Figure 1. RECOMP II System Block Diagram
PREFACE

This manual is one of a set of five published by Autonetics to provide maintenance information on the RECOMP II computer system.

System Service Manual: Publication No. 508-T-6
Computer Service Manual: Publication No. 508-T-7
Input-Output Service Manual: Publication No. 508-T-8
System Reference Schematics: Publication No. 508-T-11

The block diagram of figure 1 illustrates each system component and shows its applicable service manual. In brief, the scope of each RECOMP II service manual is as follows:

System Service Manual: This manual describes the general concept of RECOMP II maintenance and provides instructions for checkout of the computer system using the system tester in both static and dynamic test modes. The goal of these system test procedures is to isolate malfunctions to a specific operational area or system component.

Computer Service Manual: This manual describes the operational components located in the computer assembly, including memory unit, power circuits, and signal circuits. It also provides maintenance instructions and adjustments for computer components, and gives a detailed set of test procedures for computer circuit boards using the component tester.

Input-Output Service Manual: This manual describes the operational characteristics of RECOMP II input-output equipment (control console, typewriter, tape reader, and tape punch). It also provides maintenance and test instructions for these input-output devices and the associated desk assembly.

Test Equipment Service Manual: This manual describes the functional characteristics of the system tester and component tester. It also provides maintenance and test information for these two RECOMP II test equipments.

System Reference Schematics: This publication provides a set of schematics, assembly drawings, wiring charts, and signal charts for each system component; i.e., computer, desk, control console, tape reader, typewriter, and tape punch.
CONTENTS

Preface ........................................ iii

Introduction ................................... 1

Preliminary Checkout .......................... 1
Operational Checkout ........................... 1
Logic Checkout ................................. 2

Maintenance Aids ............................... 3

Control Console ............................... 4
Paper Tape Reader .............................. 11
Electric Typewriter ............................ 11
Paper Tape Punch .............................. 12
System Tester .................................. 13
System Operation and Control ................ 21

Logic Description ............................. 33

General ......................................... 33
Logic Equation Analysis ........................ 38
Computer Characteristics .................... 45
Counter Logic .................................. 55
Command Selection, $I_1$ ....................... 65
Command Interpretation and Operand Selection, $I_2$ 85
Command Execution ............................ 93
Keyboard Fill .................................. 172
Visual Readout ($D_6'D_5'D_4'D_3'D_2'D_1'$) .... 206
Display Command ($DIS, D_6'D_5'D_4'D_3'D_2'D_1'$) . 208
Setting the Code Operation Register to $D_6'D_5'D_4'D_3'D_2'D_1'$ 230

Filling the B and R Register .................. 230

Punch and Typewriter Output ................. 235
Punch Word (PNW), Type Word (TYW), and Punch and Type Word (PTW) Commands 245
<table>
<thead>
<tr>
<th>CONTENTS (Cont)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Checkout Procedures</td>
<td>261</td>
</tr>
<tr>
<td>General</td>
<td>261</td>
</tr>
<tr>
<td>Computer</td>
<td>261</td>
</tr>
<tr>
<td>Control Console</td>
<td>279</td>
</tr>
<tr>
<td>Input-Output Units</td>
<td>288</td>
</tr>
<tr>
<td>Tape Preparation Process</td>
<td>294</td>
</tr>
<tr>
<td>Test Routines</td>
<td>296</td>
</tr>
<tr>
<td>Command Test Routine</td>
<td>297</td>
</tr>
<tr>
<td>Transfer Commands Test Routine</td>
<td>304</td>
</tr>
<tr>
<td>General Machine Test I</td>
<td>308</td>
</tr>
<tr>
<td>General Machine Test II</td>
<td>312a</td>
</tr>
<tr>
<td>Type Character and Punch/Type Character Test Routine</td>
<td>316</td>
</tr>
<tr>
<td>Type Character, Type/Punch Character, and Punch</td>
<td>319</td>
</tr>
<tr>
<td>Character Consecutively Test Routine</td>
<td>319</td>
</tr>
<tr>
<td>Alpha Dump Test Routine</td>
<td>321</td>
</tr>
<tr>
<td>Command Memory Dump (Punch) Test Routine</td>
<td>323</td>
</tr>
<tr>
<td>Enter Numbers through Photoreader Test Routine</td>
<td>324</td>
</tr>
<tr>
<td>Double-Punching Test Routine</td>
<td>326</td>
</tr>
<tr>
<td>Input-Output and Memory Test Routine</td>
<td>326</td>
</tr>
<tr>
<td>Memory Test Routine</td>
<td>329</td>
</tr>
<tr>
<td>Nixie Tube Test Routine</td>
<td>330</td>
</tr>
<tr>
<td>Read Tape Test Routine</td>
<td>331</td>
</tr>
<tr>
<td>Computer-Controlled Typewriter Input Test Routine</td>
<td>332</td>
</tr>
<tr>
<td>Computer-Controlled Typewriter Alphanumeric</td>
<td></td>
</tr>
<tr>
<td>Input Test Routine</td>
<td>333</td>
</tr>
<tr>
<td>Punch Word Test Routine</td>
<td>335</td>
</tr>
<tr>
<td>Punch and Type Word Test Routine</td>
<td>336</td>
</tr>
<tr>
<td>Trapping Mode Test Routine</td>
<td>337</td>
</tr>
<tr>
<td>Loop Load Routine</td>
<td>338</td>
</tr>
<tr>
<td>Fast Memory Fill and One Channel Zero Routine</td>
<td>338</td>
</tr>
</tbody>
</table>
## CONTENTS (Cont)

<table>
<thead>
<tr>
<th>Appendix I.</th>
<th>Definition of Logic Terms</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix II.</td>
<td>Logic Equations</td>
<td>363</td>
</tr>
<tr>
<td>Appendix III.</td>
<td>Circuit Board Logic Charts</td>
<td>498</td>
</tr>
</tbody>
</table>

Computer Connector Signal Charts | 498
100 - 200 Series - Side No. 1 | 499
300 Series - Side No. 1 | 499
400 - 500 Series - Side No. 1 | 499
100 - 200 Series - Side No. 2 | 499
300 Series - Side No. 2 | 499
400 - 500 Series - Side No. 2 | 500
ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RECOMP II System Block Diagram</td>
<td>ii</td>
</tr>
<tr>
<td>2</td>
<td>Control Console (Front View)</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Visual Readout with Display of Numbers</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Control Console (Rear View)</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Tape Reader</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>Typewriter (Front View)</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>Tape Punch</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>System Tester</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>Control Unit Network Board</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>PA, Word Format for Numerical Quantities and Commands</td>
<td>47</td>
</tr>
<tr>
<td>11</td>
<td>Functional Block Diagram of RECOMP II</td>
<td>52</td>
</tr>
<tr>
<td>12</td>
<td>RECOMP II Master Sequence Control Cycle</td>
<td>53</td>
</tr>
<tr>
<td>13</td>
<td>RECOMP II Digit Counter</td>
<td>56</td>
</tr>
<tr>
<td>14</td>
<td>Sector Counter Augmentation</td>
<td>64</td>
</tr>
<tr>
<td>15</td>
<td>Example of Sector Counting</td>
<td>65</td>
</tr>
<tr>
<td>16</td>
<td>Recirculation Logic For G Counter</td>
<td>67</td>
</tr>
<tr>
<td>17</td>
<td>Sector Selection During $U_1$</td>
<td>74</td>
</tr>
<tr>
<td>18</td>
<td>Operation of Z Register</td>
<td>77</td>
</tr>
<tr>
<td>19</td>
<td>Operation of 8-Word Loop During $I_1$</td>
<td>82</td>
</tr>
<tr>
<td>20</td>
<td>Operation Code Matrix</td>
<td>88</td>
</tr>
<tr>
<td>21</td>
<td>Operation of CLA-CLS Commands</td>
<td>103</td>
</tr>
<tr>
<td>22</td>
<td>Truth Table for Binary Adder</td>
<td>107</td>
</tr>
<tr>
<td>23</td>
<td>Truth Table for Subtractor</td>
<td>108</td>
</tr>
<tr>
<td>24</td>
<td>Operation of ADD Command During $U_1$</td>
<td>111</td>
</tr>
<tr>
<td>25</td>
<td>Summary of ADD and SUB Command</td>
<td>118</td>
</tr>
<tr>
<td>26</td>
<td>Operation of ADD and SUB Commands During $U_2$</td>
<td>118</td>
</tr>
<tr>
<td>27</td>
<td>Multiplication Example</td>
<td>124</td>
</tr>
<tr>
<td>28</td>
<td>Multiplication During $U_1$</td>
<td>125</td>
</tr>
<tr>
<td>29</td>
<td>RECOMP Multiplication Process (MPY)</td>
<td>128</td>
</tr>
<tr>
<td>30</td>
<td>Multiplication During $U_1'$</td>
<td>130</td>
</tr>
<tr>
<td>31</td>
<td>Division Example</td>
<td>135</td>
</tr>
<tr>
<td>32</td>
<td>Division During $U_1$</td>
<td>137</td>
</tr>
<tr>
<td>33</td>
<td>Floating Point Add-Subtract Sequence</td>
<td>149</td>
</tr>
<tr>
<td>34</td>
<td>Floating Point Multiply-Divide Sequence</td>
<td>164</td>
</tr>
<tr>
<td>35</td>
<td>Filling the First Octal Character into A Register</td>
<td>223</td>
</tr>
<tr>
<td>36</td>
<td>Control Panel Network Board</td>
<td>267</td>
</tr>
<tr>
<td>37</td>
<td>Clock Signal Waveform</td>
<td>268</td>
</tr>
</tbody>
</table>
ILLUSTRATIONS (Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>Origin Pulse and Waveshape</td>
<td>269</td>
</tr>
<tr>
<td>39</td>
<td>Strobe Waveshape for Normal Pulse</td>
<td>273</td>
</tr>
<tr>
<td>40</td>
<td>Location of Computer Connectors Used During Signal Checks</td>
<td>276</td>
</tr>
<tr>
<td>41</td>
<td>Computer Connector Signal Chart</td>
<td>501</td>
</tr>
<tr>
<td>42</td>
<td>Circuit Board Connector Signal Chart No. 1</td>
<td>502</td>
</tr>
<tr>
<td>43</td>
<td>Circuit Board Connector Signal Chart No. 2</td>
<td>503</td>
</tr>
<tr>
<td>44</td>
<td>Circuit Board Connector Signal Chart No. 3</td>
<td>504</td>
</tr>
<tr>
<td>45</td>
<td>Circuit Board Connector Signal Chart No. 4</td>
<td>505</td>
</tr>
<tr>
<td>46</td>
<td>Circuit Board Connector Signal Chart No. 5</td>
<td>506</td>
</tr>
<tr>
<td>47</td>
<td>Circuit Board Connector Signal Chart No. 6</td>
<td>507</td>
</tr>
</tbody>
</table>
### TABLES

<table>
<thead>
<tr>
<th></th>
<th>Adjustments Used to Establish Marginal Test Conditions</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Marginal Test Conditions</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Designations of Potentiometers Used for Establishing Marginal Test Condition</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>Truth Table for 3-stage Pulse Counter</td>
<td>43</td>
</tr>
<tr>
<td>5</td>
<td>RECOMP II Commands</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>Synchronization Process</td>
<td>60</td>
</tr>
<tr>
<td>7</td>
<td>Sector Counting in 8-Word Loops</td>
<td>81</td>
</tr>
<tr>
<td>8</td>
<td>Counting Sequence</td>
<td>97</td>
</tr>
<tr>
<td>9</td>
<td>Normal Voltage Values</td>
<td>263</td>
</tr>
</tbody>
</table>
This page intentionally left blank
INTRODUCTION

Computing system maintenance comprises two general types: system maintenance and component maintenance. System maintenance is primarily concerned with checking out the operational status of each functional system component in terms of system operation. Component maintenance principally consists of preventive maintenance, and replacement of malfunctioning electronic and electromechanical parts of the system. The procedures of system maintenance are thus primarily trouble-isolation procedures. That is, whenever faulty operation occurs, the trouble is traced to a particular system component by using the system checkout procedures. If, for example, the trouble should be in the main computer circuitry, the trouble is traced to a particular functional circuit or group of circuits within the computer. Once the trouble has been identified with a particular functional area of the computer system, the technician refers to the applicable component service manual for (1) more specific maintenance procedures to isolate trouble to particular electrical or mechanical components and (2) instructions to repair or replace the defective components. In this system service manual, system maintenance is divided into three phases: Preliminary Checkout, Operational Checkout, and Logic Checkout.

PRELIMINARY CHECKOUT

Preliminary checkout procedures provide instructions for checking electrical and operational conditions that must be satisfied before the more complex operational steps can be performed. For example, all d-c power supply voltage levels and load characteristics are checked in preliminary checkout. The ability of the control console to enter information and to read out information from the computer is also checked. All preliminary checkout procedures are limited to those that can be performed relatively quickly and without referring to logic interpretation.

OPERATIONAL CHECKOUT

Operational checkout procedures provide the means for checking out the ability of the computer to perform various commands and functions. Short routines are entered from the control console (or the tape reader or typewriter) and the computer then is set to perform these
routines. Results are monitored on the visual readout of the control console and/or an oscilloscope. If the routine fails, the computer has incorrectly executed a command or malfunctioned in another manner. The maintenance engineer can then, through use of the logic descriptions, determine the circuits involved in the malfunction.

**LOGIC CHECKOUT**

Logic checkout involves determination of the characteristics of each function in terms of the logic equations which represent the detailed operational design of the computer. From this, the maintenance engineer can then readily determine the circuits and components involved. In most cases, logic checkout procedures isolate trouble to particular circuit boards, which can then be checked out in detail on the component tester and replaced.
MAINTENANCE AIDS

The RECOMP II system (see figure 1) has numerous facilities which aid in checking out the system, troubleshooting, and performing preventive maintenance. Most important of these is the System Tester, a separate self-contained unit which provides facilities for testing and monitoring the computing system in both dynamic and static modes of operation. This unit is especially useful in conducting preventive maintenance tests under marginal conditions. Other system checkout aids—all integral parts of the system—are the control console, tape reader, tape punch, and electric typewriter. All of these units are used both with and without the system tester, as checkout and troubleshooting may require.

The control console contains virtually all of the operating switches and indicators used in directing the activities of the computing system. Consequently, it is employed extensively in almost all checkout and troubleshooting procedures. The control console keyboard is used to set up memory locations and to enter commands and data into the computer. Readout switches and the visual readout panel are utilized in checking the contents of registers and memory locations. Error indicators, memory location indicators, and several operating switches aid in determining the exact location of a malfunction. The control console also contains several other checkout and troubleshooting facilities. These items, located on the rear of the console, consist of (1) power supply voltmeter and range switch which facilitates checking the secondary d-c voltages and (2) memory and register test jacks which permit direct connection of an oscilloscope to check memory and register contents. Procedures in which the control console is used in checkout, troubleshooting, and preventive maintenance of the RECOMP II system are presented later. All persons associated with maintenance of the RECOMP II system should be thoroughly familiar with the functions of all console controls and indicators before performing any work on the system. A description of the controls and indicators and their functions is given in the RECOMP II Operating Manual.

Principal use of the tape reader, tape punch, and typewriter in checkout, troubleshooting, and preventive maintenance is in the insertion and recording of data. Use of these units in this work is presented later. Operating procedures for these units and their controls are given in the Operating Manual.
Uses of the system tester are given later in this section. System tester controls are described in the Test Equipment Service Manual. Persons utilizing the system tester should completely understand its various modes of operation before using it.

CONTROL CONSOLE

Most of the controls for directing computer activities during checkout, trouble-shooting, and preventive maintenance are located on the control console (see figure 2).

Control of Internal Operations

Principal use of the control console is in the control of computer internal operations. In addition to the controls for starting, stopping, and applying power to the computer, the console contains the switches and indicators used in directing the system in the various modes of operation. The Start switches can be used in several different ways during checkout and maintenance work. The regular Start switch is utilized (1) to start the computer to execute instructions, as when running a test routine, and (2) in conjunction with the single command operation switch to cause the computer to execute one instruction at a time and stop automatically. The single-command operating mode is especially helpful when isolating a malfunction. If the computer is not executing a test program correctly, the instruction causing the difficulty can be determined by displaying the contents of registers and memory locations on the visual readout following execution of each command.

Start 1, Start 2, and Start 3 switches start the computer on execution of a sequence of instructions beginning at memory location 0001, 0002, and 0003, respectively. They are used mainly with some test routines having the initial command in one of these memory locations. All Start switches initiate the Compute state which is indicated by the Compute neon indicator in the center of the console.

Four possible methods of stopping the computer provide additional flexibility in controlling internal operations during checkout and maintenance. These are the regular Stop button, the Transfer Stop switch, the Pre-set Stop switch, and the Error Reset button.

The Transfer Stop and Pre-set Stop switches are of particular value in locating malfunctions. When placed in the Up position, the Transfer Stop switch halts computer operation after interpretation of any transfer instruction even though the condition necessary for transfer of
Figure 2. Control Console Front View
control was not met. The location of the command containing the transfer instruction is placed in the right half of the X-register from where it can be displayed on the console visual readout. This enables checking if the location of a transfer instruction in a test program is incorrect because of improper sequence or a memory read-write malfunction. The Pre-Set Stop switch, when placed in the 1st or 2nd position, stops the computer after execution of the command in the left or right half-word position, respectively, of the full-word location as set on the Channel and Sector Selectors. This permits running a test routine until it reaches the location in which the malfunction is known or believed to exist. Contents of the memory location of the last instruction executed may then be displayed on the visual readout. The location counter will display the location of the next instruction to be executed.

Other switches and indicators used in controlling internal operations include the Sense switches, Error indicators and Error Reset button, and the Fill Source button. The three Sense switches operate in conjunction with the Conditional Transfer instructions TSB, TSC, and TSD to enable manual direction of the computer along various routes of a stored program.

Error Reset permits resumption of operation following a stop caused by an output, overflow, or verify error. However, the Error indicators, are of greater aid than the Error Reset button in checkout and maintenance. The Verify Error neon indicates a discrepancy between the contents of a tape and the information originally stored in memory from that tape. Usually a Verify error indicates faulty recording of the information in memory. It may also, however, serve as a means of checking the accuracy of the tape reading process if known tape content is read in and then displayed from memory on the console visual readout. The Verify mode of operation (see Operating manual) is helpful in locating a memory write malfunction because the location counter will indicate the next memory address that is to be verified; i.e., the malfunction is at the address which is one sector less than shown by the location counter. An output error is indicated when a difference occurs between the code sent from the computer and that typed. By comparing the information at the memory location indicated by the location counter with that shown on the output register on the punch, the character being typed correctly can be determined. The Overflow Error indicator lights whenever a number exceeds the capacity of the A-register (see Operating manual). An overflow condition usually indicates an error in the program but may also signify improper command execution. All three error lights are extinguished by depressing the Error Reset button.
Environmental conditions affecting proper computer operation are indicated by three warning neon indicator lights. These conditions are open circuit panels, indicated by the Interlock neon; excessively high internal temperatures, indicated by the High Temperature neon; and a line voltage drop or increase, indicated by the Line Transient neon. The computer may be operated with one or both hinged door circuit panels open; however, internal operating temperatures may become sufficiently high to cause the computer to halt. A high temperature indication will precede a halt. Excessively high temperature inside the computer may turn off the computer almost immediately or after a long delay, depending on the rate of temperature rise. The abnormally high temperature may have originated in the power supply or memory. A line voltage drop below 108 volts or above 130 volts will cause the Line Transient neon to light. Usually a line voltage change sufficient to light the indicator will be of concern only to the operator. This is because and excessive voltage change during computing may alter information in the registers or rapid access loops. Information in main memory is unaffected by line voltage variations other than high transients.

Main use of the Fill Source button in checkout and maintenance is to set up the computer to receive instructions and data from the console keyboard. The Fill Source indicator lights when the Fill Source button is depressed. It turns off automatically when the Fill switch on the tape reader or typewriter is depressed, when an input command is executed, or when the computer enters the Compute state.

**Isolation of Malfunctions**

The location counter indicator in the lower left corner of the console panel is used extensively in maintenance work. The location counter indicator consists of 13 neon indicators and displays: (1) the address of the location in which the data is being stored or selected for comparison in the fill or verify mode (2) the address of the location of the next instruction to be executed when operating in the compute state. Instances in which the location counter indicator can aid in isolating a malfunction include: (1) location of the instruction on which a command execution malfunction occurred, (2) location of the instruction to which a program will transfer, (3) visual verification of the location in which the information is being placed or verified, and (4) visual observation of memory access and program execution sequences during single-step operation.
Input of Commands and Data

The control panel keyboard is the principal medium for entering commands and data during checkout and preventive maintenance. It is also used for setting the location counter. All types of information except alphanumeric can be entered from the console keyboard. Maintenance work in which the keyboard is utilized include: (1) modification of test routines to alter the course of operations, (2) entry of test data into memory, (3) insertion of commands to check correct execution, (4) entry of data into memory to ascertain proper functioning of output equipment, (5) entry of data in memory to verify accuracy of write and read processes in the channel in which a malfunction occurred, and (6) entry of information into channels when making memory and register gain and other adjustments. The switches identified by L, C, and N—and their indicators—are considered part of the keyboard. Their main use is to set the computer to receive numerical entries in location, command, or number format, respectively. Procedures for entering information in these formats are given later in this section.

Readout of Memory and Register Contents

The visual readout (see figure 3) is of considerable assistance in maintenance by enabling: (1) visual inspection of the information in main memory channels, rapid access loops, and registers, (2) verification of data before entry, (3) isolation of a malfunction in the write, read, and command execution processes, (4) display of results of a command execution or other operation, (5) a check of information entered or transmitted from the computer against that in memory, (6) a check on accuracy of the computations performed, and (7) determination of proper execution of transfer commands. Frequently, the visual readout is more helpful in isolating malfunctions when it is used with the single command operating mode. Commands and data entered from the keyboard are displayed on the visual readout automatically upon depression of a button. Display of information in the computer is achieved by execution of the Display command or manually. The latter is readily performed with a row of rotary memory and register location switches and three readout buttons. Contents of all main memory and rapid access loop locations and all registers except the B-register can be displayed in binary coded decimal, octal, and command formats. Procedures for obtaining readout manually are presented on pages 24 through 28 under Computer Control in the Operating manual, Pub. 512-E-3.
Figure 3. Visual Readout with Display of Numbers
Monitoring of Internal Operations

Facilities for monitoring internal computer operations on the rear of the console (see figure 4) consist of test receptacles and a power supply voltmeter range switch and percentage indicator. The test receptacles enable examination of the contents of registers, loops, and main memory locations with the use of an oscilloscope. This aids in determining if the contents of the memory or registers change, if the writing and reading processes are functioning properly, and if the signals are of proper amplitude. Contents of registers and loops can be displayed in continuous shape; contents of main memory locations can either be viewed as the contents pass through the circuitry or in continuous shape by placing the contents of a main memory location into a register or loop. A synchronization receptacle provides a synchronization signal for the oscilloscope.

All secondary d-c voltage levels of the computer and control console can be conveniently checked with the power supply voltmeter range switch and percentage indicator. The levels are tested by turning the range switch to each position and observing percentage reading on the indicator.

Figure 4. Rear of Console
PAPER TAPE READER

Use of the tape reader (see figure 5) in checkout and maintenance consists of entering test routines into the computer and verifying the recording of information into memory. The latter function is performed by comparing the contents of memory against that on the tape with which it had been entered. Verification thus serves as a rapid check of correct recording in memory. Operation of the tape reader in both the Fill and Verify modes is described later.

ELECTRIC TYPEWRITER

The electric typewriter (see figure 6) is especially useful in obtaining a printout of the contents of memory for comparison with information entered or for examination to detect program errors and machine malfunctions. The typewriter is also used frequently during checkout and trouble-shooting to enter information into the computer and, in conjunction with the paper tape punch, to prepare test tapes. Because information can be entered into the computer in the same format as from the tape reader and console keyboard, the typewriter can be utilized to enter commands and data when either of the other units is inoperative. When the tape reader is inoperative, the typewriter also is the only
means of entering or receiving alphanumeric information to or from the computer. Several typewriter control codes can also be used to direct computer operations. These control codes, procedures for entering information into the computer with the typewriter, and use of the typewriter in preparation of test tapes are presented later in this section. Typewriter output operation is controlled by program instructions TYC, TYW, PTC, and PTW. Use of these instructions is outlined in the Operating manual.

PAPER TAPE PUNCH

Main uses of the paper tape punch (see figure 7) in checkout and maintenance are the preparation of test tapes and the output of information when the typewriter and visual readout are malfunctioning. The punch is also used occasionally with the tape reader in test routines to check correct operation of the tape reader, tape punch, and typewriter. The output register, located on the punch control panel, aids in determining if the typewriter is recording correctly. If the Output Error indicator on the console is on, the output register will indicate the character mistyped. Comparison between the character displayed on the output register and that typed will help to isolate the malfunction. A check to ascertain if the register displays correctly can be made by comparing the display on the output register with the contents of the memory location being transferred to the punch. Loading of paper tape into the punch is described on the inside of the cover of the unit; use of the punch controls and punch instructions is explained later in this section. Computer controlled punch operation is governed by program instructions PNC, PNW, PTC, and PTW. Their use is described in the Operating manual.
Figure 7. Tape Punch

SYSTEM TESTER

Uses of System Tester

Main purpose of the System Tester (see figure 8) is to simplify trouble-shooting. Specific uses include (1) monitoring the computer flip-flops, including the read and write amplifier flip-flops; (2) controlling the computer flip-flops to determine if each is functioning properly and to static-check the logic; (3) checking operation of gates to determine if operating properly marginally, or if inoperative; (4) controlling a number of flip-flops under dynamic conditions to aid in isolating a malfunction; and (5) applying marginal conditions to the computer to locate incipient malfunctions. Procedures for operating the system tester in these uses are given in the System Checkout Procedures section of this manual.

Uses of System Tester Control

System tester controls consist of two power switches, the flip-flop set-reset switches and indicators, a static-dynamic mode switch, a marginal check switch, and a one-shot clock switch. Except for the power switches, all are used to establish various conditions within the
Figure 8. System Tester
tester that will cause the computer to react to desired external control rather than normal internal operation. Connecting the system tester does not, however, affect the normal dynamic characteristics of the computer nor alter the normal sequence of operations except when a change is initiated externally through the system tester buttons. Functions of these controls and the computer servicing operations in which they are used are given in the following paragraphs.

Application of Power

The power controls apply power to the tester. The Power switch applies 60-cycle power to the power supply and internal circuits of the tester and the Neon Power switch applies d-c power to the tester panel neons. A neon indicator above each of these switches indicates proper application of power within the tester.

Setting and Resetting of Flip-Flops

Set-Reset controls consist of a 1-Set All—0-Set All switch, a Preset 1-Preset 0 switch, and individual microswitches for each flip-flop. These controls permit resetting the flip-flops (terms), including those in the read and write amplifiers, to 0 or 1 states. The 1-Set All—0-Set All switch enables setting all flip-flops simultaneously to one state or the other; the Preset 1-Preset 0 in conjunction with the individual microswitches enables setting the individual flip-flops to one state or the other. The Set-Reset controls are used in determining proper switching of the flip-flops and in locating malfunctioning flip-flops. Instances in which these controls are used for isolation of malfunction are (1) improper execution of commands, (2) weak read and write signals, (3) incorrect writing on and reading from memory, (4) incorrect memory location selection, and (5) broken wiring or wiring errors.

Checkout in Static or Dynamic Modes

The Static-Dynamic mode switch enables checking the logic through testing gates in the static mode and through observing indications of tester lights in the dynamic mode. In some procedures the Static-Dynamic switch is used in conjunction with the One-Shot Clock switch, in others with the Marginal Check switch.

With the Static-Dynamic switch in the Static position, any logic gate or gates set up can be triggered with the One-Shot Clock switch. The Static position turns off the computer clock and leaves it in the true state. When the One-Shot Clock switch is depressed, the tester forces the clock line to a false level by grounding it. This initiates a positive
step function suitable for triggering the flip-flops to statically check the logic gates one at a time. It is helpful in isolating almost every malfunction of an electronic nature because gates are used in all logic and many other operations, such as console keyboard entry and visual readout.

During the dynamic testing mode, normal computer operation is monitored by the system tester. When computation ceases, the states of all flip-flops are displayed. The malfunctions can then be isolated by the utilization of the computer logic with the necessary understanding of computer operation. The system tester does not effect the operation of the computer during normal operation.

**Operation Under Marginal Conditions**

Use of the Static-Dynamic switch in conjunction with the Marginal Check switch is of greater assistance in checkout and preventive maintenance than trouble-shooting. The Marginal Check switch can be set to change various voltage levels within the computer. These changed levels can be applied in either the static or dynamic mode according to the setting of the Static-Dynamic switch. The -6 and -18 voltages are lowered in either the static or dynamic mode; the +6 and -12 voltages are raised in the static mode and raised and lowered in the dynamic mode. When the marginal condition is imposed in the dynamic mode, the clock is also advanced and retarded on successive clock pulses. Two jitter flip-flops generate a square wave of one-fourth the clock rate for modulating the +6 and -12 voltages. The frequency and the rise and fall times of these waves do not cause transient effects within the computer.

Imposing marginal conditions is especially useful in locating incipient malfunctions. Application of a marginal condition in the static mode permits testing of a specific component in the dynamic mode to continuous operation of the computer on a given program. Components and circuits which function satisfactorily under normal operating conditions will fail when marginal conditions are imposed; thus the static mode enables locating and replacing failing components before a malfunction occurs. Under marginal conditions in the dynamic mode, if the computer handles a program or routine correctly, it usually is operating satisfactorily. The marginal check is helpful in trouble-shooting through aiding in isolating a malfunction that is occurring intermittently.
Flip-flop indicators on the System Tester enable identification of a malfunctioning flip-flop. When a computer flip-flop is in the true state the appropriate neon is illuminated; when in the false state it is extinguished. There is one neon for each flip-flop, including each read and write amplifier flip-flop. Proper operation of the neons themselves can be easily confirmed by placing the 1-Set All-0-Set All switch in the 1-Set All position and noting if all neons are lighted.

The control unit network board mounts the marginal test condition potentiometers. This board is shown in Fig 9.

The adjustments available to the computer operator for establishing marginal test conditions are listed in Table 1. The marginal test conditions available in a computer are shown in Table 2. The voltage levels for marginal test conditions may be adjusted by the use of potentiometers located on the control unit network board. The designations of marginal tests voltage conditions and adjustment potentiometers are listed in Table 3.
Table 1. Adjustments Used To Establish Marginal Test Conditions

<table>
<thead>
<tr>
<th>Adjustment</th>
<th>Static margin</th>
<th>Dynamic margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>-18 v</td>
<td>Static margin = R12</td>
<td>Dynamic margin = R10</td>
</tr>
<tr>
<td>-6 v</td>
<td>Static margin = R14</td>
<td></td>
</tr>
<tr>
<td>-12 v</td>
<td>Static margin = R5</td>
<td></td>
</tr>
<tr>
<td>+6 v</td>
<td>Static margin = R13</td>
<td></td>
</tr>
<tr>
<td>-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R12, R14, R5, R3, R10, R40 located on marginal test board of System Tester.

Clock Jitter width = R4 on clock board No. 1 in computer.

Table 2. Marginal Test Conditions

<table>
<thead>
<tr>
<th>Switch Setting</th>
<th>Action</th>
<th>Clock Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mode</td>
<td>Voltage Change</td>
</tr>
<tr>
<td></td>
<td>Marginal Test</td>
<td>-6</td>
</tr>
<tr>
<td>s1 s2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static OFF</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Static No. 1</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Static No. 2</td>
<td>Lowered by 10%</td>
<td>None</td>
</tr>
<tr>
<td>Static No. 3</td>
<td>Lowered by 10%</td>
<td>None</td>
</tr>
<tr>
<td>Static No. 4</td>
<td>Lowered by 10%</td>
<td>Raised by 10%</td>
</tr>
<tr>
<td>Static No. 5</td>
<td>Lowered by 10%</td>
<td>Raised by 10%</td>
</tr>
<tr>
<td>Dynamic OFF</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Dynamic No. 1</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

18
Table 2. (Continued)

<table>
<thead>
<tr>
<th>Switch Setting</th>
<th>Action</th>
<th>Voltage Change</th>
<th>Clock Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>S1</td>
<td></td>
</tr>
<tr>
<td>Dynamic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. 2</td>
<td>Lowered</td>
<td>-6</td>
<td>Lowered</td>
</tr>
<tr>
<td></td>
<td>by 10%</td>
<td>+6</td>
<td>by 10%</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>-12</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>-18</td>
<td>None</td>
</tr>
<tr>
<td>Dynamic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. 3</td>
<td>Lowered</td>
<td>-6</td>
<td>Lowered</td>
</tr>
<tr>
<td></td>
<td>by 10%</td>
<td>+6</td>
<td>by 10%</td>
</tr>
<tr>
<td></td>
<td>Jitter</td>
<td>-12</td>
<td>±10%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-18</td>
<td>±10%</td>
</tr>
<tr>
<td>Dynamic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. 4</td>
<td>Lowered</td>
<td>-6</td>
<td>Lowered</td>
</tr>
<tr>
<td></td>
<td>by 10%</td>
<td>+6</td>
<td>by 10%</td>
</tr>
<tr>
<td></td>
<td>Jitter</td>
<td>-12</td>
<td>±10%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-18</td>
<td>±10%</td>
</tr>
<tr>
<td>Dynamic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. 5</td>
<td>Lowered</td>
<td>-6</td>
<td>Lowered</td>
</tr>
<tr>
<td></td>
<td>by 10%</td>
<td>+6</td>
<td>by 10%</td>
</tr>
<tr>
<td></td>
<td>Jitter</td>
<td>-12</td>
<td>±10%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-18</td>
<td>±10%</td>
</tr>
</tbody>
</table>

Table 3. Designations of Potentiometers Used for Establishing Marginal Test Conditions

<table>
<thead>
<tr>
<th>Adjustment</th>
<th>Voltage</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>R33</td>
<td>Line Trans</td>
<td>High Limit</td>
</tr>
<tr>
<td>R34</td>
<td>+50</td>
<td>Adjusted before adjustment of R47</td>
</tr>
<tr>
<td>R35</td>
<td>-12</td>
<td></td>
</tr>
<tr>
<td>R36</td>
<td>±6</td>
<td>Adjust for mean value</td>
</tr>
<tr>
<td>R37</td>
<td>+0.75</td>
<td></td>
</tr>
<tr>
<td>R38</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>R39</td>
<td>-18</td>
<td></td>
</tr>
<tr>
<td>R40</td>
<td>-100</td>
<td>Adjustment dependent on voltage level</td>
</tr>
<tr>
<td>R42</td>
<td>Line Trans</td>
<td>Low Limit</td>
</tr>
<tr>
<td>R47</td>
<td>+75</td>
<td>Adjusted after adjustment of R34</td>
</tr>
</tbody>
</table>
Figure 9. Control Unit Network Board
SYSTEM OPERATION AND CONTROL

Checkout, trouble-shooting, and preventive maintenance is much more easily and quickly accomplished when operation and control of the system is thoroughly understood. Procedures for operating the computer and its associated input-output units are given on the following pages, but principally as a brief, convenient reference. Detailed operating procedures are presented in the Operating manual. Procedures given here are presented in the order in which they will usually be used; exceptions are covered in the tests in which the variation in operating procedure is required.

Computer Operating Procedure

To apply power to computer:

Depress Power On button on control console (Power On indicator should light immediately; Ready indicator should light approximately 45 seconds after power turn-on).

To set Location counter from console keyboard:

1. If Fill Source indicator is extinguished, depress Fill Source button.
2. If Compute, Output Error, Overflow Error, and Verify Error indicators are lit, depress Error Reset button.
3. Depress Location button (Location indicator should light).
4. Depress in proper sequence the keyboard buttons corresponding to the location in which it is desired to enter information, consisting of four octal digits for the channels and sectors desired and one binary digit to indicate the first or second half of the sector (a zero for first and a one for second half, usually a zero). (Corresponding numerals should be displayed on the visual readout; if a wrong character has been keyed depress Clear button and repeat this step from its beginning.)
5. Depress Enter button (location will be entered into computer, visual readout will be cleared, and location counter indicator should display location set up -- location counter indicator is read by adding numbers beside lighted neons).

