This manual contains the maintenance-oriented and recall diagrams referenced in the companion 7201-02 Computing Element FETOM (Form SFN-0201) and in the 7201-02 Computing Element FEMM (Form SFN-0203).

The diagrams in this manual are arranged into six categories:

Category 1. Diagnostic Techniques
Category 2. Overall Data Flow
Category 3. Data Flow by Instruction Class
Category 4. Functional Units
Category 5. Operations
Category 6. Manual Controls and Maintenance Facilities

All diagrams are in numerical order. The first digit of the diagram number reflects the category; for example, Diagram 4-210 belongs to Category 4, Functional Units. A category may be further subdivided into functional groups; for example, in Category 4, the diagrams have been grouped as follows:

Group 1. Timing and Clock Control
Group 2. ROS
Group 3. Data and Control Registers
Group 4. Local Storage
Group 5. Serial and Parallel Adders
Group 6. Status and Control Triggers
Group 7. SCI

Prerequisite and companion manuals are:

**Prerequisite Manuals**

- 9020E System Introduction, Theory of Operation Manual, Form SFN-0103
- 9020D System Introduction, Theory of Operation Manual, Form SFN-0104

**Companion Manuals**

- 7201-02 Computing Element, Theory of Operation Manual, Form SFN-0201
- 7201-02 Computing Element, Maintenance Manual, Form SFN-0203
- 7201-02 Computing Element, Installation Manual, Form SFN-0204
- 7201-02 Parts Catalog, Form SFN-0205

First Edition (July, 1970)

This manual has been prepared by IBM Product Publications, Kingston, N.Y.

©Copyright International Business Machines, 1970
ABBREVIATIONS

ABC A,B,C.
AC automatic carrier return
ADC automatic data converter
ADL alternate data logic
ALT alternate
AM alternate memory
APSP alternate pre-stored program
ASC assign
ATC alternate traffic content
ATN alternate test number
ATR address translation register
ATT attraction
Aux auxiliary
BCD binary-coded decimal
BCU bus control unit
BL black
BR brightness
BSM basic storage module
C capacitor
CAS control automation system
CAW command word
CB circuit breaker
CC condition code, also Configuration Console
CCC Central Computer Complex
CCR configuration control register
CCW channel command word
CE control element
Chd optional field
CLD control automation system logic diagram
CPU central processing unit (alternate terminology for CE)
CROS capacitive read-only storage
CSW channel status word
CT conditional
CCT channel-to-channel
CU control unit
CVG character vector generator

STARTER SELECTION Switch Gating........... 6-10
DEFEAT INTERLAVING Switch Gating........... 6-11
RATE Switch Logic......................... 6-12
Instruction Step Routine...................... 6-13
Single-Cycle and Single-Cycle-Inhibit Routine........ 6-14
Repeat Instruction Switch Logic.................... 6-15
Repeat Instruction Switch Routine.................... 6-16
ROS TRANSFER and REPEAT ROS ADDRESS Switch Gating........ 6-17
Storage Ripple Loop (Store and Display) Routine........ 6-18
Wait State Gating.......................... 6-19
Wait State Microprogram Routine.................... 6-20
Disable Interval Timer Logic...................... 2-61
CE Check Control and Initialize CE Handshakes, Logic and Error Controls........ 6-22
Pulse Mode Controls.......................... 6-23
Pulse Mode Operation.......................... 6-24
LOG OUT Pulseout Logic......................... 6-25
SCAN MODE, ROS/PROC/FLT Switch Logic.............. 6-27
1052 Adapter Unit............................ 6-28
1052 Adapter Initial Selection............ 6-29
1052 Adapter Data Transfer.............. 6-30
1052 Adapter Data Transfer............. 6-31
1052 Adapter Ending Sequence........... 6-32
1052 Adapter Sense and Status Bytes........ 6-33

Group 2: Maintenance Features

Scan Data and Control.......................... 6-50
Scan Clock.................................. 6-102
FLT Clock.................................. 6-103
Scan Coupler Latches and Decrementor........ 6-104
Scan Store Address Generator............ 6-105
FLT Coupler Decrementor.................... 6-106
Scan-Out Bus Data Flow..................... 6-107
Logoff Control Logic......................... 6-108
Scan-Out Path For One Bit.............. 6-109
Maintenance Mode Stop Clock Logic............. 6-110
Scan Control Trigger......................... 6-111
Scan Control of ROS Microbranching........ 6-112
CE Scan/DOE Interface......................... 6-113
Logoff Sequence (2 Sheets).................. 6-114
ROS Test Sequence (2 Sheets)............. 6-115
FLT Sequence (2 Sheets)..................... 6-116
CE Logframe Formats (3 Sheets)............ 6-117
SE Logframe Formats.......................... 6-118
DE Logframe Formats.......................... 6-119

Group 3: DE Wrap Operation

DE Wrap Bus Operation.......................... 6-201

INDEX

*Note: 1052 Adapter is used only with the 9020E configuration.
Diagram 1-2. ROS Test Flowchart (Sheet 1 of 3)
Diagram 1-2. ROS Test Flowchart (Sheet 2 of 3)
Possible open sense line. More information in FEMM, 4.3.6.2.

Measure continuity from sense amp input pins to sense line of failing bit.

Possible open drive line. More information in FEMM, 4.3.6.2.

Turn CE power off.

Remove sense amp for failing bit and adjacent bits.

Turn CE power off.

Possible open drive line. More information in FEMM, 4.3.6.2.

Measure resistance from output pin to adjacent bit pins.

Measure resistance from output pin to sense line.

Open in yellow wire or joint.

Measure resistance between input pins of failing bit.

Turn power on and measure voltage across each sense line resistor.

0.8V dc indicates drive line shorted to ground.

Open in yellow wire or joint.

Remove, clean and reinstall bit plane.

Rerun ROS tests.

Possible open drive line. More information in FEMM, 4.3.6.2.

Isolate and repair.

Rerun ROS tests.

Possible open drive line. More information in FEMM, 4.3.6.2.

Loosen torque on pressure plates over sense line one by one until voltage drops to 0V dc.

Loosen torque on pressure plates over sense line one by one until voltage drops to 0V dc.

Loosen torque on pressure plates over sense line one by one until voltage drops to 0V dc.

Replace faulty bit plane.

Rerun ROS tests.

Resistance Readings:

<table>
<thead>
<tr>
<th>Condition</th>
<th>From Sense Amp Input to</th>
<th>Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Any other sense line</td>
<td>16.5 ohms</td>
</tr>
<tr>
<td></td>
<td>Any other sense line</td>
<td>200 ohms</td>
</tr>
<tr>
<td>Sense - sense line short</td>
<td>Any other sense line</td>
<td>8.25 ohms</td>
</tr>
<tr>
<td></td>
<td>Any other sense line</td>
<td>24.75 ohms</td>
</tr>
<tr>
<td>Sense - DC return short</td>
<td>Any other sense line</td>
<td>0 ohm</td>
</tr>
<tr>
<td></td>
<td>Any other sense line</td>
<td>0 ohm</td>
</tr>
<tr>
<td>Sense - drive/balance line short</td>
<td>Any other sense line</td>
<td>16.5 ohms</td>
</tr>
<tr>
<td></td>
<td>Any other sense line</td>
<td>33.0 ohms</td>
</tr>
<tr>
<td>Sense - drive/balance line short</td>
<td>Any other sense line</td>
<td>0 ohm</td>
</tr>
</tbody>
</table>

Diagram 1-2. ROS Test Flowchart (Sheet 3 of 3)
Procedure on logic page A6503.

1st depression of LOAD PB.

Check roller 6, position 4 for type of error.

Run ripple tests (FEMM, Ch 4, 4.1) and repair.

Refer to procedure in FEMM, Ch 4, 4.4.1, failure 5.

Diagram 1-3. FLT Flowchart

3rd depression of LOAD PB.

Correct indications after Backspace Key.

Repair using procedure in Ch. 4 of FEMM, 4.4.1, failure 4.

Correct switch setting and load again.

Correct wrong condition and try again.

Check IOCE and TAU for channel or tape error.

Investigate Storage error.

Do next step on maintenance strategy diagram.

See One-Cycle repair procedure in FEMM, Ch 4, 4.4.4.

See Zero-Cycle repair procedure in FEMM, Ch 4, 4.4.3.

Load FLT.
Diagram 3-1. Fixed Point Instruction Data Flow
Diagram 3-2. Floating-Point Instruction Data Flow
Diagram 3-3. Decimal and Logical Instruction Data Flow
Diagram 3-4. Branching Instruction Data Flow
Diagram 3-5. Status Switching Instruction Data Flow
Diagram 3-6. Input/Output Instruction Data Flow
Diagram 3-7. Multiple Computing Element Instruction Data Flow
Diagram 4-1. Clock Control Logic

**Notes:**
1. Assume, for this diagram, no delay in the logic's block except for 50 ns in inverter.
2. Heavy portion of timing signals indicates the active portion of the signal function.
3. Activated by switch or internal triggers, ALO KW081.
4. To SCI request sensing logic (Diagram 4-601) and SCI control logic for CE clock (Diagram 4-602).
5. Inverter block is part of logic block.
6. Adjust time delay to provide 80-ns clock, 120-ns not-clock to logic gates.
Diagram 4-2, Reference Oscillator

Notes:
1. Heavy portion of timing signals indicates the active portion for the signal function.
2. The two letter notation within the AND's is the block serial number on ALD KC021.
3. Inverter block is part of logic block.
Not Inhibit OSC, On Wrap, Or Logout

+3V dc

Not Wrap Mode Inhibit Osc Set Outstanding

Hardstop Inhibit OSC

5.0-mHz OSCILLATOR

Adjust for symmetrical 100-ns/100-ns clock signal.

Adjust for 200-ns period clock signal (Null Comparator)

Symmetry Observation Point

Gated Oscillator

(Basic Clock Pulse)

5.0-mHz OSCILLATOR

Adjust for symmetrical 97.5-ns/97.5-ns clock signal.

Adjust for 195-ns period clock signal.

Symmetry Observation Point

5.0-mHz REFERENCE

(From crystal-controlled; same as shown in Diagram 4-2.)

5.0-mHz Comparator Circuit

Machine reset

To Diagram 4-2

If line is positive, oscillator output is positive and not running. When line goes negative, oscillator output goes negative and starts stable oscillation.

Diagram 4-3. CE Clock Signal Generator
Diagram 4-105. ROS Addressing and Data Flow (Sheet 1 of 2)
Diagram 4-105. ROS Addressing and Data Flow (Sheet 2 of 2)
Diagram 4-106. Array Drivers
Diagram 4-107. ROS Data Register
Diagram 4-201. Q-Register B-Field Transfer Controls

Diagram 4-202. R-Register Transfer to LAL
Diagram 4-203. E-Register Incrementer, Bits 14 and 15
Add I to E(12-15) Not E(B-15) Bits Even

Not E(l5)

Channel P 8-11

Not E(l3)

~E~l_S~>~~~~~-+-~

A

E 14

Subtract 1 from E(12-15)

Not E(l5)

CE171

Not Change P(B-11

E(B-15) Bits Even

DIagram 4-204. E-Register Parity Prediction after Incrementing

Diagram 4-205. Parity Adjustment for IC (21, 22) Stepping

LS 51 15

OR

LS Bus 15

Gates PSW(10-15) to 5(10-15)

Gate LS Bus (10-15 to 15-10)

Set IC(21, 22)

Gate IC(16-23) to 5(16-23)

Gate LS Bus (16-23 to 5(16-23)

Gate SDB0(16-23 to 5(16-23)

Gate SADDL(0-7) to 5(16-23)

Data Keys to 5(16-23)

Diagram 4-206. S-Register, Bits 15 and 16
Diagram 4-207. AB Byte Counter
Diagram 4-208. ST Byte Counter
Diagram 4-209. Mark Trigger Logic
Diagram 4-210. CCR Output Logic and Control Paths (Sheet 1 of 3)
Diagram 2-3: CCR Bus 8-23
1. Assume an RPSB instruction and an FMTN micro-order and assume bit 8 is to be traced. Refer to the RPSB instruction chart, FMTN portion, specific micro-order. If the specific micro-order is an FMTN-7, bit 8 follows the line up to the micro-order being issued. For example, XY register. Find bit B in the XY register and follow the line up to the instruction, RPSB or CWJL; (2) if RPSB is selected, refer to FMTO.

To trace the origin of a mixer output bit: (1) select chart for specific input to Mixer for FMTW-0.

Note: L register does not supply input to XY register.
Diagram 4-211. LM to XY Reformatting via Mixer (Sheet 2 of 2)
Diagram 4-212. XY Register Parity Prediction Logic
Diagram 4-213. Select Register - Select Signal Generation and Response Reset.
Diagram 4-301. Local Storage Read/Write Controls
Diagram 4-302. 9030 Out Bus to LS Data Bus Gating Logic

- CCR 0
  - Connect CCR to LS Bus (0-6)
  - ATR-1: 3
  - Connect ATR-1 to LS Bus (0-6)
  - ATR-2: 0
  - Connect ATR-2 to LS Bus
  - CHECK REG: 5
  - Connect CHECK REG to LS BUS (0-7)
- For Bus: 0
  - Connect For Bus to LS Bus (0-7)
- For Reg: 0
  - Connect For Reg to LS Bus (00-07)
- Select Reg: 0
  - Connect Select Reg to LS Bus (0-7)
- Data 0
  - Connect Data to LS Bus (0-7)
- Gate Data Mask to LS Bus (0:..7)
- CCR 31
  - Connect CCR to LS Bus (24-31)
- ATR-1 31
  - Connect ATR-1 to LS Bus (24-31)
- ATR-2 31
  - Connect ATR-2 to LS Bus
- PSBAR 31
  - Connect PSBAR to LS Bus
- G Reg Bit 7
  - Gates G Reg to LS Bus (24-31)
- Ext Bus 31
  - Connect Ext Bus to LS Bus (24-31)
- Ext Reg 31
  - Connect Ext Reg to LS Bus (24-31)
- Select Reg 31
  - Connect Select Reg to LS Bus (24-31)
- DAR Mask 3°1
  - Connect DAR Mask to LS Bus (24-31)

Note: LS registers 1-24 are similar to register 0.
Diagram 4-303. LS Bus Parity Generation or Check
Diagram 4-401. Serial Adder Input Bus Logic.
Diagram 4-402. Carry Lookahead Logic, SAL(0-3)

Diagram 4-403. Decimal Add 6 Logic
Diagram 4-404. Decimal Correction Logic For SAL (0-3)

Diagram 4-405. Invalid Digit Logic

4-404, 405 (7/70)
Diagram 4-406. Logical Functions, SAL (0)

Diagram 4-407. Serial Adder Parity Predict Logic

Note: Shaded area indicates additional controls for decimal operations only.
Diagram 4-408. Serial Adder Product-Quotient Bit Logic

- **A.** T-Field Gate Control Trigger:
  - 'T' Micro-order (1011)
  - Dec-order T; 111
  - Gate B(32-63) to Parallel Adder B-side (32-63).

- **B.** U-Field Gate Control Trigger:
  - '+T' Micro-order (1110)
  - Dec-order U; 133
  - Gate T(32-63) to Parallel Adder A-side (32-63).

Diagram 4-409. Gate Control Triggers for 'B + T' Micro-order
Diagram 4-10. Parallel Adder Bit-Position Logic (Bit 47)

A: Section 4
B: Section 3
C: Section 2
D: Section 1

- A 'not group 4 carry' signal indicates that no actual carry from group 4 (positions 48, 49, 50, and 51) has occurred, thus allowing a predicted carry (carry-lookahead) to enter bit position 47.
- The absence of a 'not group 4 carry' signal (indication an actual carry) inhibits predicted carries from entering bit position 47.

E: ROS Micro-Order.
Diagram 4-411. Parallel Adder Carry Lookahead Logic
duplicate logic is used for generating parity-predict signals of opposite polarity simultaneously (without use of additional inversion function).

Diagram 4-412. Parity Generation, PAL (48-55)
Diagram 4-413. Parallel Adder Half-Sum Checking Logic, PA (48-55)
Diagram 4-414. Parallel Adder Full-Sum Checking Logic, PA(48-55)
Diagram 4-415. Parallel Adder Excess 6 Logic

Diagram 4-416. Parallel Adder Set-Condition-Code Logic

Note: This diagram illustrates how the condition codes are set; it is not intended to be a detailed diagram.
Diagram 4-501. STAT B Logic
Diagram 4-603. SCI Control Logic for CE Clock
Diagram 4-604. Invalid Address and Frame Stopped Logic (Sheet 1 of 2)
Diagram 4-604. Invalid Address and Frame Stopped Logic (Sheet 2 of 2)

---

Diagram 4-604. Invalid Address and Frame Stopped Logic (Sheet 2 of 2)
Notes:
1. "Storage timeout" is activated if "select outstanding" remains active within two 60-cycle pulses.
2. "Select outstanding" may occur at any time. The example shown is for the least amount of time necessary to activate storage.
3. "Select outstanding" is deactivated by "accept" from storage.

Diagram 4-605. Storage Timeout Logic
Diagram 4-606. Error Handling Logic
Diagram 4-608. PSBAR Operations
Diagram 4-609. Page Control Logic and Timing
Diagram 4-610. SAB Parity Conversion Logic
Diagram 4-611. Detailed SCI Functional Sequence (Sheet 1 of 2)
Diagram 4-611. Servicing of Storage Requests in Single-Cycle Mode (Sheet 2 of 2)
Diagram 4-612. Servicing of Storage Requests in Single-Cycle Mode

Note:
Storage requests issued by CE 1-Fetch hardware are services in the same manner as microprogram requests; the CPU's clock must be engaged automatically to permit starting control of 1-Fetch sequence.
Diagram 5-1. Operand Prefetching During End Op
Diagram 5-2. Instruction Requests During End Op

Diagram 5-3. Instruction Requests During Early End Op
Diagram 5-4. Branch Requests
Diagram 5-5. Selection of I-Fetch Sequence

Effective R decoding is done from SORO if BEOP

B Field = 0

Bits 0-5 are specified by End-Op Word as 001000
Transfer 1st operand from T to A, B, and D.

Branch on Condition

Request Instructions per D (3-cycle).

Address LS per R2 to obtain 2nd operand.

Transfer 2nd operand from T to A, B, and D.

Transfer 1st operand from LS to S and T.

Branch to 1st ROS word of execution sequence per 'E(02-07)-ROA' micro-order.

Set STC to 100 and ABC to 000.

Transfer 2nd operand from LS to S and T.

Request Instructions per IC (3-cycle).

Dependent on IC at end see table on this page.

Increment IC(21, 22) by 1, and transfer 1st halfword of next instruction to R.

Set STC to 100 and ABC to 000.

Refer to Diagram 5-30.

Exceptional Condition Cycle

Branch to 1st ROS word of execution sequence per 'E(02-07)-ROA' micro-order.

Covers SEBO to be gated to Q during 2nd execution cycle.

Issue 'RASCR' micro-order.

Refer to Diagram 5-6.

RR I-Fetch Variables

IC Setting on End Op

Incremented IC

Next Op-Code Word Transformed to R Form

<table>
<thead>
<tr>
<th>IC Setting</th>
<th>Bit 21</th>
<th>Bit 22</th>
<th>Incremented IC</th>
<th>Bit 21</th>
<th>Bit 22</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0</td>
<td>0 1</td>
<td>00</td>
<td>0 1</td>
<td>0 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 0</td>
<td>1 1</td>
<td>01</td>
<td>1 0</td>
<td>1 1</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 1</td>
<td>0 0</td>
<td>10</td>
<td>1 1</td>
<td>0 0</td>
</tr>
</tbody>
</table>

Note: These actions are inhibited when 'execute in progress' trigger is set.
Diagram 5-7. One-Cycle RR I-Fetch
Diagram 5-8. Two-Cycle RR I-Fetch
Diagram 5-39

RX Fetch

1st I-Fetch Cycle

Diagram 5-9

Problem

1. Transfer 1st operand to S and T.
2. Create address of 2nd operand and transfer to D.
3. Request 2nd operand from main storage (see Note 2).
4. Initial condition (RX is not indexed).
5. If I-Fetch of Instruction is transferred to 0, 0-indexed to Q.
6. Contents of LS register specified by 62 are transferred to T.

Notes:

1. RX Format
   - Op Code
   - R1
   - X2
   - 82
   - D2
   - 7
   - 8
   - 11
   - 12

2. Purpose:
   - 1. Transfer 1st operand to S and T.
   - 2. Compute address of 2nd operand and transfer to D.
   - 3. Request 2nd operand from main storage (see Note 2).
   - 4. Initial condition (RX is not indexed).
   - 5. If I-Fetch of Instruction is transferred to 0, 0-indexed to Q.
   - 6. Contents of LS register specified by 62 are transferred to T.

3. Initial conditions if I-Fetch is not blocked:
   - 1. 1st halfword of instruction is transferred to E; 2nd halfword is in 0.
   - 2. Contents of LS register specified by 82 are transferred to T.

4. Actions referred to:
   - 1. Index Cycle
   - 2. Increment IC(21, 22) by 2 during previous cycle
   - 3. Increment IC(20) by 1 during previous cycle
   - 4. Transfer D to PAL for SPEC test at next cycle

5. Branch to 1st ROS word of execution sequence per 'E9-07-ROA' micro-order.

6. 'RESET' micro-order:
   - a. Inhibits setting of 'D sync' latch if LA, STC, STH, or unsuccessful BC.
   - b. Sets 'IC sync' latch if unsuccessful BC.
   - c. Resets STAT's, Edit controls, STC, and ABC.
   - d. Sets LAR(O) if floating-point instruction.

7. If I-Fetch sequencers 1 and 2 cycle 9090 in Q during 3rd execution cycle, and transfer Q(62) to S during previous cycle, and transfer Q(1-15) to R during previous cycle.
Diagram 5-10. One-Cycle RX, RS, and SI I-Fetch
Diagram 5-11. Two-Cycle Indexed RX I-Fetch
Diagram 5-12. Two-Cycle Non-Indexed RX, RS, and SI I-Fetch
Diagram 5-30

Test for "Register refill needed" condition.

Yes: 2-field transfer (Q to D).

No: Add 2 fields to Q and transfer (Q to D).

Transfer Q plus 2 fields to R.

Conditional; see Note 2.

QJ001 020 Increment IC(20) by 1 (done by I-Fetch sequencers).

QJ001 024 IC(20) already is incremented.

Issue 3-cycle storage request per D.

QJ001 034 Transfer D to PAL for SPEC test during next cycle.

Conditional; see Note 4.

RS Format:

Op Code: \[ 00 \ 01 \ 02 \ 03 \ \ldots \ 12 \ \ldots \ 25 \]

SI Format:

Op Code: \[ 13 \ 14 \ 15 \ 16 \ \ldots \ 25 \]

Notes:
1. 'RESET' micro-order:
   a. Inhibits setting of 'D sync' trigger if fetching MVI, STM, TS, shift, or 1/0 instruction.
   b. Resets '3-cycle request' trigger (causing 4-cycle request) if fetching BXH or BXLE instruction.
   c. Resets STA T's, edit controls, STC, and ABC.

2. I-Fetch sequencers 1 and 3 gate SDBO to Q during 2nd execution cycle, one transfer Q(0-15) to R if IC(21,22) = 0 during previous end op.

3. Incrementing of IC, transferring of Q to R, and setting of I-Fetch sequencers are inhibited if 'execute in progress' trigger is set.

4. 'RETF' micro-order:
   a. Inhibits setting of 'O sync' trigger if fetching add, STA, STA, SUB, SUB, or 1/0 instruction.
   b. Resets '3-cycle request' trigger issuing 4-cycle request if fetching ARM or ARM0 instruction.
   c. Reset 'active flag' bit for CIC, CRS, and ABC.

5. I-Fetch sequencers 1 and 2 gate SDBO to Q during 2nd execution cycle, and handler QJ012 to R if IC(21,22) = 0 during previous end op.

Diagram 5-13. RS and SI I-Fetch

5-13 (7/70)
Diagram 5-14. SS I-Fetch (Sheet 1 of 2)
Diagram 5-15. I-Fetch Sequences (Sheet 1 of 2)

A. Basic Control Provided by I-Fetch Sequencers
### Typical Micro-Orders

<table>
<thead>
<tr>
<th>Action</th>
<th>ALD</th>
<th>n-2</th>
<th>n-1</th>
<th>n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set IF1 Trigger</td>
<td>KD101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF Sequencer 1</td>
<td>KD101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF Sequencer 2</td>
<td>KD111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF Sequencer 3</td>
<td>KD111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IC + B → PAL</td>
<td>KD201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAL → IC</td>
<td>D001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS → Q</td>
<td>D003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Notes:

1. X = Z ≠ IF2, where Z is defined in Note 3.
2. Also, gate Q(0-15) to R(0-15) if IC(21, 22) = 01 (ALD D020).
3. Z = ![Not (Predecode branch + Predecode SS + Execute)](021, 22 = 01 + Predecode not 83).

These actions are inhibited by "block IF-fetch" trigger.

---

**Diagram 5-15. I-Fetch Sequencers (Sheet 2 of 2)**

---

**Diagram 5-15. I-Fetch Sequencers (Sheet 2 of 2)**
Diagram 5-17. Timer Exceptional Condition
Diagram 5-18. CPU Store In Progress Exceptional Condition

1. Inhibit Next ROS Address
2. Force ROSAR to 02E (hex)

Establish priority for CPU store in progress exceptional condition.

Set 'block I-Fetch' trigger.

Force address 02E (hex) to ROSAR.

CPU Store In Progress

Normal end op.
Diagram 5-19. Machine Check Interruption
Diagram 5-21. SPEC Y-Branch Setting of Interrupt Code Triggers
Diagram 5-22. Program Interruption
Supervisor Call Instruction

Interrupt Code 4 Trigger

1. INTREQ-TGR (F20)
2. INT-TGR (F25)

Transfer E(B-15) to S(Qn-31)

Force ROSAR to 008 (Hex)

Set 'block I-Fetch' trigger.

Diagram 5-26

1. Force address 008 (Hex) into ROSAR.
2. Restart ROS clock if stopped (Hold state).
3. Force D to 20 (hex)
4. To 'block I-Fetch' trigger (Diagram 5-16).

Some as shown in Diagram 5-22 with all
"interrupt code" triggers reset. SVC
cannot cause program interruption.

RS241-RS311

Diagram 5-23. Supervisor Call Interruption
Diagram 5-24. External Interruption

Issued by Timer 1 -TCL-TGR (FJ 7 )

Exceptional Condition

Microprogram (Diagram 5-17)

Interrupt Pushbutton Latch

Hold DAR Latch

Hold PIR Latch

CE RDD or WDD Signal In

(Stop 7-4)

Direct Control

Control

Console Signal

Set 'console' and 'external signal' latches.

Transfer 'time clock at limit', 'console signal', and external signals 2-7 triggers to S.

End-Op Cycle

Performed by "O-STAD" micro-order.

Causes common interruption routine to recognize non-machine-check entry.

Transfer all 12 triggers to S(20-31)

Diagram 5-26

Common interruption routines.

Set STAT H.

Transfer all 12 triggers to S(20-31)

Common interruption routine.

1. Block Lower Priority Interruption and Exceptional Conditions

2. Restart CPU Clock if Stepped (Wait State)

Sample External Interruption Triggers

Set RDD or WDD hold latch.

Set 'console' and 'external signal' latches. Transfer 'time clock at limit', 'console signal', and external signals 2-7 triggers to S.

Establish priority of external interruptions.

Keep external interruption pending.

Watches next E-Fetch section.

End-Op Cycle

Perform D to 18 (Hex). To Block I-Fetch Trigger (Diagram 5-16)

PSW-St(CT)

Set DAR Hold Latch

Set RDD or WDD hold latch.

Set 'console' and 'external signal' latches.

Transfer 'time clock at limit', 'console signal', and external signals 2-7 triggers to S.

Diagram 5-26

Common interruption routines.

Set RDD or WDD hold latch.

Set 'console' and 'external signal' latches.

Transfer 'time clock at limit', 'console signal', and external signals 2-7 triggers to S.

Diagram 5-26

Common interruption routines.

Set RDD or WDD hold latch.

Set 'console' and 'external signal' latches.

Transfer 'time clock at limit', 'console signal', and external signals 2-7 triggers to S.

