.1 SUMMARY

This section summarizes the results of Questron's Static and Dynamic Exerciser testing of the GPU which was completed September 1977. As such, it supercedes Questron's static testing final report entitled, "GPU Static Testing Final Report", dated 31 July 1977, and the preliminary report on dynamic testing entitled "Interim GPU Dynamic Fault Analysis", dated 1 September 1977. Dynamic testing uncovered one additional problem with the GPU – all other faults were discovered and isolated during Static testing. A top level throughput analysis was also performed on a sample of the devices in our possession. This report itemizes the faults discovered and presents isolation and diagnosis results and presents throughput analysis results. The intent of this report is to foster communication among concerned parties and get the GPU mended.

With GPU testing now complete, seven definite problems have been detected and isolated. Full diagnosis and even simulation is required (at least in one case) before recommendations on corrective action can be made. The faults fall into all four categories of fault types; functional design, logic design, electrical design and process related. They are explained in detail in the following sections and summarized as follows:

- Register file access problem.
- Simultaneous activation of FET's on P1B input bus.
- Output disable on direct input.
- R,T and S,A and M race conditions.
- Load clock, P2B loop condition.
- No overflow detection logic.
- Decoder Sneak Pulse.

As mentioned, Dynamic testing resulted in the discovery and isolation of only one additional fault–over static testing. However, it has supplied a wealth of additional information on the faults uncovered during static testing. Also, dynamic testing has revealed some puzzling performance characteristics in the GPU's sampled; namely they are two orders of magnitude slower at 5 volts than at 10 volts – details are presented in the following sections.

The complete Static Exerciser and Dynamic Exerciser test for the GPU consists of 207,962 test vectors which each contain 36 bits of stimulus for the GPU and 14 bits of expected response; this works out to over 10,000,000 bits of information. At least, twice that number of vectors were generated to perfect the tests and to perform fault isolation and diagnosis. The fault isolation and diagnosis effort consumed 3 man months of resources (the register file access fault itself required 1 man month to isolate).

All vectors were designed, generated and run on Questron's in-house LSI test facility. Details on this test facility and the testing approach are presented in a paper entitled, "Low Cost, Functional Approach to Microprocessor Testing", by V.V. Nickel and P.A. Rosenberg, JPL Publication 77-39, "Report on Phase II of the Microprocessor Seminar held at Caltech, April 1977,"

We would like to point out that, even though there are problems with the GPU, it is an extremely successful device. The problems isolated so far all seem to be solvable. We find it amazing that there are so few for the "first crack out of the barrel". This is indicative of the talent and dedication applied to support the development of the part. However, continued success for the part is assured only if this support is continued.

3.1.2 Fault Isolation and Diagnosis

The following sections present details on the faults successfully isolated and diagnosed during both Static and Dynamic testing of the GPU. However, diagnosis of some of the fault characteristics has been unsuccessful; i.e. there seems to be no valid explanation of the test results. There are two cases of incomplete diagnosis – the first is explained in the next section 3.1.2.1 and the explaination of the second is deferred until until Section 3.2. We would like to point out that there is an element of risk involved in reworking the GPU without complete understanding of some of the failure mechanisms. The phenomena observed are real. Test set up and procedures have been checked and rechecked so there is little possibility for error. Furthermore, the unexplainable fault characteristics exhibit strong correlation in date codes.

3.1.2.1 Register File Access Problem

The register file access anomaly was the first one detected and required the greatest effort to isolate because of its high degree of pattern dependence. Figure 3.1-3 illustrates the manifestation of the anomaly. What happens is that a bit or bits in a location accessed switches from a "0" state to a "1" state (no switching from a "1" to a "0" has been observed) as a result of the access only – no load clock (LC) is issued. The three specific conditions required to trigger the fault are as follows:

1. The R and T register file address fields are identical; i.e. the same register is accessed through both ports simultaneously: R=T=n.

2. Bits in both P1B and P2B must be set to the "1" state in the same relative bit position within both port buffers: $P1B_X=P2B_X$ = 1 where x indicates bit position. The failure is not triggered when only one bit is set.

3. And finally, a certain number of bits in the register file mus be set to the "1" state. These bits must all be aligned in the same bit position (column) as the bits set in P1B and P2B and the number must be above a certain threshold. If the number is less than this threshold the failure is not triggered: $M < P$ where P is the number of bits in the same column in the register file set to a "1" and M is the bit threshold number of the device.

The absence of any one of the three conditions will result in the part functioning correctly.

The bit threshold number(M), the number of bits in the same column in the register file that must be set to "1" in order to trigger the failure, varies greatly among parts with different date codes. This suggests a process related problem.

If the date codes correspond to separate runs, then there is significant evidence to support the diagnosis of the failure being process related. The bit threshold numbers at room temperature for the different date codes are summarized as follows:

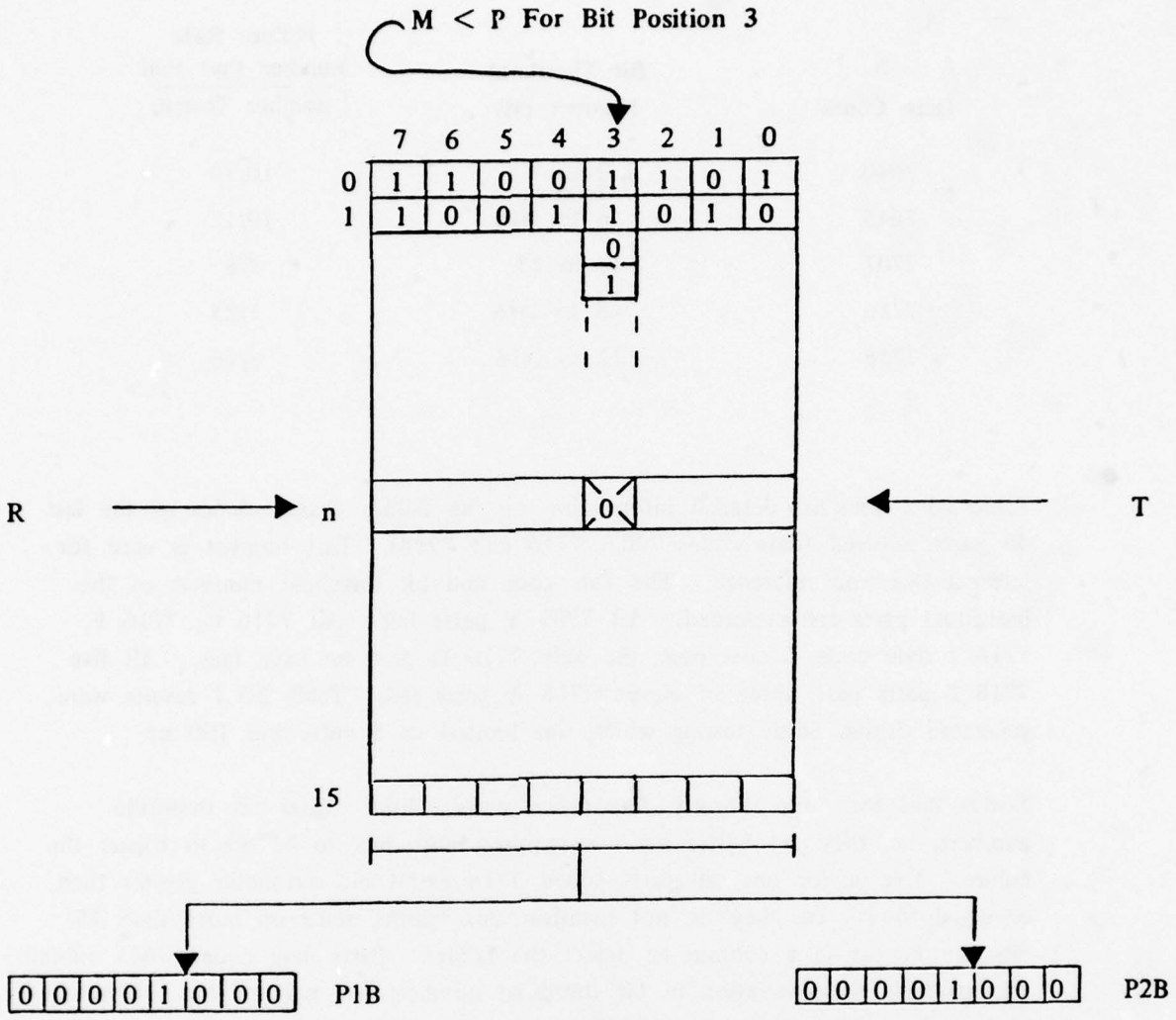


Figure 3.1-1 Register File Failure Symptom

Table 3.1-1 Register File Failure Data

Date Codes	Bit Threshold Number (M)	Failure Rate number that Fail / number Tested
7640	2 to 4	10/10
7645	6 to ⩾16	10/12
7707	11 to 13	8/8
7716	⩽8 to ⩾16	1/25
7718	12 to ⩾16	3/16

Table 3.1-2 presents detailed information on the failure characteristics of the last 49 parts received (date codes 7707, 7716 and 7718). Part number is used for internal Questron reference. The date code and bit threshold numbers of the individual parts are presented. All 7707 Y parts fail. All 7716 G, 7716 F, 7716 J date code devices pass; the only 7716 D part we have fails. All five 7718 J parts pass; three of eleven 7718 A parts fail. Table 3.1.1 results were generated during Static testing which was limited to 5 volts and 100 hz.

Notice that the more recently date coded parts exhibit higher bit threshold numbers; i.e. they get better since it requires more bits to be set to trigger the failure. Except for one, all parts coded 7716 exibit bit thresholds greater than or equal to 16, i.e. they do not manifest this failure, since no more than 15 bits can be set in a column to detect the failure. Parts date coded 7645 exhibit an extremely wide variation in bit threshold number; one part of the twelve sampled works. Parts date coded 7640 consistently exhibit very low bit threshold numbers; none of the ten tested work.

Bit threshold number is highly temperature dependent; it is inversly proportional to temperature; that is, it increases significantly with a decrease in temperature and conversely decreases with an increase in temperature. In fact, all parts date coded 7645 can be made to work if cooled sufficiently. None of the parts coded 7640 can be persuaded to work with cooling, although their bit threshold numbers increase significantly. Parts coded 7716 and 7718 have been heated, thus increasing their propensity to fail, but none of the good parts can be induced to fail.

Table 3.1-2 Register Access Failure Characteristics (Static Test Results)

Part Number	Date Code	Bit Threshold Number (1)
14	7707 Y	13
15	7707 Y	11
37	7707 Y	11
38	7707 Y	13
39	7707 Y	12
40	7707 Y	13
41	7707 Y	12
42	7707 Y	12
2	7716 D	$\leqslant 8$
23	7718 A	15
24	7718 A	12
27	7718 A	14

Figure 3.1-2 is a schematic of a single bit along the register file to port buffers data path bit slice. The output of the dual inverter regenerative circuit of the register file bit fans out to both port buffers. When R and T are equal both FET's are enabled. The figure illustrates what may be causing the failure. When both port buffer bits are charged to the "1" state, the charge may be feeding back through the FET's "overwhelming" the regenerative circuit when the bit threshold number is exceeded. The mechanism of the bit threshold number is extremely difficult to explain – *it may be excessive leakage.* This is one of the phenomena, discovered during static testing, that is difficult to explain, Dynamic testing has added to the confusion by presenting additional puzzling data on the register file (refer to Section 3.2.1.1)

As a result of Static testing, GPU's were classified as either good or bad based on the results of the register file tests. *However, dynamic testing has determined that all GPU's manifest the register file fault and are therefore bad.* When the voltage is increased (remember, static testing was limited to 5 volts and ~100Hz) these previously classified good parts began failing the tests. In conjunction with the voltage phenomena is a frequency dependence. This relationship is illustrated in Figures 3.1-3, 3.1-4, 3.1-5 and 3.1-6. In these figures voltage is plotted on the abscissa and the period (the inverse of frequency) on the ordinate. *The area above the curves is the failure region;* i.e. if the plot of voltage verses frequency, that the device is being run at, falls in this area the part fails. These graphs were generated by running the register file test at various voltage levels and frequencies; this register file test generates the three conditions (explained previously) required to trigger the failure. The curves illustrate that, in addition to those three conditions, V_{DD} must be set at a certain level and the period of time that the command (which generates those three conditions) is active must be long enough. The relationship between voltage and frequency is graphically displayed in the figures.

Figures 3.1-3, and 3.1-4 illustrate the failure characteristics of two devices that were previously classified as being good. The reason they were classified as being good parts in the static tests is obvious; the failure curve is asymptotic at 7 volts, so the failure region does not extend into voltages less than 7 volts. Since static testing was limited to 5 volts, excursions into the failure region could not be made for these parts. Figures 3.1-5 and 3.1-6 illustrate the failure characteristics of two parts that were classified as bad during static testing; one is asymptotic at 4 volts

the other at 3 volts. Because of the low asymptotic voltages, the failure regions for these last two parts is much larger than the previous two and thus detectable during static testing. *These curves suggest a serious leakage problem within the register file.* Throughput test results, presented in Section 3.2, also seem to indicate a serious leakage problem within the register file – *so serious, infact that, it appears* V_{SS}, *within the register file, is more than one volt higher than* V_{SS} *on the remainder of the part.*

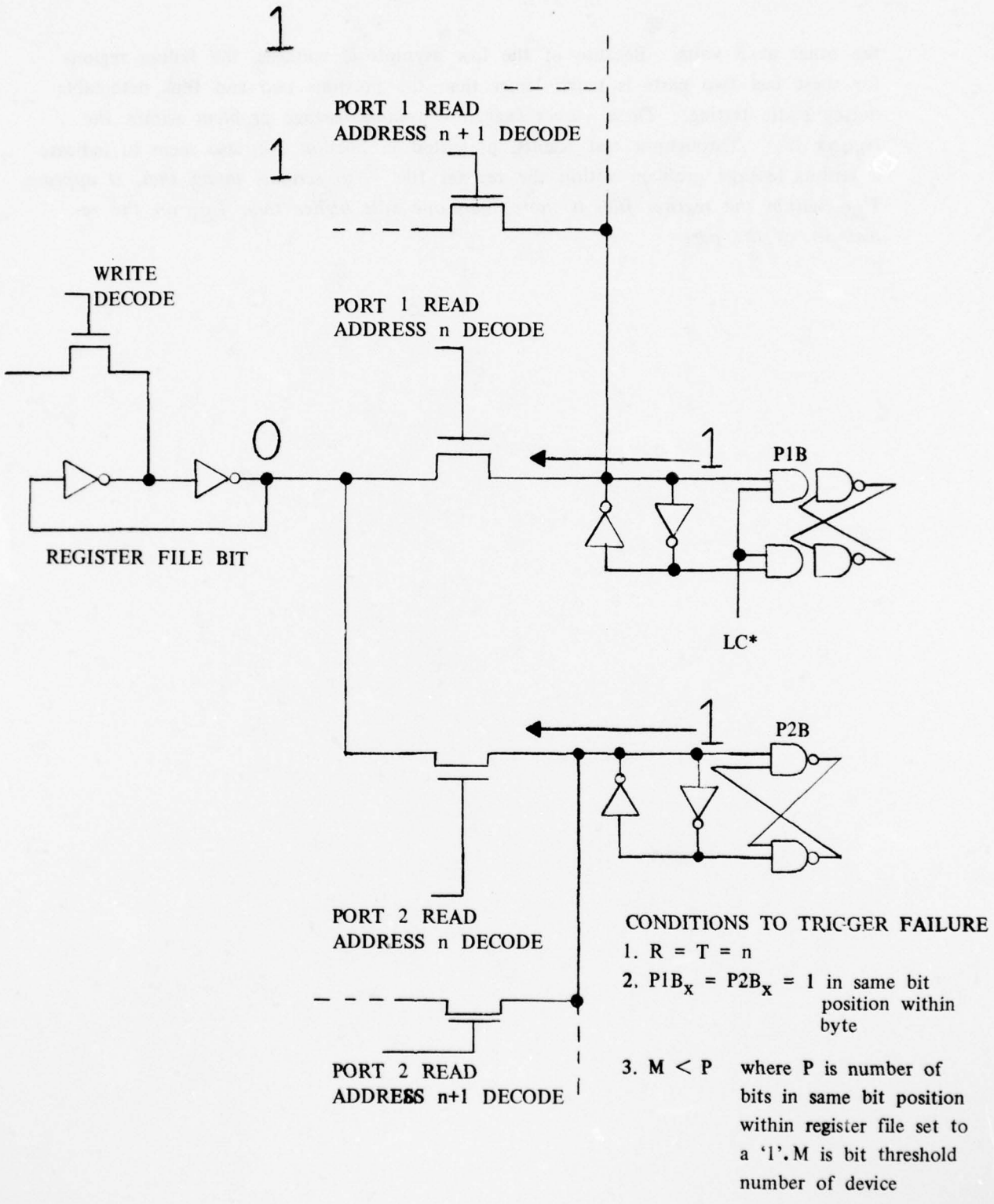


Figure 3.1-2 REGISTER FILE DUAL ACCESS BIT SLICE SCHEMATIC

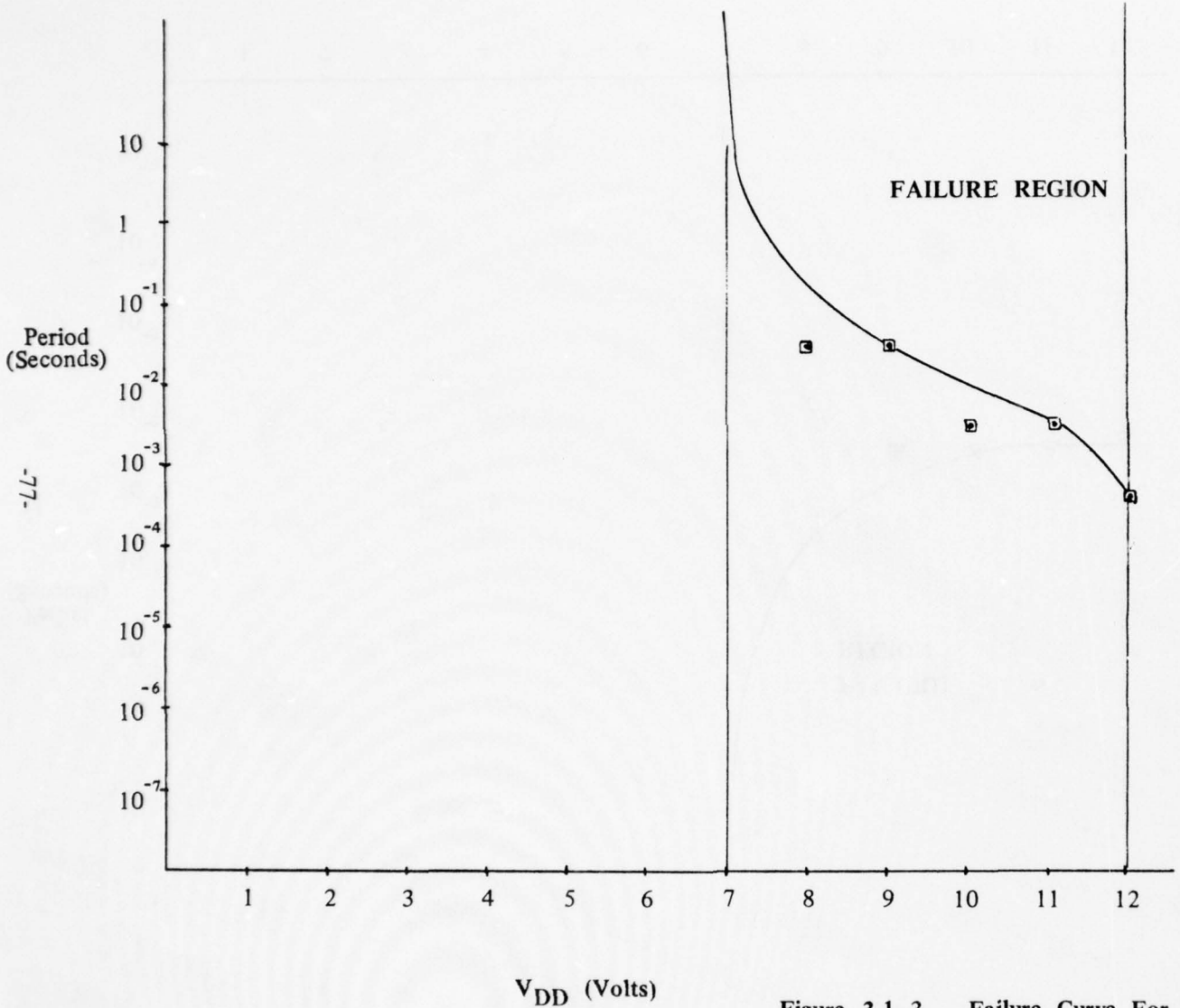
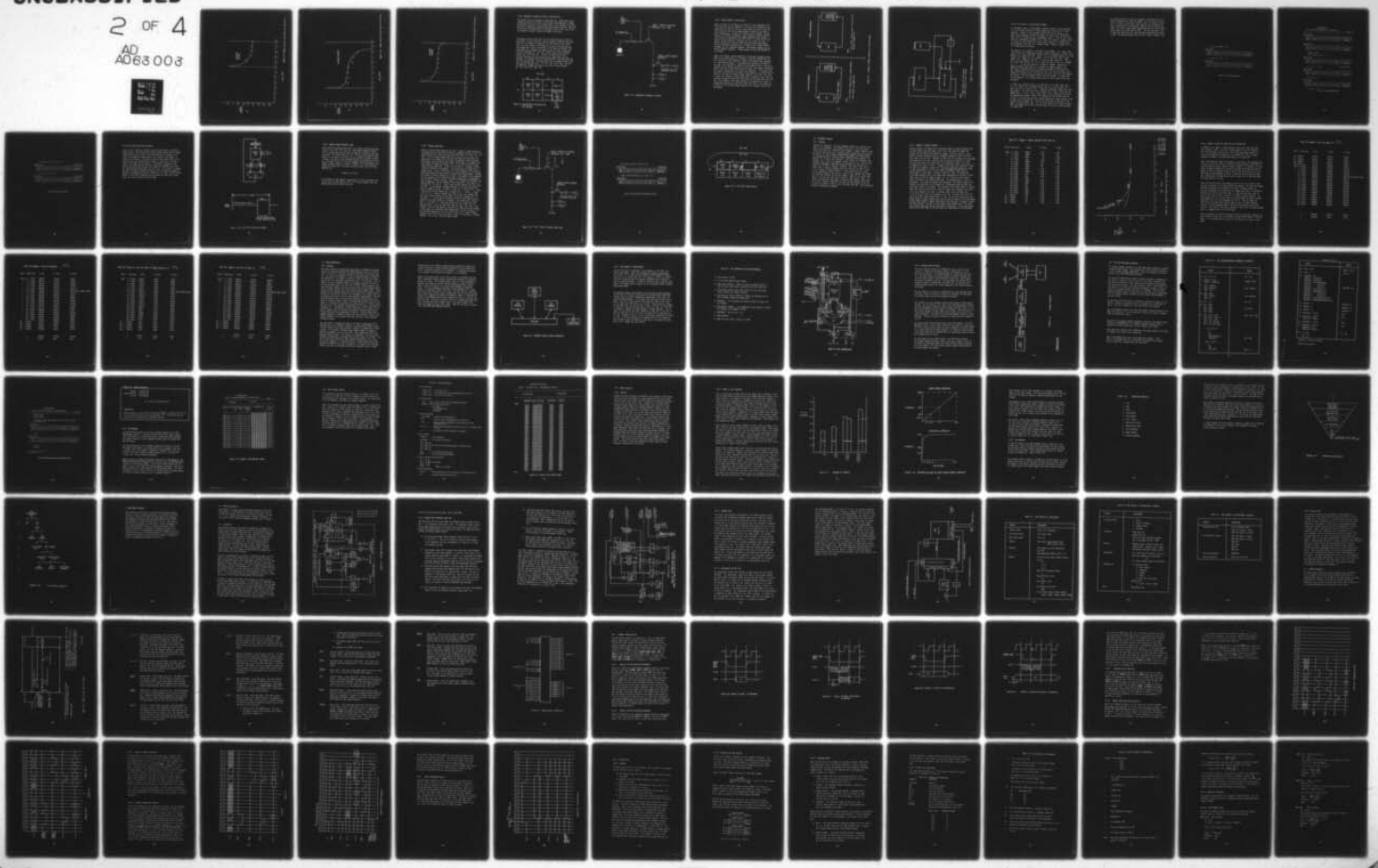


Figure 3.1–3 Failure Curve For Part AFAL No. 10 (7716 F)

AD-A063 003 QUESTRON CORP SAN DIEGO CALIF F/G 9/2
RADIATION HARDENED MICROPROCESSOR VOLUME I.(U)
MAY 78 V V NICKEL, P A ROSENBERG F33615-77-C-1001
UNCLASSIFIED AFAL-TR-78-55-VOL-1 NL

2 OF 4
AD A063003



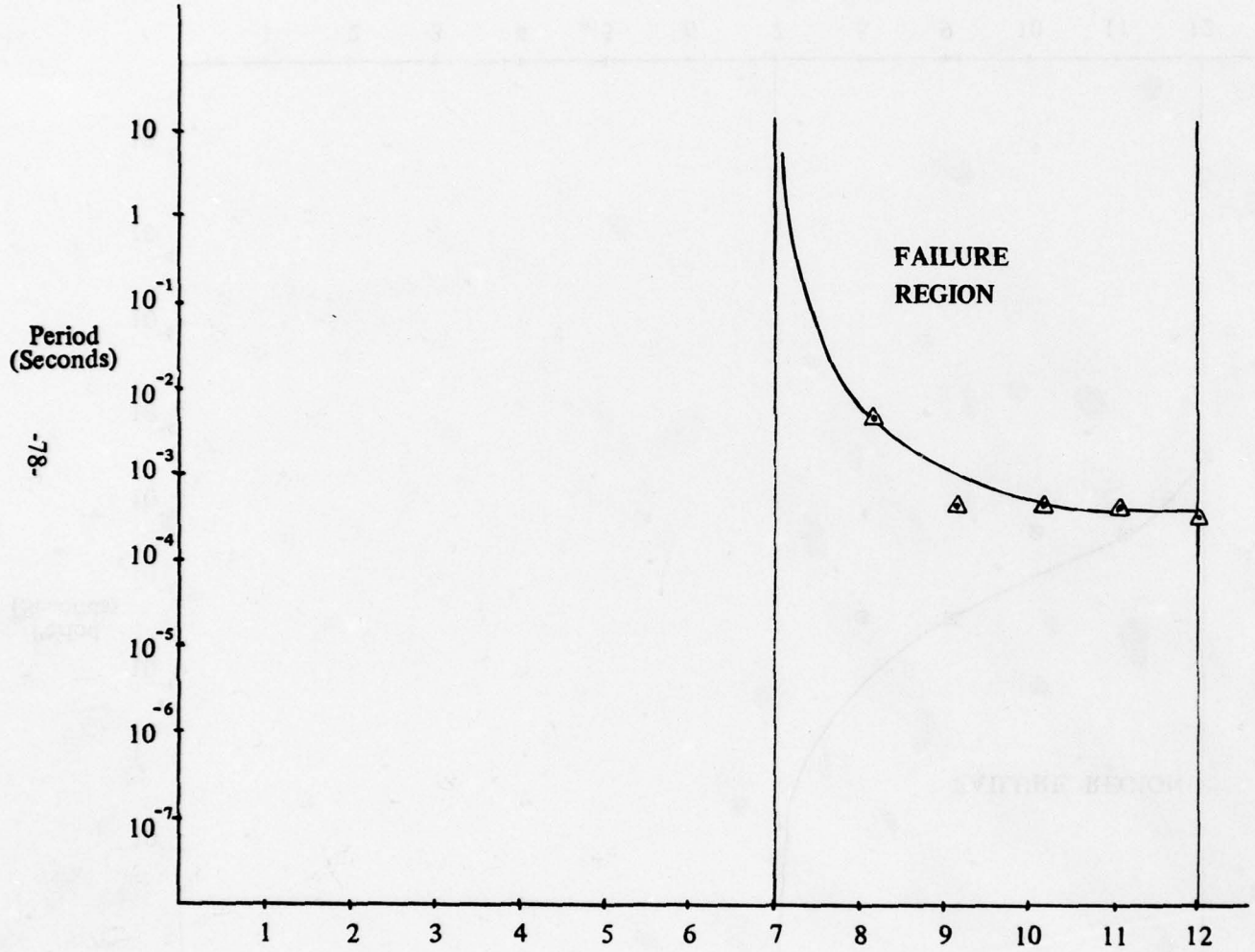


Figure 3.14 Failure Curve For Part AFAL No. 12(7716 E)

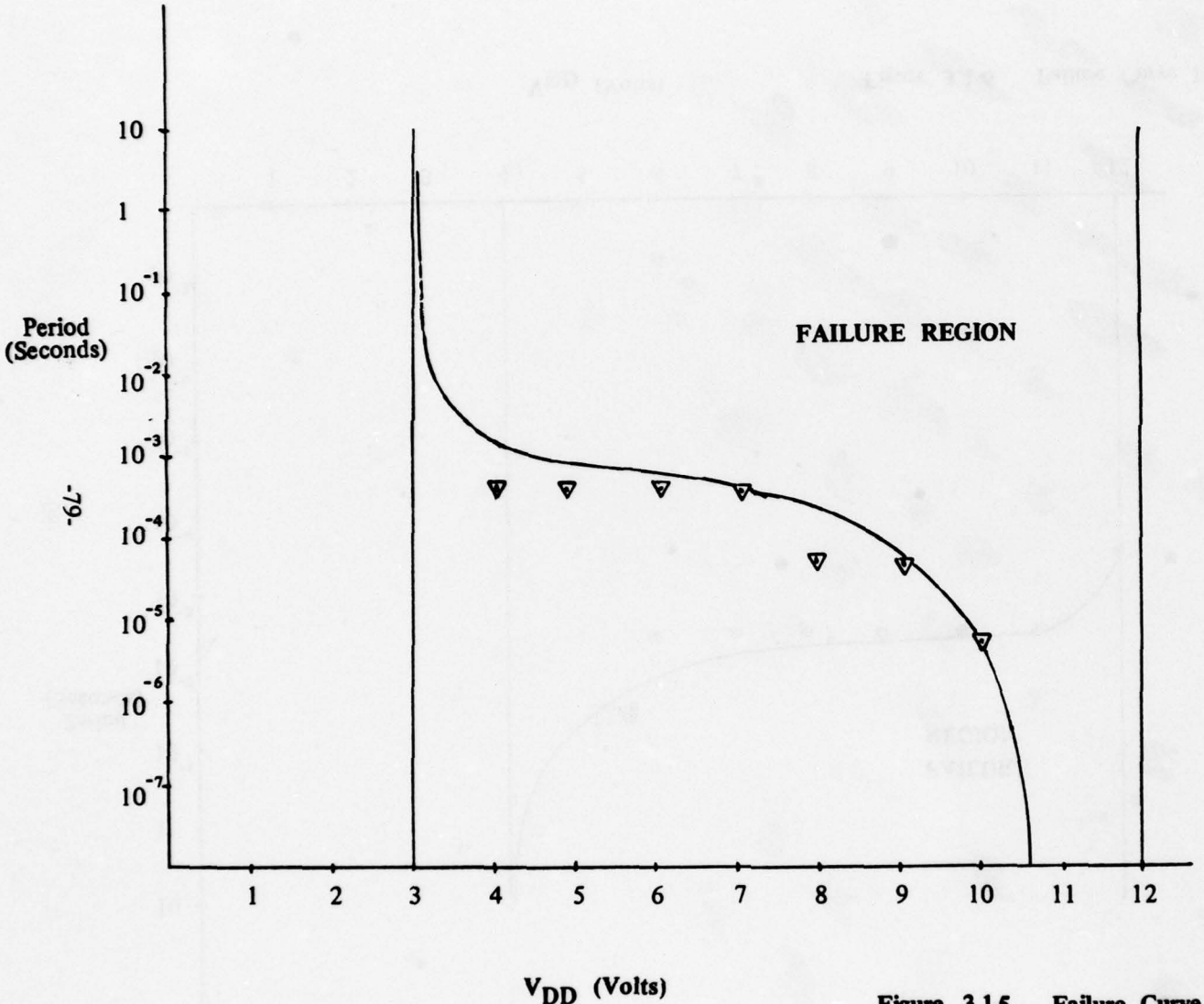


Figure 3.1-5 Failure Curve For Part AFAL No. 2(7716 D)

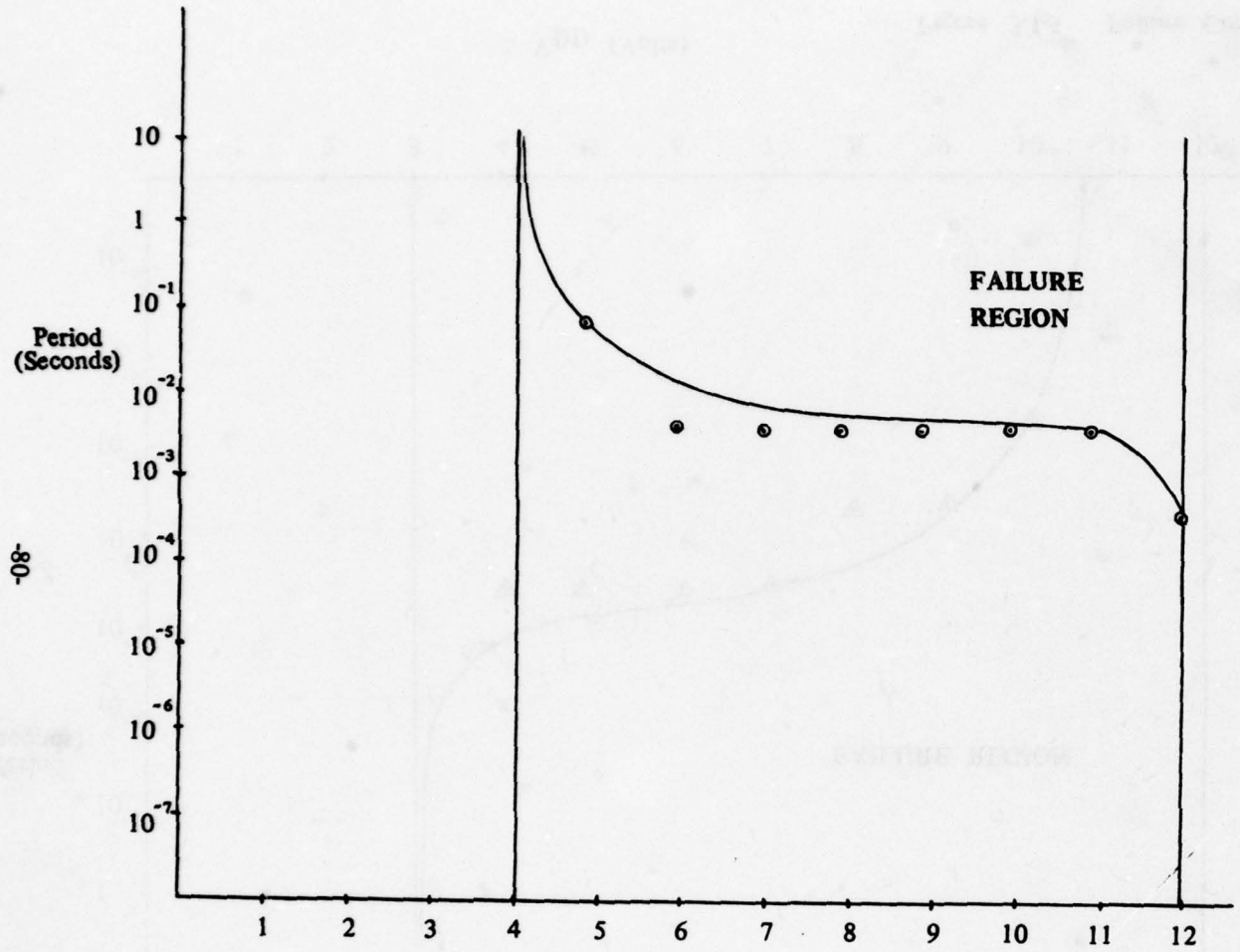


Figure 3.1-6 Failure Curve For Part AFAL No. 23(7718 A)

3.1.2.2 Simultaneous Activation of FET's on P1B Input Bus

This problem falls into the category of logic design error. Under certain control conditions two drivers are driving on the P1B input bus simultaneously. Testing has determined that there is no deleterious effect on the operation of the device except for a modicum of additional current drain during the operation. Also, since all data pthas are disabled during the operation, no data is lost. Even so the error should be corrected to insure the integrity of the device.

The problem occurs on direct input, from the Data Input pins, to P1B and then issuing the load clock (LC). Such an operation is performed when data input is to be operated on through the ALC then stored into the register file in one cycle. Figure 3.1-8 illustrates a bit slice of the P1B input bus. Each register file bit and the direct input bit are sources on this bus. The entire register file is disabled from driving on this bus, when a direct input to P1B is commanded; $\overline{S1} \cdot S0 \cdot \overline{LC}$. Notice, however, that the disable equation has a load clock term. Therefore the register file wil drive onto the P1B input bus when the load clock is raised (during the write operation). The problem is that the direct input FET is not turned off when the load clock goes high. The logic equation for the enabling of direct input data onto the P1B input bus is: $\overline{S1} \cdot S0 \cdot (M2 + M1 + M0)$. This can be remedied by including the load clock term into the enable equation: $\overline{S1} \cdot S0 \cdot (M2 + M1 + M0) \cdot \overline{LC}$.

SS1, AQ0

LC	00	01	11	10
0	Register File	Register File	DI	Direct In
1	Register File	Register File	Direct In	Direct In and Register File

LC=1
S=01
M=000

Figure 3.1-7 Karnaugh Map of P1B Input Bus FET Activation.

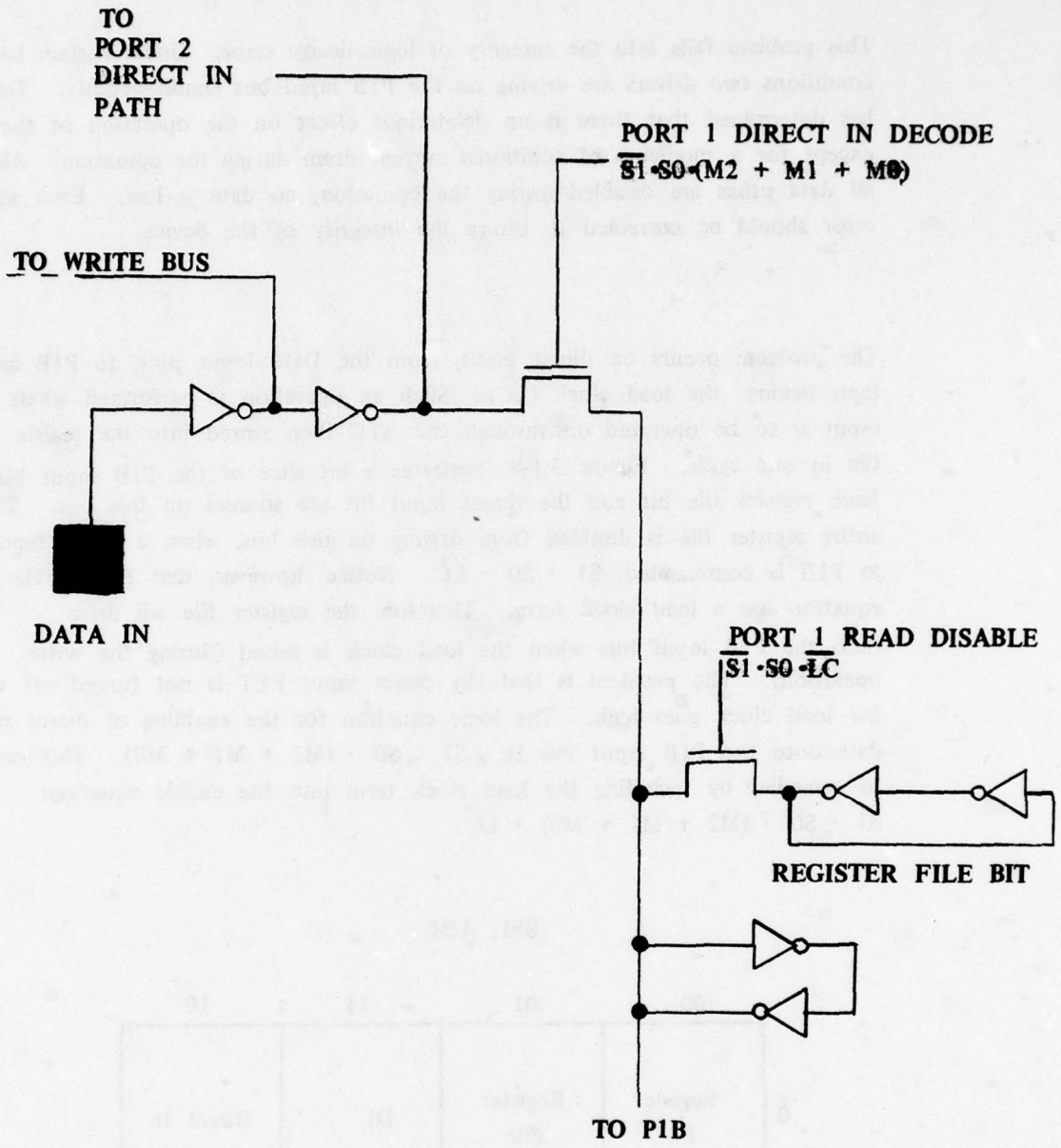


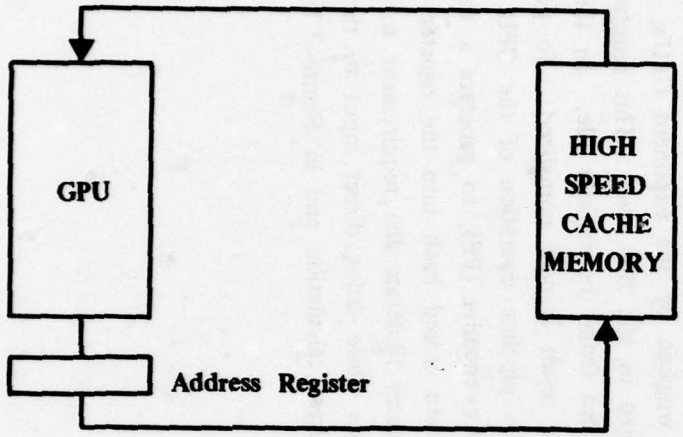
Figure 3.1-8 Simultaneous Activation of Drivers

3.1.2.3 Output Disable on Direct Input

Direct data input to the GPU can be directed to three destinations; P1B, P2B and directly into the register file. For direct input operations to the register file, the intent was that the output drivers be turned on with the ALC selected for output thus allowing pipeline operation of the GPU in system configurations. Currently the output drivers are disabled. Pipeline operation of the GPU has significant system throughput impact across the entire spectrum of applications; from 8080 Emulator to signal processor. It enables one step execution of operations that now require two steps. If pipeline operations are nested in critical processor control cycles (cycles that are continuously performed) such as instruction and operand fetching, throughput can be considerably enhanced. GPU pipeline capability also has inherent hardware advantages – holding registers for data and addresses are not required.

Figure 3.1-9 presents a simple illustration of the inherent advantages of pipelining in the control section of a processor; in this case operand fetch control. The figure is self explanatory. Figure 3.1-10 illustrates GPU pipelining in a floating point processor. The floating point operation is performed in two steps. The exponents are weighed by the exponent GPU's, a shift count generated and the mantissa aligned in the first step. This requires simultaneous direct in and direct output to and from the register file. In the second step the dyadic is performed and the result is post nomalized. To perform this operation, in one step, also requires pipeline operation of the GPU. The result is output, fed through a priority encoder (PE), to generate a shift count, and normalized through the shifter then stored back into the register file; all in one step. The examples in both figures illustrate the requirement to be able to simultaneously output ALC results while doing direct input to the register file; in Figure 3.1-9 for effective address calculation and in Figure 3.1-10 for post normalization.

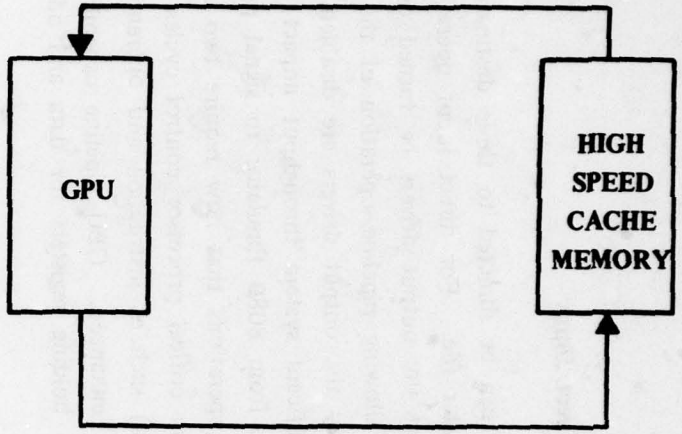
Current Implementation



Steps

1. Form effective address, output to Address Register.
2. Store operand into Register File.

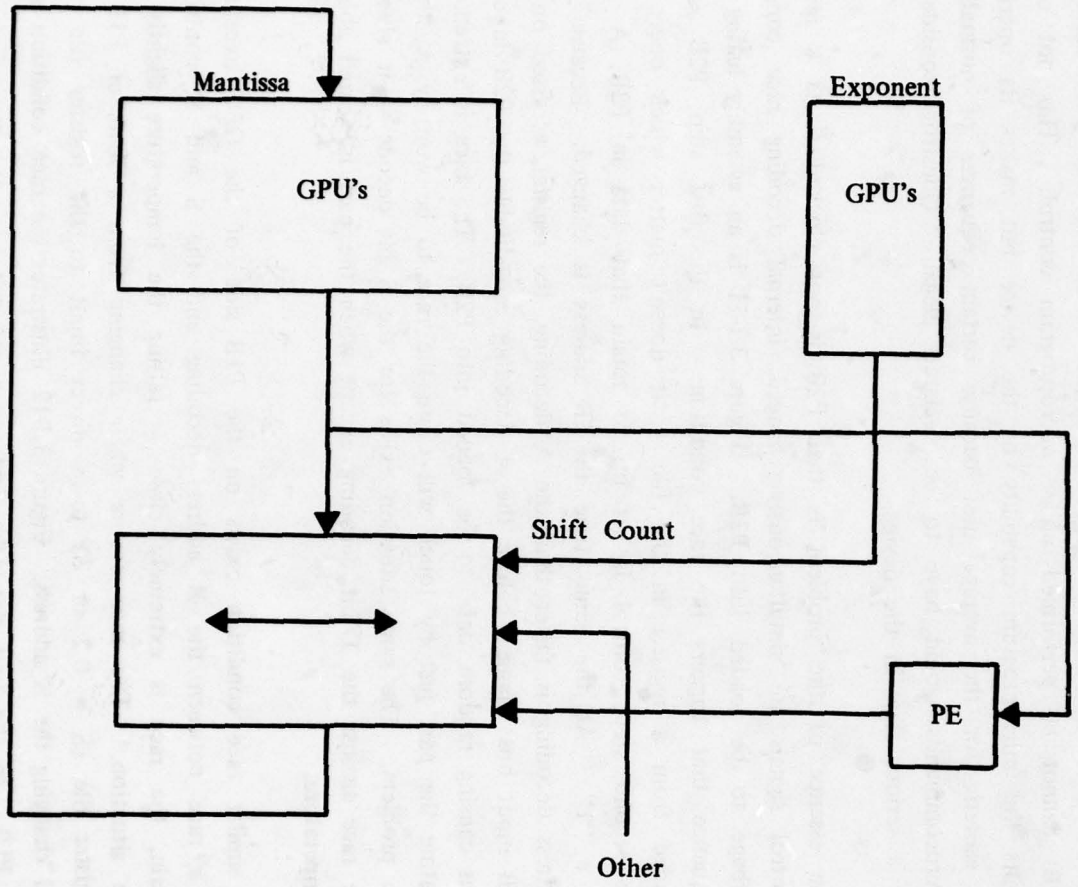
Pipeline Implementation



Steps

1. Form effective address, output to memory, store operand in Register File.

Figure 3.1-9 Example of Pipeline Operation in Control Processor



Steps

1. Weigh exponents and align Mantissa.
2. Perform dyadic and post normalize.

Figure 3.1-10 Pipelining in Floating Point Processor

3.1.2.4 R, T and S, A and M Race Condition

A fundamental aspect of microprogram controlled machines is the simultaneous (on a clock edge) switching of control signals with each fetch of a micro command. Because of race conditions between R,T and S,A and M on the GPU, certain operations between the register file and the port buffers (P1B, P2B) cannot be performed under microprogram control. This not only severly limits the microprogram capability of the device but makes its operation much to esoteric for the average user because certain sequences of normally valid microcommands would have to be declared illegal. Questron considers this to be a serious flaw in the device.

The essence of the problem is that P2B is not clocked. As a result, when control signals are simultaneously changed, internal decoding race conditions cause garbage to be loaded into P2B. Figure 3.1-11 is an assembly listing of a two step sequence that triggers the race condition. In the first step, P2B is being loaded from a register in the file – it doesn't matter which one. Then in the second step, control is set up to retain that data in P2B; A is changed to a "1". At the same time the T address is changed. Because the T address decoding is faster than the A decoding, the register accessed on the P2B input bus changes before the A decoding can disable the P2B input bus, thus causing random data to be loaded into P2B. The race is extremely close: heating the part just by touch will cause the race to be won by A, thus mending the problem. The same situation exists for the S bit decode – it always loses the race against the T bit decoding except when the part is heated above room temperature.

A similar race condition exists on the P1B side of the GPU except that it is a race between the R address decoding and the S and M control bits. Again, the race is extremely close – raising the temperature slightly will correct the situation. The race occurs when changing from a load of P1B from the register file (S = 0,2 or 3) to a direct input to the register file (S=1, M=0) and changing the R address. Figure 3.1-12 illustrates the race condition sequence on P1B. In the first step (the first command following the Repeat construct), P1B is loaded from the register file. Then in the second step a direct input to the register is command (M=0, S=1). At this point, because the R address decoding is faster, the original data in P1B is destroyed before the M and S control decoding can deselect P1B.