To enter commands from console keyboard:

1. Set location counter.
2. Depress Command button (Command indicator will light).
3. Depress in proper sequence the keyboard buttons for the pair of commands desired, consisting for each command of a sign, two octal digits for the operation code, four octal digits representing the address or other information, one binary digit (zero or one) for half-word sector indication or other information (the various meanings of sign, address, and half-word indicators are explained in the Operating manual); corresponding characters are displayed on the visual readout as the buttons for the command are depressed, if a wrong character has been keyed depress Clear button and repeat this step from its beginning.
4. Depress Enter button (pair of commands will be entered into location desired, visual readout will be cleared, and location counter will be advanced to next location).
5. To enter additional commands in consecutive locations repeat steps 3 and 4, in non-consecutive locations steps 1 through 4.

To enter mixed numbers from console keyboard.

1. Set location counter.
2. Depress Number button (Number indicator will light).
3. To enter whole-number portion, depress plus or minus sign followed by up to 11 (numeral) keys representing decimal digits (corresponding numerals are displayed on visual readout as numeral key is depressed, if wrong character has been keyed depress Clear button and repeat this step from its beginning).
4. Depress decimal point key (whole number should be entered into location desired and location counter should be advanced to next location; if it is desired to clear visual readout before entering fractional portion depress Clear key after decimal point key).
5. To enter fractional portion, depress up to 11 numeral keys representing decimal digits--if Clear or Enter key had been depressed following entry of whole number, also depress sign and decimal point keys before numeral keys (corresponding character are displayed on visual readout as keys are depressed, if wrong character has been keyed, depress Clear button and repeat this step from its beginning.
6. Depress Enter key (fractional number will be entered into location following whole number, visual readout will be cleared, and location counter will be advanced to next location).

7. To enter additional mixed numbers in consecutive locations repeat steps 3 through 6; in nonconsecutive locations, steps 1 through 6.

To enter whole numbers only:

1. Set location counter.
2. Depress Number button (Number indicator will light).
3. Depress plus or minus sign followed by up to 11 numeral keys representing decimal digits (corresponding numerals are displayed on visual readout as numeral key is depressed, if wrong character has been keyed depress Clear button and repeat this step from its beginning).
4. Depress decimal point key (whole number will be entered into location desired and location counter should be advanced to next location).
5. Depress Enter key (visual readout will be cleared).
6. To enter additional whole numbers only in consecutive locations repeat steps 3 through 6; in non-consecutive locations, steps 1 through 6.

To enter fractional numbers only:

1. Set location counter as described previously.
2. Depress Number button (Number indicator will light).
3. Depress plus or minus sign key, followed by decimal point key and up to 11 numeral keys representing decimal digits (corresponding numerals are displayed on visual readout as numeral key is depressed, if wrong character has been keyed depress Clear button and repeat this step from its beginning).
4. Depress Enter key (fractional number will be entered into location desired, visual readout will be cleared, and location counter will be advanced to next location).
5. To enter additional fractional numbers in consecutive locations repeat steps 3 and 4; in non-consecutive locations, steps 1 through 4.

To start computer (unless specified otherwise in test):

1. Check Ready indicator (should be lighted; if extinguished, turn on power).
2. Check Compute, Output Error, Overflow Error, and Verify Error indicators (should be extinguished; if lighted depress Error Reset button).
3. Check Fill Source indicator (should be lighted, if extinguished, depress Fill Source button).

4. Place Operation switch in Continuous position.

5. Place Preset Stop switch in Off position.


7. Place Transfer Stop switch down.

8. Set Location Counter (using steps 3 through 5 of procedure) to location at which program or routine starts.

9. Depress Start button (Compute indicator should light).

To stop computer, unless halted by error or programed stop, depress Stop button (Compute indicator should extinguish).

To display contents of registers and rapid access loops on visual readout:

1. Check Ready indicator (should be lighted, if extinguished, turn on power).

2. Check Compute indicator (should be extinguished; if lighted, depress Stop button).

3. Check Error (Output, Overflow, and Verify) indicators (should be extinguished; if lighted, depress Error Reset button).

4. To read out registers, rotate location selector knob until desired register is indicated by arrow below knob and depress readout button corresponding to display format desired; to read out rapid access loops, rotate location selector knob until L or V loop is indicated, set right Sector selector knob to any number 0 through 7 according to last octal digit of loop address to be displayed, depress readout button corresponding to display format desired. (For characteristics of display formats, see Operating manual).

5. To display additional registers or loop locations repeat step 4.

To display contents of main memory locations on visual readout:

1. Repeat steps 1 through 3 of register and loop display procedures.

2. Rotate location selector knob until M is indicated by arrow below knob, set both Channel and Sector selector knobs to channel and sector locations, respectively, to be displayed, then depress readout button corresponding to display format desired. (For characteristics of display formats, see Operating manual.)

3. To display additional main memory locations, set Channel and Sector selector knobs to location to be displayed and depress readout button of display format desired.
Typewriter Operating Procedure

Typewriter operating procedures are the same for preparation of paper tape with control codes as for direct input and computer control except as indicated.

To apply power to typewriter (computer power need be turned on only if typewriter is to be used for input and output), turn switch on circuit breaker at front of desk and switch typewriter on (if typewriter is to be used for tape preparation, power switch on paper tape punch must also be turned on).

To prepare typewriter for operation:

1. Check for proper setting of tab stops, line space lever, paper guides, margin stops, paper release lever, and tab override switch (see Input-Output Service Manual or User's Service Guide for identification and description of controls). Carriage setting limits are scale 5 on left and scale 95 on right.

To set location counter from typewriter:

1. If Compute indicator is lighted, depress Stop Button on control console.
2. If Output Error, Overflow Error, and Verify Error indicators are lighted, depress Error Reset button on control console.
3. Depress Fill key on typewriter.
4. Depress Letters Shift key.
5. Depress L key (location indicator on control console should light).
6. Depress Figures Shift key.
7. Depress in proper sequence the numeral keys corresponding to the location in which it is desired to enter information, consisting of four octal digits for the channels and sectors desired and one binary digit to indicate the first or second half of the sector (a zero for first and a one for second half, usually a zero). (Numerals are not displayed on the control console visual readout but can be checked on typewritten copy before entry; if a wrong character has been typed, depress X key on typewriter and repeat this step from its beginning).
8. Depress Carriage Return (location will be entered into computer and location counter will indicate location set up -- location counter indicator is read by adding numbers beside lighted neons).

To enter commands from typewriter:

1. Prepare typewriter for operation.
2. Set location counter.
3. Make certain Fill key on typewriter is depressed.
4. Depress Letters Shift key.
5. Depress C key (Command indicator on control console should light).
6. Depress Figures Shift key.
7. Depress in proper sequence the typewriter keys necessary for the pair of commands desired, consisting for each command of a sign, two octal digits for the operation code, four octal digits representing the address or other information, one binary digit (zero or one) for half-word sector indication or other information (the various meanings of sign, address, and half-word indicators are explained in the Operating manual); characters are not displayed on the console visual readout but can be checked on the typewritten copy before entry, if a wrong character has been typed depress X key on typewriter and repeat this step from its beginning.

8. Depress Carriage Return (pair of commands will be entered into location desired and location counter will be advanced to next location).

9. To enter Additional commands in consecutive locations repeat steps 7 and 8; in non-consecutive locations, steps 2 through 8.

To enter mixed numbers from typewriter:

1. Prepare typewriter for operation.
2. Set location counter.
3. Make certain Fill key on typewriter is depressed.

4. Depress Letters Shift key.
5. Depress N key (Number indicator on control console will light).
6. Depress Figures Shift key.
7. To enter whole-number portion, depress plus or minus sign followed by up to 11 numeral keys representing decimal digits (numerals are not displayed on control console visual readout but may be checked on typewritten copy before entry, if a wrong character has been typed depress X key on typewriter and repeat this step from its beginning).
8. Depress decimal point key on typewriter (whole number should be entered into location desired and location counter should be advanced to next location).

9. To enter fractional-number portion, depress up to 11 numeral keys representing decimal digits (numerals are not displayed on control console visual readout but can be checked on typewritten copy before entry, if a wrong character has been typed depress X key on typewriter and repeat this step from its beginning).

10. Depress Carriage Return (fractional number should be entered into location following whole number and location counter should be advanced to next location).

11. To enter additional mixed numbers in consecutive locations repeat steps 7 through 10, in non-consecutive locations steps 2 through 10.

To enter whole numbers only:

1. Prepare typewriter for operation.
2. Set location counter.
3. Make certain Fill key on typewriter is depressed.
4. Depress Letters Shift key.
5. Depress N key (number indicator on control console should light).
6. Depress Figures Shift key.
7. Depress plus or minus sign followed by up to 11 numeral keys representing decimal digits (numerals are not displayed on control console visual readout but may be checked on typewritten copy before entry; if a wrong character has been typed depress X key and repeat this step from its beginning).
8. Depress decimal point key on typewriter (whole number should be entered into location desired and location counter advanced to next location).
10. To enter additional whole numbers only in consecutive locations repeat steps 7 through 9; in non-consecutive locations steps 2 through 9.

To enter fractional numbers only:

1. Prepare typewriter for operation.
2. Set location counter.
3. Make certain Fill key on typewriter is depressed.
4. Depress Letter Shift key.
5. Depress N key (number indicator on control console should light).
6. Depress Figure Shift key.

7. Depress plus or minus sign key followed by decimal point key and up to 11 numeral keys representing decimal digits (numerals are not displayed on control console visual readout but may be checked on typewritten copy before entry; if a wrong character has been typed, depress X key and repeat this step from its beginning).

8. Depress Carriage Return (fractional number should be entered into location desired and location counter advanced to next location).

9. To enter additional fractional numbers only in consecutive locations, repeat steps 7 and 8; in non-consecutive locations, steps 2 through 8.

To enter alphanumeric information:

1. Prepare typewriter for operation.
2. Set location counter.
3. Make certain Fill key on typewriter is depressed.
4. Depress F key.
5. Depress eight alphanumeric character keys (characters are not displayed on control console visual readout but may be checked on typewritten copy before entry.
6. Depress Carriage Return (unless format change is desired in which case depress C as the ninth character, then Carriage Return).
7. To enter additional alphanumeric words in consecutive locations repeat steps 5 and 6; in non-consecutive locations steps 2 through 6.

To start Compute mode from typewriter:

1. Check Ready indicator on control console (should be lighted; if extinguished, turn on power).
2. Check Compute indicator (should be extinguished; if lighted, depress Stop button on control console).
3. Check Error (Output, Overflow, and Verify) indicators (should be extinguished; if lighted, depress Error Reset button on control console).
4. Place Operation, Preset Stop, Transfer Stop switches, and Sense Switches B, C, and D in desired positions.
5. Set location counter.
6. Depress S key on typewriter (make certain Fill key on typewriter is depressed before depressing S key; Compute indicator on control console should light).
To halt compute mode from typewriter (unless halted by error condition or programed stop) can not be done without causing an output error.

NOTE

S code alone at end of information on tape will initiate computer operation; from location set in location counter H code alone following information on tape halts tape reader. In all modes of tape preparation, information on the tape must consist of a minimum of seven characters between Carriage Return and/or Decimal Point codes which enter the information in memory, and between Carriage Return and/or Decimal Point codes and any of the control codes, such as Halt or Start. This procedure ensures proper recording of the information in memory. A blank may be substituted for any of the seven characters.

When entering a mixed number or a number consisting of only a fraction, a minimum of five tape sprocket holes must be punched between the Carriage Return code and the next number. The additional sprocket holes ensure sufficient time to convert the fractions to their internal representation. A special key on the typewriter punches these sprocket holes, one sprocket hole is punched each time the blank key is depressed.

A leader at least two folds in length should be prepared at the beginning of a tape by depressing the Tape Advance button on the punch control panel.
**Paper Tape Reader Operating Procedure**

To operate in Fill mode:

1. Turn canister locking handle counterclockwise to unlocked position (down).
2. Pull canister off pins.
3. Place folded tape in canister on spring clips so that beginning of tape extends downward from carrying spring clips.
4. Thread tape around pins on bottom of inside of canister, up right side (when facing open canister with its top up), around pins on top of inside of canister, and down through guide jaws in center of canister (draw tape taut and sufficiently far through guides to enable refolding procedure of read tape to begin properly; ends of tape may be spliced to form continuous loop if desired; if a loop is formed, the tape can be advanced to the starting point from any place on the tape by depressing the Tape Advance button on the tape reader panel).
5. Replace canister on tape reader over pins.
6. Turn canister locking handle clockwise to locked position (up).
7. Depress keyboard fill source button, if necessary, to light the keyboard fill source light.
8. Depress Fill button on tape reader panel (tape will feed through reader until (1) end of tape passes through read head, (2) the reader detects an S or H code, or (3) the Stop button on the tape reader panel is depressed).

To operate in Verify mode:

1. Place tape in canister as outlined in steps 1 through 6 under Fill mode procedure.
2. Depress keyboard fill source button, if necessary, to light the keyboard fill source light.
3. Depress Verify button on tape reader panel (tape will feed through reader until (1) end of tape passes through read head, (2) a Verify error is detected, or (3) the Stop button on the tape reader panel is depressed).

To splice tape, when repairing or forming loops:

1. When repairing tape, repunch deleted information on new tape, preceding and following repunched section with at least two inches of "code delete" code (splice can be in middle of word without disturbing program; this step unnecessary when forming loop if leaders are long enough).
2. Cut one section of tape between 2" and 6-1/2" from last fold (tape should be cut squarely through a sprocket hole rather than on diagonal).

3. Place uncut section under cut section, maintaining 8-1/2" ± 1/32" between folds, and mark uncut section with pencil (using cut section as template).

4. Cut uncut section on a sprocket hole such that overlap will contain two complete sprocket holes when splice is completed.

5. Clean tape ends with acetone to remove excess oil and dirt.

6. Apply household cement or plastic cement on both surfaces to be joined and press firmly together (sprocket holes must coincide and edges of tape must be parallel; thickness of splice must not be greater than 0.010" at any point).

7. Remove excess cement from sprocket and information holes with pointed instrument.

8. Allow splice to dry at least 10 minutes before using tape.

**Paper Tape Punch Operating Procedure**

Punch operating procedures are the same for preparation of tape manually from the typewriter as for direct output from the computer, except that for tape preparation the Computer-Manual Punch switch is placed in Manual Punch position and for output it is placed in Computer position.

To apply power to punch, computer power need not be turned on unless punch will be used for output, turn switch on circuit breaker at front of desk and switch for the punch on.

To prepare punch for operation:

1. Load punch with paper tape as directed on rear of tape punch front and rear canister doors.

2. Place Computer-Manual Punch switch in proper position according to function to be performed.

3. Place Punch-External switch in Punch position.

4. Depress Tape Advance button to punch leader of sprocket holes only (leader should be sufficiently long to enable at least one fold in tape to lie properly in lower left corner of front canister. During operation tape feeds and stops automatically under both computer and typewriter control).
5. After punching of tape is completed, depress Tape Advance button to punch leader of sufficient length to enable at least one fold of tape to lie properly in lower left corner of front canister.

6. Tear off tape and remove from canister as directed on canister door.

**NOTE**

It is wise to punch an H (halt code) at the end of the tape, so the photoreader machine will halt when the tape is read into the machine. This is done by switching the computer-manual switch to the manual position and depressing the H key on the typewriter. The computer-manual switch should be returned to the computer position.
LOGIC DESCRIPTION

GENERAL

The technical world is witnessing a revolution in the field of mathematical computation that is rapidly and radically affecting all of man's science, technology, and mechanics. The great advance in technology is akin to the industrial revolution of the 18th and 19th centuries. This technical revolution was initiated by the recent introduction of mechanical computers. Because of the electrical nature of present-day electronic computers, these machines can process in a few milliseconds involved mathematical problems so lengthy and complicated as to be heretofore entirely beyond the scope of human operators. A wide and ever-increasing variety of mathematical problems is now subject to quick and accurate resolution, the results of which are directly applicable to the physical sciences. With the advent of the electronic computer, man has achieved an advantage over his environment that will increasingly enable him to understand and control his environment.

Basic Operation

The RECOMP II in its basic operation performs in the simplest mathematical processes. A reasonably complicated numerical calculation is broken down into its basic elements and the resulting primary computations are performed by computer internal logic. These computations could be accomplished as well by any literate person; the one element that enables RECOMP II to out-calculate any human operator is its speed. In one eight-hour period a modern digital computer is capable of performing a block of calculations that could not be solved by the effort of all living mathematicians and their descendants all similarly trained in many thousands of years.

Computer Analysis

At first glance a computer is a very complex device. To simplify an analysis, figuratively divide the concept into two parts: one consists of its components, mechanical and electronic; the other treats of its functions. The first is made up mostly of commonly known electronic elements such as resistors, condensers, diodes, and transistors, all fabricated on plug-in printed circuit boards; specialized elements
include nixie neon tubes for readout display, a rotating magnetic oxide-coated memory disk, read and write heads that function in conjunction with the memory disk, and input-output equipment such as the electric typewriter, the tape punch, the tape-reader, and the control console. Other manuals treat the specific physical components of the computer. The explanation that follows will be principally concerned with the functional concept of the computer, namely, the function of its electronic and mechanical elements in an integrated system capable of handling mathematical problems and computing required results.

Classes of Operations

Any operation that can be reduced to simple logic or arithmetic can be handled by the RECOMP II. There does not seem to be any theoretical limit to the types of problems that it can handle. Although it is sometimes difficult to reduce a given problem to its primary elements for insertion into the computer, problems have been found to fall into distinct classes to which solution methods have been previously worked out (frequently by the use of the computer itself) and recorded in specific computer programs. The work of analyzing a complex problem may be frequently reduced to identifying it within its class, after which an operation is performed by the application of an available program or combination of programs. The latter is the work of the programmer who usually operates the computer, prepares the programs, and resolves the problems.

Because a digital computer is an information processing device, processing is the only operation it can perform. It cannot create any information. It may transform the input information into a more useful form or it may decide the mode of process operation, basing its decision upon the result of intermediate calculations; but the information that permits the machine to make such a decision is the work of the programmer and must be programmed into the machine at the outset of the computation.

The input to be processed may be the statement of a mathematical problem, rules governing the mathematical operations to be performed (commands), and the data to be operated upon. The digital computer is able to handle as wide a variety of problems as the programmer is able to prepare for computer processing, but in every case the computer simply processes the input data according to a programmed format specified by the operator and in a manner compatible with its internal logic circuitry. The computer makes use of more than one source of information during the processing; the input data, the list of operating instructions, and any intermediate commands for changing processing procedure based
upon results of calculations. Mathematical processes are partly built into the wiring of the computer as inalterable mathematical rules and partly entered as instructions for a particular problem. Information processing is performed by rearranging information from one relationship at the input to another at the output, not only in mathematical configuration but also in the dimension of time. In presenting a problem to the computer, a program or list of instructions is entered and subsequently the data on which the operations are to be performed. The computer must be able to remember the instructions until it receives the data, and because most of the operations cannot be performed simultaneously a continued use of memory or information storage is required.

**Information Switching and Information Storage**

Two functions are basic to the computer; information switching and information storage. Switching is the mathematical arrangement of information within the computer; information storage is the timed sequence handling of the information. Switching and storage in the computer are closely related and, in many functions, interrelated and interdependent. Most of the computer switching devices (flip-flops, gates) contain elements of memory or of information storage, and most of the storage devices (loops, memory disks, delay flip-flops) require switching mechanisms for access. Most switching and storage devices are binary in physical nature; ON or OFF. They exhibit two stable states. Consequently the computer is based on the binary system of numbers one and zero.

**Digits**

A digital computer uses digits or numbers. The basic computational process of RECOMP II operates through the use of binary digits, one and zero. These two digits arranged in an ordered configuration stand for all numbers, signs, factors, and commands necessary to the solution of any given problem. In computer terminology, digits refer to numbers, whether binary, octal, decimal, or of other mathematical description. In binary notation, a single binary digit is known as a bit. A group of digits (specified number) is known as a word. A group of several digits or bits, whether representing numerals or alphabetical letters, is also known as a character. A word may consist of several characters.

**Subassemblies**

In examining the basic mechanisms involved in a computer the machine and the computational process were viewed wholly. To analyze
the computer in detail, it is desirable to group the switching and storage elements into subassemblies which perform specialized functions in terms of the flow of information through the machine. First, the programer (operator) must get information into and out of the computer. This requires input and output devices which operate through input and output channels and incorporate both storage and switching functions. Input devices consist of typewriter, tape reader, keyboard; output devices consist of typewriter, tape punch, Nixie readout. Second, the operation to be performed on the input data is either logical or arithmetic and requires a logical and arithmetical unit (which is referred to in RECOMP II as a register). Third, instructions and intermediate data are stored in a storage device or memory. In RECOMP II, the storage device is a rotating magnetic oxide-coated disk upon the surface of which is magnetically inscribed the required digital information to be electronically recalled as required. Fourth, a control unit is required in computer operation. Control largely involves switching, but elements of timing are also necessary. The control portion of RECOMP II contains counters to keep track of the computational steps being performed, registers to direct the information along the proper logical paths, and a timing device known as the clock to synchronize the entire operation of the computer.

**Programing**

Programing is the fifth element of the computer concept. Programing is the direction of the control unit by the operator. Fixed programing is the part of the program built into the circuitry of the computer and available in modes to the programer. Once started along a certain control path, the computer will proceed through a fixed series of operations to a final result, processing the information fed into the machine in a rigid and predetermined manner. By selecting various fixed modes of control, the programer is able to cause some of the program steps to sample intermediate data, to transfer control to selected control points, to skip steps in programing if certain calculations occur, to stop the program, and to store in memory or in recirculation loops. The machine simply operates as directed and uses the result of its computations to perform certain other operations prescribed in advance. The term "programing" refers to those phases of the planning of a problem that consist of recording the steps required, the intermediate sampling of data desired, and the way in which the solution steps are to be altered as a result of this sampling.

**Stored Program**

The stored program consists of a coded notation of instructions in acceptable form for entry into the computer and which is entered with
the data alternately and in a programmed sequence in such a manner as to operate upon the data to resolve the problem and produce an output in desired form.

Instructions

An instruction consists of two parts; an operation part which instructs the computer what to do, and an address part which designates the specific portion of memory in which to store information or from which to read information. The instruction contains the address code of the word upon which an operation is to be performed rather than the word itself. This simplifies the substitution of data, permitting the stored data to be changed without in any way changing the instructions and vice versa. Computers constructed on this principle are known as single-address machines. RECOMP II is a single address machine. RECOMP II does not require a program counter since it takes instructions in the order in which they are listed unless a transfer of control is specified as an instruction.

There are a number of kinds of operation that RECOMP II can perform (the list of operations is available in a separate operating manual). They are coded for convenience in entering them into the computer and are referred to as commands. They may be classified into several logical categories: (a) instructions relating to input and output devices, (b) instructions enabling the operator to store information in memory or to recall it as desired, (c) mathematical instructions such as to add, subtract, multiply, and divide, (d) control transfer operations and halt commands. The stored program method of operation used in RECOMP II provides an ability to operate on the instructions as though they were ordinary data. This means that the entire course of a computation can be altered including the operations themselves, the choice of data upon which the operations are to be performed, and the location in memory at which the results are to be stored. These changes can all be done on the basis of the results obtained in the course of the calculations to resolve the problem.

The programmer need not write out every operation that he wishes the machine to perform. He may work out procedures for portions of the problem and then arrange a master program employing these 'subprograms'; the machine modifies and adapts these subprograms as required during various stages of problem solution.
Summary

Programming in its advanced form is certainly an art. Much of the apparent complexity of the computer resides in the program rather than the machine itself. A large number of basic and specialized programs have been worked out for RECOMP and new programs are constantly being devised for application in the ever-increasing variety of fields.

Programming has been briefly explained in this introduction because of a need to know some of its basic concepts in an approach to the understanding of the logic elements of the computer. Conversely, an intimate knowledge of the logic elements of the computer will broaden the programmer's ability to use the computer as a useful tool to its fullest capacity.

LOGIC EQUATION ANALYSIS

This section contains an analysis of the logic of RECOMP II in all modes of operation. It includes an analysis of computer timing, control, and information handling including input/output functions, and an examination of the system tester as it is utilized to determine the functional condition of the logical elements within the computer. Simplified dataflow diagrams are provided to facilitate a ready understanding of the logic functions; engineering drawings and schematics are available in Manual 508-T-11 for detailed reference to the circuitry and for point-to-point signal tracing.

The logical design of RECOMP II was achieved entirely by application of Boolean algebra; in this section all functional computer processes will be explained on the basis of this technique. For those who are familiar with this notation, a recognition of the notational style is all that is required to understand the equations that follow; for those who are not accustomed to Boolean representation of logic circuitry, and for clarification of the style of notation used in this manual, the following general conditions are expressed:

Flip-Flops

Flip-flops are bistable electronic switches whose two stable states are designated TRUE and FALSE:

![Flip-Flops Diagram]

TRUE

FALSE
A flip-flop is considered set to a true state and reset to a false state:

![Flip-flop diagram]

Each flip-flop is symbolized by a letter of the alphabet, for example, the sum flip-flop, S. Flip-flops are more specifically identified by a modifying subscript as in the case of one term of the input/output register, F₅. The true and false states of a flip-flop are described respectively by the upper-case letter (with modifying subscript) as in the example F₅ and its prime notation, F₅' (F-five-prime).

<table>
<thead>
<tr>
<th>F₅</th>
<th>TRUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>F₅'</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

When the flip-flop is set to the true or false state, it is identified by a function symbol specified by lower-case letters preceded by a "one" or "zero" written as follows:

\[
\begin{align*}
1^f_5 \\
0^f_5
\end{align*}
\]

When relating the flip-flop state to the binary number for which it stands (one or zero), it may be useful to make an analogy to a child's teeter-totter: when the flip-flop is set to the true state and the F₅ "end" is up, then F₅ = 1; at the same time the F₅' "end" is down and F₅' = 0. Conversely, when the flip-flop is reset to the false state and the F₅' "end" is up, then F₅' = 1; at the same time the F₅ "end" is down and F₅ = 0. By application of simple logic, for a given flip-flop state (true or false), when one term of the flip-flop equals one, the other must equal zero; change the state of the flip-flop and the terms exchange values -- the first term becomes equal to zero and the second term becomes equal to one.
The designation $l_{5}$ translates in linguistic form to "one-set-F-five"; $0_{5}$ translates to "zero-set-F-five". Specified input conditions are required to effect a one-set or zero-set operation, and these conditions are represented by equations as in the examples that follow:

$$l_{5} = K_{5} I_{4} T_{pp}$$

translates to: one-set F-five by K-five and I-four and T-p-p

$$0_{5} = Z_{1} N_{6}$$

translates to: zero-set F-five by Z-one-prime and N-six.

There may be a single specified input condition to effect the setting or resetting of a given flip-flop, but there are usually multiple conditions deriving from various computer functions. These multiple conditions are expressed by either the logic "and" sign ($\cdot$) or the logic "or" sign ($\pm$), or both in combination, and may be found for any particular function in the listing of logic equations included in this manual. The logic "and" sign ($\cdot$) is not written between the terms of a conditional equation but is understood:

$$l_{5} = K_{5} I_{4} T_{pp} \text{ is written } l_{5} = K_{5} I_{4} T_{pp}$$

Amplifying the previous example:

$$l_{5} = K_{5} I_{4} T_{pp} + Z_{1} N_{6} + F_{4} F_{3} F_{1} C + (etc)$$

$$0_{5} = Z_{1} N_{6} + N_{12} + P_{d3} T_{40} + (etc) \text{ (from Appendix II, Logic Equations)}$$

This is read as one-set F-five by K-five and I-four and T-pp OR Z-one and N-six OR F-four-prime and F-three-prime and F-one and C-t OR (etc).

The step beyond linguistically forming the equational conditions of the computer logic is to interpret the symbols of the functions: $F_{5}$ is one of five input-output register flip-flops; $K_{5}$ is one of the input channel switches; $I_{4}$ is a flip-flop that signifies one of the five operational states of the computer; $T_{pp}$ is the tape-punch-pulse flip-flop. In a final interpretation of the equation $l_{5} = K_{5} I_{4} T_{pp}$, the number five flip-flop ($F_{5}$) of the input-output register is one-set by the TRUE state of the number five input channel switch ($K_{5}$) when the computer is in $I_{4}$ state of operation and the tape-punch-pulse flip-flop is also in the TRUE state.
An interpretation and brief analysis of the conditional terms in the above examples as well as for all RECOMP flip-flops, registers, counters, etc, may be found in Appendix III, Circuit Board Charts.

**Timing**

The computer functions in a serial progression and the functions are initiated by a series of timing pulses. All flip-flop state changes are "triggered" by these pulses called "CLOCK" pulses. The timing pulses occur at regular intervals \( t_1, t_2, t_3, \ldots \ldots \ldots \text{to} \ t_{41} \) and are known as "bit" times written as the equational term "C". An equational condition coincident with a clock pulse (C) changes a flip-flop to the state indicated for the following bit time \( (t + 1) \). All equations have as a final term, whether written or not, the clock term \( C \): 1\(_{5}^{f5} = K \cdot I \cdot T \cdot pp \cdot C \). This form of computer timing is known as synchronous timing.

**Gates**

The decision elements in the computer are comprised of diode "gates." Two types of diode gates operate to interpret data and transmit control signals: these are described as "AND" and "OR" gates. An AND gate requires all of its inputs to be true in order for its output to be true. An "OR" gate requires at least one of its inputs to be true in order for its output to be true. Diode AND gates contain anywhere from two to nine input terms (including the enabling clock pulse). As many as twenty OR gates are used to trigger any one flip-flop. An AND gate is indicated by the symbol \( \cdot \). An OR gate is described by the symbol \( + \) containing an interior plus sign.

**Boolean Algebra**

For ease in understanding simplifications of Boolean equations, several useful relationships are presented. Truth tables, following, serve to verify (a) through (k). The fundamental laws (l) through (p) are stated here without proof.

\[
\begin{align*}
(\text{a}) & \quad A0 = 0 \\
(\text{b}) & \quad A + 0 = A \\
(\text{c}) & \quad A1 = A \\
(\text{d}) & \quad A + 1 = 1 \\
(\text{e}) & \quad AA = A \\
(\text{f}) & \quad AA' = 0 \\
(\text{g}) & \quad A + A' = 1
\end{align*}
\]
(h) \( A + AB = A \)

(i) \( A + A'B = A + B \)

(j) \( (AB)' = A' + B' \) \text{ DeMorgan's Theorem}

(k) \( (A + B)' = A'B' \)

(l) \( AB = BA \) \text{ Commutative Law}

(m) \( A + B = B + A \)

(n) \( A(BC) = AB(C) = ABC \) \text{ Associative Law}

(o) \( A + (B + C) = (A + B) + C = A + B + C \)

(p) \( A(B + C) = AB + AC \) \text{ Distributive Law}

<table>
<thead>
<tr>
<th>A</th>
<th>A'</th>
<th>(a) A0</th>
<th>(b) A + 0</th>
<th>(c) A1</th>
<th>(d) A + 1</th>
<th>(e) AA</th>
<th>(f) AA'</th>
<th>(g) A + A'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

(h) \( A \quad B \quad A' \quad B' \quad AB \quad A + AB \quad A'B \quad A + A'B \quad (AB)' \quad A' + B' \quad (A + B)' \quad A'B' \)

(i) \( 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \)

When two Boolean functions, \( A \) and \( B \), are so related that \( A \) is true whenever \( B \) is true, though not necessarily conversely, then we say that \( A \) is included in \( B \), written \( A \subseteq B \). If also \( B \subseteq A \), then obviously \( A = B \). From the preceding truth table, we see that \( AC \subseteq AB \).

Example of Logical Equations: To illustrate the method of deducing a set of logical equations, let us design a simple 3-stage binary counter that increases by "one" whenever a clock pulse is received up to a count of 7, resets itself to zero and repeats the process, ad infinitum. Table 4 shows the truth table for such a counter, where flip-flops \( X_3, X_2 \) and \( X_1 \) contain the bits in descending order of significance.
Table 4. Truth Table for 3-Stage Pulse Counter

<table>
<thead>
<tr>
<th>Time t</th>
<th>Time (t + 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₃</td>
<td>X₂</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

It is observed from table 4 that if X₁ = 0 at time t, then X₁ = 1 at time t + 1. In other words X₁ is set to the true state whenever it is in the false state, X₁', and a clock pulse, C, is received, or

\[ 1^X_1 = X_1'C \]

Similarly, X₁ is set to the false state whenever it is in the true state, X₁, and a clock pulse is received, or

\[ 0^X_1 = X_1C \]

Further, X₂ is changed to the true state when it is false and X₁ is true and a clock pulse is received, or

\[ 1^X_2 = X_2'X_1C \]

while X₂ is changed to false when both itself and X₁ are true and a clock pulse is received, or

\[ 0^X_2 = X_2X_1C \]

Finally, X₃ changes to the opposite state whenever both X₂ and X₁ are true and a clock pulse is received. The complete set of logical equations for this counter are therefore:
It may be noted that any stage in a binary counter changes state whenever all less significant stages are true, or

\[
\begin{align*}
X_1 \quad 1^X_1 &= X_1'C \\
0^X_1 &= X_1C \\
X_2 \quad 1^X_2 &= X_2'X_1C \\
0^X_2 &= X_2X_1C \\
X_3 \quad 1^X_3 &= X_3'X_2X_1C \\
0^X_3 &= X_3X_2X_1C \\
\end{align*}
\]

This means simply that adding a 1 changes the 0 in binary 0111111 to 1 or the leading 1 in 111111 to 0.

Summary

This, then, is the step-by-step analysis the engineer or technician will follow in reading RECOMP II logic equations. He may devise his own personal mental shortcuts and abbreviations in analyzing the equational notations, and as his familiarity with the whole system increases, the symbols will come to represent the elements of the computer circuitry, operating conditions, and logic resolution. When a competent RECOMP II maintenance engineer or technician is referred to the term I, he immediately understands the reference to be one of the operational states of the computer, the first step in a master sequence control which involves, among other things, a selection of the required command pair from memory and the gating of the command pair into the Z-register. There are a large number of flip-flops (approximately 180), gates (more than 10,000), and numerous other elements and functions designated by letters and subscripts connected with the computer logic. Because of the complexity of their interrelationships and interdependence, a thorough and intimate knowledge of the majority of the symbols and even of a large number of the more common multiple-symbol configurations used as conditional terms in the logic equations is desirable in order that they may be readily identified on sight. Working daily with the equations will lead to an understanding more easily than may be at first anticipated. It has been
found by an ever-increasing number of engineers and technicians that the logic notational form is by far the least complex and the most articulate method of logic circuit description. A compendium of logic equations representing all logic functions within the computer will be found in Appendices I and II of this manual. Further knowledge of Boolean algebra and boolean relationships may be found in Montgomery Phister's LOGICAL DESIGN OF DIGITAL COMPUTERS published by John Wiley & Sons, Inc, New York and London.

COMPUTER CHARACTERISTICS

RECOMP II is a general-purpose digital computer that performs all internal operations in the binary number system. The memory is a rotating magnetic disk, and input equipment consists of a photoelectric paper tape reader, an electric typewriter, and a control unit keyboard. Output information may be punched on paper tape, typed, or displayed on the control panel visual readout. All computer circuits are transistorized and diodes are used for logic gating.

Memory

The magnetic disk memory unit provides basic timing channels, arithmetic registers, rapid access loops, and information storage channels.