Diagram 5-26

Common interruption routines.

Set RDD or WDD hold latch.

Set 'console' and 'external signal' latches.

Transfer 'time clock at limit', 'console signal', and external signals 2-7 triggers to S.

Diagram 5-26

Common interruption routines.
Diagram 5-25. I/O Interruption (Sheet 2 of 2)
Diagram 5-26. Common Interruption Routine

- Conditions at start of this routine:
  1. Address of old PSW is in D.
  2. Interruption code is in S(16-31).
  3. STAT H is not set if a machine check
     Interruption is in progress.

A

Complete assembly of old PSW in ST.

B

IC[21,22] = 11

Transfer IC minus 8 to T(40-63) via parallel adder.

Transfer IC minus 16 to T(40-63) via parallel adder.

C

OCIE

IOCE

Transfer PSW register to T(10-15) and T(32-39)

D

Store old PSW.

Issue 4-cycle storage repeat per D.

Inhibit storage protection.

Set mark 0-7 triggers.

E

Issue resets.

F

Reset triggers that caused this interruption.

STAT H

H

G

Wait 3 cycles.

Reset system.

Rest error triggers.

Does not reset MCH Interrrupt Tgr.

Diagram 5-25

I/O Interruption

Diagram 5-23

2-cycle storage request per D.

Inhibit storage protection.

Diagram 5-26

Load PSW.

B50

350

351

5-19 Machine check
5-22 Program
5-23 Supervisor Call
5-24 External

Qu001

Complete assembly of old PSW in ST.

Store old PSW.

Issue resets.

Transfer IC minus 8 to T(40-63) via parallel adder.

Transfer IC minus 16 to T(40-63) via parallel adder.

Calculate address of new PSW.

Set 8(59) via parallel adder (hex 10).

Shift 8-registers left twice (hex 40).

Add 8 to D via parallel adder.

Request new PSW.

Issue 3-cycle storage request per D.

Diagram 5-26

Wait 3 cycles.

Reset system.

Rest error triggers.

Does not reset MCH Interrrupt Tgr.
Diagram 5-27. Manual Control Exceptional Conditions

Notes:
1. Return to Wait state after timer exceptional condition.
2. If interruption occurred, PSW(14) determines Wait or Running state.
3. If stop is pending, stop loop routine is entered.
Diagram 5-28. Program Store Compare Exceptional Condition

Diagram 5-26

Diagram 5-506

Diagram 5-601

1. Block Invalid Instruction Address Test and Q-Register refill Exceptional Conditions
2. Block setting of ROSAR<81

Transfer IC minus 16 to IC and T.

Force ROSAR to 004 (Hex).

Load PSW.

Set 'block I-Fetch' trigger.

Blocks most I-Fetch actions.

Yes

No

Yes

No

Check for Software interrupt

(Not) (TCS or lrpt)

(Not) Repeat Inst Priority

(Not) Wait Priority

OR

OR

Yes

Set 'block I-Fetch' trigger.

Blocks most I-Fetch actions.

Force ROSAR to 004 (Hex).

Transfer IC minus 8 to IC and T.

Issue 3-cycle storage request per IC.
| Diagram 5-29. Invalid Instruction Address Test Exceptional Condition (Sheet 2 of 2) |

---

**Diagram 5-29**

1. **Invalid Transaction**
   - Exceptional Condition pending
   - No Interruption or Higher Priority
   - Yes: IC(R) = 1
   - No: Continue

2. **Check Instruction Address**
   - If valid, go to **Diagram 5-30**
   - If invalid, go to **Diagram 5-22**

3. **Check Program Interruption**
   - Yes: Set 'interrupt code -2' and '-4' triggers
   - No: Perform 3-cycle storage request per D

4. **Set Branch Invalid Address Trigger**
   - Yes: Set 'branch invalid addr' and 'prot branch addr' triggers
   - No: Set 'interrupt code 4' trigger

5. **Block Setting of Interrupt Code Triggers**
   - Yes: Set 'interrupt code -1' and '-4' triggers
   - No: Set 'interrupt code 0' and '-4' triggers

---

**Diagram 5-30**

- Test for Queue pointer and exceptional condition
- Yes: Force ROSAR to 002 (hex)
- No: Perform only 1st I-Fetch cycle per format

---

**Diagram 5-22**

- Set 'interrupt code 0' and '-4' triggers
- Set 'interrupt code -1' and '-4' triggers
- Set 'interrupt code 4' trigger
- Set 'interrupt code -2' and '-4' triggers
- Block setting of interrupt code triggers by all micro-orders

---

**Diagram 5-29**

- Diagram 5-29: Invalid Instruction Address Test Exceptional Condition (Sheet 2 of 2)
A. Load, LR (18)

- **Purpose**: Load 2nd operand into 1st operand location (in GPR, per R2).

- **RR format**.

- **Conditions at start of execution**:
  1. 1st 16 bits of instruction are in E.
  2. 1st operand is in S and T (not used).
  3. 2nd operand address is in D.
  4. Main storage request for 2nd operand has been issued per D.

B. Load, L (58)

- **Purpose**: Load halfword 2nd operand (in storage) into 1st operand location (in GPR, per RI).

- **RX format**.

- **Conditions at start of execution**:
  1. 1st 16 bits of instruction are in E.
  2. 1st operand is in S and T (not used).
  3. 2nd operand address is in D.
  4. Main storage request for 2nd operand has been issued per D.

---

**Diagram 5-101. Load, LR (18); Load, L (58)**

**Diagram 5-102. Load Halfword, LH (48)**

---
Diagram 5-103. Load and Test, LTR (12)

- RR format:
  \[
  \begin{array}{ccc}
  R1 & R2 & R3 \\
  12 & 2 & 0 \\
  \end{array}
  \]

- Purpose: Load 2nd operand (in GPR, per R2) into 1st operand location (in GPR, per R1) and set CC according to result.

- Conditions at start of execution:
  1. Instruction is in E.
  2. 1st operand is in A, B, and D (not used).
  3. 2nd operand is in S and T.

- RR format:
  \[
  \begin{array}{ccc}
  R1 & R2 & R3 \\
  13 & 2 & 0 \\
  \end{array}
  \]

- Purpose: Load 2's complement of 2nd operand (in GPR, per R2) into 1st operand location (in GPR, per R1) and set CC according to result.

- Conditions at start of execution:
  1. Instruction is in E.
  2. 1st operand is in A, B, and D (not used).
  3. 2nd operand is in S and T.

Diagram 5-104. Load Complement, LCR (13)
Diagram 5-6
I-Fetch.

5-105
Load Positive, LPR (10)

RR format:

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose:** Load 2nd operand (unchanged if positive, 2's complemented if negative; in GPR, per R2) into 1st operand location (in GPR, per R1).
- **Conditions at start of execution:**
  1. Instruction is in E.
  2. 1st operand is in A, B, and D (not used).
  3. 2nd operand is in S and T.

- **A**
  - Transfer 2nd operand from T to PAA (32-63).
  - Set STAT A if PAL(32-63) = 0.
  - Transfer 2nd operand from T to GPR per (E9-11).

- **B**
  - Yes
    - T(32) = 1
  - No

- **C**
  - Transfer 2's complement of 2nd operand to PAA(32-63).
  - If overflow, set STAT B.
  - Transfer PAL(32-63) to T. Transfer T to GPR per (B-11).

- **D**
  - Yes
    - STAT B set
  - No

- **E**
  - Yes
    - STAT A set
  - No

  - Set CC to 2.
  - Set CC to 0.
  - If T(32) = 0, set CC to 2.
  - End op.
Set CC to 0.

Diagram 5-6

1. RR 1-Fetch.

Q6001

Transfer 2nd operand from T to PA (32-63).

Set STAT A if PA(32-63) = 0.

Transfer 2nd operand from T to GPR per E(8-11).

Diagram 5-106. Load Negative, LNR (11)

• RR format:

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

• Purpose: Load 2nd operand (unchanged if negative, 2's complemented if positive, in GPR per R2) into 1st operand location (in GPR, per R1).

• Conditions at start of execution:

1. Instruction is in E.
2. 1st operand is in A, B, and D (not used).
3. 2nd operand is in S and T.

4. Transfer 2's complement of 2nd operand to PAA(32-63).

Transfer PA(32-63) to T. Transfer T to GPR per E(8-11).

If T(32) = 1, set CC to 1.

5. Set CC to 0.

End op.
Diagram 5-13. Load Multiple, LM (98)
• RR format: AR, ALR, SR, SLR, CR.

- Op Code R1 R2

- Purpose:
  1. AR, ALR, SR, SLR - Algebraically add (subtract) 2nd operand (in GPR, per R2) to (from) 1st operand (in GPR, per R1) and place result into 1st operand location.
  2. CR - Algebraically compare 1st operand (in GPR, per RI) with 2nd operand (in GPR, per R2) and set CC according to result.

- Conditions at start of execution:
  1. Instruction is in E.
  2. 1st operand is in A, B, and D.
  3. 2nd operand is in S and T.
  4. Compare, CR = 1H.

- Op Codes:
  1. Add, AR = I8.
  2. Add Logical, ALR = I E.
  4. Subtract Logical, SLR = I F.

Diagram 5-6

• RX format: A, AH, AL, S, SH, SL, C, CH.

- Op Code R1 X2

- Purpose:
  1. A, AL, S, SL - Algebraically add (subtract) 2nd operand (in storage) to (from) 1st operand (in GPR, per RI) and place result into 1st operand location.
  2. AH, SH - Algebraically add (subtract) halfword 2nd operand (in storage) to (from) 1st operand (in GPR, per RI) and place result into 1st operand location.
  3. C - Algebraically compare 1st operand (in GPR, per RI) with 2nd operand (in storage) and set CC according to result.
  4. CH - Algebraically compare 1st operand (in GPR, per RI) with halfword 2nd operand (in storage) and set CC according to result.

- Conditions at start of execution:
  1. 1st 16 bits of instruction are in E.
  2. 1st operand is in S and T.
  3. 2nd operand address is in D.
  4. Main storage request for 2nd operand has been issued per D.

- Op Codes:
  1. Add, A = 5A.
  2. Add Halfword, AH = 4A.
  3. Add Logical, AL = 5E.
  4. Subtract, S = 58.
  6. Subtract Logical, SL = 5F.
  7. Compare, C = 59.
  8. Compare Halfword, CH = 49.
Diagram S-108. Fixed-Point Add-Type Instructions (Sheet 2 of 2)
Multiply (MIC) IC

• RR format:
  - IC1
  - IC2

• Purpose:
  1. Multiply 1st operand (in GPR, per R1 + 1) by 2nd operand (in GPR, per R2) and place 64-bit result into 1st operand location (in GPR, per R1 and R1 + 1).
  2. Set interruption code 6 and 'program interruption' latch; force end op.

• Conditions at start of execution:
  1. Instruction is in E.
  2. Contents of even-address GPR specified by R1 is in A, B, and D (not used).
  3. Multiplicand (1st operand) is in odd-address GPR specified by R1.
  4. Multiplier (2nd operand) is in S and T.

Diagram 5-8

Set interruption code 6 and 'program interruption' latch; force end op.

Diagram 5-9

Transfer multiplicant from GPR to T per R1 + 1.

Set interruption code 6 and 'program interruption' latch; force end op.

Diagram 5-10

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer multiplicand from GPR to T.

Diagram 5-11

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer A(32-63) to PA8(32-63).

Diagram 5-12

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer multiplicand from GPR to T.

Diagram 5-13

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer A(32-63) to PA8(32-63).

Diagram 5-14

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer multiplicand from GPR to T.

Diagram 5-15

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer A(32-63) to PA8(32-63).

Diagram 5-16

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer multiplicand from GPR to T.

Diagram 5-17

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer A(32-63) to PA8(32-63).

Diagram 5-18

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer multiplicand from GPR to T.

Diagram 5-19

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer A(32-63) to PA8(32-63).

Diagram 5-20

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer multiplicand from GPR to T.

Diagram 5-21

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer A(32-63) to PA8(32-63).

Diagram 5-22

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer multiplicand from GPR to T.

Diagram 5-23

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer A(32-63) to PA8(32-63).

Diagram 5-24

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer multiplicand from GPR to T.

Diagram 5-25

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer A(32-63) to PA8(32-63).

Diagram 5-26

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer multiplicand from GPR to T.

Diagram 5-27

Place 15 in E(12-15), and 3 into STC(0-2).

Transfer A(32-63) to PA8(32-63).
Objectives:

1. Select multiple (of T).
   a. Select M1,M2 bits from S per E(12-15) (A of Sheet 3).
   b. Select multiple (of T) per M1,M2 bits and 'TX' trigger (Table 1, Sheet 3).

2. Develop partial product (PP) bit-pair:
   a. Add multiple to PP in AB (shifted right 2) B of Sheet 3.
   b. Gate new PP to AB.
   c. Gate PP bit-pair [1064,407] to SAL per (E14,15) C of Sheet 3.

3. Develop PP byte:
   Add SAL to F.

4. Develop low-order PP in S:
   a. When PP byte is complete, gate SAL to 5 per STC.
   b. When S is full (4 bytes of PP has been loaded), low-order product in S and high-order product is in PAL.

5. Store product:
   a. Shift higher-order product (in PAL) into GPR per RI (even register).
   b. Store lower-order product (in S) into GPR per RI +1 (odd register) per RI for RR instruction.

Diagram 5-109. Fixed-Point Multiply (Sheet 2 of 3)
A. Derivation of Multiple

B. Derivation of Partial Product Byte

C. Derivation of Multiple Selection Bits

D. Table 1: Value of Multiple Determined by Multiple Selection Bits

E. Select M1, M2 bits per E(14,15)

F. Table 2

Diagram 5-109. Fixed-Point Multiply (Sheet 3 of 3)
Perform specification test and set dividend divider signs.

Diagram 5-22
Program Interruption.

Gate low-order dividend to S and high-order dividend to B, true form.

Gate divisor to T.

Set STAT D.
Transfer LSWR to S.
Transfer B to T.
Transfer 2's complement of T and B into S.
Places 2's complement of low-order bits of dividend into S.

Obtains complement of high-order bits of dividend.

Places divisor into T.

Transfer GPR to T per E(B-11) = 1.
Places low-order bits of dividend into S.

Places divisor into T.

Transfer GPR to S per E(B-11) = 1.
Places low-order bits of dividend into S.

Places divisor into T.

Set STAT 8 if B(22) = 1.
Set STAT 0 if E(22) = 1.

Set interruption under 6 and 'program interrupt' (lock, force end stop).

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Yes (Negative Dividend)
Set STAT 8 on

No (Positive Dividend)

Places divisor into T.

Transfer GPR to S per E(B-11) = 1.
Places low-order bits of dividend into S.

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Yes (Negative Dividend)

No (Positive Dividend)

Transfer GPR to S per E(B-11) = 1.
Places low-order bits of dividend into S.

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Set interruption under 6 and 'program interrupt' (lock, force end stop).

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Yes (Negative Dividend)

No (Positive Dividend)

Transfer GPR to S per E(B-11) = 1.
Places low-order bits of dividend into S.

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Set interruption under 6 and 'program interrupt' (lock, force end stop).

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Yes (Negative Dividend)

No (Positive Dividend)

Transfer GPR to S per E(B-11) = 1.
Places low-order bits of dividend into S.

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Set interruption under 6 and 'program interrupt' (lock, force end stop).

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Yes (Negative Dividend)

No (Positive Dividend)

Transfer GPR to S per E(B-11) = 1.
Places low-order bits of dividend into S.

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Set interruption under 6 and 'program interrupt' (lock, force end stop).

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Yes (Negative Dividend)

No (Positive Dividend)

Transfer GPR to S per E(B-11) = 1.
Places low-order bits of dividend into S.

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Set interruption under 6 and 'program interrupt' (lock, force end stop).

Place 1's into A.
Transfer GPR to T per E(B-11) = 1.
Set STC to 000.
2's complement T and transfer to LSWR.

Yes (Negative Dividend)

No (Positive Dividend)

Transfer GPR to S per E(B-11) = 1.
Places low-order bits of dividend into S.
Diagram 5-110. Fixed-Point Divide (Sheet 2 of 6)

A

- Perform specification test.
- Set low-order dividend to 5 and high-order dividend to B in true form, and retain signs.
- Gate divisor to T.

B

- Diagram 5-22
- Program interruption.
- Gate low-order dividend to S and high-order dividend to B in true form, and retain signs.

C

- Transfer B minus 1 to B.
- Transfer GPR (low-order dividend) to T per E(S-11)+1.
- 2's complement T and transfer to AB(24-67).
- Set STAT C if S(O) = 1.
- Set E(l2-15) to 0000.
- Transfer T to LSR.
- Set STAT B if B(32) = 1.
- Set STAT G if T(32) = 1.

D

- Transfer LSR to S, set STAT B.
- 2's complement T and transfer to AB(24-67).
- Transfer GPR (low-order dividend) to T per E[22-13] + 1.
- Set STAT C if S(10) = 1.
- Set E(13-10) to 0000.

E

- Transfer divisor from SDB0(-32-63) to T.
- Transfer divisor from SDB0(0-31) to T.

F

- Carry from P (28).
- Correct high-order dividend.
- Yes
- No

G

- Yes
- No

D (SD) Instruction Initialization

- Set STAT B if B(22) = 1.
- Set STAT G if T(22) = 1.
Diagram 5-110. Fixed-Point Divide (Sheet 3 of 6)
Diagram 5-110. Fixed-Point Divide (Sheet 4 of 6)
Establish proper sign and format for quotient and remainder, and store.

Yes (2's Complement Quotient)

Transfer 8 to PAB (32-63). Transfer 2's complement of T to PAA (32-63).

Sheet 6

Add PAA and PAB. Transfer result to B.

Reset STAT G.

Addition to correct resultant remainder from last reduction cycle.

Yes (Negative Dividend)

2's complement T to correct value.

Transfer T to GPR per E (S-11). +

Transfer B to T. Transfer T to GPR per E (S-11).

Store quotient.

Store remainder.

Termination, Quotient in 2's Complement Form

Set interruption code 9 and 'program interrupt' latch.

1-cycle early end op.

Diagram 5-22

Program interruption.

S-110, Sr 5
Diagram 5-110. Fixed-Point Divide (Sheet 6 of 6)
Diagram 5-111. Convert to Binary, CVB (4F) (Sheet 1 of 2)

Purpose: Convert radix of 2nd operand (in memory) from decimal to binary and place result in 1st operand location (in GPR, per 4F).

Conditions on start of operation:
1. 1st bits of instruction are in E.
2. For operand 0 is 2 and 1 that used.
3. 2nd operand address is in D.
4. Memory storage request for 2nd operand has been issued per 0.

RX format:

| 4F | R1 | X2 | 82 | 02 | 78 | 11 12 15 16 19 20 31 |

In effect, multiplies result by 2.

In effect, multiplies sum by 5.

In effect, multiplies sum by 5.

Sheet 2
Diagram 5-111. Convert to Binary, CVG (4F) (Sheet 2 of 2)
Diagram 5-22

Set interruption code 6 and "program interruptions" latch; force end op.

Diagram 5-22

Set mark triggers 4-7 and gate operand into right half of main storage word.

Diagram 5-22

Set "PSC" trigger.

Diagram 5-9

RX I-Fetch.

Diagram 5-113

Store, ST (50)

Diagram 5-113

RX format:

<table>
<thead>
<tr>
<th>S0</th>
<th>R1</th>
<th>X2</th>
<th>B2</th>
<th>D2</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>11</td>
<td>12</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

Purpose: Store 1st operand (in GPR, per R1) into 2nd operand location (in storage).

Conditions at start of execution:
1. 1st 16 bits of instruction are in E.
2. 1st operand is in S and T.
3. 2nd operand address is in D.
4. Main storage request for 2nd operand has been issued per D (not used).

Specification test.

Transfer 2's complement of D to PAA(40-63) and 7 to PAA(61-63).

Transfer IC to PAB(40-63).

Add PAA and PAB. Shift right 4 into PAL.

Late protection program interruption occurs after execution of next instruction.
Diagram 5-9
RX Ifetch.

Diagram 5-22
Program Interruption.

Compare Location of Halfword.

Diagram 5-114. Store Halfword, STH (40)
Diagram 5-13
RS I-fetch.

Perform specification test. Issue main storage request.

Diagram 5-115. Store Multiple, STM (90) (Sheet 1 of 2)
Diagram 5-13. Store Multiple, STM (90) (Sheet 2 of 2)
A) Simplified Flow Chart

1. **Yes (Shift Left 3 or Less)**
   - **Diagram 5-13**
   - RS I-Fetch.
   - RS format:
   - I: Rl 02
   - 8: I: Rl 02
   - Purpose: Shift 1st operand (in GPR, per Rl) left number of bit positions specified by low-order 6 bits of 2nd operand address.
   - Conditions on start of execution:
     1. Set STAT C if first operand is negative.
     2. 1st 16 bits of instruction are in E.
     3. Amount of shift is in D and PAL.

2. **No (Shift Left 4 or More)**
   - Shift left 4.
   - Decrement E(I 2-15), shift left 4.
   - Shift left 4, and check for overflow.
   - Set CC per hardware conditions.
   - End op.

B) Detailed Flow Chart (Continued on Sheet 2)

Diagram 5-116. Shift Left Single, SLA (8B) (Sheet 1 of 2)
Diagram 5-116. Shift Left Single, SLA (8B) (Sheet 2 of 2)
Diagram 5-117. Shift Left Double, SLDA (8F) (Sheet 1 of 4)
Diagram 5-17. Shift Left Double, SLDA (BF) (Sheet 2 of 4)
Diagram 5-117. Shift Left Double, SLDA (SF) (Sheet 3 of 4)
STORE HIGH-ORDER WORD
Transfer T to GPR per E(S-11).

Places high-order half of operand into LS.

Transfer STAT C to sign position of GPR per E(S-11).

STORE LOW-ORDER WORD
Subtract T from E(12-15).

Transfer B to PAL(32-63).

Reset STAT A if PAL(32-63) does not equal zero.

Transfer PAL(22-60) to T.

Transfer T to GPR per E(S-11) + 1.

Set CC per hardware conditions.

Result is zero (STAT A set): CC = 0.
Result is minus (ABS = 0): CC = 1.
Result is plus (ABS = 0): CC = 2.
Overflow (STAT B set): CC = 3.

End op.

Diagram 5-117. Shift Left Double, SLDA (8F) (Sheet 4 of 4)
Diagram 5-13
RS I-Fetch.
Test for all 0's.

Yes (Shift Right 3 or Less)

Diagram 5-18, Shift Right Single, SRA (BA) (Sheet 1 of 3)
Diagram 5-128: Shift Right Single, SRA (8A) (Sheet 2 of 3)

- Shift Right Single, SRA (8A)
  - No Shift
    - Transfer T to PAA(31-63), and propagate T(32) to PAA(26-31).
  - Shift Right 1
    - Transfer T to PAA(31-63), and propagate T(32) to PAA(26-31).
    - Transfer PAA(24-47) to PAL(4-63).
    - Transfer PAA(32-63) to AB(24-67).
  - Shift Right 2
    - Transfer T to PAA(31-63), and propagate T(32) to PAA(26-31).
    - Transfer PAA(24-47) to PAL(4-63).
    - Transfer PAA(32-63) to AB(24-67).
  - Shift Right 3
    - Transfer T to PAA(31-63), and propagate T(32) to PAA(26-31).
    - Transfer PAA(24-47) to PAL(4-63).
    - Transfer PAA(32-63) to AB(24-67).

- Conditions at start of execution:
  1. 1st 16 bits of instruction are in E.
  2. 1st operand is in S and T.
  3. Amount of shift is in D and PAL.

- Hardware Result is minus (T(32) = 1): CC = 1.
- Hardware Result is plus (T(32) = 0): CC = 2.
- End op.

- Set CC per Result is zero (STAT A set): CC = 0.

- Transfers shifted operand to LS.
Diagram 5-118. Shift Right Single, SRA (SA) (Sheet 3 of 3)
Diagram 5-119, Shift Right Double, SRDA (8E) (Sheet 1 of 4)
Diagram 5-13.

RS: Fetch.

Set instruction code and operand.

Transfer PAL(24-67) to AB(24-67).

Transfer T to GPR.

Transfer parallel adder (4-63) to PAL(8-67).

Transfer parallel adder (4-63) to PAL(32-63).
Shift Right Double, SRDA (SE) (Sheet 3 of 4)
SHIFT LOW-ORDER WORD RIGHT 4

SHIFT HIGH-ORDER WORD RIGHT 4

SHIFT LOW-ORDER WORD RIGHT 4

SHIFT HIGH-ORDER WORD RIGHT 4

STORE LOW-ORDER WORD

END OF CURRENT LOOP

SHIFT RIGHT DOUBLE, SRDA (SE) (SHEET 4 OF 4)
Diagram 5-202. Load, LER (38) - Short Operands; Load, LDR (28) - Long Operands

**LER RR Format - Short Operands**

- **Purpose:** Load 2nd operand (in FPR) per R2 into 1st operand location (in FPR) per R1.
- **Conditions at start of execution:**
  1. Instruction is in E.
  2. 1st operand is in A, B, and D.

**LDR RR Format - Long Operands**

- **Purpose:** Load 2nd operand (in FPR) per R2 and R2 + 1 into 1st operand location (in FPR) per R1 and R1 + 1.
- **Conditions at start of execution:**
  1. Instruction is in E.
  2. 1st operand is in A, B, and D.
  3. 2nd operand is in S and T.

**Diagram 5-22**

- Program interruption.
  - Yes: Set interrupt code triggers and force end-op.
  - No: Instruction
    - Yes: LER Instruction
      - Transfer low-order fraction of 2nd operand from FPR per E(12-15) + 1 to T.
      - Load low-order fraction from T into FPR per E(B-11) + 1.
      - Transfer sign, characteristic, and high-order fraction from S to T via parallel adder.
      - Prepare to load sign, characteristic, and high-order fraction.
      - Data is parity-checked in parallel adder.
      - Load high-order FPR specified by R1 now contains data from FPR specified by R2.
      - LAI(0) is forced to a 1 to address FPR's.
    - No: LDR Instruction
      - Transfer low-order fraction from T into FPR per E(B-11) + 1.
      - Load low-order fraction from T into FPR per E(B-11) + 1.
      - Prepare to load sign, characteristic, and high-order fraction.
      - Data is parity-checked in parallel adder.
      - Load sign, characteristic, and high-order fraction from S to T via parallel adder.
      - LAI(0) is forced to a 1 to address FPR's.
- End op.
Diagram 5-22. Set interrupt code triggers and force end op.

Diagram 5-23. Program interruption.

Diagram 5-203. Load, LE (78) - Short Operands; Load, LD (68) - Long Operands
Objectives:

1. Fetch operand.
2. Insert sign.
3. Store 2nd operand into 1st operand location.
4. Test for zero result.

Diagram 5-22
Program interruption.

Diagram 5-204. Load Positive, LPER (30); Load Negative, LNER (31); Load and Test, LTER (32); Load Complement, LCER (33) - Short Operand
Objectives:
1. Fetch operands.
2. Insert sign.
3. Store 2nd operand into 1st operand location.
4. Test for zero result.

Diagram 5-22

Program interruption.

Objectives:
1. Set CC per hardware conditions.
2. End op.

Diagram 5-205

Load Positive, LPDR (20); Load Negative, LNDR (21); Load and Test, LTDR (22); Load Complement, LCDR (23) - Long Operands
Diagram 5-206. Floating-Point Add, Subtract, and Compare - Short Operands (Sheet 1 of 5)
Diagram 5-206, Floating-Point Add, Subtract, and Compare - Short Operands (Sheet 2 of 5)

A

Set interrupt code triggers and forces end op.

B

Diagram 5-20

Program interruption.

C

Yes

Diagram 5-20

Objectives:
1. Place 1st operand into A.
2. Fetch 2nd operand and place into T.

No

D

Objectives:
1. Fetch 1st operand from SDB0(0-31) per effective address and place into T.
2. Fetch 2nd operand from SDB0(32-63) per effective address and place into T.

E

Objectives:
1. Place sign of 1st operand into STATF.
2. Place sign of 2nd operand into STATC.