The proposed solution, to these race conditions, of delaying the R and T address decoding will not solve the problem: *the race condition also exists on the direct in (DI) path to P1B and P2B.* For example, a direct input to P2B followed by setting P2B = P2B will not be executed correctly if the direct in data changes simultaneously with the control inputs which attempt to set P2B equal to itself. The new data *races* in and is loaded into P2B before the control decoding can disconnect the direct in path. The identical situation exists on P1B. (Refer to Figure 3.1-13).

```
      DO = P2B, P2B = REG(X), LC = 01;

-------------
ADDR = 0012 !
-------------------------------------------------------------------------------
  ! MXH1 = 0 ! DI = 00  ! R = 0 ! S = 0 ! M = 6 ! CIN  = 0 ! A = 0 ! LC = 1 !
  -----------------------------------------------------------------------------
  ! MXH0 = 0 ! MXL0 = 0 ! T = X ! D = 7 ! C = 0 ! AZIN = 1 !   0000140077   !
  -----------------------------------------------------------------------------

      DO = P2B, P2B = P2B, LC = 01;

-------------
ADDR = 0013 !
-------------------------------------------------------------------------------
  ! MXH1 = 0 ! DI = 00  ! R = 0 ! S = 0 ! M = 6 ! CIN  = 0 ! A = 1 ! LC = 1 !
  -----------------------------------------------------------------------------
  ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   0000500077   !
  -----------------------------------------------------------------------------
```

Figure 3.1-11 P2B Race Condition Sequence

```
---------------
! ADDR = 0010 !
-------------------------------------------------------------------------------
   ! MXH1 = 0 ! DI = 00  ! R = 0 ! S = 2 ! M = 4 ! CIN  = 0 ! A = 0 ! LC = 1 !
   ---------------------------------------------------------------------------
   ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   0000120067   !
   ---------------------------------------------------------------------------


---------------
! ADDR = 0011 !
-------------------------------------------------------------------------------
   ! MXH1 = 0 ! DI = 00  ! R = 0 ! S = 1 ! M = 4 ! CIN  = 0 ! A = 1 ! LC = 1 !
   ---------------------------------------------------------------------------
   ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 7 ! AZIN = 1 !   0038510067   !
   ---------------------------------------------------------------------------

        REPEAT R = COUNT;

        DO = P1B, P1B = REG(X), LC = 01;

---------------
! ADDR = 0012 !
-------------------------------------------------------------------------------
   ! MXH1 = 0 ! DI = 00  ! R = X ! S = 0 ! M = 7 ! CIN  = 0 ! A = 0 ! LC = 1 !
   ---------------------------------------------------------------------------
   ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   000018007F   !
   ---------------------------------------------------------------------------

        M = 0, S = 1, LC = 01;                          * P1B RETAINS VALUE

---------------
! ADDR = 0013 !
-------------------------------------------------------------------------------
   ! MXH1 = 0 ! DI = 00  ! R = 0 ! S = 1 ! M = 0 ! CIN  = 0 ! A = 0 ! LC = 1 !
   ---------------------------------------------------------------------------
   ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   0000110047   !
   ---------------------------------------------------------------------------

        DO = P1B, LC = 10;

---------------
! ADDR = 0014 !
-------------------------------------------------------------------------------
   ! MXH1 = 0 ! DI = 00  ! R = 0 ! S = 0 ! M = 7 ! CIN  = 0 ! A = 0 ! LC = 2 !
   ---------------------------------------------------------------------------
   ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   00001000BF   !
   ---------------------------------------------------------------------------

        DO = P1B, LC = 01;
```

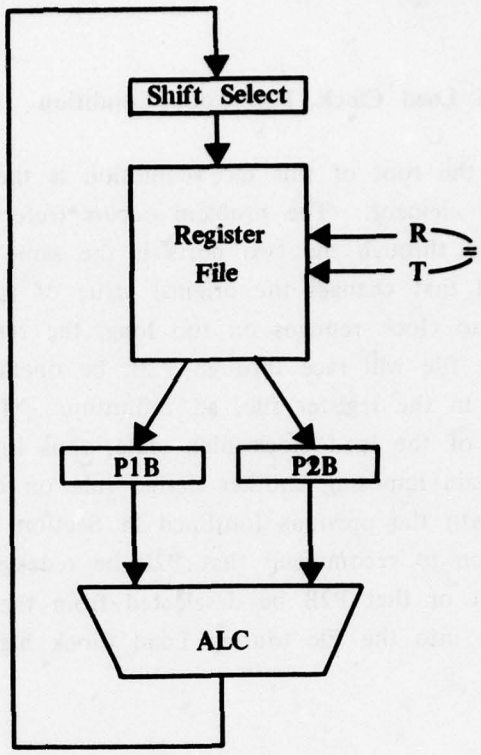
Figure 3.1-12 P1B Race Condition Sequence

```
DO = P1B, P1B = DI, DI = AA, LC = 01;

---------------                                                    ---------------
! ADDR = 0014 !                                                   ! BLOCK = 00 !
---------------------------------------------------------------------------------
   ! MXH1 = 0 ! DI = AA  ! R = 0 ! S = 1 ! M = 7 ! CIN  = 0 ! A = 0 ! LC = 1 !
   ---------------------------------------------------------------------------
   ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   AA0011007F   !
   ---------------------------------------------------------------------------

      DI = 00, M = 0, S= 1, LC = 01;

---------------                                                    ---------------
! ADDR = 0015 !                                                   ! BLOCK = 00 !
---------------------------------------------------------------------------------
   ! MXH1 = 0 ! DI = 00  ! R = 0 ! S = 1 ! M = 0 ! CIN  = 0 ! A = 0 ! LC = 1 !
   ---------------------------------------------------------------------------
   ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   0000110047   !
   ---------------------------------------------------------------------------
```

Figure 3.1-13 DI into P1B Race Condition

3.1.2.5 Load Clock, P2B Loop Condition

Again the root of this race condition is the fact that P2B is a non-clocked storage element. The problem occurs (refer to Figure 3.1-14 when the register accessed through the two ports is the same (R=T) and an operation is performed that changes the original value of the contents of the register. If the load clock remains on too long, the result of the operation stored in the register file will race through P2B, be operated on again and a new result stored in the register file, ad infinitum. This places a limitation on the length of the load clock high state, thus limiting the flexibility of the GPU and again imposing another design rule on the user. This problem, in conjunction with the previous (outlined in Section 3.1.2A), are significant enough for Questron to recommend that P2B be redesigned and be made a clocked storage element or that P2B be deselected from the register file while data is being written into the file (during Load Clock high).



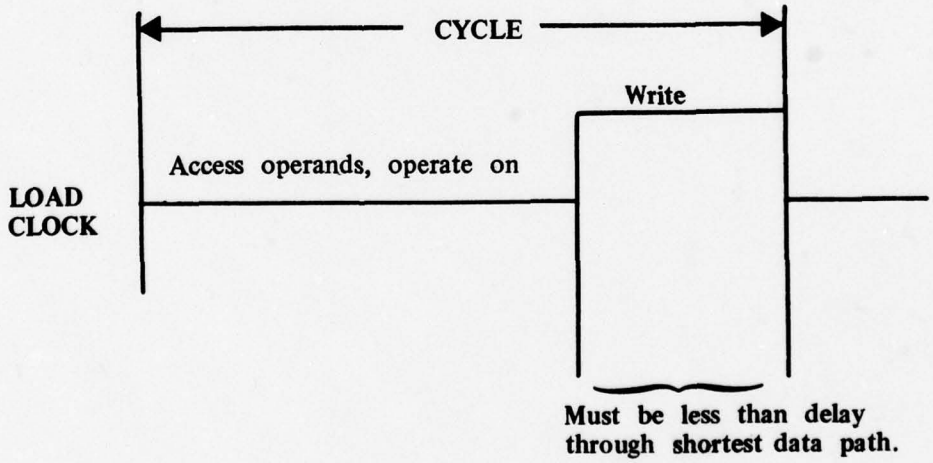


Figure 3.1-14 Load Clock, P2B Loop Condition

3.1.2.6 Missing Overflow Detection Logic

Static testing has determined that there is no overflow detection logic implemented on the GPU. The Tracor Inc. GPU design specification dictates that the overflow signal and the carry out signal are to share the same pin for output. For control decode $\overline{C2} \cdot \overline{C1} \cdot C0$ the overflow detection logic signal is to be output on the carry out pin. Overflow is defined as simply a change in the sign bit when performing addition or subtraction; for example, if the sign bit goes to a one state (negative) during the addition of two positive numbers, overflow as occurred. Overflow can be easily detected by taking the exclusive OR of the carry into and the carry out of the most significant bit:

$$\text{Overflow} \equiv C_7 \oplus C_6$$

On examining the logic diagrams it appears that, not only, the exclusive OR function but the control decode of $\overline{C2} \cdot \overline{C1} \cdot C0$ and overflow, carry out selection logic must be added to the GPU.

3.1.2.7 Decoder Sneak Pulse

There is a decoder timing problem in the GPU – namely, a decoder sneak pulse. The incident occurs on the P1B direct input, as illustrated in Figure 3.1-15 The decode function for a direct data input into P1B is: $\overline{S1} \cdot S0 \cdot (M2 + M1 + M0)$. If we are initially loading P1B from the register file then switch to a direct input to the register file in the next step, garbage is loaded into P1B (P1B should retain its previous value during a direct input to the register file). This sequence is illustrated in Figure 3.1-16 In the first step P1B is being loaded from the register file: S bits = 0 and M bits = 7; therefore, the Port 1 Direct In Decode is not generated. In the second step the register file is loaded directly: S bits 1 and M bits = 0. This combination of S and M bits should also prevent the Port I Direct In Decode from being generated. However, the contents of P1B are destroyed as a result of the operation. It is postulated that the transition in the S and M bits causes a sneak pulse to be generated from the Port 1 Direct In Decoder. Figure 3.1-17 represents, via Karnough map, what is happening. P1B is controlled by the decodes SS1 (S bits equal to 1) AO0 (M bits = 0) and LC. These control decodes determine the data sources into P1B. In the sequence described previously, we are first loading P1B from the register file; SS1 = 0, AO0 = 0 and LC = 0 and LC = 0. Then a direct input into the register file is executed; SS1 = 1, AO0 = 1, AO0 = 1 and LC = 0. Since LC remains zero there are only two possible paths that can be taken between the two states. The first is through SS1 = 0 and AO0 = 1. If this path is traversed P1B still is being loaded from the register file and there should be no problem. The second path is through SS1 = 1 and AO0 = 0. This path causes P1B to be loaded from the Data Input pins – it is this path that causes the fault. The cause of the problem is that the SS1 decode is faster than the AO0 decode; thus causing a sneak pulse to be generated while traversing path 2 (refer to Figure 3.1-17 *The interesting phenomena is that this fault occurs only at higher voltages;* typically if the GPU is run at less than or equal to 8 volts then it operates correctly. If run over 8 volts the failure occurs.

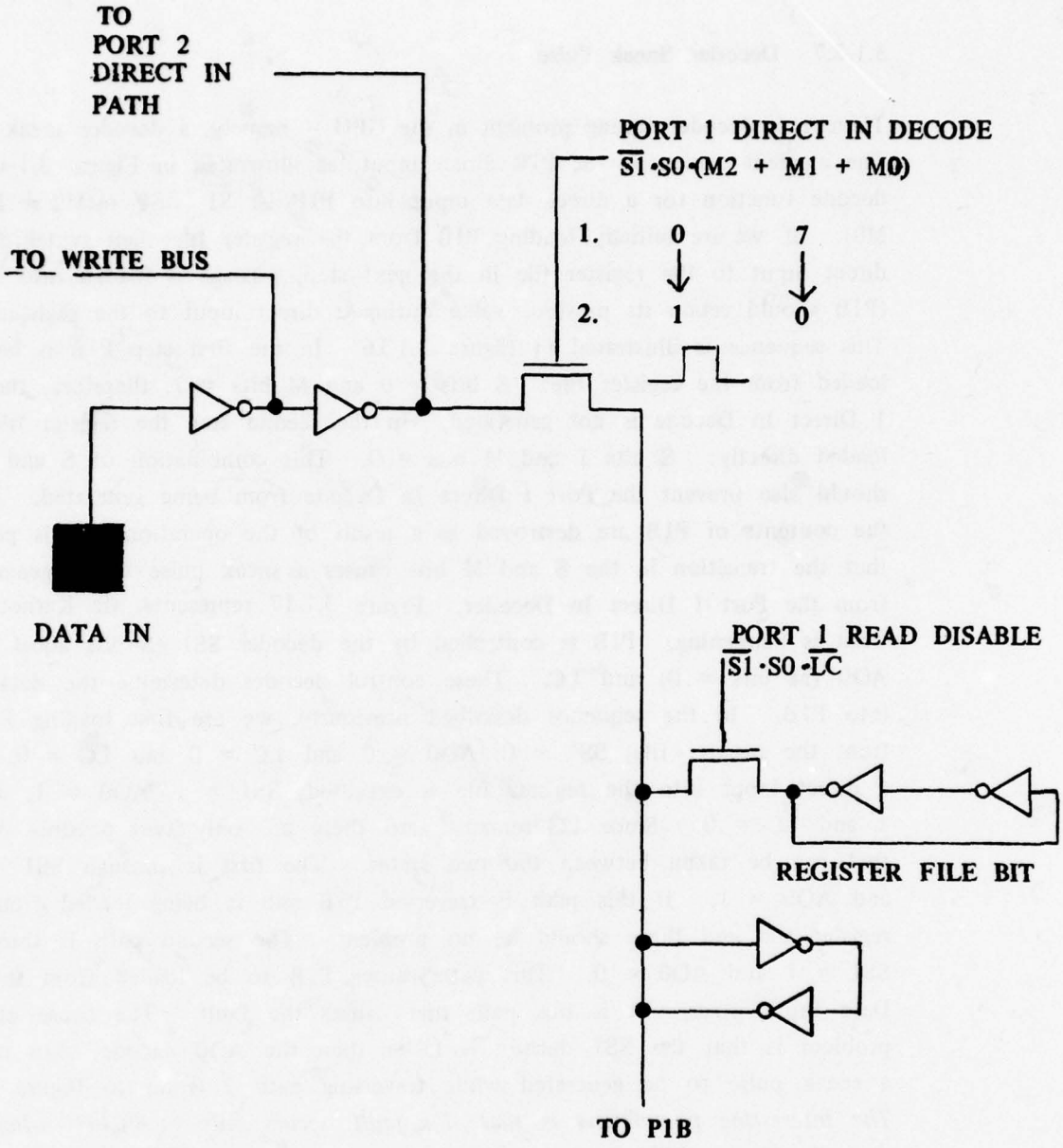


Figure 3.1-15 Port 1 Direct In Decode Sneak Pulse

```
        D0 = P1B, P1B = REG(X), DI = 00, LC = 01;

-------------                                                  -------------
! ADDR = 0014 !                                               ! BLOCK = 00 !
-------------------------------------------------------------------------------
   ! MXH1 = 0 ! DI = 00  ! R = X ! S = 0 ! M = 7 ! CIN  = 0 ! A = 0 ! LC = 1 !
   -------------------------------------------------------------------------
   ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   000010007F   !
   -------------------------------------------------------------------------

        REG(X) = SHIFTER, SHIFTER = DI, DI = 00, LC = 01;

-------------                                                  -------------
! ADDR = 0015 !                                               ! BLOCK = 00 !
-------------------------------------------------------------------------------
   ! MXH1 = 0 ! DI = 00  ! R = X ! S = 1 ! M = 0 ! CIN  = 0 ! A = 0 ! LC = 1 !
   -------------------------------------------------------------------------
   ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   0000110047   !
   -------------------------------------------------------------------------
```

Figure 3.1-16 Command Sequence That Triggers Decoder Sneak Pulse

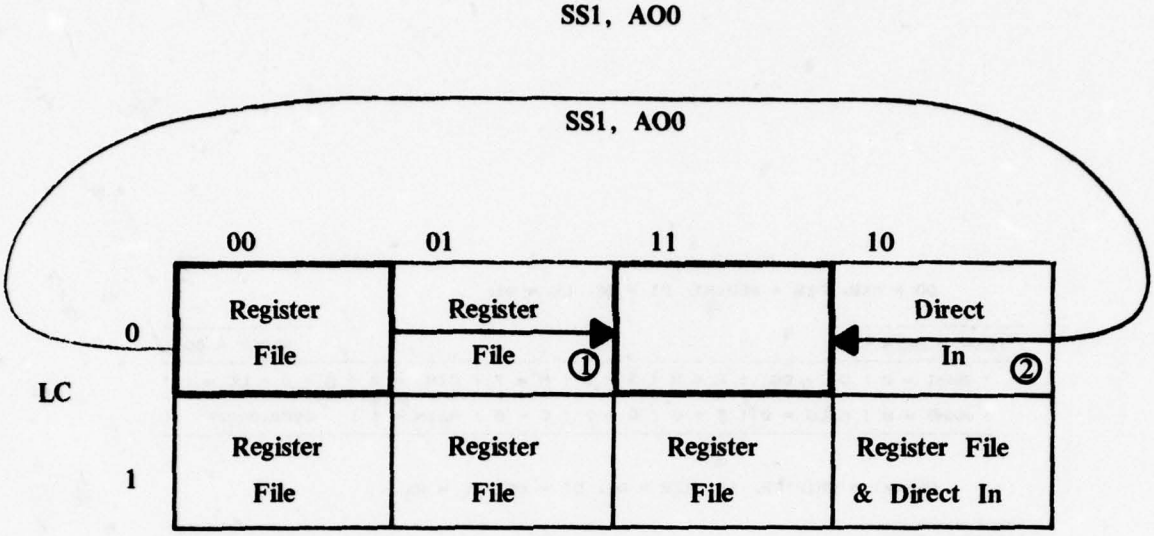


Figure 3.1-17 P1B Source State Function

3.2 Throughput Analysis

3.2.1 Summary

Questron has completed a top level throughput analysis of a sample of the GPU's in our possession. This analysis is top level in that only certain critical data paths were considered; namely, register to register, register to data out, carryout and all zero out. In general, results compare with those reported by RCA in the report entitled, "Manufacturing Methods for Silicon Devices on Insulating Substrates", Technical Report AFML-1R-516-3(F), January 1977. *However, the 49 parts tested exhibited a wide variation in performance characteristics; over 2 orders of magnitude (consult section 3.2.1).* Also, the RCA statement, "Preliminary characterization data indicates the cycle rate to be greater than 10 MHz is extremely misleading. The phrase, "cycle time" or "cycle rate" in computer language means more than just the time it takes to do a register to register operation – it also means the time it takes for all status signals to settle out as a result of the operation; these signals are, Carry Out, Overflow and All Zero Output. *Based on the correct definition of cycle time the GPU cannot operate at speeds above 6 MHz.* The register to register data path delay is extremely small typically around 80ns at 10 volts. However, coming off the part with data (register to data out or carry out) cannot be done at even twice that delay time. This phenomenon should be examined and corrected.

3.2.1.1 Register to Register Operation

Dynamic testing of the register file by performing register to register operations over the entire voltage range has yielded a clue to the cause behind the register file problem and the strange results reported in August. The clue is that at 5 volts register to register operations cannot be reasonably performed for 78% of the parts date coded in the year 1977. Referring to Table 3.2-1 this is evidenced by the unusally large delay times at 5 volts. *However, when the voltage is increased to 10 volts, the same parts exhibit the expected delay time characteristic which is two orders of magnitude less than at 5 volts.* A sample of four parts where examined in detail– the results are presented in Figure 3.2-1. This figure graphically illustrates the change in delay characteristics; in fact, the two orders of magnitude change occurs in approximately a 2 volt range – between 5 volts and 7 volts. The dashed line in the figure illustrates what is the normal delay characteristic for this type of device. Notice that not until the voltage is raised to 7 or 8 volts do the four devices begin to converge on the expected delay times. The measurements at 5 volts are approximate – the results vary greatly with temperature. This problem appears to be localized in the write path into the register file. Measurements taken on other data paths appear normal at all voltages. *The conclusion is that massive leakage is occurring within the register file and that Vss, within the register file, is floating 2 volts above Vss on the remainder of the chip, or convessly, VDD within the register file is 2 volts lower than VDD on the remainder of the chip. All parts tested with 1976 date codes exhibit normal characteristics at 5 volts.* At 10 volts all parts except one display register to register operation delay times of less than 100ns; in fact, the majority of parts exhibit register to register dealy times of less than 80ns. It should be noted that 80ns operation is the limit of our Dynamic Tester hardware; that is the reason why nothing is listed with less than 80ns delay in Table 3.2-1. At 13 volts *all* parts operate at less than 100ns; in fact, all except one exhibit delays of less than 80-ns.

Table 3.2-1 Register to Register Operation Delay Times (ns)

Part # / Date Code	*5 Volts*	*10 Volts*	*13 Volts*
AFAL 0 / 7716G	40000	< 80	< 80
3 / 7716G	10000	< 80	< 80
6 / 7716F	6000	93	< 80
9 / 7716F	1000	< 80	< 80
12/ 7716E	80000	< 80	< 80
15/ 7707Y	4000	200	92
17/ 7718J	15000	90	< 80
18/ 7718J	5000	< 80	< 80
21/ 7718J	10000	< 80	< 80
24/ 7718A	445	< 80	< 80
27/ 7718A	490	< 80	< 80
30/ 7718A	5000	95	< 80
33/ 7716J	7000	< 80	< 80
34/ 7716J	6000	< 80	< 80
39/ 7707Y	300	< 80	< 80
42/ 7707Y	330	< 80	< 80
45/ 7716G	8000	< 80	< 80
48/ 7716G	2000	< 80	< 80
Q1 / 7645AA	130	< 80	< 80
Q4 / 7645AA	190	< 80	< 80
Q7 / 7645AA	125	< 80	< 80
Q9 / 7645AA	175	< 80	< 80

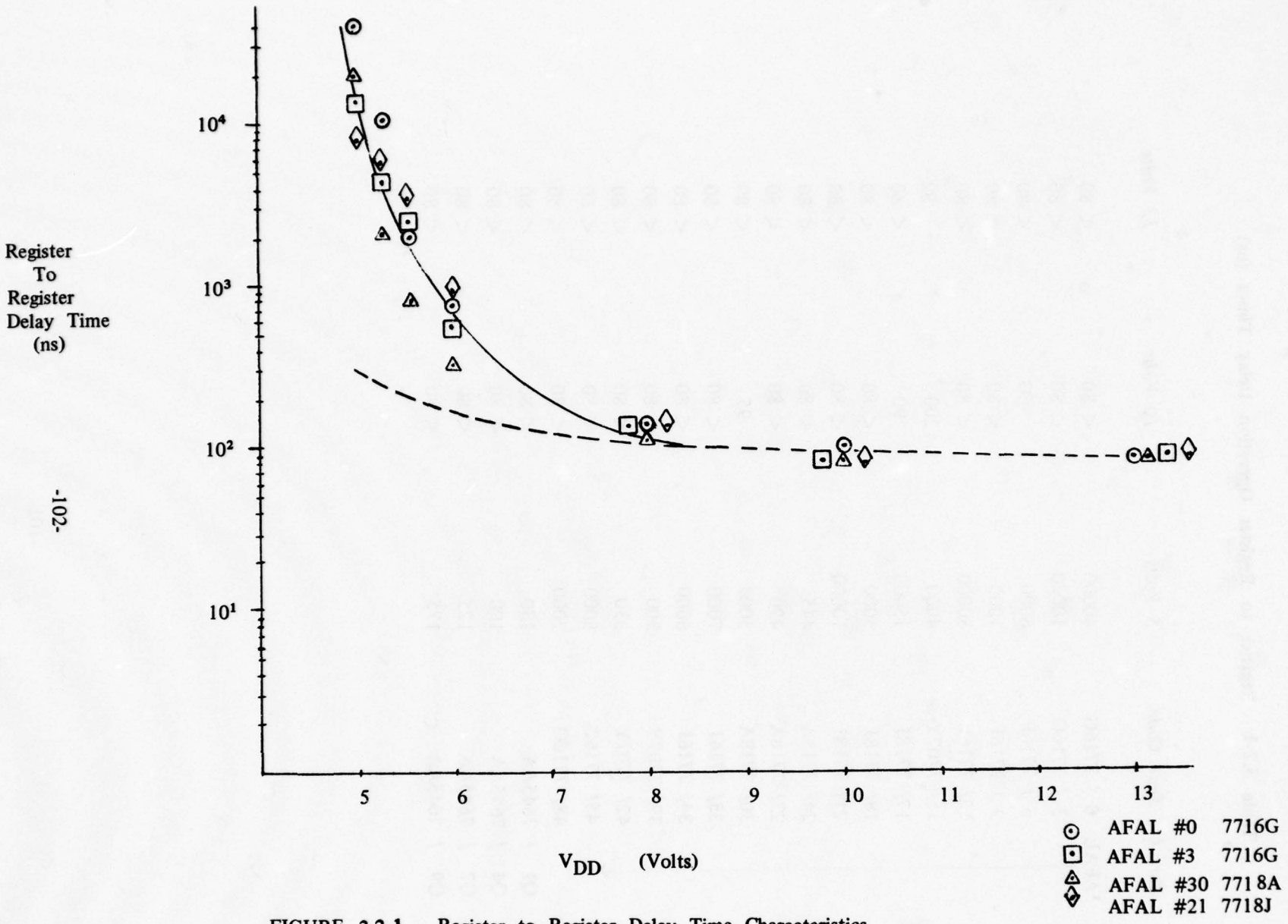


FIGURE 3.2-1 Register to Register Delay Time Characteristics

3.2.1.2 Register to Data Out, Carry Out, and All Zero Out

In addition to the register to register data path, three other critical data paths were examined in detail. They are as follows; register to data out, register to carry out, and register to all zero out. All of these paths appear normal at 5 volts, thus supporting the conclusion that the problem with register to register operations is localized in the write input region of the register file.

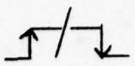
Table 3.2-2 presents the results of the register to data out test. *The significant result of this test is that the delay for coming off the chip with data is more than two times the delay for doing a register to register operation.* For example, at 10 volts register to register operations can be done in less than 80 ns while the mean for register to data out operations is 182 ns (zero to one) and 186 ns (one to zero). The reason for this large delay is unclear. In fact, the worst case carry out delay time is shorter than a straight data out (refer to table 3) at 10 volts and higher; the inverse should be the case.

Table 3.2-3 lists the results of the register to carry out test. For this test a worst case carry propagation for an ADD command was examined. The mean and standard deviation are presented at the bottom of the table. At higher voltages, the carry out propapation time is shorter than for straight data output. As mentioned previously, the reason for this is not clear. As with data out delays, carry out delay is more than two times the delay for a register to register operation. To help determine why the carry out delays are so long (relative to register to register operations) a best case carry propagation test was run. This test utilizes the *AND* and *OR* commands which directly toggle the carry out signal providing a measure of for the propagation delay of the last tier of gates that control the carry out. The results of this test are presented in Table 3.2-4 The figures illustrate there is a signigicant delay in the output stage.

Table 3.2-5 presents the results of the register to all zero output test. Delays are very close to those for carry out. As such, they are also less than register to data out delays. *It is recommended that the cause for the long data out delays be determined.*

Table 3.2-2 Register to Data Out Delays (ns) ↑/↓

Part # / Date Code	*5 Volts*	*10 Volts*	*13 Volts*
Q1 / 7645AA	260/235	150/110	130/80
Q4 / 7645AA	260/310	150/125	130/90
Q7 / 7645AA	225/165	130/90	120/75
Q9 / 7645AA	275/215	155/110	135/80
AFAL 0 / 7716G	225/525	165/200	140/150
3 / 7716G	235/485	160/195	140/145
9 / 7716F	285/530	190/200	165/155
12 / 7716E	220/465	170/190	140/140
15 / 7707	700/900	330/220	260/150 (Low Output Drive)
17 / 7718J	355/750	220/230	180/155
18 / 7718J	300/560	180/210	160/150
21 / 7718J	270/560	185/210	155/160
24 / 7718A	375/475	200/230	170/160
27 / 7718A	410/560	230/230	180/170
30 / 7718A	330/670	220/260	185/190
33 / 7716J	240/500	170/200	145/155
34 / 7716J	220/475	160/190	140/155
39 / 7707Y	350/420	180/175	155/130
42 / 7707Y	350/450	180/190	155/130
45 / 7716G	200/450	148/178	130/130
48 / 7716G	195/390	145/168	130/128
μ	299/480	182/186	155/137
σ	108/168	41/44	27/31

Table 3.2-3 Register to Carry Out Delays(ns) 

Part # / Date Code	*5 Volts*	*10 Volts*	*13 Volts*
AFAL 0 / 7716G	380/210	170/105	135/83
3 / 7716G	380/215	170/105	135/85
6 / 7716F	600/320	210/170	165/150
9 / 7716F	410/270	180/120	140/120
12 / 7716E	340/200	165/135	130/115
15 / 7707Y	620/350	250/220	190/175 (Low Output Drive)
17 / 7718J	560/290	210/265	165/135
18 / 7718J	430/245	180/120	140/95
21 / 7718J	430/250	175/120	140/90
24 / 7718A	380/235	180/165	150/135
27 / 7718A	440/270	195/165	150/140
30 / 7718A	520/275	220/170	175/145
33 / 7716J	375/215	165/110	125/85
34 / 7716J	360/200	160/105	125/85
39 / 7707Y	330/235	160/160	130/130
42 / 7707Y	330/215	165/160	135/130
45 / 7716G	320/185	148/100	120/80
48 / 7716G	310/180	145/100	120/80
Q1 / 7645AA	225/215	125/100	110/70
Q4 / 7645AA	265/165	135/90	120/75
Q7 / 7645AA	180/180	115/70	105/60
Q9 / 7645AA	235/250	135/90	115/70
μ	383/235	171/134	137/106
σ	112/46	32/45	22/32

Table 3.2-4 Register to Carry Out Delay for Logical Operations (ns)

Part #	/ Date Code	*5 Volts*	*10 Volts*	*13 Volts*	
AFAL	0 / 7716G	100/90	58/38	45/30	
	3 / 7716G	100/85	58/38	48/30	
	6 / 7716F	152/130	70/48	58/35	
	9 / 7716F	115/95	62/40	55/32	
	12 / 7716E	100/75	55/35	50/30	
	15 / 7707Y	200/130	90/48	70/38	(Low Output Drive)
	17 / 7718J	150/125	70/45	60/35	
	18 / 7718J	120/100	65/40	55/30	
	21 / 7718J	110/90	65/40	55/30	
	24 / 7716A	110/95	65/43	58/38	
	27 / 7718A	130/115	70/48	60/38	
	30 / 7718A	150/125	72/50	60/38	
	33 / 7716J	100/85	55/35	45/30	
	34 / 7716J	95/85	58/35	48/30	
	39 / 7707Y	100/85	58/38	50/30	
	42 / 7707Y	105/80	62/40	55/35	
	45 / 7716G	85/78	50/30	45/28	
	48 / 7716G	85/78	55/35	48/30	
Q1 /	7645AA	75/55	50/30	40/25	
Q4 /	7645AA	75/65	50/30	43/27	
Q7 /	7645AA	70/45	45/25	42/23	
Q9 /	7645AA	75/60	48/28	45/20	
	μ	109/90	61/38	52/31	
	σ	31/23	10/7	7/5	

Table 3.2-5 Register to All Zero Out Delay (ns)

Part # / Date Code	*5 Volts*	*10 Volts*	*13 Volts*
AFAL 0 / 7716G	300/230	160/125	140/100
3 / 7716G	320/225	170/115	140/95
6 / 7716F	460/350	215/140	180/115
9 / 7716F	350/265	190/128	170/100
12 / 7716F	300/215	170/118	148/98
15 / 7707Y	520/325	238/160	200/128 (Low Output Drive)
17 / 7718J	460/330	220/150	185/118
18 / 7718J	350/270	178/135	150/105
21 / 7718J	325/235	173/120	150/100
24 / 7718A	370/255	205/135	183/110
27 / 7718A	390/285	210/140	180/115
30 / 7718A	420/320	210/150	185/120
33 / 7716J	293/225	160/115	140/93
34 / 7716J	290/220	160/115	135/90
39 / 7707Y	305/225	170/120	155/100
42 / 7716G	305/225	170/118	150/100
45 / 7716G	265/200	145/110	135/93
48 / 7716G	270/185	150/105	135/85
Q1 / 7645AA	250/165	153/93	138/75
Q4 / 7645AA	240/175	150/98	135/80
Q7 / 7645AA	230/135	148/85	135/70
Q9 / 7645AA	290/165	180/95	160/80
μ	332/238	178/121	156/99
σ	75/57	26/19	18/15

3.3 Testing Methodology

3.3.1 Summary

This section outlines the methodology being utilized to establish the functional integrity of the General Processor Unit (GPU) fabricated by RCA for the Air Force. The GPU is a new 8-bit processor bit slice with 16 dual access registers. The concept and utilization of low-cost functional testing, as outlined in this presentation, is not limited to processor bit slice LSI types, but is applicable to any new, functionally complex device that is to be verified. The key element in low-cost functional testing is the automatic generation of test vectors and functional testing of an LSI device "off-line" from an expensive, full-blown LSI tester utilizing a low-cost test stand which has been designed specifically to facilitate automatic test vector generation and functional testing. Vectors required for AC parametric testing and eventual production line testing are also generated on this low-cost test stand, then transferred to the expensive LSI tester or production line tester to do the actual parametric characterization. This off-loads the expensive tester of all test vector generation functions, leaving only actual AC parametric testing. With the extremely high gate-to-pin ratios of LSI devices, it is functional test generation that consumes most of a tester's resources, while the cost of AC parametric characterization of LSI devices is becoming proportionally less. Unfortunately, in general LSI testers, on the market, do not provide adequate tools for functional test generation; therefore, generating and running functional tests and generating AC tests on a low-cost facility which is designed specifically to facilitate automatic test vector generation, results in more efficient utilization of the expensive LSI tester, and thus, a more economical verification program.

The total number of possible test states for the GPU is astronomical (2^{180}); therefore, the tests are designed according to a theoretical fault model to limit the total number of test vectors required to achieve an acceptable level of fault coverage. A "functionally" modified nearest neighbor model was utilized to guide the development of tests for the GPU. Over 100,000 test vectors were generated using this model. The structure and organization of the tests is also important in approaching an acceptable level of fault coverage for the GPU. Not only are GPU tests structured to verify individual sub-functions on the device, but are also organized in a hierarchial structure in which previously tested sub-functions are utilized to test other sub-functions. The design of this hierarchial structure, such that all sub-function interfaces are adequately tested, is crucial.

Relying solely on the commonly expoused heuristic argument that good fault coverage is achieved by testing individual sub-functions can have disastrous results if sub-function interfaces are not considered. A hierarchial test structure is also required if fault isolation and diagnosis is to be attempted.

Because of the large number of test vectors required to functionally verify the GPU, a sophisticated support software capability was developed to automatically generate test vectors. This support software consists of a GPU Microprogram Assembler, GPU Simulator and GPU Exerciser (Xrcisr)—all written in PLM (a dialect of PL/1) and resident on an in-house Intel MDS-800 microcomputer system. The GPU Microporgram Assembler and GPU SImulator work in conjunction to automatically generate test vectors. The GPU Xrcisr is a powerful, CRT console based, interactive support package which runs the tests and provides features which facilitate fault isolation and diagnosis. Figure 3.3-1 illustrates the structure of the MDS-800 microcomputer system which serves as a low-cost functional test stand. The GPU is connected directly to a parallel input/output port and is exercised under direct program control. The system includes a line printer, dual floppy disk and a CRT console. There are 65K bytes of memory in the Intel 8080 based MDS-800 mainframe.

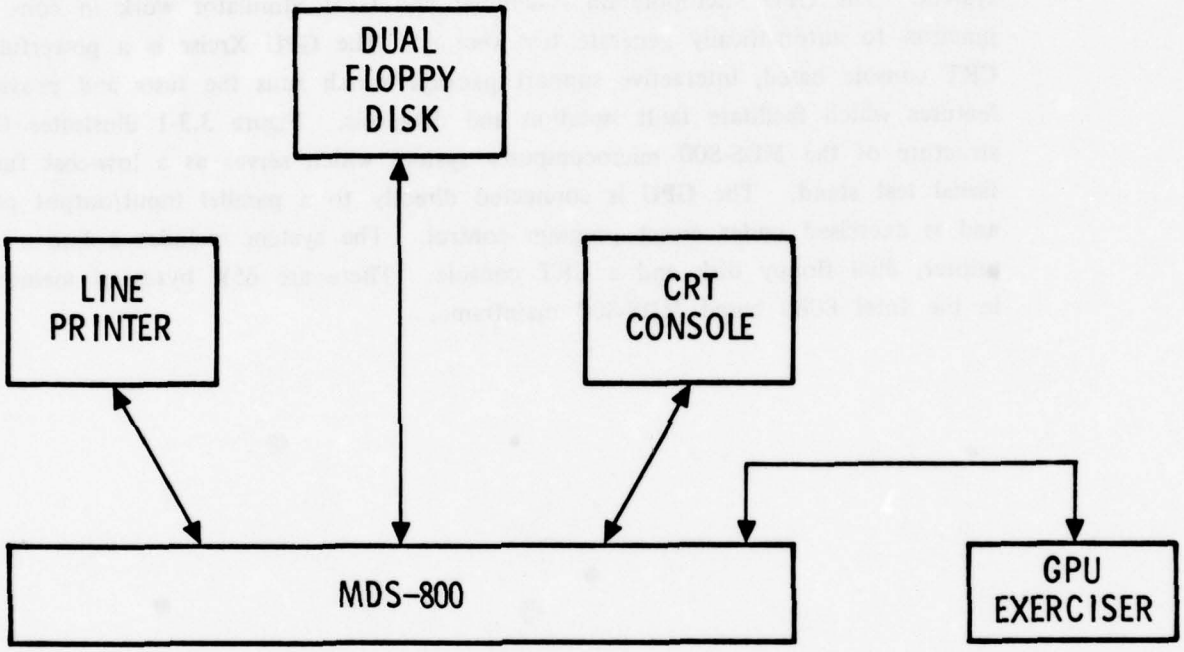


Figure 3.3-1 MDS-800 Computer System Configuration

3.3.1.1 GPU Summary of Characteristics

To place this section in perspective a brief description fo the GPU is provided in this section. Table 3.3-1 summarizes the pertinent characteristics of the GPU. It is a 48-pin device consisting of 2943 transistors of various sizes, in a 43.2 x $10^3 mil^2$ area; this is an average of $14.7 mil^2$ per transistor. The GPU is fabricated using the silicon-gate CMOS/SOS process employing only one epitaxial deposition. A single ion implant is used to dope the silicon islands. The channel oxide is formed by the conventional wet-HCL thermally grown SiO_2 method.

As illustrated in Figure 3.3-2 the GPU is as 8-bit processor bit slice consisting of; 16 word by 8 bit dual access register file, two port buffers (P1B, P2B), extensive data type selection multiplexing, arithmetic/logic circuit (ALC) and powerful shifting capability. It has separate 8-bit data input and data output paths. Data can be input directly to the register file, P1B or P2B. Data can be output from P1B, P2B or from the ALC. Output drivers are designed to drive 30pf with a rise and fall time of 30ns at 10 volts. The data paths are designed to facilitate the implementation of complex algorithms. For example, two bit multiple data paths consisting of a right shift of one into the ALC and right shift of one or two into the register file are implemented on the chip. Although the device is not yet characterized, preliminary tests indicate it is fast – full cycle (accessing of two operands from the file, operating on them through the ALC and storing the result back to the register file) operations of up to 10MHz have been observed.

Table 3.3-1 GPU SUMMARY OF CHARACTERISTICS

- 8-bit processor bit slice.
- 16 dual access general purpose registers.
- Single clock operation – ability to access two registers, operate on them and store away result in register file, all in one clock cycle.
- 8-bit parallel arithmetic logic circuit (ALC) with carry look ahead, all-zero detection and overflow indication.
- Data path and control provided for multiple step algorithms such as two-bit multiply, division and floating point.
- Expandable – can concatenate any number of GPU's for larger word size machines.
- Microprogram versatility – can independently control selection of sources, ALC operation and destination of data.
- CMOS/SOS – chip size 201 x 215.
- Static operation.
- Single DC power source 4 Volts to 15 Volts.

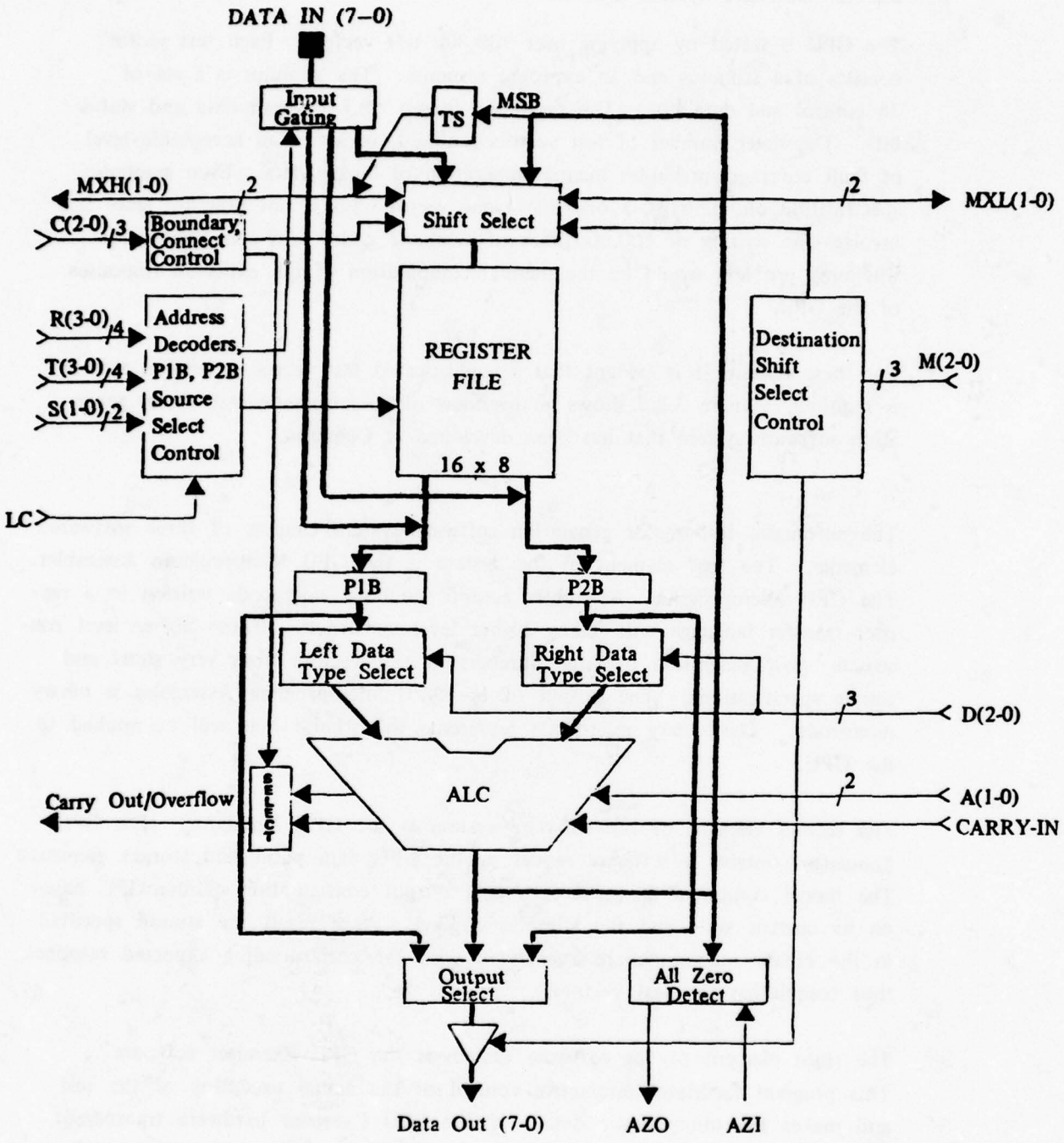


Figure 3.3-2 GPU ARCHITECTURE

3.3.1.2 Software System Overview

The GPU is tested by applying over 100,000 test vectors. Each test vector consists of a stimulus and an expected response. The stimulus is a set of 38 control and data bits. The responses consist of 14 output data and status bits. The sheer number of test vectors required, to attain an acceptable level of fault coverage, precludes manual generation of binary data. Even symbolic specification on, perhaps, a one line - one vector basis is not practical since it involves the writing of 100,000 lines of symbolic code. An even more overwhelming problem would be the manual computation of the expected responses of the GPU.

For these reasons it is evident that a sophisticated test vector generation system is required. Fiugre 3.3-3 shows an overview of an automatic test vector generation software system that has been developed at Questron.

The automatic test vector generation software system consists of three software elements. The first element of the system is the GPU Microprogram Assembler. The GPU Microprogram Assembler accepts symbolic microcode written in a register transfer language with many higher level constructs. These higher level constructs allow generation of large numbers of test vectors from very short and simple specifications. The output of the GPU Microprogram Assembler is binary microcode. The binary microcode represents the stimuli that will be applied to the GPU.

The second element of the software system is the GPU Simulator. The GPU Simulator contains a software model of the GPU data paths and storage elements. The model computes the next state and output configuration of the GPU based on its current state and the stimulus applied. As a result the stimuli specified in the binary microcode are augmented with the corresponding expected respones, thus completing the test vectors.

The third element of the software system is the GPU Exerciser software. This program facilitates interactive control of the actual execution of the test and makes the idiosyncratic details of the GPU Exerciser hardware transparent to the test engineer. The GPU Exerciser software package includes many features that facilitate fault isolation.

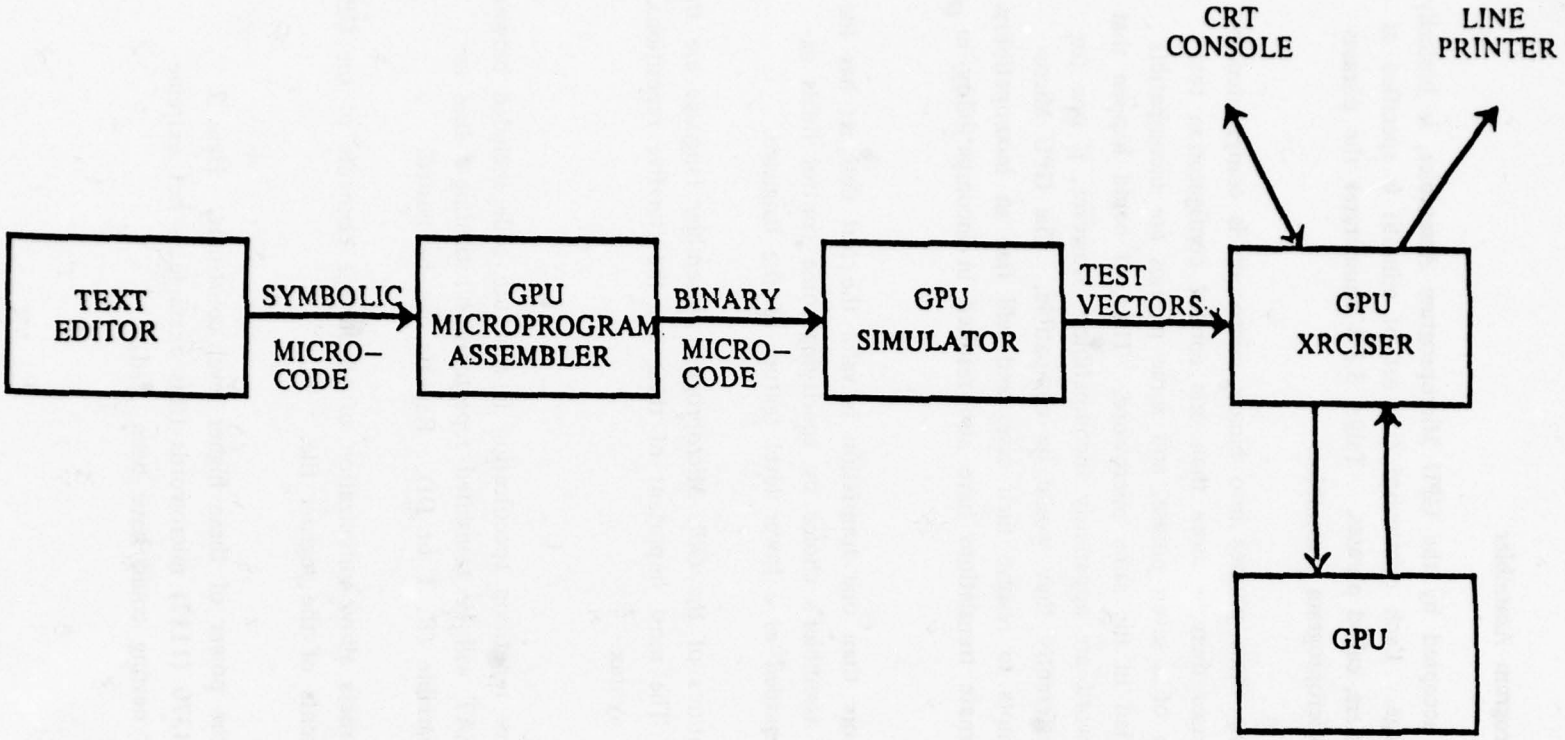


FIGURE 3.3-3 SYSTEM OVERVIEW

QUESTRON

3.3.2 The GPU Microprogram Assembler

The symbolic language, accepted by the GPU Microprogram Assembler, is basically a register transfer language. Each microword (i.e. set of stimuli) is specified as a series of register transfers, called phrases. Table 3.3-2 enumerates the phrases accepted by the GPU Microprogram Assembler.

The process of translating these phrases into binary microcode is complicated by two factors: in many cases there is more than one control configuration that will cause the execution of a given phrase, and some phrases are incompatible with others when specified in the same microword. Thus, it could happen that two phrases in a microword are apparently incompatible. However, if one (or both) were translated differently they would be compatible. The GPU Microprogram Assembler attempts to resolve such cases and will flag an incompatiblity only if all possible alternate translations have also resulted in incompatibility in the given context.

In those cases where more than one translation is valid the test designer has the option to influence the assembler's choice by specifying some control fields explicity. This can be regarded as a lower level feature of the language.

The most significant features of the GPU Microprogram Assembler language are the higher level constructs. The most important of these are the iterative repetitions. Table 3.3-3 shows their syntax.

The effect of an iterative repetition specification is that the code included between REPEAT and ENDREPEAT will be translated repeatedly substituting values as specified for the loop variable (R, T or DI). Repeats can be nested.