Information Storage

Sixty-three channels, each containing 64 words of information and one channel containing 48 words comprise the 4080-word addressable main storage. This portion of the memory permits nonvolatile information storage. The magnetic oxide-coated disk surface retains stored information until erasure is desired, even during power interruptions or power-off condition. Words in memory are addressed by four octal digits from 0000 to 7757. The first two digits identify the memory channel; the last two denote the sector or word within the channel. The average access time to stored information is one-half a disk revolution or approximately 9.0 milliseconds.

Rapid Access Loops. In order to provide rapid access to stored information, two high-speed recirculation loops, L and V, of eight words each are available in programing. Loop addresses are indicated from octal 7760 to 7767 and from 7770 to 7777, respectively. The least
A significant digit specifies which of the eight words in a loop is addressed. Access to loop information averages 0.95 millisecond. Information contained in the loops is volatile, subject to erasure during power interruption or power-off condition.

Arithmetic Register. Five recirculating loops or registers, each containing one word, are located on the memory disk. The A-register or accumulator stores the result of any arithmetic operation. The R-register stores the remainder in division, the least significant half of a double-length product or dividend, or the exponent of a floating-point number. The Z-register contains the command pair being obeyed at a given time. The B-register holds the number whose address is found in the command. The R-register holds the exponent of a floating-point number. The B and Z-registers are not directly accessible to the programmer.

Timing Channels. Two permanently-recorded tracks on the disk provide basic timing controls for the computer operation. The clock channel contains a sine wave with a nominal frequency of 151 kc, at the nominal motor speed of 3450 rpm. After wave shaping this results in 2624 equally-spaced pulses which control the triggering of logic gates on the inputs to the flip-flops. Clock pulses are denoted by the letter C in the logic equations. The origin channel has a single pulse that serves as a reference for the nonvolatile storage by resetting counters used in memory addressing.

Word Format

A word consists of 41 bits (binary digits) which may represent either numerical quantities to be operated upon or computer commands that specify the operations to be performed. The right-most bit is used for memory synchronization and is not available in programming. The word formats for numerical quantities and commands are shown in figure 10.

Numbers. If a word contains a numerical quantity, the left-most bit stores the sign (1 if plus, 0 if minus) followed by the absolute value of the number as shown in figure 10. Normally, the computer assumes the binary point to be between the sign and the most significant bit (i.e., numbers are considered less than "one" in absolute value).

Commands. Single-address commands are stored two to a word. Each command has a sign bit, an operation code of two octal digits (6 bits), and an address containing four octal and one binary digit (13 bits) as shown in figure 10. The sign associated with a command has no effect on its execution except when sense switches are used. The operation code determines the arithmetic or logic operation to be performed. The address portion is the memory location of an operand for arithmetic commands, the number of binary shifts for a
NUMERICAL QUANTITY

Assumed position of binary point

| 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign</td>
<td>Absolute value</td>
<td>of numerical</td>
<td>quantity</td>
<td>(39 bits)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COMMAND PAIR

| 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign</td>
<td>Operation Code</td>
<td>Channel Address</td>
<td>Sector Address</td>
<td>Operation Code</td>
<td>Channel Address</td>
<td>Sector Address</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
<td>Command Ind</td>
</tr>
</tbody>
</table>
shift command, the octal code for punching and printing, or it has no significance at all. For arithmetic commands, the four octal address digits refer to the memory channel and sector of the operand. Depending on whether it is a 0 or 1, the additional bit specifies the first or second command in a word. This command indicator bit is significant only in the address of transfer-of-control and partial substitution commands.

Command List

The commands available in RECOMP are shown in table 5 with their octal operation codes, execution times, and explanatory notes.

Table 5. RECOMP II Commands

<table>
<thead>
<tr>
<th>Word Times</th>
<th>Numerical Code</th>
<th>Operation Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed-Point Arithmetic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 + T 00</td>
<td>CLA w</td>
<td>Clear and add.</td>
<td>(w) → A</td>
</tr>
<tr>
<td>2 + T 01</td>
<td>ADD w</td>
<td>Add. (w) + (A) → A</td>
<td></td>
</tr>
<tr>
<td>1 + T 02</td>
<td>CLS w</td>
<td>Clear and subtract.</td>
<td>- (w) → A</td>
</tr>
<tr>
<td>2 + T 03</td>
<td>SUB w</td>
<td>Subtract. (A) - (w) → A</td>
<td></td>
</tr>
<tr>
<td>40 + T 13</td>
<td>MPR w</td>
<td>Multiply and round. (w) X (A) → A</td>
<td></td>
</tr>
<tr>
<td>40 + T 11</td>
<td>MPY w</td>
<td>Multiply (unrounded). (w) X (A) → A</td>
<td></td>
</tr>
<tr>
<td>42 + T 21</td>
<td>DSR w</td>
<td>Divide single length and round. (A)/(w) → A</td>
<td></td>
</tr>
<tr>
<td>41 + T 20</td>
<td>DSL w</td>
<td>Divide single length (unrounded). (A)/(w) → A, R (rem)</td>
<td></td>
</tr>
<tr>
<td>42 + T 23</td>
<td>DVR w</td>
<td>Divide (double length) and round. (A, R)/(w) → A</td>
<td></td>
</tr>
<tr>
<td>41 + T 22</td>
<td>DIV w</td>
<td>Divide (double length unrounded). (A, R)/(w) → A, R (rem)</td>
<td></td>
</tr>
<tr>
<td>1 + T 60</td>
<td>STO w</td>
<td>Store. (A) → w</td>
<td></td>
</tr>
<tr>
<td>* 42</td>
<td>STA w</td>
<td>Store address. (A)a → w</td>
<td></td>
</tr>
<tr>
<td>2 + n 40</td>
<td>ARS n</td>
<td>Accumulator right shift. Shift (A) n right, sign untouched.</td>
<td></td>
</tr>
<tr>
<td>2 + n 41</td>
<td>ALS n</td>
<td>Accumulator left shift. Shift (A) n places to the left, sign untouched.</td>
<td></td>
</tr>
<tr>
<td>42 + T 25</td>
<td>SQR w</td>
<td>Square root. √(w) → A</td>
<td></td>
</tr>
</tbody>
</table>

*Main Memory: 32 + T; Rapid Access Storage: 4 or 8 + T

write
<table>
<thead>
<tr>
<th>Word Times</th>
<th>Numerical Code</th>
<th>Operation Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Floating clear and add ((w, w + 1))(\rightarrow) A, R.</td>
</tr>
<tr>
<td>1 + T</td>
<td>30</td>
<td>FCA w</td>
<td>(w, w + 1) + (A, R)(\rightarrow) A, R.</td>
</tr>
<tr>
<td>5 + D + T</td>
<td>04</td>
<td>FAD w</td>
<td>Floating subtract. (-(w, w + 1))(\rightarrow) A, R.</td>
</tr>
<tr>
<td>1 + T</td>
<td>34</td>
<td>FCS w</td>
<td>Floating subtract. ((A, R) - (w, w + 1))(\rightarrow) A, R.</td>
</tr>
<tr>
<td>5 + D + T</td>
<td>06</td>
<td>FSB w</td>
<td>Floating subtract. ((A, R) - (w, w + 1))(\rightarrow) A, R.</td>
</tr>
<tr>
<td>46 + D + T</td>
<td>07</td>
<td>FMP w</td>
<td>Floating multiply. ((w, w + 1) \times (A, R))(\rightarrow) A, R.</td>
</tr>
<tr>
<td>47 + D + T</td>
<td>05</td>
<td>FDV w</td>
<td>Floating divide. ((A, R)/(w, w + 1))(\rightarrow) A, R.</td>
</tr>
<tr>
<td>1 + T</td>
<td>35</td>
<td>FST w</td>
<td>Floating store. ((A, R))(\rightarrow) (w, w + 1).</td>
</tr>
<tr>
<td>43 + S + T</td>
<td>44</td>
<td>FSQ w</td>
<td>Floating square root. (\sqrt{(w, w + 1)})(\rightarrow) A, R.</td>
</tr>
<tr>
<td>2 + n</td>
<td>45</td>
<td>FNM</td>
<td>Floating normalize. ((A, R)) normalized.</td>
</tr>
</tbody>
</table>

**Input - Output**

<table>
<thead>
<tr>
<th>Operation Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDY w</td>
<td>Read from Y reader. Location counter set to w.</td>
</tr>
<tr>
<td>RDZ w</td>
<td>Read from Z reader. Location counter set to w.</td>
</tr>
<tr>
<td>DIS w</td>
<td>Display. Display ((w)) on display register.</td>
</tr>
<tr>
<td>PNC n</td>
<td>Punch character. Punch character corresponding to (n).</td>
</tr>
<tr>
<td>TYC n</td>
<td>Type character. Type character corresponding to (n).</td>
</tr>
<tr>
<td>PTC n</td>
<td>Punch and type character. Punch and type character corresponding to (n).</td>
</tr>
<tr>
<td>PNW w</td>
<td>Punch word. Punch ((w)).</td>
</tr>
<tr>
<td>TYW w</td>
<td>Type word. Type ((w)).</td>
</tr>
<tr>
<td>PTW w</td>
<td>Punch and type word. Punch and type ((w)).</td>
</tr>
</tbody>
</table>

\(N = \text{Number of positions on the display register (excluding initial sign) activated by instructions.}\)
<table>
<thead>
<tr>
<th>Word Times</th>
<th>Numerical Code</th>
<th>Operation Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Block Transfers</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 + T</td>
<td>64</td>
<td>CTL w</td>
<td>Copy to L loop. (w) to (w + 7) → 7760 to 7767.</td>
</tr>
<tr>
<td>7 + T</td>
<td>65</td>
<td>CFL w</td>
<td>Copy from L loop. (7760) to (7767) → w to w + 7.</td>
</tr>
<tr>
<td>7 + T</td>
<td>66</td>
<td>CTV w</td>
<td>Copy to V loop. (w) to (w + 7) → 7770 to 7777.</td>
</tr>
<tr>
<td>7 + T</td>
<td>67</td>
<td>CFV w</td>
<td>Copy from V loop. (7770) to (7777) → w to w - 7.</td>
</tr>
<tr>
<td><strong>Control</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>TZE w</td>
<td>Transfer on zero. If ((A) = \pm 0), transfer to w.</td>
</tr>
<tr>
<td>3</td>
<td>52</td>
<td>TPL w</td>
<td>Transfer on plus. If ((A)_s = +), transfer to w.</td>
</tr>
<tr>
<td>3</td>
<td>51</td>
<td>TMI w</td>
<td>Transfer on minus. If ((A)_s = -), transfer to w.</td>
</tr>
<tr>
<td>3</td>
<td>53</td>
<td>TOV w</td>
<td>Transfer on overflow. If overflow occurs, transfer to w.</td>
</tr>
<tr>
<td>3</td>
<td>57</td>
<td>TRA w</td>
<td>Transfer (unconditional). Transfer to w.</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>TSB w</td>
<td>Transfer on switch B. If on, transfer.</td>
</tr>
<tr>
<td>3</td>
<td>55</td>
<td>TSC w</td>
<td>Transfer on switch C. If on, transfer.</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>TSD w</td>
<td>Transfer on switch D. If on, transfer.</td>
</tr>
<tr>
<td>3</td>
<td>77</td>
<td>HTR w</td>
<td>Halt and transfer. Stop computer and reset location counter to w.</td>
</tr>
<tr>
<td><strong>Miscellaneous</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 + T</td>
<td>33</td>
<td>EXT w</td>
<td>Extract. Erase all bits in A except those in which corresponding bits in w are &quot;ones&quot;.</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>XAR</td>
<td>Exchange A and R. Interchange ((A)) and ((R)).</td>
</tr>
<tr>
<td>1 + T</td>
<td>15</td>
<td>SAX</td>
<td>Store and exchange A and X. ((A) \rightarrow w), interchange ((A)) and ((X)).</td>
</tr>
</tbody>
</table>
Symbol

\[ w = \text{memory location whose address is } w \]
\[ (w) = \text{contents of } w \]
\[ A = \text{A-register} \]
\[ (A) = \text{contents of the A-register} \]
\[ R = \text{R-register} \]
\[ (R) = \text{contents of the R-register} \]
\[ X = \text{X-register} \]
\[ (X) = \text{contents of the X-register} \]
\[ w_a = \text{address portion of memory location } w \]
\[ (A)_a = \text{address portion of the contents of the A-register} \]
\[ (A)_s = \text{sign portion of the contents of the A-register} \]
\[ T = \text{access time} \]
\[ D = \text{difference in exponent parts of two floating-point numbers} \]
\[ n = \text{number of shifts required by instruction} \]

Timing Information

Main memory: \(33.5 \text{ word times, average access time}\)

Rapid Access Storage

Read: \(3.5 \text{ word times, average access time}\)
Write: \(5.5 \text{ word times, average access time}\)

1 word time: \(0.27 \text{ millisecond}\)

Basic Computer Operation

The functional operation of the computer may be regarded as shown in figure 11. The solid lines represent information flow and the dashed lines indicate functional control. Commands selected from memory control information flow and arithmetic processes.

Location Counter

The location counter is a 13-stage binary counter, designated \(G_{12}, G_{11}, \ldots, G_0\) which stores the memory address of the command. During execution of a command, the location counter is increased by "one" to the address of the following command. This normal sequencing of the location counter is broken only where a transfer-of-control command is to be obeyed. The location counter is set initially when filling the computer.
Master Sequence Control

Normal computer operation requires that three different functions take place successively (see figure 12). First, the word containing the desired command is located and transferred from the memory to the Z-register. This is called command selection, and is identified with the true state of the timing flip-flop I_1. Since I_1 involves the selection of a full word from memory, this function is not repeated for the second command in a pair.

During the second function, command interpretation, the word referred to by the address of the command is located and transferred from memory to the B-register. I_2 is the flip-flop associated with command interpretation. Third, the command is executed whereby the required arithmetic or logical operation is performed. Flip-flop I_3 represents command execution. During execution of each command the location counter (G) is augmented to its succeeding value unless a transfer-of-control command is encountered. The master sequence control cycle is then repeated for the new command pair and so on until a stop command or error turns...
Figure 12. RECOMP Master Sequence Control Cycle

RECOMP to the noncompute state \( I_4 \). The master control sequence is initiated and the COMPUTE indicator is lighted by the control panel START button or the "S" character from the typewriter or the tape reader.

Noncompute Modes

Computer operation as described is suspended whenever flip-flop \( I_4 \) becomes true. During the \( I_4 \) state, however, any of the following input-output functions may be performed.

Fill Computer

Filling commands and numbers into memory and setting the command counter is accomplished through operation of any one of the input devices; keyboard, tape reader, or typewriter.

Verify Input

Verification of internally stored information may be achieved by comparison with the original input tapes. This function is performed automatically by pressing the VERIFY button on the tape reader and reading the tape.
Readout

Visual readout of information stored in any memory position or arithmetic register is possible in command, octal, or decimal format. The three possible binary groupings which allow these formats are initiated by the three readout buttons marked DECIMAL, OCTAL, and COMMAND.

Timing Source

The memory clock channel furnishes clock pulses necessary for all flip-flop triggering. Binary digit and sector counters and special timing flip-flops maintain the proper memory-referencing index.

Origin Flip-Flop

The single pulse on the origin channel is sensed by a read head and amplified by the origin pulse amplifier, $X_o$. $X_o$ sets the origin flip-flop, $X_o$, which provides counter and timing flip-flop synchronization. After being set to the true state by the origin pulse, $X_o$ resets itself to the false state during the bit time immediately following:

$$
\begin{array}{c|c}
0^X_o & X_C' \\
1^X_o & X_C \\
\end{array}
$$

Young Digit Counter

The Young digit counter provides the basic bit time and interval information for the computer. The terms $P_2'$, $P_3'$, $P_4'$, $P_5$ operate as a shift counter; the terms $P_1$ and $P_6$ operate as conditional elements of and in conjunction with the counter. Its specific design goal is minimal logic, including the logic for associated timing flip-flops.

The memory reading and writing processes are serial operations, proceeding from least to most significant bit. The Young digit counter identifies each of the 41 bits within a given word and furnishes appropriate signals for control and arithmetic operations. The Young digit counter consisting of flip-flops $P_1$ through $P_6$, counts clock pulses modulo 41.

The counting cycle is set to zero (bit time 1) by $X_o$ or on completion an entire counting cycle. Logical equations for the Young digit counter are shown on the next page.
<table>
<thead>
<tr>
<th>( P_1 )</th>
<th>[ 1^{P_1} = P_5 'P_1 'X_0 'C + P_6 'P_4 'P_3 'X_0 'C ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ 0^{P_1} = P_5 'P_1 'C + X_0 'C ]</td>
<td></td>
</tr>
<tr>
<td>( P_2 )</td>
<td>[ 1^{P_2} = P_1 'X_0 'C ]</td>
</tr>
<tr>
<td>[ 0^{P_2} = P_1 'C + X_0 'C ]</td>
<td></td>
</tr>
<tr>
<td>( P_3 )</td>
<td>[ 1^{P_3} = P_2 'C ]</td>
</tr>
<tr>
<td>[ 0^{P_3} = P_2 'C ]</td>
<td></td>
</tr>
<tr>
<td>( P_4 )</td>
<td>[ 1^{P_4} = P_3 'X_0 'C ]</td>
</tr>
<tr>
<td>[ 0^{P_4} = P_3 'C + X_0 'C ]</td>
<td></td>
</tr>
<tr>
<td>( P_5 )</td>
<td>[ 1^{P_5} = P_4 'X_0 'C ]</td>
</tr>
<tr>
<td>[ 0^{P_5} = P_4 'C + X_0 'C ]</td>
<td></td>
</tr>
<tr>
<td>( P_6 )</td>
<td>[ 1^{P_6} = P_6 'P_5 'P_4 'P_3 'P_2 'C ]</td>
</tr>
<tr>
<td>[ 0^{P_6} = T_{41a} C ]</td>
<td></td>
</tr>
</tbody>
</table>

**COUNTER LOGIC**

It is observed in the logic for \( P_1 \) (figure 13) that \( P_5 'P_1 'C \) one-sets \( P_1 \) at the proper points except for \( T_{22} \). The \( X_0 'C \) flip-flop signifies the memory reset point or origin. \( X_0 'C \) allows the counter to count except when being reset. \( P_6 'P_5 'P_4 'P_3 'X_0 'C \) provides one-set of \( P_1 \) at \( T_{21} C \).

\( P_5 'P_1 'C \) zero-sets \( P_1 \) at the proper times and \( X_0 'C \) zero-sets \( P_1 \) for primary synchronization.

The logic for \( P_2 \) shows that it copies \( P_1 \) except at the origin.
Figure 13. RECOMP II Digit Counter
The logic for $P_3$ indicates a normal left shift. However, a question may arise about the validity of omitting the $X_0 C$ term in the zero-set logic. This will be considered subsequently. The logic for $P_4$ and $P_5$ indicates normal left shifting.

Observe in the logic for $P_6$ that at the end of each word time $P_6$ is zero-set by $T_{41} C$. When the counter is first turned on, $P_6$ may either be a "one" or a "zero". After a maximum of one word time, $P_6$ will be forced back to the proper sequence by $T_{41} C$.

At this point, $P_3$ and $P_6$ logic will be considered in greater detail. Note that the logic for both of these flip-flops does not contain the $X_0 C$ term for resetting the counter. Therefore, the random settings of these flip-flops, when the computer is first turned on, must be analyzed. The cases possible at $T_1$ (after the first origin pulse, $X_0 C$) are:

\[
\begin{array}{ccccccc}
P_6 & P_5 & P_4 & P_3 & P_2 & P_1 \\
\hline
\text{CASE I} & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{CASE II} & 0 & 0 & 0 & \overline{1} & 0 & 0 \\
\text{CASE III} & \overline{1} & 0 & 0 & 0 & 0 & 0 \\
\text{CASE IV} & \overline{1} & 0 & 0 & \overline{1} & 0 & 0 \\
\end{array}
\]

**Case I** - This is the proper array $T_1$ so no further discussion of this case is required.

**Case II** - The following truth table is constructed using the counter logic equations:

\[
\begin{array}{ccccccc}
P_6 & P_5 & P_4 & P_3 & P_2 & P_1 \\
\hline
T_1 & 0 & 0 & 0 & \overline{1} & 0 & 0 \\
T_2 & 0 & 0 & \overline{1} & 0 & 0 & 1 \\
T_3 & 0 & \overline{1} & 0 & 0 & 1 & 0 \\
T_4 & 0 & 0 & 0 & \overline{1} & 0 & 1 \\
\end{array}
\]

At the count of 4 the counter is back in step.
Case III

\[
\begin{array}{cccccc}
  P_6 & P_5 & P_4 & P_3 & P_2 & P_1 \\
  T_1 & 1 & 0 & 0 & 0 & 0 \\
  T_2 & 1 & 0 & 0 & 0 & 0 & 1 \\
  T_3 & 1 & 0 & 0 & 0 & 1 & 0 \\
  T_4 & 1 & 0 & 0 & 1 & 0 & 1 \\
  T_5 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

The first time the incorrect value of \( P_6 \) affects the picture is at \( T_{21} \). Here \( P_1 \) does not become one-set as is normal.

\[
\begin{array}{cccccc}
  P_6 & P_5 & P_4 & P_3 & P_2 & P_1 \\
  T_{21} & 1 & 1 & 0 & 0 & 0 & 0 \\
  T_{22} & 1 & 0 & 0 & 0 & 0 & 0 \\
  T_{23} & 1 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

Legitimate array for \( T_{22} \)

From this point on, the counter proceeds normally except that it is one count behind the actual bit time (as referenced to \( X_0 \)). At actual \( T_{40} \), the array is:

\[
\begin{array}{cccccc}
  P_6 & P_5 & P_4 & P_3 & P_2 & P_1 \\
  T_{40} & 1 & 1 & 1 & 1 & 0 & 0 \\
\end{array}
\]

This array appears to the logic as \( T_{39} \). The \( T_{40} \) flip-flop then is turned on (\( T_{40} = P_6 P_5 P_3 P_2 'P_1 'C \)).

The array at actual \( T_{41} \) is:

\[
\begin{array}{cccccc}
  P_6 & P_5 & P_4 & P_3 & P_2 & P_1 \\
  T_{41} & 1 & 1 & 1 & 0 & 0 & 0 \\
\end{array}
\]

and at actual \( T_1 \) (\( X_0 \) causes zero-set of \( P_1, P_2, P_4, P_5 \))
Since the \( T_{41} \) reset term becomes true one clock later than usual
\( (1^{T_{41}} = T_{40} \) \( C \)), \( P_6 \) is set at actual \( T_1 \).

\[
\begin{array}{ccccccc}
P_6 & P_5 & P_4 & P_3 & P_2 & P_1 \\
T_1 & 1 & 1 & 0 & 0 & 0 & 0
\end{array}
\]

The counter is still one count behind and will remain so until the
origin \((X_0)\) again comes true (one disk revolution later). At actual \( T_{41} \)
of the 64th word, the array is \( 1 \ 1 \ 1 \ 0 \ 0 \ 0 \). The origin now zero-sets
\( P_1, P_2, P_4 \) and \( P_5 \); giving \( 1 \ 0 \ 0 \ 0 \ 0 \) at actual \( T_1 \). The \( T_{41} \) flip-flop
now is true and at actual \( T_2 \), \( P_6 \) is zero-set giving \( 0 \ 0 \ 0 \ 0 \ 0 \ 1 \). The
counter is now in step. It therefore requires over one disk revolution
to synchronize the counter for Case III.

**Case IV**

\[
\begin{array}{ccccccc}
P_6 & P_5 & P_4 & P_3 & P_2 & P_1 \\
T_1 & 1 & 0 & 0 & 1 & 0 & 0 \\
T_2 & 1 & 0 & 1 & 0 & 0 & 1 \\
T_3 & 1 & 1 & 0 & 0 & 1 & 0 \\
T_4 & 1 & 0 & 0 & 1 & 0 & 0 \\
T_5 & 1 & 0 & 1 & 0 & 0 & 1
\end{array}
\]

Notice that in Case IV the above "illegitimate" arrays form a
closed recycling configuration of three terms \( (1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ ... \ etc.) \). This loop can be broken only
by \( X_0 \) \( C \) at a time when \( 1 \ 0 \ 0 \ 1 \ 0 \ 0 \) is not present. Table 6 illustrates
the process of synchronization:
Table 6. Synchronization Process

<table>
<thead>
<tr>
<th></th>
<th>P_6</th>
<th>P_5</th>
<th>P_4</th>
<th>P_3</th>
<th>P_2</th>
<th>P_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_0</td>
<td>T_1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>T_{41}</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S_1</td>
<td>T_1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>T_{41}</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S_2</td>
<td>T_1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>T_{41}</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\* 41 bits per word = 13-2/3 cycles per 41 bits. Therefore, the second array of the three is found at T_{41}.

\** 64 words per revolution = 21-1/3 cycles per revolution. Therefore, the array 1 0 0 1 0 0 occurs at T_1 S_{63}.

\*** P_1, P_2, P_4, and P_5 are zero-set at T_{41} of S_{63} by X_0.

At T_1 S_0 (after the second X) Case IV reduces to Case III. Since Case III requires over one disk revolution to provide synchronization, the total synchronization time for Case IV is over two disk revolutions.

Special Digit Times

Particular counts of the digit counter are required for numerous control and arithmetic functions in the computer logic. It is economical to create special timing flip-flops in such cases rather than to form lengthy "AND" gates using the P flip-flops. In particular, signals are required at digit times 40, 41, 1 and 2, therefore three flip-flops are provided. Flip-flop T_{40} is set to 1 by the count of 39 and is therefore true during time 40 at which time it resets itself to 0 at T_{41}. 

508-T-6
T₄₀ is turned on by T₄₀, T₁ by T₄₁, and T₂ by T₁. Each flip-flop resets itself to zero one bit-time after being turned on.

T₄₁

\[
\begin{array}{c|c|c}
T_{41} & 1^{t_{41}} = T_{40} C & T_{41a} \\
0^{t_{41}} = T_{41} C \\
\end{array}
\]

T₁

\[
\begin{array}{c|c|c}
T_{1} & 1^{t_{1}} = T_{41} C & 1^{t_{41a}} = T_{40} C \\
0^{t_{1}} = T_{1} C \\
0^{t_{41a}} = T_{41a} C \\
\end{array}
\]

T₂

\[
\begin{array}{c|c|c}
T_{2} & 1^{t_{2}} = T_{1} C & 0^{t_{2}} = T_{2} C \\
\end{array}
\]

Timing flip-flop N₉ is true for a single bit time of each word.

N₉

\[
\begin{array}{c|c|c}
N_{9} & 1^{n_{9}} = G_{0} T_{1} C + G_{0}' P_{6} P_{5} P_{4}' P_{3}' P_{2}' C & 0^{n_{9}} = N_{9} C \\
\end{array}
\]

If the right-hand command is being effected, G₀ is true, N₉ is set at T₁ and becomes true during bit time 2. If the left-hand command is being effected, G₀ is false, N₉ is set at bit time 21 and true during bit time 22. N₉ is used to turn on several time interval flip-flops such as N₁, N₄ and N₆.
Sector Counter

The sector counter consisting of flip-flops $S_1$ through $S_6$ indicates the sector address of the next word to pass under a given read head. Since the information channels on the memory contain 64 word-sectors, the sector counter increases by 1 during each word time from 0 to 63. Thus the sector counter changes by 1 during the same time that the digit counter sweeps through its entire 41-bit count. Initial zero reference is accomplished by gate $X_{OC}$ on the zero input to each $S$ flip-flop.

Associated with the sector counter is a sector address time flip-flop, $N_1$, which is true only while the sector address (bits 3 through 8, or 23 through 28) of the current command is being read (see figure 10). Accordingly, $N_1$ is turned on by $N_9$ and turned off 6 bit times later.

\[
\begin{array}{c|c|c}
N_1 & 1^n & N_9 \\
0^n & P_2 & P_3 \\
\end{array}
\]

During sector address times (bits 3 through 8 and 23 through 28), both $P_2$ and $P_3$ are true only at bit time 8 and at bit time 28. Additional $P$ diodes in these terms are therefore redundant.

Figure 14 illustrates the process by which the sector counter is augmented.

First, the increment "one" to be added to the $S$-counter sets the carry flip-flop $K_s$ at $T_1$ of each word time. Next, cyclic right-shifting of the $S$-counter occurs during the six bit times of $N_1$. During the end-around shift, if $K_s$ is false, the state of $S_1$ is copied in $S_6$; if $K_s$ is true, $S_1$ is reversed in $S_6$. $K_s$ is one-set when a zero appears in $S_1$ and remains in the false state for the duration of the word time. For example, figure 15 shows how the sector counter and related flip-flops behave when the sector count is advanced from 19 to 20. $G_0$ is assumed true. If $G_0$ is false, $N_9$ is true at bit time 22 instead of $T_2$, and $N_1$ is true from 23 to 28 instead of 3 through 8. Although $K_s$ is again set at $T_1$, whatever changes formerly occurred in the sector counter and $K_s$ during times 23 through 29.
| $K_s$ | $1^k_s = T_1 C$
|      | $0^k_s = S_1 'N_1 C$
| $S_1$ | $1^s_1 = S_2 N_1 C$
|      | $0^s_1 = S_2 'N_1 C + X_o C$
| $S_2$ | $1^s_2 = S_3 N_1 C$
|      | $0^s_2 = S_3 'N_1 C + X_o C$
| $S_3$ | $1^s_3 = S_4 N_1 C$
|      | $0^s_3 = S_4 'N_1 C + X_o C$
| $S_4$ | $1^s_4 = S_5 N_1 C$
|      | $0^s_4 = S_5 'N_1 C + X_o C$
| $S_5$ | $1^s_5 = S_6 N_1 C$
|      | $0^s_5 = S_6 'N_1 C + X_o C$
| $S_6$ | $1^s_6 = S_1 K_s 'N_1 C + S_1 'K_s N_1 C$
|      | $0^s_6 = S_1 'K_s 'N_1 C + S_1 K N_1 C + X_o C$

508-T-6
Figure 14. Sector Counter Augmentation
Bit Time

<table>
<thead>
<tr>
<th>Bit Time</th>
<th>$T_1$</th>
<th>$N_9$</th>
<th>$N_1$</th>
<th>$K_s$</th>
<th>$S_6$</th>
<th>$S_5$</th>
<th>$S_4$</th>
<th>$S_3$</th>
<th>$S_2$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>41</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 15. Example of Sector Counting

Such a counter is sometimes called a one-input adder because it performs ordinary serial addition with only one operand, the other being assumed zero and the carry stage initially set to one.

**COMMAND SELECTION** $I_1$

The first step, $I_1$, in the master sequence of normal computation involves selection of a command pair from memory and then gating it into the Z-register. Command selection is identified by flip-flop $I_1$ being one-set. The location counter, flip-flops $G_1$ through $G_{12}$, specifies the memory location of the desired command pair. $G_7$ through $G_{12}$ determine the memory channel and $G_1$ through $G_6$ determine the sector to be read in that channel.
Recirculation of Location Counter

The 12-bit G-counter experiences right-shifting during I₁ from bit times 3 through 14, or 23 through 34 (depending on whether the right-or left-hand command is being executed). N₄ furnishes the necessary timing.

\[
\begin{array}{c|c}
N₄ & 1^{n₄} = I₁N₆C \\
\hline
& 0^{n₄} = P₅P₄P₃C \\
\end{array}
\]

The terms in 0ⁿ₄ are true at bit times 14 and 34 respectively and at no other time during the intervals 3 through 14 and 23 through 34 (see figure 13). Recirculation logic for the G-counter is shown in figure 16.

The expression K₀F₀U₁ in the final G₁₂ logic is always true during command selection, I₁, in the compute mode, I₄. Reasons for this expression will be presented in later sections of this manual.

Channel Selection

The channel address originally contained in G₇ through G₁₂ can be read from G₁ during bit times 9 through 14 or 29 through 34 as the G-counter shifts. A "channel address time" flip-flop N₅ defines the time interval.

\[
\begin{array}{c|c}
N₅ & 1^{n₅} = (N₁P₅)D₁₀D₁₁'R₁I₃'C \\
\hline
& 0^{n₅} = P₅P₄P₃C \\
\end{array}
\]

Examination of figure 13 will show that the terms in parentheses are true at bit times 6 and 26 respectively while the terms in 0ⁿ₅ can turn N₅ off only at 14 and 34 respectively. The expression outside the parentheses in １ⁿ₅ is true during I₁ except for the last word time.