F

Objectives:
1. If serial adder carry occurred (2nd operand characteristic 2 is operand element), 1 if signs are alike (odd fractions).
2. If operands are within range (characteristic difference ≤ 7), serial adder carry and STC = 0 ('RESET' micro-order during I-Fetch).

See table on this page.

Diagrams 5-22, 5-23

Objectives:
1. Add 2nd operand to 1st operand (characteristics equal).
2. Subtract 1st operand from 2nd operand (characteristics equal).
3. Subtract 2nd operand from 1st operand (characteristics equal).
4. Subtract 1st operand from 2nd operand (characteristics equal).
5. Add 2nd operand to 1st operand (characteristics equal).
6. Subtract 1st operand from 2nd operand (characteristics equal).
7. Subtract 1st operand from 2nd operand (characteristics equal).
8. Add 2nd operand to 1st operand (characteristics equal).
9. Subtract 2nd operand from 1st operand (characteristics equal).
10. Subtract 2nd operand from 1st operand (characteristics equal).
11. Add 2nd operand to 1st operand (characteristics equal).
12. Subtract 2nd operand from 1st operand (characteristics equal).

Diagrams 5-22

Objectives:
1. Add 2nd operand to 1st operand (characteristics equal).
2. Subtract 1st operand from 2nd operand (characteristics equal).
3. Subtract 1st operand from 2nd operand (characteristics equal).
4. Subtract 2nd operand from 1st operand (characteristics equal).
5. Add 2nd operand to 1st operand (characteristics equal).
6. Subtract 2nd operand from 1st operand (characteristics equal).
7. Subtract 2nd operand from 1st operand (characteristics equal).
8. Add 2nd operand to 1st operand (characteristics equal).
9. Subtract 2nd operand from 1st operand (characteristics equal).
10. Subtract 2nd operand from 1st operand (characteristics equal).
11. Add 2nd operand to 1st operand (characteristics equal).
12. Subtract 2nd operand from 1st operand (characteristics equal).

Diagram 5-20

Objectives:
1. Add 2nd operand to 1st operand (characteristics equal).
2. Subtract 1st operand from 2nd operand (characteristics equal).
3. Subtract 1st operand from 2nd operand (characteristics equal).
4. Subtract 2nd operand from 1st operand (characteristics equal).
5. Add 2nd operand to 1st operand (characteristics equal).
6. Subtract 2nd operand from 1st operand (characteristics equal).
7. Subtract 2nd operand from 1st operand (characteristics equal).
8. Add 2nd operand to 1st operand (characteristics equal).
9. Subtract 2nd operand from 1st operand (characteristics equal).
10. Subtract 2nd operand from 1st operand (characteristics equal).
11. Add 2nd operand to 1st operand (characteristics equal).
12. Subtract 2nd operand from 1st operand (characteristics equal).
Diagram 5-206. Floating-Point Add, Subtract, and Compare - Short Operands (Sheet 3 of 5)
Objectives:
1. If overflow or underflow and PSW(38) = 1 ·, initiate overflow or underflow interruption request.
2. If underflow and PSW(38) = ~0, store 0's into FPR per R1.
3. If overflow or underflow and PSW(38) = 1, store 0's into FPR per R1.

Objective:
Check guard digit for significant bits.

Transfer D to DT via PAL
Reset STAT D.
Set STAT A if fraction equals 0 [PAL(7-67) = 0].

Result i0 and Fraction Normalized

Store 0's into FPR per E(B-11).

Store T into FPR per E(S-11) (0's are stored).
Add 1 to F.
Reset STAT D.

Result /i0 and Fraction Not Normalized
Shift DT L4 and place fraction into DT.
Subtract 1 from F.

Result, i0 and Fraction Not Normalized
Shift DT and B(64-67) L4 and place fraction into DT and AB.
Subtract 1 from F.

Return zero fraction/lost significance routine.

Diagram 5-206. Floating-Point Add, Subtract, and Compare - Short Operands (Sheet 4 of 5)

5-207. Floating-Point Add, Subtract, and Compare - Long Operands (Sheet 1 of 5)
Diagram 5-207. Floating-Point Add, Subtract, and Compare - Long Operands (Sheet 3 of 5)
Diagram 5-207. Floating-Point Add, Subtract, and Compare - Long Operands (Sheet 4 of 5)
Diagram 5-207. Floating-Point Add, Subtract, and Compare - Long Operands (Sheet 5 of 5)
Diagram 5-208. Halve, HER (34) - Short Operands

- **RR Format - Short Operands**
  
  
<table>
<thead>
<tr>
<th>34</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose**: Divide 2nd operand (in FPR, per R2) by 2 and place normalized quotient less for operand location (in FPR, per R1).

- **Conditions of start of execution**:
  1. Instruction is in E.
  2. 1st operand is in A, B, and D.
  3. 2nd operand is in S and T.

- **RR l-Fetch**

- **Program interruption**

- Set interrupt code triggers and force end op.

- **Diagram 5-22**

- **Get 2nd operand fraction from T to D**.

- Shift fraction R1 and gate to D (via PAL).

- **Diagram 5-6**

- **RR I-Fetch**

- **Gate sign to STAT C**.

- Save characteristic in F.

- Shift fraction R1 and gate to D (via PAL).

- Shift fraction Ld to D (includes guard digit).

- Subtract 1 from characteristic in F.

- Save characteristic in F.

- **Diagram 5-201**

- **Save signs** micro-order

- **Diagram 5-208**

- **Unnormalized**

- **Normalized**

- Test for fraction = 0 and unnormalized

- Gate characteristic and fraction to T and store into FPR per R1.

- Set sign per STAT C.

- **Underflow**

- **Yes**

- **PSWDR set**

- **No**

- **Set "program interrupt" trigger**.

- **Yes**

- **End op.**

- **No**

- **Store O's into FPR per R1.**
Set interrupt code triggers and hence end op.

Diagram 5-22
Program interruption.

Save characteristic in F.

Diagram 5-209
Hahn, HDR (24) - Long Operands

Diagram 5-209
...
A. Sign and Characteristic Data Paths.

B. Fraction Data Path.

Diagram 5-210. Floating-Point Multiply Data Paths

Notes:
1. In Z605 floating-point multiply operations, roles of 1st and 2nd operands are reversed from roles defined in System/360 Principles of Operation, SML, Form A22-0821-0. That is, 2nd operand is multiplicand and 1st operand is multiplier. (Interchanging operand roles does not affect product. Result, however, still replaces 1st operand.)

2. For an RX instruction with normalized 1st operand, the 1st operand characteristic and sign are in S or T and the 2nd operand characteristic and sign are in A.
Diagram 5-22. Floating-Point Multiply, Short Operands (Sheet 1 of 4)
Diagram 5-211. Floating-Point Multiply, Short Operands (Sheet 2 of 4)
Diagram 5-211. Floating-Point, Short Operands (Sheet 3 of 4)
Diagram 5-211. Floating-Point Multiply, Short Operands (Sheet 4 of 4)
Diagram 5-212. Floating-Point Multiply, Long Operands (Sheet 1 of 4)
Objectives:
1. Place 1st operand into AB.
2. Fetch 2nd operand and place into ST and DT.

<table>
<thead>
<tr>
<th>Diagram 5-9</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX I-Fetch.</td>
</tr>
</tbody>
</table>

Transfer sign, characteristic, and high-order fraction of 1st operand from T to A.

Set interrupt code triggers and force program interruption.

Fetch low-order fraction of 2nd operand from FPR and place into T.

Place constant 15 into E(12-15) and 4 into STC.

Transfer high-order fraction of 2nd operand from S to D.

Save signs. 2nd operand sign to STAT C; 1st operand sign to STAT F.

Add characteristics. Save carry in STAT D (SAL(0) to STAT D).

Branch test is made 1 cycle before branching. Multiply algorithm assumes that 2nd operand is normalized.

01 02 03 04 05

### Conditions at start of execution:
1. Instruction is in E.
2. 32 bits of 1st operand are in A, B, and D (24-bit fraction only).
3. 32 bits of 2nd operand are in S and T.
4. Low-order fractions of 1st and 2nd operands are in LS.
5. STC = 4.

### RR Format: MDR

<table>
<thead>
<tr>
<th>RX</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>X1</td>
<td>X0</td>
</tr>
</tbody>
</table>

### Purpose:
Multiply 2nd operand (in FPR, per R2 and R2 + 1) by 1st operand (in FPR, per R1 and R1 + 1) and place normalized product into 1st operand location.

### Conditions at start of execution:
1. Instruction is in E.
2. 32 bits of 1st operand are in A, B, and D (24-bit fraction only).
3. 32 bits of 2nd operand are in S and T.
4. Low-order fractions of 1st and 2nd operands are in LS.

### Purpose:
Multiply 2nd operand (in FPR, per R2 and R2 + 1) by 1st operand (in FPR, per R1 and R1 + 1) and place normalized product into 1st operand location.

### Conditions at start of execution:
1. Instruction is in E.
2. 32 bits of 1st operand are in S and T.
3. Low-order fraction of 1st operand is in LS.
4. Effective address of 2nd operand is in D.
5. 2nd operand is in main storage.

### RR Format: MD

<table>
<thead>
<tr>
<th>RX</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>X1</td>
<td>X0</td>
</tr>
</tbody>
</table>

### Purpose:
Multiply 2nd operand (in FPR, per R1 and R1 + 1) by 1st operand (in FPR, per R2 and R2 + 1) and place normalized product into 1st operand location.

### Conditions at start of execution:
1. Instruction is in E.
2. 32 bits of 1st operand are in S and T.
3. Low-order fraction of 1st operand is in LS.
4. Effective address of 2nd operand is in D.
5. 2nd operand is in main storage.

### RR Format: MD

<table>
<thead>
<tr>
<th>RX</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>X1</td>
<td>X0</td>
</tr>
</tbody>
</table>

### Purpose:
Multiply 2nd operand (in FPR, per R1 and R2 + 1) by 1st operand (in FPR, per R2 and R2) and place normalized product into 1st operand location.

### Conditions at start of execution:
1. Instruction is in E.
2. 32 bits of 1st operand are in S and T.
3. Low-order fraction of 1st operand is in LS.
4. Effective address of 2nd operand is in D.
5. 2nd operand is in main storage.

### RR Format: MD

<table>
<thead>
<tr>
<th>RX</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>X1</td>
<td>X0</td>
</tr>
</tbody>
</table>

### Purpose:
Multiply 2nd operand (in FPR, per R1 and R2) by 1st operand (in FPR, per R2 and R2 + 1) and place normalized product into 1st operand location.

### Conditions at start of execution:
1. Instruction is in E.
2. 32 bits of 1st operand are in S and T.
3. Low-order fraction of 1st operand is in LS.
4. Effective address of 2nd operand is in D.
5. 2nd operand is in main storage.

### RR Format: MD

<table>
<thead>
<tr>
<th>RX</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>X1</td>
<td>X0</td>
</tr>
</tbody>
</table>

### Purpose:
Multiply 2nd operand (in FPR, per R2 and R2 + 1) by 1st operand (in FPR, per R2 + 1 and R2) and place normalized product into 1st operand location.

### Conditions at start of execution:
1. Instruction is in E.
2. 32 bits of 1st operand are in S and T.
3. Low-order fraction of 1st operand is in LS.
4. Effective address of 2nd operand is in D.
5. 2nd operand is in main storage.

### RR Format: MD

<table>
<thead>
<tr>
<th>RX</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>X1</td>
<td>X0</td>
</tr>
</tbody>
</table>

### Purpose:
Multiply 2nd operand (in FPR, per R2 + 1 and R2) by 1st operand (in FPR, per R2 and R2 + 1) and place normalized product into 1st operand location.

### Conditions at start of execution:
1. Instruction is in E.
2. 32 bits of 1st operand are in S and T.
3. Low-order fraction of 1st operand is in LS.
4. Effective address of 2nd operand is in D.
5. 2nd operand is in main storage.

### RR Format: MD

<table>
<thead>
<tr>
<th>RX</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>X1</td>
<td>X0</td>
</tr>
</tbody>
</table>
Diagram 5-212. Floating-Point Multiply, Long Operands (Sheet 4 of 4)
Diagram 5-214. Floating-Point Divide, Short Operands (Sheet 1 of 4)
Diagram 5-9
QTOOl
QSOOl
RX I-Fetch.

Diagram 5-6
SS I-Fetch.
QCOAl

Objective:
Fetch 2nd operand and place into D.

D

OCR1 = 1

No

Yes

Dorl<1

Fetch 2nd operand from SRNO(30-33) and place into T.

Fetch 2nd operand from SRNO(30-33) and place into T.

Transfer 2nd operand fraction from T to D.

Objective:
1. Subtract 1st operand characteristic from 2nd operand characteristic.
2. Save signs in STA.T's.
3. Adjust characteristic.
4. Subtract 64 from characteristic difference.
5. Save ABC, T, and B.

A

Sheet 3

Figure 5-214. Floating-Point Divide, Short Operands (Sheet 2 of 4)
Diagram 5-214: Floating-Point Divide, Short Operands (Sheet 3 of 4)
Diagram 5-214. Floating-Point Divide, Short Operands (Sheet 4 of 4)
Diagram 5-215. Floating-Point Divide, Long Operands (Sheet 1 of 5)
Diagram 5-215. Floating-Point Divide, Long Operands (Sheet 2 of 5)
Diagram S-215. Floating-Point Divide, Long Operands (Sheet 3 of 5)
Diagram 5-215. Floating-Point Divide, Long Operands (Sheet 5 of 5)
Diagram 5-216. Store, STE (70) – Short Operands; Store, STD (60) – Long Operands
Diagram 5-301. GIS for Decimal Add, Subtract, and Compare

Decimal Add Instruction

<table>
<thead>
<tr>
<th>Ope Code</th>
<th>0 1</th>
<th>2 3</th>
<th>4 5</th>
<th>6 7</th>
<th>8 9</th>
<th>10 11</th>
<th>12 13</th>
<th>14 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Add Decimal (AP) - FA</td>
<td>1 Subtract Decimal (SP) - FB</td>
<td>2 Compare Decimal (CP) - F9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Purpose:
1. Add Decimal, Subtract Decimal - Algebraically add (subtract) 2nd operand to (from) 1st operand and place result into 1st operand location.
2. Compare Decimal - Algebraically compare 1st operand with 2nd operand (in storage) and set CC according to result.

Conditions at end of I-Fetch:
1. Main storage request for doubleword containing low-order byte of 1st operand (destination) has been issued per D.
2. D contains low-order byte address (contents of GPR addressed by B + D1 + L1) of 1st operand.
3. IC contains high-order byte address (contents of GPR addressed by H2) of 2nd operand.

If no higher-priority interruption, execute program interruption (Diagram 5-22).

Result recomplemented and sign inverted.

Decimal Subtract Instruction

<table>
<thead>
<tr>
<th>Ope Code</th>
<th>0 1</th>
<th>2 3</th>
<th>4 5</th>
<th>6 7</th>
<th>8 9</th>
<th>10 11</th>
<th>12 13</th>
<th>14 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Add Decimal (AP) - FA</td>
<td>1 Subtract Decimal (SP) - FB</td>
<td>2 Compare Decimal (CP) - F9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Purpose:
1. Add Decimal, Subtract Decimal - Algebraically add (subtract) 2nd operand to (from) 1st operand and place result into 1st operand location.
2. Compare Decimal - Algebraically compare 1st operand with 2nd operand (in storage) and set CC according to result.

Conditions at end of I-Fetch:
1. Main storage request for doubleword containing low-order byte of 1st operand (destination) has been issued per D.
2. D contains low-order byte address (contents of GPR addressed by B + D1 + L1) of 1st operand.
3. IC contains high-order byte address (contents of GPR addressed by H2) of 2nd operand.

If no higher-priority interruption, execute program interruption (Diagram 5-22).

Result recomplemented and sign inverted.

Decimal Compare Instruction

<table>
<thead>
<tr>
<th>Ope Code</th>
<th>0 1</th>
<th>2 3</th>
<th>4 5</th>
<th>6 7</th>
<th>8 9</th>
<th>10 11</th>
<th>12 13</th>
<th>14 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Add Decimal (AP) - FA</td>
<td>1 Subtract Decimal (SP) - FB</td>
<td>2 Compare Decimal (CP) - F9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Purpose:
1. Add Decimal, Subtract Decimal - Algebraically add (subtract) 2nd operand to (from) 1st operand and place result into 1st operand location.
2. Compare Decimal - Algebraically compare 1st operand with 2nd operand (in storage) and set CC according to result.

Conditions at end of I-Fetch:
1. Main storage request for doubleword containing low-order byte of 1st operand (destination) has been issued per D.
2. D contains low-order byte address (contents of GPR addressed by B + D1 + L1) of 1st operand.
3. IC contains high-order byte address (contents of GPR addressed by H2) of 2nd operand.

If no higher-priority interruption, execute program interruption (Diagram 5-22).

Result recomplemented and sign inverted.
Diagram 5-302. True Add Sequence for Decimal Add, Subtract, and Compare (Sheet 1 of 3)

Diagram 5-301

A

True add loop.

1

True add digit in sign byte.

B

2nd operand fetch.

C

L1 > 10

L1 = 10

D

No

Destination store.

Yes

Contains 1st Operand Address

E

Destination store.

Overflow test.

Restore initial destination address.

Start plus (+) into destination.
Diagram 5-302. True Add Sequence for Decimal Add, Subtract, and Compare (Sheet 2 of 3)
Set 'invalid data' or 'overflow' trigger, as applicable.

Restore instruction address from LSWR to IC.

Yes

If no higher-priority interruption, execute program interruption (Diagram 5-22).

No

Stare Plus (+) into Destination (QSO21)

Place plus (+) into ST per STC.

Set mark trigger per STC.

Issue storage request per D.

Add 16 to D address.

Add 8 to D address.

Yes

ID (overflow)

Set STC per D(21-23). Gate F(4-7) to parallel adder.

Number of times D was reduced.

Add 16 to D address.

Add 8 to D address.

Yes

STAT A set

No

STAT Q set

No

Result = 0

Diagram 5-302. True Add Sequence for Decimal Add, Subtract, and Compare (Sheet 3 of 3)
Diagram 5-303. Complement Add Sequence for Decimal Add, Subtract, and Compare (Sheet 1 of 3)

Note: See Diagram 5-302 for general data path.
Diagram 5-303. Complement Add Sequence for Decimal Add, Subtract, and Compare (Sheet 3 of 3)
Note: See Diagram 5-302 for general data path.

Diagram 5-304. Zero and Add (Sheet 1 of 4)
Diagram 5-14
SS I-Fetch.

Gate 1st operand from SORO to ST.

Add L2 to IC and request 2nd operand. Transfer D to STC.

Subtract D from IC.

- No: Result = 0 (Possible Word Overlap)
- Yes: Subtract L2 from 2nd operand address.
  Subtract (GPR per B2, + D2) from (GPR per B1, + D1)

Gate SORO to AB. Set ABC per (C21=14).

General Initialization Sequence (G0091)

D contains: contents of GPR addressed by B1, + D1, + L1.
IC contains: contents of GPR addressed by B2, + D2, + L2.

---

Diagram 5-304. Zero and Add (Sheet 2 of 4)

Subtract L2 from 2nd operand address.

No: Result = 0

Yes: Result = 0 (Possible Word Overlap)

Gate SORO to AB. Set ABC per (C21=14).

Return instruction address to IC.

Set 'invalid data' trigger.

If no higher-priority interruption, execute program interruption (Diagram 5-22).

Interruption (G0091)

---

SS Format

Purpose: Place 2nd operand (in storage) into 1st operand location (in storage).

Conditions at end of I-Fetch:
1. Main storage request for doubleword containing low-order byte of 1st operand (destination) has been issued per D.
2. D contains low-order byte address (contents of GPR addressed by B1, + D1, + L1) of 1st operand.
3. IC contains high-order byte address (contents of GPR addressed by B2, + D2) of 2nd operand.
Sheet 2

Set STAT if digit is invalid; go to byte (bits 0-3), per ABC, to S_AA(0-3); gate correct sign to SAA(4-7); gate 0's to SAB. Gate SAL to ST per STC.

No

Yes

Set mark trigger per STC.

Reduce L1, L2, ABC, and STC.

Set STAT C.

Set STAT A.

Diagram 5-304. Zero and Add (Sheet 3 of 4)
Set 'invalid data' or 'overflow' trigger, as applicable.

If no higher-priority interruption, execute program interruption. (Diagram 5-22).

Add 16 to D address.

Yes (Overflow) Set STC per STC per STC.

No (Zero Result) Issue storage request per D.

Yes (Compare) STAT G set

No (Result = 0)

Set STC per STC per STC, Gate F(4-7) to parallel adder.

Add 8 to D address.

Place plus (+) into ST per STC.

Add 8 to D address.

Sheet 3

Sheet 4

Diagram 5-304. Zero and Add (Sheet 4 of 4)
Diagram 5-14

SS I-Fetch.

Specification test.

Specify valid

No

Specification valid

Yes

Incorrect specification (interrupted).

Multiplier (right-adjust sequence).

Multiplier right-shift to drop signs.

Sign handling.

End op.

Invalid data (interrupted).

Yes

Basic multiply add or subtract sequence for left digit.

Yes

Invalid 0 test and partial product store.

Yes

Remain 0

Multiplicand (left-adjusted).

Yes

Basic multiply add or subtract sequence for right digit, and shift right-shift sequence.

Partial product byte store.

Complete multipl.

Yes

Multiplier right-shift to drop digit.

Multiply.

No

Basic multiply add or subtract sequence for left digit.

No

L2 restoration.

Multiplier right-shift to drop signs.

Basic multiply add or subtract sequence for right digit, and shift right-shift sequence.

Complete multipl.

No

L1 = L2

Multiplicand (right-adjusted).

End op.

A) Overall Flow Chart

Contains Multiplied Address and Final Product Address

0 0 10 30

Contains Multiplier Address

0 27

Contains Partial Product (left-adjusted)

0 1 2 13 24 31 40 47 56 63

Combine 2 Digits of Multiplied

0 7

Combine Multiplier (left-adjusted)

0 27 34 47

Controls Number of Additions or Subtractions

0 7

Invalid Digit Detection

0 7

Serial adder used for addition or subtraction of multiplier from partial product, one byte at a time.

Parallel adder used to manipulate data and pass output.

PAL is also used as an auxiliary register.

B) General Data Path

Diagram 5-305. Decimal Multiply (Sheet 1 of 7)
Diagram 5-14

SS I-Fetch.

Gate 1st operand (multiplicand) from SDBO to ST.

Add L2 to IC and Request 2nd operand (multiplier). Transfer D to STC.

Reduce IC by 8 to address next data word of multiplier.

Set STAT .F if multiplicand (destination) sign is minus.

Set STAT E if multiplicand (destination) sign is invalid.

Gate multiplier from SDBO to AB.

Set ABC per IC(21-23).

Transfer low-order multiplicand digit from ST to F(0-3).

Transfer instruction address from LSWR to S.

Divide operation.

GIS (QT031 and QS041)

Objectives:
1. Place multiplier into AB.
2. Initiate sign handling.
3. Save low-order digit of multiplicand in F.

Set STAT .F if multiplicand (destination) sign is minus.

Set STAT E if multiplicand (destination) sign is invalid.

Gate instruction address from S to JC.

Restore instruction address to IC and force interruption.

Set interrupt code triggers and force end op.

OBJECTIVES

Verify that L2 is less than 8 bytes smaller then L1.

Program interruption.

Diagram 5-22

Program interruption.

Gate Instruction address from S to IC.

Set lines opt code triggers and force op.

Diagram 5-302

Decimal Multiply (Sheet 2 of 7)
A

**Objective:**
Fetch full multiplier from storage and transfer to ST. High-order multiplier digit must occupy leftmost byte. Bank S is in ST.

Transfer instruction address from S to PAL, and hold.

B

Gate L2 from E to STC.

More multiplier sign byte from AB to ST (per ST C).

Set STAT C if multiplier sign is minus. Set STATE if multiplier sign or units digit is invalid.

Move next multiplier byte from AB to ST.

C

Reduce L2, ABC, and STC by 1 count.

Add 5(28-31) to E(l2-15) (in parallel adder).

Gate A(4-31) contains high-order multiplier digits excluding last digit.

LD RESTORATION (QS041)

Objective:
L2 must be restored to E(l2-15) before entering the multiply add or subtract loop.

A(4-31) contains high-order multiplier digits.

Hold PAL to zero last high-order multiplier digit.

Transfer shifted result from PAL to A.

D

Transfer last high-order multiplier digit from PAL (A(l2-15) to A5).

Add 5(28-31) to E(l2-15) (in parallel adder).

Gate L2 from PAL to E(l2-15),

E

Gate high-order multiplier bytes from 3 to PAL and shift right 4.

Add 1 to E(l2-15).

Set STC to 111.

To perform correct gating of L2 from F to E.

Set DFC to 111.

F

Gate PA(7-0) to SDBO(31-25) per STC.

Add SDB(31) to E(15-10) (in parallel adder).

Gate L2 from PA(7-0) to STC.

G

L2 in E(l2-15) was reduced 1 below zero and restored to zero.

0 in E(l2-15) is zero.

H

MULTIPLIER LEFT-ADJUST SEQUENCE (QS041)

Objective:
During left-adjust sequence, instruction address is held in PAL by "HOLD" micro-order.

Gate L2 from E to STC.

STC is set to where the right-most type of multiplier will be placed after left adjustment.

More multiplier sign byte from AB to ST (per ST C).

Set STAT C if multiplier sign is minus. Set STATE if multiplier sign or units digit is invalid.

Transfer performed via serial adder.

I

Reduce L2, ABC, and STC by 1 count.

Move next multiplier byte from AB to ST.

Set STAT A if multiplier digit(s) is not zero. Set STATE if multiplier digit(s) is invalid.

Stop CPU clock for two cycles, then gate multiplier from SDBO to A5.

J

Gate instruction address from PAL to IC.

Full multiplier has been fetched. Instruction address is restored to IC.

L

Reduce L2, ABC, and STC by 1 count.

Move next multiplier byte from AB to ST.

Set STAT A if multiplier digit(s) is not zero. Set STATE if multiplier digit(s) is invalid.

Transfer performed via serial adder.

Diagram 5-305: Decimal Multiply (Sheet 3 of 7)
BASIC MULTIPLY
ADD OR SUBTRACT
SEQUENCE FOR
LEFT DIGIT (QS041 AND QS071)

Objectives:

1. If multiplicand digit is 4 or less, multiplier is added to partial product
   number of times specified by multiplicand digit.

2. If multiplicand digit is 5 or greater, multiplier is subtracted from partial
   product number of times specified by 10 minus value of multiplicand digit.

3. If multiplicand digit is zero, no addition or subtraction cycles are required.

Diagram 5-305. Decimal Multiply (Sheet 4 of 7)
Set D(21-23) per STC.

To designate product byte for store operation.

Transfer low-order half of partial product from T to LSWR.

PRODUCT BYTE STORE (20-23)

Objective: Store low-order partial product byte into main storage.

To indicate that reduce L1 by 1 byte has been count.

Issue D request to store low-order partial product byte.

Cross-gate F through serial adder and gate to ST per STC.

Set Mork trigger per STC.

Place 0000 0001 into F.

PRODUCT BYTE STORE (20-23)

MULTIPLICAND REQUEST (5-305)

Objectives:

1. Reduce D address by 1 and request multiplicand from storage.
2. Set STC to point at new multiplicand byte.

Request multiplicand per D address.

Set STC per D(21-23).

Transfer high-order partial product from PAL to T.

Transfer low-order partial product from PAL(20-23) to PAL(24-27).

MULTIPLICAND REQUEST (5-305)

Objectives:

1. Subtract 1 from multiplicand address in D.
2. Set STC to point at new multiplicand byte.

Add F to D (complemented) to partially reduce address.

Complement result and gate to D.

MULTIPLICAND REQUEST (5-305)

Objective:

1. Reduce D address by 1 and request multiplicand from storage.
2. Set STC to point at new multiplicand byte.

Place 0000 0001 into T.

Subtract 1 from multiplicand address in D.

Transfer high-order partial product from PAL to T.

Transfer low-order partial product from PAL(20-23) to PAL(24-27).