Other higher level constructs allow initialization of all storage elements of the GPU and display of the contents of the register file.

Figure 3.3-4 illustrates the power of these higher level constructs. Here 7 lines of code generate 4370 (1112) microwords (this example is not extreme because a third level of nesting could have been used).

TABLE 3.3-2 GPU MICROPROGRAM ASSEMBLER LANGUAGE

Phrases	Default
1) TS = ALC (7),	TS = TS
2) REG(n) = ALC, REG(n) = SHIFTER,	REG(n) = ALC
3) P1B = REG(n), P1B = REG(n+1), P1B = DI,	P1B = REG(0)
4) P2B = P2B, P2B = REG(n), P2B = DI,	P2B = REG(0)
5) DI = xx,	DI = 00
6) DO = OFF, DO = P1B, DO = P2B, DO = ALC,	D0 = OFF
7) ALC = *left* + *right* ALC = *left* + *right* + 1 ALC = *left* – P2B ALC = *left* OR *right* ALC = *left* AND *right* *left* is one of: 0 P1B MXH1IN\\P1B(7-1) ·NOT·P1B	ALC = left + right left = P1B
right is one of: 0 P2B ·NOT·P2B	right = 0

TABLE 3.3-2 (cont.)

Phrases	Default
8) COUT = OFL	COUT = COUT
9) AZIN = x	AZIN = 1
10) SHIFTER = DI, SHIFTER = ALC(6-0)\\0, SHIFTER = ALC(6-0)\\1, SHIFTER = ALC(6-0)\\MXLOIN, SHIFTER = 0\\ALC(7-1), SHIFTER = 1\\ALC(7-1), SHIFTER = ALC(7)\\ALC(7-1), SHIFTER = MXHOIN\\ALC(7-1), SHIFTER = 00\\ALC(7-2), SHIFTER = 11\\ALC(7-2), SHIFTER = ALC(7)\\ALC(7)\\ALC(7-2), SHIFTER = MXHOIN\\MXH1IN\\ALC(7-2),	SHIFTER = DI
11) MXL0IN = x,	MXL0IN = 0
12) MXH0IN = x,	MXH0IN = 0
13) MXH1IN = x,	MXH1IN = 0
14) MXL0OUT = ALC(0),	input
15) MXL1OUT = ALC(1), MXL1OUT = P1B(0),	OFF
16) MXH0OUT = ALC(7),	input
17) MXH1OUT = ALC(6), MXH1OUT = TS, MXH1OUT = P2B(7),	input
18) LC = 00, LC = 01, LC = 10,	LC = 00

; instead of , ends the microword.

[END] ends the program.

```
                    QUESTRON

          GPU MICROPROGRAM ASSEMBLER.                    PAGE  2
-------------------------------------------------------------------------------

        REPEAT R=COUNT;

        REPEAT T=COUNT;


        P1B = REG(X), P2B = REG(X), REG(X) = ALC, ALC = P1B + .NOT.P2B,
        DO = ALC, LC = 01;

---------------
! ADDR = 0012 !
-------------------------------------------------------------------------------
    ! MXH1 = 0 ! DI = 00  ! R = X ! S = 0 ! M = 5 ! CIN  = 0 ! A = 0 ! LC = 1 !
    -------------------------------------------------------------------------
    ! MXH0 = 0 ! MXL0 = 0 ! T = X ! D = 1 ! C = 0 ! AZIN = 1 !   00001C0069   !
    -------------------------------------------------------------------------


        DISPLAY;

---------------
! ADDR = 0013 !
-------------------------------------------------------------------------------
    ! MXH1 = 0 ! DI = 00  ! R = X ! S = 0 ! M = 7 ! CIN  = 0 ! A = 0 ! LC = 1 !
    -------------------------------------------------------------------------
    ! MXH0 = 0 ! MXL0 = 0 ! T = 0 ! D = 7 ! C = 0 ! AZIN = 1 !   004010007F   !
    -------------------------------------------------------------------------


        END-REPEAT.

        END-REPEAT.


1111 IS LAST ADDRESS USED.
   0 ERRORS
```

Figure 3.3-4 EXAMPLE OF HIGHER LEVEL CONSTRUCT USAGE

Table 3.3-3 Iterative Repetitions

REPEAT { DI / R / T } = { WALK 0 / WALK 1 / COUNT };

. . . . any number of microinstructions
. . . .

ENDREPEAT;

Within repetitions registers may be referenced as REG(X). Varying values of R or T (as determined by context) will be substituted. DI = XX indicates that the value is to be determined by the REPEAT instruction.

3.3.3 GPU Simulator

The GPU Simulator operates on the binary microcode produced by the GPU Microprogram Assembler. It generates the next state and the expected outputs of the GPU based on the current state and the current input vector (stimulus). The expected outputs are used to augment the input vector thus creating complete test vectors.

The most important part of the simulation program is the model of the GPU. This model is implemented at the element device level, as opposed to the gate or transistor level. For example, a multiplexer is simulated with a select case construct. The model produces two-valued output (i.e. 0 or 1 and not 4.7V or don't care). No time delays are simulated.

Figure 3.3-5 shows a sample of the printed output from the GPU Simulator. The leftmost column contains addresses which can be used to key to GPU Microprogram Assembler of GPU Exerciser listings. The next two columns represent the contents of the test vector, stimuli and expected responses, respectively. The last two columns represent the assumed contents of all GPU storage elements. This latter information is not part of the test vector, but knowing what the simulator assumes the contents of storage elements to be is an important testing aid.

QUESTRON

GPU MICROPROGRAM SIMULATOR.

PAGE 1

:F1:P1BS1.BIN 17 MAY 1977

ADDR	DI	RT	S	D	A	CIN	AZIN	M	C	MMXXHL	LC	DO	COUT	AZOUT	MMXXHL	REGISTER FILE	P1B	P2B	TS
																36 42 2A FA 93 26 00 03	26	00	1
																09 36 2B 2A FA 93 26 00			
0000	00	00	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 42 2A FA 93 26 00 03	00	00	1
																09 36 2B 2A FA 93 26 00			
0001	00	10	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 2A FA 93 26 00 03	00	00	1
																09 36 2B 2A FA 93 26 00			
0002	00	20	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 FA 93 26 00 03	00	00	1
																09 36 2B 2A FA 93 26 00			
0003	00	30	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 93 26 00 03	00	00	1
																09 36 2B 2A FA 93 26 00			
0004	00	40	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 26 00 03	00	00	1
																09 36 2B 2A FA 93 26 00			
0005	00	50	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 03	00	00	1
																09 36 2B 2A FA 93 26 00			
0006	00	60	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 03	00	00	1
																09 36 2B 2A FA 93 26 00			
0007	00	70	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 00	00	00	1
																09 36 2B 2A FA 93 26 00			
0008	00	80	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 00	00	00	1
																00 36 2B 2A FA 93 26 00			
0009	00	90	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 00	00	00	1
																00 00 2B 2A FA 93 26 00			
000A	00	A0	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 00	00	00	1
																00 00 00 2A FA 93 26 00			
000B	00	B0	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 00	00	00	1
																00 00 00 00 FA 93 26 00			
000C	00	C0	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 00	00	00	1
																00 00 00 00 00 93 26 00			
000D	00	D0	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 00	00	00	1
																00 00 00 00 00 00 26 00			
000E	00	E0	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 00	00	00	1
																00 00 00 00 00 00 00 00			
000F	00	F0	1	7	1	0	1	0	0	00	0	FF	0	1	33	00 00 00 00 00 00 00 00	00	00	1
																00 00 00 00 00 00 00 00			

Figure 3.3-5 Sample of GPU Simulator Listing

3.3.4 GPU Exerciser Software

In its simplest use the GPU Exerciser allows the test engineer to run a file of vectors on the GPU and to receive a message on the CRT wherever the actual responses don't correspond to the expected responses. The idiosyncracies of the I/O interface to the GPU are completely transparent.

Table 3.3-4 summarizes the more advanced features of the GPU Exerciser Software. Output can be directed to CRT, line printer or both. Continuous tracing can be turned on or off, execution can take place in single steps, from beginning to end of file or repeatedly from beginning to end of file. Execution can be interrupted by pressing by pressing the INT7 control switch and can thereafter be continued. It can be specified that the test shall be halted after a mismatch or that it shall not be halted. Four breakpoints can be specified. Contents of all storage elements can be displayed non-distructively and storage elements can be initialized from the console at run time. Figure 3.3-6 is a sample of lineprinter listing from the GPU Exerciser. Here a test ran successfully, tracing listing directed to the lineprinter being specified.

TABLE 3.3-4 XRCISE SUBCOMMANDS

- OUTPUT CONTROL

  OUTPUT = CRT – ALL OUTPUT TO CRT

  OUTPUT = LP – TRACE OUTPUT TO LP ONLY, SUBCOMMANDS TO CRT AND LP

  OUTPUT = BOTH – ALL OUTPUT TO CRT AND LP

- TRACE CONTROL

  TRACE – DISPLAY STATUS VECTOR FOR EACH MICROWORD EXECUTED

  NOTRACE – DISPLAY STATUS VECTOR ONLY

  – ON MISMATCH

  – ON BREAKPOINT MATCH

  – AFTER *INT7*.

- EXECUTION CONTROL

  STEP OR *null* – EXECUTE ONE MICROINSTRUCTION

  RUN – BEGIN AT CURRENT MICROWORD, STOP AT END OF FILE

  LOOP – BEGIN AT CURRENT MICROWORD, AT END–OF–FILE START FROM BEGINNING–OF–FILE.

  LOOP = n – SAME AS LOOP, STOPS AFTER n–th END–OF–FILE (n = 1 to 5 DECIMAL DIGITS, n < 64K)

  C – CONTINUE AFTER *INT7* BREAKPOINT OR MISMATCH

- STOP CONTROL

  ERRSTOP – STOP ON MISMATCH

  NOERRSTOP – DO NOT STOP ON MISMATCH

  BP0 = XXXX
  BP1 = XXXX
  BP2 = XXXX
  BP3 = XXXX
  } SET ONE OF FOUR BREAKPOINTS (XXXX = FOUR HEX DIGITS)

  NOBP – ALL BREAKPOINTS DEACTIVATED

  INT7 – STOP AFTER CURRENT MICROWORD

- DISPLAY CONTROL AND INITIALIZATION

  P1B = XX
  P2B = XX
  REG(n) = XX
  } INITIALIZATIONS

  DISPLAY – DISPLAY ALL REGISTERS

- OTHER SUBCOMMANDS

  REOPEN – NEXT MICROINSTRUCTION TO BE EXECUTED IS AT BEGINNING–OF–FILE

  END – TERMINATE EXERCISE, RETURN TO ISIS–II

```
                    QUESTRON

            GPU STATIC EXERCISER.

--------------------------------------------------------

        :F1:P1B.BIN                          9 MAR 1977

--------------------------------------------------------

                ADDRESS  INPUT VECTOR    EXPECTED   ACTUAL
                -------  ------------    --------   ------
?RUN
                  0000    7180590007       FF0F       FF0F
                  0001    D500511007       FF0F       FF0F
                  0002    F900512007       FF0F       FF0F
                  0003    2100513007       FF0F       FF0F
                  0004    0000514007       FF1F       FF1F
                  0005    0000515007       FF1F       FF1F
                  0006    3900516007       FF0F       FF0F
                  0007    4E00517007       FF0F       FF0F
                  0008    2300518007       FF0F       FF0F
                  0009    4600519007       FF0F       FF0F
                  000A    C50051A007       FF0F       FF0F
                  000B    230051B007       FF0F       FF0F
                  000C    230051C007       FF0F       FF0F
                  000D    230051D007       FF0F       FF0F
                  000E    4E0051E007       FF0F       FF0F
                  000F    230051F007       FF0F       FF0F
                  0010    C500120067       FF0F       FF0F
                  0011    4638510067       FF0B       FF0B
                  0012    800051007C       800F       800F
                  0013    7F001200BC       800F       800F
                  0014    7F005100BC       800F       800F
                  0015    400051007C       400F       400F
                  0016    BF001200BC       400F       400F
                  0017    BF005100BC       400F       400F
                  0018    200051007C       200F       200F
                  0019    DF001200BC       200F       200F
                  001A    DF005100BC       200F       200F
                  001B    100051007C       100F       100F
                  001C    EF001200BC       100F       100F
                  001D    EF005100BC       100F       100F
                  001E    080051007C       080F       080F
                  001F    F7001200BC       080F       080F
                  0020    F7005100BC       080F       080F
                  0021    040051007C       040F       040F
                  0022    FB001200BC       040F       040F
                  0023    FB005100BC       040F       040F
                  0024    020051007C       020F       020F
                  0025    FD001200BC       020F       020F
                  0026    FD005100BC       020F       020F
                  0027    010051007C       010F       010F
                  0028    FE001200BC       010F       010F
                  0029    FE005100BC       010F       010F
?END.
```

Figure 3.3-6 Sample of GPU Exerciser Listing

3.3.5 Testing Approach

3.3.5.1 Overview

Design and process verification and validation was the purpose of GPU testing. As such, the tests were designed to detect problems that may have occured during each step of the design cycle that begins with a functional specification and ends with an LSI device in hand. Therefore, the tests detect; system design errors, logic design errors, electrical design errors, mask layout errors, processing problems, handling and packaging problems. Because of a tight schedule GPU testing conducted in two phases; a static exerciser phase and a dynamic exerciser phase. In the static phase test vectors were applied to the GPU, one at a time, under direct program control; therefore, the tests are run at the speed of the main frame. In the dynamic exerciser phase, blocks of test vectors were buffered in the dynamic exerciser and executed at full speed – over 10 MHz. Static exerciser testing could be initiated much sooner than dynamic testing (because of its relative simplicity) and it is capable of detecting functional errors (errors in interpretation of the functional specification) which require the longest lead time to ameliorate. Electrical design errors, mask layout errors, processing problems, packaging and handling problems that manifest themselves as simple "stuck ats" or pattern dependent "stuck ats" are also detectable during static testing. Static tests are also designed to clock and control decode race conditions into a storage elements. The dynamic exerciser testing phase verifies GPU operation at speed. In essence, it reruns tests developed for static testing and allows, throughput characterization of the device.

3.3.5.2 Degree of Test Complexity

One of the first major decisions made on this program was to determine to what level the GPU should be tested and what the resulting approximate number of test vectors would be. An estimate of the number of test vectors was required in order to guide us in specifying the capabilities of the test system. Because the GPU is a new part, implemented with an immature technology, that has not been functionally, logically or electrically verified, tests are not only designed to establish the top level functional (TLF) integrity of the device and that there are no node "stuck ats" (NSATS) within the device, but also that there is no interaction between "nearest neighbors". Application of the nearest neighbor fault model takes on two aspects within the verification tests of the GPU; nearest electrical neighbor (NEN) and nearest physical neighbor (NPN). Application of nearest neighbor fault model testing, of course, requires detailed part documentation – logic diagrams, electrical diagrams and masks.

What is meant by nearest electrical neighbor can best be cited by example. Consider a data bus with many drivers that can be individually enabled. While testing one driver, the source data inputs to all the other drivers are placed in the opposit state as that applied to the driver under test. Such a test can determine if there is excessive leakage in transistor nodes. In GPU verification tests, nearest electrical neighbor testing often places the GPU in electrically illegal states, such as, two drivers on simultaneously on a bus. This is done to verify the drive capability of specially sized transistors. Electrical neighbor tests for the GPU also check for race conditions into a common storage element.

Nearest physical neighbor testing verifies that there is no spatial interaction between elements. In its simplest application it entails, for example, utilizing a checkerboard pattern in the register file. However, strict adherence to the model requires detailed analysis of the mask, cataloging all parallel signal runs and cross points. Because of the nature of the layout of the GPU, this is an overwhelming task. Therefore, to limit the amount of analysis required, a "functionally" modified nearest physical neighbor fault model is applied. What is meant by this, is that, by the inherent nature of the device, relative physical location and major parallel signal runs and crossings can be identified by doing a functional analysis. It is possible to take this approach with the GPU because the 8-bit data paths of the device are layed out in parallel, top to bottom, and the major sub-functions along the data path (P1B, P2B, Data Type Select, ALC, Output Shift Select) are, more or less, allocated within specific unique areas along the data paths. For example, the three data type select lines (D),

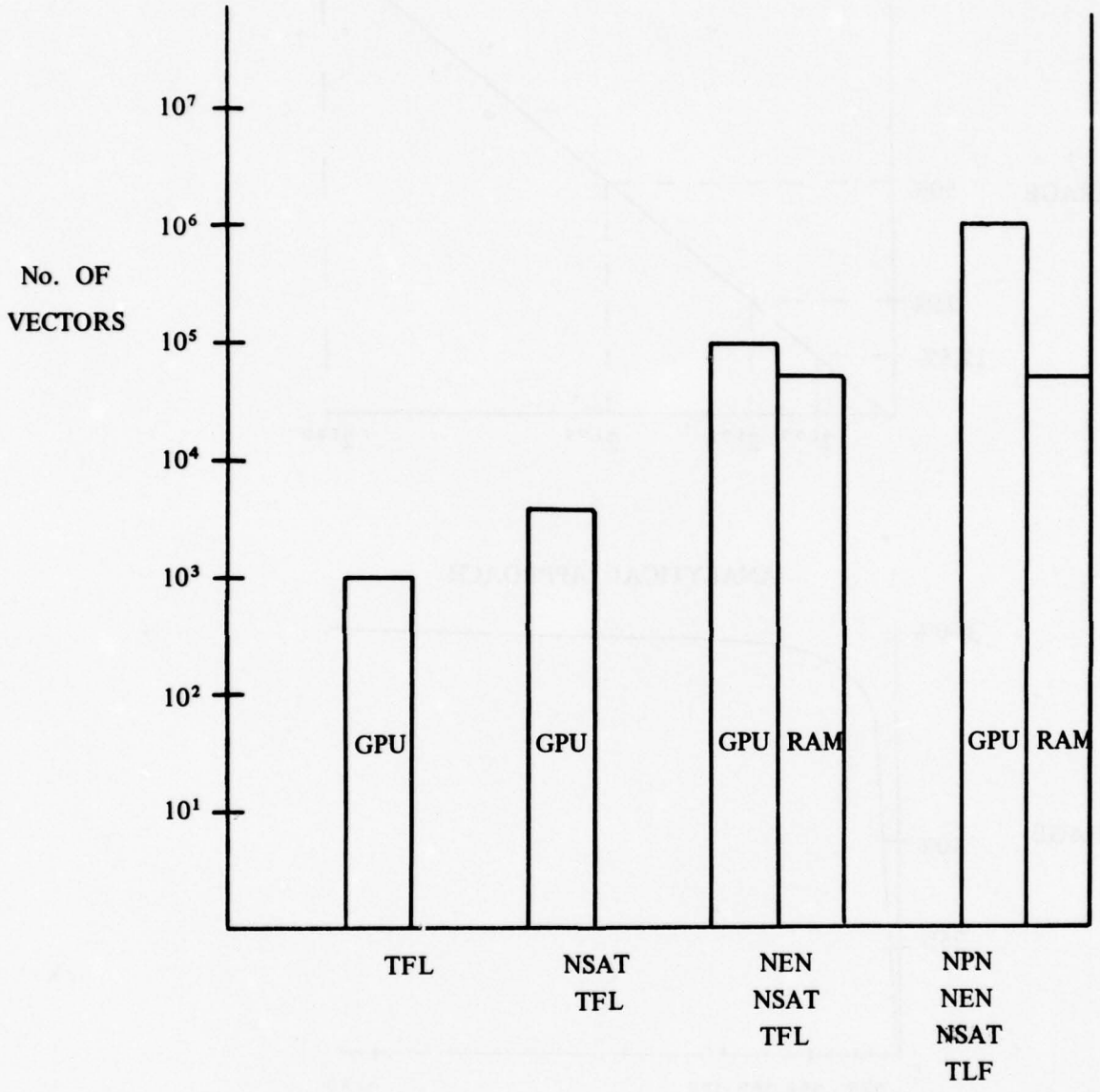


Figure 3.3-7 DEGREE OF TESTING

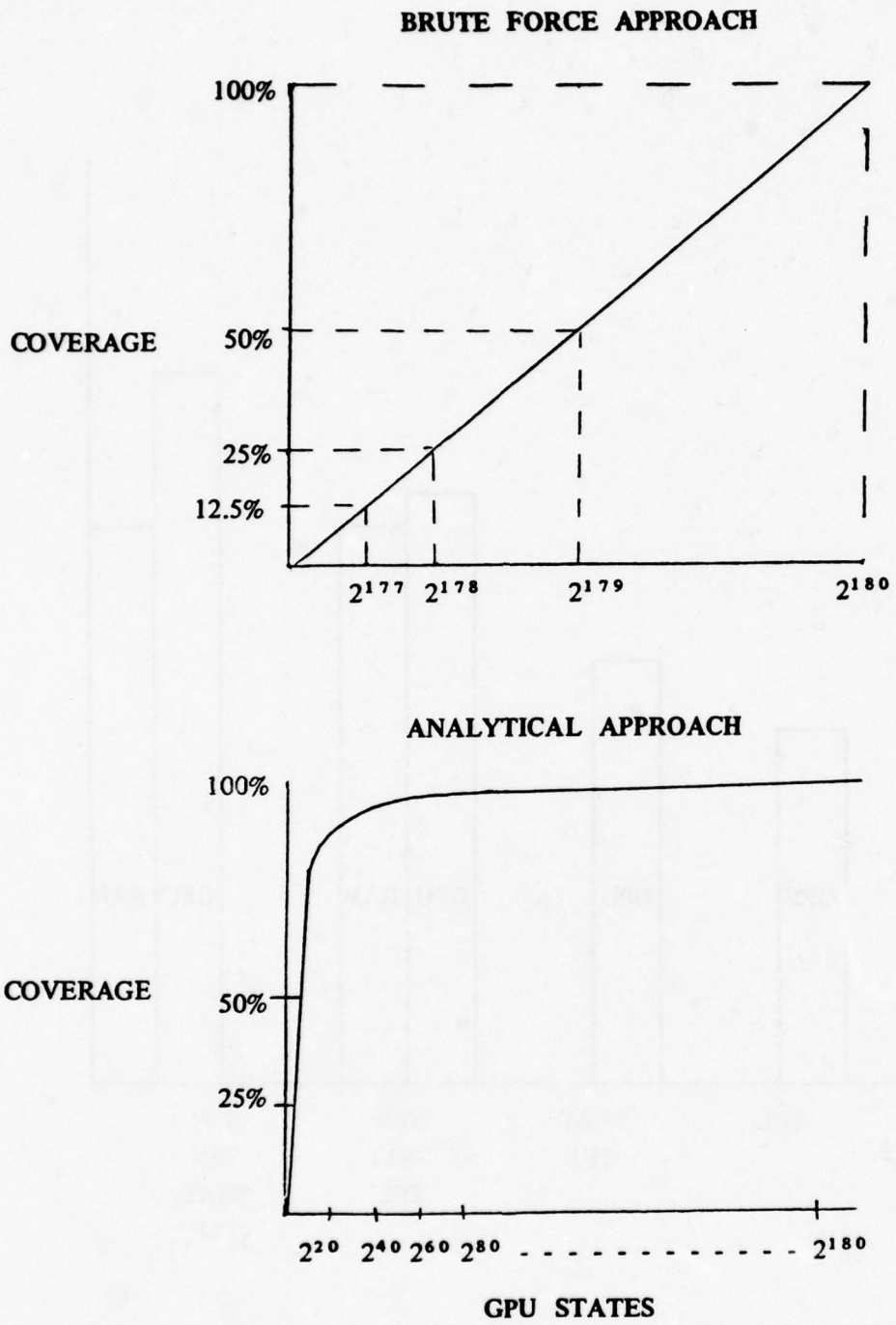


FIGURE 3.3-8 CONCEPTUALIZATION OF TEST VECTOR DESIGN APPROACH

(refer to Figure 3.3-2) are easily identifiable as an extremely long parallel signal run with many crossings. For critical control signals, such as the load clock (LC), the masks are analysed to determine parallel signal runs and crossings.

The application of the nearest electrical neighbor and functionally modified nearest physical neighbor fault models is necessary to limit the total number of test vectors while maintaining an acceptable level of fault coverage. If these models are adequate, or for that matter, even necessary to attain an acceptable level of fault coverage is not known, only process maturity and research will tell. We are relying on past experience in the application of these models.

The need for a fault model is graphically illustrated by Figure 3.3-8. If test vectors were "brute force" randomly generated it would take 2^{180} test vectors to assure 100% fault coverage. Accepting 50% coverage only cuts the number required in half to 2^{179}, and so forth. Therefore, as illustrated by the bottom graph, intelligent design of tests (based on fault models) results in better fault coverage for the same number of vectors. It should be pointed out that no claims are made based on the slope of the curve in Figure 3.3-8 – it is there only to make a point.

3.3.5.3 Test Structure

To make the testing of the GPU manageable, tests are designed on a sub function basis. That is, the GPU has been decomposed into eleven separate sub functional entities and separate test modules written, assembled and simulated for each sub function. These modules, called testware, are combined to form the complete test. Refer to Table 3.3-5.

Each testware module is designed to; establish the functional integrity of the sub function, assure there are no node stuck ats within the sub function, determine the integrity of the electrical interfaces with other sub functions and test the sub function based on the nearest neighbor fault models described previously.

TABLE 3.3.5 TESTWARE MODULES

1. P1B
2. P2B
3. RAM
4. P1B SOURCE
5. P2B SOURCE
6. ALC → RAM
7. LEFT DATA TYPE
8. RIGHT DATA TYPE
9. ALC FUNCTION
10. SHIFT SELECT
11. OUTPUT DISABLE

The structure of the complete test (i.e. the organization of the individual testware modules) is critical in approaching an acceptable level of fault coverage. Not only are GPU testware modules designed to verify individual sub functions on the device, but they are organized, as a whole, in a hierarchial structure in which previously tested sub functions are utilized to test other sub functions. The design of this hierarchial structure, such that all sub function interfaces are adequately tested is crucial in overall test success.

Fault isolation and diagnosis requirements also dictate a hierarchial test structure. With this type of structure, faults can be easily isolated within a sub function or its interface with another particular sub function. Figure 3.3-9 illustrates the concept of the GPU hierarchial test structure. The sub functions electrically closest to the interface pins are tested first, then more and more of the device is used to test itself with each new sub function test. For example, the RAM testware module utilizes P1B and P2B in performing the tests and the P1B, P2B Source tests utilize the RAM, P1B and P2B.

A detailed diagram of the test structure is presented in Figure 3.3-10. It illustrates the organization of the total test program. This organization also defines a module development schedule.

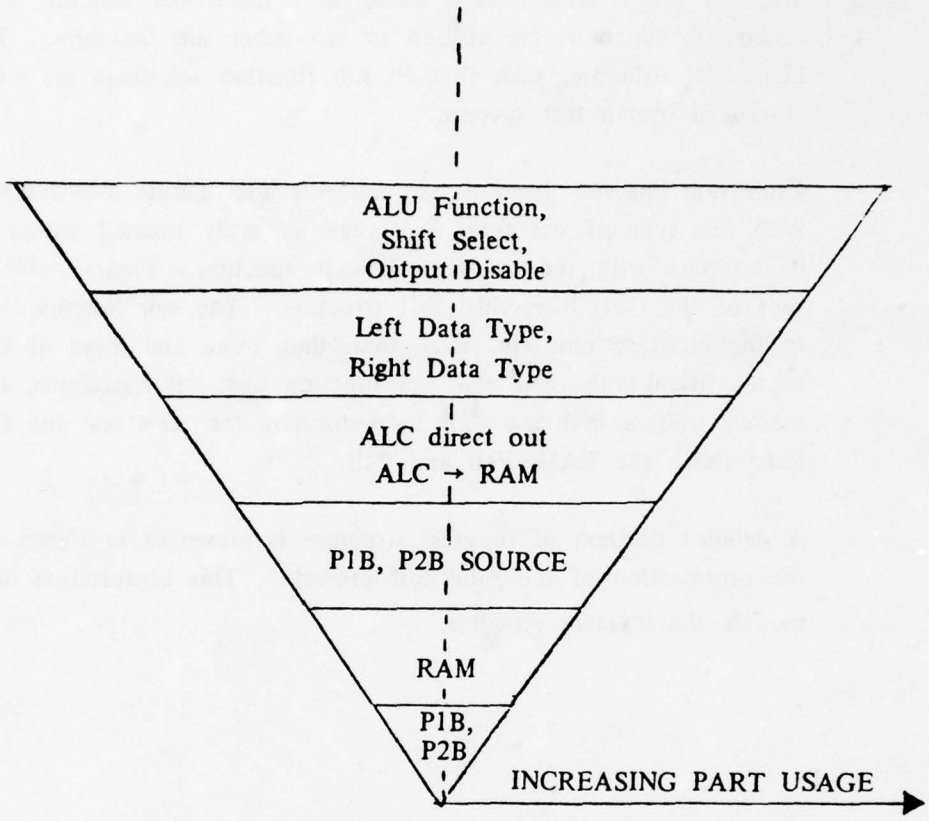


FIGURE 3.3-9 TESTWARE ORGANIZATION

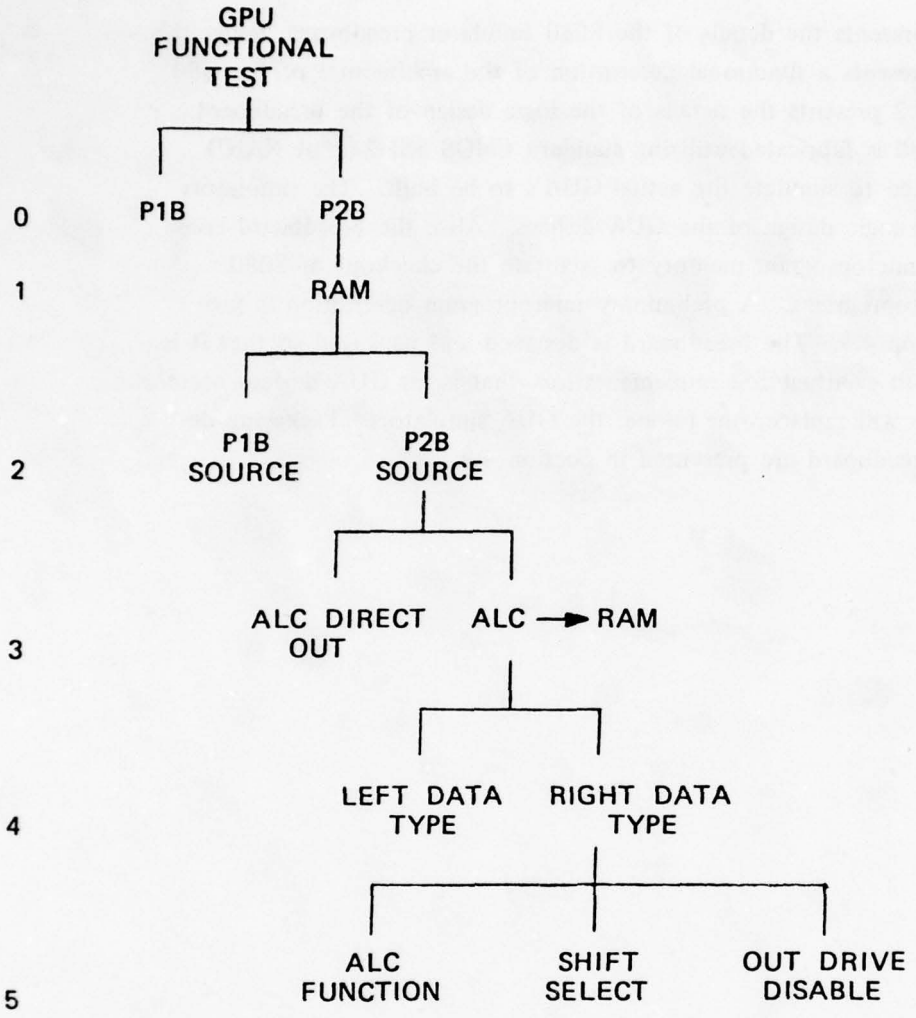


FIGURE 3.3-10 GPU TESTING STRATEGY

4. Q80 Design Description

This section presents the details of the 8080 Emulator breadboard design. Section 4.1 presents a functional description of the architecture of the Q80 and Section 4.2 presents the details of the logic design of the breadboard. The breadboard is fabricated utilizing standard CMOS SSI 2-input NAND gates partitioned to simulate the actual GUA's to be built. The simulators will verify the logic design of the GUA devices. Also, the breadboard contains a RAM microprogram memory to facilitate the checkout of 8080 Emulator microprograms. A preliminary microprogram description is provided in Section 4.3. The breadboard is designed and packaged so that it is compatible with eventual LSI implementation—that is, as GUA devices become available, they will replace, one-to-one, the GUA simulators. Packaging details of the breadboard are presented in Section 4.4.

4.1 Functional Description

The following sub sections present the functional characteristics of the Q80 8080 Emulator. The Q80 implements the central processor unit (CPU) of an 8080 based computer system; it is an instruction set emulator. The Q80 is in no way compatible with 8080 hardware. Hardware compatability is not possible if the full benefits of CMOS/SOS technology are to be derived.

4.1.1 Architecture

The Q80 8080 Emulator is fully software compatible with the 8080A CPU, offered by Intel and other manufactures, except for the implementation of the Decimal Adjust Accumulator (DAA) instruction. The Q80 does not implement the decimal adjust capability. There are several reasons for this; the primary one being that the decimal capability as implemented in the 8080A CPU is incomplete (only decimal addition, and not subtraction, can be performed) and therefore it is of marginal value. Also, the applications spectrum namely real time aerospace systems, does not place a high premium on decimal capability. The Q80 implements a Program Discrete Instruction (PDI) for the DAA operation code. PDI instruction toggles an interface signal directly. This is an extremely valuable capability for program performance evaluation and general interfacing.

For ease in understanding and for purposes of explanation the Q80 can be considered to consist of three functional areas or units; a register file arithmetic logic unit (implemented with a GPU), microprogram control unit and a bus interfacing unit (refer to Figure 4.1.1-1). These three units work in conjunction to implement the 8080A CPU instruction set. From a hardware point of view, the Q80 emulates more than just the 8080A CPU - it emulates the 8080A CPU and the functions provided by the Intel 8224 and 8228 interface devices. In fact, the Q80 interface is almost identical to the interface provided by these three Intel devices. It is not identical because the Intel N-Channel devices require a 2-phase clock system and also because the Q80 provides additional features.

The Q80 is a complete 8-bit parallel, central processor unit (CPU) for use in general purpose digital computer systems. It is fabricated out of CMOS/SOS thus it possesses the advantages of high speed, low power, single power form and single phase clock. The Q80 transfers data via an 8-bit, bidirectional 3-state Data Bus (D_0-D_7) transceiver (T/R). Memory and peripheral device addresses are transmitted over a separate 16-bit 3-state address bus (A_0-A_{15}). Twelve timing, control and status output signals establish timing prototcol for memory and peripheral devices. Five input signals indicate memory and peripheral device requirements to the Q80. A single clock (ν) controls all data transfers and control sequencing – it

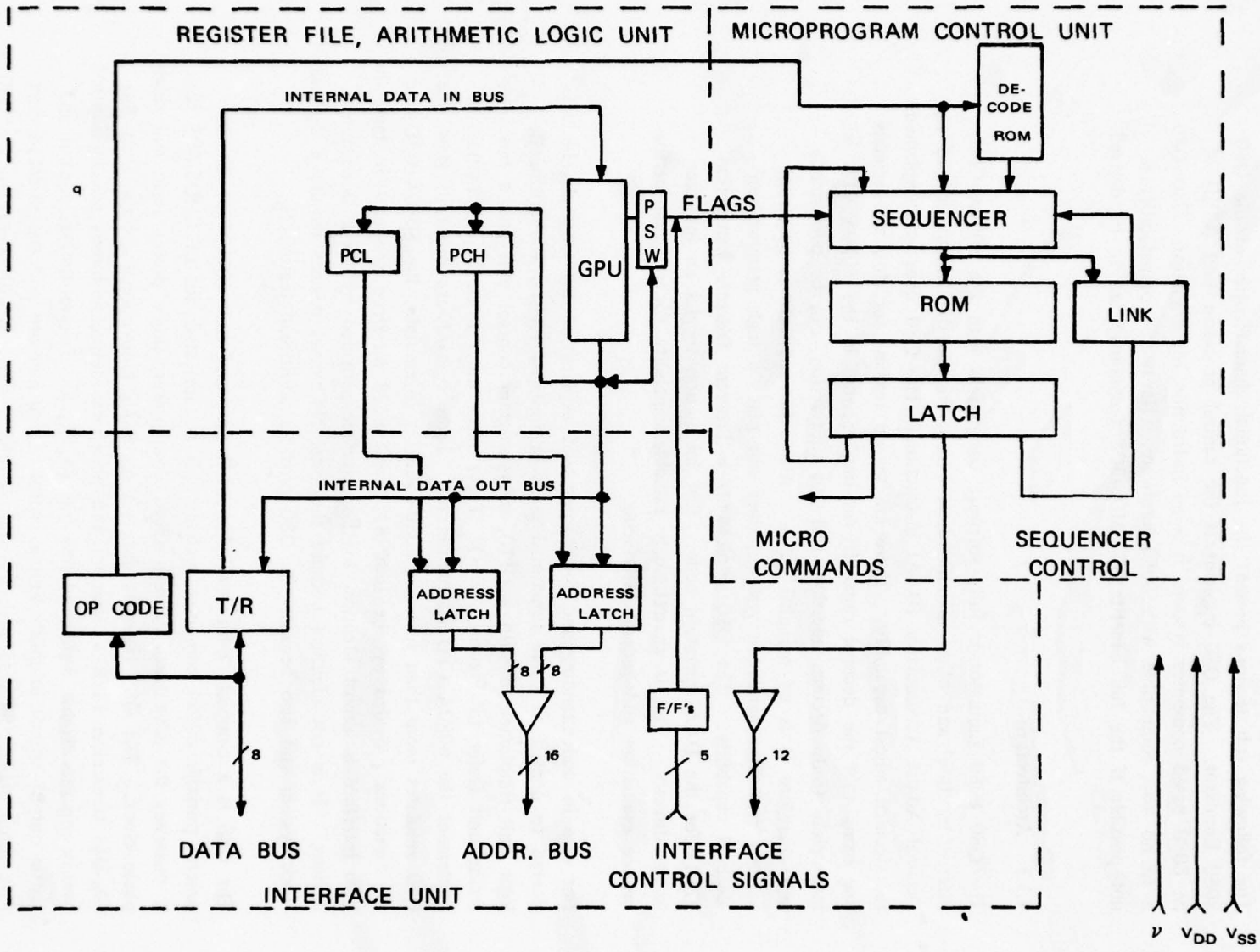


Figure 4.1-1 Q-80 8080 EMULATOR

can be run at any speed from single - step to TBD MHz.

4.1.1.1 Register File, Arithmetic Logic Unit

The GPU forms the heart of the register file arithmetic logic unit portion of the Q80 . It is implemented entirely with a GPU. All programable registers required for the 8080 emulation reside in the GPU (except the Program Counter) and all arithmetic and logical functions are executed by the GPU. The register file section contains the following register functions:

- Six 8-bit general purspoe register arranged as pairs; referred to as B, C; D, E; and H, L. They are all accessable, as general purpose registers, to the programmer and can be manipulated on a single byte or double byte (register pair) basis.

- 16-bit program counter (PC), arranged in two 8-bit halves called PCH(most significant) and PCL (least significant). It holds the 16-bit address of the current instruction being fetched or executed; it is automatically incremented when the next instruction is to be fetched. When program branches occur. the target address is automatically loaded into the PC. The PC is implemented external to the GPU for reasons of speed and microprogram efficiency.
- 16-bit stack pointer (SP) arranged in two 8-bit halves called SPH (most significant half) and SPL (least significant half). The SP holds the 16-bit address of the current top of a stack located in memory. The stack is organized as a last-in-first-out (LIFO) file. Data can be pushed onto the stack from specific Q80 registers or popped off of the stack into specific registers. Also current Q80 status can be saved on the stack and then restored via execution of CALL and RETURN type instructions.

- 8-bit Accumulator (A) register used as the primary data source and destination register for ALU type operations (arithmetic, logical, shifts, etc.)

- 24-bit Instruction Buffer Register (IB) separated into three bytes; IBH (most significant byte, or operation code), IBM (middle byte and IBL (least significant byte). The IB stores the instruction as it is fetched by the microprogram - it is not accessible to the macroprogrammer. IBH is implemented external to the GPU so that the operation code can directly drive the microprogram control unit.

- Five 8-bit temporary registers (TEMP's) are utilized by the microprogram - they are not accessible to the programmer. TEMP's are used primarily to hold microprogram constant data.

- Program Status Word register (PSW). The PSW is an 8-bit register that contains current Q80 status information, such as carry out, zero detection, parity detection, sign bit and overflow. The PSW is the only register located externally to the GPU. It is utilized to control program sequencing via conditional branch instructions.

The GPU is utilized to implement all arithmetic and logical functions of the Q80. Shift gating is required to implement Q80 shift instructions – this gating logic is under direct microprogram control. Illustrated in Figure 4.1-2 is the GPU interface to the interface unit and the microprogram control unit. The GPU data out path forms the Q80 internal data out bus. The internal data out but transfers address and data information from the GPU to the interface unit. The PSW is connected to the internal data out bus so that it can be stored on the stack during CALL type instructions and restored during RETURN instructions. The GPU data input path forms the receiving node for the internal data in bus. Data and instruction information from the interface unit is transfered to the GPU over this bus. To enhance the throughput of the Q80 the program counter (PCH and PCL) is implemented and transfer of the program counter to the address bus. The improvement in throughput (compared to an implementation in which the PC resides in the GPU, thus saving some hardware) is 15%. The PC is also multiplexed onto the internal data in bus so that programming operations can be performed on it such as CALL.

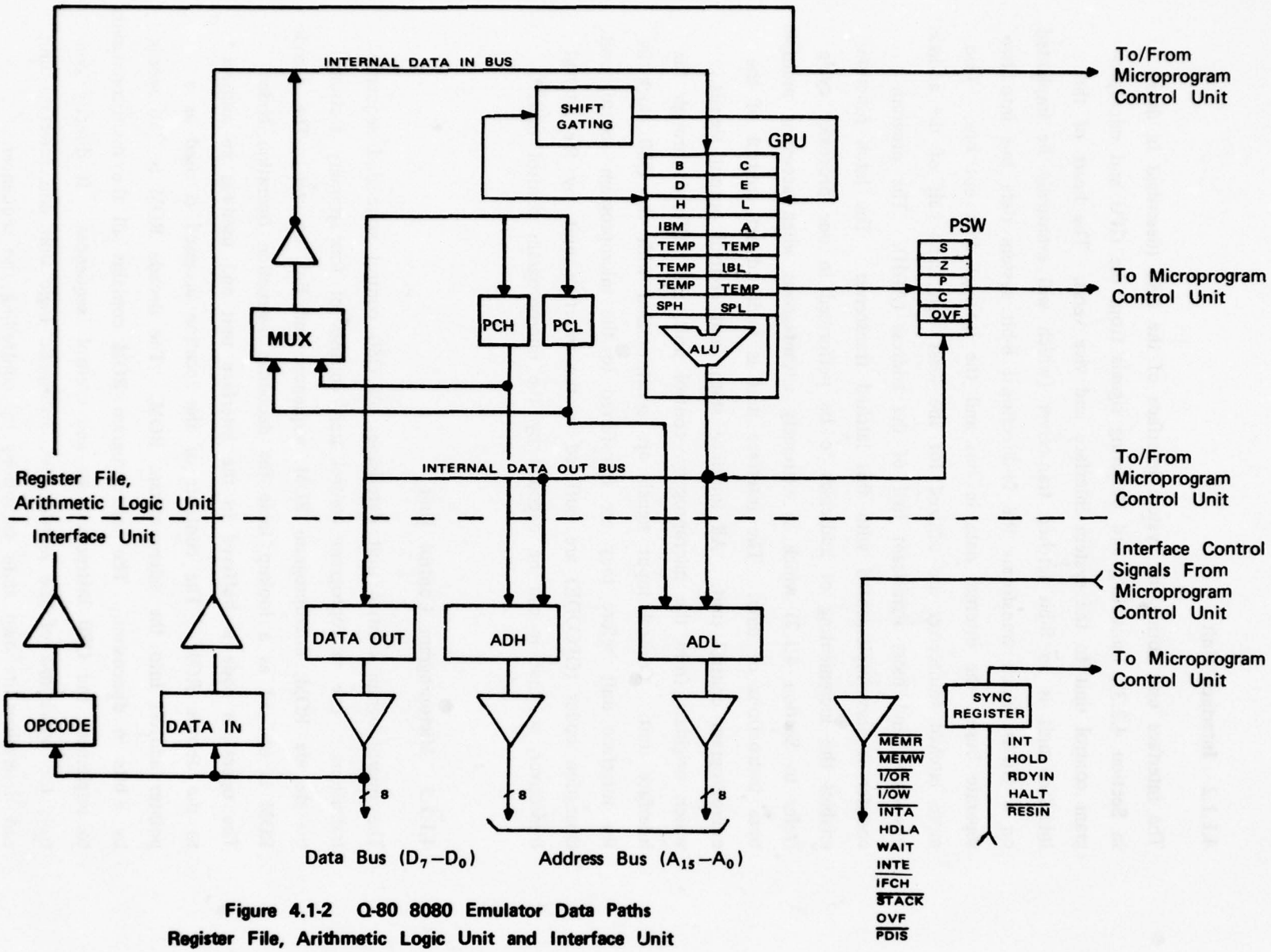


Figure 4.1-2 Q-80 8080 Emulator Data Paths
Register File, Arithmetic Logic Unit and Interface Unit

4.1.1.2 Interface Unit

The interface unit forms the system interface of the Q80 (described in detail in Section 4.2.3), translating and buffering signals from the GPU and microprogram control unit to the system interface and vice versa. The heart of the interface unit is an 8-bit latched transceiver (which will eventually be implanted on a GUA) which transforms the bi-directional 8-bit system data bus into two separate buses; the internal data in bus and the internal data out bus. Two more latched transceiver are utilized for the least significant half of the address bus (ADL) and most significant half of the address (ADH). The program counter is also implemented with this latched transceiver. The latch function enables the incrementing of addresses to be performed in one processor cycle (refer to Section 4.1.2) which is extremely advantageous when accessing multiple byte instructions or data. The interface unit is under direct control of the microprogram control unit. All interface status and control output signals, which originate from the microprogram control unit, are buffered through the interface unit. Control input signals are synchronizied with the Q80 clock in the interface unit before they are transferred to the microprogram control unit. Operation codes (OPCODE) are buffered in the interface unit, by the latched transceiver, so that it can be decoded by the microprogram control unit.

4.1.1.3 Microprogram Control Unit

The microprogram control unit determines all Q80 control state and sequence transitions. The microprogram control unit consists of four primary sections; the decode ROM, microprogram ROM, sequencer and link counter. The decode ROM is utilized as a lookup table for decoding instruction operation codes. The operation code is buffered in the interface unit and used as an address to the decode ROM. The contents of the location accessed is used as a pointer address into the microprogram ROM. The decode ROM is 256 words by 8-bits in dimension. The microprogram ROM contains all the microcommands to implement the Q80 instruction set and control sequences. It directly controls the operations of the register file, arithmetic logic unit and interface unit and determines its own state transitions by controlling the sequencer.

The microprogram ROM is 214 words by 53 bits. The sequencer determines the next microprogram state by operating on address, control and status information to generate address to the microprogram ROM. Four sources of microprogram addresses to the ROM are: the microinstruction itself (from the command latch), the decode ROM, a fixed hardwired address and a link counter (for subroutine returns). The opcode is decoded to select either the output of the decode ROM or the fixed hardwired address. The decode ROM is selected if a single byte instruction is fetched, the decode ROM profiding a branch address directly to the execution routine. If a multiple byte instructions is fetched then the fixed hardwired address is selected and the multiple instruction byte microprogram routine is entered. The link counter stores a return address for subroutine calls and returns. A return linkage is always made to the next sequential address after the address of the call microinstruction. So the address of the calling microinstruction is saved and simply incremented in the link counter. On return it is selected as the address source. The address source is selected by the sequencer based on control inputs provided by the microcommand latch. Status information from GPU flags and interface input control signals are utilized to effect conditional branching within the micro-routines. These flags are utilized to modify the next microprogram address (obtained from the current micro-instruction), again under control of the sequencer control input signals.