The channel register, flip-flops C₁ through C₆, receives the channel address bits as follows: C₆ copies G₁ during N₅ while the C-register is shifted right.
<table>
<thead>
<tr>
<th>[G_1]</th>
<th>[G_6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1g_1 = G_2 \cdot N_4 C]</td>
<td>[1g_6 = G_7 \cdot N_4 C]</td>
</tr>
<tr>
<td>[0g_1 = G_2 \cdot N_4 C]</td>
<td>[0g_6 = G_7 \cdot N_4 C]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[G_2]</th>
<th>[G_7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1g_2 = G_3 \cdot N_4 C]</td>
<td>[1g_7 = G_8 \cdot N_4 C]</td>
</tr>
<tr>
<td>[0g_2 = G_3 \cdot N_4 C]</td>
<td>[0g_7 = G_8 \cdot N_4 C]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[G_3]</th>
<th>[G_8]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1g_3 = G_4 \cdot N_4 C]</td>
<td>[1g_8 = G_9 \cdot N_4 C]</td>
</tr>
<tr>
<td>[0g_3 = G_4 \cdot N_4 C]</td>
<td>[0g_8 = G_9 \cdot N_4 C]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[G_4]</th>
<th>[G_9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1g_4 = G_5 \cdot N_4 C]</td>
<td>[1g_9 = G_{10} \cdot N_4 C]</td>
</tr>
<tr>
<td>[0g_4 = G_5 \cdot N_4 C]</td>
<td>[0g_9 = G_{10} \cdot N_4 C]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[G_5]</th>
<th>[G_{10}]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1g_5 = G_6 \cdot N_4 C]</td>
<td>[1g_{10} = G_{11} \cdot N_4 C]</td>
</tr>
<tr>
<td>[0g_5 = G_6 \cdot N_4 C]</td>
<td>[0g_{10} = G_{11} \cdot N_4 C]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[G_{11}]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1g_{11} = G_{12} \cdot N_4 C]</td>
</tr>
<tr>
<td>[0g_{11} = G_{12} \cdot N_4 C]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[G_{12}]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1g_{12} = G_1 \cdot N_{4 \cdot 1 \cdot 4} C]</td>
</tr>
<tr>
<td>[0g_{12} = G_1 \cdot N_{4 \cdot 1 \cdot 4} C]</td>
</tr>
</tbody>
</table>

Figure 16. Recirculation Logic for the G Counter
Information from eight of the 64 memory read amplifiers is gated into the memory read power amplifiers $M_{ra}$, $M_{rb}$, $M_{rc}$, $M_{rd}$, $M_{re}$, $M_{rf}$, $M_{rg}$, $M_{rh}$, as determined by the three least significant channel bits in $C_1$ through $C_3$. (Clock pulses are not required here because $M_{ra}$, $b$, $c$, $M_{rd}$, $e$, $f$, $g$, $h$, are amplifiers, not flip-flops. As a result, there is no time delay in this transfer of information.)

$$M_{ra} = (C_3 C_{C_2} C_1) M_{r0} + (C_3 C_{C_2} C_1') M_{r1} + (C_3 C_{C_2} C_1') M_{r2} + (C_3 C_{C_2} C_1') M_{r3} + (C_3 C_{C_2} C_1') M_{r4} + (C_3 C_{C_2} C_1') M_{r5} + (C_3 C_{C_2} C_1') M_{r6} + (C_3 C_{C_2} C_1') M_{r7}$$

$$0^m_{ra} = (C_3 C_{C_2} C_1') M_{r0} + (C_3 C_{C_2} C_1') M_{r1} + (C_3 C_{C_2} C_1') M_{r2} + (C_3 C_{C_2} C_1') M_{r3} + (C_3 C_{C_2} C_1') M_{r4} + (C_3 C_{C_2} C_1') M_{r5} + (C_3 C_{C_2} C_1') M_{r6} + (C_3 C_{C_2} C_1') M_{r7}$$
\[
M_{rb} = (C_3'C_2'C_1')M_{r8} + (C_3'C_2'C_1)M_{r9} + (C_3'C_2'C_1')M_{r10} + (C_3'C_2'C_1)M_{r11} + (C_3'C_2'C_1')M_{r12} + (C_3'C_2'C_1)M_{r13} + (C_3'C_2'C_1')M_{r14} + (C_3'C_2'C_1)M_{r15}
\]

\[
M_{rc} = (C_3'C_2'C_1')M_{r16} + (C_3'C_2'C_1)M_{r17} + (C_3'C_2'C_1')M_{r18} + (C_3'C_2'C_1)M_{r19} + (C_3'C_2'C_1')M_{r20} + (C_3'C_2'C_1)M_{r21} + (C_3'C_2'C_1')M_{r22} + (C_3'C_2'C_1)M_{r23}
\]

\[
M_{rd} = (C_3'C_2'C_1')M_{r24} + (C_3'C_2'C_1)M_{r25} + (C_3'C_2'C_1')M_{r26} + (C_3'C_2'C_1)M_{r27} + (C_3'C_2'C_1')M_{r28} + (C_3'C_2'C_1)M_{r29} + (C_3'C_2'C_1')M_{r30} + (C_3'C_2'C_1)M_{r31}
\]
\[ 0^m_{rd} = (C_3'C_2'C_1')Mr_{24}' + (C_3'C_2'C_1')Mr_{25}' + (C_3'C_2'C_1')Mr_{26}' + \\
(C_3'C_2'C_1')Mr_{27}' + (C_3'C_2'C_1')Mr_{28}' + (C_3'C_2'C_1')Mr_{29}' + \\
(C_3'C_2'C_1')Mr_{30}' + (C_3'C_2'C_1')Mr_{31}' \]

\[ 1^m_{re} = (C_3'C_2'C_1')Mr_{32} + (C_3'C_2'C_1')Mr_{33} + (C_3'C_2'C_1')Mr_{34} + \\
(C_3'C_2'C_1')Mr_{35} + (C_3'C_2'C_1')Mr_{36} + (C_3'C_2'C_1')Mr_{37} + \\
(C_3'C_2'C_1')Mr_{38} + (C_3'C_2'C_1')Mr_{39} \]

\[ 0^m_{re} = (C_3'C_2'C_1')Mr_{32}' + (C_3'C_2'C_1')Mr_{33}' + (C_3'C_2'C_1')Mr_{34}' + \\
(C_3'C_2'C_1')Mr_{35}' + (C_3'C_2'C_1')Mr_{36}' + (C_3'C_2'C_1')Mr_{37}' + \\
(C_3'C_2'C_1')Mr_{38}' + (C_3'C_2'C_1')Mr_{39}' \]

\[ 1^m_{rf} = (C_3'C_2'C_1')Mr_{40} + (C_3'C_2'C_1')Mr_{41} + (C_3'C_2'C_1')Mr_{42} + \\
(C_3'C_2'C_1')Mr_{43} + (C_3'C_2'C_1')Mr_{44} + (C_3'C_2'C_1')Mr_{45} + \\
(C_3'C_2'C_1')Mr_{46} + (C_3'C_2'C_1')Mr_{47} \]

\[ 0^m_{rf} = (C_3'C_2'C_1')Mr_{40}' + (C_3'C_2'C_1')Mr_{41}' + (C_3'C_2'C_1')Mr_{42}' + \\
(C_3'C_2'C_1')Mr_{43}' + (C_3'C_2'C_1')Mr_{44}' + (C_3'C_2'C_1')Mr_{45}' + \\
(C_3'C_2'C_1')Mr_{46}' + (C_3'C_2'C_1')Mr_{47}' \]
Careful study of the memory read power amplifier logic reveals that when the least significant octal channel address digit is \( n \), we may write

| \( M_{rg} \) | \( 1 \) \( m_{rg} = (C_3'C_2'C_1')M_{r48} + (C_3'C_2'C_1')M_{r49} + (C_3'C_2'C_1')M_{r50} + \\ + (C_3'C_2'C_1')M_{r51} + (C_3'C_2'C_1')M_{r52} + (C_3'C_2'C_1')M_{r53} + \\ + (C_3'C_2'C_1')M_{r54} + (C_3'C_2'C_1')M_{r55} \\ | 0 \( m_{rg} = (C_3'C_2'C_1')M_{r48} + (C_3'C_2'C_1')M_{r49} + (C_3'C_2'C_1')M_{r50} + \\ + (C_3'C_2'C_1')M_{r51} + (C_3'C_2'C_1')M_{r52} + (C_3'C_2'C_1')M_{r53} + \\ + (C_3'C_2'C_1')M_{r54} + (C_3'C_2'C_1')M_{r55} ' \\ |
| \( M_{rh} \) | \( 1 \) \( m_{rh} = (C_3'C_2'C_1')M_{r56} + (C_3'C_2'C_1')M_{r57} + (C_3'C_2'C_1')M_{r58} + \\ + (C_3'C_2'C_1')M_{r59} + (C_3'C_2'C_1')M_{r60} + (C_3'C_2'C_1')M_{r61} + \\ + (C_3'C_2'C_1')M_{r62} + (C_3'C_2'C_1')M_{r63} \\ | 0 \( m_{rh} = (C_3'C_2'C_1')M_{r56} + (C_3'C_2'C_1')M_{r57} + (C_3'C_2'C_1')M_{r58} + \\ + (C_3'C_2'C_1')M_{r59} + (C_3'C_2'C_1')M_{r60} + (C_3'C_2'C_1')M_{r61} ' \\ + (C_3'C_2'C_1')M_{r62} + (C_3'C_2'C_1')M_{r63} ' \\ |

| \( M_{ra} \) | \( 1 \) \( m_{ra} = M_{rn} \\ | 0 \( m_{ra} = M_{rn} ' \\ |
| \( M_{rc} \) | \( 1 \) \( m_{rc} = M_{r(n + 16)} \\ | 0 \( m_{rc} = M_{r(n + 16)} ' \\ |
| \( M_{rb} \) | \( 1 \) \( m_{rb} = M_{r(n + 8)} \\ | 0 \( m_{rb} = M_{r(n + 8)} ' \\ |
| \( M_{rd} \) | \( 1 \) \( m_{rd} = M_{r(n + 24)} \\ | 0 \( m_{rd} = M_{r(n + 24)} ' \\ |
\[
\begin{array}{|c|c|}
\hline
M_{re} & 1^m_{re} = M_r(n + 32) \\
& 0^m_{re} = M_r(n + 32) \\
\hline
M_{rf} & 1^m_{rf} = M_r(n + 40) \\
& 0^m_{rf} = M_r(n + 40) \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
M_{rg} & 1^m_{rg} = M_r(n + 48) \\
& 0^m_{rg} = M_r(n + 48) \\
\hline
M_{rh} & 1^m_{rh} = M_r(n + 56) \\
& 0^m_{rh} = M_r(n + 56) \\
\hline
\end{array}
\]

where \(0 < n \leq 7\).

Information from only one of the eight read power amplifiers is copied into the memory flip-flop \(M_r\) according to the states of \(C_6, C_5,\) and \(C_4\):

\[
M_r = C_6'C_5'C_4'M_{r_{a}} + C_6'C_5'C_4'M_{r_{b}} + C_6'C_5'C_4'M_{r_{c}} +
\]
\[
C_6'C_5'C_4'M_{r_{d}} + C_6'C_5'C_4'M_{r_{e}} + C_6'C_5'C_4'M_{r_{f}} +
\]
\[
C_6'C_5'C_4'M_{r_{g}} + C_6'C_5'C_4'M_{r_{h}}
\]

\[
0^m_r = C_6'C_5'C_4'M_{r_{a}} + C_6'C_5'C_4'M_{r_{b}} + C_6'C_5'C_4'M_{r_{c}} +
\]
\[
C_6'C_5'C_4'M_{r_{d}} + C_6'C_5'C_4'M_{r_{e}} + C_6'C_5'C_4'M_{r_{f}} +
\]
\[
C_6'C_5'C_4'M_{r_{g}} + C_6'C_5'C_4'M_{r_{h}}
\]

Thus, for a given binary configuration in \(C_1\) through \(C_6\), \(M_r\) will accept the information read from only one memory channel. For example if the channel address of a command is 45, \(C_6'C_5'C_4'C_3'C_2'C_1\) becomes true and \(M_{r_{5}}', M_{r_{13}}', M_{r_{21}}', M_{r_{29}}', M_{r_{37}}', M_{r_{45}}', M_{r_{53}}', M_{r_{61}}\) are copied into \(M_{r_{a}}', M_{r_{b}}', M_{r_{c}}', M_{r_{d}}', M_{r_{e}}', M_{r_{f}}', M_{r_{g}}', M_{r_{h}}\) respectively. Since \(C_6'C_5'C_4'\) is also true, \(M_{rf}(= M_{r_{45}}')\) is copied into \(M_r\) as required.
Sector Selection

In order to locate the sector identified in $G_1$ through $G_6$, it is necessary to wait for the sector counter $S_1$ through $S_6$ to reach agreement with $G_1$ through $G_6$, respectively. Obviously, such agreement must occur within a disk revolution since this is the duration of a complete sector count (0 through 63). Since the $S$ counter is right-shifted through $S_1$ during $N_1$ (bit times 3 through 8 or 23 through 28) while the $G$-counter is similarly shifted through $G_1$ during $N_4$ (bit times 3 through 14 or 23 through 34), it is clear that $S_1$ and $G_1$ will contain corresponding bits during the 6-bit interval $N_1$. (See figure 17.)

$S_o$ is the memory read selection flip-flop; it is set to zero at every $T_{41}$ pulse. During $N_1$, flip-flops $G_1$ and $S_1$ are sampled for disagreement, recognizable by their having opposite states. Any such disagreement between $G_i$ and $S_i$ ($i = 1, 2, ..., 6$) is used to turn $S_o$ on

\[
S_o = (G_1 S_1' + G_1' S_1) I_1 N_1 C + G_1 S_1' I_2' I_3' N_1 C + G_1' S_1 I_2' I_3' N_1 C +
\]

\[
0 S_o = T_{41} C
\]

Where $I_2' I_3'$ in the final $S_o$ logic includes $I_1$. Thus, if $S_o$ has not been turned on by the end of the word time, $T_{41}$, agreement must have been attained and the desired sector has been located.

Command Gating

The memory read gating flip-flop, $D_o$, is turned on if $S_o$ is true at $T_{41}$. $D_o$ resets itself to zero at the next $T_{41}$. 

508-T-6
Figure 17. Sector Selection During $I_1$
where \( L_{\phi} \) refers to a nonvolatile channel. \( D_0 \) is true for one complete word time (bit times 1 through 41) during which time it gates a command pair from memory into the Z-register.

\[
\begin{array}{c|c}
10_4 & M_D 1 \bar{C} \\
01_4 & M_1 D 1 \bar{C}
\end{array}
\]

where flip-flop \( Z_{41} \) is the transmitting end of the 41-bit command register. Since a word is read from memory serially, starting with the least significant bit, it is copied into \( Z_{41} \) in the same sequence. Thus, \( Z_{41} \) will contain the most significant (sign) bit immediately upon completion of the gating operation, i.e., during \( D_0 \). In order to conserve equipment in the mechanization of the 41-bit Z-register, only the \( Z_1, Z_2 \) and \( Z_{41} \) positions are stored in flip-flops. The 38 intervening Z-bits are stored in a recirculating loop on the memory disk. Right-shifting of the Z-register is accomplished by the physical rotation of the disk past stationary reading and writing heads. (See figure 18.) Information from \( Z_{41} \) is transferred with no clocked delay to the Z-channel write amplifier.

\[
\begin{array}{c|c}
1 w & Z_{41} \\
0 w & Z_{41} \bar{C}
\end{array}
\]

The output of \( Z_w \) activates the Z-channel write head which magnetizes in time sequence the portion of the disk passing under it. A read head located 38-1/2 bits in the direction of rotation from the write head is energized in time sequence by the same locally magnetized areas passing under it, 38-1/2 bit times later. A Z-channel read amplifier, \( Z_r \), improves the read head signal and is clocked into flip-flop \( Z_2 \).

\[
\begin{array}{c|c}
1 2 & Z_r C \\
0 2 & Z_r \bar{C}
\end{array}
\]
Information from $Z_2$ is shifted into $Z_1$

| $z_1$ | $Z_2 \cdot C$ | $0 \cdot z_1 = Z_2 \cdot 'C$ |

When a command pair has been written in the $Z$-register, the function of command selection is completed. Consequently both $I_1$ and the gating flip-flop $D_0$ are turned off simultaneously.

| $i_1 = I_1 \cdot D \cdot T_1 \cdot C$ | $0 \cdot i_1 = I_2 \cdot D \cdot T_1 \cdot C$ |

Until $I_1$ is reestablished, the selected command pair in the $Z$-register is available for sampling and interpreting. This is easily accomplished by continuously rewriting back on the disk the information that has just been read from it.

| $z_{41} = Z_1 \cdot I_1 \cdot 'C$ | $0 \cdot z_{41} = Z_1 \cdot I_1 \cdot 'C$ |

This one-word recirculation loop allows all the bits in the $Z$-register to be observed sequentially (in flip-flop $Z_1$, for example) during each one-word time cycle. In discussing the $Z$-register small letters $z_i$ represent the information bits in the $Z$-channel with subscripts corresponding to the bit positions in a word. Illustrated is the quiescent condition existing between consecutive clock pulses $T_{41}C$ and $T_1C$ (i.e., flip-flop $T_1$ is in the true state). Coincidentally with the clock pulse that set $T_1$ (i.e. $T_{41}C$), the $Z$-channel read head sensed the magnetic pattern of $z_2$ on the disk. At the same time that $z_2$ was read, it was amplified by $Z_r$ and clocked into flip-flop $Z_2$. Simultaneously, the previously sensed $z_1$ bit was clocked from $Z_2$ into the $Z_1$ flip-flop, while the $z_{41}$ bit was clocked from $Z_1$ or $M_r$ into $Z_{41}$ to be written via $Z_w$ onto the disk.

Since the memory read flip-flop $M_r$ receives its pulse train from whatever channel is indicated by the $C$-register, it is imperative that flip-flops $C_1$ through $C_6$ remain static during memory gating. Flip-flop
Figure 18. Operation of Z Register
$N_5$ must be true in order for shifting to occur in the $G$-register. Therefore $N_5$ is inhibited from coming true during memory read gating, $D_0$, as follows.

$$1^n_5 = D_0' \ldots$$

**Command Gating from Loops L and V**

Both 8-word recirculation loops may contain numerical data and/or commands. If a command to be executed is stored on a loop, the $G$-counter will contain its loop address. As always, during $I_1$ the channel address bits are shifted into the $G$-register and the sector address bits are sampled for coincidence with the $S$-counter. However, the interpretation of the loop address differs from that of the information channel address as described below.

---

**Loop Selection**

Loop addresses are octal 7760 through 7767 for the $L$ loop, and octal 7770 through 7777 for the $V$ loop. In binary notation this would be $111111110000-11111110111$ for the $L$ loop, and $111111110000--1111111111111$ for the $V$ loop. Distinction between loop and main memory is made by $L_{p1}$, the loop-memory selection flip-flop.

<table>
<thead>
<tr>
<th>$L_{p1}$</th>
<th>1$^n_{p1} = D_0' \ldots I_{3} R_{13} C_{1} T_{1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0$^n_{p1} = G_1' I_{5} N_5 C$</td>
<td></td>
</tr>
</tbody>
</table>

Notice that $L_{p1}$ is one-set at $T_1$ time of $I_3'$. This would make $L_{p1}$ one-set during $T_1$ time of the first word time of $I_1'$. Also observe the eight most significant bits of loop address are all ones (the loop address being present in the $G$-register). This indicates that the occurrence of a zero in $G_1$ during $N_5$ time (recall that the $G$-register circulates during this time $T_7$ through $T_{14}$ or $T_{27}$ through $T_{34}$) would indicate that a loop was not desired, and the logic would zero-set $L_{p1}$. Since there is no logic to one-set $L_{p1}$ again, $L_{p1}$ would indicate main memory. On the other hand, if $L_{p1}$ were not zero-set a loop address would be indicated, i.e., $L_{p1}$ would remain one-set.
If a loop is selected, it then remains to determine which loop. The L-V selection flip-flop, 

\[
\begin{array}{c|c|c|c|c|c}
1 \ p2 & G \ 1 & N \ 15 & P \ '1 & C \\
1 \ p2 & G \ 1 & N \ 1 & P \ '1 \ C \\
0 \ p2 & G \ 1 & N \ 15 & P \ '1 & C \\
\end{array}
\]

is designed to determine this; \(L_{p2}\) zero-set indicates the L loop and \(L_{p2}\) one-set indicates the V loop. In the binary notation for the loop address that the 4th least significant bit is zero for the L loop and a one for the V loop. This bit is called the loop indicator bit. At \(T_6\) or \(T_{26}\) time this bit appears in \(G_1\) of the \(G\)-register, and is indicated in the logic by the appearance of \(N_1P'5\). If \(G_1\) is one-set at \(T_6\) or \(T_{26}\) time, \(L_{p2}\) is one-set during the remainder of \(N_1\). If \(G_1\) is zero-set at \(T_6\) or \(T_{26}\) time, \(L_{p2}\) is zero-set during the remainder of \(N_1\). \(P_2\)' is included in the one-set logic for \(L_{p2}\) to keep \(L_{p2}\) one-set or zero-set during the remainder of \(N_1\).

Loop Design

A detailed description of the high-speed loops is desirable at this point. Since the two loops are identical, the description applies to either \(L\) or \(V\). For convenience, however, reference will be made to the \(L\) loop.

Loop Recirculation

As seen in figure 19, information on the loop is preserved by recirculation around a path that includes a 325-1/2-bit portion of a memory channel, associated read and write heads and amplifiers, \(L_{r8}\) and \(L_w\), and three flip-flops, \(L_2\), \(L_1\) and \(L_{41}\).

The complete recirculation path stores 328 bits which is exactly equal to eight 41-bit words. This closed path is interrupted (between \(L_1\) and \(L_{41}\)) only when new information is to be written on the loop; it will be fully described in a later section. When a word is to be read from the loop, as during \(I_1\), it is necessary only to wait until it circulates around to a position that is accessible for gating out. With this method a waiting period of from one to eight word times may be required for access to a random word on the loop.
### L-loop

<table>
<thead>
<tr>
<th></th>
<th>1  = L₂C</th>
<th>1  = L₂'C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1  = L₂'C</td>
<td>0  = L₂'C'</td>
</tr>
</tbody>
</table>

**L₂a** is the L-loop 4-word read flip-flop.

<table>
<thead>
<tr>
<th></th>
<th>1  = L₄R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1  = L₄R'</td>
</tr>
</tbody>
</table>

**L₂** is the L-loop 8-word read flip-flop.

<table>
<thead>
<tr>
<th></th>
<th>1  = L₈R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1  = L₈R'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1  = L₄R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1  = L₄R'</td>
</tr>
</tbody>
</table>

### V-loop

**L₂** is the L-loop 8-word read flip-flop.

<table>
<thead>
<tr>
<th></th>
<th>1  = L₄R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1  = L₄R'</td>
</tr>
</tbody>
</table>

**V₂** is the V-loop 8-word read flip-flop.

<table>
<thead>
<tr>
<th></th>
<th>1  = V₄R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1  = V₄R'</td>
</tr>
</tbody>
</table>

**V₂** is the V-loop 8-word read flip-flop.

<table>
<thead>
<tr>
<th></th>
<th>1  = V₄R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1  = V₄R'</td>
</tr>
</tbody>
</table>

**V₂** is the V-loop 8-word read flip-flop.
Supplementary Reading Station

Reduction of access time is achieved by placement of an additional read head within the loop approximately midway between the other heads. With two reading stations four words apart in an 8-word loop, the maximum time before a random word passes under a read head is clearly four word times. Use of supplementary read heads and associated amplifiers $L_{r_4}$ and $V_{r_4}$ thus improves reading access time by a factor of two. Determining from which head and the exact time to read information from the loop is covered in the next section.

Loop Sector Selection

Since each loop contains eight word sectors, a 3-stage binary counter is needed for sector addressing. Happily, the three least significant stages of the $S$-counter, $S_3$, $S_2$, and $S_1$, serve this purpose well. This loop sector counter counts eight word times from binary 000 to 111, resets to 000, and repeats indefinitely. The three most significant sector bits are ignored in loop addresses, e.g., sectors 010011 and 111011 both refer to loop sector 3. If the loop sector count is binary 001 when a given word is being written on the loop, it will be 101 when the same word is passing under the 4-word read head, and it will have recycled back to 001 when the word is passing under the 8-word read head. Since the loop sector counter must agree with the corresponding address bits of a given word when it is written on the loop, the same agreement must therefore exist when the word is read from the 8-word read head. It follows that when the desired word is read from the 4-word read head, the loop sector counter must differ by 4 from the corresponding address bits. Conveniently, decimal 4 is binary 100 which, added or subtracted from a binary address, has no effect on the two least significant bits and reverses the next (see table 7).

Table 7. Sector Counting in 8-Word Loops

<table>
<thead>
<tr>
<th>Write Head</th>
<th>4-Word Read Head</th>
<th>8-Word Read Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>100</td>
<td>000</td>
</tr>
<tr>
<td>001</td>
<td>101</td>
<td>001</td>
</tr>
<tr>
<td>010</td>
<td>110</td>
<td>010</td>
</tr>
<tr>
<td>011</td>
<td>111</td>
<td>011</td>
</tr>
<tr>
<td>100</td>
<td>000</td>
<td>100</td>
</tr>
<tr>
<td>101</td>
<td>001</td>
<td>101</td>
</tr>
<tr>
<td>110</td>
<td>010</td>
<td>110</td>
</tr>
<tr>
<td>111</td>
<td>011</td>
<td>111</td>
</tr>
</tbody>
</table>
Figure 19. Operation of 8-Word Loop During $I_1$
It may be recalled that flip-flop $S_8$ is turned on during $I \wedge \Sigma_1$ whenever the sector and address bits shifted into $S_1$ and $G_1$ disagree. Thus, if $S_8$ is still false after three bits have been sampled (at bit time 6 or 26), then flip-flop $S_8$ is not set, which indicates that the desired word must be gated from the 8-word read head.

$S_8 \\
1^{'S_8} = S \cdot D \cdot 'I \wedge \Sigma_1 \wedge 'P \wedge 'C \\
0^{'S_8} = S \cdot T \cdot 41 \wedge C + I \cdot D \cdot T \cdot 41 \wedge C$

Figure 13 reveals that $N_{1P2}^P$ can occur at either bit time 6 or 26 depending on the state of $G_1$. $S_8$ is reset to zero at $T_{41}$. Also, if $S_8$ is false after only two bits have been sampled (at bit time 5 or 25), flip-flop $S_7$ is not set.

$S_7 \\
1^{'S_7} = S \cdot D \cdot 'I \wedge \Sigma_1 \wedge 'P \wedge 'P \wedge 'C \\
0^{'S_7} = T \cdot 41 \wedge C$

Figure 13 shows $N_{1P4P1}^P$ is true only at bit time 5 or 25. $S_7$ is zero-set at every $T_{41}$. At the end of a word time, if $S_7$ is false and $S_8$ is true, then the S- and G-counters agree in the two least significant positions and disagree in the next, indicating that the desired word must be gated from the 4-word read head. Information from either head is copied into a common flip-flop, $K_0$ (figure 19).

$K_0 \\
1^{'K_0} = L_{2a} \cdot 'S_7 \wedge S_8 \wedge 'C + L_2 \cdot 'S_1 \wedge 'C \\
0^{'K_0} = L_{2a} \cdot 'L_2 \wedge 'S_7 \wedge 'S_8 \wedge 'C + L_2 \cdot 'L_2 \wedge 'S_8 \wedge 'C$

where $L_{p2}^P$ identifies the L-loop. The use of $L_{2a}$ instead of $L_{r8}$ is necessitated by circuit restrictions that prevent power amplifiers from driving more than one gate; also $L_{2a}$ is used instead of $L_{r4}$ for the same reason. If $L_{p2}^P = 1$, the V-loop read heads affect $K_0$ in the same manner.
Command Gating from Loop

For a loop address, \( L_{p1} \) is true, and \( S_7' \) indicates that the desired word will appear under one of the read head during the next word time. Consequently, the loop gating flip-flop \( D_{10} \) is turned on at \( T_{41} \) and remains on for a full word time.

\[
\begin{align*}
0^{d_{10}} & = D_{10}^{T_{41}} C \\
1^{d_{10}} & = S_7'L_{p1}D_{10}^{I_1}T_{41}C = S_7'L_{p1}D_{10}^{I_1}I_3'(D_1' + D_3' + D_4')T_{41}C
\end{align*}
\]

During \( D_{10}I_1 \), the command pair read from the loop into \( K_o \) is gated into the \( Z \)-register as required.

\[
\begin{align*}
0^{z_{41}} & = K_oD_{10}^{I_1}C \\
1^{z_{41}} & = K_o'D_{10}^{I_1}C
\end{align*}
\]

One additional precaution is necessary. During gating it is imperative that \( K_o \) continue to copy from the same head. This requires that flip-flops \( L_{p2}S_7 \) and \( S_8 \) not change state while \( D_{10} \) is true. \( L_{p2} \) is held fixed by \( L-V \) indicator bit in the \( G \)-register which does not change during \( I_1 \).

When reading from the 8-word head, \( S_8 \) must remain off and is therefore inhibited from coming on by \( D_{10} \) in \( I_8 \). When reading from the 4-word head, \( S_7 \) must remain off and is therefore inhibited from coming on by \( D_{10} \) in \( I_7 \), while \( S_8 \) must remain on and is therefore inhibited from going off by \( S_7 \) in \( 8_8 \). Finally, \( D_{10} \) is inserted in \( I_{10} \) to prevent \( D_{10} \) itself from being pulsed on both sides at the conclusion of the gating operation \( D_{10}T_{41} \). Command selection \( I_1 \) is terminated when gating into the \( Z \)-register is completed.

\[
0^{'1_1} = I_1'D_{10}T_{41}aC
\]
COMMAND INTERPRETATION AND OPERAND SELECTION.

Command selection, I₁, is concluded and command interpretation, I₂, is initiated when a command pair has been gated into the Z-register from either an information channel by D₀ or a high-speed loop by D₁₀.

\[
I₂ = \begin{array}{l}
i₂ = \begin{array}{llllllllll}
I₁ & D_0 & T & 4 & 1 & a & C & + & I₁ & 'D_0 & T & 4 & 1 & a & C
\end{array}
\end{array}
\]

The computer's primary function during I₂ is to select from memory the operand necessary for the execution of arithmetic commands. However, for certain nonarithmetic commands not requiring an operand, this function is superfluous. The command's operation code determines what is accomplished during I₂ and later during I₃.

**Operation Code Register**

The operation code bits in a command pair (figure 10) occupy positions 35 through 40 for the first command and positions 15 through 20 for the second command. During I₂ the selected command pair recirculates in the Z-register so that the least significant bit z₁ is in flip-flop Z₁ during bit time 1; the next bit z₂ is in Z₁ during bit time 2, and in general the nᵗʰ bit is in Z₁ during bit time n. Recirculation is accomplished by

\[
\begin{array}{c|c|c}
1 & z₁ = Z₁ & I₂ & C = Z₁ & I₁ & 'C \\
0 & z₁ = Z₁ & I₂ & C = Z₁ & I₁ & 'C \\
\end{array}
\]

The operation code bits are available in flip-flop Z₁ during bit times 15 through 20 or 35 through 40; or one bit time earlier in Z₂ they are available during bit times 14 through 19 or 34 through 39. The latter timing is provided by the operation code time flip-flop N₂.
\[ N_2 = I_{24}'N_5P_4P_2P_1'N \]
\[ 0^N_2 = P_2'P_1'N \]

where \( N_5 \) is true during the same interval (bit times 9 through 14 or 29 through 34) in \( I_2 \) as in \( I_1 \).

\[ 1^N_5 = I_2(...)= I_3'(...) \]

Examination of figure 1.3 shows that the three terms in \( 1^N_2 \) can become true only at bit times 13 or 33, making \( N_2 \) true starting with bit times 14 or 34 respectively. \( P_2'P_1' \) in \( 0^N_2 \) resets \( N_2 \) to zero at bit times 19 or 39. The choice of interval depends on which command is being interpreted. For the left-hand command, \( G_0' = 1 \) and \( N_2 \) is true during 34 through 39; for the right-hand command, \( G_0 = 1 \) and \( N_2 \) is true during 14 through 19. During \( N_2 \) the operation code bits are shifted from \( Z_2 \) into the operation code register, flip-flops \( D_1 \) through \( D_6 \):

\( D_1 \)
\[ 1^D_1 = D_2N_2'N \]
\[ 0^D_1 = D_2'N_2'N \]
\( D_2 \)
\[ 1^D_2 = D_3N_2'N \]
\[ 0^D_2 = D_3'N_2'N \]
\( D_3 \)
\[ 1^D_3 = D_4N_2'N \]
\[ 0^D_3 = D_4'N_2'N \]
\( D_4 \)
\[ 1^D_4 = D_5N_2'N \]
\[ 0^D_4 = D_5'N_2'N \]
\( D_5 \)
\[ 1^D_5 = D_6N_2'N \]
\[ 0^D_5 = D_6'N_2'N \]
\( D_6 \)
\[ 1^D_6 = Z_2N_2'N \]
\[ 0^D_6 = Z_2'N_2'N \]

It should be noted that \( N_2 \) always becomes true during the first word time of \( I_2 \) and gates the operation code into the \( D \)-register in sufficient time for sampling at the first \( I_2T_{40} \) time.
Types I₂

Sampling of the operation code bits stored in the D-register determines the logical sequence to be followed during I₂. For this purpose the computer commands may be categorized into three major types as follows.

No Memory Reference Required (Type 1)

Commands in this category do not require selection of an operand from memory. For all type-1 operations, the state change flip-flop N₁₁ is set at T₄₀ and reset at T₄₁.

\[ N_{11} = \begin{cases} 1^n_{11} = \text{Type-1 operation codes } T_{40} & C \\ 0^n_{11} = T_{41} & C \end{cases} \]

N₁₁ being true during T₄₁ then terminates I₂ at the end of the first word time

\[ 0^{I₂} = I₂ N_{11} C \]

The I₂ diode in this gate prevents the I₂ flip-flop from being pulsed on both sides when it is set at the end of I₁ since N₁₁ could be true at that time.

Referring to the Operation Code Matrix, figure 20, it will be observed that type 1 includes all transfer of control commands (first octal digit = 5), input-output and stop commands (first octal digit = 7), shift and normalize commands (octal codes 40, 41, 45), R-A interchange command (octal code 43) and transfer-to-memory command (octal code 60), and transfer to memory command with X, A interchange (octal code 15). All commands whose first octal code digit is 5 or 7 are covered by the combination D₆D₄. Operation codes 40, 41 and 60 are included in D₆D₃' D₂', operation code 43 and 45 D₆D₃'D₁ and operation code 15 in D₅D₄D₃ D₁' T₄₀'. Substituting these gates in the logical equation for N₁₁ yields

\[ 1^n_{11} = D₆D₄T₄₀C + D₆D₃'D₂'T₄₀C + D₆D₅'D₁T₄₀C + D₅D₃'D₁T₄₀C \]
<table>
<thead>
<tr>
<th>$D_6'D_5'D_4$</th>
<th>$D_6'D_5'D_4$</th>
<th>$D_6'D_5'D_4$</th>
<th>$D_6'D_5'D_4$</th>
<th>$D_6'D_5'D_4$</th>
<th>$D_6'D_5'D_4$</th>
<th>$D_6'D_5'D_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLA 00</td>
<td>ADD 01</td>
<td>CLS 02</td>
<td>SUB 03</td>
<td>FAD 04</td>
<td>FDV 05</td>
<td>FSB 06</td>
</tr>
<tr>
<td>$M_0$</td>
<td>$M_0$</td>
<td>$M_0$</td>
<td>$M_0$</td>
<td>$M_4M_4(M_0)$</td>
<td>$M_4M_4(M_0)$</td>
<td>$M_5M_5(M_0)$</td>
</tr>
<tr>
<td>$M_1$</td>
<td>$M_1$</td>
<td>$M_1$</td>
<td>$M_1$</td>
<td>$M_7$</td>
<td>$M_7$</td>
<td>$M_7$</td>
</tr>
<tr>
<td>$M_2$</td>
<td>$M_2$</td>
<td>$M_2$</td>
<td>$M_2$</td>
<td>$M_2$</td>
<td>$M_2$</td>
<td>$M_2$</td>
</tr>
<tr>
<td>$M_3$</td>
<td>$M_3$</td>
<td>$M_3$</td>
<td>$M_3$</td>
<td>$M_3$</td>
<td>$M_3$</td>
<td>$M_3$</td>
</tr>
<tr>
<td>$M_4$</td>
<td>$M_4$</td>
<td>$M_4$ (M_3)</td>
<td>$M_4$</td>
<td>$M_4$</td>
<td>$M_4$</td>
<td>$M_4$</td>
</tr>
<tr>
<td>$M_5$</td>
<td>$M_5$</td>
<td>$M_5$</td>
<td>$M_5$</td>
<td>$M_5$</td>
<td>$M_5$</td>
<td>$M_5$</td>
</tr>
<tr>
<td>$M_6$</td>
<td>$M_6$</td>
<td>$M_6$</td>
<td>$M_6$</td>
<td>$M_6$</td>
<td>$M_6$</td>
<td>$M_6$</td>
</tr>
<tr>
<td>$M_7$</td>
<td>$M_7$</td>
<td>$M_7$</td>
<td>$M_7$</td>
<td>$M_7$</td>
<td>$M_7$</td>
<td>$M_7$</td>
</tr>
</tbody>
</table>

Figure 20. Operation Code Matrix
Memory Referencing

The address portions of all commands except type 1 refer to memory or loop locations. As during I₁, the C-register receives the channel address bits which select the desired channel for gating, while the S-counter is compared for coincidence with the sector address bits to ascertain the proper gating time. The procedure during I₂ differs from that during I₁ in that the address bits are initially stored in the recirculating Z-register instead of the G-counter. The channel address occupies bit positions 9 through 14 or 29 through 34 and is therefore available in flip-flop Z₁ during the corresponding bit times. As before N₅ provides the necessary timing while information is shifted into the C-register after being copied in C₆.

N₅ provides the necessary timing while information is shifted into the C-register. After being copied in C₆.

\[
\begin{array}{l}
1^c_i = C_i + 1^N_5 C \\
0^c_i = C_i + 1^N_5 C \quad i = 1, 2, 3, 4, 5. \\
1^c_6 = Z_1 I_2 N_5 C \\
0^c_6 = Z_1 I_2 N_5 C
\end{array}
\]

The sector address is available in Z₁ during bit times 3 through 8 or 23 through 28 for comparison with the corresponding S-counter bits. As in I₁, S₎ remains false only when agreement exists during N₁.

\[
\begin{array}{l}
1^s_o = (Z_1 S'_1 + Z_1 S'_1) I_2 N_5 C = Z_1 S'_1 I'_1 4'_1 N_5 C + Z_1 S'_1 I'_1 4'_1 N_5 C \\
0^s_o = T_{41} C
\end{array}
\]

For loop addresses, flip-flops S₇, S₈, and D₁₀ are affected by the same logic as during I₁.
Number Selection Without Gating (Type 2)

For commands in this category, \( I_2 \) is terminated when the addressee has been located but not yet gated out of the memory. This occurs when the sector address of the command agrees with the sector count. As previously mentioned, this condition is indicated by the false state of the applicable sector selection flip-flop (e.g., \( S_o \)) at \( T_{41} \). Gating associated with type 2 commands is performed during \( I_3 \).