MULTIPLICAND REQUEST (5-305)

Objectives:

1. Reduce D address by 1 and request multiplicand from storage.
2. Set STC to point at new multiplicand byte.

Set STC per D(21-23).
Diagram 5-305. Decimal Multiply (Sheet 6 of 7)
BASIC MULTIPLY ADD OR SUBTRACT SEQUENCE FOR RIGHT DIGIT, AND SHIFT RIGHT 4 SEQUENCE (QS071 AND QS081)

Objectives:
1. If multiplicand digit is 4 or less, multiplier is added to partial product number of times specified by multiplicand digit.
2. If multiplicand digit is 5 or greater, multiplier is subtracted from partial product number of times specified by 10 minus value of multiplicand digit.
3. If multiplicand digit is 0, no addition or subtraction cycles are required.
4. After exit from add or subtract loop, low-order partial product digit is saved in F(4-7) for subsequent store. Partial product in ST is shifted right 4 to drop low-order digit, and a branch is made to left-digit sequence.

Diagram S-305. Decimal Multiply (Sheet 7 of 7)
A) Overall Flow Chart

SS Format:

FD  LI  L2  RI  R1  R2

Purpose: Divide 1st operand (in storage) by 2nd operand (in storage); result is in 1st operand location (quotient is leftmost in 1st operand location; remainder is rightmost).

SAL

BC

Serial adder is used for subtraction of divisor from dividend one byte at a time.

PAA

Parallel adder is used to manipulate data and to perform shift operations. PAA is also used as an auxiliary register.

Diagram 5-306. Decimal Divide (Sheet 1 of 9)
Objectives:

1. Place divisor into AB and set ABC to the low-order divisor byte. Save L1 and L2 in F.
2. Test dividend sign for validity.

Gate divisor from SDBO to AB. Set STATE if dividend sign is invalid; set STAT F if minus.

Add L2 to IC and request 2nd operand (divisor). Transfer D to STC.

Transfer L1 and L2 from (E1-15) to F0-7.

Reduce IC by 8 to address next double-word of divisor.

Set STAT G to record that divisor is 5 bytes or greater.

Transfer instruction address from LSWR to 1.
Objective: Restore instruction address to IC and force interruption.


SPECIFICATION Test (Q5041)

Objective: Verify that divisor is less than 6 bytes and smaller than dividend.

INCOMPLETE SPECIFICATION (Q5041)

Objective: Restore instruction address to IC and force interruption.


DIVIDER LEFT-ADJUST SEQUENCE (Q5041)

Objective: Fetch entire divisor from storage and transfer performed via serial adder.

High-order divisor byte must occupy leftmost byte in ST.

Sheet 4
DIVIDEND FETCH AND LEFT-ADJUST SEQUENCE (QS041 AND QS051)

Objective:
Divisor is shifted right 4 and held in PAL and LSWR. Sufficient number of dividend bytes is fetched to perform trial subtraction. (at most, two dividend fetches are required). Dividend byte is placed into ST, where high-order byte occupies leftmost position. Original D address is restored to D to reflect quotient address.

A

- Hold PAL to save last high-order divisor digit.
- Transfer shifted divisor from PAL to A.
- A(4-31) contains high-order divisor digits excluding last digit.
- Gate high-order divisor from S to PAA and shift right 4 to PAL.
- Gate 8(64-67) to PAB(28-31) and shift right 4 to PAL.
- Transfer shifted divisor from PAL to T.
- B(64-67) serves as temporary storage during right 4 shift.
- Gate dividend byte from AB (per ABC) to ST (per STC).
- This branch senses E(S-15). Because L2 is zero, test is for L1 only.
- Reduce L1 by 1 count and increase ABC and STC by 1 count.
- Reduce L1 by 1 count and increase STC by 1 count.
- Increase ABC by 1 count.
- Gate PAL(64) to D.
- Gate B to parallel adder and subtract 8.
- Gate PAL to D.
- Transfer high-order divisor from A to PAL, and hold.
- Gate 8(64-67) to PAB(28-31) and hold.

B

- Data high-order divisor from S to PAL and shift right 4 to PAL.
- Transfer high-order divisor from PAL to A.
- A(31-4) contains high-order divisor digits excluding last digit.

C

- Data low-order divisor from 1 to PAA and shift right 4 to PAL.
- Gate B(40-47) to PAB(40-47).
- Gate 8(64-67) to PAB(28-31) and shift right 4 to PAL.
- Transfer shifted divisor from PAL to T.
- Original low-order S digit has become high-order T digit.

D

- Transfer shifted divisor from PAL to T.
- Transfer shifted divisor from T to LSWR.
- Increase ABC by 1 count.

E

- Set STC to 000 to select leftmost byte in ST.
- Gate L1 from EB(11) to PAL(64) and shift right 4 to PAL.
- To place high-order L1 bit in PAL(64) for subsequent branch.
- PAL(64) = 0

F

- Set STAT D to record that divisor is less than 8 bytes.
- Data divided from S080 to AB and set ABC per D01(31).

G

- Transfer high-order divisor from A to PAL, and hold.
- During division left adjust sequence, high-order divisor is held in PAL.

H

Diagram 5-306. Decimal Divide (Sheet 4 of 9)
Diagram 5-306. Decimal Divide (Sheet 5 of 9)
TRIAL SUBTRACTION (Q5651 AND Q5661)

Objective:
- Subtract divisor from dividend to verify that quotient and remainder will fit into destination field.

Set STAT H to generate hot carry for trial subtraction.

Set ABC and STC per L2.

Add all byte (complemented) to ST byte (true) and gate result to SAL.

Set STAT A if result is not zero. Set STAT E if result is invalid.

No

ABC = 0

Yes

Reduce ABC and STC by 1 count.

Reduce ABC and STC by 1 count.

Yes

Copy from SAL(5)

No

INTERRUPTION (Q5661)

Objective:
- Set appropriate interrupt trigger to force interruption.

Reset STAT G.

Set 'divide check' trigger.

END OP.

Reset STAT G.

Set 'invalid data' trigger.

INHERIT DIVIDEND LEFT 4 SHIFT (Q5661)

Objective:
- Shift high-order dividend portion left 4.

Transfer high-order digit from T(32-35) to S(64-67).

Transfer low-order dividend from T to LSHR.

Gate high-order dividend from S(32-63) to PAA(32-63).

Gate dividend digit from S(64-67) to PAA(64-67).

Slide parallel output left 4 to PAL.

Original high-order T digit becomes low-order S digit.

Transfer low-order dividend from LSHR to S.

Gate high-order dividend from PAL to T.

y HD Dividend (Shifted Left 4)

y LD Dividend (Shifted Left 4)
Diagram 5-306. Decimal Divide (Sheet 8 of 9)
Diagram 5-306. Decimal Divide (Sheet 9 of 9)

PROCESS QUOTIENT SIGN BYTE (QSOSI)

Objective:
Establish sign for last byte of quotient and store bytes per D address.

A
Set ABC and STC per l2.
See Note 4.
Gate dividend sign byte per STC to adder when sign is.
To test for dividend sign.

B
Set STAT 1 if dividend sign is negative.
Set STAT 8 if dividend sign is negative.

C
Set STC per D2-12.
To designate where quotient sign byte is to be placed.

D
Gate where + to F(4-7).
Transfer quotient byte from F to ST per STC.
Set Mark trigger per STC.
Store ST per D address.
Increase D by 8 to obtain next storage address for reminder.

E
Increase ABC and STC by 1 count.
Reduce D by 1 count.
Reduce STC by 1 count to designate remainder sign byte.

F
Increase D by 1 count to obtain remainder address.

G
No (Signs Not Alike)
Gate minus (-) to F(4-7). Quotient o Digit 314
Sign
Transfer remainder from ST to AB via parallel adder.
Set STAT 3 if divisor is negative. See Divisor

H
Left-Adjust Sequence (Sheet 31).

I
Set STAT E if dividend sign is invalid.
Set ABC and STC per L2.
Gate dividend sign byte per STC to serial adder bus B.
Set STC per 0(21-23).

J
No (Signs Alike)
Gate plus (+) to F(4-7).
Transfer quotient byte from ST to AB per STC.
Set Mark trigger per STC and store ST per D address.

K
Increase ABC and STC by 1 count.
Reduce D by 1 count.
Reduce STC by 1 count to designate remainder sign byte.

L
Increase ABC and STC by 1 count.
Reduce D by 1 count to obtain next storage address for remainder.

M
Correct remainder sign, if necessary.
Receive sign must be same as that of dividend.

N
End op.
Diagram 5-14 · SS I-Fetch.

- SS Format:
  - Op Code
  - LI L2 B1 D0 D2 D3

- Op Codes:
  1. Pack (PACK) - F1.
  2. Unpack (UNPK) - F2.
  3. Move With Offset (MVO) - F3.

- Purpose:
  1. Pack - Convert format of 2nd operand (in storage) from zoned to pocked and place result into 1st operand location (in storage).
  2. Unpack - Convert format of 1st operand (in storage) from pocked to zoned and place result into 1st operand location (in storage).
  3. Move with Offset - Store 2nd operand (in storage) to left of and adjacent to low-order 4 bits of 1st operand (in storage).

- Conditions at end of I-Fetch:
  1. Main storage request for doubleword containing low-order byte of 1st operand (destination) has been issued per D.
  2. D contains low-order byte address (contents of GPR addressed by 81 + 81 + L1) of 1st operand.
  3. IC contains high-order byte address (contents of GPR addressed by 82 + D1) of 2nd operand.

Diagram 5-308. Pack, Not Word Overlap Sequence
Diagram 5-309. Pack, Word Overlap Sequence

*Some end-op and source fetch routines are used for Pack and Move With Offset instructions.*
Diagram 5-311. Unpack, Word Overlap Sequence
Diagram 5-313. Move With Offset, Not Word Overlap Sequence

GENERAL OPS (6200P)

*Some end-op and source fetch routines are used for Move With Offset and Pack instructions.
Diagram 5-313. Move With Offset, Word Overlap Sequence
Diagram 5-401. GIS for Logical Instructions

---

**Diagram 5-401: GIS for Logical Instructions**

**A**
- Transfer IC to PAB(40-63).
- Transfer complement of D to PAA(40-63) and aas hot 7.
- Shift sum right 4.
- Word overlap if PAL(40-64) = 0.
- Gate SDB0(0-63) B to ST.
- Issue 3-cycle storage request per IC for 2nd operand.

**B**
- Transfer (021-23) to PAB.
- Transfer IC to PAA(40-63).
- Transfer D to PAA(40-63) and complement.
- Shift sum right 4.
- Word overlap if PAL(40-64) = 0.
- Transfer IC to ABC.

**C**
- Enable I-Fetch · Tests validity invalid address of 2nd operand test circuits.
- Set appropriate Inhibited when Edit or interruption code (5 or 6) Edit and Mark and 'program interrupt' instruction is being performed.
- Branch to execution sequence per E(4-7).

**D**
- Branch to execution sequence, word overlap, byte overlap.

**E**
- Gate S90D to AR.
- Branch to execution sequence per E(4-7).

**F**
- Diagram 5-401. GIS for Logical Instructions

---

**G**

**H**
A. Move, MVI (92)

B. Move, MVC (02)

C. Move Numerics, MVN (D1)

D. Move Zones, MVZ (03)

Diagram 5-402. Logical Move Instructions

Diagram 5-403. Logical Compare Instructions
A. AND, NR (14)

**Diagram 5-404. Logical AND Instructions**

- **RR format.**
  - Purpose: AND 1st operand (in GPR, per R1) with 2nd operand (in GPR, per R2) and place result into 1st operand location.

- **RX format.**
  - Purpose: AND 1st operand (in GPR, per R1) with 2nd operand (in main storage) and place result into 1st operand location.

- **SI format.**
  - Purpose: AND immediate operand (12 of instruction) with 1st operand (in GPR) and place result into 1st operand location.

B. AND, NI (54)

- **RR format.**
  - Purpose: AND 1st operand (in GPR, per R1) with 2nd operand (in GPR, per R2) and place result into 1st operand location.

- **RX format.**
  - Purpose: AND 1st operand (in GPR, per R1) with 2nd operand (in storage) and place result into 1st operand location.

- **SI format.**
  - Purpose: AND immediate operand (12 of instruction) with 1st operand (in storage) and place result into 1st operand location.

C. AND, NC (D4)

- **RR format.**
  - Purpose: OR 1st operand (in GPR, per R1) with 2nd operand (in GPR, per R2) and place result into 1st operand location.

- **RX format.**
  - Purpose: OR 1st operand (in GPR, per R1) with 2nd operand (in storage) and place result into 1st operand location.

- **SI format.**
  - Purpose: OR immediate operand (12 of instruction) with 1st operand (in storage) and place result into 1st operand location.

D. OR, OR (16)

- **RR format.**
  - Purpose: OR 1st operand (in GPR, per R1) with 2nd operand (in GPR, per R2) and place result into 1st operand location.

- **RX format.**
  - Purpose: OR 1st operand (in GPR, per R1) with 2nd operand (in storage) and place result into 1st operand location.

- **SI format.**
  - Purpose: OR immediate operand (12 of instruction) with 1st operand (in storage) and place result into 1st operand location.

E. OR, OC (D6)

- **RR format.**
  - Purpose: OR 1st operand (in GPR, per R1) with 2nd operand (in GPR, per R2) and place result into 1st operand location.

- **RX format.**
  - Purpose: OR 1st operand (in GPR, per R1) with 2nd operand (in storage) and place result into 1st operand location.

- **SI format.**
  - Purpose: OR immediate operand (12 of instruction) with 1st operand (in storage) and place result into 1st operand location.

F. OR, OC (D6)

- **RR format.**
  - Purpose: OR 1st operand (in GPR, per R1) with 2nd operand (in GPR, per R2) and place result into 1st operand location.

- **RX format.**
  - Purpose: OR 1st operand (in GPR, per R1) with 2nd operand (in storage) and place result into 1st operand location.

- **SI format.**
  - Purpose: OR immediate operand (12 of instruction) with 1st operand (in storage) and place result into 1st operand location.
### Diagram 5-406. Logical Exclusive-OR Instructions

#### A. Exclusive-OR, XE (17)

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose:** Exclusive-OR 1st operand from GPR per R1, 2nd operand from GPR per R2, and place result into 1st operand location.

- **Result:** Store result into GPR per R1 and set CC.

#### B. Exclusive-OR, XE (57)

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose:** Exclusive-OR 1st operand (in GPR, per R1) with 2nd operand (in GPR, per R2) and place result into 1st operand location.

- **Result:** Store result into GPR per R1 and set CC.

#### C. Exclusive-OR, XI (97)

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose:** Exclusive-OR immediate operand (12 of instruction) with 1st operand (in storage) and place result into 1st operand location.

- **Result:** Store result into GPR per R1 and set CC.

#### D. Exclusive-OR, XC (67)

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose:** Insert 2nd operand (byte; in storage) into bits 24-31 of 1st operand location (in GPR, per R1).

- **Result:** Store character into main storage per 2nd operand address.

### Diagram 5-407. Test Under Mask, TM (91)

#### E. Test Under Mask, TM (91)

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose:** Test CC according to state of 1st operand bits (in storage) selected by mask bits (12 of instruction).

- **Result:** Set CC.

### Diagram 5-408. Insert Character, IC (43); Store Character, STC (42)

#### A. Insert Character, IC (43)

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose:** Insert character from main storage into bits 24-31 of GPR per R1.

- **Result:** Insert character into bits 24-31 of GPR per R1.

#### B. Store Character, STC (42)

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose:** Store character into main storage per 2nd operand address.

- **Result:** Store character into main storage per 2nd operand address.

### Diagram 5-409. Load Address, LA (41)

#### G. Load Address, LA (41)

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Purpose:** Insert 2nd operand address into bits 8-31 of GPR per R1.

- **Result:** Insert 2nd operand address into bits 8-31 of GPR per R1.
Fetch doubleword (containing 1st argument byte) from main storage per 1st operand address.

Select argument byte.

Add argument byte to base address of function byte (2nd operand address).

Fetch function byte per result address.

Store function byte into argument byte location.

A. Translate, TR (DC)

- SS format.
- Purpose: Add 1st operand byte (argument, in storage) to 2nd operand address, use result as storage address, and place function byte from resulting storage address into corresponding 1st operand byte location.

Diagram 5-410. Translate, TR (DC); Translate and Test, TRT (DD)

B. Translate and Test, TRT (DD)

- SS format.
- Purpose: Add 1st operand byte (argument, in storage) to 2nd operand address, use result as storage address, and test function byte from resulting storage address. If zero, translate and test next argument byte. If nonzero, complete operation by inserting related argument address into GPR1 and function byte into GPR2.

Fetch doubleword (containing 1st argument byte) from main storage per 1st operand address.

Select argument byte.

Add argument byte to base address of function byte (2nd operand address).

Fetch function byte per result address.

Yes

No

Function byte all '0'?

Yes

Translate next argument byte.

Store function byte into GPR2.

Store address of argument byte into GPR1.

End op; set CC.

End op.
Diagram 5-411. Edit, ED (DE); Edit and Mark, EDMK (DF)

### SS format:

<table>
<thead>
<tr>
<th>Op Code</th>
<th>LL</th>
<th>B1</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
</tr>
</thead>
</table>

### Purpose:

1. Edit always starts from source (2nd operand) in storage. First pass is counts on errors. On second pass to correct errors, all source under control of pattern (1st operand) is scanned. End point must have last significant digit location.

2. Edit and Mark come on Edt, but, in addition, place location of each 1st significant digit into GPR's.

### Conditions at end of GIS:

1. Pattern (1st operand) is in ST.
2. Pattern address is in D.
3. Source address (contents of GPR per ED, + D D is in IC.
4. Set 16 bits of instruction in E.

#### Diagram 5-411

- **Digit Select (20) or Significant Start (21)**
  - Examine source digit and gate AB byte to SAB
  - Store D contents into GP!!.

- **Significant Separator (22)**
  - Set 'right digit' trigger to examine right digit.

- **Source Trigger (23)**
  - Source digit set.

- **Significant Start (21)**
  - Source digit set.

- **Reset 'S' Trigger and STAT A**
  - Reset 'S' trigger and STAT A.

- **Digit (20)**
  - Digit Select (20) or Significant Start (21)

- **Store ST per D address**
  - Source digit set.

- **CC setting**
  - Source digit set.

- **Significant Separator (22)**
  - Increase STC and D by 1.
  - Reduce LL by 1.

- **Field Separator (20)**
  - Other Character

- **APP Prop code**
  - LL = 0
  - STC = 0

- **Other Character**
  - Increase IC by 8 and fetch new source to AB.

- **Insert fill character into ST**
  - Insert fill character into ST.

- **Set Mark Trigger per STC**
  - Set mark trigger per STC.

- **Other Character**
  - Other Character

- **Significant Start (21)**
  - Significant Start (21)

- **Set 'S' Trigger and STAT A**
  - Reset 'S' Trigger and STAT A.

- **Significant Start (21)**
  - Significant Start (21)

- **Store ST per D address**
  - Source digit set.

- **CC setting**
  - Source digit set.

- **Significant Separator (22)**
  - Increase STC and D by 1.
  - Reduce LL by 1.

- **Field Separator (20)**
  - Other Character

- **APP Prop code**
  - LL = 0
  - STC = 0

- **Other Character**
  - Other Character

- **Significant Start (21)**
  - Significant Start (21)

- **Set 'S' Trigger and STAT A**
  - Reset 'S' Trigger and STAT A.

- **Significant Start (21)**
  - Significant Start (21)

- **Store ST per D address**
  - Source digit set.

- **CC setting**
  - Source digit set.

- **Significant Separator (22)**
  - Increase STC and D by 1.
  - Reduce LL by 1.

- **Field Separator (20)**
  - Other Character

- **APP Prop code**
  - LL = 0
  - STC = 0

- **Other Character**
  - Other Character

- **Significant Start (21)**
  - Significant Start (21)

- **Set 'S' Trigger and STAT A**
  - Reset 'S' Trigger and STAT A.

- **Significant Start (21)**
  - Significant Start (21)

- **Store ST per D address**
  - Source digit set.

- **CC setting**
  - Source digit set.

- **Significant Separator (22)**
  - Increase STC and D by 1.
  - Reduce LL by 1.

- **Field Separator (20)**
  - Other Character

- **APP Prop code**
  - LL = 0
  - STC = 0

- **Other Character**
  - Other Character

- **Significant Start (21)**
  - Significant Start (21)

- **Set 'S' Trigger and STAT A**
  - Reset 'S' Trigger and STAT A.

- **Significant Start (21)**
  - Significant Start (21)
Diagram 5-412. Logical Shift Instructions
Diagram 5-501. Branch On Condition, BCR (07); BC (47) (Sheet 1 of 2)
Diagram 5-501. Branch On Condition, BCR (07); BC (47) (Sheet 2 of 2)
Diagram 5-502, Branch and Link, BALR (05) (Sheet 1 of 2)

1. Link information stored per R1.
2. Link information stored per R1.
4. Determine address of instruction (link address) following BALR instruction.
5. Normal end op.
6. Next instruction to be executed is the instruction that sequentially follows the BRANCH instruction.

Purpose: Store PSW(32-63), link information, into GPR (addressed by R1) and branch to location specified by GPR (addressed by R2).

- Store link data into LS, add 8 to D and gate SDBO to Q.
- Transfer D to IC and generate 3-cycle storage request per IC.
- Refill Q with a new doubleword to allow continuous operation.
- Update IC.
- Gate SDBO contains the branch-to instruction requested during I-Fetch of the branch instruction.

If the 'execute' trigger is set, the link address will be the address that follows the branch instruction.

Determine value of IC(21,22): If equal to 11, gate SDBO to Q. If not equal to 11, gate SDBO to Q and first halfword is transferred to R.

Next instruction to be executed is the instruction that sequentially follows the branch instruction.

- Refill Q with a new doubleword to allow continuous operation.
- Update the IC.
- Gate SDBO contains the branch-to instruction requested during I-Fetch of the branch instruction.

Normal end op.

Branch and op.
Diagram 5-502. Branch and Link BALR (05) (Sheet 2 of 2)

Objectives:
1. Determine address to be stored as link address.

G001
- QE031
- QE041

Add to PAA and transfer result to IC.

Set "branch invalid" trigger.

Transfer IC to PA(H4)-60.

Set "execution" trigger.

Branch end op.

Objectives:
2. Determine value of IC(21,22).

G002
- QE01
- QE02

Update IC for next sequential doubleword.

Transfer IC value to D.

Update IC for next sequential doubleword.

Transfer D to PA(H4)-60.

No QE02
Yes QE02

Transfer IC to PA(H4)-60.

Yes QE01
No QE01

No QE02
Yes QE02

Objectives:
1. Determine value of IC(21,22).

G003
- QE03
- QE04

Transfer IC to PA(H4)-60.

Transfer IC to GPR.

Gate SDB0(0-63)

Transfer Q to R.

Transfers link information into LS.

Transfers link information into LS.

No QE03
Yes QE03

No QE04
Yes QE04

Objectives:
1. Store link data into LS.

G011
- QE05
- QE06

No QE05
Yes QE05

Yes QE06
No QE06

Objective:
Store PSW(32-39), link information, into GPR, address, by GPR (addressed by R2).

Objectives:
1. 2nd operand is in A, B, and D.
2. 3-cycle storage request has been issued per D for branch-to instruction.
3. Instruction is in E.

Conditions at start of execution:
1. 2nd operand is in A, B, and D.

Purpose: Store PSW(32-63), link information, into GPR, address, by GPR (addressed by R2).

Objectives:
1. Store link data into LS.
2. Obtain next sequential doubleword.

Branch end op.

Objectives:
1. Update IC.
2. Transfer IC value to D.

Branch end op.
Diagram 5-503. Branch and Link, BAL (45) (Sheet 1 of 2)

A

Diagram 5-9

RX I-Fetch.

Determine address of instruction (link address) following BAL instruction.

B

Yes

If set, indicates that the branch instruction is the subject instruction of an Execute instruction.

No

'C' trigger set?

C

Store link data into LS, calculate address of next sequential doubleword following the Execute instruction, refill Q, and transfer Q to R per DST1,320.

If yes, branch instruction being executed was located in 3rd and 4th halfwords of Q.

If yes, branch instruction being executed was located in 3rd and 4th halfwords of Q.

If no, branch instruction being executed was located in 3rd and 4th halfwords of Q.

D

Update IC, refill Q.

Update IC, refill Q.

Branch and op.

Normal end op.

E

Diagram 5-503. Branch and Link, BAL (45) (Sheet 1 of 2)

F

G

H

- RX format:

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>X2</th>
<th>X3</th>
<th>D2</th>
<th>D1</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>R1</td>
<td>X2</td>
<td>X3</td>
<td>D2</td>
<td>D1</td>
</tr>
</tbody>
</table>

- Purpose: Store PSW(32-63), link information, link GPR (addressed by X3) and branch to location specified by 3rd operand address.

- Link information stored per RI:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>Prog</td>
<td>Mask</td>
<td>Link Address</td>
<td></td>
</tr>
</tbody>
</table>

- Instruction format to be executed next is decoded off the SDBO, and Instruction address is decoded from D.
Diagram 5-504. Branch On Count, BCTR (06); BCT (46) (Sheet 1 of 2)

- **RR format**: BCTR
  - Purpose: Subtract from 1st operand (in GPR, per R1) and, if result $\neq 0$, branch to location specified by GPR (addressed by R2).

- **RX format**: BCT
  - Purpose: Subtract from 1st operand (in GPR, per R1) and, if result $\neq 0$, branch to location specified by 2nd operand address.

---

**A**

Diagram 5-6

**B**

Determine if branch is successful.

**C**

Yes (Unsuccessful Branch)

- Obtain next sequential doubleword from main storage per IC.
  - If yes, branch instruction is the subject instruction of an Execute Instruction.
  - Next Instruction is located in last halfword of the doubleword requested during I-Fetch phase of branch instruction.

**D**

Yes (Successful Branch)

- Update IC and transfer result to D.

**E**

- Refill Q and transfer correct halfword to R.
  - Data requested during I-Fetch is present at the SDB0.
  - Normal end op.

**F**

- Data requested by the storage request given during the execution phase of the branch instruction is present at the SDB0 and is gated into D.
  - Transfer into Q doubleword which sequentially follows the branch-to instruction.

**G**

**H**

Diagram 5-9

**I**

Update IC, refill Q, and transfer result to D per C(21, 22).

---

2

3

4

5

6

9

10
Diagram 5-104. Branch On Count, BCTR (06); BCT (46) (Sheet 2 of 2)
Diagram 5-505. Branch on Index High, BXH (86); Branch on Index Low or Equal, BXLE (87) (Sheet 1 of 3)
Diagram 5-13

**Objectives:**
1. Transfer index to GPR per E(8-11).
2. Refill Q.
3. Transfer Q to R per 0(21,22).

**Conditions at start of execution:**
1. 1st operand is in S and T.
2. Branch address is in 0.
3. 3-cycle storage request has been issued per 0 for the branch-to instruction.
4. 1st 16 bits of instruction are in R.

**Purpose:**
1. **BXH:** Add increment (3rd operand; in GPR, per R3) to 1st operand (in GPR, per R1), algebraically compare result (index) with comparand (in odd-address GPR specified by R3 or R3+1), and, if index is greater than comparand, branch to location specified by 2nd operand address.
2. **BXLE:** Add increment (3rd operand; in GPR, per R3) to 1st operand (in GPR, per R1), algebraically compare result (index) with comparand (in odd-address GPR specified by R3 or R3+1), and, if index is equal to or less than comparand, branch to location specified by 2nd operand address.

**RS format:**

- **Purpose:**
  1. BXH-Add increment (3rd operand) to 1st operand (in GPR, per R1), algebraically compare result (index) with comparand (in odd-address GPR specified by R3 or R3+1), and, if index is greater than comparand, branch to location specified by 2nd operand address.
  2. BXLE-Add increment (3rd operand) to 1st operand (in GPR, per R1), algebraically compare result (index) with comparand (in odd-address GPR specified by R3 or R3+1), and, if index is equal to or less than comparand, branch to location specified by 2nd operand address.
Diagram 5-22
Program interruption.

Prepare to modify subject instruction and to update D.

Determine format of subject instruction.

Updating of D allows selection of doubleword following subject instruction of Execute instruction.

PAL(61-63) is used by the ABC to select correct byte of subject instruction for modification.

If yes, subject instruction is in last halfword. If no, in some other halfword.