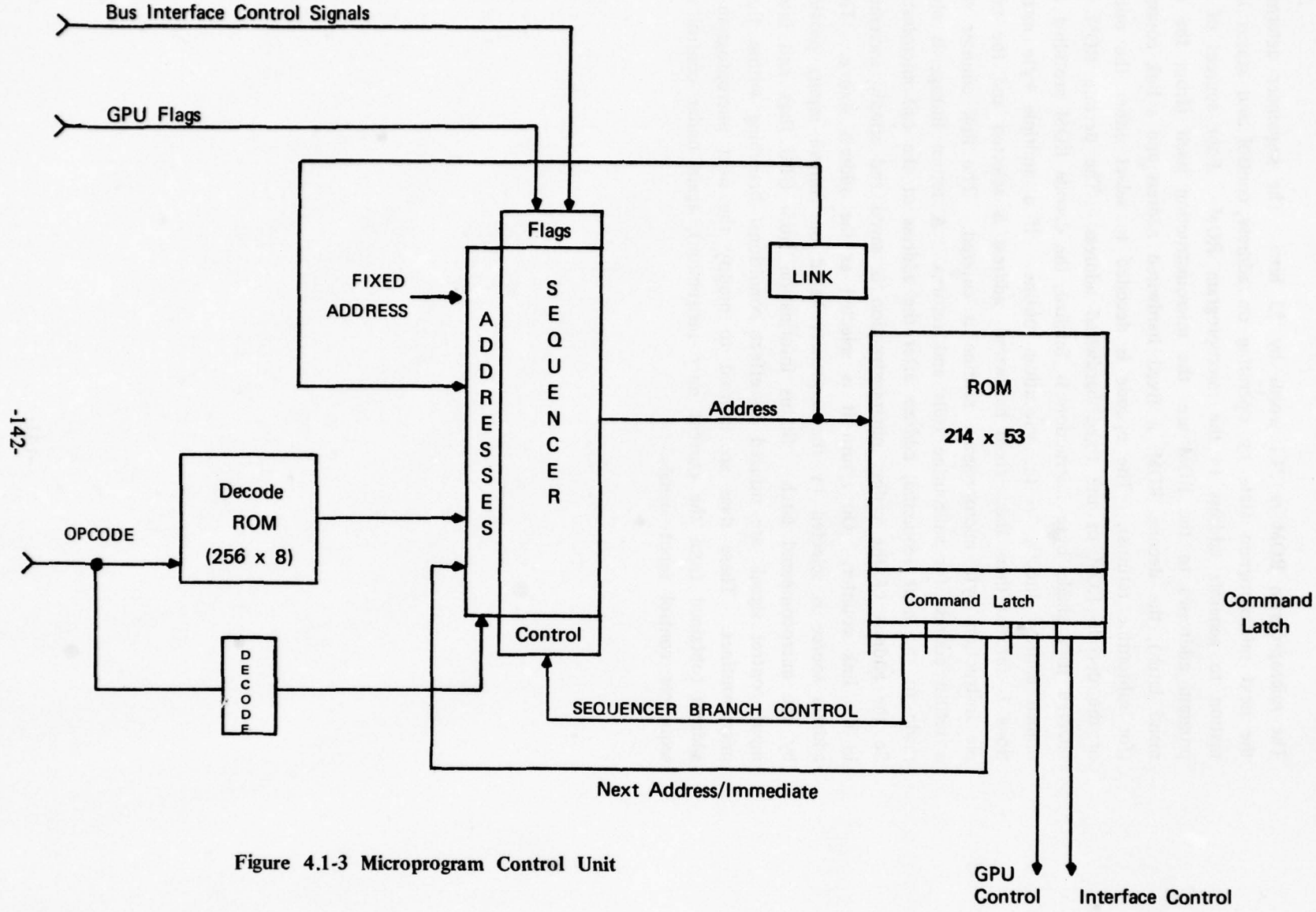


Figure 4.1-3 Microprogram Control Unit

Table 4.1-1 Q80 Summary of Characteristics

Element	Characteristic
Number System	Binary, two's complement
Data Representation	Fixed point integer
Basic Word Length	8-bit byte
Precision	Fixed point; single precision, 8-bits double precision, 16-bits
Operation	8-bit parallel; data paths implemented with GPU. Microprogrammed; ROM size 214 x 53
Registers	Six 8-bit general purpose registers arranged in pairs: B, C D, E H, L Single 8-bit accumulator register: A Program Counter, 16-bit: PC Stack Pointer, 16-bit: SP 8-bit Program Status Word: PSW Two 8-bit Microprogram Working Registers: TEMP1, TEMP2, TEMP3, TEMP4, TEMP5

Table 4.1-1 Q80 Summary of Characteristics (continued)

Element	Characteristic
Addressing Modes	1. Direct 2. Register to Register 3. Register Indirect 4. Immediate
Addressing	Single data byte Double data bytes 64K byte memory addressing capability with 16-bit address bus (K = 1024).
Interrupts	Multilevel priority interrupts possible with additional logic. Interrupts can be enabled and disabled under direct program control.
Input/Output	Up to 256 input ports and up to 256 output ports accessable in I/O instructions OR Any number utilizing system wide addressing
Instruction Set	72 Instructions Total 13 Data Transfer 19 Arithmetic 19 Logical 8 Branch 13 Stack, I/O, and Control Instructions are: 1, 2, and 3 bytes in length
Clock	Single Phase Clock

Table 4.1-1 Q80 Summary of Characteristics (continued)

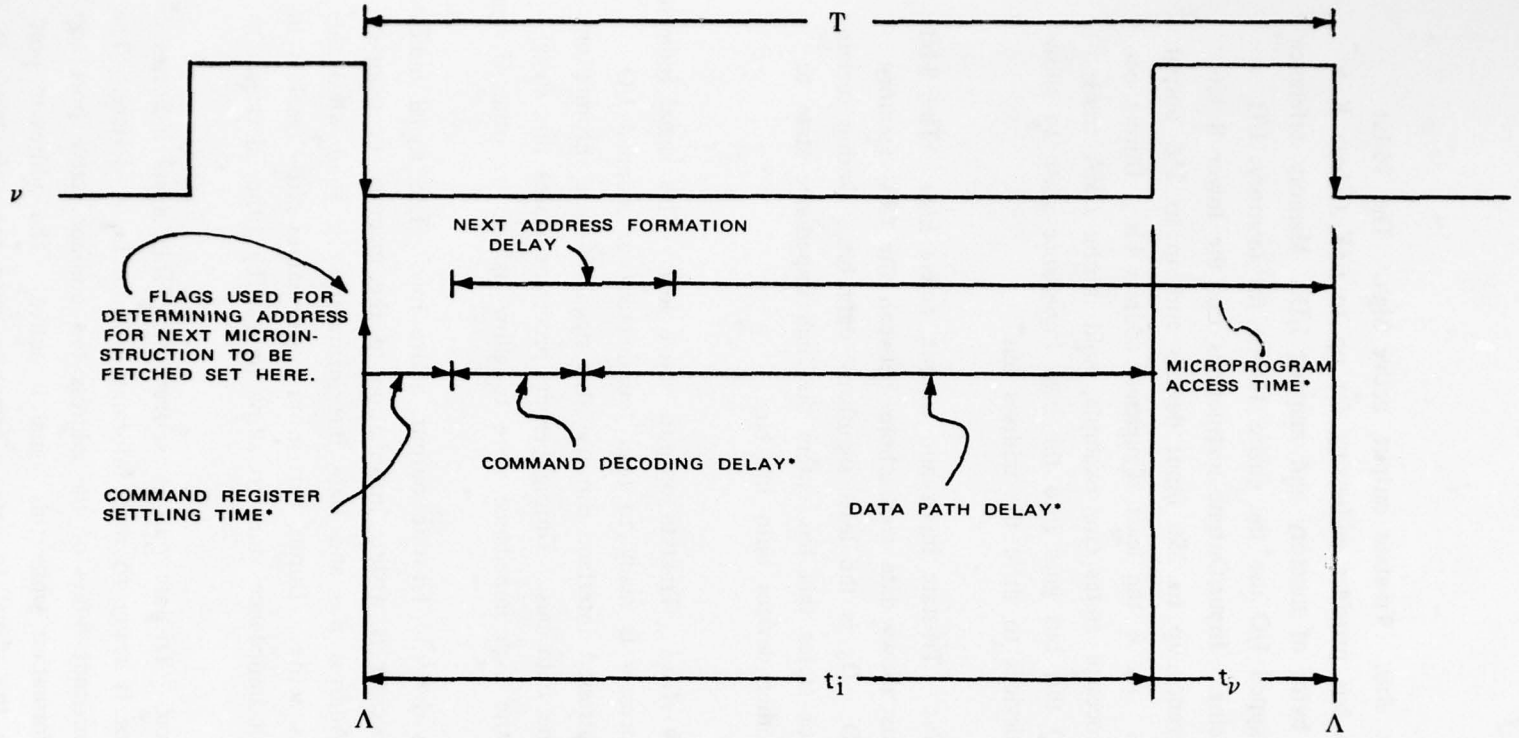
Element	Characteristic
Processor Cycle Time	To be determined (TBD)
	TBD move register to register
	TBD move register to memory
Typical Execution Times	TBD Add register to register
	TBD Add register to memory
	TBD JMP
	TBD CALL
	TBD IN
	TBD OUT
Circuit Implementation	CMOS/SOS
Power Requirements	TBD watts @ 10 volts

4.1.2 Processor Cycle

All data path transactions and microprogram control sequencing are controlled by a single forcing function; the Q80 clock signal (ν) (refer to Figure 4.1-4. The frequency of ν is derived from the data and control delays in the Q80. During the time that ν is low (ti), the data path information is indeterminate – delays causes by microcommand register settling time, command decoding and data path propogation are allowed to transpire. When ν goes high ($t\nu$), data on all data paths has settled out and is valid. The length of $t\nu$ is determined by set up time requirements for the GPU register file and external registers. With the falling edge of $t\nu$ (Λ) data are clocked into their respective destinations. Also at Λ the next microinstruction is clocked into the command register. Microinstruction sequencing in the Q80 is pipelined, which means that, during the execution of the current microinstruction the next microinstruction is fetched. This means that flags that control conditional branching within the microprogram must be set up on the previous clock period. Then during the execution of the current microcommand these flags are used to determine the address of the next microcommand to be executed. This means that flags must be set up two microcommands in advance of the target microcommand. The sum of $t\nu$ and ti form the basic Q80 clock period (T). This period, called T cycle, is the unit for determining instruction execution times. All execution times given for instructions are expressed as multiples of T.

4.1.3 Interface Description

The Q80 interface emulates the interface provided by the Intel 8080 device in conjunction with the Intel 8224 and 8228 interface devices. Additional status signals are also provided on the Q80 interface to facilitate implementation of systems. In all, there are 41 input and output signals plus V_{DD}, V_{SS} and the system clock (CLK). Refer to Figure 4.1-5 The following paragraphs specify the function of each signal.



T ≡ Q-80 clock period; referred to as "T cycles" in calculating instruction execution times

ν ≡ Q-80 clock signal

t_i ≡ data path and control settling time. During this time data is indeterminate on data paths

t_ν ≡ set up time for clocking. During this time data is valid on data paths

Λ ≡ clocking edge. Next microcommand is clocked into command register, data is clocked into registers

*not to scale

Figure 4.1-4 Effects of the clock in the Q-80.

A_{15} - A_0 *Address Bus.* Tri-state output, active high. The 16-bit address bus provides addressing for up to 64K (where K = 1024) bytes of memory and mapped I/O. Memory reference and mapped I/O use the entire 16-bits for memory, I/O referencing. Input/Output instructions use the lower 8 bits to reference up to 256 input devices and up to 256 output devices. A_0 is the least signigicant address bit. During certain processor states (for example, hold during DMA transactions) this bus goes into the high impedance state to allow other devices to drive the address bus.

D_7 - D_0 *Data Bus.* Tri-state input and output, active high. The 8-bit data bus allows data transactions between the CPU memory and I/O. D_0 is the least significant data bit. During certain processor states this bus enters the high impedance state to allow other devices onto the bus.

$\overline{\text{MEMR}}$ *Memory Read.* Tri-state output, active low. This signal indicates the processor is ready to read the memory or mapped I/O. The addressed interface can use this signal to gate information onto the data bus. During certain processor states this signal enters the high impedance state to allow its use by other devices.

$\overline{\text{MEMW}}$ *Memory Write.* Tri-state output, active low. This signal indicates the processor is writing information to the memory or mapped I/O. Address Bus and Data Bus information is valid when this signal is active. During certain processor states this signal is in the high impedance state to allow its use by other devices.

$\overline{\text{I/O R}}$ *I/O Read.* Tri-state output, active low. This signal indicates processor is ready to read information from input devices. The last significant 8-bits of the address bus contain stable port address information while this signal is active. The addressed port can use this signal to gate information onto the data bus. During certain processor cycles this signal enters the high empedance state to allow its use by other devices.

$\overline{\text{I/O W}}$ — *I/O Write.* Tri-state output, active low. This signal indicates processor is ready to write data to an output port. Address Bus (least significant 8-bits) and Data Bus Signals are valid while this signal is active. During certain processor states this signal enters the high impedance state to allow its use by other devices.

$\overline{\text{INTA}}$ — *Interrupt Acknowledge.* Tri-state output, active low. This signal indicates that the processor has recognized an interrupt request (INT) and that the interrupting device must place the interrupt instruction on the Data Bus. Three $\overline{\text{INTA}}$'s will be generated which allows the use of 3-byte CALL instructions for interrupt processing. When $\overline{\text{INTA}}$ is active the Address Bus contains the current program counter state. During certain processor states this signal enters the high impedance state to allow its use by other devices.

HLDA — *Hold Acknowledge.* Active high output. This signal indicates the processor has recognized the HOLD request and that the requesting device is free to use the system bus. While HLDA is active $A_{15} - A_0$, $D_7 - D_0$, $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{I/OR}}$, $\overline{\text{I/OW}}$, and $\overline{\text{INTA}}$ are all in the high impedance state. HLDA becomes inactive when the HOLD request is released.

WAIT — *Waiting In Halt.* Active high output. This signal indicates either the execution of a HALT instruction or recognition of the HALT input signal. Halt state occurs in between execution of instructions and only the following occurrences will take the processor out of the halt state:

- Activation of the reset ($\overline{\text{RESIN}}$) signal. This clears the program counter and the processor starts fetching instructions at address zero.

- A HOLD input will cause the processor to enter the hold state. When the HOLD input becomes inactive the processor will reenter the halt state.
- An interrupt request (INT) will cause the halt state to be exited.
- Release of the $\overline{\text{HALT}}$ input signal.

INTE *Interrupts Enabled.* Active high output set and reset under direct program control. When the interrupt system is enabled this signal is active. It is to be used as a status signal by peripherals.

$\overline{\text{IFCH}}$ *Instruction Fetch.* Active low output signal. This signal is low whenever instruction fetching occurs. It is a processor status indicator.

$\overline{\text{STACK}}$ *Stack Access.* Active low output signal indicates when stack accesses occur. It is to be used as a processor status indicator.

INT *Interrupt Request.* Active high input. Indicates interrupt service request. If interrupts are enabled processor will enter interrupt state before fetching next instruction. The INT signal can be a level or a pulse (of at least one clock cycle); if a level, $\overline{\text{INTA}}$ should be used to clear INT so it is not recognized more than once.

HOLD *DMA Hold Request.* Active high input indicates another device wishes to use the system bus. It is recognized inbetween instruction executions. Recognition is indicated by a HLDA output (refer to HLDA signal description). HOLD is a level input and must be active during the entire DMA transaction.

RDYIN *Ready Input.* Active high input signal which indicates that the accessed memory device or I/O device is ready to receive or transmit the requested data. After the issuance of an $\overline{\text{I/OR}}$, $\overline{\text{I/OW}}$, $\overline{\text{MEMR}}$ or $\overline{\text{MEMW}}$ the RDYIN signal must go inactive before the next falling edge of the system clock if additional clock periods are required to perform the access request. This signal allows slow speed devices to be interfaced to the processor.

$\overline{\text{RESIN}}$ — *Reset Input.* Active low input signal that causes the processor to suspend execution and restart execution at address zero. All program flags are reset and interrupts are disabled. Can be a pulse (at least one clock cycle indirection) or a level.

$\overline{\text{HALT}}$ — *Halt Input.* Active low input signal that causes the processor to enter the halt state. Processor will remain halted as long as this signal is active (this is different than a program halt where an interrupt or restart will take the processor out of the halt state). Occurence of an interrupt while the $\overline{\text{HALT}}$ signal is active will be remembered so that the interrupt state will be entered when $\overline{\text{HALT}}$ is released. $\overline{\text{RESIN}}$ will not be remembered so execution will proceed from where the halt occured. While $\overline{\text{HALT}}$ is active DMA operations can occur as explained previously.

OVF — *Overflow Signal.* Active high output signal that indicates that the results fo the previous dyadic operation results in an overflow. This signal can be integrated into the interrupt system to allow software handling of overflow situations.

$\overline{\text{PDIS}}$ — *Program Discrete.* Active low output signal. Normally in the high impedance state except during execution of Program Discrete Instruction.

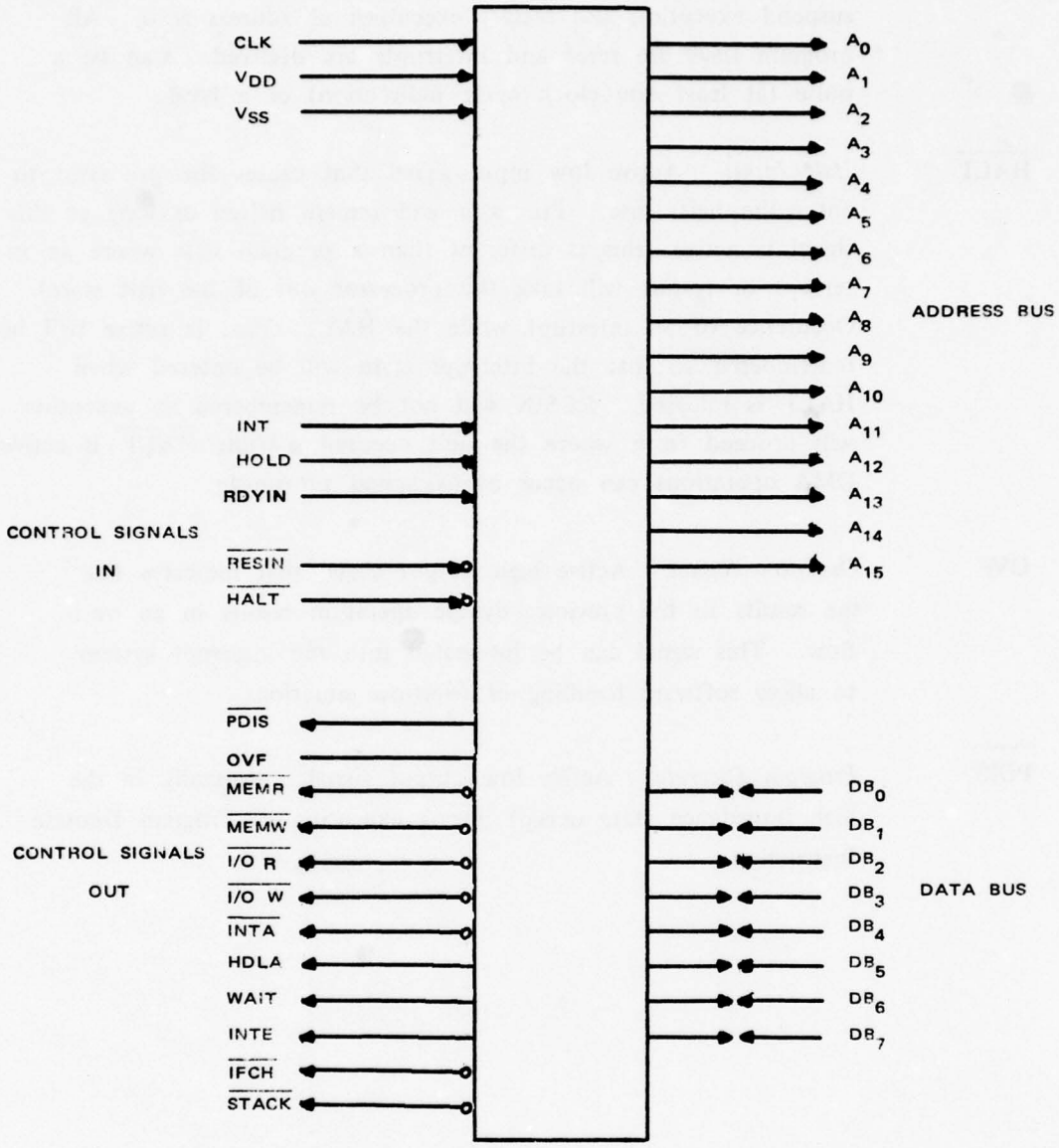


FIGURE 4.1-5 8080 EMULATOR INTERFACES

4.1.4 Interface Timing Protocol

All Q80 interface timing can be illustrated via a few very simple timing diagrams which depict the functions of interface ready/wait handshake for; instruction fetch, memory data read and write, I/O read and write, interrupt recognition protocol, hold request timing and halt protocol. The following sections describe, in detail, the Q80 interface timing. Consult Section 4.1.2 for definition of basic processor clock cycle. When all Q80 bus control signals are inactive ($\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{INTA}}$, $\overline{\text{I/OR}}$, and $\overline{\text{I/OW}}$) the data bus is in the high impedance state. Concurrent with $\overline{\text{MEMR}}$, $\overline{\text{INTA}}$ and $\overline{\text{I/OR}}$ the bus transceiver is in receive mode and overlapping $\overline{\text{I/OW}}$ and $\overline{\text{MEMW}}$ the bus transceiver is in transmit mode.

4.1.4.1 Memory, I/O Read Ready/Wait Handshake

Figure 4.1-7 illustrates the basic handshake transaction between the memory and I/O read control signals ($\overline{\text{MEMR}}$, $\overline{\text{I/OR}}$ and $\overline{\text{INTA}}$) and the ready in (RDYIN) signal. The figure shows the memory ($\overline{\text{I/O}}$) read control signal going low (i.e. active) on the first falling edge of the clock. *If a wait period is required the accessed device must lower the RDYIN signal before the next falling clock edge* (clock edge labeled number 2 in the figure). If the RDYIN is not low by then the Q80 will assume that valid data can be read from the accessed device between the second and third clock edges (refer to Figure 4.1-7). Simply, if RDYIN is high on the second clock edge, the Q80 will raise the memory read control signal on the third clock edge assuming it has received valid data. If the RDYIN signal is lowered *after* the second clock edge the Q80 will not recognize it and continue on. If only one clock cycle is required for the wait period, the accessed device must raise the RDYIN signal by the third falling clock edge, one falling clock edge latter the Q80 will raise the memory read control signal thus completing the read transaction. The RDYIN signal can be held low for as long as required by the accessed device.

4.1.4.2 Memory, I/O Write Ready/Wait Handshake

Figure 4.1-9 illustrates the basic handshake transaction between the Q80 memory and I/O write control signals ($\overline{\text{MEMW}}$ and $\overline{\text{I/OW}}$) and the ready in (RDYIN) signal. The figure shows the memory (or $\overline{\text{I/O}}$) write control signal going

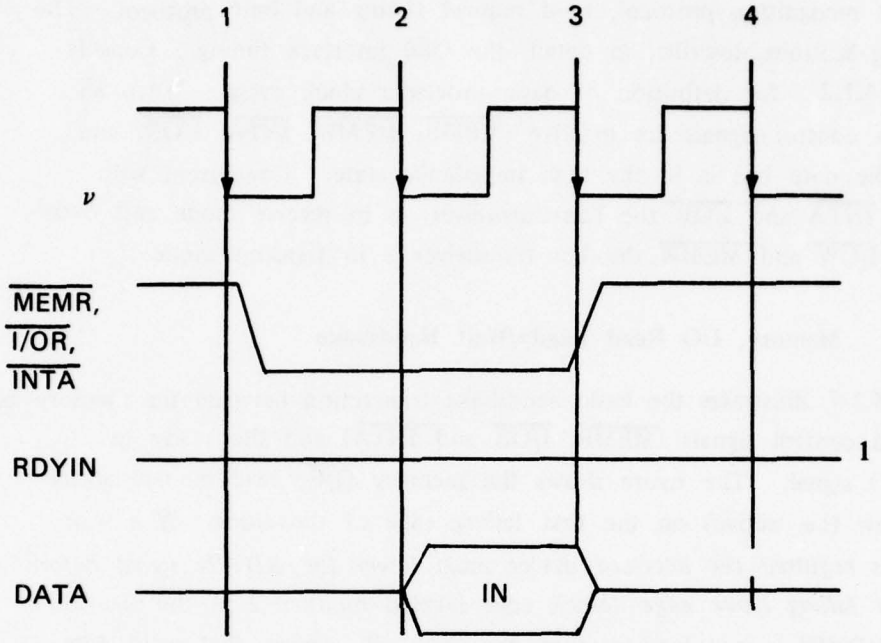


Figure 4.1-6 Memory, I/O Read – No Handshake

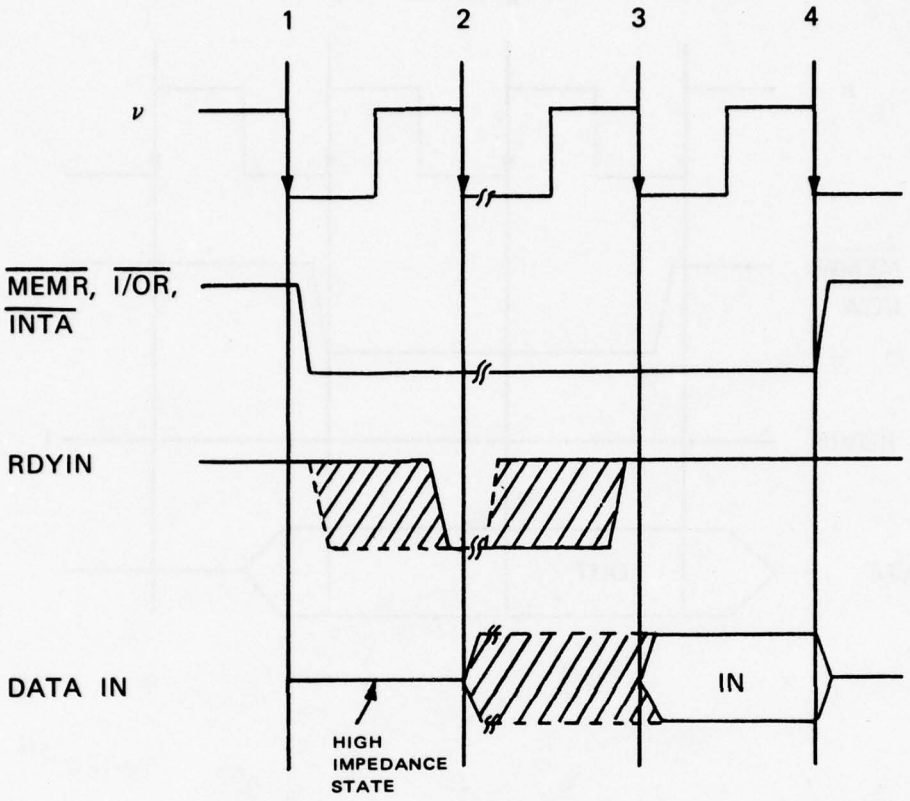


Figure 4.1-7 Memory, I/O READ READY/WAIT HANDSHAKE

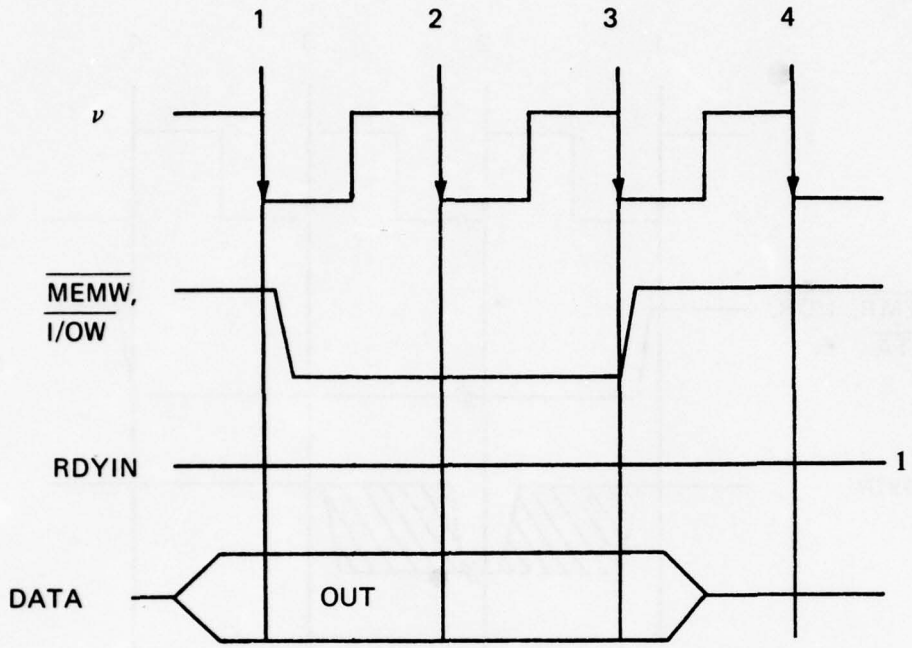


Figure 4.1-8 MEMORY, I/O WRITE—NO HANDSHAKE

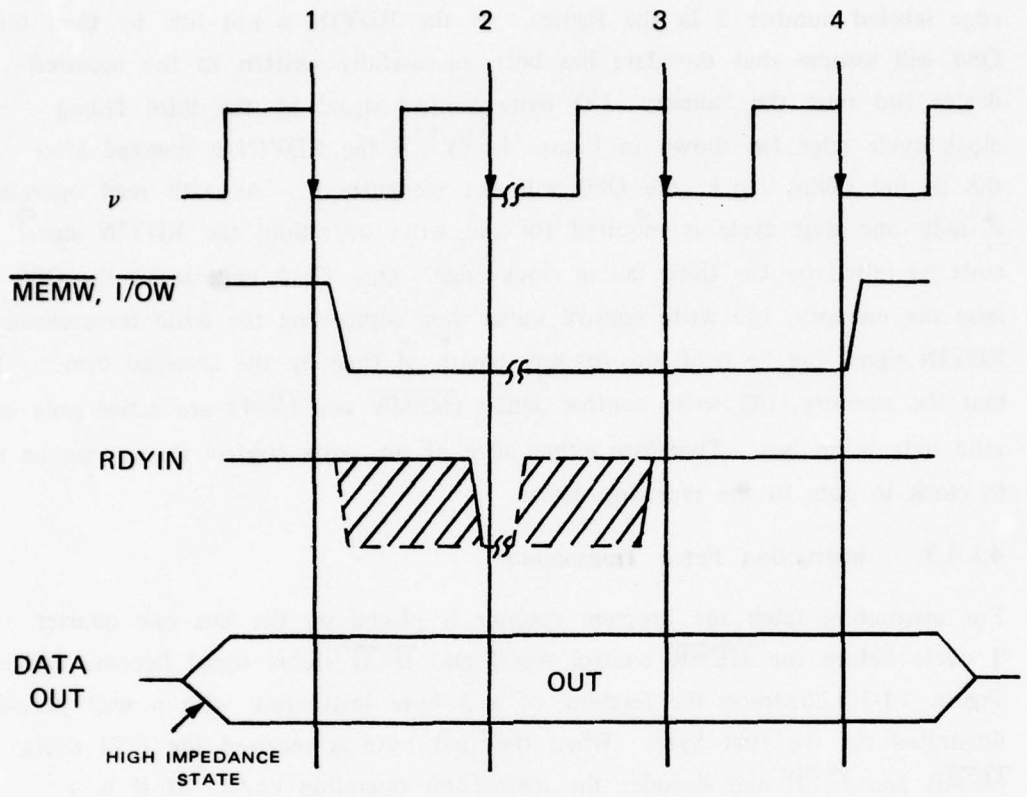


Figure 4.1-9 MEMORY, I/O WRITE READY/WAIT HANDSHAKE

low on the first falling clock edge. *If a wait period is required, the accessed device must lower the RDYIN signal before the next falling clock edge* (clock) edge labeled number 2 in the figure). If the RDYIN is not low by then the Q80 will assume that the data has been successfully written to the accessed device and raise the memory, I/O write control siganl by the third falling clock cycle edge (as shown in Figure 4.1-9). If the RDYIN is lowered after the second falling clock, the Q80 will not recognize it. As with read operations, if only one wait cycle is required for the write operation, the RDYIN signal must be raised by the third falling clock edge. One clock edge latter the Q80 will raise the memory, I/O write control signal thus completing the write transaction. The RDYIN signal can be held low for any length of time by the accessed device. Notice that the memory, $\overline{I/O}$ write control signals ($\overline{MEMW}$ and $\overline{I/OW}$) are active only when valid data is on bus. Therefore either edge of the write control signals can be used to clock in data to the receiving device.

4.1.4.3 Instruction Fetch Transaction

For instruction fetch the program counter is placed on the bus one quarter T cycle before the $\overline{MEMR}$ control signal and $\overline{IFCH}$ status signal become active. Figure 4.1-10 illustrates the fetching of a 3 byte instruction with a wait period illustrated for the first byte. When the first byte is received the Q80 raises $\overline{MEMR}$ and $\overline{IFCH}$ and decodes the instruction operation code. If it is a two or three byte instruction the program counter is incremented and placed on the bus one quarter T cycle before the $\overline{MEMR}$ and $\overline{IFCH}$ control signals are again actuated. And exactly the same protocol is followed for the third byte. With no wait cycles, a one byte instruction is fetched in 3 T cycles; a 2 byte instruction is fetched in 6 T cycles and a 3 byte instruction is fetched in 9 T cycles.

4.1.4.4 Memory Read and Write Transactions

Figure 4.1-11 illustrates the timing of 2 byte memory read and write operations. Each memory read is generally 3 to 4 cycles unless wait cycles are required. Likewise memory write transactions generally require 3 to 4 cycles unless wait cycles are required. If no wait cycles are required, the first byte can be fetched in 4 T cycles – 2 T cycles are required to set up the address and 2 T cycles are required to perform the bus transaction. The second byte can be fetched in 3 T

1 T cycle required to increment (or decrement) the address and 2 T cycles to perform the bus transaction. All single byte memory read transactions requrie 4 T cycles if no wait cycles are required. Exactly the same timing relationships are true for memory write transactions.

Memory read transactions are controlled by the Q80 $\overline{\text{MEMR}}$ signal. When this signal is active, and the $\overline{\text{IFCH}}$ signal is inactive, data information is being fetched from memory. It should be noted that $\overline{\text{STACK}}$ may be active if a stack pop or return type operation is being performed. Likewise, write operations are controlled by the $\overline{\text{MEMW}}$ signal – the $\overline{\text{IFCH}}$ signal is always inactive at this time. Also, if a stack push or call type operation is being performed the $\overline{\text{STACK}}$ signal will also be active.

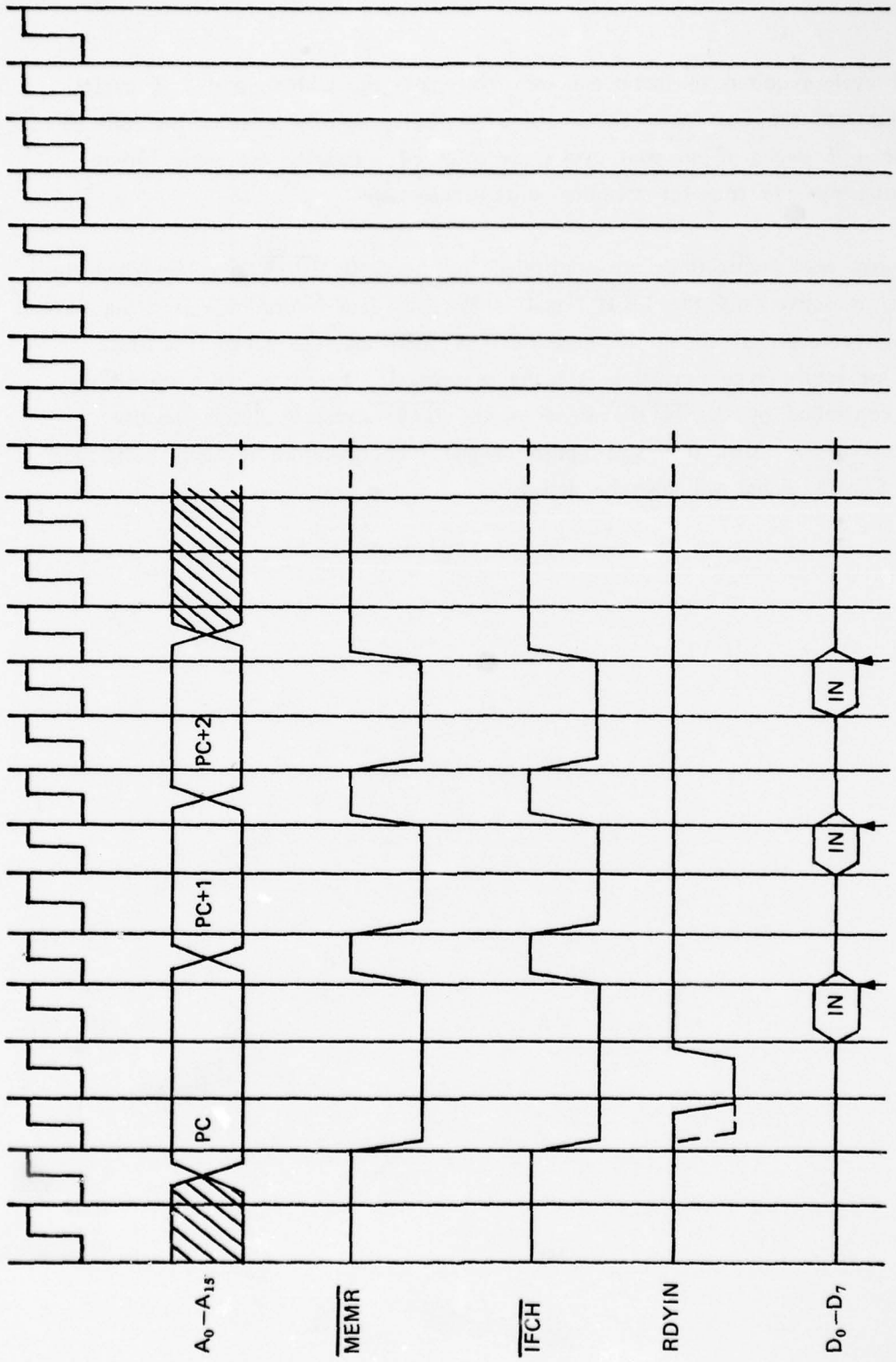


Figure 4.1-10 INSTRUCTION FETCH

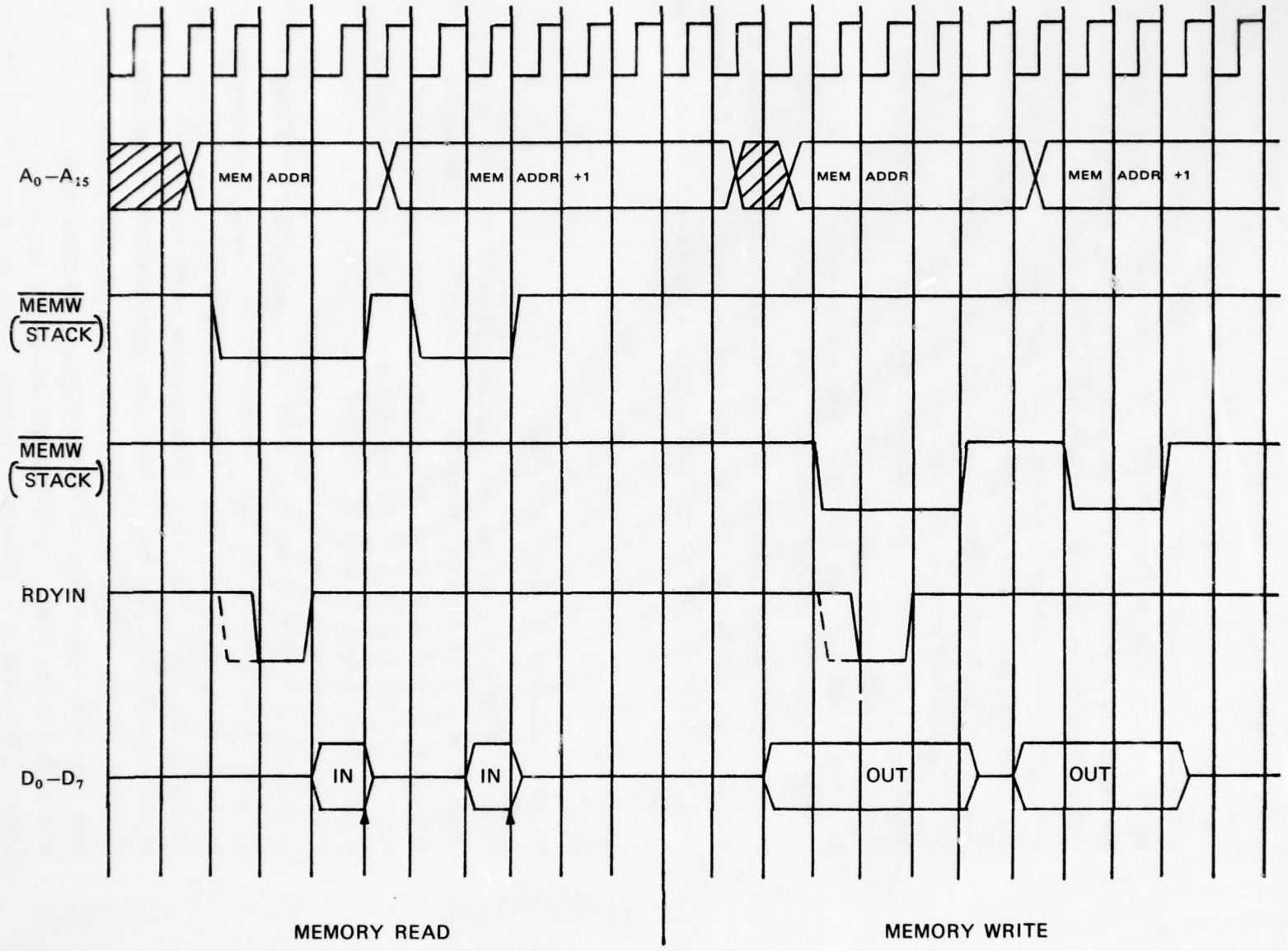

$A_0 - A_{15}$
MEM ADDR
MEM ADDR +1
MEM ADDR
MEM ADDR +1
$\overline{MEMW}$ ($\overline{STACK}$)
$\overline{MEMW}$ ($\overline{STACK}$)
RDYIN
$D_0 - D_7$
IN
IN
OUT
OUT
MEMORY READ
MEMORY WRITE

FIGURE 4.1-11

4.1.4.5 Input and Output Transactions

The timing protocol for input and output instructions is exactly the same as memory read and write operations except that $\overline{I/OR}$ and $\overline{I/OW}$ control the transactions (refer to Figure 4.1-12). The port number is placed on the least significant 8-bits (A_0- A_7) of the address bus – this requires 2 T cycles. Then for an input (read) operation $\overline{I/OR}$ is activated. If no wait cycles are required the bus transaction can be completed in 2 T cycles. The figure illustrates a read transaction with a wait cycle. When the peripheral device raises the RDYIN it must place the data, to be input, on the data bus. The Q80 will clock in the data in the next clock cycle. For an output (write) operation the port number is placed on the least significant 8-bits of the address bus – this requires 2 T cycles. Then the data to be output is placed on the data bus – this requires another T cycle. Then I/OW is activated for the bus transaction. This transaction can be completed in 2 T cycles if no wait cycles are required. As with memory transactions, if wait cycles are required, the RDYIN signal must be lowered by the first falling clock edge following $\overline{I/OR}$ or the Q80 will assume the transaction was completed successfully.

4.1.4.6 Interrupt Trapping Bus Protocol

Figure 4.1-13 illustrates the bus timing protocol associated with the recognition of an interrupt. The interrupt signal INT is sampled by the Q80 on the falling edge of the clock period prior to the last T cycle at the end of execution of any instruction. However, the interrupt will not be recognized if the internal, software controlled, interrupt enable flip-flop is not enabled. If INT is recognized, a special interrupt trapping sequence is enterred in which an interrupt instruction is fetched from the interrupting device. The Q80 first places the program counter value (for the next instruction to sequence be executed) on the address bus, then the $\overline{INTA}$ and $\overline{IFCH}$ signals are activated. On recognizing the $\overline{INTA}$ signal the interrupting device places the first byte of the interrupt instruction on the bus. The figure illustrates a wait cycle handshake for the fetching of this first byte. Then the Q80 proceeds to fetch two more bytes irregardless of the interrupt instruction length. If it is a single byte instruction the Q80 simply ignores the last too bytes it fetched. The point is it makes three requests to the interrupting device for

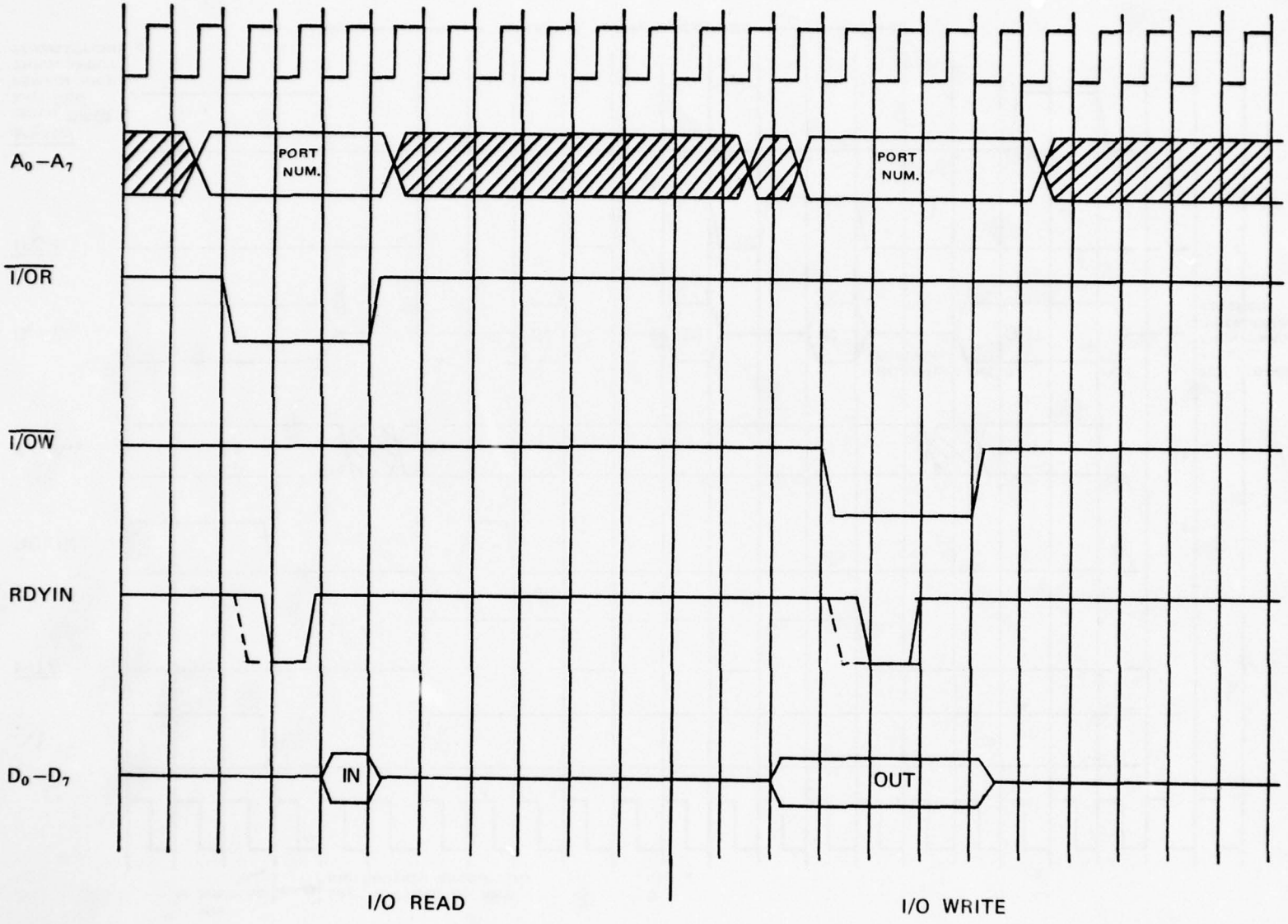

$A_0 - A_7$
PORT NUM.
PORT NUM.
$\overline{I/OR}$
$\overline{I/OW}$
RDYIN
$D_0 - D_7$
IN
OUT
I/O READ
I/O WRITE

Figure 4.1-12

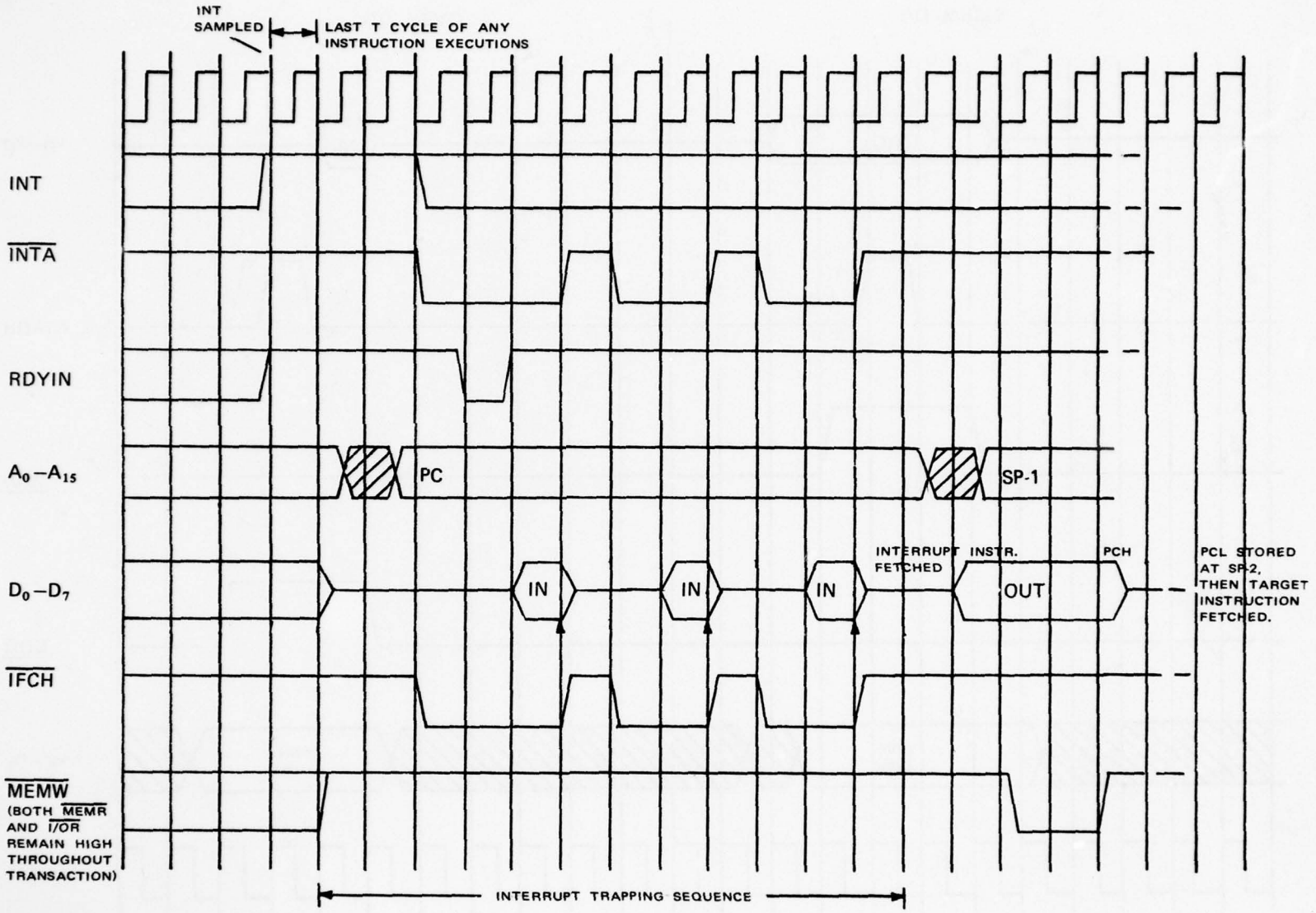


Figure 4.1-13 INTERRUPT PROTOCOL

each interrupt. When the three requests have been made, the Q80 proceeds to decode and execute the instruction. The interrupt trapping sequence also disables the interrupt recognition logic. All bus control signals (except INTA and IFCH) remain inactive during the interrupt trapping sequence. If wait cycles are required, the interrupting device must lower the RYDIN signal before the first falling clock edge after each $\overline{\text{INTA}}$ activation; otherwise the Q80 will assume it successfully fetched an interrupt instruction byte.