Figure 20 shows that type-2 commands include the 8-word transfers to or from the L and V loops (octal codes 64 through 67) and the 2-word transfer-to-memory (octal code 35). Of the five, only the CTL and CTV commands (codes 64 and 66) require reading from memory. \( D_6 D_5 D_1 \) is one of several combinations that covers these codes. Inclusion of type-1 codes in the combination causes no difficulty since such codes terminate \( I_2 \) after the first word time. \( N_{11} \) is set at \( T_{40} \) when the desired sector is ready for gating from memory.

\[
I_{11}^n = S_o' D_6 D_5 D_1' T_{40} C
\]

As with type-1 commands, \( N_{11} \) immediately terminates \( I_2 \) at \( T_{41} \).

The remaining type-2 commands (octal codes 35, 65, 67) require writing on the memory. \( S_{11} \) is the sector selection flip-flop used for memory writing and corresponds to \( S_o \) for memory reading.

The nonvolatile memory channels utilize separate read and write heads spaced 180 deg or 32 sectors apart. Therefore the sector addresses of words passing under the read and write heads differ by 32 (= binary 100000). It follows that the binary addresses agree in all but the most significant bit. Since the S-counter is synchronized to indicate the addresses of words under the read head, words under the write head may be identified by the sector counter with its most significant bit reversed. \( S_{11} \), initially zero, is turned on if the desired sector address in \( G \) or \( Z \) and the S-counter disagree in any of the first five bits or agree in the sixth.
Thus if \( S \) is still false after five bits have been sampled (at bit time 8 or 28) and becomes true after the sixth bit has been sampled (at bit time 9 or 29), then \( S_{11} \) is not set, indicating that the desired sector is the next to appear under the write head. If any of the above conditions is not met, \( S_{11} \) is set, indicating that the desired sector is not about to appear under the write head at this time. \( S_{11} \) is reset to zero every \( T_{41} \).

Returning to the type-2 commands that require memory writing, clearly \( S_{11}' \) and an appropriate D-register combination are sufficient to terminate \( I_2 \). Because this group of commands includes codes 35, 65, and 67, figure 20 reveals \( D_{6} D_{1} \) and \( D_{4} D_{3} D_{1} \) as the configurations. Thus

\[
1^{n}_{11} = S_{11}' D_{4} D_{3} D_{1} T_{40} C + S_{11}' D_{6} D_{1} T_{40} C
\]

One additional possibility must be considered. The 2-word transfer-to-memory command, FST (octal code 35), could contain a loop address. Since \( S_{8} \) is the sector selection flip-flop associated with loop writing and \( D_{4} D_{3} D_{1} \) covers octal code 35, we may write

\[
1^{n}_{11} = S_{8} L_{p1} D_{4} D_{3} D_{1} T_{40} C
\]

where \( L_{p1} \) indicates a loop address.

**Number Selection with Gating (Type 3)**

This category includes all commands not covered by types 1 and 2. Here the operand is located and gated out of memory during \( I_2 \). As during \( I_1 \), the applicable sector selection flip-flop (\( S_{9} \) or \( S_{7} \)) indicates by its false state at \( T_{41} \) the correct time to turn on the memory or loop gating flip-flop (\( D_{0} \) or \( D_{10} \)).
Instead of being gated into the Z-register as during I_1, the addressee is gated into the B-register during I_2. This is a one-word recirculating register consisting of four flip-flops (B_1, B_2, B_40, B_41) and a portion of the disk storing the remaining 38 bits between the B read and write heads. Thus the B-register differs from the Z-register only in storing one more bit in a flip-flop and therefore one less bit on the disk. Information from a memory channel or loop is gated into B_41 as follows:

\[
\begin{align*}
B_{41}^{b} &= M_r D_0 I_2 C + K_0 D_{10} I_2 C \\
B_{41}^{0} &= M_r' D_0 I_2 C + K_0' D_{10} I_2 C
\end{align*}
\]

B_{40} copies B_{41}.

\[
\begin{align*}
B_{40}^{b} &= B_{41} I_2 C = B_{41} M_r' M_0' C \\
B_{40}^{0} &= B_{41} I_2 C = B_{41} M_r M_0 C
\end{align*}
\]

where M_0' in the final logic is always true in I_2. B_{40} is connected directly to the B-channel write amplifier, B_w.

\[
\begin{align*}
B_w^{b} &= B_{40} \\
B_w^{0} &= B_{40}^{'}
\end{align*}
\]

At the reading end of the B-channel, the read power amplifier B_r feeds into B_2 which in turn shifts into B_1.

\[
\begin{align*}
B_2^{b} &= B_r C \\
B_2^{0} &= B_r' C
\end{align*}
\]
Having gated the desired number into the B-register, D₀ or D₁₀ is reset.

$$0^{d}_0 = D_0 T_{41} C$$
$$0^{d}_{10} = D_{10} T_{41} C$$

At the same time I₂ is terminated by N₁₁.

$$1^{n}_{11} = D_0 T_{40} C + D_{10} T_{40} C$$
$$0^{i}_{2} = I_2 N_{11} C$$

**COMMAND EXECUTION, I₃**

During command execution, I₃, the arithmetic and logical operations specified by the commands are actually carried out. I₃ is initiated at the same time I₂ is terminated.

$$1^{i}_{3} = J_0 'I' N_{11} C$$

where J₀, the overflow flip-flop, is normally false.

**Functional Groups**

The computer's commands separate roughly into eight functional groups that are identified by flip-flops Mᵢ (i = 0, 1, ..., 7) where i is the most significant octal digit of the operation code (see figure 20). Three additional flip-flops, M₈-₁₀, specify floating point operations. The D-register, having been filled with the operation code bits during I₂, is used to set the appropriate M flip-flop simultaneously with the start of I₃.
Most of the commands require a fixed number of word times for their execution during \( I_3 \). It is necessary therefore to keep count of these word times in order to terminate the operations at the proper time. A computation cycle counter consists of flip-flops \( Q_1 \) through \( Q_6 \).
and an associated carry flip-flop, $K_q$, to form a one-input adder which counts word times in the same manner as the sector counter. The $Q$-counter counts only during $I_3$.

$$Q_1 \begin{cases} 1q_1 = Q_2N_1M_8 + M_8X_4T_1'C \\ 0q_1 = Q_2N_1M_8 + I_5T_4I + M_8T_1C \end{cases}$$

$$Q_2 \begin{cases} 1q_2 = Q_3N_1M_8 + Q_1T_4M_8C + X_41M_8U_1'T_1C \\ 0q_2 = Q_3N_1M_8 + I_5T_4I \end{cases}$$

$$Q_3 \begin{cases} 1q_3 = Q_4N_1M_8 + M_8L_1'C_2'T_40C \\ 0q_3 = Q_4N_1M_8 + U_1T_4I + M_10T_41C \end{cases}$$

$$Q_4 \begin{cases} 1q_4 = Q_5N_1M_8 + M_8M_0'I_5'T_41C \\ 0q_4 = Q_5N_1M_8 + M_8Q_5'I_2T_41C + M_8U_1'C_1'T_41C \end{cases}$$

$$Q_5 \begin{cases} 1q_5 = Q_6N_1M_8 + M_8D_2'R_2'R_2'C + M_8D_2'R_2'R_2'Q_5'T_40C \\ 0q_5 = Q_6N_1M_8 + M_8X_2'R_2'R_2'C + M_8U_2'R_2'Q_5'T_40C \end{cases}$$

Counter Operation During $M_8'M_0'M_4'M_10' + M_8'M_0'M_9'$ Commands

Inspection of the logic equations for $Q_1$ through $Q_5$ indicates that the usual one-input adder shift logic is present except for $M_8$-type commands. $M_8$ commands are FAD, FSB, and FSQ. These commands use some of the $Q$ flip-flops for various tests required by the floating-point mechanism. The addition of $M_8$ to gates for the logic of $Q_1$ through $Q_5$ will be discussed later with the floating commands.
Inspection of the logic for $Q_6$ reveals that this flip-flop has the usual one-input adder recirculation logic (similar to the logic for $S_6$ of the sector counter; plus the additional terms $P_m C + I_3 'C$ in the zero-set logic. The $P_m C$ term is involved in the tape punch output mode, therefore we will neglect its presence for compute operation. $I_3 'C$ determines that $Q_6$ through $Q_1$ will be in their zero-set states by the start of $I_3$. The $I_3$ terms in the one-set logic determine how long the counter will count. This, in turn, depends on the word times required by the command in $I_3$.

The logic for the carry flip-flop $K_q$ indicates that the carry will one-set at $T_1$ (similar to $K_s$) for $M_0 'M_4 'M_{10} ' + M_0 'M_9 '$. These gates are to exclude operation of the counter during the portions of the floating-point instructions that require the $Q$ flip-flops for testing. The zero-set logic for $K_q$ is the usual one-input adder logic.

$U_{41}$, $U_2$, and $U_1$ Flip-Flops (Neglecting FDV, FMP, FAD, FSB, FSQ)

At the beginning of $I_3$ the $U_{41}$, and $U_2$, flip flops are in their zero-set state by the logic:

\[
\begin{align*}
0 U_{41} &= U_{41} \cdot T_{41} \cdot C \\
0 U_2 &= U_2 \cdot T_{41} \cdot C
\end{align*}
\]

The $U_1$ flip-flop is in its one-set state by the beginning of $I_3$ by the logic:

\[
U_1 = I_3 'C + M_0 U_2 \cdot T_{41} \cdot C
\]

Table 8 gives the counting sequence.
### Table 8. Counting Sequence

<table>
<thead>
<tr>
<th>TIME</th>
<th>LOGIC INVOLVED</th>
<th>U₄₁</th>
<th>U₂</th>
<th>U₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁I₃ Word 1</td>
<td>0U₄₁ = U₄₁T₄₁aC</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>U₂ = U₂T₄₁C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1U¹ = I₃'C +</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₁I₃ Word 2</td>
<td>1U₂ = A₄'I₃U₄₁T₄₁I₅'N₁₂'C</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0U¹ = I₃'T₄₁U₁'I₅'C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₁I₃ Word 3</td>
<td>0U₂ = U₂T₄₁C</td>
<td>0</td>
<td>0</td>
<td>0,1</td>
</tr>
<tr>
<td></td>
<td>1U¹ = M₀U₂T₄₁aC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₁I₃ Word 41</td>
<td>U₄₁ = M₂D'I₆Q₄Q₄T₄₁aC</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>For DSL and DIV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₁I₃ Word 42</td>
<td>U₄₁ = M₂D'I₆Q₄Q₄T₄₁aC</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>For DSR, DVR, SQR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₂I₃'</td>
<td>0U₄₁ = U₄₁T₄₁aC</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1U¹ = I₃'C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₂I₃'</td>
<td>U¹ = I₃'C +</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inasmuch as frequent reference must be made to the first and second word times in I₃, special timing flip-flops, designated U₁ and U₂, help to simplify the logic associated with such times. U₁ is true during I₃' and the first word time in I₃, while U₂ is true only during the second word time in I₃. U₄₁ is true during the last word time which may be 41 or 42 in I₃ for certain type-3 commands.

\[ Q₆ \]
\[ 1q₆ = Q₁K 'N₁,I₄ P 'C + Q₁K 'N₁,I₄ P 'C \]
\[ 0q₆ = Q₁K 'N₁C + Q₁K 'N₁C + P₈ C + M₈ C + I₃'C \]

\[ K₉ \]
\[ 1q₉ = T₉M₀'M₄'Mₐ'T₉M₀'M₉'C \]
\[ 0q₉ = Q₁K 'N₁C \]
The A- and R-registers are available for use by the programer and, together with the B-register, hold operands and intermediate and final results of all arithmetic operations performed by RECOMP. Like the B- and Z-registers, they are one-word recirculating loops which store part of their information in flip-flops and the rest on the disk. The A-register has six flip-flops, A₁, A₂, A₃, A₃⁹, A₄₀ and A₄¹, plus a 35-bit portion of the disk, while the R-register has four flip-flops, R₁, R₂, R₄₀ and R₄¹, plus a 37-bit portion of the disk. Results produced by execution of a given command must be preserved until needed as operands for the next computation. This is accomplished if the A- and R-registers exhibit normal recirculation during I₃.<br><br>In the A-register, A₁ copies A₂ except at T₄¹ when A₁ is set to zero. During T₄¹, A₂ contains the synchronizing bit which must be zero for certain operations.
where the E terms can be true only when filling the computer. $A_2$ in turn copies $A_3$.

<table>
<thead>
<tr>
<th>$a_2$</th>
<th>$A_3$</th>
<th>$E$</th>
<th>$E$</th>
<th>$E$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_3$</td>
<td>$I$</td>
<td>$E$</td>
<td>$20$</td>
<td>$40$</td>
</tr>
<tr>
<td></td>
<td>$A_3$</td>
<td>$I$</td>
<td>$E$</td>
<td>$3$</td>
<td>$40$</td>
</tr>
<tr>
<td>$1_{a_2}$</td>
<td>$1_{1_3}I_{3}E_{20}E_{40}E_{40}C$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0_{a_2}$</td>
<td>$A_{3}I_{3}E_{20}E_{40}E_{40}C$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$A_3$ receives its information from the A-channel read power amplifier, $A_r$.

<table>
<thead>
<tr>
<th>$a_3$</th>
<th>$A$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_r$</td>
<td></td>
</tr>
<tr>
<td>$1_{a_3}$</td>
<td>$I_{3}A_{r}C$</td>
<td></td>
</tr>
<tr>
<td>$0_{a_3}$</td>
<td>$A_{r}C$</td>
<td></td>
</tr>
</tbody>
</table>

On the other side of the A-channel, the write amplifier, $A_w$, is directly connected to $A_{39}$.

<table>
<thead>
<tr>
<th>$a_w$</th>
<th>$A_{39}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_{39}$</td>
</tr>
<tr>
<td>$1_{a_w}$</td>
<td>$I_{3}A_{39}$</td>
</tr>
<tr>
<td>$0_{a_w}$</td>
<td>$A_{39}$</td>
</tr>
</tbody>
</table>

which copies $A_{40}$

<table>
<thead>
<tr>
<th>$a_{39}$</th>
<th>$A$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_{40}$</td>
<td></td>
</tr>
<tr>
<td>$1_{a_{39}}$</td>
<td>$U_{1}T_{1}M_{1}C$</td>
<td></td>
</tr>
<tr>
<td>$0_{a_{39}}$</td>
<td>$A_{40}U_{1}T_{1}M_{1}C$</td>
<td></td>
</tr>
</tbody>
</table>

which in turn copies $A_{41}$

<table>
<thead>
<tr>
<th>$a_{40}$</th>
<th>$A$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_{41}$</td>
<td></td>
</tr>
<tr>
<td>$1_{a_{40}}$</td>
<td>$I_{3}A_{41}M_{0}U_{41}C$</td>
<td></td>
</tr>
<tr>
<td>$0_{a_{40}}$</td>
<td>$A_{41}M_{0}U_{41}C$</td>
<td></td>
</tr>
</tbody>
</table>

Finally $A_{41}$ copies $A_1$, closing the loop.
$A_{41}$

\[
\begin{array}{c|c}
1_{41}^a & A_1^I 1_3^P 0^C \\
0_{41}^a & A_1^I 1_3^P 0^C \\
\end{array}
\]

$P_0$ is true only during filling.

During recirculation in the R-register, $R_1$ copies $R_2$ which receives information from the R-channel read power amplifier, $R_r$.

$R_1$

\[
\begin{array}{c|c}
1_{1}^r & R_2 1_3^P 0^C \\
0_{1}^r & R_2 1_3^P 0^C \\
\end{array}
\]

$R_2$

\[
\begin{array}{c|c}
1_{2}^r & R^r_C \\
0_{2}^r & R^r_C \\
\end{array}
\]

The R-channel write amplifier, $R_w$, receives its signal from $R_{40}$

$R_w$

\[
\begin{array}{c|c}
1_{w}^r & R_{40} \\
0_{w}^r & R_{40}' \\
\end{array}
\]

which copies $R_{41}$.

$R_{40}$

\[
\begin{array}{c|c}
1_{40}^r & R_{41} E 30_{41}^U 'C \\
0_{40}^r & R_{41} E 30_{41}^U 'C \\
\end{array}
\]

$R_{41}$ copying $R_1$ completes the recirculation.

$R_{41}$

\[
\begin{array}{c|c}
1_{41}^r & R_{1}^I 1_3^P 0^C \\
0_{41}^r & R_{1}^I 1_3^P 0^C \\
\end{array}
\]
Clear-Add (CLA) and Clear-Subtract (CLS) Commands

The command CLA causes the word in memory location m to be transferred to the A-register. The command CLS differs only in that the sign of the word in m is reversed in the transfer to the A-register. The combination $M_0 D_1$ is true for both commands, with $D_2^i$ or $D_2$ added to specify the CLA or CLS commands respectively.

Since these commands belong to $I_2$ type-3, the word in memory location m is stored in the B-register during $I_2$. It suffices therefore during $I_3$ to shift the B-register contents into the A-register, taking required action on the algebraic sign. This procedure is illustrated for the CLA-CLS commands by figure 21. The B-register is shifted right and the sign bit in $B_{41}$ is duplicated in $B_{40}$ for CLA command and reversed in $B_{40}$ for CLS command.

<table>
<thead>
<tr>
<th>For CLA</th>
<th>For CLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_{40} = B_{41} D_3 D_2 D_1 C$</td>
<td>$l_{40} = B_{41} M_0 D_2 D_1 C$</td>
</tr>
<tr>
<td>$0_{40} = B_{41} D_3 D_2 D_1 C$</td>
<td>$0_{40} = B_{41} M_0 D_2 D_1 C$</td>
</tr>
</tbody>
</table>

The rest of the B-register follows the normal right shift.

<table>
<thead>
<tr>
<th>B_w</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_w = B_{40}$</td>
</tr>
<tr>
<td>$0_w = B_{40}$</td>
</tr>
</tbody>
</table>
Everything from the B-register including the sign is copied from \(B_1\) into \(A_{41}\) and on through the A-register.
Figure 21. Operation of CLA-CLS Commands
B_{40} receives the sign bit stored in B_{41} while the entire A-register contents are shifted-right one more bit. For a CLA command, B_{41} is duplicated in B_{40}.

\[
\begin{array}{ll}
A_{41} & A_w \\
1^a_{41} = B_{41}^1 M_{0} D_{1} 'C & 1^a_w = A_{39} \\
0^a_{41} = B_{1} 'M_{0} D_{1} 'C & 0^a_w = A_{39}'
\end{array}
\]

\[
\begin{array}{ll}
A_{40} & A_3 \\
1^a_{40} = A_{41}^1 M_{0} U_{1} 'C & 1^a_3 = A_r 'C \\
0^a_{40} = A_{41} 'M_{0} U_{1} 'C & 0^a_3 = A_r 'C
\end{array}
\]

\[
\begin{array}{ll}
A_{39} & A_1 \\
1^a_{39} = A_{40}^1 (U_{1} + T_{1} + M_{1}^1) 'C & 1^a_1 = A_{2} '1 M_{2} 'M_{10} 'T_{41} 'C \\
0^a_{39} = A_{40} ' (U_{1} + T_{1} + M_{1}^1) 'C & 0^a_1 = A_{2} '1 M_{2} 'M_{10} 'C
\end{array}
\]

whereas for a CLS command, B_{41} is reversed in B_{40}.

\[
\begin{array}{ll}
A_{41} & A_1 \\
1^b_{40} = B_{41}^1 (CLA command) C = B_{41} D_{3} 'D_{2} 'D_{1} 'C & 1^a_1 = A_{2} '1 M_{2} 'M_{10} 'T_{41} 'C \\
0^b_{40} = B_{41} ' (CLA command) C = B_{41} 'D_{3} 'D_{2} 'D_{1} 'C & 0^a_1 = A_{2} '1 M_{2} 'M_{10} 'C
\end{array}
\]

The R-register recirculates the information. Because R_1 is used as a gate, the recirculation takes place from R_2 to R_{42}.
The R-register contents at the end of U₁ will be identical with that of the R-register one word earlier. Also, to force a zero sync bit in R₁,

$$0^r_1 = M_0 \cdot T_{40} \cdot C$$

The functions of the CLA and CLS commands are completed in a single word time, therefore the state change flip-flop N₁₂ is set at T₄₀ for one bit time and terminates I₃ and M₀ at T₄₁.

$$N_{12}$$

| $$1^n_{12} = M_0 \cdot D_{1} \cdot T_{40} \cdot C$$ |
| $$0^n_{12} = T_{41} \cdot a$$ |
| $$0^i_3 = N_{12} \cdot C$$ |
| $$0^m_0 = N_{12} \cdot C$$ |
It will be observed that $N_{12}$ is used to terminate $I_3$ in the same manner that $N_{11}$ was used to terminate $I_2$.

**Add (ADD) and Subtract (SUB) Commands**

The command ADD causes the word in memory location $m$ to be algebraically added to the word in the A-register. The command SUB causes the word in $m$ to be algebraically subtracted from the word in the A-register. The R-register is unaffected by either command. $M_0 D_1$ identifies this pair of commands, with $D_2$ being false for ADD and true for SUB (see figure 20). Being type-3 commands, the word in $m$ is gated into the B-register during $I_2$. It is therefore necessary, during $I_3$, for the B-register contents to be added to or subtracted from the A-register contents and the result stored back in A.

The question of whether to perform absolute addition or subtraction depends not only upon the nature of the command but upon the algebraic signs of the operands as well. Addition of absolute values is required for ADD commands if the operands have the same sign and for SUB commands if the operands have opposite signs. Subtraction of absolute values is indicated for SUB commands if the operands have the same sign and for ADD commands if the operands have opposite signs. Flip-flop $E_o$ indicates whether absolute addition or subtraction is required: $E_o$ indicates addition; $E_o'$ indicates subtraction. The signs of the operands are in flip-flops $A_{41}$ and $B_{41}$ at $T_1$.

$$
\begin{array}{c|c}
E_o & N_{5C} \\
\hline
0_e_o & A_{41} B_{41} M_0 D_2 'U_1 T_1 C + A_{41} 'B_{41} M_0 D_2 'U_1 T_1 C + A_{41} B_{41} M_0 D_2 C + A_{41} 'B_{41} M_0 D_2 C \\
1_e_o & \text{Add} \\
\end{array}
$$

$$
\begin{array}{c|c}
\hline
\text{Sub} & \\
\end{array}
$$
Serial Binary Adder

It is well at this point to consider the characteristics of serial binary addition and subtraction of absolute values.

Serial addition means simply adding one digit position at a time starting from the least significant end and working toward the most significant end, each time recording the sum digit and temporarily storing the carry digit for use at the next digit time. It may be noted that this is the usual pencil-and-paper method of addition.

Assume the addend and augend are initially stored in the A- and B-registers, and during addition these registers are shifted-right so that flip-flops \( A_1 \) and \( B_1 \) contain corresponding bits at any given time. The bits in \( A_1 \), \( B_1 \), and the carry from the previous bit time in flip-flop \( K_a \), determine the sum bit, \( S \), and new carry bit. Figure 22 lists the eight possibilities that may occur.

<table>
<thead>
<tr>
<th>Inputs, Time ( t )</th>
<th>Outputs, Time ( (t+1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_1 )</td>
<td>( B_1 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 22. Truth Table for Binary Adder

Inspection of figure 22 yields the following logic for the sum flip-flop, \( S \),

\[
\begin{align*}
1S &= A_1'B_1'K_a'C + A_1'B_1K_a'C + A_1B_1'K_a'C + A_1B_1K_a'C \\
0S &= A_1'B_1'K_a'C + A_1'B_1K_a'C + A_1B_1'K_a'C + A_1B_1K_a'C
\end{align*}
\]

Note in figure 22 that the carry flip-flop, \( K_a \), changes from 0 to 1 only when \( A_1 \) and \( B_1 \) are both true, and from 1 to 0 only when \( A_1 \) and \( B_1 \) are both false, or
For other combinations of $A_1$ and $B_1$, $K_a$ remains in the same state. $K_a$ must always be zero when the addition is started.

If the number in the B-register is to be subtracted from the number in the A-register, the truth table in figure 23 applies. Note that the "difference" column is identical in all cases with the $S$ column in the adder, which means that the $S$ logic can also be used to generate the difference when performing subtraction. $K_a$, which now contains the "borrow" bit, changes from 0 to 1 only when $A_1'B_1$ is true and from 1 to 0 only when $A_1B_1$ is true. Thus the $K_a$ logic may be expanded to cover both absolute addition and subtraction.

\[
\begin{array}{c|c|c|c|c|c|}
\text{Inputs, time } t & A_1 & B_1 & K_a & \text{Difference} & K_a \\
\hline
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Figure 23. Truth Table for Subtractor

\[
K_a = E_0A_1B_1'C + E_0'A_1'B_1'C
\]

(Carry borrow logic)

Suppose we desire a more versatile logic system capable of addition and subtraction involving either the A- and B-registers or the A- and R-registers. This is exactly the requirement in RECOMP where the operands, normally in A and B, occur in A and R when converting decimal to binary during fill. Instead of duplicating equipment with two
adders, the one described above may be modified to perform both jobs by utilizing \( R_1 \) and \( B_1 \) as control flip-flops to gate the B- and R-registers respectively into the adder. It is necessary merely to replace \( B_1 \) by the "AND" combination \( B_1 R_1 \) in the logic for \( S \) and \( K_a \) above. \( B_1 \) then becomes \( (B_1 R_1)' = B_1' + R_1' \) (by DeMorgan's Theorem). The final logic then becomes

\[
\begin{align*}
S & \quad \text{(adder-subtractor logic)} \\
1_s & = A_1'B_1'K'C + A_1'R_1'K'C + A_1'B_1'R_1'K'C + A_1'B_1'R_1'K'C \\
& \quad + A_1'B_1'K'C + A_1'R_1'K'C + A_1'B_1'R_1'K'C \\
0_s & = A_1'B_1'K'C + A_1'R_1'K'C + A_1'B_1'R_1'K'C + \\
& \quad + A_1'B_1'K'C + A_1'R_1'K'C + A_1'B_1'R_1'K'C
\end{align*}
\]

\[
\begin{align*}
K_a & \quad \text{where } T_{41'} \text{ in the } k \text{ logic and } T_{41'}' \text{ in the } k_a \text{ terms ensure an initial zero carry. Also, } M_{4}' \text{ is obviously true for add-subtract commands, and } T_{41}' \text{ in the } k_a \text{ term is needed for divide commands and causes no difficulty here since only the sync (zero) bits are sampled at } T_{41}. \text{ If we set } R_1 = 1 \text{ in this logic it reduces to the simple } A + B \text{ adder previously considered. Similarly } B_1 = 1 \text{ yields the logic for an } A + R \text{ adder. It may be noted further that if } R_1 = 0, \text{ the sum logic becomes}
\end{align*}
\]

\[
\begin{align*}
1_s & = A_1'B_1'K'C + A_1K'C + A_1'B_1'K'C + A_1'K'C \\
& \quad + A_1'B_1'K'C + A_1K'C + A_1'B_1'K'C + A_1'K'C \\
0_s & = A_1'B_1'K'C + A_1K'C + A_1'B_1'K'C + A_1'K'C
\end{align*}
\]
which may be further reduced to

\[
\begin{align*}
1s &= A_1K_1'C + A_1'K_aC \\
0s &= A_1'K_1'C + A_1K_aC
\end{align*}
\]

If \( K_a = 0 \) also, \( S \) will copy \( A_1 \). Thus, depending on whether \( R_1 = 1 \) or \( 0 \), \( S \) will generate the sum \( A + B \) or copy \( A \), a fact that will prove extremely useful in multiplication.

**First Computation Cycle, \( U_1 \)**

Having mastered the fundamental serial adder-subtractor, we are now in a position to consider the detailed logic associated with the ADD and SUB commands. At the start of \( I_3 \), the A- and B-registers contain the operands in their normal positions, \( E_0 = 1 \) indicating absolute addition, and \( K_a = 0 \).

\[
\begin{align*}
e_0 &= N_5C \\
0 &= T_41C
\end{align*}
\]

Figure 24 illustrates the events that transpire during \( U_1 \) for ADD commands. The diagram can also apply to SUB commands by interchanging the \( B_41 \) output lines. The first clock pulse, \( U_1T_1C \), sets \( R_1 = 1 \) for gating into the adder, sets \( A_{41} = 0 \) for clearing the A-register to zero, and sets \( E_0 = 0 \) if absolute subtraction is required.

\[
\begin{align*}
r_1 &= M_T1'C \\
a_{41} &= M_U1'T1C
\end{align*}
\]

Operands in both the A- and B-registers are shifted-right during \( U_1 \).
Figure 24. Operation of ADD Command During $U_1$
R₁ being true during bit times 2, 3, ..., 40, the significant operand bits a₂, a₃, ..., a₄₀, and b₂, b₃, ..., b₄₀, appear successively in flip-flops A₁ and B₁ respectively for gating into the serial adder-subtractor.

The sum or difference is generated serially in flip-flop S and then re-written back in B₄₀:

\[
\begin{align*}
A_{40} & = A_{41}M\bar{U}C \\
0a_{40} & = A_{41}^\prime M_0U_1C \\
A_{39} & = A_{40}^\prime M_1'C \\
0a_{39} & = A_{40}^\prime M_1'C \\
A_w & = A_{39} \\
0a_w & = A_{39}^\prime \\
A_3 & = A_{r}C \\
0a_3 & = A_{r}^\prime C \\
A_2 & = A_{3}^\prime M_0C = A_{3}M_2'C \\
0a_2 & = A_{3}M_0C = A_{3}^\prime M_2'C \\
A_1 & = A_{2}M_0C = A_{2}^\prime M_0'C \\
0a_1 & = A_{2}M_0C = A_{2}^\prime M_0'C \\
B_w & = 0b_w = B_{40} \\
B_2 & = 0b_2 = B_{r}C \\
B_1 & = 0b_1 = B_{2}M_0C = B_{2}M_{10}'N_7'E_{30}'C \\
1b_1 & = B_{2}M_0C = B_{2}M_{10}'N_7'E_{20}'E_{30}'C \\
1b_{40} & = SM_0D_1C \\
0b_{40} & = SM_0D_1C \\
\end{align*}
\]

The sum digits are now shifted through the B-register by the logic for B₃, B₂ and B₁ above. Note that the sum is written in B₄₀ rather than...
because the S flip-flop in the adder introduces a delay of one bit time. This delay must be compensated for by shortening the B-register by one flip-flop, B_{41}. (S may be considered as replacing B_{41} in the 41-bit B-register).

The zero written in flip-flop A_{41} at T_1 is propagated down through the A-register by the regular shift logic above, and the A-register is cleared since R_1 is being used for gating. The contents of the R-register are preserved by recirculation from R_2 to R_{42} instead of the normal R_1 to R_{41}.

\[
\begin{array}{c|c}
R_{42} & r_{42} = R_2 'M_0 C \\
& 0 r_{42} = R_2 'M_0 C \\
\end{array}
\]

\[
\begin{array}{c|c}
R_{41} & r_{41} = R_2 M_0 Q_3 'C \\
& 0 r_{41} = R_2 'M_0 Q_3 'C \\
\end{array}
\]

\[
\begin{array}{c|c}
R_{40} & r_{40} = R_4 M_0 C \ll R_1 E_3 0 41 'U 'C \\
& 0 r_{40} = R_4 'M_0 C \ll R_1 'E_3 0 41 'U 'C \\
\end{array}
\]

\[
\begin{array}{c|c}
R_w & r_w = R_{40} \\
& 0 r_w = R_{40} \\
\end{array}
\]
The most significant non-sign bits, \(a_{40}\) and \(b_{40}\), are gated into the adder-subtractor at \(T_{40}C\), and \(R_1\) is turned off at this time.

At \(T_{41}\), \(R_1 = 0\) reduces the sum logic to

\[
\begin{align*}
1^S &= (A_1K_a' + A_1'K_a') T_{41}C \\
0^S &= (A_1'K_a' + A_1K_a') T_{41}C
\end{align*}
\]

at which time \(A_1\) contains the sign of \(A\), and \(K_a\) the carry or borrow bit resulting from the addition or subtraction of \(a_{40}\) and \(b_{40}\).

Summarizing the situation at the end of the first computation cycle, \(U_1\), the A-register has been cleared to zero, the B-register contains \(|A| \pm |B|\) and the R-register is unchanged. Interpretation of the new B-register contents in terms of \(S\), \(K_a\) and \(E_0\) occurs in the second computation cycle, \(U_2\).

Second Computation Cycle, \(U_2\)

Depending upon the relative signs and magnitudes of the original operands, four distinct cases will be considered during \(U_2\). These are summarized in figure 25 and described in detail below. Figure 26 illustrates the general flow of information during \(U_2\).

**Case I**: \(A\) and \(B\) have the same sign (i.e., \(|A| + |B|\) is performed during \(U_1\)) and \(|A| + |B| < 1\).

Since RECOMP considers the binary point between the sign and most significant bit, \(|A| + |B| < 1\) means that there is no carry bit left over from the addition of \(a_{40}\) and \(b_{40}\) so that \(K_a = 0\) at \(U_1T_{41}\) at which time \(S\) copies the sign of \(A\) from flip-flop \(A_1\). Obviously in Case I, both \(A\) and the sum \(A + B\) have the same sign which, if positive (binary 1), sets flip-flop \(A_{41}\) at the next bit time.
1^a_{41} = S M_0 J_0^U_2 T_1 C

where \( J_0 = 0 \) for Case I. If the sign is negative (binary 0), no logic is needed since \( A_{41} \) was zeroed in \( U_1 \).

During \( U_2 \) the zero contents of the A-register and the sum \( |A| + |B| \) in the B-register are added, since \( E_0 \) remains true, and the resulting sum, again \( |A| + |B| \), is written back in the A-register.

\[
\begin{align*}
1^a_{40} &= S M_0 U_2 C \\
0^a_{40} &= S'M_0 U_2 C
\end{align*}
\]

Writing occurs in \( A_{40} \) instead of \( A_{41} \) to compensate for the one-bit delay in the adder. Logic for shifting the A- and B-registers and recirculating the R-register is the same during \( U_2 \) as \( U_1 \). \( R_1 \) is also set for gating B into the adder as in \( U_1 \).

**Case II:** A and B have the same sign (i.e., \( |A| + |B| \) is performed during \( U_1 \)) and \( |A| + |B| \geq 1 \).

In this case, \( |A| + |B| \geq 1 \) means that a "one" carry bit is left in \( K_a \) at \( U_1 T_{41} \) from the addition of the most significant bits \( a_{40} \) and \( b_{40} \). The overflow flip-flop, \( J_0 \), is set and activates the neon overflow light on the computer's control panel whenever such an addition overflow occurs.

\[
J_0^I = E_0 K M_0 U_1 T_{41} D_1 C
\]

The overflow carry bit is the integer part of \( |A| + |B| \) and has the numerical value \( \pm 1 \), while the quantity remaining in the B-register at the end of \( U_1 \) is the fractional part \( |A| + |B| -1 \). With \( E_0 \) remaining on during \( U_2 \), this B-register content is added to the A-register's zero contents and the sum is written back in A.

Since \( K_a = 1 \) and \( R_1 = 0 \) at \( U_1 T_{41} \), the sign of A is reversed in flip-flop S. But the sign of the result should agree with the sign of A in this case. Therefore, at the next bit time, \( U_2 T_1 \), another sign reversal occurs in the transfer from S to \( A_{41} \).

\[
1^a_{41} = S'M_0 J_0^U_2 T_1 C
\]

where \( J_0 \) identifies Case II. Again, no logic is needed for the minus sign since \( A_{41} \) was already zero.
Case III: A and B have opposite signs (i.e., \(|A| - |B|\) is performed during \(U_1\)) and \(|A| \geq |B|\).

During \(U_1\) the smaller absolute value was subtracted from the larger leaving no overborrow in \(K_a\) at \(U_1T_{41}\). Hence, there is no sign reversal in \(S\) nor does \(J_0\) get turned on, and the Case I term

\[ l^{a41} = SMO'J_0'U_2T_1C \]

copies the sign of A as required for Case III.