If yes (No Modification) No (Modify Instruction)

Determines if subject instruction is to be modified.

No (Not RR Format) No (Modify Instruction)

Transfer lost AB byte to T per ABC and STC, generate 3-cycle storage request per D, and refill Q.

Yes (RR Format)

Modify subject instruction, generate 3-cycle storage request per D, and set 'PSC' trigger.

No (Not RR Format)

Refill Q and AB, and transfer Q to R per D(21, 22), and add 1 to ABC.

Setting of the 'PSC' trigger and STAT G ensures the re-insertion of the original Q-data back into Q after completion of the subject instruction.

Issue a 4-cycle D request if the subject instruction is in the SS format Execute Instruction.
**Diagram 5-22: Program Interruption**

**Diagram 5-9: RX I-Fetch**

1. Prepare subject instruction for modification.
2. Update D.

**Objective:**

1. Prepare subject instruction for modification.
2. Update D.

**Conditions at start of execution:**

1. First operand is in S and T.
2. Address of subject instruction is in D.
3. 3-cycle storage request for subject instruction has been issued per D.
4. 1st 16 bits of instruction are in E.

**RX Format:**

- Yes (RR Format, No Modification)
- No (Not RR Format, Modify Instruction)

**Diagram 5-26: Execute, EX (44) (Sheet 2 of 2)**

1. Transfer last byte to R per ABC.
2. Generate 3-cycle storage request per D.
3. Set 'PSC' trigger.
4. Issue 3-cycle storage request per D.
5. Refill Q.

**Objective:**

1. Transfer last byte to R per ABC.
2. Generate 3-cycle storage request per D.
3. Set 'PSC' trigger.
Diagram 5-17: External Event Timer Exceptional Condition.


Diagram 5-26: Common interruption routine.

Diagram 5-36: Program store compare exception.

Diagram 5-38: Program store compare exception.

Diagram 5-41: Operation from this point on is the same as for the Branch On Count instruction following a successful branch.

Conditions at start of execution:
1. 1st 16 bits of instruction are in E.
2. Storage address is in D.
3. Storage request has been issued per D.

New PSW is fetched from main storage address 0 when PSW RESTART or LOAD pushbutton is depressed.

Invalid instruction address exceptional condition handled during next IFetch.

Interrupts at high level address exceptional condition handled during next IFetch.

Reset 'I-Fetch request' trigger.

Gate SDBO to Q.

Load doubleword storage operand (designated by storage operand address) into CE, thus replacing current PSW, and branch to new instruction sequence.

Conditions at start of execution:
1. 1st 16 bits of instruction are in E.
2. Storage address is in D.
3. Storage request has been issued per D.

New PSW is fetched from main storage address 0 when PSW RESTART or LOAD pushbutton is depressed.

Invalid instruction address exceptional condition handled during next IFetch.

Interrupts at high level address exceptional condition handled during next IFetch.

Reset 'I-Fetch request' trigger.

Gate SDBO to Q.

Load doubleword storage operand (designated by storage operand address) into CE, thus replacing current PSW, and branch to new instruction sequence.

Conditions at start of execution:
1. 1st 16 bits of instruction are in E.
2. Storage address is in D.
3. Storage request has been issued per D.

New PSW is fetched from main storage address 0 when PSW RESTART or LOAD pushbutton is depressed.

Invalid instruction address exceptional condition handled during next IFetch.

Interrupts at high level address exceptional condition handled during next IFetch.

Reset 'I-Fetch request' trigger.

Gate SDBO to Q.

Load doubleword storage operand (designated by storage operand address) into CE, thus replacing current PSW, and branch to new instruction sequence.

Conditions at start of execution:
1. 1st 16 bits of instruction are in E.
2. Storage address is in D.
3. Storage request has been issued per D.

New PSW is fetched from main storage address 0 when PSW RESTART or LOAD pushbutton is depressed.

Invalid instruction address exceptional condition handled during next IFetch.

Interrupts at high level address exceptional condition handled during next IFetch.

Reset 'I-Fetch request' trigger.

Gate SDBO to Q.

Load doubleword storage operand (designated by storage operand address) into CE, thus replacing current PSW, and branch to new instruction sequence.

Conditions at start of execution:
1. 1st 16 bits of instruction are in E.
2. Storage address is in D.
3. Storage request has been issued per D.

New PSW is fetched from main storage address 0 when PSW RESTART or LOAD pushbutton is depressed.

Invalid instruction address exceptional condition handled during next IFetch.

Interrupts at high level address exceptional condition handled during next IFetch.

Reset 'I-Fetch request' trigger.

Gate SDBO to Q.

Load doubleword storage operand (designated by storage operand address) into CE, thus replacing current PSW, and branch to new instruction sequence.

Conditions at start of execution:
1. 1st 16 bits of instruction are in E.
2. Storage address is in D.
3. Storage request has been issued per D.

New PSW is fetched from main storage address 0 when PSW RESTART or LOAD pushbutton is depressed.

Invalid instruction address exceptional condition handled during next IFetch.

Interrupts at high level address exceptional condition handled during next IFetch.

Reset 'I-Fetch request' trigger.

Gate SDBO to Q.

Load doubleword storage operand (designated by storage operand address) into CE, thus replacing current PSW, and branch to new instruction sequence.

Conditions at start of execution:
1. 1st 16 bits of instruction are in E.
2. Storage address is in D.
3. Storage request has been issued per D.

New PSW is fetched from main storage address 0 when PSW RESTART or LOAD pushbutton is depressed.

Invalid instruction address exceptional condition handled during next IFetch.

Interrupts at high level address exceptional condition handled during next IFetch.

Reset 'I-Fetch request' trigger.

Gate SDBO to Q.
Diagram 5-6

Set Program Mask, SPM (04)

Replace CC and program mask (bits 34-39) of current PSW with bits 2-7 of 1st operand (in GPR; per R1).

Conditions at start of execution:
1. Instruction is in E.
2. 1st operand is in A, B, and D.
3. 2nd operand is not used.

Bits 2-7 of 1st operand may have been loaded into PSW-register by a previous Branch and Link instruction.

Program mask format (set mask bit permits interruption):
- Bit 36 = Fixed-point overflow mask.
- Bit 37 = Decimal overflow mask.
- Bit 38 = Exponent underflow (floating-point) mask.
- Bit 39 = Significance (floating-point) mask.

RR Format:

Condition:

Transfer T034-39 to PSW-register.

Sets new CC and program mask into current PSW.

Issue early end op 2 cycles early.

Transfer B to T and D via parallel inverter.

Diagram 5-602.
Diagram 5-13

SI - Fetch

Diagram 5-22

Program Interruption

Gate SDBO to AB.

Gate first byte of new system mask from AB (thru serial adder) into ST (0-7).

Gate S (0-7) to PSW (0-7) and 5(16-19) to PSW (16-19).

End Op

Diagram 5-603. Set System Mask, SSM (80)

<table>
<thead>
<tr>
<th>System mask</th>
<th>Key</th>
<th>AAPDP</th>
<th>System mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>8</td>
<td>16-19</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>19</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>23</td>
<td></td>
</tr>
</tbody>
</table>

Gate second byte of new system mask from AB (thru serial adder) into ST (16-23).

ABC will gate new system mask from input doubleword.

Gate SDBO to AB.

Bits 16-19 of this byte contain PSW data. Bits 20-23 are ignored.

Gate SDBO to AB.

Bits 16-19 of this byte contain PSW data. Bits 20-23 are ignored.

Diagrams 5-13 and 5-22 are used for the process of setting a system mask (SSM). The process involves fetching the mask value into a register (SI), checking if the process is interrupted, and then setting the system mask by gating the mask into the PSW register. The system mask format is defined and conditions at the start of execution are specified. The purpose of the SSM instruction is to replace the system mask (bits 0-7 and 16-19) of the current PSW with a new system mask obtained from a storage operand address specified by the storage operand address register (D). The new system mask is obtained from a doubleword of storage, and the request for fetching the next doubleword is issued as necessary.
Set "supervisor call" trigger.

Requests supervisor call interruption.

Issue early end-op 1 cycle before normal end-op.

Supervisor call interruption.

RR Format:

<table>
<thead>
<tr>
<th>Ca</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Causes supervisor call interruption; replace old-PSW(24-31) with I-field (bits 8-15) of instruction, providing interruption code.

Conditions at start of execution:
1. Instruction is in E.
2. EB(15) contains interruption code.
Diagram 5-6

RR I-Fetch.

1. Initialize.
   - Set key into even address.
   - Decrement ABC and STC.

   - ABC block key in byte 7.
   - STC is set to 3.

2. ABC selects key in byte 7; STC is set to 3.

3. Transfer S to A via parallel adder. Transfer A minus 8 to D via parallel adder.

   - Allows construction of STAT D loop.

4. Privileged operation test.
   - Yes (Problem State)
   - Specification test.
   - No
   - (2IB-31)

   - Set 'set key' and mark 0-7 triggers; set STAT D; gate F(0-4) to 'key in' bus.

5. Transfer byte 7 of AB per ABC to F via serial adder.

   - Places new key into F.

6. Add 8 to D via parallel adder.

   - Leave D-cycle storage request per D.

   - Set 'set key' and mark 0-7 triggers set STAT D; gate (0-6) to 'key in' bus.

   - Set 'PSC' trigger.

   - Process Q until the new protection key of next instruction(s) may be changed.

7. Set key into odd address.

   - Repetition of above operation is made necessary by the default interface maintenance aid.

8. Test if key has been set for the last time.

   - Normal endings.

   - Yes
   - STAT D set

   - No
   - Set STAT D.

Diagram 5-605. Set Storage Key, SSK (08)
Fetch key from even address.

Add 8 to D via parallel adder, and issue 3-cycle storage request per D.

Set 'insert key' trigger and STAT D:

Gate 'key out' bus to FIP(0-4), and restart CE clock.

Test if last fetch was made.

Diagram 5-32. Program interruption.

Diagram 5-606. Insert Storage Key, ISK (09)
Diagram 5-607. Write Direct, WRD (84)
Diagram 5-13. SI i-Fetch

**Objectives:**
Gate data from direct control bus via F to Bl+Dl location in storage.

**Diagram:**

- **SI Format**
  - **Objectives:** Gates a data byte from the control bus to the byte location (specified by the Bl, Dl fields) in storage.
  - **Conditions at start of execution:**
    1. 1st 16 bits of instruction are in E.
    2. Storage operand address is in D.
    3. Storage request has been issued per D.

- **SI Format**
  - **Objectives:**
  - **Conditions at start of execution:**
    1. 1st 16 bits of instruction are in E.
    2. Storage operand address is in D.
    3. Storage request has been issued per D.
Determine if this is an operational kernel or a diagnostic kernel.

Load FF (Hex) in F.

F - 1 - F
(FD hex in F).

Gate - SDBO (0-31) to S and SDBO (0-63) to AB.

Diagram 5-13.

SJ I-Fetch

Objectives:

Diagram 5-22

Program Interrupts.

Set scan mode. Gate SDBO (0-31) to S and SDBO (0-63) to AB.

Exclusive or Contents of F and Contents of S (byte 1)

FD ¥ FD = 0

Set scan mode.

F = 1 = F
(FD hex in F).

Diagram 5-32

Program Interrupts.

F = 1 = F
(FF hex in F).

Set scan mode. Gate SDBO (0-31) to S and SDBO (0-63) to AB.

Exclusive or Contents of F and Contents of S (byte 1)

FD ¥ FD = 0

Set scan mode. Gate SDBO (0-31) to S and SDBO (0-63) to AB.

Diagram 5-22

Set scan mode. Gate T and B to MCW, 12 to control triggers.

Timing Blocks Reset of Scan Counter Control Trigger and Latch if MCW (6) = 1.

Hardware Blocks Reset of Scan Counter Control Trigger and Latch if MCW (6) = 1.

Sheet 2

Sheet 3

Diagram 5-609. Diagnose (B3) (Sheet 1 of 3)
Table 1

<table>
<thead>
<tr>
<th>1 Reg</th>
<th>STAT</th>
<th>STAT</th>
<th>STAT</th>
<th>Register, Logoff, Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-23</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off, F1, Abort</td>
</tr>
<tr>
<td>00-15</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Gen Purpose</td>
</tr>
<tr>
<td>01</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>CS1</td>
</tr>
<tr>
<td>01</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>DAR Hold</td>
</tr>
<tr>
<td>01</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>End Req</td>
</tr>
<tr>
<td>01</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>Check Req</td>
</tr>
<tr>
<td>01</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>DAR 1</td>
</tr>
<tr>
<td>00</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>DAR 2</td>
</tr>
<tr>
<td>00</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>DAR</td>
</tr>
</tbody>
</table>

E3 = 0 and STAT H off mean layout complete.

Branch on ROS Address (See note 4)

Table J

<table>
<thead>
<tr>
<th>Req H</th>
<th>Req L</th>
<th>STAT</th>
<th>STAT</th>
<th>STAT</th>
<th>STAT</th>
<th>Register, Logoff, Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-15</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Flt, Adu.</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>CSA1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>DAR Hold</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>End Req</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Check Req</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>DAR 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>DAR 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>DAR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram 5-609. Diagnose (83) (Sheet 2 of 3)
If MCW (6) is set, then hardware decrements the cycle counter by 1 (one) for each ROS word executed. When the cycle counter equals 0, SCROS trigger is turned on and a logout is taken.

Notes:
1. The Diagnose instruction may branch to any ROS address; only the most frequently used are shown here.
2. Log-on-count bit of the CE's MCW may be used to branch from this loop.
3. ROS word 000 purposely contains even (incorrect) parity. Therefore, use of Diagnose instruction without specifying ROS address in the fetched MCW results in a machine check interruption.

Diagram 5-609. Diagnose (Sheet 3 of 3)
Diagram 5-701. I/O Instructions

A

Objectives:
1. Assembles data required for IOCE to execute the I/O instructions into a one-word format and gates it to the proper IOCE via the control bus.

B

1. Operand address (address of channels and I/O unit) is in D.

D

1. Operand address (address of channels and I/O unit) is in D.

3

1. Operand address (address of channels and I/O unit) is in D.

E

1. Operand address (address of channels and I/O unit) is in D.

F

1. Operand address (address of channels and I/O unit) is in D.

G

1. Operand address (address of channels and I/O unit) is in D.

H

1. Operand address (address of channels and I/O unit) is in D.
Objectives:
- Gate identity to F(4-7).
- Transfer identity to T.
- Store identity in a GPR (28-31).

Diagram 5-801. Load Identity, LI (0C)

RR Format: LI

Objectives:
Gate ATR2 to T (0-07).
Store ATR2 in GPR specified by R2 field.

Gate E (8-15) to A (24-31) to save R1 value.

Shift E(8-15) left 4 to move R2 to R1 position.

Store ATR2 in local store R2 in E (8-15).

Objectives:
Gate ATR1 to T(32-63).
Store ATR1 in GPR specified by R1.

Restore R1 field in E (8-15).

Store ATR1 in local store per R1 in E (8-15).


Diagram 5-9. Insert ATR, IATR (OE)
Diagram 5-803. Delay, DLY (OB)

- **Purpose:** Provides a variable delay (256 usec x N) dependent on the value of N.
- **Conditions at the beginning of execution:** Instruction and N are in E.

**Diagram 5-803.**

- **Step-1 and Step-2 micro-orders stop CE clock for 5 cycles.**
- **Delay 1 usec**
- **Add 1 to F register.**
- **Any pending interrupt**
- **F value equals 256**
- **Subtract 1 from EB-13.**

- **Save count in K register.**
- **Turn on STAT A.**
- **Execute timer update microprogram.**
- **Abort Timer update microprogram.**
- **Normal timer update microprogram End Op.**

Objectives:  
Combine logical and physical PSBAR in S and T.  
Store S or T in main storage word location specified by 1st operand address.

Gate logical and physical PSBAR into S(9-19) and Zeros T(41-51), and S(28-31) and T(60-63) respectively.

Specification error occurs if:  
1. The operand address is not on a word boundary.

Store PSBAR's in main storage per D (operand address).

Make address store compare test.

Diagram 5-13
SI-I-Fetch.

Objectives:
- Check for specification errors.
- Load new PSBA to T.
- Correct T (byte 6).

Yes

Main storage request
per D(new PSBA to T).

Set T(52-55) to zeros (possible extraneous
bits), and correct
byte 6 parity.

No

Objectives:
- Check that new PSBA is
a valid address.
- Load logical and physical
PSBAR.

Load logical PSBAR
from ATR position.

8th h-12 of logical PSBAR select
physical PSBAR from appropriate
ATR slot.

End Op.,

Diagram 5-805. Load PSBAR, LPSB (A1)
Objectives:
- Make word overlap test.
- Make address boundary test.

QQ071

Save next instruction address in K.

Request 1st doubleword of source (B2, D2) per LC.

Gate the 3 low-order bits of destination address to STC.

Update the source address by S-.

Note:
- Normally, SS instructions save the instruction address in LSWR. However, the MVW instruction may require use of LSWR for operand alignment.

SS Format

<table>
<thead>
<tr>
<th>DS</th>
<th>L</th>
<th>B1</th>
<th>S1</th>
<th>B2</th>
<th>S2</th>
<th>D2</th>
</tr>
</thead>
</table>

| Purpose: Moves up to 256 words from one storage location (B2, D2) to another storage location (B1, D1). The number of words to be moved is specified by the L-field. |
| Conditions at the beginning of execution: |
| 1. Source address is in IC and T. |
| 2. Destination address is in D. |
| 3. Updated IC is in LSWR. |

Word Overlap Test:
Word Overlap exists if starting address of destination field is one or two words higher than starting address of source field. This condition propagates the first source word (or doubleword) through storage for the number of words specified in the L-field. Once the word or doubleword of source data has been fetched by the CPU, further fetches are not necessary, and repeated stores can be made per the destination address. Because both fields must start on a word or doubleword boundary, four possible word overlap conditions can occur. These are:

Overlap conditions when:

1. Source is on a doubleword boundary, destination is on a word boundary.
2. Source is on a word boundary, destination is on a doubleword boundary.
3. Source is on a doubleword boundary, destination is on a doubleword boundary.
4. Source is on a word boundary, destination is on a word boundary.

Note: If the source and destination addresses differ by one word, that word is propagated throughout the destination field. If the difference is a doubleword, that doubleword is propagated.
Diagram 5-806. Move Word, MVW (D8) (Sheet 2 of 3)
Objectives:
Gate source word 1 to S and T. Store 1 word if the count field is zero. Check for source/destination field overlap, which will allow high-speed moving of words. Store source word 1 to align destination field to doubleword boundary.

Gate S to T. (S and T now contain source word 1.)

Yes

Set marks 0-7 and store T.

No

Update destination address +8.

Make a storage request per IC for source data.

Diagram 5-806. Move Word, MVW (D4) (Sheet 3 of 3)
Objectives:
Align key and address in external register.
(Key in bits 0-3 and address in bits 4-27.)

Gate key from E(S-11) to B(60-63).

Gate D(0-23) to T(36-59).

Gate B(60-63) to T(32-35).

Gate T to external register.

Objectives:
Gate IOCE select bits to select register.

Issue select to IOCE.

Gate E (12-15) to T (60-63).

Gate T to select register.

Turn on STAT B and "Timing gate" trigger.

Objectives:
Wait for condition code to return from IOCE.

Set timeout constant in B.

Decrement B by 1.

Response from IOCE

Gate B to PAL to check for timeout.

Set condition code to value received from IOCE.

Turn off "Timing gate" trigger.

Diagram 5-6

RR Fetch

Objectives:
Check select mask for 0's.
Check ATR slots 1-5 for valid SE frame specified.

Gate ATR word 2 (0) to ATR
(slots 9 and 10).

Gate select mask from 24-31 to SAL.

Yes (No elements selected)
Set condition code to 3.


Gate ATR slot (per ABC) from A to F
for validity check.

Gate ATR slot from F to decode
and checking logic.

Valid SE specified

Yes

Reset STAT C
and set 'Hold-
Invalid' latch.

No

Yes (Slots 1-5 checked)

STC = 7

STC = 4 or 6

Yes

Step STC = 1.

No

Step STC and ABC + 1

Diagram 5-808. Set Address Translator, SATR (6D), Execution in Issuing CE (Sheet 1 of 6)

Diagram 5-808. Sheet 1 (7/10)
Objectives:
Check ATR slots 6-10 for a valid frame specified.
Check for specification error.
Gate zeros (working register) to S.
Set STC to 3. (set up, for counting slots 6-10)
Turn on STAT A (to check slots 6-10 for DE's specified).
Gate ATR slot (per ABC) from A (B for slots 9 and 10) to F.
Gate ATR slot from F to complete and checking circuits.
Valid frame specified
Yes
No
This latch prevents setting of STAT C after an invalid condition is detected.
Yes
No
Set STAT C...

Step STC = 7

Step STC = 4 or 5

Set ABC to 7.

Test specification error.

Diagram 5-808. Set Address Translator, SATR (OD), Execution in Issuing CF (Sheet 2 of 6)
Objectives:
Gate CE and IOCE select bits into proper select register bit positions.
Issue 1st SATR select to elements.

Set ABC to 7 and STC to 3.

Gate CE select bits to F(4-7).
Gate IOCE select bits to S(29-31).

Decrement STC to 2.

Gate CE select to S(20-23).
Gate S to select register (via T).

ABC and STC are serial adder bus gates for A, B, S, and T registers.

Select Register Bit Position Assignments

Note:
Only the CE (bits 20-23) and IOCE (bits 29-31) positions are used on SATR.

Diagram 5-808. Set Address Translator, SATR (00), Execution in Issuing CE (Sheet 3 of 6)
Objectives:
Check for all responses to 1st SATR select.
Send ATR Slots 1-8 to selected elements.
Load own ATR if select mask specifies this CE.

After a 3-cycle delay gate the select register to T with good parity.

Gate select reg to T and T to PAL to check responses.

If (Some element did not respond) set condition code to 1.

Gate ATR slots 1-8 to external register.

5 to select register.

Turn on 'timing gate' trigger.

Delay 2 cycles

Decrement ABC counter by 1.

Upon entry ABC = 7, this delay loop holds the 2nd SATR select active for approximately 4.4 usec. No response is expected.

Turn off 'timing gate' trigger.

Read ATR slots 1-8 into T and slots 9 and 10 into T (2-7)

Issuing CE ATR to be set

Issuing CE to ATR (0-39).

Gate 5 (0-31) and T (2-7) to ATR (00-39).

Diagram 5-808. Set Address Translator, SATR (OD), Execution in Issuing CE (Sheet 4 of 6)
Objectives:
Send ATR slots 9 and 10 to selected elements.
Check for a response from all selected elements.

Set time constant in A.

Gate ATR slots 9 and 10 into external register.

Turn on 'timing gate' trigger.

Delay 2 cycles
Add 1 to A.

PAL 20-02 = 0

Turn off 'timing gate' trigger.

After a 2-cycle delay, gate select register with good parity to 1.

Responses reset select register bits as before (Sheet 4).

Reset issuing CE's input latch if its own ATR was set.

Each selected element's ATR has been set with no errors if PAL equals zero.

Set condition code to 2.

Sheet 6

Diagram 5-808. Set Address Translator, SATR (OD), Execution in Issuing CE (Sheet 5 of 6)
Objectives:
Gate response bits from Select register.
Arrange the response bits in the response byte format and store in GPR per R2.

Gate select register to Y (via 9020 and local store out buses).

Gate IOCE bits from T(60-63) to F(4-7).

Combine IOCE bits and CE bits in T(48-55).

Store response bits in GPR specified in R2 field.


Note:
A CE receiving a SATR select signal from another CE must execute a microprogram to ingate the new ATR information. Refer to Diagram 5, MDM for explanation of the SATR instruction in the receiving CE.

Diagram 5-808. Set Address Translator, SATR (3D), Execution in Issuing CE (Sheet 6 of 6)
Objectives:
- Receive SATR select for the 1st time.
- Turn on 'time clock step' trigger and wait for select to drop.
- Send response and wait for 2nd SATR select.

The CE receiving a SATR select signal responds to the issuing CE and executes a microprogram to log new ATR information from the external bus. This diagram shows the functions performed by the receiving CE.

- Set CE 1, 2, 3, or 4 input latch.
- Turn on 'time clock step' trigger.
- This trigger forces ROS address 014 to enter response ATR microprogram.
- Set delay constant in F.
- Add 1 to F.
- Yes
- Carry out of serial adder
- No
- Reset ATR select; CE input latch, and 'time clock step' trigger.

Timeout before select drops is an error.

Reset ATR select; CE input latch, and 'time clock step' trigger.

Timeout before select comes back is an error.

Receive SATR Timings - Related to Issuing 7201's SATR Select

<table>
<thead>
<tr>
<th>SATR Select (Issuing CE)</th>
<th>Response (Receiving CE)</th>
<th>Gate External Bus to S</th>
<th>Gate External Bus to T</th>
<th>Gate S and T to ATR</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.6 usec</td>
<td>4.8 usec</td>
<td>9.6 usec</td>
<td>-</td>
<td>(ATR Slots 1-8)</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(ATR Slots 9-10)</td>
</tr>
</tbody>
</table>

Diagram 5-809. Set Address Translator, SATR (ID), Execution in Receiving CE (Sheet 1 of 3)
Objectives:
Receive 2nd SATR select.
Gate ATR slots 1-8 to S.
Wait for 3rd SATR select.
Gate external bus (ATR slots 1-8) to S.

Yes (Machine Check)
Set delay constant in F.
Add 1 to F.
Yes
No
Timeout before select comes back is an error.
Reset ATR select; CE input latch, and 'time clock step' trigger.

Yes
Receive ATR select line active
Set delay constant in F.
Add 1 to F.
SA1 = 0b1000
Yes
No
Reset ATR select; CE input latch, and 'time clock step' trigger.

Yes
Receive ATR select line active
(3rd SATR select is active.)

Diagram 5-609. Set Address Translator, SATR (OD), Execution in Receiving CE (Sheet 2 of 3)
Diagram 5-809. Set Address Translator, SATR (ID), Execution in Receiving CE (Sheet 3 of 3)
- **RR Format**

<table>
<thead>
<tr>
<th>GT</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>11</td>
</tr>
</tbody>
</table>

- **Purpose:** To transfer configuration mask to CCR of element(s) specified by selection mask.

- **Conditions at the end of I-Fetch:**
  1. A-reg has 1st word of configuration mask (contents of GPR specified by R1).
  2. T-reg has selection mask (contents of GPR specified by R2).

- **Configuration and selection mask formats:**
  1. 90300 System

![Diagram](image)

<table>
<thead>
<tr>
<th>State</th>
<th>Scn</th>
<th>SE</th>
<th>CE</th>
<th>IOCCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2nd Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 18 19 20 23 24 26 29 31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Selection Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 2 3 5 6 7 8</td>
</tr>
</tbody>
</table>

2. 90302 System

![Diagram](image)

<table>
<thead>
<tr>
<th>State</th>
<th>Scn</th>
<th>SE</th>
<th>DE</th>
<th>CE</th>
<th>IOCCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2nd Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 18 19 20 23 24 26 29 31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Selection Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 2 3 5 6 7 8</td>
</tr>
</tbody>
</table>

- **Objectives:**
  - Get select mask from T
  - Get 1st word of configuration mask to external reg.

Diagram 5-6. Set Configuration, SCON (01) (Sheet 1 of 6)
Objectives:

Check for specification and privileged operation errors.

Hardware checks that (1) own SCON bit is on in CCR and, (2) at least 1 valid SCON bit is on in configuration mask KR101.

Develop constant 3C to test Scan field of configuration mask.

AND constant 3C with Scan field in serial adder.

Hardware checks that (1) own SCON bit is on in CCR and, (2) at least 1 valid SCON bit is on in configuration mask KR101.

Objective: Check if any IOCEs are selected.

Develop mask Q7 to test IOCE select field.

AND mask Q7 with IOCE mask field.

Specification error occurs if:
1. CE is in state 1 or 2.
2. CE's scan bit is off.
3. No valid CE specified in configuration mask.
4. RI field specifies an odd GPR.

Check if any IOCEs are selected.

Develop mask Q7 to test IOCE select field.

AND mask Q7 with IOCE mask field.

Specification error occurs if:
1. CE is in state 1 or 2.
2. CE's scan bit is off.
3. No valid CE specified in configuration mask.
4. RI field specifies an odd GPR.
Objectives:
Check that only one CE communication bit is set in configuration mask.