4.1.4.7 Hold Acknowledge Protocol

The hold signal is sampled on the falling edge of the clock period prior to the last T cycle at the end of execution of any instruction. If hold and INT are present concurrently the hold will take precedance over INT; i.e. not until the hold state is exited will the interrupt be processed. When hold is recognized the Q80 generates the HLDA signal and concurrently places all control and status signals, address and data buses into the high impedance state. During this time (as long as the hold signal is held high by the holding device) the Q80 bus can be utilized for DMA transfer. The Q80 continues to sample the hold signal while in the hold state. On the first falling edge of ν following the deactivation of the hold signal the Q80 recognizes it and one T cycle latter the Q80 will resume its normal operations.

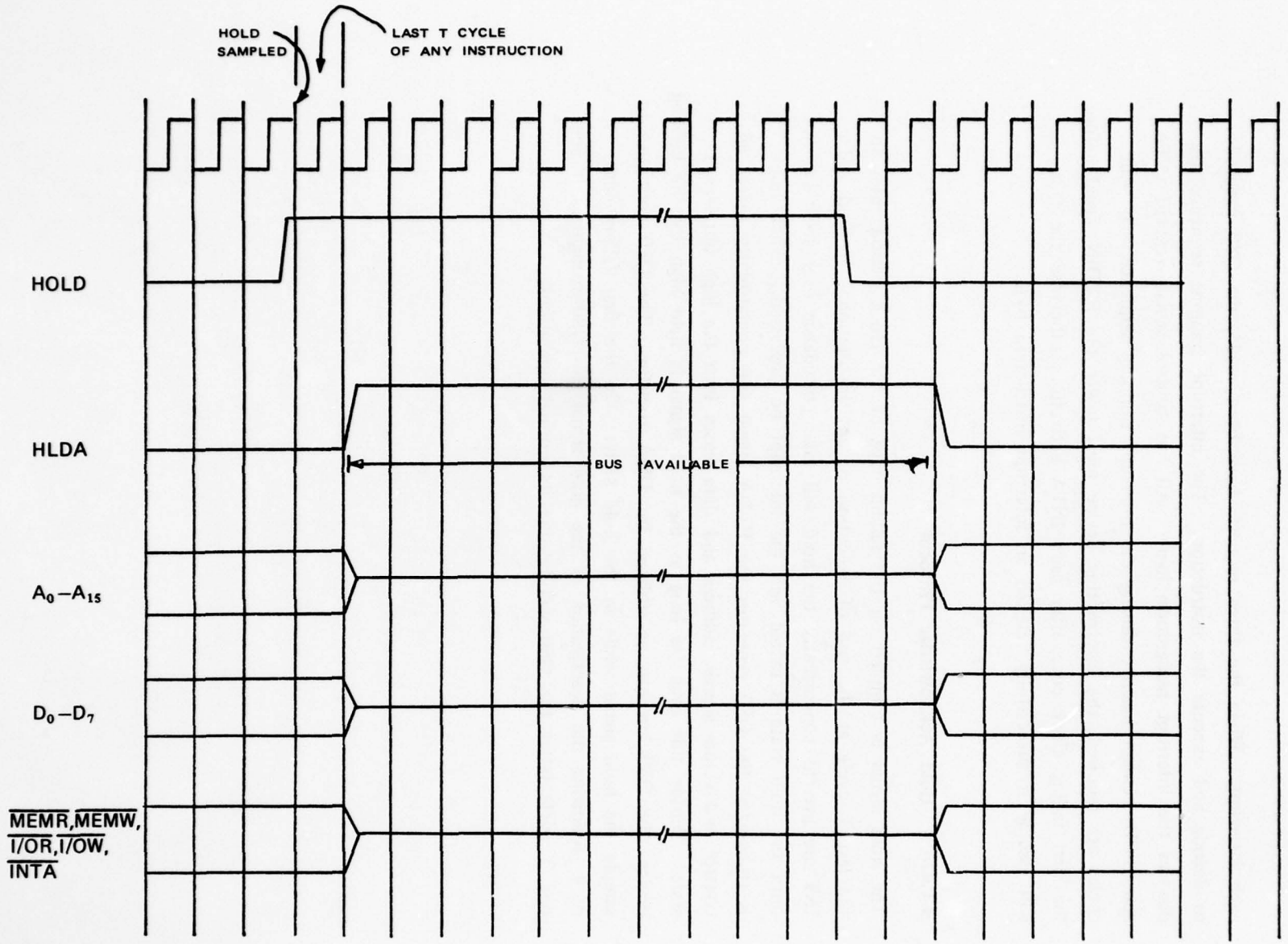


Figure 4.1-14 HOLD PROTOCOL

4.1.5 Instruction Set

4.1.5.1 Summary

The Q80 executes a total of 76 instructions; these instructions are segregated into five different classes as follows:

- Data Transfer Group–move data between registers or between memory and registers.
- Arithmetic Group–add, subtract, increment or decrement data in registers or in memory.
- Logical Group–AND, OR, EXCLUSIVE-OR, compare, rotate or complement data in registers or in memory.
- Branch Group–conditional and unconditional jump instructions, subroutine call instructions and return instructions.
- Stack, I/O and Machine Control Group–includes I/O instructions, as well as instructions for maintaining the stack and internal control flags.

The Q80 exactly emulates the 8080 microprocessor instruction set except in one case – it does not implement the DAA (Decimal Adjust Accumulator) instruction. There are two overriding reasons why this instruction is not implemented. First, the decimal capability as implemented by the 8080 is incomplete (only decimal addition can be performed) and therefore of marginal value. Second, the applications spectrum for the Q80 (namely, Aerospace) places very little value on packed decimal capability. The hardware penalty for providing this questionable decimal capability was judged to be too great. Because the decimal capability is not implemented the Auxilliary Carry Flag (AC) is not implemented in the Q80 (since it is used exclusively by the DAA instruction). The Q80 implements a Program Discreet Interface Signal (PDIS) instruction for the DAA operation code. The PDIS instruction toggles an interface signal. This capability is extremely useful in performing program performance evaluation and for general interfacing use.

4.1.5.2 Instruction and Data Formats

Memory for the Q80's is organized into 8-bit quantities, called Bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory. The Q80 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

Data in the Q80 is stored in the form of 8-bit binary integers:

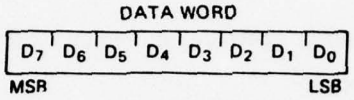


FIGURE 4.1-15 DATA FORMAT

When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the Q80 BIT 0 is referred to as the Least Significant Bit (LSB), and BIT 7 (of an 8 bit number) is referred to as the Most Significant Bit (MSB).

The Q80 program instructions may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.

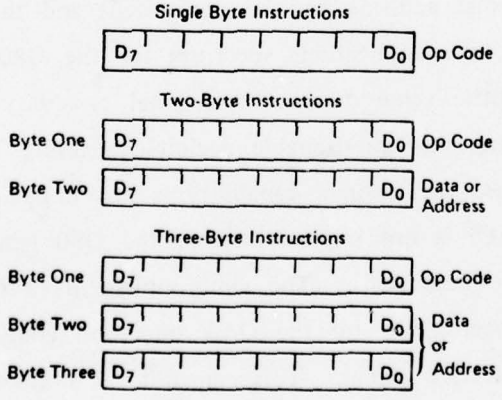


FIGURE 4.1-16 INSTRUCTION FORMATS

4.1.5.3 Addressing Modes

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The Q80 has four different modes for addressing data stored in memory or in registers:

- Direct – Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).
- Register – The instruction specifies the register or register-pair in which the data is located.
- Register Indirect – The instruction specifies a register-pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).
- Immediate – The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- Direct – The branch instruction contains the address of the next instruction to be executed. (Except for the 'RST' instruction, byte 2 contains the low-order address and byte 3 the high-order address).
- Register indirect – The branch instruction indicates a register-pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second).

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences). RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

4.1.5.4 Symbols and Abbreviations

The symbols and abbreviations used in the subsequent description of the Q80 instructions are summarized in Table 4.1-2.

Table 4.1-2 Symbols and Abbreviations

SYMBOLS	MEANING
accumulator	Register A
addr	16-bit address quantity
data	8-bit data quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r, r1, r2	One of the registers A,B,C,D,E,H,L
DDD,SSS	The bit pattern designating one of the registers A,B,C,D,E,H,L (DDD=destination, SSS=source):

DDD or SSS	REGISTER NAME
111	A
000	B
001	C
010	D
011	E
100	H
101	L

Table 4.1-2 (cont.) Symbols and Abbreviations

rp One of the register pairs:

B represents the B,C pair with B as the high-order register and C as the low-order register;

D represents the D,E pair with D as the high-order register and E as the low-order register;

H represents the H,L pair with H as the high-order register and L as the low-order register;

SP represents the 16-bit stack pointer register.

RP The bit pattern designating one of the register pairs B,D,H,SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh The first (high-order) register of a designated register pair.

rl The second (low-order) register of a designated register pair.

PC 16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively).

SP 16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively).

r_m Bit m of the register r (bits are number 7 through 0 from left to right).

Table 4.1-2 (cont.) Symbols and Abbreviations

Z,S,P,CY	The condition flags: Zero, Sign, Parity, Carry
( )	The contents of the memory location or registers enclosed in the parentheses.
←	"Is transferred to"
∧	Logical AND
∀	Exclusive OR
∨	Inclusive OR
+	Addition
–	Two's complement subtraction
*	Multiplication
↔	"Is exchanged with"
—	The one's complement (e.g., $(\overline{A})$)
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7 respectively.

Instruction execution times are calculated based on the general formula:

$$\text{Execution Time} = C + \left(\frac{\text{Macc}}{T}\right) N$$

C is a constant number of processor cycles required to perform the operation For conditional type instructions the general equation form becomes:

$$\text{Execution Time} = {}^{C}/_{C'} + \left(\frac{\text{Macc}}{T}\right) N$$

C is the number of processor cycles required if condition is not satisfied and C' is the number of processor cycles required if condition is satisfied. Macc is the memory access time and T is the Q80 processor period. The $^{\text{Macc}}/_{T}$ term calculates the number of processor cycles that are required for memory accesses. If Macc is less than T no processor cycles are required. N is the number of memory requests made during the execution of the instruction.

4.1.5.5 Instruction Descriptions

The following sections present brief discussions of Q80 instructions. For more information consult Intel document "8080/8085 Assembly Language Programming Manual", 98-301A.

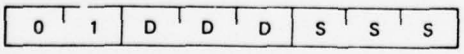
4.1.5.5.1 Data Transfer Group

This group of instructions transfers data to and from registers and memory. Condition flags are not affected by any instruction in this group.

MOV r1, r2 (Move Register)

(r1) ← (r2)

The content of register r2 is moved to register r1.



Cycles: $5 + \left\lfloor {}^{\text{Macc}}/_{T} \right\rfloor$

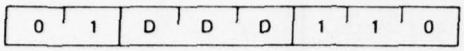
Addressing: register

Flags: none

MOV r, M (Move from memory)

(r) ← ((H) (L))

The content of the memory location, whose address is in registers H and L, is moved to register r.



Cycles: 7 + $\left(\text{Macc}/\text{T}\right\rfloor$ 2

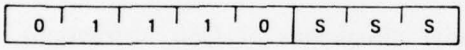
Addressing: reg. indirect

Flags: none

MOV M, r (Move to memory)

((H) (L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.



Cycles: 7 + $\left(\text{Macc}/\text{T}\right\rfloor$ 2

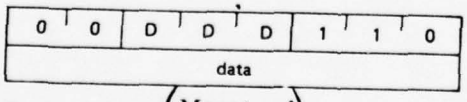
Addressing: reg. indirect

Flags: none

MVI, data (Move immediate)

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.



Cycles: 7 + $\left(\text{Macc}/\text{T}\right\rfloor$ 2

Addressing: immediate

Flags: none

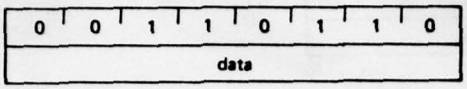
AD-A063 003 QUESTRON CORP SAN DIEGO CALIF F/G 9/2
RADIATION HARDENED MICROPROCESSOR VOLUME I.(U)
MAY 78 V V NICKEL, P A ROSENBERG F33615-77-C-1001
UNCLASSIFIED AFAL-TR-78-55-VOL-1 NL

3 OF 4
AD
A063003

MVIM, data (Move to memory immediate)

((H) (L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.



Cycles: 10 + Macc

Addressing: immed./reg. indirect

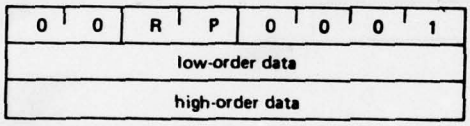
Flags: none

LXI rp, data 16 (Load register pair immediate)

(rh) ← (byte 3),

(rl) ← (byte 2)

Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.



Cycles: 10 + Macc/T 3

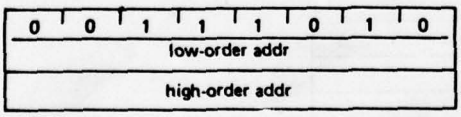
Addressing: immediate

Flags: none

LDA addr (Load Accumulator direct)

(A) ← ((byte 3) (byte 2))

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.



Cycles: 13

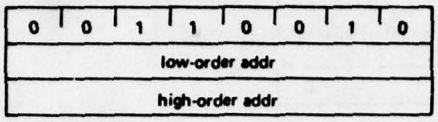
Addressing: direct

Flags: none

STA addr (Store Accumulator direct)

((byte 3) (byte2)) ← (A)

The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.



Cycles: 13

Addressing: direct

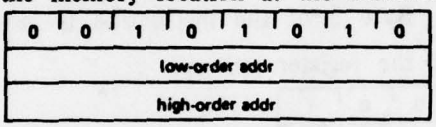
Flags: none

LHLD addr (Load H and L direct)

(L) ← ((byte 3) (byte 2))

(h) ← ((byte 3) (byte 2) +1)

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.



Cycles: 16

Addressing: direct

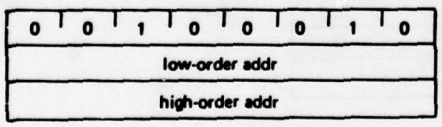
Flags: none

SHLD addr (Store H and L direct)

((byte 3) (byte 2)) ← (L)

((byte 3) (byte 2) +1) ← (H)

The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.



Cycles: 16

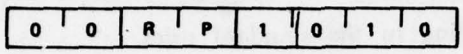
Addressing: direct

Flags: none

LDAX rp (Load accumulator indirect)

(A) ← ((rp))

The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.



Cycles: 7

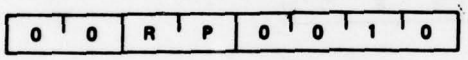
Addressing: reg. indirect

Flags: none

STAX rp (Store accumulator indirect)

((rp)) ← (A)

The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.



Cycles: 7

Addressing: reg. indirect

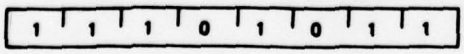
Flags: none

XCHG (Exchange H and L with D and E)

(H) ↔ (D)

(L) ↔ (E)

The contents of registers H and L are exahnged with the contents of registers D and E.



Cycles: 4

Addressing: register

Flags: none

4.1.5.5.2 Arithmetic Group

This group of instructions performs arithmetic operations on data in registers and memory.

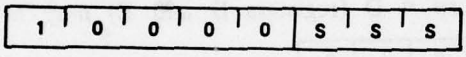
Unless indicated otherwise all instructions in this group affect the Zero, Sign, Parity, and Carry according to the standard rules.

All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

ADD r (Add Register)

(A) ← (A) + (r)

The content of register r is added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 4

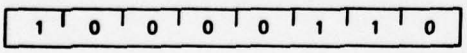
Addressing: register

Flags: Z,S,P,CY

ADD M (Add memory)

(A) ← (A) + ((H) (L))

The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 7

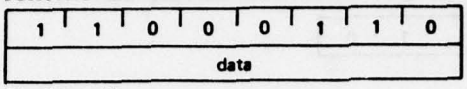
Addressing: reg. indirect

Flags: Z,S,P,CY

ADI data (Add immediate)

(A) ← (A) + (byte 2)

The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 7

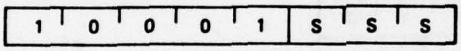
Addressing: immediate

Flags: Z,S,P,CY

ADC r (Add Register with carry)

(A) ← (A) + (r) + (CY)

The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 4

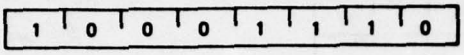
Addressing: register

Flags: Z,S,P,CY

ADC M (Add memory with carry)

(A) ← (A) + ((H) (L)) + (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.



Cycles: 7

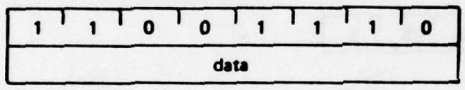
Addressing: reg. indirect

Flags: Z,S,P,CY

ACI data (Add immediate with carry)

(A) ← (A) + (byte 2) + (CY)

The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.



Cycles: 7

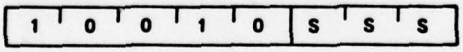
Addressing: immediate

Flags: Z,S,P,CY

SUB r (Subtract Register)

(A) ← (A) – (r)

The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.



Cycles: 4

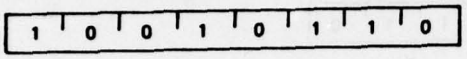
Addressing: register

Flags: Z,S,P,CY

SUB M (Subtract memory)

(A) ← (A) – ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.



Cycles: 7

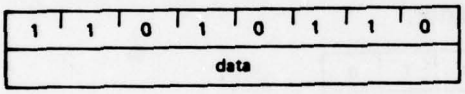
Addressing: reg. indirect

Flags: Z,S,P,CY

SUI data (Subtract immediate)

(A) ← (A) – (byte 2)

The content of the second byte of the instruction is subtracted from the content of the accumulator. The results is placed in the accumulator.



Cycles: 7

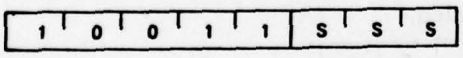
Addressing: immediate

Flags: Z,S,P,CY

SBB r (Subtract Register with borrow)

(A) ← (A) – (r) – (CY)

The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



Cycles: 4

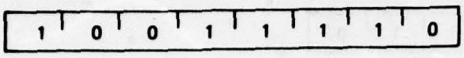
Addressing: register

Flags: Z,S,P,CY

SBB M (Subtract memory with borrow)

(A) ← (A) – ((H) (L)) – (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



Cycles: 7

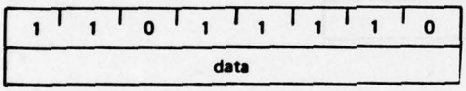
Addressing: reg. indirect

Flags: Z,S,P,CY

SBI data (Subtract immediate with borrow)

(A) ← (A) – (byte 2) – (CY)

The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



Cycles: 7

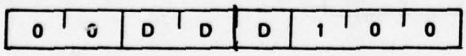
Addressing: immediate

Flags: Z,S,P,CY

INR r (Increment Register)

(r) ← (r) + 1

The content of register r is incremented by one. Note: All condition flags except CY are affected.



Cycles: 5

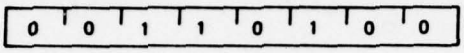
Addressing: register

Flags: Z,S,P

INR M (Increment memory)

((H) (L)) ← ((H) (L)) + 1

The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags except CY are affected.



Cycles: 10

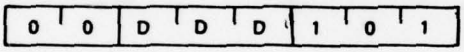
Addressing: reg. indirect

Flags: Z,S,P

DCR r (Decrement Register)

(r) ← (r) – 1

The content of register r is decremented by one. Note: All condition flags except CY are affected.



Cycles: 5

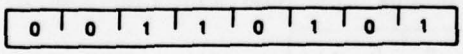
Addressing: register

Flags: Z,S,P

DCR M (Decrement memory)

((H) (l)) ← ((H) (L)) – 1

The content of the memory location whose addres is contained in the H and L registers is decremented by one. Note: All condition flags except CY are affected.



Cycles: 10

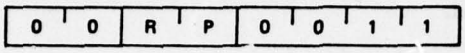
Addressing: reg. indirect

Flags: Z,S,P

INX rp (Increment register pair)

(rh) (rl) ← (rh) (rl) + 1

The content of the register pair rp is incremented by one. Note: No condition flags are affected.



Cycles: 5

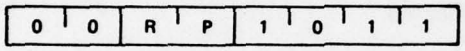
Addressing: register

Flags: none

DCX rp (Decrement register pair)

(rh) (rl) ← (rh) (rl) – 1

The content of the register pair rp is decremented by one. Note: No condition flags are affected.



Cycles: 5

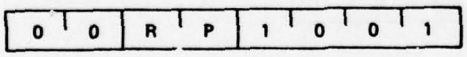
Addressing: register

Flags: none

DAD rp (Add register pair to H and L)

(H) (L) ← (H) (L) + (rh) (rl)

The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: **Only the CY flag is affected.** It is set if there is a carry out of the double precision add; otherwise it is reset.



Cycles: 10

Addressing: register

Flags: CY

4.1.5.5.3 Logical Group

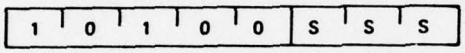
This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, and Carry flags according to the standard rules.

ANA r (AND Register)

(A) ← (A) ∧ (r)

The content of register r is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared.**



Cycles: 4

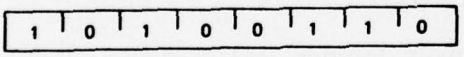
Addressing: register

Flags: Z,S,P,CY

ANA M (AND memory)

(A) ← (A) ∧ ((H) (L))

The contents of the memory location whose address is contained in the H and L registers is logically anded with the content of the accumulator. The results is placed in the accumulator. **The CY flag is cleared.**



Cycles: 7

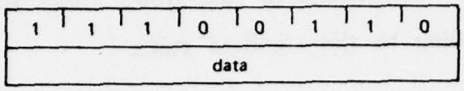
Addressing: reg. indirect

Flags: Z,S,P,CY

ANI data (AND immediate)

(A) ← (A) ∧ (byte 2)

The content of the second byte of the instruction is logically anded with the contents of the accumulator. The result is placed in the accumulator. **The CY flag cleared.**



Cycles: 7

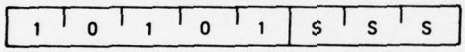
Addressing: immediate

Flags: Z,S,P,CY

XRA r (Exclusive OR Register)

(A) ← (A) ∨ (r)

The content of register r is exclusive-or'd with the content of the accumulator. The result is placed in the accumulator. **The CY flag cleared.**



Cycles: 4

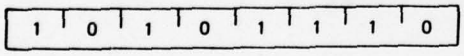
Addressing: register

Flags: Z,S,P,CY

XRA M (Exclusive OR Memory)

(A) ← (A) ∨ ((H) (L))

The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY flag cleared.**



Cycles: 7

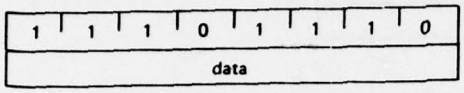
Addressing: reg. indirect

Flags: Z,S,P,CY

XRI data (Exclusive OR immediate)

(A) ← (A) ∨ (byte 2)

The content of the second byte of the instruction is exclusive-OR'd with the current of the accumulator. The result is placed in the accumulator. **The CY flag cleared.**



Cycles: 7

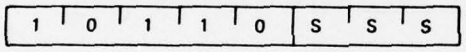
Addressing: immediate

Flags: Z,S,P,CY

ORA r (OR Register)

(A) ← (A) ∨ (r)

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY flag cleared.**



Cycles: 4

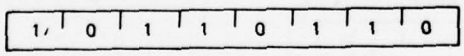
Addressing: register

Flags: Z,S,P,CY

ORA M (OR memory)

(A) ← (A) ∨ ((H) (L))

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY flag cleared.**



Cycles: 7

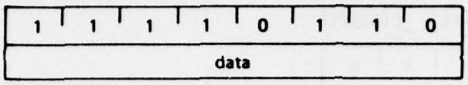
Addressing: reg. indirect

Flags: Z,S,P,CY

ORI data (OR immediate)

(A) ← (A) ∨ (byte 2)

The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY flag cleared.**



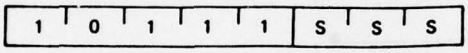
Cycles: 7

Addressing: immediate

Flags: Z,S,P,CY

CMP r (Compare Register

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = (r). The CY flag is set to 1 if (A) < (r).**



Cycles: 4

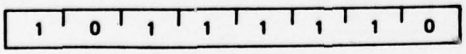
Addressing: register

Flags: Z,S,P,CY

CMP M (Compare memory)

(A) – ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a results of the subtraction. The Z flag is set to 1 if (A) = ((H) (L)). The CY flag is set to 1 if (A) < ((H) (L)).



Cycles: 7

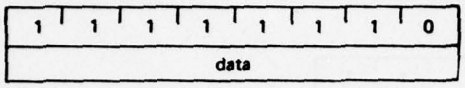
Addressing: reg. indirect

Flags: Z,S,P,CY

CPI data (Compare immediate)

(A) – (byte 2)

The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).



Cycles: 4

Addressing: immediate

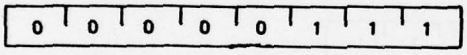
Flags: Z,S,P,CY

RLC (Rotate left)

$(A_n+1) \leftarrow (A_n);(A_0) \leftarrow (A_7)$

$(CY) \leftarrow (A_7)$

The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. **Only the CY flag is affected.**



Cycles: 4

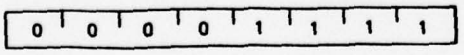
Flags: CY

RRC (Rotate right)

$(A_n) \leftarrow (A_n-1);(A_7) \leftarrow (A_0)$

$(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. **Only the CY flag is affected.**



Cycles: 4

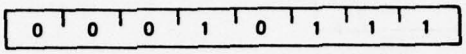
Flags: CY

RAL (Rotate left through carry)

$(A_n+1) \leftarrow (A_n)$;$(CY) \leftarrow (A_7)$

$(A_0) \leftarrow (CY)$

The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. **Only the CY flag is affected.**



Cycles: 4

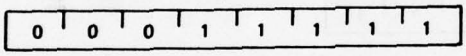
Flags: CY

RAR (Rotate right through carry)

$(A_n) \leftarrow (A_n+1)$;$(CY) \leftarrow (A_0)$

$(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. **Only the CY flag is affected.**



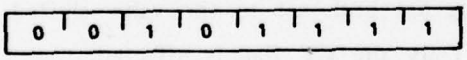
Cycles: 4

Flags: CY

CMA (Complement accumulator)

$(A) \leftarrow (\bar{A})$

The contents of the accumulator are complemented (zero bits become 1, one bits become 0). **No flags are affected.**



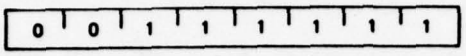
Cycles: 4

Flags: none

CMC (Complement carry)

(CY) ← ($\overline{\overline{CY}}$)

The CY flag is complemented. **No other flags are affected.**



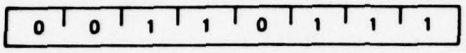
Cycles: 4

Flags: CY

STC (Set carry)

(CY) ← 1

The CY flag is set to 1. **No other flags are affected.**



Cycles: 4

Flags: CY

4.1.5.5.4 Branch Group

This group of instructions alter normal sequential program flow.

Condition flags are not affected by any instruction in this group.

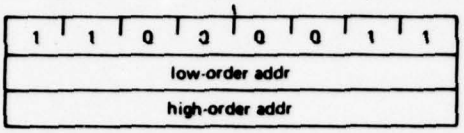
The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

CONDITION	CCC
NZ – not zero (Z=0)	000
Z – zero (Z=1)	001
NC – no carry (CY=0)	010
C – carry (CY= 1)	011
PO – parity odd (P=0)	100
PE – parity even (P=1)	101
P – plus (S=0)	110
M – minus (S=1)	111

JMP addr (Jump)

(PC) ← (byte 3) (byte 2)

Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



Cycles: 10

Addressing: immediate

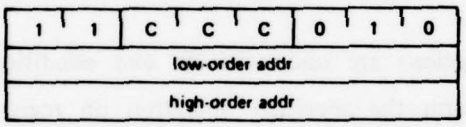
Flags: none

Jcondition addr (Conditional jump)

If (CCC),

(PC) ← (byte 3) (byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.



Cycles: 10

Addressing: immediate

Flags: none

CALL addr (Call)

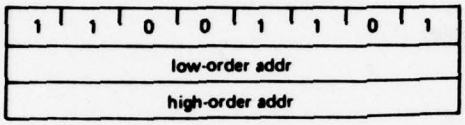
((SP) −1) ← (PCH)

((SP) −2) ← (PCL)

(SP) ← (SP) −2

(PC) ← (byte 3) (byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



Cycles: 17

Addressing: immediate/reg. indirect

Flags: none

Ccondition addr (Condition call)

If (CCC),

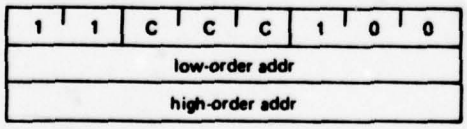
((SP) −1) ← (PCH)

((SP) −2) ← (PCL)

(SP) ← (SP) −2

(PC) ← (byte 3) (byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.



Cycles: 11/17

Addressing: immediate/reg. indirect

Flags: none

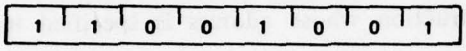
RET (Return)

(PCL) ← ((SP));

(PCH) ← ((SP) + 1);

(SP) ← (SP) + 2;

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.



Cycles: 10

Addressing: reg. indirect

Flags: none

Rcondition (Conditional return)

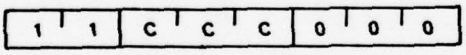
IF (CCC),

(PCL) ← ((SP))

(PCH) ← ((SP) + 1)

(SP) ← (SP) + 2

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.



Cycles: 5/11

Addressing: reg. indirect

Flags: none

RST n (Restart)

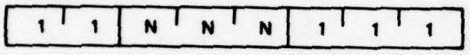
((SP) −1) ← (PCH)

((SP) −2) ← (PCL)

(SP) ← (SP) −2

(PC) ← 8 * (NNN)

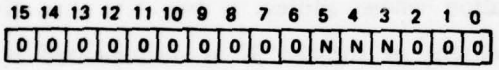
The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.



Cycles: 11

Addressing: reg. indirect

Flags: none



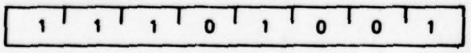
Program Counter After Restart

PCHL (Jump H and L indirect – move H and L to PC

(PCH) ← (H)

(PCL) ← (L)

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.



Cycles: 5

Addressing: register

Flags: none

4.1.5.5.5 Stack, I/O, and Machine Control Group

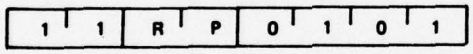
This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, **condition flags are not affected by any instructions in this group.**

PUSH rp (Push)

((SP) −1) ← (rh)
((SP) −2) ← (rl)
(SP) ← (SP) −2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **Note: Register pair rp = SP may not be specified.**



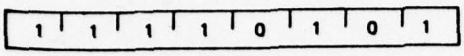
Cycles: 11
Addressing: reg. indirect
Flags: none

PUSH PSW (Push processor status word)

((SP) −1) ← (A)
((SP) $-2)_0$ ← (CY), ((SP) $-2)_1$ ← 1
((SP) $-2)_2$ ← (P), ((SP) $-2)_3$ ← 0
((SP) $-2)_4$ ← 0, ((SP) $-2)_5$ ← 0VF
((SP) $-2)_6$ ← (Z), ((SP) $-2)_7$ ← (S)
(SP) ← (SP) −2

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are

assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.



Cycles: 11

Addressing: reg. indirect

Flags: none

WORD

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	OVF	0	0	P	1	CY

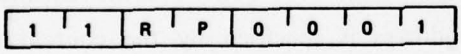
POP rp (Pop)

(rl ← ((SP))

(rh) ← ((SP) + 1)

(SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register or register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. **Note: Register pair rp = SP may not be specified.**



Cycles: 10

Addressing: reg. indirect

Flags: none

POP PSW (Pop processor status word)

$(CY) \leftarrow ((SP))_0$

$(P) \leftarrow ((SP))_2$

$(AC) \leftarrow ((SP))_4$

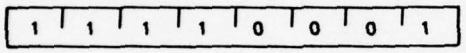
$(Z) \leftarrow ((SP))_6$

$(S) \leftarrow ((SP))_7$

$(A) \leftarrow ((SP) + 1)$

$(SP) \leftarrow (SP) + 2$

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.



Cycles: 10

Addressing: reg. indirect

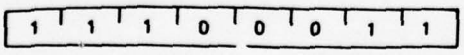
Flags: Z,S,P,CY

XTHL (Exchange stack top with H and L)

$(L) \leftrightarrow ((SP))$

$(H) \leftrightarrow ((SP) + 1)$

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.



Cycles: 18

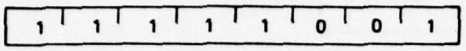
Addressing: reg. indirect

Flags: none

SPHL (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.



Cycles: 5

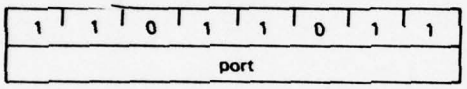
Addressing: register

Flags: none

IN port (Input)

(A) ← (data)

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A.



Cycles: 10

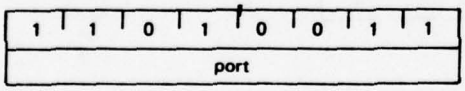
Addressing: direct

Flags: none

OUT port (Output)

(data) ← (A)

The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.



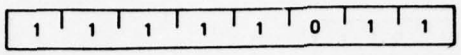
Cycles: 10

Addressing: direct

Flags: none

EI (Enable interrupts)

The interrupt system is enabled following the **execution of the next instruction.**

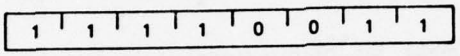


Cycles: 4

Flags: none

DI (Disable interrupts)

The interrupt system is disabled immediately following the **execution of the DI instruction.**

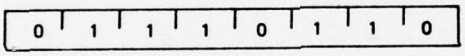


Cycles: 4

Flags: none

HLT (Halt)

The processor is stopped. The registers and flags are unaffected.

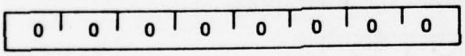


Cycles: 7

Flags: none

NOP (No op)

No operation is performed. The registers and flags are unaffected.

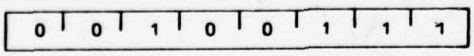


Cycles: 4

Flags: none

PDIS (Program Discrete Interface Signal)

Toggles the PDIS signal on the system interface. Register and flag values are unaffected.



Cycles: 6

Flags: none

4.1.6 Flags

There are four condition flags associated with the execution of instructions on the Q80. They are Zero, Sign, Parity, and Carry, and are each represented by a 1-bit register in the Q80. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

Zero(Z): If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

Sign(S): If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.

Parity(P): If the module 2 sum of the bits of the result of the operation is 0, (i.e., if the result has even parity), this flag is set; otherwise it is reset (i.e., if the result has odd parity).

Carry(CY): If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

The Q80 provides an additional status output on execution of instructions – it is an overflow indication (OVF). This status flag is set in the PSW as the result of a Push PSW and it is also output on the system interface. It is not a condition flag since it cannot be directly branched on. However, the OVF signal on the interface can be used to trigger an interrupt thus facilitating software handling of overflow situations. The condition flags and the OVF status signal are formatted in the PSW as shown in Figure 4.1-17

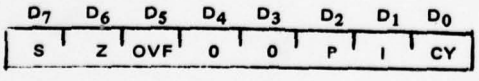


Figure 4.1-17 PSW Format

Table 4.1-3 Setting of Flags

	S	Z	OVF	Ø	Ø	P	I	C
ARITHMETIC								
ADDr, ADDm, ADI, ADCr, ADCm, ACI, SUBr, SUBM, SUI, SBBr, SBBM, SBI	↕	↕	↕	0	Ø	↕	I	↕
INRr, INRM, DCRr, DCRM	↕	↕	•	0	Ø	↕	I	•
DAD	•	•	↕	0	Ø	•	I	↕
LOGICAL								
ANAr, ANAM, ANI, XRAr, XRAM, XRI, ORAr, ORAM, ORI	↕	↕	0	0	Ø	↕	I	0
CMPr	↕	↕	↕	0	Ø	↕	I	↕
RLC, RRC, RAL, RAR	•	•	↕	0	Ø	•	I	↕
CMC	•	•	↕	0	Ø	•	I	I↔0
STC	•	•	I	0	Ø	•	I	I

The OVF status signal is enabled to be set or reset whenever the CY flag is enabled. Likewise it is cleared whenever CY is cleared or reset and set whenever CY is set.

Table 4.1-3 lists how each flag bit is affected by Q80 instructions. In this table a '•' indicates that the instruction does not change the flag, a '0' means that it is reset, a '1' means that it is set and the symbol '↕' indicates that it is set or reset according to the previous discussion. Note that any instruction not appearing in this table does not affect the flags. The slashed column in the table corresponding to the OVF status signal indicates that it is not a conditional branch flag.

4.1.7 Interrupts

The purpose of an interrupt is to allow peripheral devices to suspend Q80 operation in an orderly manner and force the Q80 to start a peripheral service routine. Usually this service routine is involved with the exchange of data, or status and control information, between the Q80 and the peripheral. Once the service routine is completed, the Q80 returns to the operation from which it was interrupted.

There is no limit to the number of interrupts the Q80 can handle – it is a system design consideration to configure Q80 software and hardware to handle the required interrupts. If more than one interrupt is required in a system some external hardware, namely interrupt priority hardware, is required by the Q80 – just as with the 8080. This interrupt priority hardware allows only one peripheral device to interrupt at a time.

The interrupt input (INT) to the Q80 is maskable – it can be selectively enabled or disabled by the programmer. This allows the programmer to disable the interrupts during periods where program timing constraints are too critical to allow it to be

interrupted. Disabling is accomplished with a DISABLE INTERRUPTS (DI) instruction. Interrupts are disabled immediately following the execution of this instruction; i.e., the INT signal will not be recognized. When interrupts are disabled the INTE signal on the interface is low (off). This signal can be used as a status signal by peripheral devices and/or the interrupt priority hardware. When the programmer desires to enable interrupts to be recognized, he executes an ENABLE INTERRUPTS (EI) instruction. Interrupts will be enabled immediately following this instruction and the INTE signal on the interface will become active (go high).

Interrupt timing on the interface is explained in detail in Section 4.1.3. Briefly the interrupting device can place any instruction on the data bus (when it sees the interrupt acknowledge signal) and the Q80 will execute it. Thus the interrupting device provides the next instruction to be executed. Often this instruction is a RESTART instruction since the interrupting device need only supply a single byte instruction. However, any other instruction, such as a three byte call to any locaiton, could be supplied. The Q80 automatically generates three interrupt acknowledge signals for each interrupt recognized, therefore instructions of one, two or three bytes can be supplied by the peripheral device. If it is supplying a single byte instruction, the Q80 simply ignores the next two acknowledges (refer to Section 4.1.3).

When an interrupt is recognized the interrupt input is automatically disabled by the Q80. Therefore, the interrupt handling routine need not disable interrupts. When it is time to recognize interrupts again the program executes an EI instruction.

4.1.8 State Diagram

The Q-80 Emulator has 24 states. Each of these states corresponds to a wait-loop in the microprogram of the Q-80 in which a stimulus is expected from the Q-80 environment.

Transitions between states are caused by the arrival of new stimuli from the environment of the Q-80. A state transition corresponds to the execution of microcode between two wait-loops. It begins with the exit from one loop, continues with the execution of non-iterative code and ends with entering another (perhaps the same) loop.

- **The state RESET**

This state is entered from any state if RESIN = 1. At power on the RESIN signal should be active until power is in tolerance to insure the Q-80 starts out in this state.

The program counter is set to 0 and the address and data buses are floated. Depending on the value of the stimulus RESIN, the next state is RESET or IFCH1; i.e. as soon as RESIN is deactivated IFCH1 is entered.

- **The state IFCH1**

In this state the opcode of the next instruction to be executed is fetched from memory. The value of the program counter is placed on the address bus and IFCH and MEMR are activated. After the op code is received, the program counter is incremented and OFL is cleared.

If the opcode fetched indicates that the instruction is a 2 or 3 byte instruction the next state is IFCH2. If it is a 1-byte instruction which requires an operand to be popped from the stack, the next

state is POP1 or COND*. If it is an instruction which requires an operand to be detched from memory (other than pop from the stack) the next state is OFCH1. If the instruction is the IN-instruction, the next state is INPUT. If it is a 1-byte instruction which does not require any sort of operand fetch, the instruction is executed at this point by a transition to the pseudo-state EXEC.

- **The state IFCH2**

In this state the 2nd byte of a 2 or 3-byte instruction is fetched from memory. The value of the program counter is placed on the address bus and IFCH and MEMR are activated. After the 2nd byte is received, the program counter is incremented.

If the opcode, fetched in IFCH1, indicates that the instruction is 3 bytes long the next state is IFCH3. Otherwise the next state is EXEC, OFCH1, POP1 or INPUT depending on the opcode (as in state IFCH1).

- **The state IFCH3**

In this state the 3rd byte of a 3-byte instruction is fetched from memory. The value of the program counter is placed on the address bus and IFCH and MEMR are activated. After the 3rd byte is received, the program counter is incremented. The next state is EXEC, OFCH1, POP1 depending on the opcode (as in state IFCH1).

- **The pseudo-state COND**

This pseudo-state is an intermediate step in the execution of conditional return instructions which serves to decide the next state after IFCH1. If the condition is true, the next state is POP1, otherwise it is the pseudo-state EXEC.

---

*The 8 conditional return instructions are a special case. The condition must be examined first, to determine whether to pop the stack. This is done in the pseudo-state COND. A pseudo-state is a state in which the Q-80 proceeds on its own to perform a function – no stimulus is required from the Q-80 external environment.

- **The pseudo-state COND**

In this state single-byte operands and the 1st bytes of 2-byte operands are fetched. Depending on the opcode one of the following is placed on the address bus:

- the register pair BC
- the register pair DE
- bytes 2 and 3 of the instruction
- the register pair HL,

and MEMR is activated. Depending on the opcode, the next state is either OFCH2 (2-byte operands) or EXEC (1-byte operands).

- **The state OFCH2**

In this state the 2nd byte of a 2-byte operand is fetched. The address used in OFCH1 is incremented and placed on the address bus. MEMR is activated. The next state is always the pseudo-state EXEC.

- **The state POP1**

In this state the 1st of 2 bytes (the LSB) are popped from the stack. The stack pointer (SP) is placed on the address bus and STACK and MEMR are activated. After the byte is received from memory the stack pointer is incremented. The next state is always POP2.

- **The state POP2**

In this state the 2nd of 2 bytes (the MSB) is popped from the stack. As in POP1, the stack pointer is placed on the address bus and STACK and MEMR are activated. After the byte is received from memory the stack pointer is incremented. The next state is always the pseudo-state EXEC.

- **The state INPUT**

In this state a byte is read from the port indicated by the 2nd byte of the instruction. The port number (2nd byte of the instruction) is placed on the address bus, padded with leading zeros. IOR is activated. The next state is always the pseudo-state EXEC.

- **The pseudo-state EXEC**

The pseudo-state EXEC represents the execution phase of the 256 Q-80 instructions. Direct transitions from the states IFCH1, IFCH2, IFCH3, OFCH1, OFCH2, POP2, and INPUT to the states STORE1, PUSH1, OUTPUT, HALTE, HOLD, HHOLD, HALTI, HALTIE, IHHOLD, IEHHOLD, INTJAM1 and IFCH1 all pass through this pseudo-state. The next state is determined as follows. If the opcode indicates that 2 bytes must be pushed onto the stack then the next state is PUSH1. If a 1 or 2-byte result must be stored in the memory (not on the stack) the next state is STORE1. If the instruction is the OUT-instruction the next state is OUTPUT. If the instruction is an HLT-instruction the next state is one of HALTI, HALTIE, IHHOLD, IEHHOLD or INTJAM1 (see table 4.1-4). For all other instructions the next state is one of IFCH1, INTJAM1, HALTE, HOLD or HHOLD (see table 4.1-5).

Table 4.1-4 Next state at end of execution for HLT-instructions

HALT	HOLD	INT. INTE	NEXT STATE
0	0	0	HALTI
0	0	1	INTJAM1
0	1	X	IHHOLD
1	0	X	HALTIE
1	1	X	IEHHOLD

Table 4.1-5 State transition at end of instruction execution (except HLT)

HALT	HOLD	INT. INTE	NEXT STATE
0	0	0	IFCH1
0	0	1	INTJAM1
0	1	X	HOLD
1	0	X	HALTE
1	1	X	HHOLD

- **The state STORE1**

In this state single-byte results and the lst byte of 2-byte results are stored in the memory. Depending on the opcode one of the following is placed on the addresss bus:

- the register pair BC
- the register pair DE
- bytes 2 and 3 of the instruction
- the register pair HL

They byte to be stored is placed on the data bus. MEMW is activated. For l-byte results the next state is one of IFCH1, INTJAM1, HALTE, HOLD, or HHOLD depending on the value of HALT, HOLD and of INT.INTE (see table 4.1-5). For 2-byte results the next state is STORE2.

- **The state STORE2**

In this state the 2nd byte of 2-byte results is stored in memory. The address used in STORE1 is incremented and placed on the address bus. The bute to be stored is placed on the data bus. MEMW is activated. The next state is one of IFCH1, INTJAM1, HALTE, HOLD, or HHOLD (see Table 4.1-5).

- **The state PUSH1**

In this state the 1st of 2 bytes (the MSB) is pushed onto the stack. The stack pointer (SP) is decremented first and the decremented value is placed on the address bus. The byte to be pushed is placed on the data bus. STACK and MEMW are activated. The next state is always PUSH2.

- **The state PUSH**

In this state the 2nd of 2 bytes (the LSB) is pushed onto the stack. The stack pointer (SP) is decremented first and the decremented value is placed on the address bus. The byte to be written into memory

is placed on the data bus. STACK and MEMW are activated. The next state is one of IFCH1, INTJAM1, HALTE, HOLD or HHOLD (see table 4.1-5).

- **The state OUTPUT**

In this state a byte is written to the port indicated by the 2nd byte of the instruction. The port number (2nd byte of the instruction) is placed on the address bus, padded with leading zeros. The byte to be written is placed on the data bus. IOW is activated. The next state is one of IFCH1, INTJAM1, HALTE, HOLD, or HHOLD (see table 4.1-5).

- **The state HALTE**

This state is entered at the end of the execution of any instruction other than HLT if HALT = 1 and HOLD = 0. WAIT is set to 1. The next state entered is IFCH1, INTJAM1, HALTE, HOLD or HHOLD (see Table 4.1-5).

- **The state HOLD**

This state is entered at the end of the execution of any instruction other than HLT if HOLD = 1 and HALT = 0. The address bus and the data bus are floated and HLDA is set to 1. The next state is one of IFCH1, INTJAM1, HALTE, HOLD or HHOLD (see table 4.1-5).

- **The state HHOLD**

This state is entered at the end of the execution of any instruction other than HLT if HALT = 1 and HOLD = 1. The address bus and the data bus are floated. HLDA and WAIT are set to 1. The next state is IFCH1, INTJAM1, HALTE, HOLD, or HHOLD (see table 4.1-5)

- **The state HALTI**

This state is entered at the end of an HLT-instruction if HALT = 0 and HOLD = 0. WAIT is set to 1. The next state is one of HALTI, HALTIE, IHHOLD, IEHHOLD or INTJAM1 (see table 4.1-5).

- **The state HALTIE**

This state is entered at the end of an HLT-instruction if HALT = 1 and HOLD = 0. WAIT is set to 1. The next state is one of HALTI, HALTIE, IHHOLD, IEHHOLD or INTJAM1 (see table 4.1-4).

- **The state IHHOLD**

This state is entered at the end of an HLT-instruction if HOLD = 1 and HALT = 0. HLDA and WAIT are set to 1 and both the address bus and the data bus are floated. The next state is one of HALTI, HALTIE, IHHOLD, IEHHOLD or INTJAM1 (see table 4.1-4).

- **The state IEHHOLD**

This state is entered at the end of an HLT-instruction if HALT = 1 and HOLD = 1. HLDA and WAIT are set to 1 and both the address and data buses are floated. The next state is one of HALTI, HALTE, IHHOLD, IEHHOLD or INTJAM1 (see table 4.1-4).

- **The state INTJAM1**

In this state the first of 3 bytes (opcode) is jammed into the Q-80 as a result of the recognition of an interrupt. Further interrupts are disabled by setting INTE to 0. The value of the program counter is placed on the address bus and INTA and IFCH are activated. After the first jammed byte is received, OFL is set to 0. The next state is IFCH2.

- **The state INTJAM2**

In this state the 2nd of 3 bytes is jammed into the Q-80 as a result of the recognition of an interrupt. The value of the program counter is placed on the address bus and INTA and IFCH are activated. The next state is INTJAM3.

- **The state INTJAM3**

In this state the 3rd of 3 bytes is jammed into the Q-80 as a result of the recognition of an interrupt. The value of the program counter is placed on the address bus and INTA and IFCH are activated. The next state is one of EXEC, OFCH1, POP1 or INPUT depending on the opcode (first byte jammed).