During \(U_1\) the correct result, \(|A| - |B|\), was produced in the B-register while the A-register was cleared to zero. Thus the adder must be set for addition in \(U_2\),

\[ l^{e0} = K_a'M_0T_{41}aC \]

From the above it is clear that Case III requires identically the same logic as Case I during \(U_2\).

Case IV: A and B have opposite signs (i.e., \(|A| - |B|\) is performed during \(U_1\)) and \(|A| < |B|\).

During \(U_1\) the larger absolute value was subtracted from the smaller leaving an overborrow in \(K_a\) at \(U_1T_{41}\). Thus the contents of the B-register are actually the result of subtracting \(|B|\) from \(1 + |A|\), or \(A - B + 1\) as indicated in figure 25. The A-register is cleared to zero during \(U_1\) as usual. If we repeat absolute subtraction during \(U_2\), where again the larger absolute value is in the B-register \((A - B + 1 = 0)\), an overborrow recurs. As seen during \(U_1\), this is equivalent to subtracting the B-register contents from the A-register contents augmented by 1. Thus the \(U_2\) result, written back in the A-register, becomes \((1 + 0) - (A - B + 1) = B - A\) as desired. With \(K_a = 1\) at \(U_1T_{41}\), \(E_0\) remains false during \(U_2\), thereby assuring subtraction.

For Case IV, clearly the sign of the original B-register contents should prevail. At \(U_1T_{41}\) the sign of A was reversed in \(S\) because of \(K_a = 1\). Also, \(J_0\) is not set since \(E_0 = 0\) for Case IV. Therefore at \(U_2T_1\) the reversed sign of A is copied into \(A_{41}\) by

\[ l^{a41} = SMO'J_0'U_2T_1C \]

Since A and B originally had opposite signs, this reversed sign of A is the correct sign of B exactly as required.
<table>
<thead>
<tr>
<th>Case</th>
<th>Signs of A &amp; B</th>
<th>Relative Magnitudes</th>
<th>During U₁</th>
<th>During U₂</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A-</td>
<td>B-</td>
<td>S</td>
</tr>
<tr>
<td>I</td>
<td>Same</td>
<td>A + B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>Same</td>
<td>A + B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>III</td>
<td>Opposite</td>
<td>A - B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IV</td>
<td>Opposite</td>
<td>A - B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 25. Summary of ADD and SUB Command Cases
Figure 26. Operation of ADD and SUB Commands During $U_2$
CASE I: Example of ADD Command with Absolute Addition Required

(ADD) \( +13 \)
(ADD) \( +1.101 \)
\( +2 \)
\( +0.010 \)
\( +15 \)
\( +1.111 \)

<table>
<thead>
<tr>
<th>( E_0 )</th>
<th>( K_a )</th>
<th>( S )</th>
<th>( A )</th>
<th>( B )</th>
<th>( R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 )</td>
<td>1</td>
<td>0</td>
<td>( X )</td>
<td>11010</td>
<td>0</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0+1101</td>
<td>+X0010</td>
</tr>
<tr>
<td>( U_1 )</td>
<td>( T_3 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00110</td>
</tr>
<tr>
<td>( T_4 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>000+11</td>
<td>+10X0</td>
</tr>
<tr>
<td>( T_5 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00000+1</td>
<td>+110X0</td>
</tr>
<tr>
<td>( T_6 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00000+</td>
<td>+1110X0</td>
</tr>
<tr>
<td>( T_1 )</td>
<td>1</td>
<td>0</td>
<td>+</td>
<td>0000000 +</td>
<td>11110</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>+0000000+</td>
<td>1111</td>
</tr>
<tr>
<td>( U_2 )</td>
<td>( T_3 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+0000</td>
</tr>
<tr>
<td>( T_4 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+10+00</td>
<td>+10+11</td>
</tr>
<tr>
<td>( T_5 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+110+0</td>
<td>+110+1</td>
</tr>
<tr>
<td>( T_6 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+1110+</td>
<td>+1110+</td>
</tr>
<tr>
<td>( T_1 )</td>
<td>1</td>
<td>0</td>
<td>( X )</td>
<td>+11110</td>
<td>+11110</td>
</tr>
</tbody>
</table>

Notes:
- \( R_6 \) through \( R_2 \) irrelevant
- \( X \) -- Irrelevant bit
- \( 0 \) -- Means flip-flop zero-set by logic term
- \( B_{41} \) Remains static in \( I_3 \)
- Other terms in \( A \) & \( B \) are standard recirculating logic.

Important Logic:

\[ 1^{e}0 = N_5 C \]
\[ 1^f1 = M_0 T_1 C \]
\[ 0^f1 = M_0 T_{40} C \]
\[ 1^{a41} = SM_0 J_0 'U_2 T_1 C \]
\[ 0^{a41} = M_0 U_1 T_1 C \]
\[ 1^{b41} = SM_0 D_1 C \]
\[ 0^{b41} = S'M_0 D_1 C \]

\[ 1^{a40} = SM_0 U_2 C \]
\[ 0^{a40} = S'M_0 U_2 C \]
\[ 0^{ka} = T_{41} C \]

\[ 1^s = (A_{1a} K_a ' + A_{1a} K_a ) T_{41} C \]
\[ 0^s = (A_{1a} K_a ' + A_{1a} K_a ) T_{41} C \]

508-T-6 119
CASE III: Example of ADD Command with Absolute Subtraction Required

\[
\begin{array}{c}
(ADD) +13 \\
-2 \\
+11
\end{array} 
\begin{array}{c}
(ADD) +.1101 \\
-.0010 \\
+.1011
\end{array}
\]

<table>
<thead>
<tr>
<th>Eo</th>
<th>Ka</th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>654321</td>
<td>654321</td>
<td>1</td>
</tr>
<tr>
<td>T₁</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>+11010</td>
<td>-00100</td>
</tr>
<tr>
<td>T₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>_         +1101</td>
<td>-X0010</td>
</tr>
<tr>
<td>T₃</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>00110</td>
<td>-00010</td>
</tr>
<tr>
<td>T₄</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>00000     +11000</td>
<td>-10100</td>
</tr>
<tr>
<td>T₅</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>_         +0000     +10110</td>
<td>1</td>
</tr>
<tr>
<td>T₆</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>00000     +01100</td>
<td>-01100</td>
</tr>
<tr>
<td>T₁</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>000000</td>
<td>-10110</td>
</tr>
<tr>
<td>T₂</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>_         +0000     +10110</td>
<td>1</td>
</tr>
<tr>
<td>T₃</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+0000</td>
<td>-010110</td>
</tr>
<tr>
<td>T₄</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+1000</td>
<td>-10101</td>
</tr>
<tr>
<td>T₅</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>+1100</td>
<td>-11011</td>
</tr>
<tr>
<td>T₆</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+0110+</td>
<td>-01100</td>
</tr>
</tbody>
</table>

Notes:
- \(R₆\) - \(R₂\) irrelevant
- X - Irrelevant bit
- 0 - Means flip-flop zero-set by logic term
- B₄₁ remains static in \(I₃\)
- Other terms in \(A\) & \(B\) are standard re-circulating logic

Important Logic:

\[
\begin{array}{ccc}
1^e_0 &=& N_5C + K_a M_0T_{41a} C \\
0^e_0 &=& A_{41}B_{41}M_0D_2U_{1T_{1}}C \\
1^r_1 &=& M_0 T_{1}C \\
0^r_1 &=& M_0T_{0}C \\
1^a_{41} &=& SM_1U_{1T_{1}C} \\
0^a_{41} &=& M_0U_{1}T_{1}C
\end{array}
\begin{array}{ccc}
1^{b}_{40} &=& SM_0D_{1}C \\
0^{b}_{40} &=& S'M_0D_{1}C \\
1^{a}_{40} &=& SM_0U_{2}C \\
0^{a}_{40} &=& S'M_0U_{2}C \\
1^s &=& (A_{1}K_a' + A_{1}K_a)T_{41}C \\
0^s &=& (A_{1}K_a' + A_{1}K_a)T_{41}C
\end{array}
\]
CASE IV: Example of SUB Command with Absolute Subtraction Required

<table>
<thead>
<tr>
<th>(Subtract)</th>
<th></th>
<th>(Subtract)</th>
<th>+.0010</th>
<th>+.0011</th>
<th>-.0001</th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>+2</td>
<td>+3</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$E_o$</th>
<th>$K_a$</th>
<th>$S$</th>
<th>$A$</th>
<th>$B$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>+00100</td>
<td>+00110</td>
</tr>
<tr>
<td>T₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+00100</td>
<td>+X0011</td>
</tr>
<tr>
<td>T₃</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>00010</td>
<td>+0X001</td>
</tr>
</tbody>
</table>

| $U₁$ | T₄ | 0 | 1 | 1 | 00000 | +10X00 | 1 |
|------|----|---|---|---|--------|--------|
| T₅   | 1   | 1 | 1 | 00000 | +110X0 | 1 |
| T₆   | 1   | 1 | 1 | 00000 | +1110X | 0 |

| $U₂$ | T₃ | 0 | 1 | 1 | -00000 | +00110 | 1 |
|------|----|---|---|---|--------|--------|
| T₄   | 0   | 1 | 0 | -10000 | +10011 | 1 |
| T₅   | 0   | 1 | 0 | -01000 | +01001 | 1 |
| T₆   | 0   | 1 | 0 | -00100 | +00100 | 0 |

T₁ 0 0 X -00010 +00100 0

Notes:
- $R₆$-$R₂$ irrelevant
- X--irrelevant bits
- 0--means flip-flop
- zero-set by logic
- $B₄₁$ remains static
- Other terms in A & B are standard re-circulating logic

Important Logic:

| $1^o$₀ = N₅ C | $1^b$₀ = SM₀ D₁ C |
| $0^o$₀ = A₄₁ B₄₁ M₀ D₂ C | $0^b$₀ = S'M₀ D₁ C |
| $1^T₁$ = M₀ T₁ C | $1^a$₀ = SM₀ U₂ C |
| $0^T₁$ = M₀ T₄₀ C | $0^a$₀ = S'M₀ U₂ C |
| $1^a$₄₁ = SM₀ J₄¹ U₄₁ T₄₁ C | $1^s$ = (A₄¹ K₄¹ + A₄¹ K₄¹) T₄₁ C |
| $0^a$₄₁ = M₀ U₁ T₁ C | $0^s$ = (A₄¹ K₄¹ + A₄¹ K₄¹) T₄₁ C |
| $0^k$ₐ = T₄₁ C |

508-T-6 121
The ADD and SUB commands having been completed at the end of U₂, N₁₂ is set which terminates I₃.

\[ 1^{n}12 = M_0 M_8 U₂ T_{40} C \]
\[ 0^{i}3 = N₁₂ C \]

**Multiply Unrounded (MPY) and Multiply Round (MPR) Commands**

The command MPY causes the word in memory location m to be multiplied by the word in the A-register to obtain a double-length (78-bit) product whose most- and least-significant halves are stored in the A-register and R-register, respectively. MPR performs the same operation and rounds-off the product to the portion in the A-register, i.e., 39 binary places. Flip-flop M₁ identifies the multiply commands (figure 20) with D₂' and D₂ indicating the MPY and MPR commands respectively.

Several things may be noted from the preceding example. A 1 or 0 multiplier bit times the multiplicand yields the multiplicand itself or a string of 0's respectively. This means that a partial product is formed by either adding or not adding the shifted multiplicand to the previous partial product. In RECOMP this addition is performed by the same adder that is used for the ADD and SUB commands. The option of adding or not adding the multiplicand, stored in the B-register, to the partial product, stored in the A-register, is provided by flip-flop R₁ which gates B₁ into the adder. R₁ must therefore contain the applicable multiplier bit. This is accomplished by initially placing the multiplier in the R-register and subsequently shifting it one bit to the right before performing each addition.

Note also that the length of the partial product starts at one word and increases by one or two bits for each multiplier bit until it becomes two words long as the final product. This appears to indicate that another arithmetic register is required. Fortunately, once a multiplier bit has performed its function, it is no longer needed and may therefore be sloughed off to make room for an additional partial product bit. Thus, the next least significant partial product bit is right-shifted out of A and into R each time a multiplier bit is discarded from R until the final double-length product occupies the entire A- and R-registers as indicated in figure 27.

Furthermore, shifting the A-register one bit to the right is equivalent to shifting the B-register one bit to the left when adding A and B. The first method has the advantage of being a natural outcome.
of the process mentioned in the preceding paragraph and at the same time simplifies the B-register logic to that of normal recirculation.

**Serial Binary Multiplier**

Multiplication of binary numbers follows the same pattern as multiplication of decimal numbers as demonstrated in the examples of figure 27. Starting at the least significant end, digits of the multiplicand are successively multiplied by the least significant digit of the multiplier and the result recorded. The process is repeated using each digit of the multiplier in turn, with the result at each step being shifted-left corresponding to the position of the multiplier digit. When all multiplier digits have been exhausted, the results are added together to obtain the final product.

Ignoring signs for the moment, consider the operation in terms of the computer. The addition process that yields the final product would appear to require an adder having 39 inputs, not to mention temporary storage for 39 words. This problem is overcome by performing the addition as the multiplication proceeds, obtaining a new partial product after each multiplier bit is used up. In figure 27 for example the least significant multiplier bit (1) times the multiplicand (=1010) yields the first partial product (=1010). The second least significant multiplier bit (0) times the multiplicand (=1010) yields the quantity 0000 which is shifted one bit left and added to the first partial product to obtain the second partial product (=01010). In general the nth least significant multiplier bit times the multiplicand is shifted and added to the (n-1)th partial product to obtain the nth partial product. When finally n becomes equal to the number of bits in the multiplier, the nth partial product is the final product and the multiplication process is completed. In the example shown, the multiplier (=1101) has four bits; hence the fourth partial product is the final product.

**First Computation Cycle, U1**

Multiplication being I2 type-3 commands, the multiplicand was read from memory and stored in the B-register during I2, which is exactly where it is required for the operation. Before starting the multiplication process, the multiplier must be transferred from the A- to the R-register, and the A-register must be cleared to zero before accumulating any partial products. These preliminary operations are performed during the first computation cycle, U1, as illustrated in figure 28. The A-register is cleared by setting A41 to zero at T1 and shifting right. Simultaneously the A-register contents are transferred serially to the R-register by having R41 copy A2 while the R-register is likewise shifted right.
Figure 27. Multiplication Example
Figure 28. Multiplication During $U_1$
Note that A1 was by-passed in the preceding process. Consequently, at the end of U1 the entire multiplier will be displaced one bit to the right in R with respect to its original orientation in A, i.e., the least significant bit a2 will be in R1, a3 in R2; ... , the sign bit a41 in R40, and zero in R41. Obviously the original contents of R are destroyed during U1.

The multiplicand in the B-register is preserved by recirculation.
The multiplication example of figure 27 is detailed in figure 29 to show exactly what takes place in the computer. For convenience, this 6-bit example has been chosen for illustrative purposes in lieu of a standard 41-bit RECOMP number. As a result, T6 in the example corresponds to T41 in RECOMP, etc. During U1, note that zero is set into the leading A-register bit, A6, at T1 (thereby making A6 = 0 during T2) and is shifted down through A. In the mean-time the multiplier bits are shifted from A2 to R6 and down through R while the original R contents, indicated by X's, are dumped off the right end of R. The multiplicand is recirculated by B6 copying B1 and shifting.

Several other functions are performed during U1. The algebraic sign of the product is determined and appropriately stored. The signs of the multiplier and multiplicand, initially in A41 and B41 respectively, are sampled in A_{A1} and B1 at T41. If the signs are the same, a plus sign is indicated and R_{43} is turned on.

\[
\begin{align*}
1^{r}_{43} &= A_1 B_1 I_{13} U_{1T} T_4 I_{41a} M_C + A_1 B_1 I_{3U} T_x I_{41a} M_C (M_4 T_{41a} = M_{41})
\end{align*}
\]

If the signs are opposite, a minus sign is indicated. No logic is necessary for R_{43} since it is already off due to

\[
0^{r}_{43} = T_{40C}
\]

At T1 in the second word time (U2) A_{41} is turned on (for a plus sign) by R_{43} true.

\[
1^{a}_{41} = R_{43} M_{1U} T_{I_{1C}}
\]

For a minus sign, A_{41} was turned off at T1 in U1 for clearing the A-register and just remains off.

\[
0^{a}_{41} = M_{1D_3} T_{I_{1C}}
\]

At first observation, it would appear that A_{41} is hit on both sides at U2T1 time. However, this is prevented by turning D3 on at T41 of U1.

\[
1^{d}_{43} = M_{1T_{41}C}
\]

Thus, D3 is on for the remainder of multiplication.
**Figure 29. RECOMP Multiplication Process (MPY)**
Rounding the final product to 39 bits is accomplished by adding binary "1" in the next bit position to the right, which happens to be the most significant bit of the first partial product. The example of figure 27 illustrates this fact wherein the most significant bit of the first partial product appears in the same vertical column as the most significant bit of the less significant half of the final double-length product. Clearly, the first partial product will be properly augmented for round-off if the multiplication process starts (at $U_2 T_1$) with a "1" in $A_{40}$. Thus

$$1^a_{40} = M_1 D_2 U_{41} T_{41} C$$

where $D_2$ identifies the MPR command. For the MPY command, $A_{40}$ remains zero as copied from $A_{41}$, the previous bit time.

**Subsequent Computation Cycles, $U_1$**

Having completed the necessary preparations in $U_1$, we are now ready for the multiplication process which is shown in figure 30. Each of the 39 non-sign multiplier bits calls for a serial addition of the multiplicand into the previous partial product. Since each such addition requires one word time, the entire multiplication, including $U_1$, requires 40 word times.

Because signs are handled separately in $U_1$, one word time per addition suffices since addition of absolute values (Case I) is forced by

$$1^c_{0} = N_5 C$$

At the start of $U_2$ the least significant (non-sync) multiplier bit is in $R_1$ and it must remain in $R_1$ during the entire word time in order to gate $B_1$ into the adder. The rest of $R$ is preserved by recirculation from $R_2$ to $R_{41}$ except at $T_{40}$ and $T_{41}$.

$$R_{41} 1^r_{41} = R_{41} M_{41} B_{41} U_{41} T_{40} T_{41} C$$
$$0^r_{41} = R_2 M_{41} B_{41} U_{41} T_{40} T_{41} C$$
$$R_{40} 1^r_{40} = R_{40} M_{40} C = R_{40} E_{30} U_{41} C$$
$$0^r_{40} = R_{41} M_{41} C = R_{41} E_{30} U_{41} C$$
Figure 30. Multiplication During $U_1'$
At \( T_{40} \), \( R_{41} \) receives the next least significant partial product bit from the A-register.

\[
\begin{align*}
1^r_{41} &= A_3 M_1 B_{42}^r T_{40}^C \\
0^r_{41} &= A_3^r M_1 B_{42} T_{40}^C
\end{align*}
\]

The reader may convince himself that \( A_3 \) does contain the proper bit by referring to the example of figure 29. During any \( U_1^r T_1 \), the least significant A-register bit of the previous partial product’s in \( A_2 \). Proceeding in time sequence, the same bit is shifted into \( A_1 \) at \( T_1^C \), and is gated along with \( B_1 \) into \( S \) at \( T_2^C \), copied into \( A_4 \) at \( T_3^C \), shifted into \( A_3 \) at \( T_4^C \), and finally copied into \( R_6 \) at \( T_5^C \). \( A_4 \) in the example corresponds to \( A_{39} \) in RECOMP, so that the subject bit is copied into \( A_{39} \) at \( T_3^C \), shifted into \( A_{38} \) at \( T_4^C \), \( A_{37} \) at \( T_5^C \), ..., \( A_4 \) at \( T_{38}^C \), \( A_3 \) at \( T_{39}^C \), and copied into \( R_{41} \) at \( T_{40}^C \) as indicated in the logic.

At \( T_{41} \), \( R_{41} \) picks up the sign of the product.

\[
\begin{align*}
R_{41}^1 &= A_{41}^r M_1 B_{42}^r T_{41a}^C \\
0^r_{41} &= A_{41}^r M_1 B_{42} T_{41a}^C
\end{align*}
\]

The next least significant multiplier bit, originating in \( R_2 \) during \( U_1^r T_1 \), is copied into \( R_{41} \) and shifted down through \( R \) to return to flip-flop \( R_2 \) during \( T_{41} \). At \( T_{41} \) it is shifted into \( R_1 \) in preparation for gating the multiplicand into the adder during the next word time.

\[
\begin{align*}
R_{1}^1 &= R_2 M_1 T_{41a}^C \\
0^r_{1} &= R_2^r M_1 T_{41a}^C
\end{align*}
\]

Each new multiplier bit in \( R_1 \) is used to gate the entire multiplicand in \( B \) into the adder. Consequently the B-register exhibits normal recirculation during each word time in \( U_1^r \) by means of the same logic as during \( U_1 \).

Logic for the sum and carry flip-flops is identically that used for addition of absolute values, where \( E_0 = 1 \). Flip-flop \( S \) generates a new partial product during each \( U_1^r \) word time, which is the sum of the previous partial product and either the multiplicand or zero. This sum is rewritten back in \( A \) and at the same time shifted one bit to the right as required. The shifting is accomplished by having \( A_{39}^r \), rather than \( A_{40}^r \), copy \( S \), thereby shortening the A-register by one bit.
At \( T_1 \), \( S \) contains irrelevant information. However, the most significant bit of the new partial product is in \( A_{40} \) at \( T_1 \) and is therefore prefixed to the partial product by means of
\[
\begin{aligned}
1^{a}_{39} &= A_{40}\ T_1\ 'C \\
0^{a}_{39} &= A_{40}\ 'T_1\ 'C
\end{aligned}
\]

This left-most partial product bit was generated at \( T_{40} \) of the previous word time as the carry bit resulting from the addition of the most significant bits in \( A \) and \( B \) and was copied into \( A_{40} \) at \( T_{41} \) by means of
\[
\begin{aligned}
1^{a}_{40} &= K\ M_1\ U_1\ 'C \\
0^{a}_{40} &= K\ 'M_1\ U_1\ 'C
\end{aligned}
\]

Here \( T_{41} \) need not be specified since \( A_{40} \) is not used at \( U_1'T_{41}' \). Each new partial product thus formed is shifted down through the \( A \)-register including flip-flop \( A_1 \) and reentered into the adder. Thus \( A_3 \) reads the disk and in turn is copied into \( A_2 \) and \( A_1 \).
\[
\begin{aligned}
1^{a}_{2} &= A_{3}\ M_1\ C = A_{3}\ I_3\ M_2\ 'C \\
0^{a}_{2} &= A_{3}\ 'M_1\ C = A_{3}\ 'I_3\ M_2\ 'C
\end{aligned}
\]
\[
\begin{aligned}
1^{a}_{1} &= A_{2}\ M_1\ C = A_{2}\ I_3\ M_2\ 'T_{41}\ 'C \\
0^{a}_{1} &= A_{2}\ 'M_1\ C = A_{2}\ 'I_3\ M_2\ 'M_{10}\ 'C + M_1\ 'T_{41}\ 'C
\end{aligned}
\]

At the end of 39 word times in \( U_1 \) (corresponding to the 39 multiplier bits), the 39th partial product is the final product thus completing the multiplication process. For MPY commands the double-length product is in the \( A \)- and \( R \)-registers and for MPR commands the rounded product is in the \( A \)-register. Both registers contain the sign of the product. Including \( U_1 \), a total of 40 word times elapse during \( I_3 \) so that the \( C \)-counter finally contains a count of 40 which is used to turn on \( N_{12} \) which terminates \( I_3 \).
\[
\begin{aligned}
1^{n}_{12} &= M_1\ M_9\ 'Q_6\ 'Q_4\ 'T_{40}\ 'C \\
0^{i}_{3} &= N_{12}\ 'C
\end{aligned}
\]
Division (DSL, DIV, DSR, DVR) Commands

Four separate division commands may be executed in RECOMP. DSL causes division of a dividend A in the A-register by a divisor B in memory location m to produce a quotient Q in the A-register and a remainder R in the R-register. Thus,

\[
\frac{A}{B} = Q + 2^{-39} \frac{R}{B} \quad \text{where } |R| < |B| \text{ and } A \text{ and } R \text{ have the same sign. Those conditions uniquely determine } Q. \text{ Retaining the sign of the dividend } A \text{ in the remainder } R \text{ facilitates programming of multiple precision division. If the binary point is assumed at the left of the most significant bit, then } |A| \text{ and } |B| \text{ are both < } 1, \text{ and if also } |A| < |B|, \text{ then } |Q| < 1.

If a double-length (78-bit) dividend is available whose more and less significant halves are in the A- and R-registers respectively, then the command DIV causes division as described above.

The commands DSR and DVR perform the same operations as DSL and DIV respectively and also include final rounding of the quotient to 39 binary places.

One may think of the binary points as being at the right of a 78-bit dividend and a 39-bit divisor. Then the equation above becomes

\[
\frac{2^{78} A}{2^{39} B} = 2^{39} Q + \frac{R}{B}
\]

and, the binary point is now considered at the right of the quotient.

Figure 20 shows flip-flop \( M_2 \) true for all divide commands. Further, \( D_2 \) and \( D_2' \) specify division with double and single-length dividends respectively, while \( D_1 \) and \( D_1' \) distinguish between the rounded and unrounded divide commands respectively.

Serial Binary Divider

Division in binary, as in decimal, may be performed by successive subtraction of the divisor, properly positioned, from the full and reduced dividends.
In the decimal example of figure 31, the two-digit divisor, 11, is first subtracted from the two most significant dividend bits, 14, in order to obtain the most significant (tens) digit of the quotient. Clearly, this quotient digit, 1, is the number of such subtractions possible before an overborrow occurs, as shown in the example. When an overborrow occurs, the reduced dividend, 037, is restored by adding the divisor, 11, back in again. Next, the divisor is shifted one digit right and the same process is repeated to obtain the next most significant (units) digit, 3. This being the least significant digit, the entire quotient, 13, has now been found and the final reduced dividend, 004, is the remainder. It should be noted that nine subtractions are the maximum required to obtain a given quotient digit.

The same operation is performed in binary as illustrated in figure 28. Quotient bits obtained in descending order of significance are 0 or 1 according to whether subtraction of the divisor did or did not cause an overborrow. If an overborrow occurs, restoration of the reduced dividend is again achieved by adding back the divisor. Note that only a single subtraction is required to determine a given quotient bit in binary as against nine in decimal. It appears that this method requires as many subtractions as there are quotient bits and as many additions as there are zero quotient bits.

An improvement of the previous method yields the so-called Von Neumann division that is used in RECOMP and illustrated in figure 28. In the previous method an overborrow requires addition to restore the reduced dividend, followed by subtraction of the right-shifted divisor. Clearly, this two-step operation is equivalent to a single addition of the shifted divisor, (i.e., \( +D - \frac{1}{2}D = +\frac{1}{2}D \)). In the Von Neumann process, this technique is used to obtain each quotient bit as a result of a single addition or subtraction. Depending upon the operation performed and the existence of an overborrow or overcarry, four distinct cases must be considered. They are summarized in the table below. Verification of the tabulated results is left as an exercise for the reader.

<table>
<thead>
<tr>
<th>Current operation is</th>
<th>and caused</th>
<th>Quotient bit is</th>
<th>and next operation is</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtraction</td>
<td>No overborrow</td>
<td>1</td>
<td>Subtraction</td>
</tr>
<tr>
<td>Subtraction</td>
<td>Overborrow</td>
<td>0</td>
<td>Addition</td>
</tr>
<tr>
<td>Addition</td>
<td>No overcarry</td>
<td>0</td>
<td>Addition</td>
</tr>
<tr>
<td>Addition</td>
<td>Overcarry</td>
<td>1</td>
<td>Subtraction</td>
</tr>
</tbody>
</table>

Rules for Von Neumann Division
Figure 31. Division Example
These rules may be applied to the example in figure 31.

In order to perform this division in RECOMP, several things must be considered. A possible double-length dividend must be provided at the start. The A- and R-registers fulfill this requirement. Also, the computer must be capable of performing both addition and subtraction of the divisor in B upon the full or reduced dividend in A. The adder-subtractor logic previously described is used, with flip-flop E₀ true for addition and false for subtraction. Note from the example of figure 31 that the divisor is shifted right for each operation. In the computer it is more convenient to maintain the original orientation of the divisor by recirculating B, and left-shift the reduced dividend in A and R instead. As each new quotient bit is determined, it is written in the R-register. Simultaneously, another dividend bit (if double length) is transferred from R to A while the loading A-register bit, no longer needed, is dropped.

**Division during U₁**

At the start of I₃ the divisor is stored in the B-register after having been transferred from the memory during I₂. The dividend stands in the A-register or the A- and R-registers depending upon whether a single or double-length dividend is specified (figure 32).

During U₁ the divisor in B is subtracted from the dividend in A using the normal subtract logic. Hence, E₀ is reset for subtraction

\[ 0^e_0 = M_2 U_1 C \]

and R₁ is set to gate B into the subtractor

\[ 1^r_1 = M_2 U_1 C \]

The difference, A-B, is written back in A and shifted one bit left because of the delay in S.

\[ 1^a_41 = S M_2 T U_{41} 'T_1 C \]

\[ 0^a_41 = S'M_2 T U_{41} 'C \]

where U₄₁ identifies only the final word time of division and, therefore,

\[ U_1 \subseteq U_{41} ' \]
Figure 32. Division during $U_1$
At $T_1$, $A_{41}$ is set to zero for the proper sync bit.

$$0^a_{41} = M_2'E_20'U_41'T_1'C$$

If the divisor is less than or equal to the dividend, no overborrow occurs at $T_{40}$ and the overflow flip-flop $I_0$ is turned on at $T_{41}$ to indicate improper division and possibly stop computation.

$$i_0 = M_2'K'x'T_x'U_{1T_{41}}C$$

Left-shifting of the R-register is accomplished using flip-flops $R_{43}$ and $R_{42}$ but not $R_1$.

- $1^r_{43} = R_2'T_40'T_{41}a'C$
- $0^r_{43} = R_2'T_{41}a'C + T_{40}C$
- $1^r_{42} = R_43'M_2'C$
- $0^r_{42} = R_43'M_2'C$
- $1^r_{41} = R_42'M_2U_{41}C$
- $0^r_{41} = R_42'M_2U_{41}C$

At $U_{1T_{41}}$, $R_{43}$ receives the sign of the quotient, plus if $A_1 = B_1$. If minus ($A_1 = B_1'$), no logic is needed since $R_{43}$ has already been zero-set at $T_{40}$ time.

$$1^r_{43} = A_1B_1'I_3U_1T_x'M_4'T_{41}a'C + A_1'B_1'I_3U_1M_4'T_{41}a'C (M_4'T_{41}a = M_{41})$$
- $0^r_{43} = T_{40}C$

The sign of the dividend is in $A_{41}$ at $T_1$ and, therefore, in $A_{40}$ at $T_2$. It is copied at this time into $B_{42}$ and held there.

- $1^b_{42} = A_{40}M_2'M_9'U_1T_{42}C$
- $0^b_{42} = N_{11}I_3'T_{41}C$
No logic is needed for a minus sign since \( B_{42} \) is set to zero previously. The B-register exhibits normal recirculation throughout the entire division process.

\[
\begin{array}{c}
1_{41}^b = B_1 \times M_2 C = B_1 \times T_1 'M_1 'M_0 'M_0 'I_3 C \\
0_{41}^b = B_1 'M_2 C = B_1 'T_1 'M_1 'M_0 'M_0 'I_3 C
\end{array}
\]

Division in \( U_{1'} \) and \( U_{41} \)

The same logic operations are performed each word time between the first, \( U_1 \), and the last, \( U_{41} \), and then are essentially the mechanization of the Von Neumann process. The special logic for \( U_{1'} \) and \( U_{41} \) follow and are tabulated along with logic for \( U_1 \).

**Division Logic**

\( U_{1'} \) Unrounded and Rounded, Double Length, \( (D_2) \)

\[
\begin{array}{c}
1^e_0 = N_5 C \\
0^e_0 = M_2 U_1 C \\
0^k_a = T_41 C \\
0^b_{42} = N_1 'I_3 'T_41 C \\
0^a_{41} = M_2 E_20 'U_41 'T_1 C \\
1^r_{43} = R_2 'T_40 'T_41 'T_1 'M_1 'I_41 'T_41 a C \\
+ A_1 'B_1 'I_3 'U_41 'M_1 'T_41 a C \\
0^r_{43} = R_2 'T_41 'C + T_41 C \\
0^a_{41} = S'M_2 'U_2 'T_41 C
\end{array}
\]
If single length is desired, delete. \( A_2 \) remains zero for \( D_2 \).

\[
\begin{align*}
1^a_2 &= R_{42} \cdot M \cdot D \cdot U_{41} \cdot T \cdot C \\
0^a_2 &= A_3 \cdot M_2 \cdot T \cdot C
\end{align*}
\]

\[
1^r_1 = M_2 \cdot U \cdot C
\]

\[
1^i_0 = K_1 \cdot M_2 \cdot U \cdot T \cdot 41 \cdot C
\]

\( U_{11} \cdot U_{41} \), Unrounded and Rounded, Double Length, \( (D_2) \)

\[
\begin{align*}
1^k_0 &= T \cdot C \\
0^a_0 &= A_2 \\
1^e_0 &= M_2 \cdot T \cdot U \cdot 2 \cdot 41 \cdot C + A_1 \cdot K_1 \cdot M_2 \cdot U \cdot T \cdot 41 \cdot C \\
0^e_0 &= A_1 \cdot K_1 \cdot M_2 \cdot T \cdot 41 \cdot C \\
0^a_{41} &= M_2 \cdot E_{20} \cdot T \cdot 41 \cdot T \cdot C
\end{align*}
\]

\[
\begin{align*}
1^r_{43} &= R_{21} \cdot T \cdot T \cdot 40 \cdot 41 \cdot C + R_{21} \cdot U \cdot T \cdot 41 \cdot C + A_1 \cdot K_1 \cdot U \cdot T \cdot 41 \cdot C
+ E_0 \cdot K_1 \cdot U \cdot M_4 \cdot T \cdot 41 \cdot C \\
0^r_{43} &= R_{21} \cdot T \cdot 41 \cdot C + M_1 \cdot E_0 \cdot K_1 \cdot T \cdot 41 \cdot C + A_1 \cdot K_1 \cdot U \cdot T \cdot 41 \cdot C + T \cdot 40 \\
0^a_{41} &= S \cdot M_2 \cdot T \cdot U \cdot 41 \cdot T \cdot C \\
0^a_{41} &= S \cdot M_2 \cdot T \cdot 41 \cdot C
\end{align*}
\]

If single length is desired, delete. \( A_2 \) remains zero for \( D_2 \).

\[
\begin{align*}
1^a_2 &= R_{42} \cdot M \cdot D \cdot U_{41} \cdot T \cdot C \\
0^a_2 &= A_3 \cdot M_2 \cdot T \cdot C
\end{align*}
\]

Note: 41st word time rounded same as \( U_{11} \cdot U_{41} \)
$U_{41}$, Rounded, ($D_1$)

$1^{u}_{41} = M_2 D_1 Q_6 Q_4 T_{41a} C$

$1^{e}_{0} = U_{41} C$

$0^{r}_{1} = D_1 U_{41} C + U_{41} T_{41a} C$

$k_a = R_{43} D_1 U_{41} T_{1} C$

$0^{k}_{a} = E_{0} A_{1} 'B_{1} 'M_{4} 'T_{1} 'C + E_{0} A_{1} 'R_{1} 'M_{4} 'C$

$1^{a}_{41} = R_{41} U_{41} T_{1} C$

$0^{a}_{41} = R_{41} 'U_{41} T_{1} C$

$1^{a}_{40} = S D_{1} U_{41} C$

$1^{a}_{1} = R_2 D_1 U_{41} T_{41} 'C$

$0^{a}_{1} = R_2 'D_{1} U_{41} C + U_{41} T_{41} C$

$1^{r}_{41} = B_{42} T_{1} 'U_{41} M_{9} 'C$

$0^{r}_{41} = B_{42} 'T_{1} 'U_{41} M_{9} 'C$

$1^{s} = A_{1} 'R_{1} 'K_{a} C$

$0^{s} = A_{1} R_{1} 'K_{a} C$

$1^{i}_{0} = K_{a} D_1 U_{41} T_{41} C$

Note: Check carefully $S$ and $K_a$ logic for entire word time.
Unrounded, (D₁')

\[ l^{u_{41}} = M_{21}^Q Q_{641}^Q Q_{1}^T 4_{1a} \]

\[ 0^r_{1} = U_{41} T_{41a} C + E_{0}^{U} C \]

\[ a_{41} = R_{41} U_{141} T_{1} C + R_{42} D_{1}^{U} C \]

\[ 0^a_{41} = R_{41}^{U} U_{41} T_{1} C + R_{42}^{D_{1}} U_{41} C \]

\[ r_{43} = R_{T} T_{40}^{4} C + R_{U}^{'T} T_{41a} 2_{1} C \]

\[ l^{r_{40}} = S U_{41} T_{41}^{'M} C \]

\[ 0^{r_{40}} = S^{U} T_{41} T_{9}^{'C} \]

\[ a_{41} = A_{3} D_{1}^{U} C \]

\[ 0_{41} = U_{41} T_{41} C + A_{3} D_{1}^{U} C \]

\[ 0^e_{0} = U_{41} C \]

\[ a_{41} = A_{3} U_{41} C \]

\[ 0^{a_{2}} = A_{3} U_{41} C \]

\[ r^{1}_{41} = B_{42} T_{41}^{'M} C \]

\[ 0^{r_{41}} = B_{42} T_{41}^{'U} M_{9} \]

**Square Root**

The basis for finding arithmetic square root is illustrated in the following example:
Example 1: Find $\sqrt{X}$

1. Estimate $R_1$ and obtain a remainder $A_1 = X - R_1^2$.
2. Estimate an increment $h_1$ such that $h_1(2R_1 + h_1) \leq A_1$.
3. $A_2 = A_1 - h_1 (2R_1 + h_1)$
   \[ = X - R_1^2 - 2R_1 h_1 - h_1^2 \]
   \[ = X - (R_1 + h_1)^2 = X - R_2^2 \]
4. In general, after $i$ steps $A_{i+1} = A_i - h_i (2R_i + h_i)$.