A

AND mask 01 with CE mask field in serial adder.

B

SAL = 0

Yes

CE 4 bit on. Check CE 3, 2, and 1.

C

Shift T left one to develop mask 02.

AND mask 02 with CE mask field in serial adder.

D

SAL = 0

No (CE 2 bit on)

Yes

Two CE bits on

Yes

To test CE 2 communication bit. Use same data path as above.

E

SAL = 0

No (CE 2 bit on)

Yes

No

Two CE bits on

No

To test CE 3 communication bit. Use same data path as above.

F

SAL = 0

No (CE 1 bit on)

Yes

Two CE bits on

Yes

Set specification error.

End Op

G

C

Diagram 5-810. Set Configuration, SCON (01) (Sheet 3 of 6)
Objectives:
Set count of 19 (hex) in B for 'select timeout' (5.0 µsec).
Set count of 19 in D for 'response timeout' (0.5 µsec).

Diagram 5-810. Set Configuration, SCON (01) (Sheet 4 of 6)
Diagram 5-810. Set Configuration, SCON (01) (Sheet 5 of 6)
The example below shows the basic logic active during the select and response timeouts. CE1 is executing the SCON instruction, and CE2 is receiving the 'scon select'. Similar select and response logic is used for all elements in the system.

**Diagram 5-810. Set Configuration, SCON (01) (Sheet 6 of 6)**
Test and Set, TS (93)

Diagram 5-13. I-Fetch

A

QK021  093
SI I-Fetch.

B

Test high-order bit (bit 0) of storage operand (byte in storage), set CC according to state of tested bit, and set addressed byte block into storage as all '1's.

C

Test high-order bit (bit 0) of instruction byte in E.

D

Transfer D(2 l-23) to STC directly and to ABC via parallel cable.

E

Set STC to 4.

F

Gate SDBO to AB.

G

Enable resetting of mark register after late BCU cleanup.

Notes:
1. Because PAL(64-67) is ignored but may be equal to 0 through 255 (decimal),
   IC + 7 = D + 7.
2. Because PAL(64-67) is ignored but may be equal to 0 through 15 (decimal),
   IC + 7 = D + 7.

Conditions at start of execution:
1. First 16 bits of instruction are in E.
2. Storage operand address is in D.
3. Storage request per D is blocked during I-Fetch ('D sync' latch blocked) because data byte could change before executing this instruction.

'Test and set' trigger: MC181.
Objectives:
Determine entry type. (Initial entry or re-entry after an interrupt or after overflow.)

No
Entry is a return after an interrupt. Re-entry is back to the routine exited from.

Yes

Perform delete descriptors routine.

Delete descriptors from the ODT by incrementing the ODT address without incrementing the NDT address.

Move history descriptors.

Moves history descriptors from the ODT to the NDT.

Yes

Point WDT order or modify order.

No

Perform modify routine.

Moves descriptor from the ODT (current in ODT) to the NDT; changes these to history descriptors.

Set history count to zero.

Single symbol data has no history data.

No

Perform insert descriptors routine.

Calculates the number of symbols to be inserted by this insert order and builds a descriptor (batch No. and symbol count) and stores it in the NDT as part of the current descriptors.

Move history symbols.

Moves symbols from old refresh memory to new refresh memory, resetting all brightness bits.

Move current symbols.

Moves symbols from old refresh memory to new refresh memory, leaving brightness bits as is.

Insert new symbols routine.

Moves symbols from a sort bin to new refresh memory, reformattng them if necessary.

Yes

Prepare to process next class type.

Invalid sequence - specification check.

No

Point WDT order EOD.

New Refresh Memory

Move history symbols.

Move current symbols.

Insert new symbols.

Yes

End-up

Note:
All six blocks shown below represent blocks of storage whose sizes are

Diagram 5-901. Repack Symbols, Simplified Flow Chart
Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 1 of 21)

- **Purpose:** Assembles a new updated display image (refresh memory) for 1/16 of a PVD's area. It accomplishes this by deleting from the old display image, all information which is no longer to be displayed by the page, by moving (and modifying, if necessary) the data remaining in old refresh memory (ORM) to new refresh memory (NRM), and by adding new data to be displayed by inserting data from a sort bin (created by CSS execution) into NRM.

- **RPSB moves two types of data, descriptors and display words,**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Batch Number</th>
<th>Symbol Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td></td>
</tr>
</tbody>
</table>

- **Diagram Details:**
  - **Sheet 2:** CFE Sheet 2
  - **Sheet 3:** FEMDM (7/70) 5-902, Sh 1

- **Reference:** 3-letter designations identify actual line edge connectors on CAS page.
Diagram 5-902. Repack Symbols, RPSB (Sheet 2 of 21)
Determine if there is more data to process or if the instruction execution is to be terminated.

Gate WCT address (S) to D.

Make memory request for next WCT orders.

Yes

Reset E27 K to zeros.

Gate ODTA (GPR 2) to T.

Gate SDBO (WCT orders) to AB.

Update ODT address (T) +2 result to IC.

Set paging latch if 512 carry on IC update.

Make memory request per IC (for ODT data).

Update WCT address +2.

Set condition code to 0.

Gate SDBO (ODT data) to Q and LM.

Gate LM to N and Q to R.

Gate R (ODT entry) to E.

Set FPR 2 to O.

This re-entry is after a complete class type has been processed.

Gate WCT address (D) to T.

Set condition code to 2.

Turn on STAT G.

Store updated ODT address in GPR 2.

Diagram 5-902. Repack Symbols, RPSB (IF) (Sheet 3 of 21)

* 3-letter designations identify actual line edge connections on CAS page.
Objectives: Examine WCT order and branch to proper delete routine or to move history routine.

- Both descriptor tables (ODT and NDT) and the WCT are physically located in SEs.
- Make memory request for WCT doubleword.
- Gate GPR 4 to T.
- Gate first ODT descriptor to E.
- Gate constant 4 to FPR 4.
- Store current WCT address in GPR 7.
- Gate WCT order to SAL for testing.
- Gate symbol count to FPR 4 (from K).
- Gate batch numbers from N (ODT) and A (WCT) to SAL for compare.
- Turn on STAT G. Indicates to move routine that entry is from delete routine.
- If delete symbol count is odd, the symbol count will be increased by one when the first descriptor is moved from ODT to NDT and the associated history or current symbol count will be increased by one. The move symbols routines will insert a corresponding null symbol in NRM.

Diagram 5-902. Repack Symbols, RPSB (Sheet 4 of 4)
Objectives:
Delete all descriptors from ODT up to but not including the first null descriptor.

Goto descriptor (N0-3) to SAL.

Yes

Step ODT address to next descriptor.

No

Delete null descriptor.

Q0411

Add K (symbol count) to E(S-15) result to K.

turn on STAT D.

Add K (symbol count) to E(S-15) result to K.

turn on STAT D.

Yes

Step ODT address to next descriptor.

No

Turn off STAT D.

Gate descriptor from R to E.

Add this delete order symbol count (E) to the accumulated symbol count (K).

To update the count of the deleted symbols.

To check if it is a null.

Make memory request per IC.

Gate descriptor from R to E.

Leaves the null descriptor in the ODT.

Deleted null routine is complete.

Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 5 of 21)
Objectives:
Delete descriptors from ODT which have same batch number as the delete order in WCT.

1. Turn on STAT O.
2. Set paging latch if 10/2 carry on ODT address update.
3. Add WCT order symbol count (BB-15) to K.
4. Gate next descriptor from LM to Q and from R to AB.
5. Update WCT address (01-2) to next WCT order.
6. Gate 512 to SAL.
7. Gate next descriptor from R to E.
8. Gate SDB to Q and LM.

Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 6 of 21)

*3-letter designations identify actual line edge connection on CAS page.
Objectives:
Move history descriptors from the old descriptor table (ODT) to the new descriptor table (NDT).

G0441: POI
Gate even ODT address to FPR 0.

G0441: POI
Set paging latch if 512 carry on ODT address update.

G0441: POI
Store ST per DIN NDT. Gate SDBO (ODT doubleword) to LM and Q.

G0441: POI
Gate next ODT entry from LM to N and Q to R.

G0441: POI
Gate updated ODT address to D.

G0441: POI
Gate next ODT entry from LM to N and Q to F.

Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 7 of 21)
**Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 8 of 21)**

- **A**: Check work control table (WCT) orders for modify orders and compare ODT and WCT botch numbers.
  - Store WCT doubleword.
  - Gate next WCT order (F) to SAL for check.
  - Gate current NOT address (GPR 7) to T.
  - Gate WCT address (GPR 7) to S.

- **B**: Store NOT doubleword.
  - Gate next WCT order (F) to SAL for check.
  - Gate current NOT address (GPR 7) to T.
  - Gate WCT address (GPR 7) to S.

- **C**: Gate WCT address (GPR 7) to S.
  - Make memory request for next WCT doubleword.

- **D**: Store current NOT address (GPR 7) to T.
  - Gate WCT address (GPR 7) to S.
  - Make memory request for next WCT doubleword.

- **E**: Update ODT address +2.
  - Gate ODBO (WCT data) to LM and Q.
  - Gate LM to N and Q to R.

- **F**: Compare 3-letter designations identify actual line edge connection on ODT page.
  - Update ODT address +2.
  - Set paging latch if 122 carry on ODT update.
  - Make memory request for next ODT data.

- **G**: Gate F (WCT order) to SAL for check.
  - Gate ODBO (WCT order) to SAL for check.

- **H**: Gate F (WCT order) to SAL for check.

- **I**: Gate next WCT entry thru SAL for check.
Objectives:
Move old descriptor from ODT to new descriptor into NDT.

Update IC (ODT address) +2.

Set paging latch if 512 carry on ODT update.

Gate NDT address (GPR 3) to S.

Gate NDT address (S) to D.

Move first byte of ODT entry N (0-7) to ST.

Add 1 to symbol count if odd delete symbol count.

Move second byte of ODT entry N (8-15) to ST.

Store NDT entry in NDT (per D).

Update NDT address +2 result to T.

Add symbol count (ES-15) to accumulated history symbol count (K).

Gate GPR 7 to S.

Gate (next ODT descriptor) LM to N and Q to R.

Gate WCT address (S) to D.

Store next NDT address in GPR 3.

Gate next WCT entry thru SAL for check.

Set DP bits in GPR 11.

Gate (next ODT descriptor) LM to N and Q to R.

Gate OMR (ODT done) N LM and Q.

Diagram 5-902. Repack Symbols, RPSR (6F) (Sheet 9 of 21)
End of modify descriptors

store null descriptor in current descriptors.

Set all zeros (null) descriptor in next NULL slot.

Set K to all zeros.

Gate next OOT entry (N-7) to SAL for check.

Store null descriptor in NULL.

Gate N(7) to SAL for check.

Store new NULL address at end of old NULL page.

Update NULL address +2 result to D.

Turn off STAT G.

Yes

Set bit in GPR 11.

Turn off STAT G.

No

Store new NULL address at end of old NULL page.

Update NULL address +2 result to D.

Turn off STAT G.

Yes

* 3-letter designations identify actual line edge connection on CAS page.

Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 10 of 21)
Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 11 of 21)
Objectives:

- Determine the correct insert symbol count.
- Assemble the new descriptor (batch number and symbol count) and store it in the NDT.

Add (insert symbol count) to accumulated current symbol count.

Sort bin stop address > sort bin start address

- Symbols to insert contained in two pages
- Sort bin stop address = sort bin start address
- Add 504 to symbol count in B.

- Divide symbol count by 4.
- Insert Gate WCT order batch number to ST byte.
- Gate insert symbol count (B) to ST byte to form new descriptor.

This converts count in bytes to count in words.

- Store new descriptor in NDT.
- Update WCT address (IC) + 2.
- Update NDT address (D) + 2.
- Make memory request for next WCT order.
- Update WCT address + 2.
- Gate WCT order from L,M to N and Q to R.
- Read GPR 11 (new NDT page address) to T.
- Store new NDT address at end of old page.
- Gate new NDT page address to D.
- Store DP (page) bits in GPR 11.

Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 12 of 21)
Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 13 of 21)
Move data words containing history symbols or current symbols from old refresh memory into LM, thru the mixer into XY and back to store.

Add B to NRMA and gate result to D.

Make memory request for next ORM data.

Turn on STAT D.

Make memory request for next ORM data.

Make memory request for next ORM data.

Move history symbols in current symbols from ORM memory into new refresh memory.

Issuing 'FMTO E13-15' micro-order to gate ORM data to XY.

Update ORM A (IC) + B result to IC and GPR 4.

Update CRM data (D) + B result to D.

Make memory request for next ORM data.

Gate next CRM address (GPR 6) to S.

Issue 'FMTO E13-15' to gate ORM data to XY.

Two symbols have now been moved from ORM to NRM.

Gate current count (S) to B.

Add history count to current count result to B.

Subtract 1 from E12-15.

Gate current count (GPR 1) to S.

Subtract 1 from E12-15.

Count is checked for this decision.

Add B to NRMA and gate result to D.

Count is checked for this decision.

Sheet 15

Add B to NRMA and gate result to D.

Sheet 15

Subtract B from ORMA (IC) result to GPR 4 (via T).

Issue 'FMTO E13-15' micro-order to gate ORM data to XY.

Update ORM A (IC) + B result to IC and GPR 4.

Sheet 15

Update CRM data (D) + B result to D.

Sheet 15

Make memory request for next ORM data.

Sheet 15

Sheet 15

DIagram 5-902. Repack Symbols, RPS8 (OF) (Sheet 14 of 21)
Determine if there are any symbols (history, or current) left to move from old refresh memory and branch to the proper routine.

Subtract 1 from E12-15.

Turn on STAT D.

Add history count to current count result to B.

Gate ORM (IC) to GPR 4.

Turn off STAT D.

Issue 'FMTO* E13-15' to gate ORM data to XY.

Subtract 1 from E12-15.

Add 8 to current count result to PAL for check.

Diagram 5-902: Repack Symbols, RPSB (OF) (Sheet 15 of 21)
Objectives:

- Insert symbols from a sort bin (bin data) into new refresh memory (LM) when both addresses and boundaries are even (doubleword) and symbols are loaded at a time.

- Insert symbols count to B.

- Gate constant (4) into ST.

- Make memory request for sort bin data.

- Subtract 8 from insert symbol count (BJ • E08).

- Update NRM (D) +8.

Diagrams 5-902. Repack Symbols, RPSB (OF) (Sheet 16 of 21)
Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 17 of 21)
Objectives:

Insert symbols from a sort bin (SB) into NRM when the sort bin address is on an add boundary. Aligns the data on an even boundary before storing it.

- Subtract 4 from sort bin address (IC).
- Subtract 8 from symbol byte count (B).
- Adjust the sort bin address to an even boundary.
- Issue 'FMTN* E13-15' microorder to gate bin data in M to X.
- Make memory request for next sort bin data.
- Update symbol count (IC) +4.
- Set paging latch if 512 carry on update.
- Issue 'FMTN* E13-15' microorder to gate sort bin data in M to X.
- Make memory request per IC.
- Update sort bin address +8.
- Set paging latch if 512 carry on update.
- Issue 'FMTN* E13-15' microorder to gate last bin data from L to Y.
- Make memory request per IC.

This moves the second word in the doubleword into the first word position.
Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 19 of 21)
Diagram 5-902. Repack Symbols, RPSB (DF) (Sheet 20 of 21)

5-902, Sh 20 (7/70)
Diagram 5-902. Repack Symbols, RPSB (OF) (Sheet 21 of 21)
Diagram 5-903. Convert and Sort Symbols, CSS (02) (Sheet 1 of 10)

5-903, Sh 1 (7/70)
Objectives:
Prepare to process the right word of the input doubleword.

Gate brightness control bit (R) to F.
Check for header word, AND A [6-7] with 00 and store result in F.

Check for header word, AND A [6-7] with 00 and store result in F.

Objectives:
Determine if this is initial entry or re-entry to this input stream and if the input word is a header or a data word.

Sheet 3

Re-entry instruction execution just beginning.
Moves BL, BR bits to T(44-45) on initial entry-left data word taken on re-entry.

Objectives:
Convert and Sort Symbols, CSS (02) (Sheet 2 of 10)
Objectives:

Take new header information from input word and replace old header information with it.

New symbol to A(0-7)...

Sheet

Word count (S) to B.

Input stream address (D) + 8 and back to D.

Set carry latch if input address update goes 512 carry.

Make memory request for new input double word.

Set STRAT H if R bit (GPR II bit) in F is off.

No G Sheet

Indicates the end of page.

Replaces old header information with new.
Objectives:
Perform geographic filter to determine if the input target is within the geographic boundaries of the PVD.

Geographic origin.
Geographic origin from input paint.

Results are saved for diagnostic program use.

Sheet 5, 9, 10

Diagram 5-903. Convert and Sort Symbols, CSS (02) (Sheet 4 of 10)
Perform corner filters. (Corners cannot be displayed on PVD face.)

If tgt is in corner bin., it will be bin 4 or 16.

Target is not in a corner bin. Begin output for formatting for output word.

The target is rejected if there is no carry out of bit 0 or if SAL is all zeros or any combination of the two.

Add co-ordinates together. Result is in SAL.

The point of target rejection. If the result of the addition of the X and Y co-ordinates is 256 or greater (carry out of SAL), the target is not in the corner (shaded area) and will be displayed.

Add the eight low-order X co-ordinate bits (true or complemented) to the eight low-order Y co-ordinate bits thru SAL.

Combine Y and X co-ordinates in GPR 13 (Y-31-21, X-23-31).

Gate header info (T) to B shifted right four places.

Read GPR 10 (word count) to S.

Gate K (next input address) to D.

Set B to true.
Objectives:

Assemble the CA2, CA3 -- Y and X coordinates in IT for assembly of output word (Y-45,53, X-55,63).

CB2 --- Gate F(4-7) 8(64-67) will be 0's if an even bin. Shifted right one.

Gate Bl and Br bits to F(4-7) and bin number to F(0-3). ---

Assembling the output word in IT.

Gate F(4-7) (BL and BR) to T(44-45).

Gate completed output word to GPR 13.

Gate sort bin base address (S) to A.

Update bin displacement T(48-63) + 4 and gate it to B.

Read GPR 9 (next input address) into S.

Add sort bin base (A) to sort bin displacement (T) and gate result to D.

Gate updated bin displacement (in B) back to T.

Store updated bin displacement in appropriate GPR (per E12-15).

Diagram 5-903. Convert and Sort Symbols, CSS (02) (Sheet 7 of 10)
A

Gate S (next input address) to K.

Turn on STAT B.

Add 6 to D and go to E.

Add 6 to D and go to E.

Gate S (sort bin base address) to D.

Diag 5-903. Convert and Sort Symbols, CSS (02) (Sheet 8 of 10)
Check the PVD index (GPR 11) and input word 2, 3, or 4 for this PVD. If the bit is a one, the selected Beacon data Sheet 9 is to be deployed on this PVD.

Make memory request for second double word of input data block.

Gate words 3 and 4 of input data block into S, T.

Check PVD index bits 0 and 1 to choose the correct index word.

Word 2 is in B.

PVD index (GPR 11) (address)

Byte Bit

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

C86

No

Selected

Objectives: Prepare to submit the input target to the filters for display.

End-up

Diagram 5-903. Convert and Sort Symbols, CSS (02) (Sheet 10 of 10)
Read an input-doubleword into AB (from the storage address in GPR9).

Update the input address and store it back in GPR9.

Decrement word count.

Yes

Save the header information (needed for output doubleword).

No

Check XY coordinates in left input word (A) to see if that end of the weather line is within the PVD geographic boundaries.

Yes

Turn on STAT G (word one needs truncation).

No

Truncate the weather line until the coordinates in the right word are within the PVD geographic area (in the outer one-tenth of the area).

Yes

Check XY coordinates in right input word (B) to see if that end of the weather line is within the PVD geographic boundaries.

Yes

Truncate the weather line until the coordinates in the left word are within any of 3 sterile areas.

No

Yes

Turn on STAT G. (Truncate left word)

No

Reset STAT G.

Assemble data (header info) for output doubleword in T and gate it to M (as format word 0).

Gate format word 0 (M) thru mixer into XY.

Assemble data (right coordinates) for output doubleword in T and gate it to M (as format word 1).

Gate format word 1 (M) thru mixer into XY.

Assemble data (left coordinates) for output doubleword in T and gate it to M (as format word 2).

Gate format word 2 (M) thru mixer into XY.

Store XY (output doubleword) in refresh memory.

Yes

No

Save all data required to continue processing input stream after the interrupt has been handled.

Input word count ≠ 0

No

Program interrupt

No

End op

Yes

No

End op

Branch to the interrupt-handling routine.
Objectives:

- Gate the first doubleword from the input stream into A and B.
- Test to see if the doubleword contains header information or weather line co-ordinates.
- Gate the address of the next input word to D.
- Gate constant (80) to F.

Diagram 5-905: Convert Weather Lines, CWL (03) (Sheet 1 of 9)
**Objective:**
Save the header information for use later (to build an output doubleword).

**Objective:**
Goto header information (control bits) from A (byte 1) to T (byte 0).

**Objective:**
Make a memory request for the next input doubleword.

**Objective:**
Update the input word address.

**Objective:**
Goto header information (symbol) from B (byte 0) to T (byte 1).

**Objective:**
Store T in GPR 8 (saves the header information).

**Objective:**
Gate next input doubleword into AB.

**Objective:**
Test to see if the doubleword contains header information or whether line coordinates.

**Objective:**
Gate the new input doubleword into A and B.

**Objective:**
Gate constant (80) to $S$.

**Objective:**
Gate A (second coordinates) to $K$.

**Objective:**
Gate the input word count to IC (via T).

ANDS (byte 0) with a (byte 0).

---

If the input word is a header, SAL will contain 80 as a result of this AND.

**Diagram 5-905.** Convert Weather Lines, CVWL (03) (Sheet 2 of 9)
Objectives:

Perform the geographic filter (determine if the input point is within the PVD geographic area).

Perform the first and second sterile area filters (determine if the input point is within a sterile area on the PVD screen).

Two sets of coordinates are used to describe the geographic boundaries of a PVD. Two sets are also used to describe each sterile area within a PVD's geographic boundaries. The set of coordinates which describe the geographic area is referred to as the geographic origin, the other set, identifying the uppermost right point of an area, is referred to as the geographic limit.

Diagram 5-905. Convert Weather Lines, CVWL (03) (Sheet 3 of 9)
Objectives:
Perform third sterile area filter (determine if the input point is within a sterile area on the PVD screen).

Assign the major position.
Assign the secondary position.

Objectives:
The input word one passed all filters; input word two must be truncated.

Objectives:
The input word two passed all filters; input word one must be truncated.

Objectives:
The line described by the current input doubleword has been rejected.
Prepare to read in a new input doubleword.
Prepare to honor any pending interrupt.

Objectives:
The major position is the set of co-ordinates obtained by subtracting the origin co-ordinates from the input co-ordinates. It can be a point or end of the weather line or the final point, whichever will be the end whose co-ordinates pass all previous filters first.

Objectives:
The intercept point is the set of co-ordinates obtained by subtracting the origin co-ordinates from the input co-ordinates. It can be a point or end of the weather line or the final point, whichever will be the end whose co-ordinates pass all previous filters first.

Objectives:
The line described by the current input doubleword has been rejected.
Prepare to read in a new input doubleword.
Prepare to honor any pending interrupt.

Objectives:
The intercept point is the set of co-ordinates obtained by subtracting the origin co-ordinates from the input co-ordinates. It can be a point or end of the weather line or the final point, whichever will be the end whose co-ordinates pass all previous filters first.

Objectives:
The intercept point is the set of co-ordinates obtained by subtracting the origin co-ordinates from the input co-ordinates. It can be a point or end of the weather line or the final point, whichever will be the end whose co-ordinates pass all previous filters first.
Objectives:
One end of the weather line has failed the geographic filter and must be truncated. Proceed to the secondary point (maximum of 9 times) until it is within the PVD's geographic area.

2

Sheet 4

2

Objectives:
Compute the secondary position (from the truncated point) and save it. Proceed to enter the routine to convert this position to the PVD scale.

3

Sheet 4

3

Objectives:
One end of the weather line has failed the geographic filter and must be truncated. Proceed to the secondary point (maximum of 9 times) until it is within the PVD's geographic area.

4

Sheet 4

4

Objectives:
Compute the secondary position (from the truncated point) and save it. Proceed to enter the routine to convert this position to the PVD scale.

5

Sheet 4

5

Objectives:
Compute the secondary position (from the truncated point) and save it. Proceed to enter the routine to convert this position to the PVD scale.

6

Sheet 4

6

Objectives:
Compute the secondary position (from the truncated point) and save it. Proceed to enter the routine to convert this position to the PVD scale.

Diagram 5-905. Convert Weather Lines, CVWL (03) (Sheet 5 of 9)
Objectives:

Multiply both X and Y coordinates by the 6-bit conversion constant.

AND (24-27) with F(0-3) thru the serial adder.

Checking scaling control constant bits 24-25 to determine which scale.

Conversion Constant

Scale: Multiplier

10H-2BH (conversion constant) to F.

Multiplier X and Y coordinates by second two bits of the conversion constant.

Gate X (X and Y coordinates) to T.

Multiply X and Y coordinates by last two bits of the conversion constant.

AND 5(24-27) with F(0-3) thru the serial adder.

Checking bits 24 and 25 to see if this is large-scale conversion.

No Large or small scale conversion

Yes Multiplying X and Y coordinates by last two bits of the conversion constant.

No Large or small scale conversion

Yes Multiplying X and Y coordinates by last two bits of the conversion constant.

No Large or small scale conversion

Yes Multiplying X and Y coordinates by last two bits of the conversion constant.

No Large or small scale conversion

Yes Multiplying X and Y coordinates by last two bits of the conversion constant.

No Large or small scale conversion

Yes Multiplying X and Y coordinates by last two bits of the conversion constant.

Divide the multiplication result by 16 and gate the result to B and T.

Large scale conversion

Multiply B (X and Y coordinates) by 4.

Gate result to T.

Conversion is Complete

Move converted Y coordinates (T 32-47) to A (16-31) and B (48-63).

STAT D off indicates that this routine was entered from the CVWL instruction. The microprogram branches back to CVWL now.

STAT D on indicates that this routine was entered from the CSS instruction. The microprogram branches back to CSS now.

This routine is complete.

STAT D off indicates that the second set of coordinates must be converted.

This routine is complete.

Diagram 5-905, Convert Weather Lines, CVWL (03) (Sheet 6 of 9)
Objectives:

A. Compute delta X and delta Y values for the output doubleword. Store the signs of delta X and Y for the output doubleword. Decide if the symbol is to be displayed at both ends of the weather line and set the CO and Cl bits accordingly.

B. Gate T (secondary X coordinate) to T.

C. Subtract T (major X coordinate) from B (secondary X).

D. Gate result of the above subtraction (delta X) to B.

E. Gate GPR 3 (secondary Y coordinate) to T.

F. Gate result of the above subtraction (delta Y) to A.

G. Gate GPR 8 (header info) to T.

H. Gate F to T (56-63).

I. Gate GPR 4 (truncation flag) to S.

J. Gate S to PAL.

K. Set the AXS and AYS bits into T for output formatting.

L. The symbol, from the header information, is needed now for the next decision. The header information remains in T and is the beginning of format word 0 which is assembled in T for input to the mixer.

M. Format word 0

N. The symbol = 1, means symbol to be displayed at major position of weather line.

O. The symbol = 1, means symbol to be displayed at secondary position of weather line.

Diagram 5-905. Convert Weather Lines, CVWL (03) (Sheet 7 of 9)
Objectives:

1. Assemble data to appear in output doubleword in T. Gate this data, in the form of format words 0, 2, and 1, to M.
2. Assemble output doubleword in XY by gating the format words (one at a time) from M thru mixer into XY.
3. Store output doubleword into refresh memory.

Q281 Reset E (B-15).

Reset CO and Cl bits in F (no symbol to display).

Gate T (format word 0) to M.

Preparing to issue first "FMTW* E14-15" micro-order.

"FMTW* E14-15" puts format word 0 into the correct XY bit positions for the output doubleword.

Add one to E (12-15).

Gate GPR 3 (major X) to T (via B).