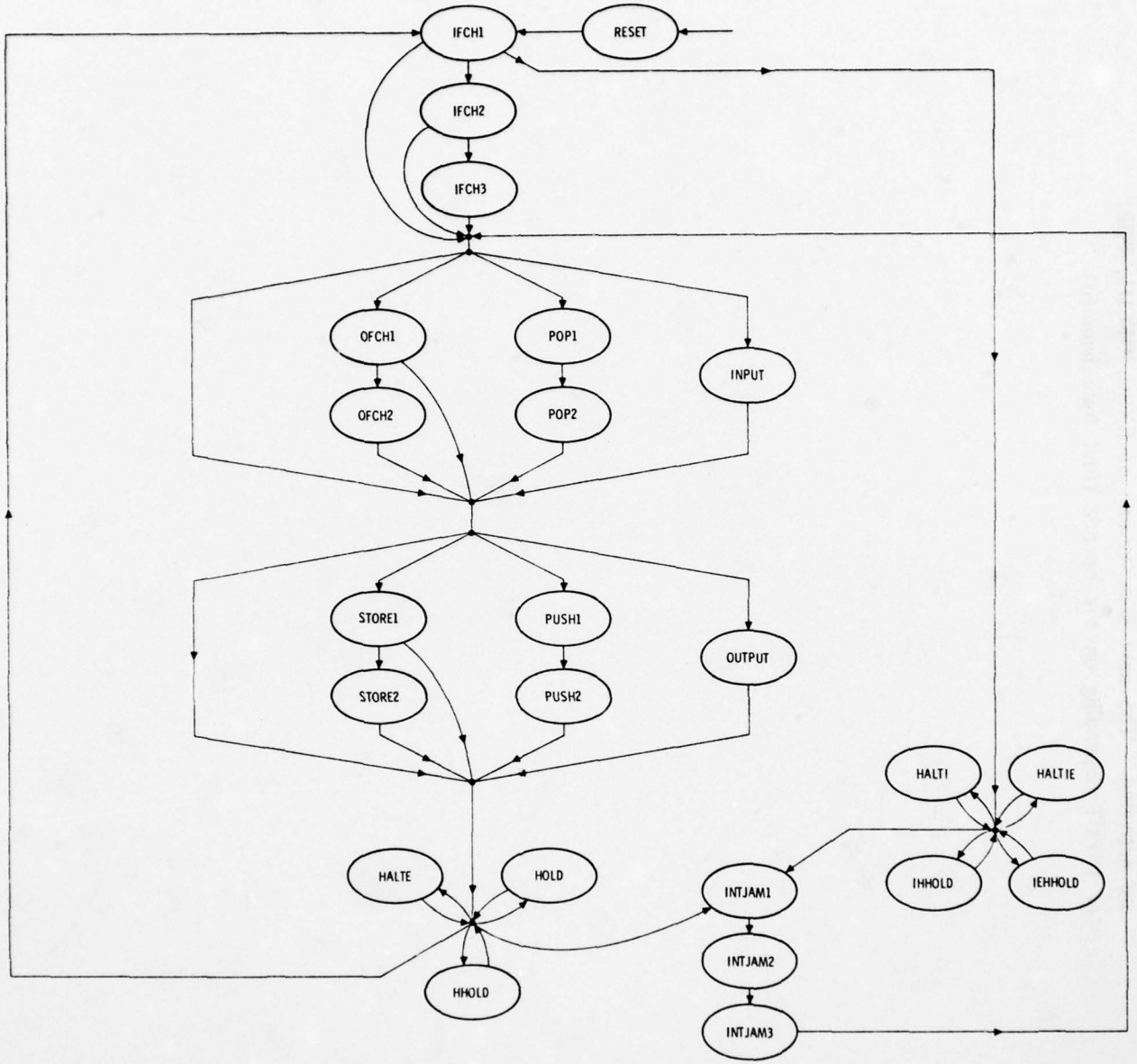


Figure 4.1-18 Q-80 State Design

4.2 Q80 Breadboard Logic Design Description

The discussion in the previous section was limited to the functional characteristics of the Q80. This section presents the detailed design of the Q80 breadboard. Even though functionally identical, in many ways, the breadboard is different than the final Q80 LSI brassboard will be. For example, the breadboard has a RAM microprogram memory which is larger than ROM microprogram memory that the Q80 LSI brassboard will contain. This RAM microprogram memory will be loaded via the MDS–800 system and microprograms generated and run under direct MDS–800 control. Thus the breadboard contains a significant amount of additional logic, over the final LSI brassboard, required to facilitate microprogram debugging and hardware checkout.

4.2.1 Architecture

Figure 4.2-1 illustrates the top level architecture of the Q80 breadboard and its interface to the MDS–800 system. The Q80 and MDS–800 communicate over two separate interfaces; the Memory Simulator Interface and the Q80 Control Interface. During the checkout phase of the Q80 breadboard development the MDS–800 will simulate the main memory for the Q80 through an I/O port. That is, under software control, the MDS–800 will respond to Q80 memory requrests by fetching data from memory and outputing to the I/O port and vice versa for Q80 writes. To handle this operation, a block of logic called the Memory Simulator Interface is required. Taking this approach greatly facilitates firmware and hardware checkout since all the debugging tools resident the MDS–800 can be utilized for the Q80. In addition the MDS–800 can directly monitor all activities of the Q80 interface and compare them with expected results generated by the 8080 Simulator program (refer to Section 5.1.2). Another I/O port is required for loading the Q80 microprogram memory and controlling the operation of the Q80. This is done through the Q80 Control Interface. This interface contains the logic required to load microprograms and control their execution. Control features such as single clock, breakpoint, run, reset and halt are provided. The control functions will be input to the Q80 Control Interface via the CRT keyboard interface to the MDS–800 system. Q80 response to control functions will be displayed on the CRT, thus facilitating on–line, interactive debugging of Q80 firmware and hardware.

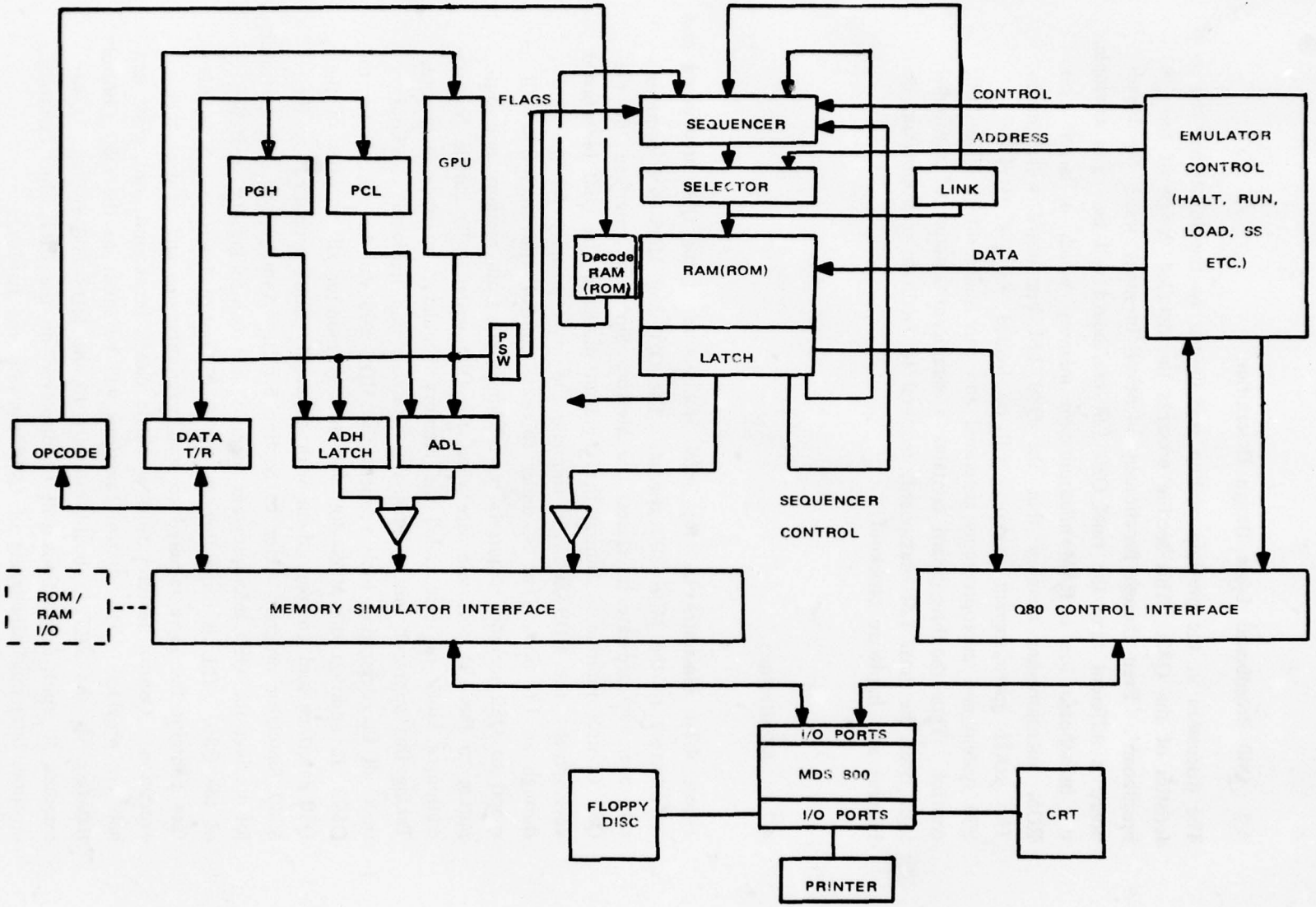


Figure 4.2-1 Q80 Breadboard/Firmware Development Station

The Q80 breadboard system as described in the previous paragraphs is not standalone – an MDS–800 system with adequate support hardware and software is required to operate the Q80. Therefore, when Q80 firmware and hardware are fully verified the breadboard will be upgraded to be made standalone. The initial Q80 breadboard is designed such that upgrading is a simple matter of adding memory (RAM and/or ROM), I/O ports, and ROM microprogram control – no charges will be required in the basic Q80 breadboard hardware. Details on how this is acomplished are presented in Sections 4.2.4 and 4.2.5.

Q80 breadboard data paths (as illustrated in Figure 4.2-1) are identical to the data paths for the final Q80 LSI brassboard, except in the area of the microprogram control, of the microprogram RAM. Microinstruction routines will be transferred to the Q80 through the Q80 control interface and loaded into specified locations in the RAM. To do so requires an address selector and additional logic in the sequencer. Details are presented in the following sections.

4.2.2 LSI Partitioning

The GPU microprocessor chip comprises the core of Q80 Emulator hardware. However, five additional custom part types implemented with universal gate arrays are required to complete the hardware design. Figure 4.2-2 contains a block diagram illustrating the Q80 hardware organization. The upper right-hand corner shows the microprogram control unit. Two GUA types are required to implement this function. The lower left portion of Figure 4.2-2 shows the Q80 Address and Data Bus interface. An Interface GUA type is utilized to connect the GPU to this interface. As indicated by the broken boxes, four copies of the Interface GUA are required. The two remaining GUA types contain various decoding functions, status indicators, and shift control logic. Functional descriptions of the five GUA types are contained in the following sections.

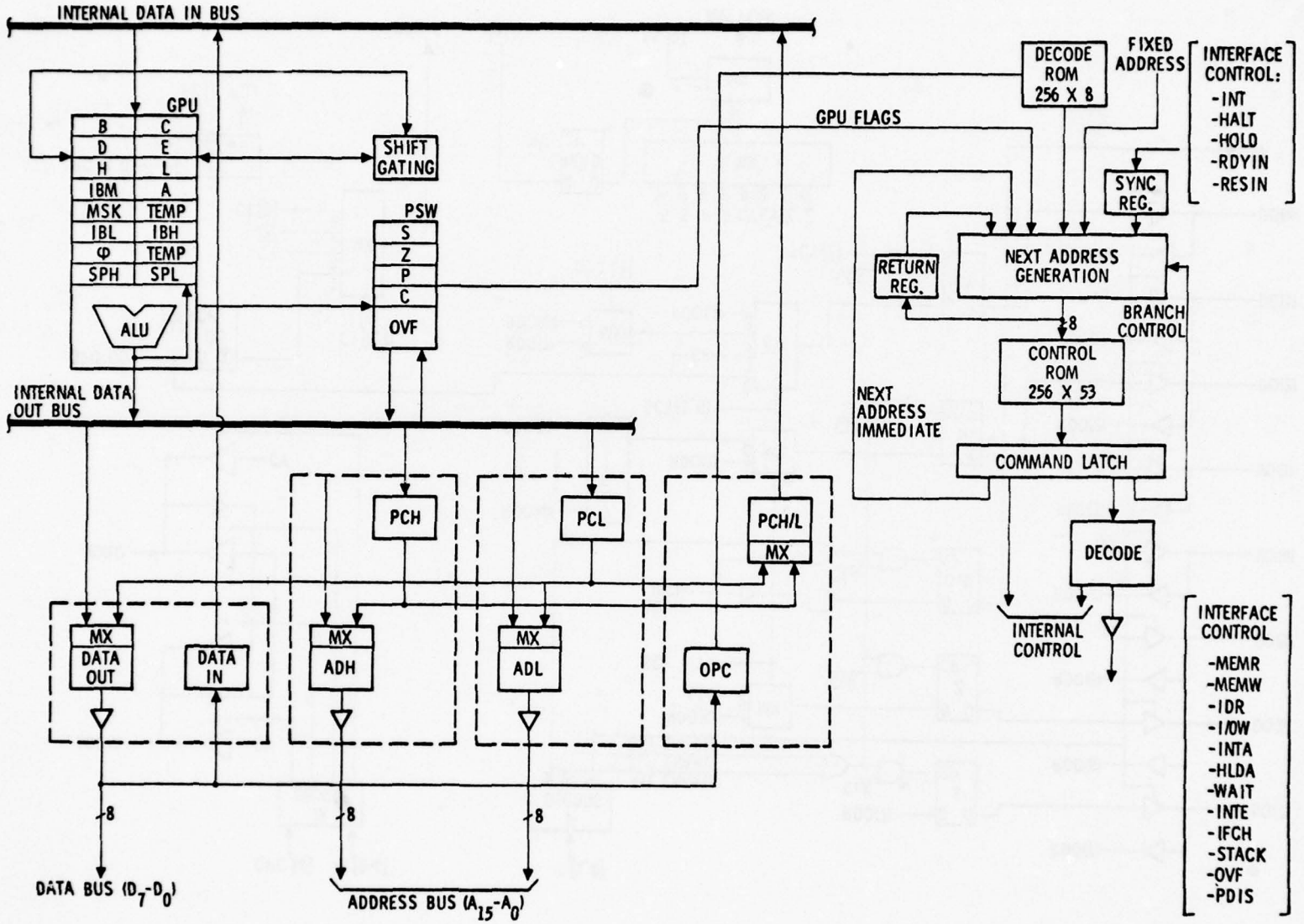


FIGURE 4.2-2 Q80 EMULATOR BLOCK DIAGRAM

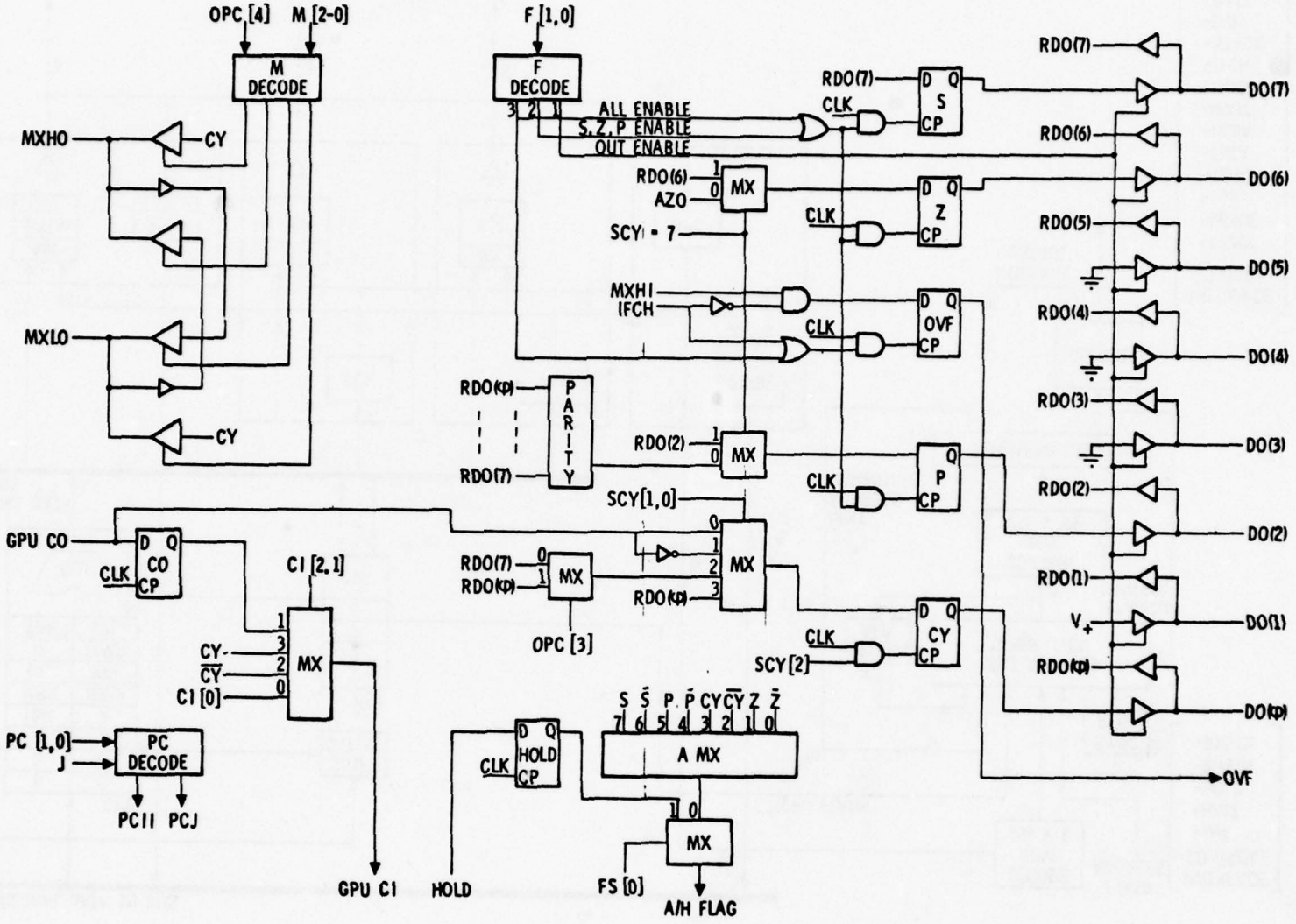


FIGURE 4.2-3 STATUS FLAG SHIFT CONTROL GUA

4.2.3 Condition Flag and Shift Control GUA

Figure 4.2-3 illustrates the functional organization of the Condition Flag and Shift Control GUA (CFS). CFS contains the Q80 condition flags and provides the source selection for the carry input to the GPU. In addition, it contains the load control for the external program counter registers and provides the end condition generation required by the GPU during shift operations. Each of these functions is described in the following paragraphs.

Condition Flags

There are five condition flags associated with the execution of Q80 instructions, including sign, zero detect, parity, carry and overflow. They are designated S, Z, P, CY and OVF, respectively. Each flag is implemented as a D flip/flop with a gated clock input. When enabled, a flag flip/flop will retain the input value present on the falling edge of the system clock (ν).

Microinstruction field F[1,0] controls the loading of condition flags. When F[1,0] = 3, OVF, S, Z and P are enabled. When F[1,0] = 2, only S, Z and P are enabled.

The S flag receives its input from bit 7 of the internal Data Out bus.

The Z flag receives its input from bit 6 of the internal Data Out bus when microinstruction field SCY [2–0] = 7, otherwise its input is the all zeroes (AZO) signal from the GPU.

The P flag receives its input from bit 2 of the internal Data Out bus when SCY [2–0] = 7, otherwise its input is from a parity network. The parity network outputs a one if there are an even number of ones on the internal Data Out bus.

Input to the CY flag is a function of microinstruction field SCY [1,0] and OPC [3], as shown in table 4.2-1.

Table 4.2-1 Control of the CY input

SCY [1]	SCY [0]	OPC [3]	CY INPUT
0	0	X	GPU CO
0	1	X	GPU CO
1	0	0	DO [7]
1	0	1	DO [0]
1	1	X	DO [0]

The OVF flag receives its input from the GPU MXH1 signal. OVF is unconditionally reset at the beginning of each instruction fetch cycle.

CFS provides the capability of storing the condition flags in memory in the form of a processor status word. This occurs during the PUSH PSW instruction. Flags are enabled onto the internal Data Out bus when F [1,0] = 1. The format of the PSW is as shown in Figure 4.2-4.

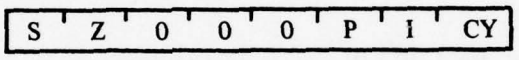


FIGURE 4.2-4 PSW FORMAT IN THE Q-80

During a POP PSW the flags are restored by first transferring the status word to the GPU and then to the flag flip/flops via the internal Data Out bus.

A-Multiplexer and HOLD Flip/Flop

The A-multiplexer provides the microprogram with a means of selecting one of the condition flags for use in a conditional branch. This 8 to 1 mulplexer is controlled by opcode register field OPC [5-3] with channel assignments as shown in the Table 4.2-2.

OPC [5-3]	A-mx Output
000	Z
001	Z
010	CY
011	CY
100	P
101	P
110	S
111	S

Table 4.2-2 Channel Assignments for the A-multiplexer.

The CFS GUA also contains a synchronization flip/flop for the external HOLD signal. The output of the flip/flop and the A-multiplexer are multiplexed onto a single signal called A/HFLAG for transmission to the microprogram control unit. When microinstruction field FS [0] = 1, the value of the HOLD flip/flop is sent. When FS [0] = 0, the A multiplexer output is sent.

GPU Carry Input Selection

The CFS GUA provides the source of the carry input (CI) to the GPU. CI can receive one of four possible values, depending on microinstruction field CI [2,1], as shown in Table 4.2-3.

Table 4.2-3 Control of the CI input to the GPU

CI [2,1]	CI (to GPU)
0 0	CI [0]
0 1	CO f/f
1 0	CY
1 1	$\overline{CY}$

CI [0] comes directly from the microinstruction; the CO f/f contains the value of the GPU carry out signal during the previous processor cycle.

Program Counter Load Control

The CFS GUA also contains a decoding network which generates the register load enables for the program counter (PCH and PCL). These load enables are a function of the J flag (discussed in section 4.2.4) and the microinstruction field PC [1,0] as described in Table 4.2-4.

Table 4.2-4 Program Counter load control.

PC [1,0]	J	PCH load enable	PCL load enable
0 0	X	0	0
0 1	1	0	1
1 0	X	0	1
1 1	X	1	0

GPU End Condition Generation

This function provides the shift gating for the left and right end bits to or from the GPU. It consists of four tri-state drivers and the M decoding network. Two of the drivers are connected to the GPU MXHO signal. One receives its input from the CY condition flag and the other from MXLO. The other two drivers are connected to the GPU MXLO signal. One receives its input from the CY condition flag and the other from MXLO. The enabling of these tri-state drivers is controlled by microinstruction field M [2-0] and OPC [4], as shown in Table 4.2-5.

Table 4.2-5 End Condition Generation Control

OPC [4]	M [2-0]	OPERATION
0	0 0 1	MXLO ← MXHO
0	0 1 0	MXHO ← MXLO
1	0 1 0	MXHO ← CY
1	0 1 1	MXLO ← CY

4.2.4 Decode GUA

The Decode GUA contains four separate decoding functions necessary to control the operation of the Q80. These decode functions are performed on certain fields of the microinstruction word and on the contents of the opcode register. A block diagram of the Decode GUA is shown in Figure 4.2-5.

The I/O decoding function generates the 12 control signals shown in Figure 4.2-5. The functional definitions of these signals are contained in Section 4.1.2. The I/O decode is controlled by microinstruction field I/O [3-0]. More than one signal may be activated for a given combination of I/O [3-0] as shown in Table 4.2-6. Except for INTE and WRITE all I/O signals are driven by edge-triggered D flip/flops. Thus, to continuously activate these signals the microprogram must continuously output the appropriate I/O [3-0] combination.

The output signal INTE is driven by a J/K flip/flop. It is set when I/O [3-0] = 1101 and remains set until I/O [3-0] = 1100.

The WRITE signal is a direct decode of I/O [3-0]. It is used to enable Q80 data onto the system data bus during write operations.

The Opcode decoding function is used to determine whether the current instruction format is either one, two, or three bytes. It is a combinational function of the contents of the opcode register, OPC [7-0]. The outputs, KJ are equal to 00 for one byte instructions, 01 for two byte instructions, and 11 for three byte instructions.

The MIC decoding function generates the load control signals for the address register (ADH and ADL), the DATAIN register, DATAOUT register, and the opcode register (OPC). This function is controlled by microinstruction field MIC [2-0].

The T MUX function generates the four register select T inputs to the GPU. These signal are a function of microinstruction fields T [3, 2] and TS [2-0] and opcode bits OPC [5, 4, 2, 1, 0].

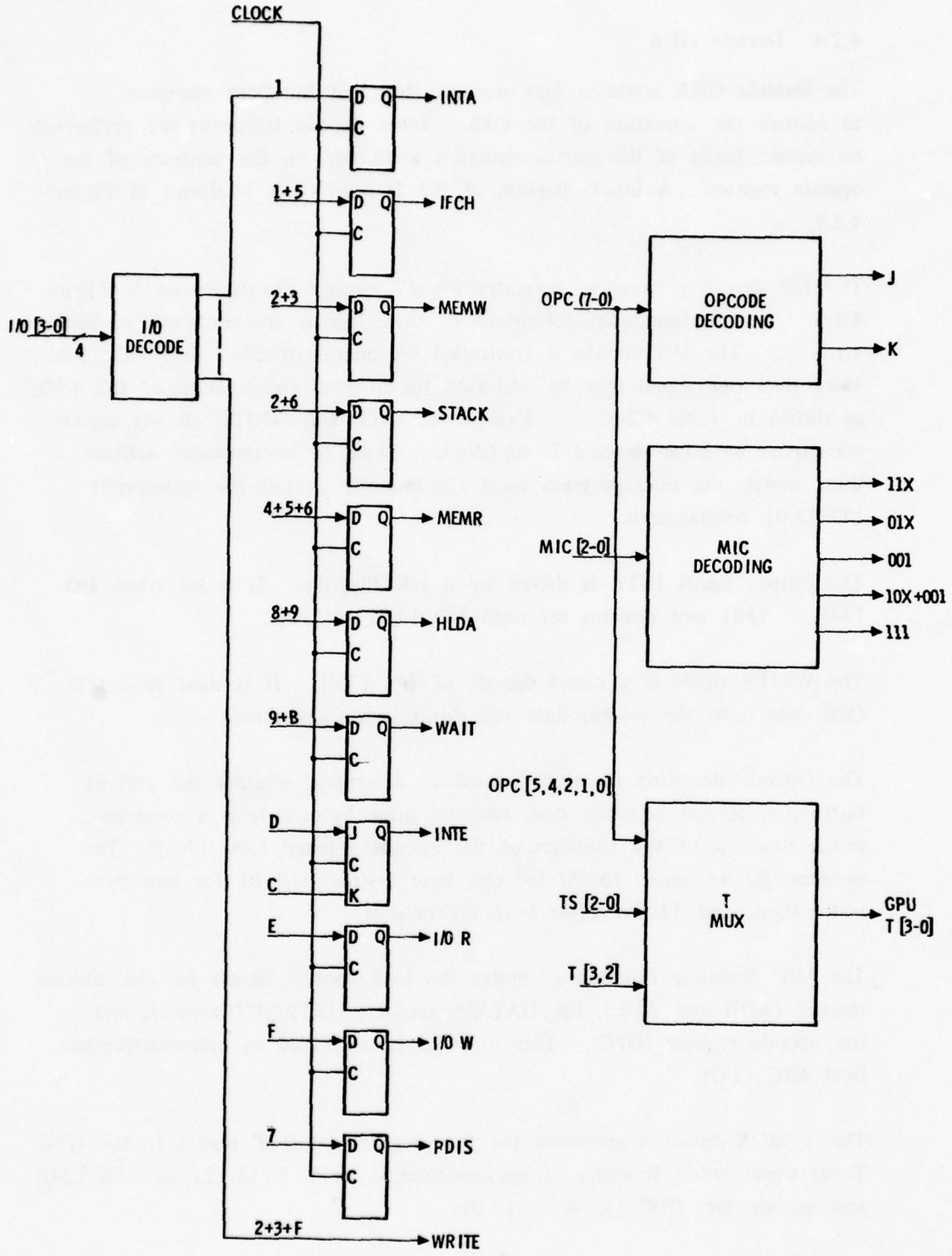


FIGURE 4.2-5 DECODE GUA

Table 4.2-6 I/O Decode Operations

I/O[3-0]	Control Signals Activated
0000	NULL
0001	INTA, IFCH
0010	MEMW, STACK, WRITE
0011	MEMW, WRITE
0100	MEMR
0101	MEMR, IFCH
0110	MEMR, STACK
0111	PDIS
1000	HLDA
1001	HLDA, WAIT
1010	UNUSED
1011	WAIT
1100	INTE ← 0
1101	INTE ← 1
1110	IOR
1111	IOW, WRITE

4.2.5 Interface GUA

This section contains a functional description of the Interface GUA. It is a general-purpose part which can be used to interface the GPU microprocessor to a wide variety of system bus structures. For the Q80 application four Interface GUA's are required, each serving a different function.

Refer to the block diagram of Figure 4.2-6. The Interface GUA contains two 8-bit registers, designated X and Y. Each register has an independant load control. There are three 8-bit wide input busses designated A, B, and C. Register X can be loaded from either bus A or bus B depending on the state of the SEL input. If SEL is zero bus A is selected, if SEL is one bus B is selected. Bus C is the input to register Y.

Separate output busses are provided for each register, designated XOUT and YOUT. When the X output enable (OEX) is LOW the eight X outputs are enabled. When the OEX input is high the X outputs are in the high-impe dance state. When the Y output enable (OEY) is HIGH the eight Y outputs are enabled. When OEY is LOW the Y outputs are in the high-impedance state.

The X outputs are driven with low-impedance tri-state drivers and are thus capable of driving TTL loads directly. The Y output tri-states are intended for driving a standard CMOS bus.

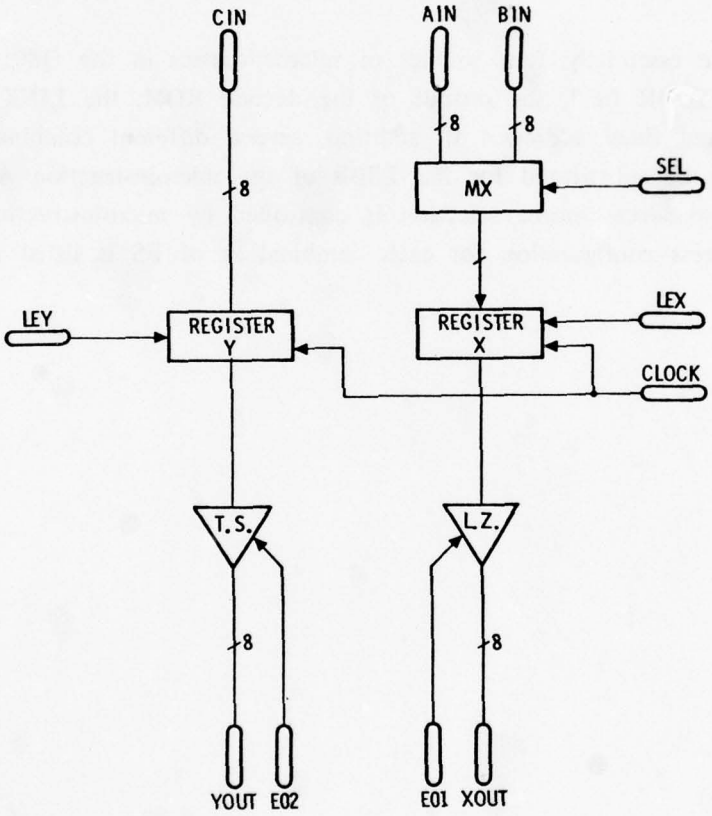


Figure 4.2-6 Interface GUA Block Diagram

4.2.6 Q80 Microsequencer

The Q80 Microsequencer controls the sequence of execution of microinstructions stored in microprogram memory. A block diagram of the Microsequencer is shown in Figure 4.2-7. Two GUA part types are required to implement this function. They are referred to as μC-A and μC-B.

Q80 firmware is contained in a 256 word address space. Thus, the Microsequencer generates 8 bits of microaddress. It allows unconditional branching to any microinstruction within its 256-word range. It also provides for conditional branching on several of the Q80 condition flags and status indicators. In addition, it provides the linkage capability required for microsubroutines.

There are essentially four sources of microaddresses in the Q80: the command register ADDR field, the output of the decode ROM, the LINK counter, and a pre-wired fixed address. In addition, several different combinations of condition flags can be substituted for the LSB's of the microinstruction ADDR field. The microaddress source selection is controlled by microinstruction field FS [2-0]. The address configuration for each combination of FS is listed in Table 4.2-7.

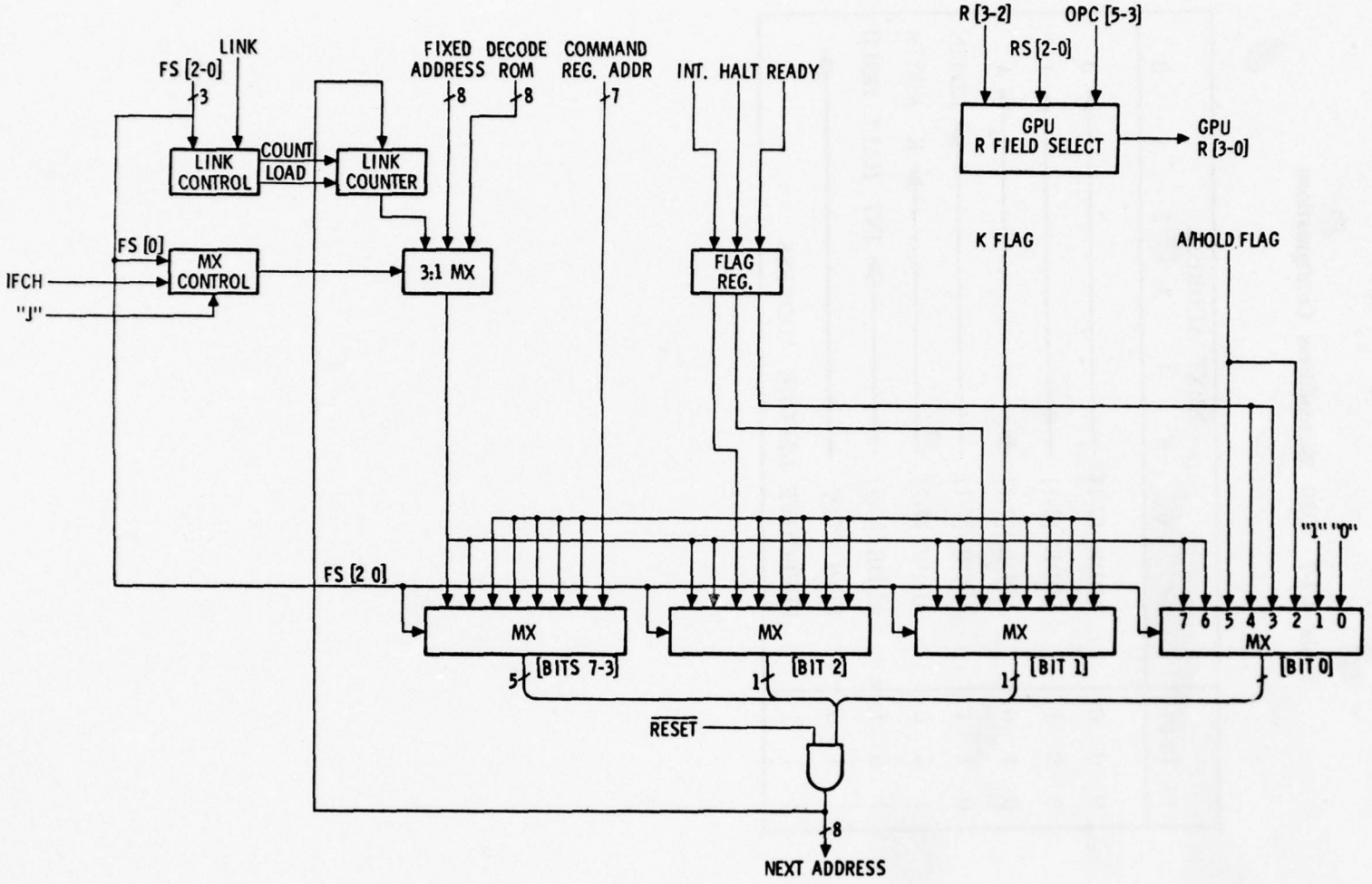


Figure 4.2-7 Q-80 Microsequencer

Table 4.2-7 Q80 Microaddress Configurations

FS [2-0]	NEXT ADDRESS 7 6 5 4 3 2 1 0
0 0 0	ADDR [7-1] → 0
0 0 1	ADDR [7-1] → 1
0 1 0	ADDR [7-1] → A
0 1 1	ADDR [7-1] → RDYIN
1 0 0	ADDR [7-2] → K RDYIN
1 0 1	ADDR [7-3] → INT HALT HOLD
1 1 0	LINK [7-0] →
1 1 1	EFFECTIVE DECODE ADDRESS

4.3 Firmware

4.3.1 Overview

The Q-80 Firmware controls the internal and outward behavior of the Q-80. Thus the firmware defines the characteristics of the Q-80: the instruction set, interface protocol, etc.

The Q-80 is a horizontally microprogrammed computer, meaning that vectors of control signals are stored in each microword. The microprogram control memory consists of 256 words of 56 bits each. Of these, 46 words are currently unused and within each microword only 53 bits are used, leaving 3 spare bits.

4.3.2 The Q-80 Instruction Cycle

The Q-80 Firmware consists of the following parts:

- The instruction fetch routine.
- 60 instruction execution routines.
- 4 subroutines, which fetch operands, store results and push and pop the stack.
- Two sets of 8 words which implement the external HALT, HOLD and INT features.
- The interrupt jam routine.
- 4 subroutines, which fetch operands, store results and push and pop the stack.

The most common instruction cycle begins in the instruction fetch routine where 1, 2 or 3 bytes are fetched and the program counter is accordingly incremented. In all cases the instruction fetch phase ends with a branch to one of the 60 instruction execution routines, based on the opcode fetched.

Depending on what instruction is being executed, the instruction execution routine may call some of the subroutines to fetch an operand, store a result, push or pop the stack. At the end of the instruction execution routine all register transfers associated with the respective instruction have been accomplished. In the last clock cycle of every instruction execution routine, the program counter's value is placed on the address bus and an 8-way branch is taken based on the Q-80 inputs HALT, HOLD and INT (ANDed with INTE). Commonly, all three of these inputs are zero, in which case the next instruction fetch phase commences immediately following the 8-way branch. Figure 4.3-1 shows a flow diagram of the Q-80 instruction cycle.

The presence of a HALT and/or a HOLD delays entering of the next instruction fetch. After both HALT and HOLD become inactive or if HALT or HOLD are both inactive at the time when the instruction execution routine is done, the INT signal is considered. If INT is active and INTE is set, the Q-80 microprogram enters the interrupt triggered instruction jam routine and accepts 3 jammed bytes. Depending on the first byte jammed (opcode), 1, 2, or all 3 bytes are considered to constitute the next instruction to be executed. The instruction jam routine ends by branching to one of the 60 instruction execution routines.

4.3.3 Partitioning of the microaddress space.

The 256 addresses available in the microprogram control memory are assigned as follows:

- address 0: start of reset routine
- address 1-9F: 60 instruction execution routines and remainder of reset routine
- addresses A0-A7: operand fetch subroutine (OFCH)
- addresses A8-AF: store subroutine
- addresses B0-B7: pop subroutine
- addresses B8-BF: push subroutine
- addresses C0-CF: interrupt jam routine

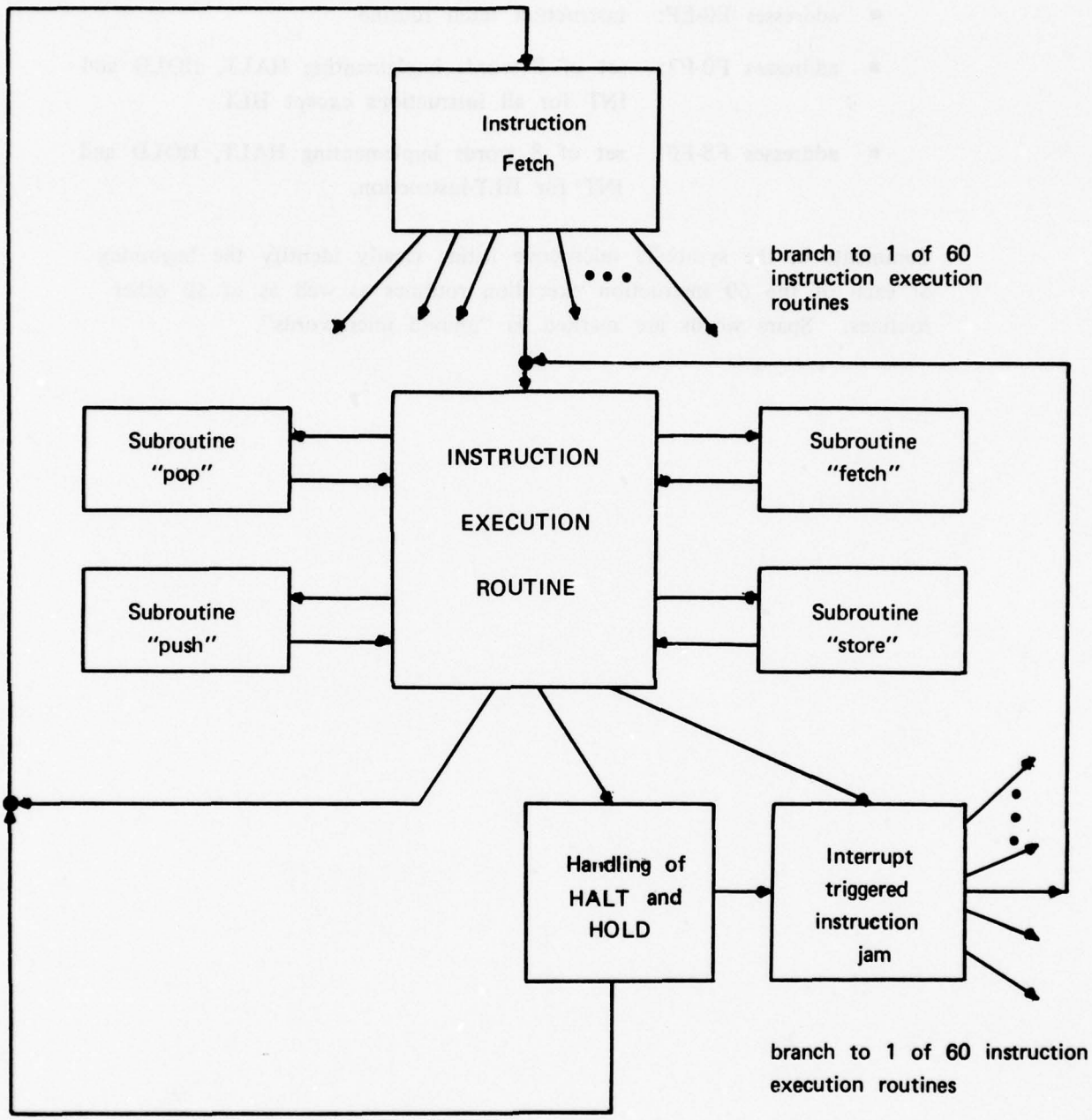


Figure 4.3-1 Q-80 Instruction Cycle

- addresses D0-DF: block of spare words
- addresses E0-EF: instruction fetch routine
- addresses F0-F7: set of 8 words implementing HALT, HOLD and INT for all instructions except HLT
- addresses F8-FF: set of 8 words implementing HALT, HOLD and INT for HLT-instruction.

Comments in the symbolic microcode listing clearly identify the beginning of each of the 60 instruction execution routines as well as of all other routines. Spare words are marked as "unused microwords".