Decimal Example

Find $\sqrt{.063504}$

\[
\begin{array}{cccc}
\text{063504} & \div & \text{.252} \\
\hline
\text{.45} & \text{0235} & \text{0225} & \text{.04} \\
\hline
\text{.502} & \text{1004} & \text{1004} & \text{R_1 = .2} \quad \text{A_1 = .023504} \\
\hline
\text{h_1 = .05} & \text{h_2 = .25} & \text{A_2 = .001004} \\
\hline
\text{A_3 = .000000} \\
\hline
\text{R_3 = .252} \\
\end{array}
\]

The same procedure still holds in binary arithmetic with certain simplifications: (a) considering only proper fractions, $h_i = 2^{-i-1}$; (b) establishing this as a fixed routine, 1 will always be tried as a "next" bit in $R_i$. If it does not work, it is necessary to restore the previous remainder and put a 0 as the next digit of $R_i$ before continuing. This provides a "restoring" square root procedure. The general step is provided by:

\[
A_{i+1} = A_i - h_i (2R_i + h_i)
\]

\[
= A_i - 2^{-i-1} (2R_i + 2^{i-1})
\]

\[
= A_i - 2^{-i} (R_i + 2^{-i-2})
\]
If \( A_{i+1} \) is negative, \( R_{i+1} = R_i \), add back the subtracted quantity to restore \( A_i \) and then proceed to step \( i+1 \).

\[
A_{i+1} = A_i - 2^{-i+1} (R_i + 2^{-i+1} - 2)
\]

\[
= A_i - 2^{-i-1} R_i - 2^{-2i-4}
\]

\[
= A_i - 2 (2^{-i-1} R_i) + 2^{-i-1} R_i - 4 (2^{-2i-4}) + 3 (2^{-2i-4})
\]

\[
= A_i - 2^{-i} R_i - 4 (2^{-2i-4}) + 2^{-i-1} R_i + 3 (2^{-2i-4})
\]

\[
= \left[A_i - 2^{-i} (R_i + 2^{-i-2})\right] + 2^{-i-1} (R_i + 3 (2^{-i-3}))
\]

The quantity in brackets is the partial remainder before restoring; a nonrestoring procedure is given by the last term.

If \( A_i \geq 0, A_{i+1} = A_i - 2^{-i} (R_i + 2^{-i-2}) \), and

if \( A_i < 0, A_{i+1} = A_i + 2^{-i} (R_i + (3) 2^{-i-2}) \)

**Binary Example** (negatives expressed by true binary compliments)

Find \( \sqrt{0.11001} \)

\[
\begin{array}{c|c}
0.11001 & .101 \\
01 & \\
0010 & 101 \\
110101 & A_1 > 0, R_1 = .1 \\
1011 & A_2 < 0, R_2 = .10 \\
000000 & A_3 \leq 0, R_3 = .101 \\
\end{array}
\]

**Example 2:** Find \( \sqrt{\frac{81}{256}} = \sqrt{0.1010001} \)
Let $R_i$ = Partial root at end of $i$th step
Let $A_i$ = Remainder at end of $i$th step

Rules of Operation:

If $A_i$ is $+$, 1 has just been inserted as right-most digit of $R_i$.
If $A_i$ is $-$, 0 has just been inserted as right-most digit of $R_i$.
If $A_i$ is $+$, next step is to subtract $2^{-i} \left\{ \left[ (R_i) + 1 \right] \left[ 2^{-i+2} \right] \right\}$
If $A_i$ is $-$, next step is to add $2^{-i} \left\{ \left[ (R_i) + 3 \right] \left[ 2^{-i+2} \right] \right\}$

Mechanization of Square Root

The following description of the mechanization can be easily followed if the operation is traced through the example on the following page. The primed headings indicate extensions of normal registers. The primed entries in the $R$-register indicate the partial root.

Three registers are used, $A$, $B$, and $R$. Each is extended by two flip-flops over its normal length.

During the first word time, the operand, initially in $A$, is shifted into $B$. $R$ is cleared. In the table on the following page, arrows indicate recirculation paths, and underlines indicate the insertion of bits from logic, not recirculation.

At the last bit time of each word and the first bit time of the next word (bit times 5 and 1 of the example), two bits are shifted from $B$ into the right-hand bit position of $A$ (from $B_6$ to $A_1$). These two bits are in each case the two "next" bits not yet operated upon. At the end of each word time, the right-hand bit of $R$ ($R_7$) is set to 1 and the new bit of the partial root is inserted in the left-hand bit position ($R_6$). At the next bit time, the right-hand bit of $R$ is set to 0 if $A_i$ is positive (i.e., if $A_1$ is 0) and to 1 if $A_i$ is negative (i.e., $A_1$ is 1). During every word time after the first, $B$ recirculates. Its 2 additional bits cause a left-shift of two bits each word time.
The contents of the right-hand bit of the R-register is subtracted from (or added to) the right-hand bit of A, according to the sign of $A_i$ and the rules of the preceding section. The difference (or sum) is entered into the left-hand bit of the A-register.

R has a complicated recirculation pattern. The bit in the next-to-right-hand bit position shifts into the left-hand bit position ($R_1$ to $R_6$), then shifts one to the right ($R_5$), and then goes two places: the next position to the right ($R_4$) and also the right-hand bit position ($R_7$). In this way the partial root is built up in R.

The square root process consists of three parts: a shift from A into B (one word time), operation (for an n-bit word this takes n word times), and a shift of the root from R into A (one word time).

During this last word time the root is shifted into A in such a manner that it is in the "correct" position for further use elsewhere (in the example $R_6$ shifts into $A_5$).

Example 3: Find $\sqrt{\frac{1}{16}} = \frac{1}{4} (\sqrt{.0001} = .0100)$.

<table>
<thead>
<tr>
<th>Operand</th>
<th>Partial Root</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  00  .00  01</td>
<td>0</td>
</tr>
<tr>
<td>0  01</td>
<td></td>
</tr>
<tr>
<td>1  11  11  01</td>
<td>0.01</td>
</tr>
<tr>
<td>0  00  11</td>
<td></td>
</tr>
<tr>
<td>0  00  00  00</td>
<td>0.010</td>
</tr>
<tr>
<td>0  01  01</td>
<td></td>
</tr>
<tr>
<td>1  10  11  00</td>
<td>0.0100</td>
</tr>
<tr>
<td>0  10  11</td>
<td></td>
</tr>
<tr>
<td>1  01  11</td>
<td></td>
</tr>
</tbody>
</table>

Definitions of Floating-Point Operations

Given a pair of floating-point numbers having the form $x_1 = m_1 \cdot 2^{e_1}$ and $x_2 = m_2 \cdot 2^{e_2}$, form their sum, difference, product, and quotient as follows.

$$x_1 \pm x_2 = (m_1 \pm m_2 \cdot 2^{e_2-e_1}) \cdot 2^{e_1}, \text{ assuming } e_1 \geq e_2$$
Floating-Point Addition and Subtraction

Before \( m_1 \) and \( m_2 \) can be added or subtracted, they must be scaled to the same exponent. This is accomplished by denormalizing the number having the smaller exponent until the two exponents are equal (i.e., shifting \( m_1 \) right and augmenting \( e_1 \) by the number of such shifts). In the example above, \( e_2 \) was assumed \( \leq e_1 \) and, therefore, \( x_2 \) was denormalized to

\[
x_2 = m_2 \cdot 2^{e_2 - e_1}
\]

where \( m_3 = m_2 \cdot 2^{e_2 - e_1} \)

Procedure

The following must be accomplished in the order mentioned.

1. The four arguments \( m_1, e_1, m_2, e_2 \) must be properly stored in arithmetic registers.
2. Exponents \( e_1 \) and \( e_2 \) must be compared to determine which is smaller (assume \( e_2 \leq e_1 \) here).
3. If \( x_2 \) has the smaller exponent, it must be denormalized by right-shifting \( m_2 \) and augmenting \( e_2 \) according to the following:
   a. If \( e_2 \geq 0 \), perform the addition \( /e_2/ + 1 \) and assign a plus sign to the result.
   b. If \( e_2 < 0 \), perform the subtraction \( /e_2/ - 1 \) and assign a minus sign to the result.
4. Comparison of \( e_1 \) and \( e_2 \) occurs at each step during denormalization until \( e_1 = e_2 \) at which time denormalization is concluded.
5. Addition or subtraction of \( m_1 \) and \( m_2 \) is performed according to the operation code of the floating-point command.
6. Normalization of resulting sum or difference occurs to insure

\[
1/2 \leq |m_1 \pm m_2| < 1
\]
Additional details that must be considered are:

1. If \( m_1 \) or \( m_2 \) is initially zero or becomes zero as a result of denormalization, the denormalization process must be concluded and addition or subtraction begun.

2. If the result of the addition or subtraction (paragraph 5, preceding page) is zero, the final normalization process must be omitted.

3. To avoid possible overflow from addition, both \( x_1 \) and \( x_2 \) are subjected to an initial one-bit denormalization so that

\[
/m_1' \pm m_2' < 1/2
\]

where \( m_1' = 1/2m_1 \) and \( m_2' = 1/2m_2 \).

**Operation Identification**

The floating-point add (FAD) and floating-point subtract (FSB) commands have operations codes of 04 and 06, respectively. Both commands are identified with a new flip-flop \( M_8 \) which is one-set simultaneously with \( I_3 \). Floating-point multiply (FMP) and floating-point divide (FDV) commands have operation codes of 07 and 05, respectively, and are identified with another flip-flop designated \( M_9 \).

**Operand Working Storage**

Before any floating-point command is to be executed, the first operand \( x_1 \) must appear with its mantissa \( m_1 \) in the A register and its exponent \( e_1 \) in the R register. The second operand, \( x_2 \), exists initially in two consecutive memory locations, the first location being given by the address of the floating-point command. Thus, during \( I_2 \), the first word of \( x_2 \), which is \( m_2 \), is gated from memory into the B-register. The second word, \( e_2 \), is gated during the next word time \( I_3 U_1 \) into the X-register.*

**Sequence Control**

Events occurring during the execution of the FAD and FSB commands are controlled by several flip-flops as indicated in figure 33.

* A new one-word recirculating register
Figure 33. Floating Point Add-Subtract Sequence
Logic for FAD and FSB Commands

A detailed description of the changes in RECOMP logic necessitated by the FAD and FSB commands follows. Octal operation codes 04, 06, 44, and 46 set flip-flop M₈ at the end of I₂.

\[ l_{m8} = D_5'D_4'D_3'D_2'N_1'J_1'C \]

M₄ is also set by the FAD and FSB commands codes by

\[ l_{m4} = D_5'D_4'D_3'D_2'N_1'J_1'C \]

while M₀ must be inhibited by D₃' in

\[ l_{m0} = D_5'D_4'D_3'N_1'J_1'C \]

With M₄D₃ now true, terms required for the FNM command (code 45) must be inhibited by M₈'M₉'.

\[ l_{m6} = M_4'M_8'M_9'D_3'U'T'C \]

\[ l_{n12} = A_1M_4M'_M'D_9'D_4'D_3'U'T_40'C + A_2M_4M'_M'D_9'D_3'U'T_40'C \]

Also, since M₄D₂ is for FSB as well as STA commands M₆ and D₂', must be inhibited to prevent STO code from being activated at end of U₁. D₃' (or M₈') may be used.

\[ l_{m6} = M_4'D_3'D_2'D_1'T_41a'C \]

\[ l_{a2} = M_4'D_3'T_41a'C \]

\[ l_{m4} = M_4'D_3'T_41a'C \]

To prevent STA logic from affecting FSB command, D₃' (or M₈') must be inserted in the following

\[ l_{n7} = A_2M_4D_3'P_6'P_5'P_3'P_1'C \]
During the first word time of $M_8$, the A, B, and R-registers must recirculate $m_1$, $m_2$, and $e_1$, respectively.

For A-register recirculation, add

\[
\begin{align*}
1^a_{41} &= A_1M_4D_3'D_2D_1'C \\
0^a_{41} &= A_1'M_4D_3'D_2D_1'C
\end{align*}
\]

For B-register recirculation, add gates

\[
\begin{align*}
1^b_{41} &= B_1M_8M_0'Q_4'C \\
0^b_{41} &= B_1'M_8M_0'Q_4'C
\end{align*}
\]

where $Q_4'$ is certainly true during the first word time.

For R-register recirculation during the first word time:

\[
\begin{align*}
1^r_{41} &= R_1K_1'M_4D_3'T_41a'C + E_0'M_4D_3'T_41a'C \\
0^r_{41} &= R_1K_1'M_4D_3'T_41a'C + E_0'M_4D_3'T_41a'C
\end{align*}
\]

where $E_0$ contains the sign of the R contents.

During the first word time the X-register receives the exponent $e_2$ from memory or high-speed loop

\[
\begin{align*}
1^x_{41} &= M_r'p_1'M_4U_1'C + K_Lp_1'M_4U_1'C \\
0^x_{41} &= M_r'p_1'M_4U_1'O_1'C + K_Lp_1'M_4U_1'O_1'C
\end{align*}
\]
During the second word time of Mₙ₂, both operands are denormalized by one bit in order to avoid overflow from addition of mantissas. Denormalization of x₁ is accomplished by shifting the A contents one bit to generate m₁/2, and augment the R contents by one to generate e₁+1:

\[
x₁ = m₁ \cdot 2^{e₁} = m₁/2 \cdot 2^{e₁+1}
\]

Right-shifting of the A-register as exists for the ARS command (code 40) may be extended to the FAD and FSB commands merely if D₂' is omitted from:

\[
\begin{array}{c}
1^₁₄₁ = A₂ M₄ D₁ 'U₁ 'T₁ 'T₄₀ 'T₄₁a 'C \\
0^₁₄₁ = A₂ 'M₄ D₁ 'U₁ 'T₄₁a 'C \\
\end{array}
\]

and preserve the sign by replacing D₂' with U₁' in:

\[
\begin{array}{c}
1^₁₄₁ = A M₄ U₁ 'T₄₁a 'C \\
0^₁₄₁ = A₁ 'M₄ U₁ 'T₄₁a 'C \\
\end{array}
\]
and remove $D_2'$ from:

$$0_{41}^a = M_{41}'T_1'C + M_{41}'U_{11}'T_{40}C$$

No difficulty arises with the STA command since code 42 exists during $U_1$ only.

Augmentation of the R-register requires sampling to determine whether $e_1 \geq 0$ or $< 0$. If $e_1 \geq 0$, $/e_1/$ must be increased by one and if $e_1 < 0$, $/e_1/$ must be decreased by one.

$Q_0$ samples $e_1$ for zero as it shifts through the R-register. $Q_0$ is set at $T_1$ and reset if a "one" bit is sensed in $R_{41}$.

<table>
<thead>
<tr>
<th>$1q_0$</th>
<th>$D_3T_1'C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0q_0$</td>
<td>$R_{41}D_3T_1'T_2'C$</td>
</tr>
</tbody>
</table>

If $Q_0$ is still true at $T_{41}$, all bits in $R$ except possibly the sign and most significant bit are zero. It is unreasonable to assume the most significant bit "one" since this would mean:

$$x_1 \geq \left(2^{38} - 1\right)$$

which is larger than the number of grains of sand that would fill the universe.

The states of $R_{41}$ and $Q_0$ at the end of a word time depend upon the exponent $e_1$ and are, therefore, used to set $E_0$ for augmenting or reducing this exponent.

<table>
<thead>
<tr>
<th>$1e_0$</th>
<th>$R_{41}M_4D_3T_1'C + Q_0M_8M_0'T_{40}C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0e_0$</td>
<td>$R_{41}'M_4D_3T_1'C$</td>
</tr>
</tbody>
</table>

$E_0$ determines whether the R-register acts as a one-input adder or subtractor. If the exponent is positive, $R_{41} = 1$ at $T_1$ and if the exponent is zero $Q_0 = 1$ at $T_{40}$. In either case, $E_0$ is set and one input addition occurs because of
\[ k_a = E_0 R_1 M_4 D_1 T_1' C \]

(i.e., the carry is turned off by the first zero bit in \( R_1 \)). If the exponent is negative, \( R_{41}' \) turns off \( E_0 \) at \( T_1 \) which causes subtraction by means of:

\[ k_a = E_0 R_1 M_4 D_1 T_1' C \]

The use of \( T_{40} \) in the second gate in \( E_0 \) above is necessary to obtain a plus sign (since \( R_{41} \) copies the sign from \( E_0 \) at \( T_{41} \)) when the exponent is zero. If this were not done, \( E_0 \) would get zero-set at the next \( T_1 \) by \( R_{41}' \) which would result in subtraction instead of addition and the exponent would oscillate between -0 and +1 instead of going from -1 to +0 to +1 to +2, etc.

Gates in \( k_a \) used for normalization are inhibited at this time by \( D_1 \) in:

\[ k_a = E_0 R_1 M_4 D_1 T_1' C + E_0 R_1 M_4 D_1 T_1' C \]

Recirculation of \( R \) during execution of ARS, ALS, and STA commands is assured by \( D_3' \) in:

\[ r_{41} = R_1 M_4 D_1' D_1' C + R_1 M_4 D_3 D_2' C \]

\[ 0^r_{41} = R_1 M_4 D_1' D_1' C + R_1 M_4 D_3 D_2' C \]

In order to denormalize the floating-point operand in the B and X registers a number of flip-flops corresponding to \( Q_0, E_0, \) and \( K_a \) are needed. Since the \( Q \) counter is not needed for FAD and FSB, and the C-register is used only for gating \( e_2 \) during the first word time, twelve flip-flops are available for time sharing. In order to avoid conflict with their previous functions, \( M_8' \) is placed in each existing gate in \( Q_1 \) through \( Q_5 \) (\( Q_6 \) is not used). No difficulty occurs from the C flip-flops since their normal shift logic requires \( N_5 \) which is inhibited during \( I_3 \). Hence, add \( M_8' \) to:
for $i = 1, 2, 3, 4, 5$.

Let $Q_4$ be true during the time that denormalization of the operand in $B$ and $X$ must occur, just as $M_4$ is true as long as the operand in $A$ and $R$ requires denormalization. During the second word time, $U_2$, both $M_4$ and $Q_4$ must be true along with $M_8$.

During $M_8Q_4$ the mantissa in $B$ is shifted right.

Let $Q_1$ sample the exponent in $X$ for zero (corresponds to $Q_0$ for $R$). It is zeroed at $T_1$ and turned on if a one is sensed in $X_{41}$.

Let $Q_2$ indicate whether the absolute value of the exponent in $X$ is to be increased or decreased (corresponds to $E_0$ for $R$). If the exponent is positive or zero, one-input addition must occur in $X$; if it is negative and not zero, subtraction must occur.

Since $Q_2$ is initially zero-set at the beginning of $I_3$ and there is not a requirement to re-zero-set $Q_2$ if it becomes one-set, no additional logic is needed for zero-setting $Q_2$. 
A possible difficulty arises when the exponent read from memory is -0 since in this case $X_{41}'$ would normally copy $M_r'$ or $K_0'$ during $U_1 T_{41}$ while $Q_2$ would be set by $Q_1'$ for exponent augmentation. During the succeeding word time then, $Q_2$ would be zeroed by $X_{41}'$ at $T_1$ causing exponent diminution to occur. This problem is solved by using $Q_2$ to inhibit $X_{41}'$ from copying the sign from memory or loop at $U_1 T_{41}$ since $Q_2$ true at this time indicates a zero exponent.

Thus:

| $1^x_{41}$ = $M_{8} Q_2 Q_4 T_{41} C$ |
| $0^x_{41}$ = $M_{8} Q_2 Q_4 T_{41} C$ |

Let $C_4$ be the carry flip-flop for the $X$-register one-input adder-subtractor. Since the $C$ flip-flops are used for memory gating during $U_1$, time-shared gates must be inhibited during this time with $U_1'$. The carry is set at $T_1$ and turned off by $X_1'$ when adding or $X_1$ when subtracting:

| $1^c_{4}$ = $M_{8} U_1' T_{1} C$ |
| $0^c_{4}$ = $M_{8} Q_2 X_1' U_1' T_{1} C + M_{8} Q_2 X_1 U_1' T_{1} C + M_{8} T_{41} C$ |

The $X$-register receives the result of the one-input addition or subtraction during $Q_4$

| $1^x_{41}$ = $X_{1} C_4 Q_4 T_{41} C + X_{1} M_{8} C_4 Q_4 T_{41} C$ |
| $0^x_{41}$ = $X_{1} M_{8} C_4 Q_4 T_{41} C + X_{1} M_{8} C_4 Q_4 T_{41} C$ |

In order to determine which floating-point number requires denormalization after $U_2$, flip-flop $Q_5$ samples the exponents in $R$ and $X$ to determine which is smaller and therefore needs additional denormalization.
Since $Q_5$ must be explained at $T_{41}$, $R_2$ and $X_2$ must contain all bits from least significant to sign during $T_1$ through $T_{40}$ which is clearly the case. Note that if $R > X$, $Q_5 = 1$ at $T_{41}$ and the floating-point number in $B$ and $X$ requires further denormalization, whereas if $R < X$, $Q_5 = 0$ at $T_{41}$ and the floating-point number in $A$ and $R$ requires further denormalization.

Thus if $Q_5 = 0$ we turn off $Q_4$ so that $B$ and $X$ merely recirculate, and leave $M_4$ on:

$$
\begin{align*}
1_{q_4} &= M_8 D_6 X_{2} R_{2} U_{2} C \\
0_{q_4} &= M_8 X_2 R_{2} U_{2} C
\end{align*}
$$

whereas if $Q_5 = 1$ we turn off $M_4$ so that $A$ and $R$ recirculate, and leave $Q_4$ on:

$$
\begin{align*}
0_{m_4} &= M_8 M_0 Q_{5} U_{2} T_{41} C \\
1_{a_4} &= A_1 M_0 M_1 M_2 M_3 M_4 M_1 P_0 C \\
0_{a_4} &= A_1 M_0 M_1 M_2 M_3 M_4 M_1 P_0 C \\
1_{r_4} &= R_1 M_0 M_1 M_2 M_3 M_4 M_1 I_0 C \\
0_{r_4} &= R_1 M_0 M_1 M_2 M_3 M_4 M_1 I_0 C
\end{align*}
$$
After U₁, the exponents in R and X are compared for equality every word time. When agreement occurs, denormalization of both FP numbers is terminated. Flip-flop C₁ samples for agreement by being zeroed at T₄₁ and turned on when a pair of corresponding bits disagree.

\[
\begin{align*}
1c₁ &= M₈ \cdot U₁ \cdot 'R₄₁ \cdot X₄₁ 'I₅ T₂ 'C + M₈ \cdot U₁ \cdot 'R₄₁ 'X₄₁ 'I₅ T₂ 'C \\
0c₁ &= M₈ \cdot T₄₁ 'C
\end{align*}
\]

Note that C₁ samples R₄₁ and X₄₁ at T₁ for sign agreement, does not sample the sync bits at T₂, and samples the numerical bits during T₃-T₄₀. As before, if we assume the most significant bits to be zero, C₁ can not be triggered on the true side at T₄₁. One anomaly still exists, namely, when both exponents are zero, for in this case the exponent that is being increased from -1 to +0 contains a minus in the sign position until zero sensing is completed. This is overcome by turning C₁ off again at T₄₀ if both exponents are zero. Thus:

\[
0c₁ = M₈ \cdot U₁ 'Q₁ 'Q₁ 'T₄₀ 'C
\]

Note that the two leading exponent bits are not being sampled in this situation.

When C₁' exists at T₄₁, exponents are equal and denormalization is terminated as follows.

\[
\begin{align*}
0q₄ &= M₈ \cdot U₁ 'C₁ 'T₄₁ 'C \\
0m₄ &= M₈ \cdot M₀ 'U₁ 'C₁ 'T₄₁ 'C
\end{align*}
\]

Denormalization is also terminated when either mantissa is initially zero or becomes zero during denormalization. Flip-flops C₂ and C₃ sample the A and B registers respectively for zero.
$c_2 = A_2 M_2 U_2 'C$

$0 c_2 = M_2 T_41 C$

$1 c_3 = B_2 M_2 U_2 'C$

$0 c_3 = M_2 T_41 aC$

$T_{41}$ is not needed since zero sync bits were written into $A_{41}$ and $B_{41}$ at $T_1$.

Now if $C_2 = 0$ at $T_{40}$, $A$ must be zero, whereas if $C_3 = 0$ at $T_{40}$, $B$ must be zero. In either case $C_1$ is turned off which terminates denormalization.

$0 c_1 = M_1 U_1 'C' T_40 C + M_1 U_1 'C' T_40 C$

Note that the signs appear in $A_2$ and $B_2$ at $T_{40}$ and therefore cannot set $C_2$ or $C_3$ in time to prevent $C_1$ from being turned off.

Between $U_2$ and the end of denormalization, one of the floating-point numbers is being denormalized while the other must be preserved by recirculation. The number in $A$ and $R$ must recirculate during $M_4'$ and the number in $B$ and $X$ during $Q_4'$.

$a_{41} = A_1 M_0 ' M_1 M_2 M_3 M_4 ' M_1 ' M_0 ' P_0 ' C$

$0 a_{41} = A_1 M_0 ' M_1 M_2 M_3 M_4 ' M_1 ' M_0 ' P_0 ' C$

$r_{41} = R_1 M_0 ' M_1 M_2 M_3 M_4 ' M_1 ' I_3 ' C$

$0 r_{41} = R_1 M_0 ' M_1 M_2 M_3 M_4 ' M_1 ' I_3 ' C$

$b_{41} = B_1 M_0 ' M_1 ' Q_4 ' C$

$0 b_{41} = B_1 M_0 ' M_1 ' Q_4 ' C$
Upon termination of denormalization, addition or subtraction of mantissi is initiated. \( M_0 \), \( U_1 \), and \( E_0 \) are turned on.

\[
\begin{align*}
1^{x_{41}} &= X_{1} Q_{4} 'U_{1}'M_{10}'C \\
0^{x_{41}} &= X_{1} Q_{4} 'U_{1}'M_{10}'C
\end{align*}
\]

To prevent \( U_1 \) from being pulsed on the prime side we have

\[
0^{u_{1}} = I_{3} U_{1} 'I_{5} 'T_{41} C
\]

At this time the \( \text{FAD} (04) \) and \( \text{FSB} (06) \) codes must be made to look like \( \text{ADD} (01) \) and \( \text{SUB} (03) \). But \( D_6, D_5, D_4, \) and \( D_3 \) do not appear in any logic involving \( M_0 \), and \( D_2 \) is already set correctly for addition or subtraction. Hence it suffices only to set \( D_1 \):

\[
1^{d_{1}} = M_{8} U_{1} 'C_{1} 'T_{41} C
\]

\( M_0 \) remains on for two word times during which the computer performs ordinary addition or subtraction as required. At the end of \( U_2 \) the sum of the mantissi is in \( A \), so that addition or subtraction is terminated and final normalization begins. \( M_0 \) is reset

\[
0^{m_{0}} = M_{9} U_{2} 'T_{41} C
\]

and \( N_{12} \) inhibited by \( M_{8} ' \)

\[
1^{n_{12}} = M_{0} M_{8} 'U_{2} 'T_{40} C
\]

The \( \text{FNM} \) code (45) is set up.
U₁ is set on and M₈ is terminated.

From here on the computer sees an ordinary FNM command, and I₃ is terminated when normalization of the result in A is accomplished.

If the resulting mantissa in A is zero, H₀ is set and terminates the operation. Thus

Note that the FNM command no longer transfers control when the argument is zero but is ignored instead. Also note that the exponents in R and X will usually be equal upon termination of denormalization. Hence we may choose R in which case it is preserved by recirculation during M₀.

while R₁ is used to gate B₁ into the adder.
The exception to the above (i.e., when R and X are unequal at the end of denormalization) occurs if A or B becomes zero. In this case, denormalization is immediately terminated and addition or subtraction is begun. However, the exponent that prevails in the final result must be that associated with the non-zero mantissa. Thus if B = 0, the exponent in R must remain in R, while if A = 0, the exponent in X must be transferred into R. In the latter case, C_2' will exist at T_40 when A = 0 and this turns on Q_3 which is used for gating X into R.

\[
\begin{align*}
q_3 &= M_8 T' U C_2' T_40 C \\
q_3 &= U_1 T C \\
-q_3 &= X_1 M_0 Q_3 C + R_42 M'_0 Q_3 C \\
-r_41 &= X_1 M_0 Q_3 C + R_42 M'_0 Q_3 C
\end{align*}
\]

Q_3 being false during M_0 U_2, the right-hand gates preserve the exponent in R.

Another point is that the G-counter must be inhibited from counting every time I_3 U_1 comes true during the floating-point commands, since this occurs three times. This is solved by allowing N_4 to come true only during the last I_3 U_1 when M_8 is no longer on.

\[
\begin{align*}
-n_4 &= I_3 M' M' U_1 T C
\end{align*}
\]

**Floating-Point Multiplication and Division**

Multiplication of two floating-point numbers, x_1 and x_2, involves addition of their exponents e_1 + e_2 and multiplication of their mantissi m_1, m_2. Division of two floating-point numbers involves subtraction of exponents e_1 - e_2 and division of mantissi m_1/m_2.

**Procedure**

The following must be accomplished in the order described.

1. The four arguments m_1, e_1, m_2, e_2 must be properly stored in arithmetic registers.
2. Exponents $e_1$ and $e_2$ must be added or subtracted as determined by the operation codes.

3. Mantissi $m_1$ and $m_2$ must be multiplied or divided as determined by the operation code.

4. Normalization of resulting floating-point product or quotient. An additional detail to be considered is that overflow may result from division if the dividend is larger than the divisor. If it is assumed that both numbers are in normalized form initially, then a one-bit denormalization of the dividend will ensure that no division overflow occurs.

Operation Identification

Floating-point multiply (FMP) has operation code 07 and floating-point divide (FDV) has operation code 05. Both are identified with flip-flop $M_9$. For both operations the addition or subtraction of exponents occurs during the time $M_{10}$ is true.

Operand Working Storage

These are identical with the registers used for the FAD and FSB commands, namely $m_1$ in A, $e_1$ in R, $m_2$ in B, and $e_2$ in X.

Sequence Control

Control of the FMP and FDV operations is as shown in figure 34 on the following page.