Add one to E (12-15).

Gate GPR 2 (refresh memory address) to T (via B).

Make memory request per D (to store in refresh memory).

Set marks 0-7.

Gate GPR 2 (refresh memory address) to T (via B).

Refresh memory address 16 to T (via B).

Step 7 to GPR 5.

Gate AX (format word 2) thru mixer to X and Y.

"FMTW* E14-15" puts format word 1 into the correct XY bit positions for the output doubleword.

Set STAT Hand STAT G.

Diagram 5-905. Convert Weather Lines, CVWL (03) (Sheet 8 of 9)
Objective:
Honor any pending interrupts; if none, continue to process the input stream.

Gate LSWR (address of next instruction) to S.

Gate S to IC.

Gate IC (next input doubleword address) to D.

Make a memory request per D.

Gate D (next input doubleword address) into GPR 9.

Gate new input data doubleword into A and B.

Update input doubleword address +8 (in D).

Gate 5 (input count) to PAL.

PAL = 0

Yes (No more input to process)

End op

This instruction does not have to be re-entered.

No

External interrupt

No (Program interrupt error)

End op

There has been a program error - terminate the instruction execution.

Yes

One input weather line has been completely processed and stored in refresh memory. All information required to begin processing the next input weather line upon reentry has been saved. The pending interrupt is now honored by branching to the interrupt handling routine.
Objectives:
Move the first operand to the third operand location.

Gate first operand (GPR specified by R1 field) to T.

Objectives:
Move the second operand to the first operand location.

Gate doubleword containing the second operand (SDBO) into ST.

Second operand in?

Yes

Gate S to T (via PAL).

No

Objectives:
Set the proper condition code.

T(63) equal to zero?

Yes

Set condition code to 1.

No

Set condition code to 3.

Diagram 5-906. Load Chain, LC (52)
Diagram 6-2. CE Roller Switch Indicators (Sheet 1 of 2)

**Display No. 1**

<table>
<thead>
<tr>
<th>Diagram</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>G Register (Write Direct)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTV Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Display No. 2**

<table>
<thead>
<tr>
<th>Diagram</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address Translation Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTV Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Display No. 3**

<table>
<thead>
<tr>
<th>Diagram</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address Translation Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTV Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Display No. 4**

<table>
<thead>
<tr>
<th>Diagram</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address Translation Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTV Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

*Diagram 6-2 includes details on CE Roller Switch Indicators with various registers, controls, and indicators for the switch positions.*
Diagram 6-2. CE Roller Switch Indicators (Sheet 2 of 2)
Diagram 6-3. Pushbutton Signal Generation (Sheet 2 of 2)
Diagram 6-4. Stop Loop Routine (Sheet 2 of 2)
Diagram 6-5. Stop Loop Monitored Pushbutton Gating
Diagram 6-6. Stop, Manual, Address Compare Triggers, and Block Interrupt Latch (Sheet 1 of 2)
For STOP function: ADDRESS switches 8-28 set to desired stop address. For LOOP function: ADDRESS switches 8-28 set to branch address; DATA switches 40-60 set to restart address.

Diagram 6-6: Stop, Manual, Address Compare Triggers, and Block Interrupt Latch (Sheet 2 of 2)
Notes:
1. Power on Reset: Force ROSAR = \text{Clear local switch (OR enter stop loop, QY041)}.
2. Other Resets: Force ROSAR to \text{Clear local switch (OR enter stop loop, QY041)}.

Diagram 6-7. CE Machine Reset and Force Address
**Legend**

- CE selected by CC/SC if IPL is initiated at CC/SC.
- CE at which IPL or PSW Restart is initiated.
- Signal origin.
- Signal destination.

**Functions performed by IPL PSW Restart**

- **Reset CC/SC**
  - CE of which IPL or PSW Restart is initiated.
  - Reset any other system element not in test (CC/SC, IOCEs, SEs, TCUs, PAMs, RCUs).
- **Set SCON**
  - CE's SCON bit on and not in test.

**Summary of System Operations: IPL or PSW Restart**

- **Computer Element**
  - Main Storage select switch set, Load Unit Select switches set, System Interlock key on, Load or PSW Restart pushbutton depressed.
  - (Not selected by CC/SC with its System Interlock key on.)

**Diagram 6-8A. System Operation: IPL or PSW Restart**

1. **Begin Reset**
   - From own CCR gate (Configuration Mask) into ST for SE; into External register (on Control Bus) for other elements.
2. **CPU Entry to the stop loop (Diagram 6-4, Stop Loop Routine)**
   - Begin ROS control.
3. **Put all 1's into Select register**
4. **All elements not in test will be selected to receive SCON**

**Diagram 6-9. Common Routine: IPL or PSW Restart**

1. **Begin Reset**
   - Turn on all SCON bits.
2. **Reset PSW and DAR mask, PSBAR, ATR, and CCR (all SCON bits on)**
3. **Set own ATR slot 1 and PSBAR equal to SE**
4. **Load TCU per LOAD UNIT switches**

**Legend**

- True multiplex lines.
Functions performed by IPL PSW Restart

- IPL Restart
  - (IPL only)

Legend:
- CE which is selected by CC if IPL is initiated at the CC.
- CE at which IPL or PSW Restart is initiated.
- IOCE and TCU selected by CC or CE
- LOAD UNIT switches and SE selected by CC or CE MAIN STORAGE SELECT switch.
- Signal origin
- Signal destination

Summary of Subsystem Operations: IPL or PSW Restart

- Reset any other system element configured to the issuing CE.
- Reset any I/O configured to or switched to this IOCE.

Diagram 6-3B. Subsystem Operation: IPL or PSW Restart
Computing Element

Diagram 6-4A
System Operation: IPL or PSW Restart,

Set STAT 0 for
IPL branching,

PSW
Restart

Diagram 6-4B
Subsystem Operation: IPL or PSW Restart,

Set IOCE bit in
select register,

IPL Select IOCE 00

B
Issue 1st SATR,

FS531

"SCON Response"

Yes

Set ATR 1 bits in
external register,

Issue 2nd SATR,

"SATR Select IOCE 00"

D
Issue 3rd SATR,

FS531

"SCON Response"

Yes

Set STAT B,

Set IOCE bit in
the select register.

E
Set STAT 8,
Set IOCE bit in
the select register,

"IPL IOCE (X)"

F
Sheet 2

I/O Control Element

Select and configure
IPL branch CE

Yes

ROAS is forced to D8
Return response,

QY118,404

Receive 2nd SATR before
timeout

Yes

When 2nd SATR fails,
check parity of data
for ATR 1,

QY118,403

Receive 3rd SATR before
timeout

Yes

Check parity of data
for ATR 2, load
both ATR 1 and 2,

QY118,454

Normal break out,
Return to the halt
loop,

Q200,284

IPL
(Stats 0 and 1)

Yes

GK307,84D

GK800,899

Set unit channel
and PSBAR bits in
external reg (on the
Control Bus) and
issue IPL.

Sheets 1

GK800,42A

Set up CCWs from
Control Bus, take
PSBAR to SAR,

GK800,419

Issue IPL SIO and
I/O to channel.

Sheet 2

Diagram 6.9, Common Routine: IPL or PSW Restart (Sheet 1 of 2)
Diagram 6-11  DEFEAT INTERLEAVING Switch Gating
Diagram 6-12. RATE Switch Logic
Diagram 6-13. Instruction Step Routine

2

Depress STOP pushbutton or SYSTEM RESET pushbutton.

3

Place RATE switch in 'INSN STEP position.'

No

Yes

Inhibit stepping internal timer and pulse mode operation when PULSE MODE switch is in TIME position.

4

Set 'instruction step' trigger.

Diagram 6-4

Stop loop.

5

Depress START pushbutton.

6

Reset 'step' and 'manual' triggers (ROS control).

End op.

7

Set 'step' trigger during I-Fetch of instruction.

End op.

8

Execute instruction.

9

All interruptions are executed (if not masked off) before entering stop loop.

No

Yes

10

Execute interruption routine.

Interruption pending

Set 'step' trigger during interruption (micro-order).

End op.

Diagram 6-15. Repeat Instruction Switch Logic
Diagram 6-16. Repeat Instruction Switch Routine
Diagram 6-17. ROS TRANSFER and REPEAT ROS ADDRESS Switch Gating

A

B

C

D

E
Place ROS address into ADDRESS switches for storage ripple display routine.

Depress ROS TRANSFER.
Perform ROS transfer routine.

Transfer ROS address from D to JC and B.

Set 'scan mode' trigger.
Reset E(B-15).
Reset A, B, and E.
Transfer contents of DATA switches to ST.

Set 800000 in ADDRESS switches for storage-ripple-display routine.
Set 800006 in ADDRESS switches for storage-ripple-store routine.

Must be in stop loop before pushbutton action occurs.
See Diagram 8-4. Transfer contents of ADDRESS switches to D; D to T; T(40-51) to ROSAR; load T(32-63) with 1's.

Transfer contents of ADDRESS switches to ST.

NOTE: Manual intervention (e.g., system reset or load) is required to get out of storage ripple loop.
Following this block, if ST-2, -3, or -4 microprogram is executed, these microprograms are ignored or its pointer will exercise circuits; they do not affect the storage ripple test.

Diagram 6-18. Storage Ripple Loop (Store and Display) Routine

Diagram 6-19. Wait State Gating
Diagram 6-20. Wait State Microprogram Routine

Diagram 6-21. Disable Interval Timer Logic
Diagram 6-22. CE Check Control and Inhibit CE Hardstop Switches, Logic and Error Controls
Diagram 6-23. Pulse Mode Controls
End np.

No

Yes

Set 'scan mode' trigger.

Reset 'scan mode' trigger and set 'scan counter control' trigger.

Transfer T(32-38) and T(54-63) to MCW.

Reset 'scan mode' trigger and set 'scan counter control' trigger.

Transfer starting address from B to IC, T, and D.

Reset 'stop' and 'manual' triggers.

Reset D.

Load l's into S, T, and LSWR.

Transfer contents of DATA switches to ST.

Transfer T(32-38) and T(54-63) to MCW.

Reset 'stop' and 'manual' triggers.

Reset D.

Transfer contents of DATA switches to ST.

Set 'scan mode' trigger.

Transfer T(32-38) and T(54-63) to MCW.

RESET

Set 'scan mode' trigger.

Transfer T(32-38) and T(54-63) to MCW.

Reset 'stop' and 'manual' triggers.

Reset D.

Load l's into S, T, and LSWR.

Transfer contents of DATA switches to ST.

Set 'scan mode' trigger.

Transfer T(32-38) and T(54-63) to MCW.

Reset 'stop' and 'manual' triggers.

Reset D.

Load l's into S, T, and LSWR.

Transfer contents of DATA switches to ST.

Set 'scan mode' trigger.

Transfer T(32-38) and T(54-63) to MCW.

Reset 'stop' and 'manual' triggers.

Reset D.

Load l's into S, T, and LSWR.

Transfer contents of DATA switches to ST.

Set 'scan mode' trigger.

Transfer T(32-38) and T(54-63) to MCW.

Reset 'stop' and 'manual' triggers.

Reset D.

Load l's into S, T, and LSWR.

Transfer contents of DATA switches to ST.

Set 'scan mode' trigger.
Diagram 6-25. LOG OUT Pushbutton Logic

Diagram 6-26. SCAN MODE, ROS/PROC/FLT Switch Logic
Diagram 6-115
ROS Test sequence.

Diagram 6-116
FLT sequence.

KW371
CE in FLT or EOS test mode and FLT BACKSPACE depressed.

KW371
Reset 'manual' and 'pass pulse' triggers.

KU291
Subsystem Reset.

FE321
Set Unit Channel and PSBA bits in the external register (on the Control Bus).

Enter stop loop. Wait for further operator action.

Diagram 6-115
ROS Test sequence.

Diagram 6-116
FLT Sequence

KW371
Start CE clock.

"Subsystems Reset"

CHECK

Reset pb depressed
when BACKSPACE FLT
was depressed

KU311
Set the Rewind Latch.

KX131
Set the Rewind Latch.

Send 'FLT backspace' to IOCE specified by the LOAD UNIT switches.

"FLT Backspace IOCE (X)"

No

KU311
Keep 'FLT backspace IOCE (X)' up until LOAD or RESET pb is depressed.

Reset the FLT Backspace trigger. Drop 'FLT backspace IOCE (X)'.

Diagram 6-27. FLT BACKSPACE Pushbutton Logic and Flow

"I/O CONTROL ELEMENT"

Set Stats 0 and 1 to cause IPL branch from the halt loop.

Take Control Bus data to PSBAR and to the CCW for backspace.

Issue IPL SIO, and TIO to channel.

Backspace FLT tape one record.

Prepare unit and channel addresses for PSW (0).

Send "FLT complete" to the CE.

Return to halt loop.

I/O CONTROL ELEMENT (Selected by LOAD UNIT switches)

Machine reset. Propagate reset to I/O units configured or switched to this IOCE. Enter halt loop.
Diagram 6-28. 1052 Adapter Unit
Diagram 6-29. 1052 Adapter Initial Selection — Read, Write, Sense
Diagram 6-30. 1052 Adapter Data Transfer – Write
Diagram 6-31. 1052 Adapter Data Transfer - Read

- Start read/write clock at SS 5 when a key is operated.
- Gate keyboard translator output to data register.
- Raise "request in".
- Raise "operational in" and "address in", gate address byte to "bus in" lines.
- Raise "service in" and gate data register output to "bus in" lines when "command out" falls.
- Start read/write clock at SS 3 time (to operate printer) when channel accepts data byte replying "service out" to "service in".

Objectives:
1. Start read/write clock at SS 5 when a key is operated.
2. Gate keyboard translator output to data register.
3. Raise "request in".
4. Raise "operational in" and "address in", gate address byte to "bus in" lines.
5. Raise "service in" and gate data register output to "bus in" lines when "command out" falls.
6. Start read/write clock at SS 3 time (to operate printer) when channel accepts data byte replying "service out" to "service in".
Objectives:
1. Set 'stop' and 'channel end' latches when 'command out' line indicates 'Stop.'
2. Start the read/write clock if carrier is to be returned.
3. Set 'store device end' latch when carrier is through moving (returning).
4. Set 'device end' latch:
   A. At the same time as the 'channel end' latch if the carrier is not to be returned;
   B. After channel status has been accepted by the channel, and the carrier finishes returning.
5. Raise 'request in' for any pending status interrupt conditions.
6. Set 'channel end' and 'device end' latches during initial selection for either control command.
7. Set 'channel end' and 'device end' latches during sense byte transfer for Sense command.

Diagram 6-32. 1052 Adapter Ending Sequence
Objectives:
1. Set equipment check latch if:
   A. Printer parity disagrees with keyboard parity bit during a Read command;
   B. Keyboard output is not odd parity;
   C. Printer fails to take a mechanical cycle when directed to print, up- or down-shift, tab, space, or backspace.

2. Set 'unit check' latch when 'status in' trigger is on for any of the following conditions:
   A. 'Equipment check' latch is on
   B. 'Ready' latch is not on
   C. 'Command reject' latch is on
   D. 'Bus out check' latch is on (even parity byte on 'bus out' lines).

Diagram 6-33. 1052 Adapter Sense and Status Bytes
Diagram 6-102. Scan Clock
Diagram 6-103. FLT Clock

Note: All clock and not-clock signals to FLT clock are from scan clock.
‡ in the ALD, this signal is 'gate ST bus to scan ctr'.
Diagram 6-104. Scan Counter Latches and Decrementer

A

L
KU101

B
L
KU111

C
L
KU121

D
L
KU131

E

F

G

H

2

3

4

5

6

Not 0
Not 1
Not 2
Not 3
Not 4

To Address Seq (6), FLT Ctr (0)
To Address Seq (1), FLT Ctr (1), ROS Test Seq (6)
To Address Seq (2), FLT Ctr (2), ROS Test Seq (1)
To Address Seq (3), FLT Ctr (3), ROS Test Seq (2)
To Address Seq (4), FLT Ctr (4), ROS Test Seq (2)
Diagram 6-105. Scan Storage Address Generator

Note: Scan-generated main storage addresses start with the highest value required and are decremented to the lowest value required.
Diagram 6-106. FLT Counter Decrementing
Diagram 6-107. Scan-Out Bus Data Flow
When the Diagnose instruction specifies a log on count, the cycle counter decrements each cycle, and when the cycle counter reaches zero, the 'SOROS' flag is set to initiate a logout.

Set Marks S and T
Scan Clock
Reset SCAN Controls
Enable Scan Bypass
Scan Out Right Halfword
Gate PAL to T
Set Marks for T
Force Address 019 into ROSAR

Scan words 13 through 0 are scanned under ROS control. Gate ROS Fields

Diagram 6-108. Logout Control Logic
Diagram 6-113. CE Scan/IOCE Interface
Initiate a logout.

1. Set External Logout Request via WOO.
2. Set External Logout Latch.

3. Diagnose and MCN(6) and cycle counter = 0.
5. Set 'error' trigger.
6. Stop CE clock at following not clock time.

- Set 'MMSC' trigger.
- Send ELC to all CE's.
- Inhibit D and IC storage requests.
- Block roller switches with 'enable SCAN bypass' signal for words 21-14.

- Set 'scan Contents of S are not stored' mode' trigger.
- Force 19 to ROSAR.
- Reset 'MMSC' trigger.

- Decrease address sequencer.
- Scan out right-half word to T.
- Set marks and store T.
- Contents of T are not stored.
- Address sequencer = 14.

- Set 'scan word' trigger.
- Force 19 to ROSAR.
- Reset 'MMSC' trigger.
- Decrease address sequencer.

- No.
- Yes.

- Set address sequencer = 23.
- Starting address for microprogram.
- Start CE clock.

- Diagram 6-114. Logout Sequence (Sheet 1 of 2)

Note: LAO's 6521-6561 and Diagram 6-117 identify contents of each logout word.
Diagram 6-114. Logout Sequence (Sheet 2 of 2)
No
ROS state 3:
fetch word 3 (TN/A TN or test) after I/OCE has placed it in buffer.

No
ROS state 4:
compare TN in 5 with last test A TN in T.

Yes (Start next test)
LOAD
publication suppressed

Yes
TIC or GAP

No
FLY
BACKSPACE suppressed twice and CMD depressed

Yes (Restart last test)
FLY
BACKSPACE suppressed twice and CMD depressed
CE Clock:
Unsymmetrical (80 ns + 120 ns).
Controlled by 'maintenance mode stop clock' trigger or by 'Pass Pulse' trigger during ROS Tests.

Scan Clock:
Symmetrical (100 ns + 100 ns).
Controlled by the 'Pass Pulse' trigger.

Note: The rise of PO of the Scan Clock always coincides with the rise of PO of the CE Clock when both are running.

ROS Test State is always decremented at Scan Clock PO time of FLT Clock Time 3 except when in State 6. State 6 requires PO of FLT Clock 3 plus TIC latch on.

Diagram 6-115. ROS Test Sequence (Sheet 2 of 5)
Objective: Set up subsystem to run ROS Tests.

**Manual Operations**

- Configure a maintenance subsystem (ICE to be tested, SE, IOCE and TCU) in State 0.
- Mount ROS Test tape.
- Set LOAD UNIT switches to select IOCE, channel, and tape only.
- Set MAIN STORAGE SELECT switch to the number of the configured SE.
- Set the TEST switch on (down).
- Set the SCAN MODE, ROS/PROC/FLT switch to ROS.

**Diagram 6-115. ROS Test Sequence (Sheet 3 of 5)**

- Force CE into the stop loop, 'manual' trigger on.
- Set CE CHECK CONTROL switch to DSBL.
- Set FLT IPL latch.
- Set 'Release' trigger.
- Set 'Start ROS Test' latch, "Manual" trigger on.
- Set 'Scan Counter Control' trigger and latch. "Buffer 1" trigger and latch all of the MCW.
- Execute CCW at location 8: No-op and chain command.
- Execute CCW at location 10: read the first eight bytes of record 2 (Loader loop) into PSA0, Chain data.
- Execute CCW at location 18: read 8 bytes into loc 0, Chain data.
- Execute CCW at location 28: TIC to location 20.
- Execute CCW at location 30: read 8 bytes (hex) into buffer 2 (location 200). Chain data.
- Execute CCW at location 38: TIC to location 20.
- Send GAP to CE.
- Return to the halt loop (received subsystem reset from CE).
- TIC releases CE to execute test in buffer 1.
- TIC releases CE to execute test in buffer 2.
- FLT Clock Time 3

**Diagram 8-27**

- Received IPL IOCE (0), load unit and main storage addresses
- First IPL Yes
- Read 24 bytes into SE starting at PSA 0 (location 0 of the selected SE).
- Execute CCW at location 5: read the first eight bytes of record 2 (Loader loop) into PSA0, Chain data, Chain command.
- Execute CCW at location 15: No-op and chain command, Chain data, Chain command.
- Execute CCW at location 150: No-op and chain command, Chain data, Chain command.

**Backspace FLT Pushbutton Flow**

- Set 'Sync' trigger and latch, 'Buffer 1' trigger and latch all of the MCW.
- Set 'Scan Counter Control' trigger and latch. "Buffer 1" trigger and latch all of the MCW.
- Execute CCW at location 8: No-op and chain command.
- Execute CCW at location 10: read the first eight bytes of record 2 (Loader loop) into PSA0, Chain data.
- Execute CCW at location 18: read 8 bytes into loc 0, Chain data.
- Execute CCW at location 28: TIC to location 30.
- Execute CCW at location 30: read 8 bytes (hex) into buffer 2 (location 200). Chain data.
- Execute CCW at location 38: TIC to location 30.
- Send GAP to CE.
- Return to the halt loop (received subsystem reset from CE).

---

**Diagram 6-115. ROS Test Sequence (Sheet 3 of 5)**
Diagram 6-115. ROS Test Sequence (Sheet 4 of 5)
Diagram 6-115. ROS Test Sequence (Sheet 5 of 5)
Set up subsystem to run FLTs.

Initiate IPL, and pass control to the IOCE.

IOCE channel loads record 1 or 4 into storage, reads in record 2, and releases CE.

ROS-controlled operations: set up CE to run FLTs.

Non-ROS-controlled operations: set up CE to run FLT microprograms.

ROS operations: test for input error from IOCE. Set address sequencer to 16. Determine if test is in storage (TIC pulse). Determine if alternate test is to be fetched by comparing ATN with next TN.

Scan in: address sequencer is decremented from 15 to 0; test words are placed into S, T, or Q, and the data is distributed throughout the CE under microprogram control.

Test cycle: CE cycles until the FLT counter equals 0 ('exit' trigger state is changed).

Scan Out: the exit trigger condition is scanned out to T.

Result comparison: perform scan-out S and T, set 'pass' and 'fail' triggers per PAL and ERSLT bit.

Terminate or continue: stop or continue per UT or CT bits and 'pass' and 'fail' triggers.

Enter stop and restart routines.

Note: This is a summary flow chart. For detailed FLT sequence refer to sheets 2 - 5.

Diagram 6-116. FLT Sequence (Sheet 1 of 5)
**MANUAL OPERATIONS**

- Configure a maintenance subsystem (ICE to be tested, SE, IOCE and TCU) in state 0.
- Mount FLT Test tape.
- Set LOAD UNIT switches to select IOCE channel, and tape unit.
- Set the MAIN STORAGE SELECT switch to the number of the configured SE.
- Set the TEST switch on (above).
- Set the SCAN-MODE, ROU/PROC/FLT switch to FLT.

**DIAGRAM 6-27**

**BACKSPACE FLT pushbutton flow.**

**A**

- Degrees SYSTEM RESET pushbutton.
  - Force the CE into the stop loops manually! Trigger on.

**B**

- SCAN NODE in FLT TEST on, CE in state 0 and Menos.
  - Yes: KU291
  - No

**C**

- **MANUAL OPERATIONS**
  - Set CE CHECK CONTROL switch to DSBL.
  - Depress LOAD pushbutton.

**E**

- Set 'Pass Pulse' trigger.
  - KW441
  - Enable the IPL PWL branch.

**F**

- Branch from the stop loop.

**H**

- Diagram 6-116. FLT Sequence (Sheet 2 of 5)

**OBJECTIVES:**

- Set up subsystem to run FLTs.

---

**I/OCE**

- (Main loop)
  - YES: Send 'IPL IOCE (X)' to the selected IOCE.
  - NO: Initiate IOCE operations.

**D**

- Set FLT IPL latch.
  - Send 'IPL IOCE (X)' to the selected IOCE.

**G**

- Sheet 3

---

- **DIAGRAM 6-116.**

---

**H**

- Sheet 3
Objective: Perform ROS-controlled scan into CE hardware (from buffer) into CE hardware.

Algorithm: Buffer is scanned into CE hardware (from buffer) into CE hardware.

Procedure:
1. Set the address sequencer to 16.
2. Fetch word 16 into S.
3. Gate S to K.
4. Scan into miscellaneous triggers.
5. Decrement address sequencer.
6. Fetch word 15 into ST.
7. Gate T to external register.
8. Scan into miscellaneous triggers.
9. Decrement address sequencer.
10. Fetch word 14 into ST.
11. Gate T to DAR mask.
12. Decrement address sequencer.
13. Fetch word 13 into ST.
14. Gate T to DAR mask.
15. Decrement address sequencer.
16. Fetch word 12 into ST.
17. Gate T to DAR mask.
18. Decrement address sequencer.
19. Fetch word 11 into ST.
20. Gate T to DAR mask.
21. Decrement address sequencer.
22. Fetch word 10 into ST.
23. Gate T to DAR mask.
24. Decrement address sequencer.
25. Fetch word 9 into ST.
26. Gate T to DAR mask.
27. Decrement address sequencer.
28. Fetch word 8 into ST.
29. Gate T to DAR mask.
30. Decrement address sequencer.
31. Fetch word 7 into ST.
32. Gate T to DAR mask.
33. Decrement address sequencer.
34. Fetch word 6 into ST.
35. Gate T to DAR mask.
36. Decrement address sequencer.
37. Fetch word 5 into ST.
38. Gate T to DAR mask.
39. Decrement address sequencer.
40. Fetch word 4 into ST.
41. Gate T to DAR mask.
42. Decrement address sequencer.
43. Fetch word 3 into ST.
44. Gate T to DAR mask.
45. Decrement address sequencer.
46. Fetch word 2 into ST.
47. Gate T to DAR mask.
48. Decrement address sequencer.
49. Fetch word 1 into ST.
50. Gate T to DAR mask.
51. Decrement address sequencer.
52. Fetch word 0 into ST.
53. Gate T to DAR mask.
54. Decrement address sequencer.

Diagram 6-116. FLT Sequence (Sheet 3 of 5)
Diagram 6-116. FLT Sequence (Sheet 4 of 5)
Objectives:
1. Scan-out the half logword which includes the trigger being tested
   (specified by the MCW address sequencer),
2. Fetch mask from storage.

Notes:
1. To restart after a step, depress FLT BACKSPACE twice, then the LOAD
   Pushbutton once. This will bring in the next test and run it (Diagram 6-27).
2. To loop on failing test, operator must place SCAN MODE REPEAT Switch
down before depressing START pushbutton.

Diagram 6-116. FLT Sequence (Sheet 5 of 5)
<table>
<thead>
<tr>
<th>LOG WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
<th>ISA WORD NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

**Table: Bit Positions in Main Storage**

**Comments:**

- Log Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0
- ISA Word No. 0

**Diagram 6-117. CE Logword Formats (Sheet 1 of 3)**
•

LOG
WORD
NO.

PSA
LOCATION

PSA
WORD
NO.