:F1:Q80.MIC 6 JAN 1978

```
                ****************************
                *  REGISTER UTILIZATION    *
                ****************************

*               REG(0): B-REGISTER
*               REG(1): C-REGISTER
*               REG(2): D-REGISTER
*               REG(3): E-REGISTER
*               REG(4): H-REGISTER
*               REG(5): L-REGISTER
*               REG(6): BYTE-2 OF INSTRUCTION
*               REG(7): A-REGISTER
*               REG(8): MASK USED IN RST-INSTRUCTION (00111000)
*               REG(9): NO SPECIFIC ASSIGNMENT
*               REG(A): OPCODE USED IN RST INSTRUCTION
*               REG(B): BYTE-3 OF INSTRUCTION
*               REG(C): =0, CLEARED BY RESET, READ ONLY
*               REG(D): NO SPECIFIC ASSIGNMENT
*               REG(E): SP (MSB)
*               REG(F): SP (LSB)
```

```
00000000:       *** RESET ***

        PCH = DO, DO = ALC,
              REG(C) = ALC, ALC = P1B AND 0,
        GOTO 01010110;              * WHERE RESET CONTINUES

-------------                                                  ------------------
! ADDR = 00 !                                                 ! C0D8E000030056 !
-------------------------------------------------------------------------------------
! R = C ! A = 2 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 56 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 3 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ------------------------                    -------------
```

```
00000001:       *** MVI R,DATA8 ***
                *** MOV R1,R2   ***

        REG(DDD) = ALC, ALC = 0 + P2B, P2B = REG(SSS),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ------------------
! ADDR = 01 !                                                 ! 005080501005F0 !
-------------------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 1 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------------
! T = X ! D = 4 ! DOE = 0 !! TS  = 1 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
00000010:       *** MOV R,M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                          * CALL OFCH

-------------                                   ------------------
! ADDR = 02 !                                  ! 05E0E000C000A0 !
-------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------
! S = 0 ! C = 0 !         ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------         -----------------------                 -------------

00000011:

        REG(DDD) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                   ------------------
! ADDR = 03 !                                  ! 0000E8401005F0 !
-------------------------------------------------------------------
! R = X ! A = 0 ! M   = 0 !! RS  = 1 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------
! S = 1 ! C = 0 !         ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------         -----------------------                 -------------
```

```
00000100:       *** MVI M,DATA8 ***
                *** MOV M,R     ***

        DATAOUT = DO, DO = P2B, P2B = REG(SSS),
        GOTO 10101000;                          * MERGE INTO STORE

-------------                                                 -----------------
! ADDR = 04 !                                                ! 00E0E0102000A8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = A8 !
-------------------------------------------------------------------------------
! T = X ! D = 7 ! DOE = 1 !! TS  = 1 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------
```

```
00000101:       *** LXI RP,DATA16 ***

        REG(RP0) = ALC, ALC = 0 + P2B, P2B = REG(B),
        GOTO 00000110;

-------------                                                  ------------------
! ADDR = 05 !                                                 ! 0B508080000006 !
-------------------------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 2 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 06 !
-------------------------------------------------------------------------------------------------
! T = B ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

00000110:

        REG(RP1) = ALC, ALC = 0 + P2B, P2B = REG(6),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ------------------
! ADDR = 06 !                                                 ! 065080C01005F0 !
-------------------------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 3 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------------------
! T = 6 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
00000111:       *** LDA ADDR ***

        ADL = DO, DO = P2B, P2B = REG(6),
        GOTO 00001000;

------------                                                    ----------------
! ADDR = 07 !                                                  ! 06E0E000400008 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = 08 !
-------------------------------------------------------------------------------------
! T = 6 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

00001000:

        DO = P2B, P2B = REG(B),
        CALL 10100001;                          * CALL OFCH

------------                                                    ----------------
! ADDR = 08 !                                                  ! 0BE0E0008001A0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------------
! T = B ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  ------------

00001001:

        REG(7) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                    ----------------
! ADDR = 09 !                                                  ! 7000E8001005F0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------
```

```
00001010:       *** STA ADDR ***

        DATAOUT = DO, DO = P2B, P2B = REG(7),
        GOTO 00001011;

------------                                                  ----------------
! ADDR = 0A !                                                ! 07E0E00020010A !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = 0A !
-------------------------------------------------------------------------------
! T = 7 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ---------------------                   ------------

00001011:

        ADL = DO, DO = P2B, P2B = REG(6),
        GOTO 00001100;

------------                                                  ----------------
! ADDR = 0B !                                                ! 06E0E00040000C !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = 0C !
-------------------------------------------------------------------------------
! T = 6 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ---------------------                   ------------

00001100:

        DO = P2B, P2B = REG(B),
        GOTO 10101010;          * MERGE INTO STORE AT SECONDARY ENTRY POINT

------------                                                  ----------------
! ADDR = 0C !                                                ! 0BE0E0000000AA !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = AA !
-------------------------------------------------------------------------------
! T = B ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ---------------------                   ------------
```

```
00001101:       *** LHLD ADDR ***

        ADL = DO, DO = P2B, P2B = REG(6),
        GOTO 00001110;

------------                                                  ------------------
! ADDR = 0D !                                                ! 06E0E00040000E !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = 0E !
-------------------------------------------------------------------------------
! T = 6 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ---------------------                    ------------

00001110:

        DO = P2B, P2B = REG(B),
        CALL 10100001;                          * CALL OFCH

------------                                                  ------------------
! ADDR = 0E !                                                ! 0BE0E0008001A0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------
! T = B ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ---------------------                    ------------

00001111:

        REG(5) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        GOTO 00010000;

------------                                                  ------------------
! ADDR = 0F !                                                ! 5000E800000010 !
-------------------------------------------------------------------------------
! R = 5 ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 10 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ---------------------                    ------------

00010000:

        REG(6) = ALC, ALC = P1B + 0 + 1, P1B = REG(6),
        GOTO 00010001;
```

```
------------                                                    ----------------
! ADDR = 10 !                                                  ! 6050E000002110 !
-------------------------------------------------------------------------------
! R = 6 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 1 ! ADDR = 10 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          --------------------                    ------------
```

00010001:

```
        REG(B) = ALC, ALC = P1B + 0 + CO, P1B = REG(B),
        ADL = DO, DO = P2B, P2B = REG(6),
        GOTO 00010010;
```

```
------------                                                    ----------------
! ADDR = 11 !                                                  ! B6E0E000404012 !
-------------------------------------------------------------------------------
! R = B ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 2 ! ADDR = 12 !
-------------------------------------------------------------------------------
! T = 6 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          --------------------                    ------------
```

00010010:

```
        DO = P2B, P2B = REG(B),
        CALL 10100001;                          * CALL OFCH
```

```
------------                                                    ----------------
! ADDR = 12 !                                                  ! 0BE0E0008001A0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------
! T = B ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          --------------------                    ------------
```

00010011:

```
        REG(4) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

```
-------------                                                    ------------------
! ADDR = 13 !                                                  ! 4000E8001005F0 !
-------------------------------------------------------------------------------------
! R = 4 ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ---------------------                     -------------
```

```
00010100:       *** SHLD ADDR ***

        DATAOUT = DO, DO = P2B, P2B = REG(5),
        GOTO 00010101;

-------------                                                  ------------------
! ADDR = 14 !                                                 ! 05E0E000200114 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = 14 !
-------------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00010101:

        ADL = DO, DO = P2B, P2B = REG(B),
        GOTO 00010110;

-------------                                                  ------------------
! ADDR = 15 !                                                 ! 0BE0E000400016 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = 16 !
-------------------------------------------------------------------------------------
! T = B ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00010110:

        DO = P2B, P2B = REG(B),
        GOTO 00010111;

-------------                                                  ------------------
! ADDR = 16 !                                                 ! 0BE0E000000116 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 16 !
-------------------------------------------------------------------------------------
! T = B ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00010111:

        DO = P2B, P2B = P2B,
        I/O = MEMW,
        GOTO 00011000;
```

```
-------------                                                    ------------------
! ADDR = 17 !                                                   ! 00E4E003000018 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 3 ! CI = 0 ! ADDR = 18 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00011000:

        DO = P2B, P2B = P2B,
        I/O = MEMW,
        GOTO 0001100R;

-------------                                                    ------------------
! ADDR = 18 !                                                   ! 00E4E003000318 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 3 ! CI = 0 ! ADDR = 18 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00011001:

        DATAOUT = DO, DO = P2B, P2B = REG(4),
        REG(6) = ALC, ALC = P1B + 0 + 1, P1B = REG(6),
        GOTO 00011010;

-------------                                                    ------------------
! ADDR = 19 !                                                   ! 64E0E00020201A !
-------------------------------------------------------------------------------------
! R = 6 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 1 ! ADDR = 1A !
-------------------------------------------------------------------------------------
! T = 4 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00011010:

        REG(B) = ALC, ALC = P1B + 0 + CO, P1B = REG(B),
        ADL = DO, DO = P2B, P2B = REG(6),
        GOTO 00011011;
```

```
------------                                                    ------------------
! ADDR = 1A !                                                  ! B6E0E00040411A !
-------------------------------------------------------------------------------------
! R = B ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 2 ! ADDR = 1A !
-------------------------------------------------------------------------------------
! T = 6 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
----------------           ----------------------                  ------------
```

00011011:

```
        DO = P2B, P2B = REG(B),
        GOTO 10101010;          * MERGE INTO STORE AT SECONDARY ENTRY POINT
```

```
------------                                                    ------------------
! ADDR = 1B !                                                  ! 0BE0E0000000AA !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = AA !
-------------------------------------------------------------------------------------
! T = B ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
----------------           ----------------------                  ------------
```

```
00011100:       *** LDAX RP ***

        ADL = DO, DO = P2B, P2B = REG(RP1),
        GOTO 00011101;

-------------                                                  -----------------
! ADDR = 1C !                                                 ! 00E0E03040011C !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = 1C !
-------------------------------------------------------------------------------
! T = X ! D = 7 ! DOE = 1 !! TS  = 3 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00011101:

        DO = P2B, P2B = REG(RP0),
        CALL 10100001;                          * CALL OFCH

-------------                                                  -----------------
! ADDR = 1D !                                                 ! 00E0E0208001A0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------
! T = X ! D = 7 ! DOE = 1 !! TS  = 2 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  -------------

00011110:

        REG(7) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  -----------------
! ADDR = 1E !                                                 ! 7000E8001005F0 !
-------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
00011111:       *** STAX RP ***

        DATAOUT = DO, DO = P2B, P2B = REG(7),
        GOTO 00100000;

-------------                                                   ------------------
! ADDR = 1F !                                                  ! 07E0E000200020 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = 20 !
-------------------------------------------------------------------------------------
! T = 7 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00100000:

        ADL = DO, DO = P2B, P2B = REG(RP1),
        GOTO 00100001;

-------------                                                   ------------------
! ADDR = 20 !                                                  ! 00E0E030400120 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = 20 !
-------------------------------------------------------------------------------------
! T = X ! D = 7 ! DOE = 1 !! TS  = 3 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00100001:

        DO = P2B, P2B = REG(RP1),
        GOTO 10101010;          * MERGE INTO STORE AT SECONDARY ENTRY POINT

-------------                                                   ------------------
! ADDR = 21 !                                                  ! 00E0E0300000AA !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = AA !
-------------------------------------------------------------------------------------
! T = X ! D = 7 ! DOE = 1 !! TS  = 3 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
00100010:       *** XCHG ***

        REG(D) = ALC, ALC = 0 + P2B, P2B = REG(2),
        GOTO 00100011;

-------------                                                  ------------------
! ADDR = 22 !                                                 ! D2508000000122 !
-------------------------------------------------------------------------------------------
! R = D ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 22 !
-------------------------------------------------------------------------------------------
! T = 2 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

00100011:

        REG(2) = ALC, ALC = 0 + P2B, P2B = REG(4),
        GOTO 00100100;

-------------                                                  ------------------
! ADDR = 23 !                                                 ! 24508000000024 !
-------------------------------------------------------------------------------------------
! R = 2 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 24 !
-------------------------------------------------------------------------------------------
! T = 4 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

00100100:

        REG(4) = ALC, ALC = 0 + P2B, P2B = REG(D),
        GOTO 00100101;

-------------                                                  ------------------
! ADDR = 24 !                                                 ! 4D508000000124 !
-------------------------------------------------------------------------------------------
! R = 4 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 24 !
-------------------------------------------------------------------------------------------
! T = D ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

00100101:

        REG(D) = ALC, ALC = 0 + P2B, P2B = REG(3),
        GOTO 00100110;
```

```
-------------------------------------------------------------------------------

-------------                                                 ----------------
! ADDR = 25 !                                                ! D3508000000026 !
-------------------------------------------------------------------------------
! R = D ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 26 !
-------------------------------------------------------------------------------
! T = 3 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------

00100110:

        REG(3) = ALC, ALC = 0 + P2B, P2B = REG(5),
        GOTO 00100111;

-------------                                                 ----------------
! ADDR = 26 !                                                ! 35508000000126 !
-------------------------------------------------------------------------------
! R = 3 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 26 !
-------------------------------------------------------------------------------
! T = 5 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------

00100111:

        REG(5) = ALC, ALC = 0 + P2B, P2B = REG(D),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                 ----------------
! ADDR = 27 !                                                ! 5D5080001005F0 !
-------------------------------------------------------------------------------
! R = 5 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = D ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------
```

```
00101000:       *** ADI DATA8 ***
                *** ADD R     ***

        REG(7) = ALC,
            DO = ALC, ALC = P1B + P2B, P1B = REG(7), P2B = REG(SSS),
        COUT = GPUCOUT, MXH1OUT = OFL, MFLAGS = ENABLE(ALL),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                  ----------------
! ADDR = 28 !                                                ! 70D2A410101DF0 !
-------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = X ! D = 5 ! DOE = 1 !! TS  = 1 ! PC   = 0 ! SCY = 4 ! F  = 3 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 2 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ------------------------                 ------------
```

```
00101001:       *** ADD M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                                   ------------------
! ADDR = 29 !                                                  ! 05E0E000C000A0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  -------------

00101010:

        REG(7) = ALC,
            DO = ALC, ALC = P1B + P2B, P1B = REG(7), P2B = DI, DI = DATAIN,
        COUT = GPUCOUT, MXH1OUT = OFL, MFLAGS = ENABLE(ALL),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                   ------------------
! ADDR = 2A !                                                  ! 70D2B400101DF0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 5 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 4 ! F  = 3 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 2 ! C = 2 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
00101011:       *** ACI DATA8 ***
                *** ADC R     ***

        REG(7) = ALC,
            DO = ALC, ALC = P1B + P2B + COUT, P1B = REG(7), P2B = REG(SSS),
        COUT = GPUCOUT, MXH1OUT = OFL, MFLAGS = ENABLE(ALL),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ------------------
! ADDR = 2B  !                                                 ! 70D2A410109DF0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 4 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = X ! D = 5 ! DOE = 1 !! TS  = 1 ! PC   = 0 ! SCY = 4 ! F  = 3 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 2 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
00101100:       *** ADC M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                                   ------------------
! ADDR = 2C !                                                  ! 05E0E000C000A0 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  1 !
-----------------          ----------------------                    ------------

00101101:

        REG(7) = ALC,
            DO = ALC, ALC = P1B+P2B+COUT, P1B = REG(7), P2B = DI, DI = DATAIN,
        COUT = GPUCOUT, MXH1OUT = OFL, MFLAGS = ENABLE(ALL),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                   ------------------
! ADDR = 2D !                                                  ! 70D2B400109DF0 !
-------------------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 4 ! ADDR = F0 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 5 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 4 ! F  = 3 ! FLAG =  5 !
-------------------------------------------------------------------------------------------
! S = 2 ! C = 2 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

```
00101110:       *** SUI DATA8 ***
                *** SUB R     ***

        REG(7)' = ALC,
            DO = ALC, ALC = P1B - P2B, P1B = REG(7), P2B = REG(SSS),
        COUT = .NOT.GPUCOUT, MXH1OUT = OFL, MFLAGS = ENABLE(ALL),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

ADDR = 2E														70D22510103DF0	
R = 7	A = 0	M = 5		RS = 0		MIC = 1		I/O = 0		CI = 1		ADDR = F0			
T = X	D = 1	DOE = 1		TS = 1		PC = 0		SCY = 5		F = 3		FLAG = 5			
S = 0	C = 2			DIS = 0		PCHL = 0								LINK = 0	

```
00101111:       *** SUB M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                                   ------------------
! ADDR = 2F !                                                  ! 05E0E000C000A0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  ------------

00110000:

        REG(7) = ALC,
            DO = ALC, ALC = P1B - P2B, P1B = REG(7), P2B = DI, DI = DATAIN,
        COUT = .NOT. GPUCOUT, MXH1OUT = OFL, MFLAGS = ENABLE(ALL),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                   ------------------
! ADDR = 30 !                                                  ! 70D23500103DF0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 1 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 1 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 5 ! F  = 3 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 2 ! C = 2 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------
```

```
00110001:       *** SBI DATA8 ***
                *** SBB R     ***

        REG(7) = ALC,
            DO = ALC, ALC = P1B +. NOT. P2B +. NOT. COUT, P1B=REG(7), P2B=REG(SSS),
        COUT =  NOT. GPUCOUT, MXH1OUT = OFL, MFLAGS = ENABLE(ALL),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

ADDR = 31							70D2251010DDF0	
R = 7	A = 0	M = 5	RS = 0	MIC = 1	I/O = 0	CI = 6	ADDR = F0	
T = X	D = 1	DOE = 1	TS = 1	PC = 0	SCY = 5	F = 3	FLAG = 5	
S = 0	C = 2		DIS = 0	PCHL = 0			LINK = 0	

```
00110010:       *** SBB M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                           ------------------
! ADDR = 32 !                                          ! 05E0E000C000A0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  -------------

00110011:

        REG(7) = ALC,
            DO=ALC, ALC=P1B+.NOT.P2B+.NOT.COUT, P1B=REG(7), P2B=DI, DI=DATAIN,
        COUT = .NOT.GPUCOUT, MXH1OUT = OFL, MFLAGS = ENABLE(ALL),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                           ------------------
! ADDR = 33 !                                          ! 70D2350010DDF0 !
-------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 6 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 1 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 5 ! F  = 3 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 2 ! C = 2 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
00110100:       *** INR R ***

        REG(DDD) = ALC,
              DO = ALC, ALC = P1B + 0 + 1, P1B = REG(DDD),
        MFLAGS = ENABLE(SZP),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ----------------
! ADDR = 34 !                                                 ! 00D0E0401035F0 !
-------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 1 ! MIC  = 1 ! I/O = 0 ! CI = 1 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 2 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------
```

```
00110101:       *** INR M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                           ------------------
! ADDR = 35 !                                          ! 05E0E000C000A0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  -------------

00110110:

        ADL = DO, DO = ALC, ALC = P1B + 0 + 1, P1B = DI, DI = DATAIN,
        MFLAGS = ENABLE(SZP), NOLOAD,
        GOTO 10101000;                                  * MERGE INTO STORE

-------------                                           ------------------
! ADDR = 36 !                                          ! 00C0E8004030A8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 1 ! ADDR = A8 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 2 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
00110111:       *** DCR R ***

        REG(DDD) = ALC,
              DO = ALC, ALC = P1B +. NOT. P2B, P1B = REG(DDD), P2B = REG(C),
        MFLAGS = ENABLE(SZP),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                    ------------------
! ADDR = 37 !                                                   ! 0CD020401015F0 !
-------------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 1 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = C ! D = 1 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 2 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
00111000:       *** DCR M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                           ------------------
! ADDR = 38 !                                          ! 05E0E000C000A0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  -------------

00111001:

        ADL=DO, DO=ALC, ALC = P1B +. NOT. P2B, P1B=DI, DI=DATAIN, P2B=REG(C),
        MFLAGS = ENABLE(SZP), NOLOAD,
        GOTO 10101000;                                  * MERGE INTO STORE

-------------                                           ------------------
! ADDR = 39 !                                          ! 0CC028004010A8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A8 !
-------------------------------------------------------------------------------
! T = C ! D = 1 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 2 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
00111010:       *** INX RP ***

        REG(RP1) = ALC, ALC = P1B + 0 + 1, P1B = REG(RP1),
        GOTO 00111011;

-------------                                                  ------------------
! ADDR = 3A !                                                 ! 0050E0C000213A !
-------------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 3 ! MIC  = 0 ! I/O = 0 ! CI = 1 ! ADDR = 3A !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

00111011:

        REG(RP0) = ALC, ALC = P1B + 0 + CO, P1B = REG(RP0),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ------------------
! ADDR = 3B !                                                 ! 0050E0801045F0 !
-------------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 2 ! MIC  = 1 ! I/O = 0 ! CI = 2 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
00111100:       *** DCX RP ***

        REG(RP1) = ALC, ALC = P1B +. NOT. P2B, P1B = REG(RP1), P2B = REG(C),
        GOTO 00111101;

------------                                                  ------------------
! ADDR = 3C !                                                ! 0C5020C000013C !
-------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 3 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 3C !
-------------------------------------------------------------------------------
! T = C ! D = 1 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

00111101:

        REG(RP0) = ALC, ALC = P1B +. NOT. P2B + CO, P1B = REG(RP0), P2B = REG(C),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                  ------------------
! ADDR = 3D !                                                ! 0C5020801045F0 !
-------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 2 ! MIC  = 1 ! I/O = 0 ! CI = 2 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = C ! D = 1 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

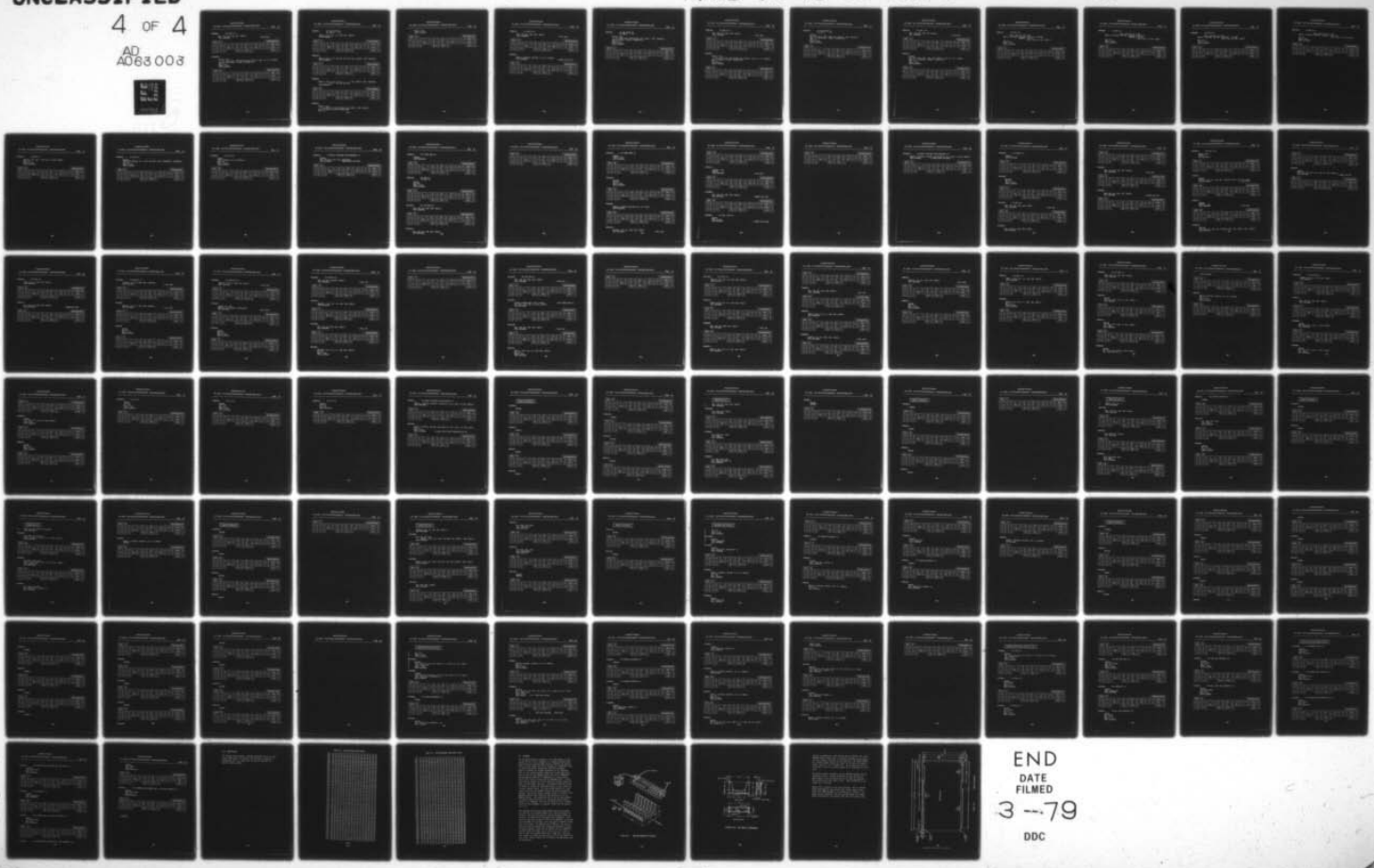
```
00111110:       *** DAD RP ***

        REG(5) = ALC, ALC = P1B + P2B, P1B = REG(5), P2B = REG(RP1),
        GOTO 00111111;

------------                                                  ------------------
! ADDR = 3E !                                                ! 5050A03000013E !
-------------------------------------------------------------------------------
! R = 5 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 3E !
-------------------------------------------------------------------------------
! T = X ! D = 5 ! DOE = 0 !! TS  = 3 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
---------------------      ------------------------                 -------------

00111111:

        REG(4) = ALC, ALC = P1B + P2B + CO, P1B = REG(4), P2B = REG(RP0),
        COUT = GPUCOUT,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                  ------------------
! ADDR = 3F !                                                ! 4050A4201045F0 !
-------------------------------------------------------------------------------
! R = 4 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 2 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = X ! D = 5 ! DOE = 0 !! TS  = 2 ! PC   = 0 ! SCY = 4 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------        ------------------------                 -------------
```

```
01000000:       *** ANI DATA8 ***
                *** ANA R     ***

        REG(7) = ALC,
            DO = ALC, ALC = P1B AND P2B, P1B = REG(7), P2B = REG(SSS),
        COUT = .NOT.GPUCOUT, MFLAGS = ENABLE(SZP),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

ADDR = 40							70D8A5101015F0	
R = 7	A = 2	M = 5	RS = 0	MIC = 1	I/O = 0	CI = 0	ADDR = F0	
T = X	D = 5	DOE = 1	TS = 1	PC = 0	SCY = 5	F = 2	FLAG = 5	
S = 0	C = 0		DIS = 0	PCHL = 0			LINK = 0	

AD-A063 003 QUESTRON CORP SAN DIEGO CALIF F/G 9/2
RADIATION HARDENED MICROPROCESSOR VOLUME I.(U)
MAY 78 V V NICKEL, P A ROSENBERG F33615-77-C-1001
UNCLASSIFIED AFAL-TR-78-55-VOL-1 NL

4 OF 4
AD
A063003



END
DATE
FILMED
3-79
DDC

```
01000001:       *** ANA M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                                   -----------------
! ADDR = 41 !                                                  ! 05E0E000C000A0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  1 !
-----------------          ----------------------                    ------------

01000010:

        REG(7) = ALC,
            DO = ALC, ALC = P1B AND P2B, P1B = REG(7), P2B = DI, DI = DATAIN,
        COUT = .NOT. GPUCOUT, MFLAGS = ENABLE(SZP),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                   -----------------
! ADDR = 42 !                                                  ! 70D8B5001015F0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 2 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 5 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 5 ! F  = 2 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 2 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

```
01000011:       *** XRI DATA8 ***
                *** XRA R     ***

        REG(D) = ALC, ALC = 0 + P2B, P2B = REG(7),
        GOTO 01000100;

-------------                                                   ------------------
! ADDR = 43 !                                                  ! D7508000000044 !
-------------------------------------------------------------------------------------
! R = D ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 44 !
-------------------------------------------------------------------------------------
! T = 7 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ------------------------                  -------------

01000100:

        REG(D) = ALC, ALC = P1B AND .NOT. P2B, P1B = REG(D), P2B = REG(SSS),
        GOTO 01000101;

-------------                                                   ------------------
! ADDR = 44 !                                                  ! D0582010000144 !
-------------------------------------------------------------------------------------
! R = D ! A = 2 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 44 !
-------------------------------------------------------------------------------------
! T = X ! D = 1 ! DOE = 0 !! TS  = 1 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ------------------------                  -------------

01000101:

        REG(7) = ALC, D = 6, A = 2,           P1B = REG(7), P2B = REG(SSS),
                    * ALC = .NOT. P1B AND P2B,
        GOTO 01000110;

-------------                                                   ------------------
! ADDR = 45 !                                                  ! 7058C010000046 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 2 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 46 !
-------------------------------------------------------------------------------------
! T = X ! D = 6 ! DOE = 0 !! TS  = 1 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ------------------------                  -------------

01000110:

        REG(7) = ALC,
            DO = ALC, ALC = P1B OR P2B, P1B = REG(7), P2B = REG(D),
        COUT = GPUCOUT, MFLAGS = ENABLE(SZP),
        AD = PC,
```

```
PCH/L = PCL,
GOTO 11110IHH;
```

```
-------------                                                    ------------------
! ADDR = 46 !                                                   ! 7DDCA4001015F0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 3 ! M  = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = D ! D = 5 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 4 ! F  = 2 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
01000111:       *** XRA M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                           ------------------
! ADDR = 47 !                                          ! 05E0E000C000A0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !         ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------         ---------------------                    -------------

01001000:

        REG(6) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        GOTO 01000011;                                  * MERGE INTO XRA R

-------------                                           ------------------
! ADDR = 48 !                                          ! 6000E800000142 !
-------------------------------------------------------------------------------
! R = 6 ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 42 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 1 ! C = 0 !         ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------         ---------------------                    -------------
```

```
01001001:       *** ORI DATA8 ***
                *** ORA R     ***

        REG(7) := ALC,
            DO = ALC, ALC = P1B OR P2B, P1B = REG(7), P2B = REG(SSS),
        COUT = GPUCOUT, MFLAGS = ENABLE(SZP),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ----------------
! ADDR = 49 !                                                 ! 70DCA4101015F0 !
-------------------------------------------------------------------------------
! R = 7 ! A = 3 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = X ! D = 5 ! DOE = 1 !! TS  = 1 ! PC   = 0 ! SCY = 4 ! F  = 2 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------
```

```
01001010:       *** ORA M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                           ------------------
! ADDR = 4A !                                          ! 05E0E000C000A0 !
-------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                ! LINK =  1 !
-----------------          ----------------------                -------------

01001011:

        REG(7) = ALC,
            DO = ALC, ALC = P1B OR P2B, P1B = REG(7), P2B = DI, DI = DATAIN,
        COUT = GPUCOUT, MFLAGS = ENABLE(SZP),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                           ------------------
! ADDR = 4B !                                          ! 70DCB4001015F0 !
-------------------------------------------------------------------------
! R = 7 ! A = 3 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------
! T = 0 ! D = 5 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 4 ! F  = 2 ! FLAG =  5 !
-------------------------------------------------------------------------
! S = 2 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                ! LINK =  0 !
-----------------          ----------------------                -------------
```

```
01001100:       *** CPI DATA8 ***
                *** CMP R     ***

        NOLOAD,
        DO = ALC, ALC = P1B - P2B, P1B = REG(7), P2B = REG(SSS),
        COUT = .NOT.GPUCOUT, MFLAGS = ENABLE(SZP),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ------------------
! ADDR = 4C !                                                 ! 70C025101035F0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 1 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = X ! D = 1 ! DOE = 1 !! TS  = 1 ! PC   = 0 ! SCY = 5 ! F  = 2 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
01001101:       *** CMP M ***

        ADL = DO, DO = P2B, P2B = REG(5),
        CALL 10100000;                                  * CALL OFCH

-------------                                           ------------------
! ADDR = 4D !                                          ! 05E0E000C000A0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  ------------

01001110:

        NOLOAD,
        DO = ALC, ALC = P1B - P2B, P1B = REG(7), P2B = DI, DI = DATAIN,
        COUT = .NOT. GPUCOUT, MFLAGS = ENABLE(SZP),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                           ------------------
! ADDR = 4E !                                          ! 70C035001035F0 !
-------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 1 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 1 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 5 ! F  = 2 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 2 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------
```

```
01001111:       *** RLC ***

            COUT = DO(7/0), DO = ALC,
        REG(7) = SHIFTER, SHIFTER = ALC(6-0) \\ MXL0IN,
                          MXH0OUT = ALC(7),
                                    ALC = P1B + 0, P1B = REG(7),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ----------------
! ADDR = 4F  !                                                 ! 7090E6001005F0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 1 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 6 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

```
01010000:       *** RRC ***

                          COUT = DO(7/0), DO = ALC,
        REG(7) = SHIFTER, SHIFTER = MXH0IN \\ ALC(7-1),
                                  MXL0OUT = ALC(0),
                                            ALC = P1B + 0, P1B = REG(7),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                  ----------------
! ADDR = 50 !                                                ! 70A0E6001005F0 !
-------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 2 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 6 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                ! LINK =  0 !
----------------           ---------------------                  ------------
```

```
01010001:       *** RAL ***

          COUT = DO(7/0), DO = ALC,
      REG(7) = SHIFTER, SHIFTER = ALC(6-0) \\ MXL0IN,
                                  ALC = P1B + 0, P1B = REG(7),

      AD = PC,
      PCH/L = PCL,
      GOTO 11110IHH;
```

! ADDR = 51 !						! 7090E6001005F0 !	
! R = 7 ! A = 0 ! M = 1 !! RS = 0 ! MIC = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !							
! T = 0 ! D = 7 ! DOE = 1 !! TS = 0 ! PC = 0 ! SCY = 6 ! F = 0 ! FLAG = 5 !							
! S = 0 ! C = 0 !		! DIS = 0 ! PCHL = 0 !				! LINK = 0 !	

```
01010010:       *** RAR ***

                          COUT = DO(7/0), DO = ALC,
        REG(7) = SHIFTER, SHIFTER = MXH0IN \\ ALC(7-1),
                                              ALC = P1B + 0, P1B = REG(7),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

```
-------------                                                  ------------------
! ADDR = 52 !                                                 ! 70A0E6001005F0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 2 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 6 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
01010011:       *** CMA ***

        REG(7) = ALC, ALC = .NOT. P1B + 0, P1B = REG(7),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ----------------
! ADDR = 53 !                                                 ! 705060001005F0 !
-------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 3 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
01010100:       *** CMC ***

        NOLOAD,
        COUT=.NOT.GPUCOUT, ALC = P1B +.NOT.P2B + COUT, P1B=REG(C), P2B=REG(C),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

ADDR = 54													CC4025001085F0		
R = C	A = 0	M = 4		RS = 0	MIC = 1	I/O = 0	CI = 4	ADDR = F0							
T = C	D = 1	DOE = 0		TS = 0	PC = 0	SCY = 5	F = 0	FLAG = 5							
S = 0	C = 0		DIS = 0	PCHL = 0									LINK = 0		

```
01010101:       *** STC ***

        NOLOAD,
        COUT = GPUCOUT, ALC = P1B AND 0,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                  ------------------
! ADDR = 55 !                                                ! 0048E4001005F0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 2 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 4 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                   ! LINK =  0 !
-----------------          ----------------------                   -------------
```

```
01010110:       *** RESET (CONTINUED FROM 00000000) ***

        NOLOAD,.
        PCL = DO, DO = ALC, ALC = P1B AND 0,
        GOTO 01100010;          * WHERE RESET CONTINUES

-------------                                                  -----------------
! ADDR = 56 !                                                 ! 00C8E000020062 !
-------------------------------------------------------------------------------
! R = 0 ! A = 2 ! M  = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 62 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 2 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ---------------------                   -------------
```

```
01010111:       *** JCOND ADDR ***

        NOLOAD,
        GOTO 0101100A;

------------                                                    ------------------
! ADDR = 57 !                                                  ! 0040E000000258 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 58 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  2 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
----------------           ----------------------                    ------------

01011000:       *** NOP ***
                *** DAA ***

        NOLOAD,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                    ------------------
! ADDR = 58 !                                                  ! 0040E0001005F0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
----------------           ----------------------                    ------------

01011001:       *** JMP ADDR ***

        PCL = DO, DO = P2B, P2B = REG(6),
        GOTO 01011010;

------------                                                    ------------------
! ADDR = 59 !                                                  ! 06E0E00002005A !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 5A !
-------------------------------------------------------------------------------------
! T = 6 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 2 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
----------------           ----------------------                    ------------

01011010:

        PCH = DO, DO = P2B, P2B = REG(B),
        GOTO 01011000;
```

```
-------------                                                    ------------------
! ADDR = 5A !                                                   ! 0BE0E000030058 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 58 !
-------------------------------------------------------------------------------------
! T = B ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 3 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
---------------            ---------------------                     ------------
```

```
01011011:       *** CCOND ADDR ***

        NOLOAD,
        PCH/L = PCH,
        GOTO 0101110A;

------------                                                  ------------------
! ADDR = 5B !                                                ! 0040E00004025C !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 5C !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  2 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 1 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

01011100:

        NOLOAD,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                  ------------------
! ADDR = 5C !                                                ! 0040E0001005F0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

01011101:

        REG(D) = SHIFTER, SHIFTER = DI, DI = PCH/L,
        GOTO 01011110;

------------                                                  ------------------
! ADDR = 5D !                                                ! D000E80008005E !
-------------------------------------------------------------------------------
! R = D ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 5E !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 1 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

01011110:

        DATAOUT = DO, DO = P2B, P2B = REG(D),
        CALL 10111000;                                  * CALL PUSH
```

```
------------                                                  ----------------
! ADDR = 5E !                                                ! 0DE0E000A000B8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI  = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------
! T = D ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F   = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
----------------           ---------------------                   ------------
```

```
01011111:

        DATAOUT = PCL,
        NOLOAD,
        CALL 10111000;                                  * CALL PUSH
```

```
------------                                                  ----------------
! ADDR = 5F !                                                ! 0040E000B000B8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 3 ! I/O = 0 ! CI  = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F   = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
----------------           ---------------------                   ------------
```

```
01100000:

        PCL = DO, DO = P2B, P2B = REG(6),
        GOTO 01011010;                                  * MERGE INTO JUMP
```

```
------------                                                  ----------------
! ADDR = 60 !                                                ! 06E0E00002005A !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI  = 0 ! ADDR = 5A !
-------------------------------------------------------------------------------
! T = 6 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 2 ! SCY = 0 ! F   = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
----------------           ---------------------                   ------------
```

```
01100001:       **** CALL ADDR ****

        NOLOAD,
        PCH/L = PCH,
        GOTO 01011101;                                  * MERGE INTO CCOND
```

```
-------------                                                  ----------------
! ADDR = 61 !                                                 ! 0040E00004015C !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 5C !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 1 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
01100010:       *** RESET CONTINUED FROM 01010110 ***

        REG(8) = SHIFTER, SHIFTER = 11\\ALC(7-2), ALC = P1B + 0, P1B = REG(8),
        GOTO 10010111;          * WHERE RESET CONTINUES

-------------                                                 ------------------
! ADDR = 62 !                                                ! 8032E000000196 !
-------------------------------------------------------------------------------
! R = 8 ! A = 0 ! M   = 3 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 96 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------
! S = 0 ! C = 2 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------
```

```
01100011:       *** RCOND ***

        NOLOAD,
        GOTO 0110010A;

-------------                                                  ------------------
! ADDR = 63 !                                                 ! 0040E000000264 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 64 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  2 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

01100100:

        NOLOAD,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ------------------
! ADDR = 64 !                                                 ! 0040E0001005F0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

01100101:       *** RET ***

        ADL = DO, DO = P2B, P2B = REG(F),
        CALL 10110000;                                  * CALL POP

-------------                                                  ------------------
! ADDR = 65 !                                                 ! 0FE0E000C000B0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = B0 !
-------------------------------------------------------------------------------------
! T = F ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  1 !
-----------------          ----------------------                    -------------

01100110:

        PCL = DO, DO = P2B, P2B = REG(D),
        GOTO 01100111;
```

```
------------                                                    ------------------
! ADDR = 66 !                                                  ! 0DE0E000020166 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M  = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 66 !
-------------------------------------------------------------------------------------
! T = D ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 2 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

01100111:

        ADL = DO, DO = P2B, P2B = REG(E),
        CALL 10110000;                                  * CALL POP

------------                                                    ------------------
! ADDR = 67 !                                                  ! 0EE0E000C000B0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M  = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = B0 !
-------------------------------------------------------------------------------------
! T = E ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  ------------

01101000:

        PCH = DO, DO = P2B, P2B = REG(D),
        GOTO 01011000;                                  * MERGE INTO NOP

------------                                                    ------------------
! ADDR = 68 !                                                  ! 0DE0E000030058 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M  = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 58 !
-------------------------------------------------------------------------------------
! T = D ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 3 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------
```

```
01101001:       *** RST N ***

        PCH/L = PCH,
        NOLOAD,
        GOTO 01101010;

-------------                                                   ------------------
! ADDR = 69 !                                                  ! 0040E00004006A !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 6A !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 1 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

01101010:

        NOLOAD,
        DATAOUT = DO, DO = ALC, ALC = P1B OR 0, P1B = DI, DI = PCH/L,
        CALL 10111000;                                          * CALL PUSH

-------------                                                   ------------------
! ADDR = 6A !                                                  ! 00CCE800A800B8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 3 ! M   = 4 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 1 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  -------------

01101011:

        NOLOAD,
        DATAOUT = PCL,
        CALL 10111000;                                          * CALL PUSH

-------------                                                   ------------------
! ADDR = 6B !                                                  ! 0040E000B000B8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 3 ! I/O = 0 ! CI = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  -------------

01101100:

        NOLOAD,
        PCL = DO, DO = ALC, ALC = P1B AND P2B, P1B = REG(A), P2B = REG(8),
        GOTO 01101101;
```

```
-------------                                                    ------------------
! ADDR = 6C !                                                   ! A8C8A00002016C !
-------------------------------------------------------------------------------------
! R = A ! A = 2 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 6C !
-------------------------------------------------------------------------------------
! T = 8 ! D = 5 ! DOE = 1 !! TS  = 0 ! PC   = 2 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
----------------           ----------------------                    -------------

01101101:

        NOLOAD,
        PCH = DO, DO = ALC, ALC = P1B + 0, P1B = REG(C),
        GOTO 01011000;                                    * MERGE INTO NOP

-------------                                                    ------------------
! ADDR = 6D !                                                   ! C0C0E000030058 !
-------------------------------------------------------------------------------------
! R = C ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 58 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 3 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
----------------           ----------------------                    -------------
```

```
01101110:       *** PCHL ***

        PCH = DO, DO = P2B, P2B = REG(4),
        GOTO 01101111;

-------------                                                    ------------------
! ADDR = 6E !                                                   ! 04E0E00003016E !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 6E !
-------------------------------------------------------------------------------------
! T = 4 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 3 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------

01101111:

        PCL = DO, DO = P2B, P2B = REG(5),
        GOTO 01011000;                                  * MERGE INTO NOP

-------------                                                    ------------------
! ADDR = 6F !                                                   ! 05E0E000020058 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 58 !
-------------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 2 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

```
01110000:       *** PUSH RP ***

        DATAOUT = DO, DO = P2B, P2B = REG(RP0),
        CALL 10111000;                                  * CALL PUSH

------------                                                    ------------------
! ADDR = 70 !                                                  ! 00E0E020A000B8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------------
! T = X ! D = 7 ! DOE = 1 !! TS  = 2 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  1 !
-----------------          ----------------------                    ------------

01110001:

        DATAOUT = DO, DO = P2B, P2B = REG(RP1),
        CALL 10111000;                                  * CALL PUSH

------------                                                    ------------------
! ADDR = 71 !                                                  ! 00E0E030A000B8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------------
! T = X ! D = 7 ! DOE = 1 !! TS  = 3 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  1 !
-----------------          ----------------------                    ------------

01110010:

        NOLOAD,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                    ------------------
! ADDR = 72 !                                                  ! 0040E0001005F0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

```
01110011:       *** PUSH PSW ***

        DATAOUT = DO, DO = P2B, P2B = REG(7),
        CALL 10111000;                                  * CALL PUSH

-------------                                           ------------------
! ADDR = 73 !                                          ! 07E0E000A000B8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------
! T = 7 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  -------------

01110100:

        NOLOAD, DO = OFF,
        DATAOUT = DO, MFLAGS = ENABLE(OUT),
        CALL 10111000;                                  * CALL PUSH

-------------                                           ------------------
! ADDR = 74 !                                          ! 0040E000A008B8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 1 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  -------------

01110101:

        NOLOAD,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                           ------------------
! ADDR = 75 !                                          ! 0040E0001005F0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
01110110:       *** POP RP ***

        ADL = DO, DO = P2B, P2B = REG(F),
        CALL 10110000;                          * CALL POP

-------------                                   ------------------
! ADDR = 76 !                                  ! 0FE0E000C000B0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = B0 !
-------------------------------------------------------------------------------
! T = F ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  ------------

01110111:

        REG(RP1) = ALC, ALC = 0 + P2B, P2B = REG(D),
        GOTO 01111000;

-------------                                   ------------------
! ADDR = 77 !                                  ! 0D5080C0000078 !
-------------------------------------------------------------------------------
! R = X ! A = 0 ! M   = 5 !! RS  = 3 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 78 !
-------------------------------------------------------------------------------
! T = D ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

01111000:

        ADL = DO, DO = P2B, P2B = REG(F),
        CALL 10110000;                          * CALL POP

-------------                                   ------------------
! ADDR = 78 !                                  ! 0FE0E000C000B0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = B0 !
-------------------------------------------------------------------------------
! T = F ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
-----------------          ----------------------                  ------------

01111001:

        REG(RP0) = ALC, ALC = 0 + P2B, P2B = REG(D),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

```
-------------                                                  ------------------
! ADDR = 79 !                                                 ! 0D5080801005F0 !
-------------------------------------------------------------------------------------------
! R = X ! A = 0 ! M  = 5 !! RS  = 2 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------------
! T = D ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                      ! LINK =  0 !
---------------            ----------------------                      ------------
```

```
01111010:       *** POP PSW ***

        ADL = DO, DO = P2B, P2B = REG(F),
        CALL 10110000;                                  * CALL POP

-------------                                                   ------------------
! ADDR = 7A !                                                  ! 0FE0E000C000B0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = B0 !
-------------------------------------------------------------------------------------
! T = F ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !           ! DIS = 0 ! PCHL = 0 !                    ! LINK =  1 !
-----------------           ----------------------                    ------------

01111011:

        MFLAGS = ENABLE(SZP), COUT = DO(0),             * SZP LOADED FROM DO
        DO = ALC, ALC = P1B + 0, P1B = REG(D),
        GOTO 01111100;

-------------                                                   ------------------
! ADDR = 7B !                                                  ! D090E70000107C !
-------------------------------------------------------------------------------------
! R = D ! A = 0 ! M   = 1 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 7C !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 7 ! F  = 2 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !           ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------           ----------------------                    ------------

01111100:

        ADL = DO, DO = P2B, P2B = REG(F),
        CALL 10110000;                                  * CALL POP

-------------                                                   ------------------
! ADDR = 7C !                                                  ! 0FE0E000C000B0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = B0 !
-------------------------------------------------------------------------------------
! T = F ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !           ! DIS = 0 ! PCHL = 0 !                    ! LINK =  1 !
-----------------           ----------------------                    ------------

01111101:

        REG(7) = ALC, ALC = 0 + P2B, P2B = REG(D),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

```
------------                                                    ----------------
! ADDR = 7D !                                                  ! 7D5080001005F0 !
-------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M  = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = D ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
----------------           --------------------                    ------------
```

```
01111110:       *** XTHL ***

        REG(9) = ALC, ALC = 0 + P2B, P2B = REG(5),
        GOTO 01111111;

-------------                                                  ------------------
! ADDR = 7E !                                                 ! 9550800000017E !
-------------------------------------------------------------------------------------
! R = 9 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 7E !
-------------------------------------------------------------------------------------
! T = 5 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------

01111111:

        REG(A) = ALC, ALC = 0 + P2B, P2B = REG(4),
        GOTO 10000000;

-------------                                                  ------------------
! ADDR = 7F !                                                 ! A4508000000080 !
-------------------------------------------------------------------------------------
! R = A ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 80 !
-------------------------------------------------------------------------------------
! T = 4 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------

10000000:

        ADL = DO, DO = P2B, P2B = REG(F),
        CALL 10110000;                                          * CALL POP

-------------                                                  ------------------
! ADDR = 80 !                                                 ! 0FE0E000C000B0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = B0 !
-------------------------------------------------------------------------------------
! T = F ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  1 !
-----------------          ----------------------                    ------------

10000001:

        REG(5) = ALC, ALC = 0 + P2B, P2B = REG(D),
        GOTO 10000010;
```

```
------------                                                  -----------------
! ADDR = 81 !                                                ! 5D508000000082 !
-------------------------------------------------------------------------------
! R = 5 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 82 !
-------------------------------------------------------------------------------
! T = D ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
---------------            --------------------                    -------------
```

10000010:

```
        ADL = DO, DO = P2B, P2B = REG(F),
        CALL 10110000;                                          * CALL POP
```

```
------------                                                  -----------------
! ADDR = 82 !                                                ! 0FE0E000C000B0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = B0 !
-------------------------------------------------------------------------------
! T = F ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
---------------            --------------------                    -------------
```

10000011:

```
        REG(4) = ALC, ALC = 0 + P2B, P2B = REG(D),
        GOTO 10000100;
```

```
------------                                                  -----------------
! ADDR = 83 !                                                ! 4D508000000084 !
-------------------------------------------------------------------------------
! R = 4 ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 84 !
-------------------------------------------------------------------------------
! T = D ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
---------------            --------------------                    -------------
```

10000100:

```
        DATAOUT = DO, DO = P2B, P2B = REG(A),
        CALL 10111000;                                          * CALL PUSH
```

```
------------                                                  -----------------
! ADDR = 84 !                                                ! 0AE0E000A000B8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------
! T = A ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  1 !
---------------            --------------------                    -------------
```

```
10000101:

        DATAOUT = DO, DO = P2B, P2B = REG(9),
        CALL 10111000;                                          * CALL PUSH

-------------                                                   ------------------
! ADDR = 85 !                                                  ! 09E0E000A000B8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------------
! T = 9 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  1 !
-----------------          ---------------------                     -------------

10000110:

        NOLOAD,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                   ------------------
! ADDR = 86 !                                                  ! 0040E0001005F0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ---------------------                     -------------
```

```
10000111:       *** SPHL ***

        REG(F) = ALC, ALC = 0 + P2B, P2B = REG(5),
        GOTO 10001000;

------------                                                  ------------------
! ADDR = 87 !                                                ! F5508000000088 !
-------------------------------------------------------------------------------
! R = F ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 88 !
-------------------------------------------------------------------------------
! T = 5 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------

10001000:

        REG(E) = ALC, ALC = 0 + P2B, P2B = REG(4),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                  ------------------
! ADDR = 88 !                                                ! E45080001005F0 !
-------------------------------------------------------------------------------
! R = E ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 4 ! D = 4 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------
```

```
10001001:       *** IN PORT ***

        ADL = DO, DO = P2B, P2B = REG(6),
        GOTO 10001010;

------------                                                    ------------------
! ADDR = 89 !                                                  ! 06E0E00040008A !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = 8A !
-------------------------------------------------------------------------------------
! T = 6 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10001010:

        NOLOAD,
        DO = ALC, ALC = P1B + 0, P1B = REG(C),
        GOTO 10001011;

------------                                                    ------------------
! ADDR = 8A !                                                  ! C0C0E00000018A !
-------------------------------------------------------------------------------------
! R = C ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 8A !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10001011:

        NOLOAD,
        DO = ALC, ALC = P1B + 0, P1B = REG(C),
        I/O = IOR,
        GOTO 10001100;

------------                                                    ------------------
! ADDR = 8B !                                                  ! C0C0E00E00008C !
-------------------------------------------------------------------------------------
! R = C ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = E ! CI = 0 ! ADDR = 8C !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10001100:

        NOLOAD,
        DO = ALC, ALC = P1B + 0, P1B = REG(C),
        I/O = IOR, DATAIN = M,
```

```
        GOTO 1000110R;

-------------                                                  ------------------
! ADDR = 8C !                                                 ! C0C0E00E60038C !
-------------------------------------------------------------------------------------
! R = C ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 6 ! I/O = E ! CI = 0 ! ADDR = 8C !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

10001101:

        REG(7) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ------------------
! ADDR = 8D !                                                 ! 7000E8001005F0 !
-------------------------------------------------------------------------------------
! R = 7 ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
10001110:       *** OUT PORT ***

        DATAOUT = DO, DO = P2B, P2B = REG(7),
        GOTO 10001111;

-------------                                                   ------------------
! ADDR = 8E !                                                  ! 07E0E00020018E !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 2 ! I/O = 0 ! CI = 0 ! ADDR = 8E !
-------------------------------------------------------------------------------------
! T = 7 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10001111:

        ADL = DO, DO = P2B, P2B = REG(6),
        GOTO 10010000;

-------------                                                   ------------------
! ADDR = 8F !                                                  ! 06E0E000400090 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = 90 !
-------------------------------------------------------------------------------------
! T = 6 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10010000:

        NOLOAD,
        DO = ALC, ALC = P1B + 0, P1B = REG(C),
        GOTO 10010001;

-------------                                                   ------------------
! ADDR = 90 !                                                  ! C0C0E000000190 !
-------------------------------------------------------------------------------------
! R = C ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 90 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10010001:

        NOLOAD,
        DO = ALC, ALC = P1B + 0, P1B = REG(C),
        I/O = IOW,
        GOTO 10010010;
```

```
------------                                                  ------------------
! ADDR = 91 !                                                ! C0C0E00F000092 !
-------------------------------------------------------------------------------
! R = C ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = F ! CI = 0 ! ADDR = 92 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ---------------------                    ------------
```

10010010:

```
        NOLOAD,
        DO = ALC, ALC = P1B + 0, P1B = REG(C),
        I/O = IOW,
        GOTO 1001001R;
```

```
------------                                                  ------------------
! ADDR = 92 !.                                               ! C0C0E00F000392 !
-------------------------------------------------------------------------------
! R = C ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = F ! CI = 0 ! ADDR = 92 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ---------------------                    ------------
```

10010011:

```
        NOLOAD,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

```
------------                                                  ------------------
! ADDR = 93 !                                                ! 0040E0001005F0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ---------------------                    ------------
```

```
10010100:       *** EI ***

        NOLOAD,
        I/O = EI,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                    ------------------
! ADDR = 94 !                                                   ! 0040E00D1005F0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = D ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
10010101:       *** DI ***

        NOLOAD,
        I/O = DI,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

------------                                                  ----------------
! ADDR = 95 !                                                ! 0040E00C1005F0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = C ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
----------------           ----------------------                  ------------
```

```
10010110:       *** HLT ***

        NOLOAD,
        I/O = EI,
        AD = PC,
        GOTO 11111IHH;

------------                                                    ----------------
! ADDR = 96 !                                                  ! 0040E00D1005F8 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = D ! CI = 0 ! ADDR = F8 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
----------------           ---------------------                     -------------
```

```
10010111:       *** RESET CONTINUED FROM 01100010 ***

        REG(8) = SHIFTER, SHIFTER = 1\\ALC(7-1), ALC = P1B + 0, P1B = REG(8),
        GOTO 10011000;

-------------                                                  ------------------
! ADDR = 97 !                                                 ! 8022E000000098 !
-------------------------------------------------------------------------------------
! R = 8 ! A = 0 ! M   = 2 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 98 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 2 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

10011000:

        REG(8) = SHIFTER, SHIFTER = 00\\ALC(7-2), ALC = P1B + 0, P1B = REG(8),
        AD = PC,
        PCH/L = PCL,
        GOTO 11110000;          * ENTER IFCH WITHOUT BRANCHING ON IHH

-------------                                                  ------------------
! ADDR = 98 !                                                 ! 8033E0001000F0 !
-------------------------------------------------------------------------------------
! R = 8 ! A = 0 ! M   = 3 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 3 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
*************************
*  UNUSED MICROWORDS  *
*************************

10011001:

        NOLOAD;

-------------                                                    ------------------
! ADDR = 99 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                   ! LINK =  0 !
-----------------          ------------------------                  -------------

10011010:

        NOLOAD;