192
(CO)

48

4

6

5

9

8

7

BIT POSITIONS IN MAIN STORAGE
COMMENTS

0

3

4

12

p
9-12

PHYSICAL PSBAR

9

8

•

3

2

10

11

5

6

7

8

11

p
9-15

10

10

11

12

13

14

15

16

17

18

19

20

LOGICAL PSBAR

PSBAR COUNTER

9

9

9

10

11

12

13

14

15

16

17

22

23

25

24

26

27

PROGRAM STATUS WORD
SYSTEM MASK

ALT
PS BAR

19

18

21

17

16

18

D

19

28

29

SE
STPD

l

2

3

24

25

26

27

31

_;30
PSA LOCKOUT

PS BAR

CONFIGURATION CONTROL REGISTER (CCR)

A
196

STATE

49
So

SCON
S1

2

!LOS

3

4

SE/DE

1/

6

4/

3/

2/

5/

IOCE

CE
6/1

7/2

8/3

9/4

18

10/5

19

3

4

24

3

23

25

26

OTC

OBS

SPARE
26

24

25

26

9

10

28

27

CCR

3

DIAGNOSE ACCESSIBLE REGISTER MASK
200
(CB)

IOCE

50
SPARE

SE/DE

SPARE

SPARE

3

l/

2/

3/

4/

5/

6/1

7/2

9/4

8/3

10/5

SPARE
16

SERIAL ADDER LATCHES (SADDL)
204

1/1

4

CE OWN

CE
4

SPARE
31

DAR MASK

INSTRUCTION COUNTER (IC)

5

6

7

8

9

10

11

12

13

14

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

22

21

R REGISTER

2

3

4

5

15

4

3

2

0

F REGISTER (READ DIRECT)

B

27

28

29

30

31

11

12

13

14

15

E REGISTER

52

10

6

5

7

8

3

4

R REG,
E REG

D REGISTER
DREG,
F REG

53
2

216
(D8)

3/

SADDL, IC

0

212

TCU

2/2

51
2

208
(DO)

PAM/RCU

SPARE
17

5

6

7

8

10

9

11

12

14

13

15

16

17

18

19

20

21

22

23

24

25

26'

27

28

29

30

31

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

3

OTC

4

SPARE
31
SADD
FULL
SUM
CHK

31

Q REGISTER
Q REG

54

11

2

3

34

35

4

5

6

37

38

7

8

9

10

11

12

39

40

41

42

43

44

13

14

15

45

46

47

16

Q REGISTER

220

Q REG

55

33

32

48

DIAGNOSE ACCESSIBLE REGISTER (DAR)

224

(EO)

36

IOCE2

IOCEl

56

IOCE3

A

A

SE/DE ELC

I/

A

2/

3/

5/

4/

6/1

7/2

9/4

8/3

12

10/5

SPARE
16

SPARE
17

2/2

CE OWN

TCU ELC

PAM/RCU ELC

1/1

3/

2

OBS

CE ELC

SPARE
26

3

CHECK REGISTER l

c
228

232

(EB)

PARALLEL ADDER FULL SUM-CHECK

57
4-7

8-15

p
0-7

p
8-15

16-23

24-31

32-39

40-47

p

E REG
PTY
0-7

PADD
ULLSU
.64-67

13

48-55

16-23

CHECK REG 1

LSWR
58

p
16-23

24-31

2-42

43-68

69-99

SADD
HALF
SUM
CHK

26

27

28

29

30

E
12-15
TO
PADDB

E
8-11
TO
PAD DB
60-63

ROS PARITY CHEC,K

. PARALLEL ADDER HALF-SUM CHECK

N REG

EXT TGRS
PERMIT
IOCE
IOCE
INTRPT
1
REQ

PSW REG

NO
RETRY

p

p

p

p

0-7

8-15

0-7

8-15

SVC
INTRPT

PHYS
PSBAR
PTY
9-12

24-31

PSBAR LOGICAL
CNTR
PSBAR
PTY
12
16-19

32-39

40-47

48-55

SADDL
p
0-7

p
8-15

p

p

16-23

24-31

23

24

25

EX 6
TO
PADDB
28-63

HOT
ONE TO
PAD DB
60

E
8-11

56-67

DAR

CRl

IC REG

LOCAL STORAGE WORK! NG REG! STER (LSWR)
236

59

LSWR
0

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

IC
8-31
TO
PADDB
40-63

AB
4-7
TO
PADDB
4-7

AB
8-31
TO
PAD DB
8-31

AB
32-63
TO
PADDB
32-63

AB
64-67
TO
PADDB
64-67

AB
6-31
TO
PADDB
4-29L2

AB
32-67
TO
PADDB
30-65L2

6f-~7

240
(FO)

60

14

T

T32-63 TO PADDA

D
244

61
32-63

248
(F8)

62

252

63

32-63
COMP

31-62
L1

32-47
31-62
C-Ll

PA1~A
48-63

T

48-63
TO
PADDA
48-63

K

0-31
TO
PADDA
32-63

DB-31 TO PADDA
8-31

7-30
L1

7-30
C-L 1

40-63

40-63
COMP

FMTO

FMTM

FMTW

TO
PADDB
28-31

TO
PAD DB
56-59

60-63

Q

Q

Q

Q

4-15
PAD DB
52-63

20-31
TO
PAD DB
52-63

36-47
TO
PADDB
52-63

52-63
TO
PADDB
52-63

RIGHT
DIGIT
TGR

TGR

LEAVE
TGR

STEP
ABC
TGR

TO

15
READ ONLY STORAGE DATA REGISTER (ROSDR)
70

-...J

2

72

73

Diagram 6-117. CE Logword Formats (Sheet 2 of 3)

,-.
-...J

71

E

74

75

76

77

79

80

81

82

83

84

p
43-68

86

88

89

90

p
69-99

92

93

94

95

97

98

99

s

ROS DR


**Diagram 6-11B. SE Logword Formats**

| WORD NO. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| COMMENTS |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A        |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B        |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C        |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D        |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E        |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Legend:**
- **SESDR** (EVEN) = SE storage device register (EVEN)
- **SESDR** (ODD) = SE storage device register (ODD)
- **SESR** (EVEN) = SE status register (EVEN)
- **SESR** (ODD) = SE status register (ODD)
- **SESR** (EVEN) = SE status register (EVEN)
- **SESR** (ODD) = SE status register (ODD)
- **MARK** = Mark
- **TYPE OF OP** = Type of operation
- **CHK** = Check
- **OP** = Operation
- **SAB** = Status register (EVEN)
- **SBO** = Status register (ODD)
- **OBC** = Operation code
- **SPP** = Status register (ODD)
- **SPO** = Status register (ODD)
- **SHR** = Shift register
- **SMR** = Storage memory register
- **TAG** = Tag
- **TAG** = Tag
- **LOD** = Load

**Bit Positions in Main Storage:**
- WORD BIT POSITIONS IN MAIN STORAGE
- WORD BIT POSITIONS IN MAIN STORAGE
- WORD BIT POSITIONS IN MAIN STORAGE
- WORD BIT POSITIONS IN MAIN STORAGE
- WORD BIT POSITIONS IN MAIN STORAGE

**Comments:**
- SESDR (EVEN) = SE storage device register (EVEN)
- SESDR (ODD) = SE storage device register (ODD)
- SESR (EVEN) = SE status register (EVEN)
- SESR (ODD) = SE status register (ODD)
- MARK = Mark
- TYPE OF OP = Type of operation
- CHK = Check
- OP = Operation
- SAB = Status register (EVEN)
- SBO = Status register (ODD)
- OBC = Operation code
- SPP = Status register (ODD)
- SPO = Status register (ODD)
- SHR = Shift register
- SMR = Storage memory register
- TAG = Tag
- TAG = Tag
- LOD = Load
Diagram 6-11. DE Logword Formats

<table>
<thead>
<tr>
<th>BITPOSITIONS IN MAINSTORAGE</th>
<th>WORD 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO.</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>MARKS (EVEN)</td>
<td>0-7</td>
<td>8-15</td>
<td>16-23</td>
<td>24-31</td>
<td>32-39</td>
</tr>
<tr>
<td>T &amp; S STORE FETCH DATA CHK ADDR CHK MARK PARITY CANCEL CHK LATCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARKS (ODD)</td>
<td>0-7</td>
<td>8-15</td>
<td>16-23</td>
<td>24-31</td>
<td>32-39</td>
</tr>
<tr>
<td>DESD R (ODD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHECK CONDITIONS (COMMON)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DG ADDR REG ID CHECKS DG ADDR REG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA REG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram 6-119. DE Logword Formats

<table>
<thead>
<tr>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES DR (EVEN) DES DR (EVEN) MARKS DESD R, DESAR (EVEN) CHECKS (EVEN AND COMMON) SPIKE, SPOKE</td>
</tr>
<tr>
<td>DESDR (ODD) DESD R (EVEN) MARKS DESD R, DESAR (ODD) CHECKS (ODD) CCR RESPONSE LATCHES</td>
</tr>
</tbody>
</table>
The chart at the left shows the timing for transferring one quadword of data to the CE via the DE Wrap Bus. The Diagnose instruction controls the total number transferred by setting the quadword count in the MCW.

Note: The output of this latch stops the CE clock to compensate for the difference in timing between the CE clock and the sample pulses received from the DE. One CE clock cycle is 200 ns while one sample pulse is 225 ns. Note that the CE clock is not allowed to run until the Data latch turns on.
DAR: Data Flow 2-1
Usage (Diagram): 5-609
DAR MASK: Data Flow 2-1
Usage (Diagram): 5-609
Data Flow: Branching Instructions 3-4
CE 2-1
Direct Instructions 3-3
Direct Control Operation 5-607 (Sheet 2)
Divide, Fixed-Point 5-110 (Sheet 4)
Divide, Floating-Point 5-213
Fixed-Point Instructions 3-1
Floating-Point Instructions 3-2
I/O Instructions 3-6
Logical Instructions 3-3
Multiply, Fixed-Point 5-109 (Sheet 3)
Multiply, Floating-Point 5-213
Fixed-Point Instructions 3-1
Floating-Point Instructions 3-2

I/O Instructions 3-7
Logical Instructions 3-3
Multiply, Fixed-Point 5-210
RR I-Fetch, One-Cycle 5-7
RR I-Fetch, Two-Cycle 5-8
RX I-Fetch, One-Cycle 5-10
RX I-Fetch, Two-Cycle 5-12
RX I-Fetch, Two-Cycle Indexed 5-11
RX I-Fetch, Two-Cycle Nonindexed 5-12
SI I-Fetch, One-Cycle 5-10
SI I-Fetch, Two-Cycle 5-12
Status Switching Instructions 3-5

E-Register Incrementer 3-6
Bits 14 and 15, Positive Logic Diagram 4-203
Parity Prediction After Incrementing 4-204
Early End I/F, Instruction Request 5-3
Edit, ED (DE); Logic, SS 5-411
Edit and Mark, EDM; Logic, SS 5-411
EEOP Request Trigger 5-50
EE(15) Parity Trigger 4-204
End Op:
Instruction Requests 5-2
Operand Processing 5-1

Error Controls 6-22
Error Trigger 6-400, 6-413, 6-414, 6-22, 6-108
Exceptional Conditions 6-22

CPU Store In Progress 5-18
Invalid Instruction Address Test 5-29
Manual Control 5-27
Program Store Compare 5-28
Q-Register Refill 5-30
Timer 5-17

Exclusive-4 Logic, Parallel Adder 4-415
Exclusive-OR Function, SAL(0-3) 4-406
Exclusive-OR, X (17); Logic, RX 5-406
Exclusive-OR, X (27); Logic, SS 5-406
Exclusive-OR, X (97); Logic, SI 5-406
Exclusive-OR, XR (17); Logic, RR 5-406
Execuse, EX(44); Be, RX 5-506

External Intersections 5-24
External Register:
Data Flow 2-1
Usage:
SAR 5-808, 809
SCON 5-810
SDIP 5-807
External Signal Flip-Latches 5-607
External Trigger 5-24

F-Register:
Data Flow 2-1
Usage:
Add, Subtract, and Compare, Decimal 5-502 (Sheet 1)
Direct Control Operation 5-607 (Sheet 2)
Divide, Decimal 5-506 (Sheet 1)
Divide, Fixed-Point 5-110 (Sheet 4)
Divide, Floating-Point 5-110 (Sheet 4)
Floating-Point Instructions 3-1
Floating-Point Instructions 3-2
Multiply, Decimal 5-305 (Sheet 1)
Multiply, Fixed-Point 5-109 (Sheet 3)
Multiply, Floating-Point 5-210
Status Switching Instructions 3-5

Full Triger 6-111, 6-112
Fault Locating Test (see FLT)
Final Error Latch 4-413, 4-414
Floating-Point Instructions 3-7

Add, A(5A); RX 5-108
Add, A(1A); RX 5-108
Add HalfWord, AH(4A); RX 5-108
Add Logical, AL(58); RX 5-108
Add Logical, ALK(15); RR 5-108
Compare, C(59); RX 5-108
Compare, CR(15); RX 5-108
Compare HalfWord, CH(49); RX 5-108
Convert to Binary, CVB(47); RX 5-111
Convert to Decimal, CVD(48); RX 5-112
Data Flow 3-1
Divide:
Algorithm 5-110 (Sheet 3)
Data Flow 5-110 (Sheet 4)
Initialization 5-110 (Sheets 1 and 2)
Termination 5-110 (Sheets 1 and 2)
Divide, Decimal 5-506 (Sheet 2)
Divide, DR (1D); RR 5-110 (Sheet 1)
Divide, Floating-Point 5-210
Algorithm 5-110 (Sheet 3)
Data Flow 5-110 (Sheet 4)
Initialization 5-110 (Sheets 1 and 2)
Termination 5-110 (Sheets 1 and 2)
Divide, Floating-Point 5-213
Division with Transfer of Low-Order; Fixed-Point 5-110 (Sheet 4)
Divisor Multiple, Division of; Fixed-Point 5-110 (Sheet 4)
Drive Line Decoders, ROS 4-105 (Sheet 1)

F-Register:
Data Flow 2-1
Incrementer, with 15 and 14 5-203
Parity Prediction After Incrementing 4-204
Usage:
Branch Requests 5-4
Branching Instructions 3-4
Decimal Instructions 3-3
Direct Control Operation 5-607
Divide, Floating-Point 5-213
Fixed-Point Instructions 3-1
Floating-Point Instructions 3-2

Shift Left Single, SLA (8B); RS 5-116
Shift Right Single, SRDA (8E); RS 5-119
Halve, HDR (24); Fl Pt, RR (Long) 5-108
Subtract, SR (18); RR 5-108

Floating-Point Instructions:
Add Normalized, AD (6A); RX (Long) 5-207
Add Normalized, ADD (4A); RR (Long) 5-207
Add Normalized, AER (3A); RR (Short) 5-206
Add Unnormalized, AU (7E); RX (Short) 5-206
Add Unnormalized, AUR (3E); RR (Short) 5-206
Add Unnormalized, AE (6E); RX (Long) 5-207
Add Unnormalized, AER (3E); RR (Long) 5-207

CC Setting 5-201
Compare, CD (69); RX (Long) 5-207
Compare, CD (79); RR (Long) 5-206
Compare, CER (79); RR (Short) 5-206
Data Flow 2-1
Divide, Data Flow 5-213
Divide, DD (60); RX (Long) 5-215
Divide, DDR (22); RR (Long) 5-215
Divide, DE (70); RX (Short) 5-214
Divide, DDR (22); RR (Short) 5-214
Halve, HX (24); RX (Long) 5-209
Halve, HX (24); RR (Short) 5-208
Load and Test, LTDR (22); RR (Long) 5-205
Load and Test, LTDR (22); RR (Short) 5-205
Load Complement, LCMR (33); RR (Short) 5-204
Load, LD (68); RX (Long) 5-203
Load, LDR (28); RX (Long) 5-202
Load, LE (78); RX (Short) 5-203
Load, LER (38); RX (Short) 5-202
Load Positive, LPDR (20); RX (Long) 5-205
Load Positive, LPDR (20); RR (Short) 5-204
Load Positive, LPDR (20); RR (Short) 5-204
Load Positive, LPDR (20); RR (Short) 5-204
Load Positive, LPDR (20); RR (Short) 5-204
Load Positive, LPDR (20); RR (Short) 5-204
Load Positive, LPDR (20); RR (Short) 5-204
Load Positive, LPDR (20); RR (Short) 5-204
Load Positive, LPDR (20); RR (Short) 5-204
Multiply, MD (6C); RX (Long) 5-212
Multiply, MDR (1C); RR (Long) 5-212
Multiply, ME (7C); RX (Short) 5-211
Multiply, MDR (1C); RR (Short) 5-211
Save Signs and Insert Sign Function 5-201
Store, STD (60); RX (Long) 5-216
Store, ST (50); RX (Long) 5-216
Subtract Normalized, SD (68); RX (Long) 5-207
Subtract Normalized, SE (78); RX (Short) 5-206
Subtract Normalized, SFR (38); RR (Short) 5-206
Subtract Unnormalized, SU (7F); RX (Short) 5-206
Subtract Unnormalized, SFR (38); RR (Short) 5-206
Subtract Unnormalized, SW (6F); RX (Long) 5-207
Subtract Unnormalized, SWR (3F); RR (Long) 5-207

FLT:
Backspace Pb 6-27
Clock 6-109
Counter, Decrementing 6-106
Sequence 6-116
FLT Counter * Latch 6-106
FLT Test Trigger 6-26, 6-111
Force Address Latch 4-1, 6-7
Fraction Data Path:
Floating-Point Divide 5-213
Floating-Point Multiply 5-210

FREQUENCY ALTERATION Switch, Clock Signal Generator 4-3
FullSum Checking Logic, PA(48-55) 4-514
FullSum Error Trigger 4-413
Half-Sum Precheck Error (48-55) Trigger 4-413
Half-Sum Checking Logic, PA(48-55) 4-514
Half-Sum Error Trigger 4-413
Halve, HDR (24); Fl Pt, RR (Long) 5-209

Halve, HDR (24); Fl Pt, RR (Short) 5-208
Hold In Flip-Latch 5-607 (Sheet 2)
Hold MC IFL Latch 6-9

I-Fetch:
Block F-Fetch Trigger 5-14
Branch Requests 5-4
Common Instruction Routine 5-26
CPU Store in Progress Exceptional Condition 5-18
External Interruptions 5-24
Instruction Requests During Early End Op 5-3
Instruction Requests During Early End Op 5-3
Interrupt Code Triggers 5-26
Nonbranch Setting of 5-20
Program Interruption Code 5-22
Spec-Y Branch Setting of 5-21
Invalid Instruction Address Test Exceptional Condition 5-29
I/O Interruption 5-23
Machine Check Interrupt 5-19
Manual Control Exceptional Conditions 5-27
Openad Prefetching During End Op 5-1
Program Interruptions 5-22
Program Compare Exceptional Condition 5-28
Q-Register Refill Exceptional Condition 5-30
RR, Flowchart 5-6
RR, One-Cycle 5-7
RR, Two-Cycle 5-8
RS, Flowchart 5-13
RS, One-Cycle 5-10
RS, Two-Cycle 5-12
RX, Flowchart 5-9
RX, One-Cycle 5-10
RX, Two-Cycle Indexed 5-11
RX, Two-Cycle Nonindexed 5-12
Sequence, Selection of 5-5
Sequential 5-15
SI, Flowchart 5-13
SI, One-Cycle 5-10
SI, Two-Cycle 5-12
SS, Flowchart 5-14
Superimposed Call Interrupt 5-23
Time Exceptional Condition 5-17
I-Fetch Invalid Address Trigger 5-29 (Sheet 1)
I-Fetch Request Trigger 5-29 (Sheet 1)
IC:
Address Getting, BCU 4-607
Data Flow 2-1
Parity Adjustment 4-205
Usage:
Add, Subtract, and Compare, Decimal 5-302 (Sheet 1)
Binary Transfer to LAL 4-201
Branching Instructions 3-4
Decimal Instructions 3-5
Divide, Decimal 5-306 (Sheet 1)
Fixed-Point Instructions 3-1
F-Fetch Sequence Control 5-15
Instruction Requests During Early End Op 5-3
Instruction Requests During End Op 5-2
Logical Instructions 3-3
Multiply, Decimal 5-305 (Sheet 1)
Operand Prefetching During End Op 5-1
RR I-Fetch, One-Cycle 5-7
RR I-Fetch, One-Cycle 5-10
RX I-Fetch, One-Cycle 5-10
RX I-Fetch, Two-Cycle Indexed 5-11
Selection of F-Fetch Sequences 5-5
SI I-Fetch, One-Cycle 5-10
Status Switching Instructions 3-5
IC (26-27) Flip-Latch 5-204
IC Request Trigger 4-601
IC Sync-Latch 4-601, 5-13 (Sheet 1)
IC Trigger 4-601 (Sheet 2)
Inhibit LS Write Trigger 5-21
Initial Program Load (See IFL) 4-601
Input/Output (see I/O)
Insert Character, IC(43); Logic, RX 5-408
Insert Key Trigger 4-601
Insert Storage Key, ISK (69); Stat Sw, RR 5-406
Instruction Requests:
During Early End Op 5-3
During End Op 5-2
Step Routine 6-13
Instruction Counter (see IC)
Instruction Fetching (see I-Fetch)
Instruction Length Not Available Trigger 5-20
Instruction Step Latch 6-12
Instructions (see Specific Instruction or Class)
Interrupt Code Triggers 5-20, 5-21, 5-22
Interrupts:
Common Routine 5-26
External 5-24
Gating, Block 6-6
Gating, Block 6-6
Interrupt Code Triggers 5-20, 5-21, 5-22
I/O 5-25
Machine Check 5-19
Program 5-22
Supervise Call 5-23
Superimposed Call 5-23
Invalid Address Logic, SCI 4-604
Invalid Digit Logic, Serial Adder 4-405
Invalid Instruction Address Test Exceptional Condition 5-29
Invalid Instruction Address Trigger 5-29 (Sheet 1)
MMSC Logic 6-110
ROS Microbranching, Control 6-112
Scan Clock 6-102
Scan CounterLatchs and Decrementer 6-104
Scan/DIC Interface 6-113
Sense-Out Bus, Data Flow 6-107
Scan-Out Path 6-109
Scan Storage Address Generator 6-105
Scan MODE ROG/PROC/FLT Switch Logic 6-26
Scan Mode Trigger and Latch 6-108
Scan-Out Bus, Data Flow 6-107
Scan-Out Path 6-109
Scan-Out BUS (a/a/SOROS)
Scan Request Trigger 4-601, 6-105, 6-108
Scan Storage Address Generator 6-105
Scan Sync Trigger and Latch 4-601, 6-105, 6-108
SCI:
Address Decode Logic, Basic System 4-402
Address Gating Logic 4-602
CE Request Timing, Typical 4-601
Clock Control Logic 4-603
CPU Sequencers 4-603
Data Flow 6-1
Invalid Address Logic 4-604
Request Selecting Logic 4-601
SAB Parity Conversion Logic 4-610
Single-Cycle Mode, Servicing of Requests 4-612
Single-Cycle Sign Data Path 4-606
Segmented Clock Inhibit Latch 4-2
Segmented Clock Storage Selected Latch 4-2
Select Line Decode, ROS 6-105 (Sheet 1)
Select Register:
Data Flow 3-1
Usage:
SATR 3-608, 809
SCON 5-610
Sense Amplifier, ROS 4-105 (Sheet 1)
Sense Latches, ROS:
Addressing 4-105 (Sheet 1)
Data Flow 4-105 (Sheet 2)
Sense Line and Driver Distribution, ROS 4-105 (Sheet 1)
Serial Adder:
AND Function, SLA(O) 4-406
Caref Lookahead Logic, SAL(O) 4-402
Data Flow 3-1
Decimal Add 6 Logic 4-403
Decimal Correction Logic, SAL(O) 4-302
Exclusive-OR Function, SAL(O) 4-406
Input Bus Logic 4-401
Invalid Digit Logic 4-405
Logical Function, SAL(O) 4-406
OR Function, SAL(O) 4-406
Parity Predict Logic 4-404
Product/Quotient Bit Logic 4-408
Usage:
Add, Subtract, and Compare, Decimal 5-302 (Sheet 1)
Decimal Instructions 2-1
Direct Control Operation 5-307 (Sheet 2)
Divide, Decimal 5-306 (Sheet 1)
Divide, Floating-Point 5-213
Floating-Point Instructions 3-2
Logical Instructions 3-3
Multiply, Decimal 5-305 (Sheet 1)
Multiply, Floating-Point 5-207
Multiply, Floating-Point 5-210
Status Switching Instructions 3-5
Set Program Mask, SPM(OA) 5-605
Set Storage Key, SSK(OA) 5-603
Supervisor Call, SVC(OA) 5-604
Test and Set, TS(93) 5-609
Write Direct, WRD(OA) 5-607
STC:
Bipolar Latches 4-208
Data Flow 2-1
Incrementer Latches 4-208
Positive Logic Diagram 4-208
Triggers 4-208
Usage:
Decimal Instructions 3-3
Direct Control Operation 5-307 (Sheet 2)
Divide, Decimal 5-306 (Sheet 1)
Divide, Floating-Point 5-213
Fixed-Point Instructions 3-1
Floating-Point Instructions 3-2
Logical Instructions 3-3
Multiply, Floating-Point 5-210
Status Switching Instructions 3-5
Stop Clock Trigger 4-1
Stop Loop:
Pushbutton Gating 6-5
Routine 6-4
Stop Trigger 6-6
Storage Error Logic, SCI 4-606
Storage Ripple Loop Routine 6-18
STORAGE SELECT Switch Gating 6-10
Store Character, STC(42); Logic, RX 5-408
Store Halfword, STH(OA); Fix Pt, RS 5-114
Store Latch 4-209
Store Multiple, STM(OA); Fix Pt, RS 5-115
Store, ST(OA); Fix Pt, RX 5-113
Store, STD(OA); Fix Pt, RX (Long) 5-207
Store, STE(OA); Fix Pt, RX (Short) 5-216
Store to LCS Flip-Latch 5-18
Store to Memory Storage Flip-Latch 5-18
Subtract Halfword, SSH(OA); Fix Pt, RX 5-108
Subtract Logical, SL(OA); Fix Pt, RX 5-108
Subtract Logical, SLR(OA); Fix Pt, RR 5-108
Subtract Normalized, SN(OA); Fix Pt, RX (Long) 5-207
Subtract Normalized, SNR(OA); Fix Pt, RR (Long) 5-207
Subtract Normalized, SRE(OA); Fix Pt, RX (Short) 5-207
Subtract Normalized, NRE(OA); Fix Pt, RR (Short) 5-207
Subtract, S(OA); Fix Pt, RX 5-210
Subtract, SF(BF); Dec, SS:
Complement Add Sequence 5-303
GIS 5-501
Tie Add Sequence 5-302
Subtract, SR(18); Fix Pt, RR 5-108
Subtract Unnormalized, SUR(3F); Fix Pt, RX (Short) 5-206
Subtract Unnormalized, SUR(3F); Fix Pt, RR (Short) 5-206
Subtract Unnormalized, SW(RF); Fix Pt, RX (Long) 5-207
Subtract Unnormalized, SW(RF); Fix Pt, RR (Long) 5-207
Supervisor Call, SVC(OA) 5-604
Supervisor Call Interruption 5-23
Supervisor Call Trigger 5-23
Sync Latch 6-108, 6-111
Sync Trigger 6-27, 6-108, 6-111
T-Register (see ST-Register)
1052 Adapter (9020E Only):
Block Diagram 6-28
Data Transfer-Read 6-31
Data Transfer-Write 6-30
Ending Sequence 5-37
Initial Selection-Read, Writ, Sense 6-29
Sense and Status Bytes 6-33
Test and Set, TS(93); Stat Sw, SI 5-811
Test Under Mask, TM(91); Logic, SI 5-407
TIC Latch 6-112, 6-113
Time Clock at Limit Trigger 5-24
Time Clock Step Trigger 5-17
Time Exceptional Condition 5-17
Timing Gate Trigger 3-6, 5-407 (Sheet 2)
Transfer In Channel (Log TIC)
Translate, TR(OC); Logic, SS 5-410
Unpack, UNPK (F3); Dec, SS:
    GS  5-307
Not Word Overlap Sequence  5-310
Word Overlap Sequence  5-311

Wait State Gating and Microprogram  6-20
Wait State Trigger  6-20
Word Overlap Sequence:
    Move With Offset, MVO (F1); Dec, SS  5-313
    Pack, PACK (F2); Dec, SS  5-309
    Unpack, UNPK (F3); Dec, SS  5-311
Write Direct, WRD (84); Stat Sw, SI  5-607
Write Local Store Trigger  4-301

XY-Register:
    Data Flow  2-1
    Usage:
        Convert Weather Lines  5-905
        LM to XY Formatting  4-211
        Repack Symbols  5-902
        XY Parity Prediction  4-212

Zero and Add, ZAP (F8); Dec, SS  5-304