-------------                                                    ------------------
! ADDR = 9A !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                   ! LINK =  0 !
-----------------          ------------------------                  -------------

10011011:

        NOLOAD;

-------------                                                    ------------------
! ADDR = 9B !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                   ! LINK =  0 !
-----------------          ------------------------                  -------------

10011100:

        NOLOAD;
```

```
------------                                                  ----------------
! ADDR = 9C !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
---------------            --------------------                    ------------
```

10011101:

NOLOAD;

```
------------                                                  ----------------
! ADDR = 9D !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
---------------            --------------------                    ------------
```

10011110:

NOLOAD;

```
------------                                                  ----------------
! ADDR = 9E !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
---------------            --------------------                    ------------
```

10011111:

NOLOAD;

```
------------                                                  ----------------
! ADDR = 9F !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
---------------            --------------------                    ------------
```

```
        ************************
        *  SUBROUTINE OFCH  *
        ************************

*       ADL = DO, DO = P2B, P2B = REG(5),
*       CALL 10100000;

10100000:

        DO = P2B, P2B = REG(4),
        GOTO 10100001;

-------------                                                    ------------------
! ADDR = A0 !                                                   ! 04E0E0000001A0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = A0 !
-------------------------------------------------------------------------------------
! T = 4 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------

10100001:

        DO = P2B, P2B = P2B,
        I/O = MEMR,
        GOTO 10100010;

-------------                                                    ------------------
! ADDR = A1 !                                                   ! 00E4E0040000A2 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 4 ! CI = 0 ! ADDR = A2 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------

10100010:

        DO = P2B, P2B = P2B,
        I/O = MEMR, DATAIN = M,
        GOTO 1010001R;

-------------                                                    ------------------
! ADDR = A2 !                                                   ! 00E4E0046003A2 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 6 ! I/O = 4 ! CI = 0 ! ADDR = A2 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

```
10100011:

        NOLOAD,
        RETURN;

------------                                                    ------------------
! ADDR = A3 !                                                  ! 0040E000000600 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  6 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                   ! LINK =  0 !
-----------------          ------------------------                  -------------
```

```
        **************************
        *  UNUSED MICROWORDS  *
        **************************

10100100:

        NOLOAD;

-------------                                                  ------------------
! ADDR = A4 !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10100101:

        NOLOAD;

-------------                                                  ------------------
! ADDR = A5 !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10100110:

        NOLOAD;

-------------                                                  ------------------
! ADDR = A6 !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10100111:

        NOLOAD;
```

```
-------------                                                    ------------------
! ADDR = A7 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
                    **********************
                    *  SUBROUTINE STORE  *
                    **********************

*       DATOUT = DO, DO = .....,
*       GOTO 10101000;

10101000:

        ADL = DO, DO = P2B, P2B = REG(5),
        GOTO 10101001;

------------                                                    ------------------
! ADDR = A8 !                                                  ! 05E0E0004001A8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = A8 !
-------------------------------------------------------------------------------------
! T = 5 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ---------------------                   ------------

10101001:

        DO = P2B, P2B = REG(4),
        GOTO 10101010;

------------                                                    ------------------
! ADDR = A9 !                                                  ! 04E0E0000000AA !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = AA !
-------------------------------------------------------------------------------------
! T = 4 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ---------------------                   ------------

10101010:

        DO = P2B, P2B = P2B,
        I/O = MEMW,
        GOTO 10101100;

------------                                                    ------------------
! ADDR = AA !                                                  ! 00E4E0030000AC !
-------------------------------------------------------------------------------------
! R = 0 ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 3 ! CI = 0 ! ADDR = AC !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ---------------------                   ------------
```

```
10101011:       *** UNUSED MICROWORD ***

        NOLOAD;

-------------                                                   ------------------
! ADDR = AB !                                                  ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10101100:

        DO = P2B, P2B = P2B,
        I/O = MEMW,
        GOTO 1010110R;

-------------                                                   ------------------
! ADDR = AC !                                                  ! 00E4E0030003AC !
-------------------------------------------------------------------------------------
! R = 0 ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 3 ! CI = 0 ! ADDR = AC !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10101101:

        NOLOAD,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                   ------------------
! ADDR = AD !                                                  ! 0040E0001005F0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
        ************************
        *  UNUSED MICROWORDS   *
        ************************

10101110:

        NOLOAD;

-------------                                                    ------------------
! ADDR = AE !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

10101111:

        NOLOAD;

-------------                                                    ------------------
! ADDR = AF !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------
```

```
                    **********
                    *  SUBROUTINE POP  *
                    **********

*       ADL = DO, DO = P2B, P2B = REG(F),
*       CALL 10110000;

10110000:

        DO = P2B, P2B = REG(E),
        REG(F) = ALC, ALC = P1B + 0 + 1, P1B = REG(F),
        GOTO 10110001;

-------------                                                  ------------------
! ADDR = B0 !                                                 ! FEE0E000021B0 !
-------------------------------------------------------------------------------------
! R = F ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 1 ! ADDR = B0 !
-------------------------------------------------------------------------------------
! T = E ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ------------------------                    -------------

10110001:

        DO = P2B, P2B = P2B,
        REG(E) = ALC, ALC = P1B + 0 + CO, P1B = REG(E),
        I/O = MEMR.STACK,
        GOTO 10110010;

-------------                                                  ------------------
! ADDR = B1 !                                                 ! E0E4E0060040B2 !
-------------------------------------------------------------------------------------
! R = E ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 6 ! CI = 2 ! ADDR = B2 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ------------------------                    -------------

10110010:

        DO = P2B, P2B = P2B,
        I/O = MEMR.STACK, DATAIN = M,
        GOTO 1011001R;
```

```
-------------                                                    ------------------
! ADDR = B2 !                                                   ! 00E4E0066003B2 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 6 ! I/O = 6 ! CI = 0 ! ADDR = B2 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ---------------------                     -------------
```

10110011:

```
        REG(D) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        RETURN;
```

```
-------------                                                    ------------------
! ADDR = B3 !                                                   ! D000E800000600 !
-------------------------------------------------------------------------------------
! R = D ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  6 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ---------------------                     -------------
```

```
        **************************
        *  UNUSED MICROWORDS  *
        **************************

10110100:

        NOLOAD;

-------------                                                  ------------------
! ADDR = B4 !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

10110101:

        NOLOAD;

-------------                                                  ------------------
! ADDR = B5 !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

10110110:

        NOLOAD;

-------------                                                  ------------------
! ADDR = B6 !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

10110111:

        NOLOAD;
```

```
------------                                                  ----------------
! ADDR = B7 !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !         ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
----------------          ---------------------                    ------------
```

```
                    **********************
                    *  SUBROUTINE PUSH   *
                    **********************

*       DATAOUT = DO, DO = P2B, P2B = REG(?),
*       CALL 10111000;

10111000:

        ADL = DO, DO = ALC,
              REG(F) = ALC, ALC = P1B +. NOT. P2B, P1B = REG(F), P2B = REG(C),
        GOTO 10111001;

-------------                                                  ------------------
! ADDR = B8 !                                                 ! FCD020004001B8 !
-------------------------------------------------------------------------------------
! R = F ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 4 ! I/O = 0 ! CI = 0 ! ADDR = B8 !
-------------------------------------------------------------------------------------
! T = C ! D = 1 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10111001:

        REG(E) = ALC, ALC = P1B +. NOT. P2B + CO, P1B = REG(E), P2B = REG(C),
        GOTO 10111010;

-------------                                                  ------------------
! ADDR = B9 !                                                 ! EC5020000040BA !
-------------------------------------------------------------------------------------
! R = E ! A = 0 ! M   = 5 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 2 ! ADDR = BA !
-------------------------------------------------------------------------------------
! T = C ! D = 1 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10111010:

        DO = P2B, P2B = REG(E),
        GOTO 10111011;

-------------                                                  ------------------
! ADDR = BA !                                                 ! 0EE0E0000001BA !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = BA !
-------------------------------------------------------------------------------------
! T = E ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
10111011:

        DO = P2B, P2B = P2B,
        I/O = MEMW. STACK,
        GOTO 10111100;

-------------                                                  ------------------
! ADDR = BB !                                                 ! 00E4E0020000BC !
-------------------------------------------------------------------------------------
! R = 0 ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 2 ! CI = 0 ! ADDR = BC !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10111100:

        DO = P2B, P2B = P2B,
        I/O = MEMW. STACK,
        GOTO 1011110R;

-------------                                                  ------------------
! ADDR = BC !                                                 ! 00E4E0020003BC !
-------------------------------------------------------------------------------------
! R = 0 ! A = 1 ! M   = 6 !! RS  = 0 ! MIC  = 0 ! I/O = 2 ! CI = 0 ! ADDR = BC !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

10111101:

        NOLOAD,
        RETURN;

-------------                                                  ------------------
! ADDR = BD !                                                 ! 0040E000000600 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  6 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
        ************************
        *  UNUSED MICROWORDS   *
        ************************

10111110:

        NOLOAD;

-------------                                                    ------------------
! ADDR = BE !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------

10111111:

        NOLOAD;

-------------                                                    ------------------
! ADDR = BF !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

```
        ******************************
        *  INTERRUPT JAM ROUTINE  *
        ******************************

*       .....,
*       AD = PC,
*       PCH/L = PCL,
*       GOTO 11110IHH;
*
*11110100:
*
*       NOLOAD,
*       I/O = INTA. IFCH,
*       GOTO 11000000;

11000000:

        NOLOAD,
        I/O = INTA. IFCH, DATAIN/OPC = M,
        GOTO 1100000R;

-------------                                                  ------------------
! ADDR = C0 !                                                 ! 0040E0017003C0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 7 ! I/O = 1 ! CI = 0 ! ADDR = C0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

11000001:

        REG(A) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        I/O = DI,
        AD = PC,
        GOTO 11000010;

-------------                                                  ------------------
! ADDR = C1 !                                                 ! A000E80C1000C2 !
-------------------------------------------------------------------------------------
! R = A ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 1 ! I/O = C ! CI = 0 ! ADDR = C2 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

11000010:

        NOLOAD,
        I/O = INTA. IFCH,
        GOTO 11000100;
```

```
-------------                                                  ------------------
! ADDR = C2 !                                                 ! 0040E0010000C4 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 1 ! CI = 0 ! ADDR = C4 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11000011:       *** UNUSED MICROWORD ***

        NOLOAD;

-------------                                                  ------------------
! ADDR = C3 !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11000100:

        NOLOAD,
        I/O = INTA. IFCH, DATAIN = M,
        GOTO 1100010R;

-------------                                                  ------------------
! ADDR = C4 !                                                 ! 0040E0016003C4 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 6 ! I/O = 1 ! CI = 0 ! ADDR = C4 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11000101:

        REG(6) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        AD = PC,
        GOTO 11000110;
```

```
-------------                                                    ------------------
! ADDR = C5 !                                                   ! 6000E8001000C6 !
-------------------------------------------------------------------------------------
! R = 6 ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = C6 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11000110:

        NOLOAD,
        I/O = INTA IFCH,
        GOTO 11001000;

-------------                                                    ------------------
! ADDR = C6 !                                                   ! 0040E0010000C8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 1 ! CI = 0 ! ADDR = C8 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11000111:       *** UNUSED MICROWORD ***

        NOLOAD;

-------------                                                    ------------------
! ADDR = C7 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11001000:

        NOLOAD,
        I/O = INTA IFCH, DATAIN = M,
        GOTO 1100100R;
```

```
-------------                                                    ------------------
! ADDR = C8 !                                                   ! 0040E0016003C8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 6 ! I/O = 1 ! CI = 0 ! ADDR = C8 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

11001001:

```
        REG(B) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        GOTO OPCODE;
```

```
-------------                                                    ------------------
! ADDR = C9 !                                                   ! B000E800000700 !
-------------------------------------------------------------------------------------
! R = B ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  7 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
**************************
*  UNUSED MICROWORDS  *
**************************

11001010:

        NOLOAD;

-------------                                                  ------------------
! ADDR = CA !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11001011:

        NOLOAD;

-------------                                                  ------------------
! ADDR = CB !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11001100:

        NOLOAD;

-------------                                                  ------------------
! ADDR = CC !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11001101:

        NOLOAD;
```

```
-------------                                                  ------------------
! ADDR = CD !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

11001110:

NOLOAD;

```
-------------                                                  ------------------
! ADDR = CE !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

11001111:

NOLOAD;

```
-------------                                                  ------------------
! ADDR = CF !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

11010000:

NOLOAD;

```
-------------                                                  ------------------
! ADDR = D0 !                                                 ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    ------------
```

11010001:

```
        NOLOAD;

-------------                                                    ------------------
! ADDR = D1 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ------------------------                 -------------

11010010:
        NOLOAD;

-------------                                                    ------------------
! ADDR = D2 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ------------------------                 -------------

11010011:
        NOLOAD;

-------------                                                    ------------------
! ADDR = D3 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ------------------------                 -------------

11010100:
        NOLOAD;

-------------                                                    ------------------
! ADDR = D4 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ------------------------                 -------------
```

```
11010101:

        NOLOAD;

-------------                                                    ------------------
! ADDR = D5 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                   ! LINK =  0 !
-----------------          ----------------------                   ------------

11010110:

        NOLOAD;

-------------                                                    ------------------
! ADDR = D6 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                   ! LINK =  0 !
-----------------          ----------------------                   ------------

11010111:

        NOLOAD;

-------------                                                    ------------------
! ADDR = D7 !                                                   ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                   ! LINK =  0 !
-----------------          ----------------------                   ------------

11011000:

        NOLOAD;
```

```
-------------------------------------------------------------------------------

-------------                                                 ----------------
! ADDR = D8 !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------

11011001:

        NOLOAD;

-------------                                                 ----------------
! ADDR = D9 !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------

11011010:

        NOLOAD;

-------------                                                 ----------------
! ADDR = DA !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------

11011011:

        NOLOAD;

-------------                                                 ----------------
! ADDR = DB !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 -------------

11011100:
```

```
        NOLOAD;

-------------                                                   ------------------
! ADDR = DC !                                                  ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

11011101:

        NOLOAD;

-------------                                                   ------------------
! ADDR = DD !                                                  ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

11011110:

        NOLOAD;

-------------                                                   ------------------
! ADDR = DE !                                                  ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

11011111:

        NOLOAD;

-------------                                                   ------------------
! ADDR = DF !                                                  ! 0040E000000000 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------
```

```
        **********************************
        *  INSTRUCTION FETCH ROUTINE  *
        **********************************

*       .....,
*       AD = PC,
*       PCH/L = PCL,
*       GOTO 11110IHH;
*
*11110000:
*
*       NOLOAD,
*       PCL = DO, DO = ALC, ALC = P1B + 0 + 1, P1B = DI, DI = PCH/L,
*       I/O = MEMR. IFCH,
*       PCH/L = PCH,
*       GOTO 11100000;

11100000:

        NOLOAD,
        PCH = DO, DO = ALC, ALC = P1B + 0 + CO, P1B = DI, DI = PCH/L,
        I/O = MEMR. IFCH, DATAIN/OPC = M,
        GOTO 1110001R;

------------                                                  ----------------
! ADDR = E0 !                                                ! 00C0E8057B43E2 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 7 ! I/O = 5 ! CI = 2 ! ADDR = E2 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 3 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 1 ! PCHL = 0 !                 ! LINK =  0 !
----------------           ----------------------                 ------------

11100001:       *** UNUSED MICROWORD ***

        NOLOAD;

------------                                                  ----------------
! ADDR = E1 !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
----------------           ----------------------                 ------------


11100010:

        NOLOAD,
        I/O = MEMR. IFCH, DATAIN/OPC = M,
        GOTO 1110001R;
```

```
-------------                                                  ------------------
! ADDR = E2 !                                                 ! 0040E0057003E2 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 7 ! I/O = 5 ! CI = 0 ! ADDR = E2 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

11100011:

```
        REG(A) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        AD = PC,
        PCH/L = PCL,
        GOTO 11100100;
```

```
-------------                                                  ------------------
! ADDR = E3 !                                                 ! A000E8001000E4 !
-------------------------------------------------------------------------------------------
! R = A ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = E4 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

11100100:

```
        NOLOAD,
        PCL = DO IF J, DO = ALC, ALC = P1B + 0 + 1, P1B = DI, DI = PCH/L,
        I/O = MEMR. IFCH,
        PCH/L = PCH,
        GOTO OPCODE;     * IF J THEN GOTO OPCODE;
```

```
-------------                                                  ------------------
! ADDR = E4 !                                                 ! 00C0E8050D2700 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 5 ! CI = 1 ! ADDR = 00 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 1 ! SCY = 0 ! F  = 0 ! FLAG =  7 !
-------------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 1 ! PCHL = 1 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------
```

```
                        *       ELSE GOTO 11100101;    HARD-WIRED
```

11100101:

```
        NOLOAD,
        PCH = DO, DO = ALC, ALC = P1B + 0 + CO, P1B = DI, DI = PCH/L,
        I/O = MEMR. IFCH, DATAIN = M,
        GOTO 111010RK;
```

```
-------------------------------------------------------------------------------

-------------                                                 ------------------
! ADDR = E5 !                                                ! 00C0E8056B44E8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 6 ! I/O = 5 ! CI = 2 ! ADDR = E8 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 3 ! SCY = 0 ! F  = 0 ! FLAG =  4 !
-------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 1 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11100110:       *** UNUSED MICROWORD ***

        NOLOAD;

-------------                                                 ------------------
! ADDR = E6 !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11100111:       *** UNUSED MICROWORD ***

        NOLOAD;

-------------                                                 ------------------
! ADDR = E7 !                                                ! 0040E000000000 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11101000:

        NOLOAD,
        I/O = MEMR. IFCH, DATAIN = M,
        GOTO 111010RK;

-------------                                                 ------------------
! ADDR = E8 !                                                ! 0040E005E004E8 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 6 ! I/O = 5 ! CI = 0 ! ADDR = E8 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  4 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------
```

```
11101001:

        NOLOAD,
        I/O = MEMR. IFCH, DATAIN = M,
        GOTO 111010RK;

-------------                                                  ------------------
! ADDR = E9 !                                                 ! 0040E0056004E8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 6 ! I/O = 5 ! CI = 0 ! ADDR = E8 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  4 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

11101010:

        REG(6) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        GOTO OPCODE;

-------------                                                  ------------------
! ADDR = EA !                                                 ! 6000E800000700 !
-------------------------------------------------------------------------------------
! R = 6 ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  7 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

11101011:

        REG(6) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        AD = PC,
        PCH/L = PCL,
        GOTO 11101100;

-------------                                                  ------------------
! ADDR = EB !                                                 ! 6000E8001000EC !
-------------------------------------------------------------------------------------
! R = 6 ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 1 ! I/O = 0 ! CI = 0 ! ADDR = EC !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
-----------------          ----------------------                    -------------

11101100:

        NOLOAD,
        PCL = DO, DO = ALC, ALC = P1B + 0 + 1, P1B = DI, DI = PCH/L,
        I/O = MEMR. IFCH,
```

```
        PCH/L = PCH,
        GOTO 11101101;

-------------                                                    ------------------
! ADDR = EC !                                                   ! 00C0E8050E21EC !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 5 ! CI = 1 ! ADDR = EC !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 2 ! SCY = 0 ! F  = 0 ! FLAG =  1 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 1 ! PCHL = 1 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11101101:

        NOLOAD,
        PCH = DO, DO = ALC, ALC = P1B + 0 + CO, P1B = DI, DI = PCH/L,
        I/O = MEMR. IFCH, DATAIN = M,
        GOTO 1110111R;

-------------                                                    ------------------
! ADDR = ED !                                                   ! 00C0E8056B43EE !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 6 ! I/O = 5 ! CI = 2 ! ADDR = EE !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 3 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 1 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11101110:

        NOLOAD,
        I/O = MEMR. IFCH, DATAIN = M,
        GOTO 1110111R;

-------------                                                    ------------------
! ADDR = EE !                                                   ! 0040E0056003EE !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 6 ! I/O = 5 ! CI = 0 ! ADDR = EE !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  3 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11101111:

        REG(B) = SHIFTER, SHIFTER = DI, DI = DATAIN,
        GOTO OPCODE;
```

```
-------------                                                  ------------------
! ADDR = EF !                                                 ! B000E800000700 !
-------------------------------------------------------------------------------------------
! R = B ! A = 0 ! M   = 0 !! RS  = 0 ! MIC  = 0 ! I/O = 0 ! CI = 0 ! ADDR = 00 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  7 !
-------------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                    ! LINK =  0 !
----------------            ---------------------                     ------------
```

```
        ****************************************
        *  END-OF-INSTRUCTION BRANCH ON IHH    *
        ****************************************

11110000:       *** IFCH ***

        NOLOAD,
        PCL = DO, DO = ALC, ALC = P1B + 0 + 1, P1B = DI, DI = PCH/L,
        I/O = MEMR. IFCH,
        PCH/L = PCH,
        GOTO 11100000;

-------------                                                  ------------------
! ADDR = F0 !                                                 ! 00C0E8050E20E0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 5 ! CI = 1 ! ADDR = E0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 1 !! TS  = 0 ! PC   = 2 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 1 ! C = 0 !          ! DIS = 1 ! PCHL = 1 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11110001:       *** HOLD ***

        NOLOAD,
        I/O = HLDA,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ------------------
! ADDR = F1 !                                                 ! 0040E0081005F0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 8 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11110010:       *** HALT ***

        NOLOAD,
        I/O = WAIT,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

```
-------------                                                  ------------------
! ADDR = F2 !                                                 ! 0040E00B1005F0 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = B ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11110011:       *** HALT AND HOLD ***

        NOLOAD,
        I/O = HLDA. WAIT,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;

-------------                                                  ------------------
! ADDR = F3 !                                                 ! 0040E0091005F0 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 9 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11110100:       *** INTERRUPT ***

        NOLOAD,
        I/O = INTA. IFCH,
        GOTO 11000000;

-------------                                                  ------------------
! ADDR = F4 !                                                 ! 0040E0010000C0 !
-------------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 1 ! CI = 0 ! ADDR = C0 !
-------------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11110101:       *** HOLD AND INTERRUPT ***

        NOLOAD,
        I/O = HLDA,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

```
------------                                                  ------------------
! ADDR = F5 !                                                ! 0040E0081005F0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 8 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 ------------
```

```
11110110:       *** HALT AND INTERRUPT ***

        NOLOAD,
        I/O = WAIT,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

```
------------                                                  ------------------
! ADDR = F6 !                                                ! 0040E00B1005F0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = B ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 ------------
```

```
11110111:       *** HOLD, HALT AND INTERRUPT ***

        NOLOAD,
        I/O = HLDA.WAIT,
        AD = PC,
        PCH/L = PCL,
        GOTO 11110IHH;
```

```
------------                                                  ------------------
! ADDR = F7 !                                                ! 0040E0091005F0 !
-------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 9 ! CI = 0 ! ADDR = F0 !
-------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                 ! LINK =  0 !
-----------------          ----------------------                 ------------
```

```
        ************************************
        *  HALT-INSTRUCTION BRANCH ON IHH  *
        ************************************

11111000:       *** INTERNAL HALT ***

        NOLOAD,
        I/O = WAIT,
        AD = PC,
        GOTO 11111IHH;
```

ADDR = F8													0040E00B1005F8
R = 0	A = 0	M = 4	RS = 0	MIC = 1	I/O = B	CI = 0	ADDR = F8						
T = 0	D = 7	DOE = 0	TS = 0	PC = 0	SCY = 0	F = 0	FLAG = 5						
S = 0	C = 0	DIS = 0	PCHL = 0	LINK = 0									

```
11111001:       *** INTERNAL HALT AND HOLD ***

        NOLOAD,
        I/O = HLDA. WAIT,
        AD = PC,
        GOTO 11111IHH;
```

ADDR = F9													0040E0091005F8
R = 0	A = 0	M = 4	RS = 0	MIC = 1	I/O = 9	CI = 0	ADDR = F8						
T = 0	D = 7	DOE = 0	TS = 0	PC = 0	SCY = 0	F = 0	FLAG = 5						
S = 0	C = 0	DIS = 0	PCHL = 0	LINK = 0									

```
11111010:       *** INTERNAL AND EXTERNAL HALT ***

        NOLOAD,
        I/O = WAIT,
        AD = PC,
        GOTO 11111IHH;
```

ADDR = FA													0040E00B1005F8
R = 0	A = 0	M = 4	RS = 0	MIC = 1	I/O = B	CI = 0	ADDR = F8						
T = 0	D = 7	DOE = 0	TS = 0	PC = 0	SCY = 0	F = 0	FLAG = 5						
S = 0	C = 0	DIS = 0	PCHL = 0	LINK = 0									

```
11111011:       *** INTERNAL AND EXTERNAL HALT AND HOLD ***

        NOLOAD,
        I/O = HLDA. WAIT,
        AD = PC,
        GOTO 11111IHH;

-------------                                                  ------------------
! ADDR = FB !                                                 ! 0040E0091005F8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 9 ! CI = 0 ! ADDR = F8 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11111100:       *** INTERRUPT ***

        NOLOAD,
        I/O = INTA. IFCH,
        GOTO 11000000;

-------------                                                  ------------------
! ADDR = FC !                                                 ! 0040E0010000C0 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 0 ! I/O = 1 ! CI = 0 ! ADDR = C0 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  0 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11111101:       *** INTERNAL HALT, HOLD AND INTERRUPT ***

        NOLOAD,
        I/O = HLDA. WAIT,
        AD = PC,
        GOTO 11111IHH;

-------------                                                  ------------------
! ADDR = FD !                                                 ! 0040E0091005F8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 9 ! CI = 0 ! ADDR = F8 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  -------------

11111110:       *** INTERNAL AND EXTERNAL HALT AND INTERRUPT ***
```

```
        NOLOAD,
        I/O = WAIT,
        AD = PC,
        GOTO 11111IHH;

-------------                                                    ------------------
! ADDR = FE !                                                   ! 0040E00B1005F8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = B ! CI = 0 ! ADDR = F8 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------

11111111:       *** INTERNAL AND EXTERNAL HALT, HOLD AND INTERRUPT ***

        NOLOAD,
        I/O = HLDA. WAIT,
        AD = PC,
        GOTO 11111IHH;

-------------                                                    ------------------
! ADDR = FF !                                                   ! 0040E0091005F8 !
-------------------------------------------------------------------------------------
! R = 0 ! A = 0 ! M   = 4 !! RS  = 0 ! MIC  = 1 ! I/O = 9 ! CI = 0 ! ADDR = F8 !
-------------------------------------------------------------------------------------
! T = 0 ! D = 7 ! DOE = 0 !! TS  = 0 ! PC   = 0 ! SCY = 0 ! F  = 0 ! FLAG =  5 !
-------------------------------------------------------------------------------------
! S = 0 ! C = 0 !          ! DIS = 0 ! PCHL = 0 !                  ! LINK =  0 !
-----------------          ----------------------                  ------------
```

0 ERRORS

4.3.4 ROM Pattern

The following listing represents a complete, hexidecimal map of the microprogram ROM pattern. It should be noted that since it represents the microprogram before any actual check out has been performed, it is bound to change somewhat.

Table 4.3-1 Q-80 Microprogram ROM Pattern

```
000  C0 D8 E0 00 03 00 56 00 50 80 50 10 05 F0 05 E0
010  E0 00 C0 00 A0 00 00 E8 40 10 05 F0 00 E0 E0 10
020  20 00 A8 0B 50 80 80 00 00 06 06 50 80 C0 10 05
030  F0 06 E0 E0 00 40 00 08 0B E0 E0 00 80 01 A0 70
040  00 E8 00 10 05 F0 07 E0 E0 00 20 01 0A 06 E0 E0
050  00 40 00 0C 0B E0 E0 00 00 00 AA 06 E0 E0 00 40
060  00 0E 0B E0 E0 00 80 01 A0 50 00 E8 00 00 00 10
070  60 50 E0 00 00 21 10 B6 E0 E0 00 40 40 12 0B E0
080  E0 00 80 01 A0 40 00 E8 00 10 05 F0 05 E0 E0 00
090  20 01 14 0B E0 E0 00 40 00 16 0B E0 E0 00 00 01
0A0  16 00 E4 E0 03 00 00 18 00 E4 E0 03 00 03 18 64
0B0  E0 E0 00 20 20 1A B6 E0 E0 00 40 41 1A 0B E0 E0
0C0  00 00 00 AA 00 E0 E0 30 40 01 1C 00 E0 E0 20 80
0D0  01 A0 70 00 E8 00 10 05 F0 07 E0 E0 00 20 00 20
0E0  00 E0 E0 30 40 01 20 00 E0 E0 30 00 00 AA D2 50
0F0  80 00 00 01 22 24 50 80 00 00 00 24 4D 50 80 00
100  00 01 24 D3 50 80 00 00 00 26 35 50 80 00 00 01
110  26 5D 50 80 00 10 05 F0 70 D2 A4 10 10 1D F0 05
120  E0 E0 00 C0 00 A0 70 D2 B4 00 10 1D F0 70 D2 A4
130  10 10 9D F0 05 E0 E0 00 C0 00 A0 70 D2 B4 00 10
140  9D F0 70 D2 25 10 10 3D F0 05 E0 E0 00 C0 00 A0
150  70 D2 35 00 10 3D F0 70 D2 25 10 10 DD F0 05 E0
160  E0 00 C0 00 A0 70 D2 35 00 10 DD F0 00 D0 E0 40
170  10 35 F0 05 E0 E0 00 C0 00 A0 00 C0 E8 00 40 30
180  A8 0C D0 20 40 10 15 F0 05 E0 E0 00 C0 00 A0 0C
190  C0 28 00 40 10 A8 00 50 E0 C0 00 21 3A 00 50 E0
1A0  80 10 45 F0 0C 50 20 C0 00 01 3C 0C 50 20 80 10
1B0  45 F0 50 50 A0 30 00 01 3E 40 50 A4 20 10 45 F0
1C0  70 D8 A5 10 10 15 F0 05 E0 E0 00 C0 00 A0 70 D8
1D0  B5 00 10 15 F0 D7 50 80 00 00 00 44 D0 58 20 10
1E0  00 01 44 70 58 C0 10 00 00 46 7D DC A4 00 10 15
1F0  F0 05 E0 E0 00 C0 00 A0 60 00 E8 00 00 01 42 70
200  DC A4 10 10 15 F0 05 E0 E0 00 C0 00 A0 70 DC B4
210  00 10 15 F0 70 C0 25 10 10 35 F0 05 E0 E0 00 C0
220  00 A0 70 C0 35 00 10 35 F0 70 90 E6 00 10 05 F0
230  70 A0 E6 00 10 05 F0 70 90 E6 00 10 05 F0 70 A0
240  E6 00 10 05 F0 70 50 60 00 10 05 F0 CC 40 25 00
250  10 85 F0 00 48 E4 00 10 05 F0 00 C8 E0 00 02 00
260  62 00 40 E0 00 00 02 58 00 40 E0 00 10 05 F0 06
270  E0 E0 00 02 00 5A 0B E0 E0 00 03 00 58 00 40 E0
280  00 04 02 5C 00 40 E0 00 10 05 F0 D0 00 E8 00 08
290  00 5E 0D E0 E0 00 A0 00 B8 00 40 E0 00 B0 00 B8
2A0  06 E0 E0 00 02 00 5A 00 40 E0 00 04 01 5C 80 32
2B0  E0 00 00 01 96 00 40 E0 00 00 02 64 00 40 E0 00
2C0  10 05 F0 0F E0 E0 00 C0 00 B0 0D E0 E0 00 02 01
2D0  66 0E E0 E0 00 C0 00 B0 0D E0 E0 00 03 00 58 00
2E0  40 E0 00 04 00 6A 00 CC E8 00 A8 00 B8 00 40 E0
2F0  00 B0 00 B8 A8 C8 A0 00 02 01 6C C0 C0 E0 00 03
300  00 58 04 E0 E0 00 03 01 6E 05 E0 E0 00 02 00 58
310  00 E0 E0 20 A0 00 B8 00 E0 E0 30 A0 00 B8 00 40
320  E0 00 10 05 F0 07 E0 E0 00 A0 00 B8 00 40 E0 00
330  A0 08 B8 00 40 E0 00 10 05 F0 0F E0 E0 00 C0 00
340  B0 0D 50 80 C0 00 00 78 0F E0 E0 00 C0 00 B0 0D
350  50 80 80 10 05 F0 0F E0 E0 00 C0 00 B0 D0 90 E7
360  00 00 10 7C 0F E0 E0 00 C0 00 B0 7D 50 80 00 10
370  05 F0 95 50 80 00 00 01 7E A4 50 80 00 00 00 80
```

(cont.)

Table 4.3-1 Q-80 Microprogram ROM Pattern (cont.)

```
380  0F E0 E0 00 C0 00 B0 5D 50 80 00 00 00 82 0F E0
390  E0 00 C0 00 B0 4D 50 80 00 00 00 84 0A E0 E0 00
3A0  A0 00 B8 09 E0 E0 00 A0 00 B8 00 40 E0 00 10 05
3B0  F0 F5 50 80 00 00 00 88 E4 50 80 00 10 05 F0 06
3C0  E0 E0 00 40 00 8A C0 C0 E0 00 00 01 8A C0 C0 E0
3D0  0E 00 00 8C C0 C0 E0 0E 60 03 8C 70 00 E8 00 10
3E0  05 F0 07 E0 E0 00 20 01 8E 06 E0 E0 00 40 00 90
3F0  C0 C0 E0 00 00 01 90 C0 C0 E0 0F 00 00 92 C0 C0
400  E0 0F 00 03 92 00 40 E0 00 10 05 F0 00 40 E0 0D
410  10 05 F0 00 40 E0 0C 10 05 F0 00 40 E0 0D 10 05
420  F8 80 22 E0 00 00 00 98 80 33 E0 00 10 00 F0 00
430  40 E0 00 00 00 00 00 40 E0 00 00 00 00 00 40 E0
440  00 00 00 00 00 40 E0 00 00 00 00 00 40 E0 00 00
450  00 00 00 40 E0 00 00 00 00 00 40 E0 00 00 00 00
460  04 E0 E0 00 00 01 A0 00 E4 E0 04 00 00 A2 00 E4
470  E0 04 60 03 A2 00 40 E0 00 00 06 00 00 40 E0 00
480  00 00 00 00 40 E0 00 00 00 00 00 40 E0 00 00 00
490  00 00 40 E0 00 00 00 00 05 E0 E0 00 40 01 A8 04
4A0  E0 E0 00 00 00 AA 00 E4 E0 03 00 00 AC 00 40 E0
4B0  00 00 00 00 00 E4 E0 03 00 03 AC 00 40 E0 00 10
4C0  05 F0 00 40 E0 00 00 00 00 00 40 E0 00 00 00 00
4D0  FE E0 E0 00 00 21 B0 E0 E4 E0 06 00 40 B2 00 E4
4E0  E0 06 60 03 B2 D0 00 E8 00 00 06 00 00 40 E0 00
4F0  00 00 00 00 40 E0 00 00 00 00 00 40 E0 00 00 00
500  00 00 40 E0 00 00 00 00 FC D0 20 00 40 01 B8 EC
510  50 20 00 00 40 BA 0E E0 E0 00 00 01 BA 00 E4 E0
520  02 00 00 BC 00 E4 E0 02 00 03 BC 00 40 E0 00 00
530  06 00 00 40 E0 00 00 00 00 00 40 E0 00 00 00 00
540  00 40 E0 01 70 03 C0 A0 00 E8 0C 10 00 C2 00 40
550  E0 01 00 00 C4 00 40 E0 00 00 00 00 00 40 E0 01
560  60 03 C4 60 00 E8 00 10 00 C6 00 40 E0 01 00 00
570  C8 00 40 E0 00 00 00 00 00 40 E0 01 60 03 C8 B0
580  00 E8 00 00 07 00 00 40 E0 00 00 00 00 00 40 E0
590  00 00 00 00 00 40 E0 00 00 00 00 00 40 E0 00 00
5A0  00 00 00 40 E0 00 00 00 00 00 40 E0 00 00 00 00
5B0  00 40 E0 00 00 00 00 00 40 E0 00 00 00 00 00 40
5C0  E0 00 00 00 00 00 40 E0 00 00 00 00 00 40 E0 00
5D0  00 00 00 00 40 E0 00 00 00 00 00 40 E0 00 00 00
5E0  00 00 40 E0 00 00 00 00 00 40 E0 00 00 00 00 00
5F0  40 E0 00 00 00 00 00 40 E0 00 00 00 00 00 40 E0
600  00 00 00 00 00 40 E0 00 00 00 00 00 40 E0 00 00
610  00 00 00 40 E0 00 00 00 00 00 40 E0 00 00 00 00
620  00 C0 E8 05 7B 43 E2 00 40 E0 00 00 00 00 00 40
630  E0 05 70 03 E2 A0 00 E8 00 10 00 E4 00 C0 E8 05
640  0D 27 00 00 C0 E8 05 6B 44 E8 00 40 E0 00 00 00
650  00 00 40 E0 00 00 00 00 00 40 E0 05 60 04 E8 00
660  40 E0 05 60 04 E8 60 00 E8 00 00 07 00 60 00 E8
670  00 10 00 EC 00 C0 E8 05 0E 21 EC 00 C0 E8 05 6B
680  43 EE 00 40 E0 05 60 03 EE 00 00 E8 00 00 07 00
690  00 C0 E8 05 0E 20 E0 00 40 E0 08 10 05 F0 00 40
6A0  E0 0B 10 05 F0 00 40 E0 09 10 05 F0 00 40 E0 01
6B0  00 00 C0 00 40 E0 08 10 05 F0 00 40 E0 0B 10 05
6C0  F0 00 40 E0 09 10 05 F0 00 40 E0 0B 10 05 F8 00
6D0  40 E0 09 10 05 F8 00 40 E0 0B 10 05 F8 00 40 E0
6E0  09 10 05 F8 00 40 E0 01 00 00 C0 00 40 E0 09 10
6F0  05 F8 00 40 E0 0B 10 05 F8 00 40 E0 09 10 05 F8
```

4.4 Packaging

The breadboard hardware is designed to be an LSI simulator; i.e. each eventual LSI device will be fabricated on a standard wire-wrap module utilizing small scale integrated (SSI) devices to implement the function. These wire wrap modules will then be mounted in a motherboard chasis in which all intermodule communication paths will be handled. Figure 4.4-1 illustrates the physical characteristics of the breadboard. There are nine wire-wrap modules required for the basic Q-80 central processor unit (μA, μB, DECODE, FLAG, GPU, and 4 INTERFACE). The microprogram RAM will be implemented on a single wire-wrap module with a cable to interface with the MDS-800 system. As LSI devices become available, they will be placed on wire-wrap modules and replace the SSI versions one-for-one. Two additional wire-wrap modules are required to complete the breadboard interfaces—a panel module and a memory simulator module. The panel module interfaces to a maintenance panel which will facilitate checkout and periodic maintenance functions. The memory simulator interface provides a connection to the MDS-800 mainframe which allows the Q-80 breadboard to use the MDS-800 memory. This configuration will be used for initial hardware and firmware checkout because of the considerable support software available in the MDS-800. After that the breadboard can be expanded (consult the next paragraph) to incorporate memory and I/O interfaces of its own.

The breadboard chasis will be initially implemented with six standard card cages that can be ganged together into a single motherboard configuration. Each card cage holds two universal wire-wrap modules; so a total of 12 modules can be connected in this configuration. Additional card cages can be attached to this basic configuration, for memory and I/O interfaces, in the future. The card cages are provided by Intel; there are two types – the SBC-604 and the SBC-614. The only differences are that the SBC-604 has a single male motherboard connector and signal termination circuits while the SBC-614 has two motherboard connectors (male and female), one on each end of the card cage so that the cages can be ganged together into a single unit. Each card cage contains the required connectors (P1 and P2) for the universal wire-wrap boards. Refer to Figure 4.4-2 for details on the SBC-604/614 physical characteristics.

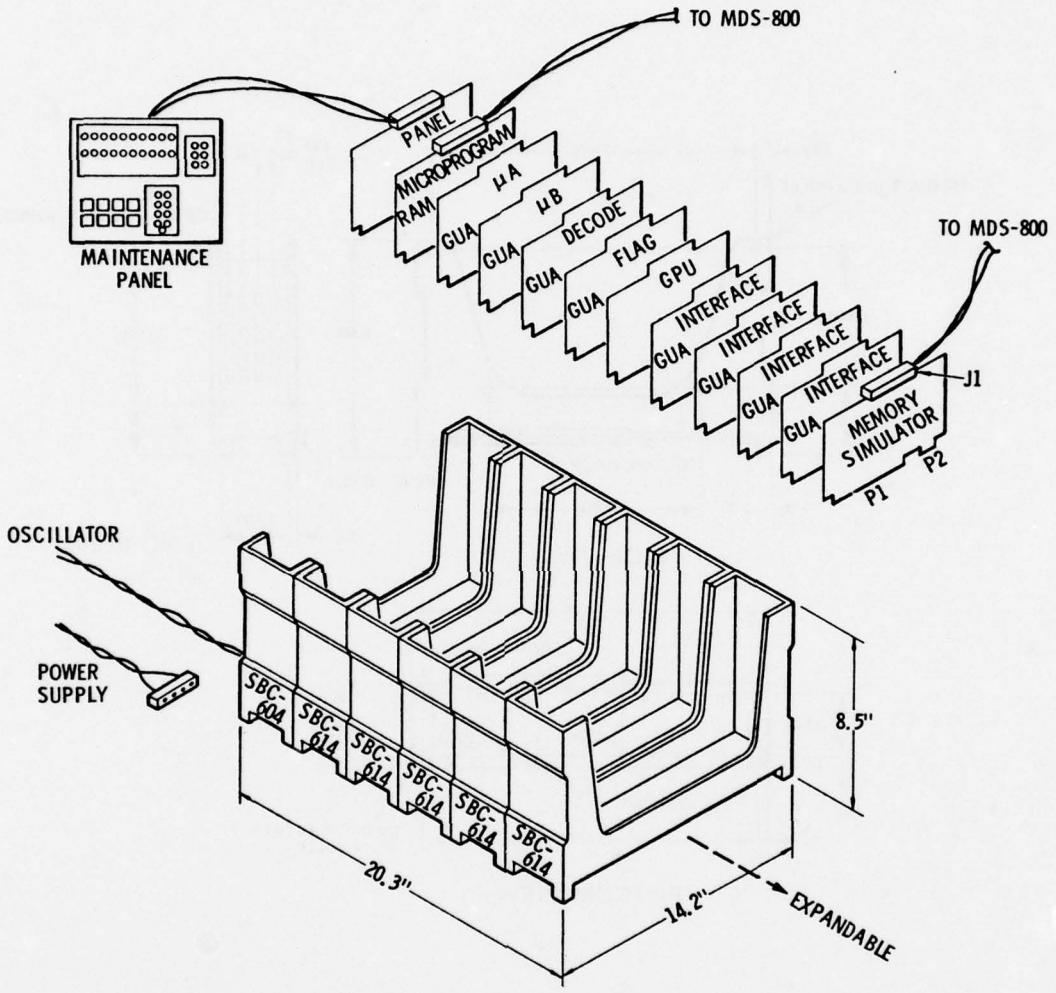


Figure 4.4-1 Q80 BREADBOARD PACKAGE

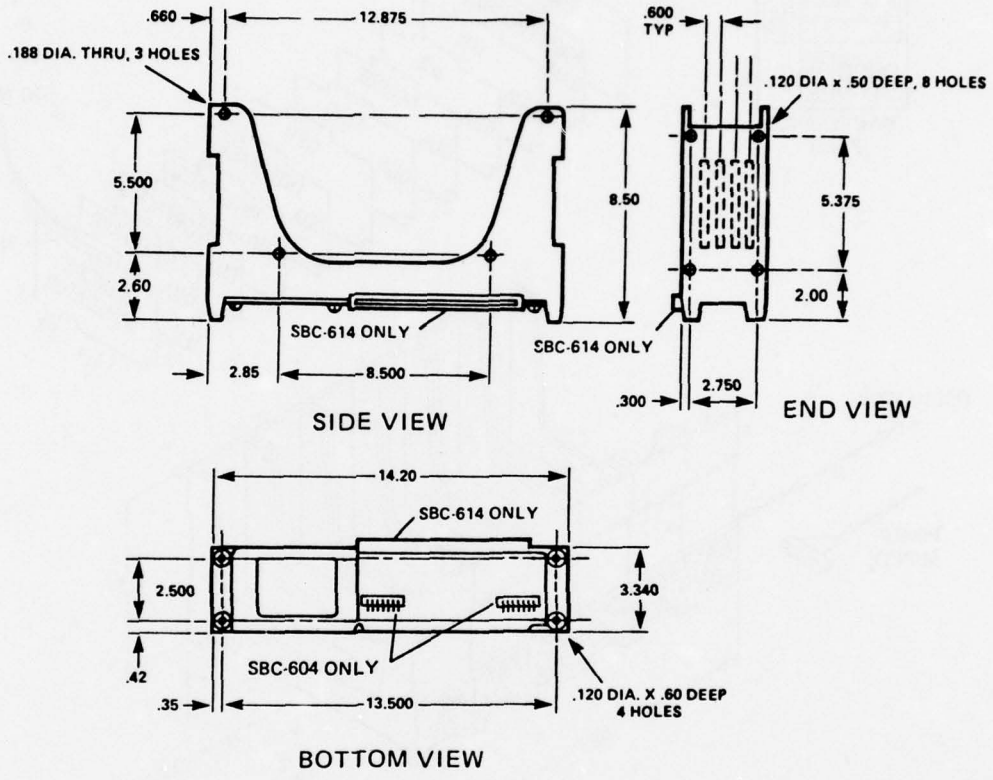


FIGURE 4.4-2 SBC 604/614 DIMENSIONS

All logic is implemented on Intel Universal Wire-wrap Modules; part number MDS-600 (refer to Figure 4.4-3). The module is designed to plug directly into three standard, double-sided PC edge connectors. An 86-pin connector and a 60-pin auxiliary connector are located on one edge of the board; a 100-pin connector is on the opposite edge. The connectors can serve as a mounting, as well as an electrical junction, if the environment is not too severe.

The interface modules communicate with the MDS-800 mainframe and the maintenance panel via a 100-pin, double-sided PC edge connector (J1) which attaches to the edge opposite that of the other two connectors. This connector has 0.1-inch contact centers. Viking 3VH50/1JN5 is one suitable type of connector.

Opposite the J1 connector lie two other connectors. One is a standard 86-pin, double-sided PC edge connector (P1), 0.156-in. contact centers. Control Data VPB01E43A00A1 is one suitable type of connector. The module can also communicate with other modules through the auxiliary 60-pin, double-sided PC edge connector (P2), with 0.1-in. contact centers.

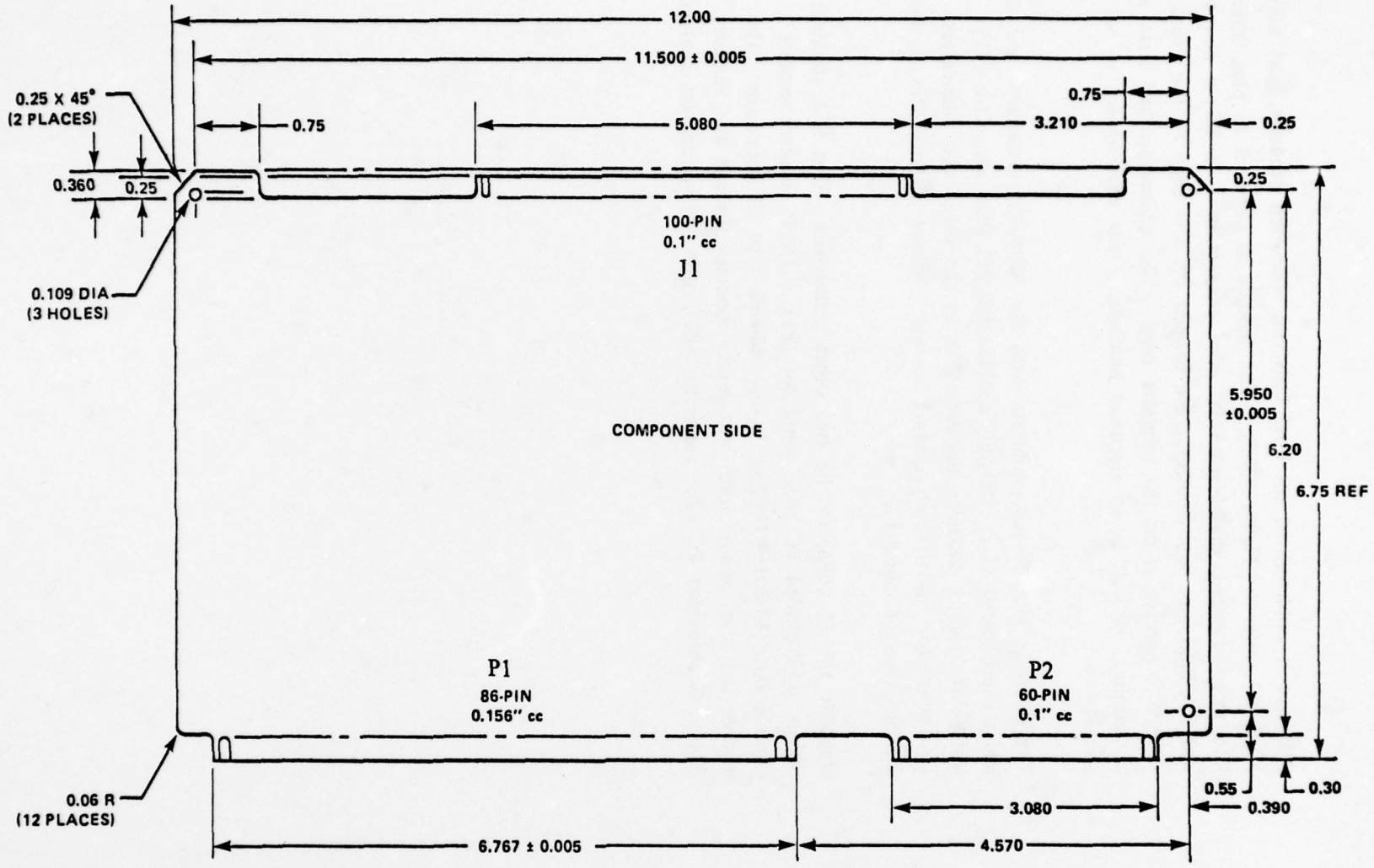


Figure 4.4-3 Module Dimensions

☆U.S.Government Printing Office: 1978 — 657-002/241