Logic for FMP and FDV Commands

Operation codes 05 and 07 set $M_9$ by means of:

$$m = D'1'D'1'D'D I N J'C$$

$M_4$ is also set by:

$$1^{m_4} = D'1'D'1'D'I N J'C$$

$D_1$ is immediately turned off by:

$$0^{d_1} = M_9 M_4 C$$

so that the XAR command (code 43) is preserved by:
Figure 34. Floating Point Multiply-Divide Sequence
| $1^a_{41}$ | $R_1 M_4 D_3 T_4 D_1 C$ |
| $0^a_{41}$ | $R_1 M_4 D_3 T_2 D_1 C$ |
| $1^r_{41}$ | $A_1 M_4 D_3 T_2 D_1 C$ |
| $0^r_{41}$ | $A_1 M_4 D_3 T_2 D_1 C$ |
| $1^n_{12}$ | $M_4 D_3 T_2 D_1 40 C$ |

During $U_1$ the $A$, $B$, and $R$ registers recirculate while $X$ receives the exponent $e_2$ from memory:

| $1^a_{41}$ | $A_1 M_4 D_3 U_{1} C$ |
| $0^a_{41}$ | $A_1 M_4 D_3 U_{1} C$ |
| $1^b_{41}$ | $B_1 T M_0 M_8 10 I_{1} C$ |
| $0^b_{41}$ | $B_1 T M_0 M_8 10 I_{1} C$ |
| $1^r_{41}$ | $R_1 K_1 M_4 D_3 T_4 D_1 C + E_0 M_4 D_3 T_4 D_1 C$ |
| $0^r_{41}$ | $R_1 K_1 M_4 D_3 T_4 D_1 C + E_0 M_4 D_3 T_4 D_1 C$ |
| $1^x_{41}$ | $M_{r} M_{1} T_{p} M_{4} U_{1} C + K_{r} M_{1} T_{p} M_{4} U_{1} C$ |
| $0^x_{41}$ | $M_{r} M_{1} T_{p} M_{4} U_{1} Q_{2} C + K_{r} M_{1} T_{p} M_{4} U_{1} Q_{2} C$ |

During $U_2$ the floating-point number in $A$ and $R$ is denormalized as during FAD and FSB commands. $M_4 D_3 D_1$ exists to generate ARS logic for $A$, and one-input add logic for $R$. $M_4$ is reset at $U_2 T_{41}$.

| $0^m_{4}$ | $M_{4} U_{1} T_{41} C$ |

$B$ and $X$ continue recirculation during $U_2$, $B$ by the same terms as during $U_1$, and $X$ by means of
Inasmuch as three arithmetic registers (A, B, and R) are re-
quired to perform the multiplication or division, it is necessary to
first add or subtract exponents and store the resulting exponent in X.
M_{10} identifies this operation involving the exponents. Both U_1 and E_0
must be turned on at this time.

\[
\begin{align*}
1_x^{41} &= X_1 \cdot Q_4 \cdot U_1 \cdot M_{10} \cdot C \\
0_x^{41} &= X_1 \cdot Q_4 \cdot U_1 \cdot M_{10} \cdot C \\
\end{align*}
\]

\[
\begin{align*}
1^m_{10} &= M \cdot M \cdot U \cdot T \cdot 41a \\
1^u_{1} &= M \cdot M \cdot U \cdot T \cdot 41a \\
1^e_{0} &= M \cdot M \cdot U \cdot T \cdot 41a \\
\end{align*}
\]

During the two word times of M_{10}, R and X must be added or subtracted.
In order to utilize the existing add-subtract logic these numbers must
pass through A_1 and B_1 with R_1 set for gating B_1 into the adder. This is
accomplished by having A_1 copy R_2, B_1 copy X_2 and R_1 set to one.
However, since the mantissi awaiting multiplication or division are in
A and B, these must be preserved by recirculation from A_2 to A_{42} and
from B_2 to B_{42}. The sum or difference is written into X.

\[
\begin{align*}
1^a_{1} &= R \cdot M_{10} \\
0^a_{1} &= R \cdot M_{10} \\
1^b_{1} &= X \cdot M_{10} \\
0^b_{1} &= X \cdot M_{10} \\
1^r_{1} &= M_{10} \cdot T_{1} \\
0^r_{1} &= M_{10} \cdot T_{40} \\
\end{align*}
\]
The following must be inhibited by $M_{10}'$.

\[ \begin{align*}
1^{a_1} &= A_2' M_{2}' M_{10}' T_{41}' C \\
0^{a_1} &= A_2' M_{10}' C
\end{align*} \]

Recirculation of A and B contents results from:

\[ \begin{align*}
1^{a_{42}} &= A_2 M_{10}' C \\
0^{a_{42}} &= A_2' M_{10}' C \\
1^{a_{41}} &= A_2' M_{10}' C \\
0^{a_{41}} &= A_2 M_{10}' C \\
1^{b_{42}} &= B_2 M_{10}' T_{41}' C \\
0^{b_{42}} &= B_2' M_{10}' C \\
1^{b_{41}} &= B_2' M_{10}' C \\
0^{b_{41}} &= B_2 M_{10}' C
\end{align*} \]

The logic for the addition or subtraction parallels the regular add-subtract logic. $E_0$, initially on, samples signs in $X_{41}$ and $R_{41}$ in conjunction with operation codes:

\[ \begin{align*}
0^{e_0} &= X_{41} R_{41}' M_{10} D_2 U_1 T_1 C + X_{41} R_{41}' M_{10} D_2 U_1 T_1 C \\
&\quad + X_{41} R_{41}' M_{10} D_2 U_1 T_1 C + X_{41} R_{41}' M_{10} D_2 U_1 T_1 C
\end{align*} \]

Just as the A register is cleared to zero during the first word time of $M_0$, so R is cleared to zero during $M_{10}$.
\[ 0\overset{r}{41} = M_{10} C \]

with \( M_{10} \) inhibiting:

\[ 1\overset{r}{41} = M_{9} M_{10}^{B} \overset{42}{1} M_{14}^{C} \]

Note that it is not necessary to restrict the clearing to \( U_1 \) since the result here is written into \( X \) both during \( U_1 \) and \( U_2 \).

\[ 1\overset{x}{40} = S M_{10} C \]
\[ 0\overset{x}{40} = S' M_{10} C \]

If addition is to be performed during \( U_2 \)

\[ 1\overset{e}{0} = K_{a} M_{10} T_{41a} C \]

corresponds to a similar gate during \( M_0 \). The sign is inserted into \( X_{41} \) by:

\[ 1\overset{x}{41} = S M_{10} U_{2} T_{1} C \]
\[ 0\overset{x}{41} = M_{10} U_{1} C \]

\( M_{10} \) is reset at the end of \( U_2 \):

\[ 0\overset{m}{10} = M_{10} U_{2} T_{41a} C \]

From here on, \( X \) must preserve the resulting exponent by recirculation.

\[ 1\overset{x}{41} = X_{M_{9}} M_{10}^{B} M_{10}^{C} \]
\[ 0\overset{x}{41} = X_{1} M_{9} M_{10}^{B} M_{4}^{C} \]

Depending on whether FMP (Code 07) or FDV (Code 05) is indicated, MPR (code 13) or DSR (code 21) is set up.
$1^{m_1} = M_{10} D_2 U_2 T_{41} C$

$1^{m_2} = M_{10} D_2' U_2 T_{41} C$

$1^d_1 = M_{10} C$

By the end of $M\_10$, four word times have elapsed in $I\_3$ (two during $M\_4$ and two during $M\_10$) so that the $Q$ counter now contains a count of four ($Q_3 = 1$). It is necessary to reset the $Q$ counter to zero before starting the multiplication or division process. Thus:

$0^{q_3} = M_{10} T_{41} C$

$U_1$ is again set by:

$1^{u_1} = M_9 M_1' M_2' U_2 T_{41} C$

For multiplication, $E_0$ is again set by:

$1^{e_0} = M_9 M_2' U_2 T_{41} C$

Both multiplication and division are performed in the same manner as for the corresponding fixed-point operations including round-off. However, during the last word time, the exponent in $X$ is transferred into $R$. $B_{42}$ identifies this last word time for both multiplication and division.

$1^{b_{42}} = M_9 M_1 Q_6 Q_3 Q_2 Q_1 T_{41} C + M_9 Q_6 Q_4 Q_1 T_{41} C$

$0^{b_{42}} = M_9 B_{42} T_{41} C$

The gate in $0^{b_{42}}$ not only turns $B_{42}$ off after one word time but also guarantees $B_{42}'$ prior to $M_9 (M_1 + M_2)$. During fixed-point division, $B_{42}$ is used to store the sign of the remainder. However, the remainder is not used in $FDV$ and therefore, $M_9'$ is used to inhibit $B_{42}$:

$1^{b_{42}} = A_40 M_2 M_1' U_1 T_{2} C$
During $B_{42}$, R copies the exponent from X:

$$
1^{r}41 = X_{1}M_{9}M_{10}M_{4}B_{42}C
$$

$$
0^{r}41 = X_{1}M_{9}B_{42}C
$$

$$
1^{r}40 = R_{41}M_{9}C
$$

$$
0^{r}40 = R_{41}M_{9}C
$$

Inhibiting diodes $B_{42}'$ and $M_{9}'$ must be included in the following gates:

$$
1^{r}41 = R_{2}M_{1}B_{42}'U_{1}T_{1}40T_{41a}C + A_{3}M_{1}B_{42}'T_{40}C
$$

$$
+ A_{41}M_{1}B_{42}'T_{41a}C + B_{42}U_{41}M_{9}'T_{C}
$$

$$
0^{r}41 = R_{2}M_{1}B_{42}'U_{1}T_{1}40T_{41a}C + A_{3}M_{1}B_{42}'T_{40}C
$$

$$
+ A_{41}M_{1}B_{42}'T_{41a}C + B_{42}U_{41}M_{9}'T_{C}
$$

$$
1^{r}40 = S U_{41}M_{9}'T_{C}
$$

$$
0^{r}40 = S'U_{41}M_{9}'T_{C}
$$

Upon completion of the multiplication or division process, the floating-point result stands in the A and R registers and it is necessary only to normalize them. Accordingly, $M_{1}$, $M_{2}$, and $M_{9}$ are terminated:

$$
0^{m}1 = M_{1}B_{42}T_{41a}C
$$

$$
0^{m}2 = U_{41}T_{41a}C
$$

$$
0^{m}9 = B_{42}Q_{1}T_{41a}C
$$
\( N_{12} \) is inhibited by \( M_9' \) in:

\[
1^{n}_{12} = M_1 M_9' Q_6 Q_4 T_{40} C + M_9' U_{41} T_{40} C
\]

in order that normalize logic may be activated. \( M_4 \) and \( U_1 \) are established by:

\[
1^{m}_{4} = M_9 B_{42} Q_6 T_{41} C
\]

\[
1^{u}_{1} = M_9 B_{42} Q_6 T_{41a} C
\]

It will be recalled that \( D_3 \) and \( D_1 \) are true at this time. If division was performed, \( D_2' \) existed (code 05) and if multiplication was performed, \( D_2 \) is turned off by:

\[
0^{d}_{2} = M_1 U_{2} C
\]

In either case the computer sees the normalize command code and proceeds to normalize the result standing in the A and R registers. As with the FAD and FSB commands the FNM command logic terminates the FMP and FDV operations.
Both numbers and commands may be filled into any location of the main memory. Numbers are filled as mixed decimal or fractional decimal quantities. Commands are filled in the usual binary-octal "command format."

**Keyboard Procedure for Entering Commands**

**Procedure.**

**Step 1** Energize the machine as per normal operation

**Step 2** The "Ready" light indicates ON condition.

**Step 3** The "Fill Source" light indicates ready condition of console keyboard; if off, press "Fill Source" button.

**Step 4** The "Compute," "Output Error," "Overflow Error," "Verify Error," lights not illuminated; if on, press the "Error Reset" button.

**Step 5** Press the "Location" button. Light indicates the computer is ready to accept information to set the location counter.

**Step 6** Strike in the proper sequence the keyboard buttons corresponding to the desired location to be filled (2 octal digits per channel, 2 octal digits per sector, 1 bit for word half). Corresponding nixie tubes in the control panel readout indicate characters selected on keyboard. If keyboard selection in error, press the "Clear!" button and start from number 6 above.

**Step 7** After the proper information has been entered into the nixie readout, press the "Enter" button. This enters the information into the computer and clears the readout. The "Location Counter" displays the location entered. The location counter is read as illustrated.

**Step 8** After checking the location counter, press the "Command" button (the command light indicates the computer will accept information in command format).
Procedure.

Step 9 Strike in proper sequence the keyboard characters required for the command desired (a sign, two octal command digits, four octal digits, one binary digit-1st half word indicator; a sign, two octal command digits, four octal digits, one binary digit-2nd half word indicator).

Step 10 The nixie readout will display this information (check against desired entry).

Step 11 After a readout has been checked, press the "Enter" button. This enters the command pair into the selected location and clears the readout. The "Enter" button also advances the location counter in such a way that if the next consecutive memory location requires a command no further location entering is needed. Merely start with step 8 and proceed. If any other than the next location is desired, commence again from step 5.

Command Fill Example

Problem: To enter the command pair +0077020 +400010 in Location 1020

Steps 1 - 7 Set location counter to 1020.0

Step 8 "C" key

Step 9 "+" key
0077020 on keyboard
"+" key
400010 on keyboard

Step 10 The panel display at this point would be:

```
0 0 7 7 0 2 0 1 4 0 0 0 0 1 0
```

Step 11 If the display indicates an error, DEPRESS "Clear" key. If display is correct, DEPRESS "Enter" key.
a. Display clears
b. Command pair is put into memory at location 1020
c. The location counter is augmented by "1.0" upon depressing "enter"

Keyboard Procedure for Entering Mixed Numbers

Procedure

Steps 1 - 7  The same steps (1 - 7) as given in "entering commands" are used here.

Step 8  Depress "N" button (the number light indicates number mode enabled)

Step 9  Filling whole number portion

a. Strike sign key "+" or "-"
b. Strike up to 11 whole number digits, most significant digit first.
c. Check nixie display ("Clear" if incorrect).
d. Strike "." (decimal) key.

Step 10  Filling fractional portion

a. Strike up to 11 digits of the fractional portion of the mixed number.
b. Check nixie display ("Clear" if incorrect).
c. Strike "Enter" key.

Example of Mixed Decimal Fill

Problem: To fill +12.75 into memory beginning at location 6010:

Steps 1 - 7  Set location counter to 6010.0

Step 8  "N" key

Step 9  a. "+" key
b. "1" "2" on keyboard
c. At this point the display would be:
d. Strike "." key

1. +12 is filled into location 6010 of memory as:

```
S
Y
S
100000000000000000000000000000001100 C
```

2. Location counter is automatically augmented by "1.0"

---

**Step 10**

a. Depress ".7" then ".5" on keyboard. At this point the display is:

```
1 2 . 7 5
```

b. Strike "Enter" key

1. Display is cleared
2. +.75 is filled into location 6011 of memory as:

```
S
S
1110000000000000000000000000000000000000
```

3. Location counter is automatically augmented by "1.0"

---

**Detailed analysis of Steps 1, 2, 3, 4 of II**

**Step 1**

Energize the machine as per normal operation

With circuit breakers CB1 and CB2 closed and the POWER ON switch on the control unit depressed, a-c power is applied to the POWER ON indicator, blower, and running-time meter in the control unit, and to the full-wave bridge rectifier. The bridge supplies d-c power which energizes relay K1 and K2. Closing of K1 and K2 contacts causes a-c power to be applied to (1) the computer blowers, (2) the power transformer, (3) the memory motor, and (4) to delay relay K3. Relay K3 has a 30-second energizing time. The power supplies and all circuits deriving power from them are energized, with the exception of the main memory write switches. No current can flow to these switches because of K6 contacts 13 and 12 in open position. In addition, the computer cannot be operated because of an initial synchronization delay signal T_d generated as a result of K6 contacts 10 and 9 in open position.
Thirty seconds after power application to relay K3, it is fully energized. Closing of K3 contacts allows the memory solenoid and relay K5 to be energized. Solenoid action positions the memory disk, thereby permitting clock signals to be generated for zeroing of computer circuits. Solenoid and relay K5 current is distributed through a memory thermostat and a power supply thermostat, both of which are set to open at 120 degrees F. The memory thermostat is mounted to the memory housing and the power supply thermostat is mounted on the etched-circuit board of the -18-volt power supply.

Closing of K5 contacts 5 and 6 provides a holding circuit for the memory solenoid when the normally closed K5 contacts 1 and 3 open.

Closing of K5 contacts 12 and 13 permits power to be supplied to K4. This relay has an energizing time of 10 seconds. Zeroing of computer circuitry continues for 10 seconds, at the end of which time relay K4 is energized. Closing of K4 contacts energizes relay K6 and causes the POWER READY indicator to glow. This signifies that the computer is ready for operation, because (1) closing of relay K6 contacts 13 and 12 have applied power to the main memory write selection circuits and (2) closing of K6 contacts 10 and 9 have removed the initial synchronization delay signal.

When power is removed from the computer, the memory solenoid is deactivated, thereby moving the memory disk away from the head-plate. To prevent transients from being written on the disk at this time, power is maintained to the computer secondary power supplies for approximately 0.5 second by the r-c networks across relays K1 and K2.

Step 2 Ready Light On

Step 3 The Fill Source Light On: if Off, Press the Fill Source button

Control panel fill can only be accomplished if the fill source button, on the panel, has been pressed. The signal flow is shown on the following page.
Panel
"Fill Source"
Button

Step 4
The Compute, Output Error, Overflow Error, and Verify Error lights off; if on, press the Error Reset button. The Error Reset button zero-sets, among others, $J_0$ and $J_{13}$

\[
0^0 = T_d
\]

\[
0^1_{13} = T_d
\]

Step 5
Press "Location" button. Light indicates "Locations" mode enabled.

a. Generating $L_B$
The "Location" button generates $L_b$ as follows:
b. $F_0$ and related actions.

The generation of $L_b$ in turn, turns on $F_0$ (Fill location counter flip-flop), and zero-sets $0_0$ (Fill Orders flip-flop) and $N_0$ (fill decimal number flip-flop).

$$f_0 = L_b C_r M_3 T_4 C$$

$$0^0_0 = L_b +$$

$$0^n_0 = L_b +$$

$F_0$, in its one-set state, is involved in the following action:

1. The Input Counter reset flip-flop, $C_r$, is one-set.

$$1^c_r = F_0 T_2 C$$

$C_r$ was initially zero-set by:

$$0^c_r = F_0^0 0 0 N_0 A_n 3 C$$

2. Since the Nixie display is involved in the location counter fill procedure, the H-counter is required. The H-counter is set to 1 (thus skipping the sign position) by:

$$1^h_1 = H_3 H_2 H_1 T_0 R_c 4 FC$$

$H_4, H_3, H_2, H_1, R_c 4$ were previously zero-set by:

$$0^h_1 = U_1 C_r A_0 C$$

(where $l_{u_1} = I_3 C; 0^a_0 = I_4^T_4 + T_0$)

$$0^r_{c4} = R_c 4 T_{40} C$$ (where $0^r_{c1} = T_d$)
3. The input Sequence Counter \((E_3, E_2, E_1)\) is set to "4", \((1 0 0)\). This Counter is involved in separating the octal characters from the half-word indicator. Initially the "E" counter is "001" by:

\[
0^e_3 = C_1' I_2' T_1 C^+; \quad 0^e_2 = C_1' I_2 C^+; \]

\[
\begin{bmatrix}
\begin{array}{c}
e \\
11
\end{array}
\end{bmatrix}
\]

It is set to "100" by:

\[
0^e_1 = F E_3' T_4 0 C
\]

\[
1^e_3 = F E_3 R_1' T_4 0 C
\]

**Step 6** Strike the keyboard buttons corresponding to the desired location to be filled (2 octal digits per channel, 2 octal digits per sector and 1 bit for half-word indicator). Corresponding NIXIE tubes in the control panel readout indicate characters selected on keyboard. If keyboard selection in error, press the "clear" button and start from the beginning of Step 6.

a. TPP, Pm, Pd3 (see timing diagram)

Striking a keyboard button causes TPP to one-set:

\[
1^{TPP} = TPP^{k_1} P^{k_4} P^{k_5} P^{k_4} P^{k_3} P^{k_2} P^{k_1} C
\]

Then Pm is one-set, one word later, by:

\[
1^{P_m} = (TPP)^{J_1} A^{k_1} P^{k_5} P^{k_4} P^{k_3} P^{k_2} P^{k_1} C
\]

When the keyboard button is released, TPP is zero-set:

\[
0^{TPP} = (TPP') (\quad )' (\quad )' \ldots
\]

This, in turn, one-sets Pd3 at T40:

\[
1^{P_{d3}} = (TPP') T_0' P^{k_0} T_4 0 C
\]

Pd3' in its one-set state, terminates Pm:
And finally $p_{d3}$ is zero-set, after it has been on for 1 word time, by:

$$0^p_{d3} = p_{d3} \cdot c$$

b. Filling the Input Register (F5, 4, 3, 2, 1)

1. Keyboard-to-binary code

Depressing a keyboard button generates a 5-bit code as follows:

<table>
<thead>
<tr>
<th>Keyboard Character</th>
<th>K5</th>
<th>K4</th>
<th>K3</th>
<th>K2</th>
<th>K1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>+</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>.</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ENTER</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CLEAR</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

2. Setting $F_5$, $F_4$, $F_3$, $F_2$, $F_1$

During TPP, the F's receive the signals from the Input lines $K_5$, $K_4$, $K_3$, $K_2$ and $K_1$.

$$I_{f_i} = K_i (TPP5) \cdot I_4 \cdot c \quad i = 1-5$$

F's are initially in their zero-set state.

R_{c3} is one-set by $p_m$ at this time.
c. Filling "A" Register (See figure 35)

1. Legitimate Code Flip-Flop ($P_0$) and Fill Orders Flip-Flop ($0_1$)

If the code received from the K lines is a legitimate order address character (that is, if the characters are 0 through 7), $P_0$ one-sets for one word-time by:

\[
\begin{align*}
1P_0 &= F_5 F_4 'P_0 F_C \\
0P_0 &= P_0 E_2' T_{41a} C
\end{align*}
\]

$0_1$ is also set to one for the octal characters:

\[
\begin{align*}
1^{0_1} &= N_0' P_0 E_3 0_1' F_4' + N_0' P_0 E_2 0_1' F_4'
\end{align*}
\]

This, by DeMorgan's Theorem is:

\[
N_0' P_0 0_1' F_4' (E_3' E_2')'
\]

The term $(E_3' E_2')'$ only allows $0_1$ to be one-set at character count of 4, 5, 6 and 7. Therefore, only the channel and sector address bits are considered octal (remember "E" counter started at 100) and excludes the 1/2 word indicator.

$0_1$ is one-set (for each octal character) for one word-time only:

\[
0^{0_1} = 0_1 T_{41a} C
\]

2. Shifting octal characters into "A" Register

a. Shift Diagram

For each of the first 4 characters of "A" Location (Order) counter fill, the shift flow diagram is:
0₁ is one-set during \( T₂ - T₄₁ \) (for octal characters). In one-word time \( (T₁) \) therefore, the shift configuration will have left-shifted 4 bit positions. The bits originally in \( F₃, F₂, \) and \( F₁ \) will then be in \( A₄, A₃, \) and \( A₂ \) respectively. At \( T₂ \) the bits will have moved to \( A₃, A₂, \) and \( A₁ \) since \( A \)'s normal right movement is not inhibited at \( T₁ \).

The left-shifting time \( (T₂ - T₄₁) \) shifts a new octal group from \( F₃, F₂ \) and \( F₁ \) to \( A₄, A₃, A₂ \) respectively while the first character moves to \( A₇, A₆, A₅ \). Again by \( T₂ \) the entire configuration moves right one-bit.

The configuration for the first four characters (octal) are summarized in the following table. The last two steps will be explained in paragraph e.

<table>
<thead>
<tr>
<th>Cell Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A₁₄ ) ( A₁₃ ) ( A₁₂ ) ( A₁₁ ) ( A₁₀ ) ( A₉ ) ( A₈ ) ( A₇ ) ( A₆ ) ( A₅ ) ( A₄ ) ( A₃ ) ( A₂ ) ( A₁ )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T₁ )</td>
<td>( T₂ )</td>
<td>( T₁ )</td>
<td>( T₂ )</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

*See Paragraph e.
The filled information is labeled as follows:

1 2 3, 4 5 6, 7 8 9, 10 11 12, 13

CHANNEL    SECTOR    1/2

b. The logic involved is:

\[
\begin{align*}
f_3 &= A_1 0_1 A_n' C + \\
o_3 &= A_1' 0_1 A_n' C + \\
f_2 &= F_3 0_1 C + \\
o_2 &= F_3' 0_1 C + \\
f_1 &= F_2 0_1 C + \\
o_1 &= F_2' 0_1 C + \\
a_{41} &= F_1 N_0' P_0 T_1' C + \\
o_{41} &= F_1' N_0' P_0 T_1' C + 
\end{align*}
\]

The rest of "A" copies right, inclusive of \( T_1 \).

c. Timing

1. H-Counter

The H-counter, as usual, counts each one-setting of \( N_8 \). \( N_8 \) is controlled by:

\[
\begin{align*}
n_8 &= (TPP) R_{c3} T_{40} C + \\
o_8 &= N_8 C
\end{align*}
\]
2. E Counter

The E-counter counts the number of times $P_0$ is one-set:

\[
\begin{array}{c}
1^e_1 = E_1' P_0 T_{41} C+ \\
0^e_1 = E_1 P_0 T_{41} C+
\end{array}
\]

\[
\begin{array}{c}
1^e_2 = E_2' E_1 P_0 T_{41} C+ \\
0^e_2 = E_2 E_1 P_0 T_{41} C+
\end{array}
\]

\[
\begin{array}{c}
1^e_3 = E_3' E_2 E_1 P_0 T_{41} C+ \\
0^e_3 = E_3 E_2 E_1 P_0 T_{41} C+
\end{array}
\]

d. Clearing F's

The F's are cleared after each character by $Z_{ch}$ (the input Zeroing-keyboard stop flip-flop) employing the following logic:

\[
\begin{array}{c}
0^f_i = Z_{ch} I_3 'C; i = 1 - 5
\end{array}
\]

$Z_{ch}$ is set to one at the same time as the E-counter increases:

\[
\begin{array}{c}
1^z_{ch} = N_0' P_0 T_{41} C+ \\
0^z_{ch} = Z_{ch} I_3 'T 41 'C+ J_13
\end{array}
\]

e. Shifting Half-Word Indicator into A-Register

The E-counter, after the 4 octal characters are filled into A, will be at a count of "000" (since it started as 100). This causes 01 not to be one-set at this time ($1^0_1 = N_0' P_0 F_4' 01' (E_3' E_2')$). Therefore the next character (5th) in the F-register is considered binary.
1. Shift Diagram

Since $0_1' = 1$, for the 5th character, the shifting mechanism gives a 2-bit left shift. The 1/2 word indicator will appear in position $A_2$ at $T_1$ and in position $A_1$ at $T_2$.*

2. The logic involved follows:

\[
\begin{align*}
1_f &= A_1N_0'0_1'P_0T_1'C \\
0_f &= A_1'N_0'0_1'P_0T_1'C \\
1_a &= F_1N_0'P_0T_1'C \\
0_a &= F_1'N_0'P_0T_1'C
\end{align*}
\]

The rest of A copies right, inclusive of $T_1$.

f. Effect of "Clear" button

If at any time during the A-register fill sequence (before "enter" is depressed) the Nixie readout shows that an error has been made, then the "clear" button may be used to clear the readout and allow filling A again.

The "clear" button generates the coded character 11101 (F5, 4, 3, 2, 1). $C_r$ is turned off by this code:

\[
0_r = A_n'F_4F_3P_3T_2C
\]

*See Cell Position Table
C_r' resets the H-counter* to "0001" and the E-counter to "100." R_c4 is also one-set for one word-time:

\[
\begin{align*}
\text{r}_c^r &= T_0' C_r' A'I_4 P_d3 F_5 C \\
\text{r}_c^4 &= R_{c1}' T_{40} C
\end{align*}
\]

R_c4 clears the readout.

The F-register is cleared by the usual gate:

\[
\begin{align*}
\text{i}_f &= Z_{ch 1} I_3 C
\end{align*}
\]

**Step 7** After the proper information has been entered into the NIXIE readout, press the "enter" button. This enters the information into the computer and clears the readout. The location counter displays the next location into which information will be entered (augments one).

a. **General**

The "enter" button causes the following:

1. Turn C_r off at T_2.
2. Reset H-counter and E-counter.
3. Turn R_c4 on to clear readout.
4. Turn I_1 on for one word-time.
5. Transfer A to G.
6. Light location counter control panel neons.

b. **Turn C_r off**

The "enter" code is 11110 (F_5, 4, 3, 2, 1').

C_r is turned off by the same gate as in Step 6-d:

\[
\begin{align*}
0^r_c &= A_n' F_4 F_3 P_d3 T_2 C
\end{align*}
\]

*See Step 5*
c. Reset H-counter and E-counter

Cr' resets H to "0001" and E to "100."

d. Turn Rc4 on

Rc4 is turned one-set (to clear the Nixie display) by:

\[ 1'_{c4} = T_0'C_r'A_0'I_4P_3F_5C \]

e. Turn I1 on

Cr is now turned on by:

\[ 1'_{c} = F_0T_2'C \]

I1 then is one-set for one-word time by:

\[ i_1 = F_5F_4F_3F_2CrP_3d_3N_0'A_n'C** \]

\[ 0_{i1} = I_1F_0T_41aC \]

f. Transfer A to G

N4 is one-set during T2 through T14 during I1 I4

\[ 1^*_{n4} = F_5F_0I_1T_1C \]

\[ 0^*_{n4} = P_5P_4P_3C \]

During N4 the bits passing through A1 are copied by G12 and shifted into the G-counter:

\[ 1^*_{g12} = A_1F_0N_4C+ \]

*See Step 5

**Cr defines, among others, \( F_0 = 1 \)

***See cell position table for initial \( (T_2) \) positions of bits in A.
The location counter is now filled with the keyboard request (with the half-word indicator in \( G_0 \) and the most significant channel bit in \( G_{12} \)).

The control panel neons are driven as follows:

In Computer

\[
\begin{align*}
0^{g_{12}} &= A_1' F_0 N_4 C \\
1^{g_i} &= G(i + 1) N_4 \\
0^{g_i} &= G'(i + 1) N_4 \\
\end{align*}
\]

\( (i = 11 - 1) \)
Filling Commands Steps 8, 9, 10, 11

Step 8  After checking the location counter, press the "Command" button (the command light indicates command mode).

a. $C_b$ Action

The command button generates the signal "$C_b$" as follows:

$$-i2v \rightarrow C_b$$

$C_b$ itself turns $0_0$ (fill orders flip-flop) on (provided $C_r$ is off) and $F_0$ and $N_0$ off:

$$C_r = F \ 0 \ N \ A \ I \ C$$

$$0_0 = C_b \ C_r \ M_3 \ I_4 \ C$$

$$f_0 = C_b$$

$$0^n_0 = C_b$$

b. Resetting the H and E-Counters

$C_r'$ sets the H-counter to "0" and the E-counter to "1."

$$h_i = U_1 C_r' A_0' C$$

$$l_1 = C_r' C$$

$$0^e_2 = C_r' I_2' C$$

$$0^e_3 = C_r' I_2' T_1' C$$
c. Driving the "Command" Neon

The following indicates the control panel neon indicator system:

```
+---+     +---+     +---+
|   | ->  |   | ->  |   |
| IDRIVER | | NE-32 |
|        | | "C"  |
```

Command Fill

Step 9 Strike the keyboard characters required for the command desired (sign, two octal command digits, four octal address digits, one binary digit-first half-word indicator-a sign, two octal command digits, four octal address digits, and one binary digit-2nd half-word indicator.

a. Legitimate Code Flip-Flop (P₀)

Legitimate command characters are 0 through 7, "+" and "-". P₀ will one-set for one-word time for each of these characters only.

```
\[
\begin{array}{cccccc}
1^1P₀ &=& 0_5F_4P_3 & C+0_5F_3F_2P_3 & C  \\
0P₀ &=& N_0P_0T_4 & C+P_0 & E_2T_4 & C
\end{array}
\]
```

The (N₀E₂)' inhibit, of the zero-set logic, does not affect 0₀ operation.

b. Putting Characters into F-Register

The process from keyboard to F-register for commands is the same as that discussed for filling the location counter.

c. Shifting the Sign from F into A

The shift mechanism for the sign is shown on the following page:
0₁ cannot become one-set (for the sign) because of the (E₃'E₂') inhibit term:

\[
\begin{align*}
1₀₁ &= (E₃'E₂')' F₄' 0₁' N₀' P₀ C  \\
0₀₁ &= 0₁ T₄₁a C
\end{align*}
\]

Since T₁' P₀ is 1 during T₂ - T₄₁, the sign bit**, originally in F₁ will be found in A₂ at T₁ after the one word time of shifting.

d. Shifting the First Octal character into "A"

0₁ is one-set (for T₂ through T₄₁) for each of the octal characters

\[
\begin{align*}
1₀₁ &= (E₃'E₂')' F₄' 0₁' N₀' P₀ C  \\
0₀₁ &= 0₁ T₄₁a C
\end{align*}
\]

The shifting mechanism for the octal characters follows:

---

*The E-counter is initially "001"

**Note from the table under Step 6, that F₁ = 1 for "+", F₁ = 0 for "-".
After the shifting time, \( T_2 - T_{41} \), the bits originally in \( F_3, F_2 \), and \( F_1 \) will be in \( A_4, A_3 \), and \( A_2 \). The sign would, at this time, be in \( A_5 \) cell position.

e. Shifting Remaining Characters from \( F \) to \( A \)

As shown in c and d, the E-counter configuration determines the octal and binary shifts (via \( 01 \)). The term \( (E_3'E_2')' \) inhibits \( 01 \) at the configurations \( 001 \), and \( 000 \). These occur at the character counts of 1, 8, 9, and 16. Therefore, the signs and half-word indicators are processed by a binary shift mechanism and the remaining characters are involved in an octal shift.

Step 10  This information will be displayed in the NIXIE readout and should be checked against the desired entry.

The "Clear" button may be depressed if the readout is in error.

Step 11  After the readout has been checked, depress the "Enter" button. This enters the command pair into the selected location and clears the readout. The "Enter" button also advances the location counter.

a. General

The "Enter" button causes the command pair in the A-register to be transferred into the memory location specified by the location counter. The memory search will be done during a forced \( I_1 \) state.

b. Entering \( I_1 I_4 \) State

Depression of the "Enter" button generates the code 11110 (\( F_5, 4, 3, 2, 1' \)). The \( I_1 \) flip-flop is turned one-set by:

\[
\begin{array}{cccccc}
1 & 1 & = & F & 5 & 4 & 3 & 2 & r & d3 & N & 0 & A & 'C
\end{array}
\]
The $I_1$ state persists until the memory location is found and the command pair is written into it:

$$0_{11} = D_{11} T_{40} C$$

c. Setting Channel Register

The channel address characters are loaded into the channel register (via $C_6$) from $G_1$ of the location counter as shown below:

![Diagram of channel register]

$N_4$ is one-set for $T_3$ through $T_{14}$ or $T_{23}$ through $T_{34}$ depending on the state of $G_0$.

$$1_{n4} = I_1 N_9 F_0 ' C^+$$
$$0_{n4} = P_5 P_4 P_3 C^+$$

$$1_{n9} = G_0 T_1 C^+ G_0 ' P_6 ' P_5 P_4 ' P_3 ' P_2 ' C T_2$$
$$0_{n9} = N_9 C$$

The channel register logic is:

$$1_{c6} = G_1 I_1 N_5 C^+$$
$$0_{c6} = G_1 ' I_1 N_5 C^+$$

The rest of the channel register copies right
\[
\begin{align*}
1^c_i &= C_i + 1 N_5 C^+ \\
0^c_i &= C_i + 1 N_5 C^+
\end{align*}
\]

where \( i = 1-5 \)

d. Memory Write Gating Flip-Flop (D\(_{11}\))

D\(_{11}\) is turned one-set for the word-time of writing by:

\[
\begin{align*}
1^d_{11} &= S_{11} T_{40} C \\
0^d_{11} &= D_{11} I_1 T_{40} C
\end{align*}
\]

The \( S_{11} T_{40} \) term indicates that the next sector is the proper sector for writing. \( S_{11} \) (memory write sector selector) remains zero-set for agreement of the least significant 5-sector address bits and disagreement in the 6th which allows for the 180-degree read-write head spacing. \( S_{11} \) thus remains zero-set if the proper sector for writing is about to come under the write head.

The logic for \( S_{11} \) monitors \( S_0 \) (memory read sector selection flip-flop) in making the write sector selection. The one-set logic for \( S_{11} \) is:

\[
S_{11} = S_0 N_1 C + S_0 P_6 P_5 P_4 \quad \begin{cases} T_3 - T_7 & \text{T}_{30} \\
\text{or} \\
T_{23} - T_{28}
\end{cases}
\]

The first gate (\( S_0 N_1 \)) checks for any disagreement in the first five (least significant) bits which are available (inputs) to \( S_0 \) during \( T_3 - T_7 \) (or \( T_{23} - T_{28} \)) and are seen as outputs during \( T_4 - T_8 \) (or \( T_{24} - T_{28} \)).
The second gate (S₀ P₆ P₅ P₄) checks for the required disagreement in the sixth bit. This information is available from S₀ by T₉ (or T₂₉) and is monitored at T₃₀.

The logic for S₀ itself is:

\[
\begin{align*}
1^s_0 &= G_1 S_1 I_1' I_3' N_1 C+ G_1 S_1 I_2' I_3' N_1 C+ \\
0^s_0 &= T_{41} C
\end{align*}
\]

where \( N_1 = 1 \) at \( T_3 - T_8 \) (or \( T_{23} - T_{28} \))

e. Writing on Memory

The information in the A-register (see Step 9) is placed in the X-register before placement in memory. This frees the A-register (while sector search is being done) for processing another input character. *

At the transfer time, I₅ is turned one-set.

\[
\begin{align*}
1^i_5 &= F_0 T_1 I_1 P_{d3} F_4 F_3 F_4' A_n' C \\
0^i_5 &= I_5 T_{41a} C
\end{align*}
\]

During I₅ I₁, the X-register copies the A-register:

\[
\begin{align*}
1^x_{41} &= A_1 I_1 I_5 C \\
0^x_{41} &= A_1' I_1 I_5 C
\end{align*}
\]

The main memory write flip-flops (\( M_{w1} \) and \( M_{w2} \)) copy X₂ during I₁ D₁₁.

*This is useful for photoreader input.

The photoreader can process 400 characters a second (2.5 milliseconds per character). A disk revolution takes 17.4 milliseconds.
f. Advancing the Location Counter

The "Enter" code also advances the location counter. Actually the entrance into writing process \((D_{11} I_1)\) accomplishes the up-count.

The location counter, you will recall, is a one-input counter. The carry flip-flop for this counter \((K_g)\) is set to "1" by:

\[
1^{m}_g = (S_0) D_{11} I_1 T_1 C + T_0' D_{11} I_1 T_1 C
\]

The "1" placed in \(K_g\) will, before the next word time, increase the count held by \(G\)-counter to "\(m + 1.0\)" where "\(m\)" is the location where information was just entered.

Filling Decimal Numbers. The filling of mixed decimal numbers involves the \(R\), \(B\), and \(A\)-registers and also the input-output register \(F\). For example, let us enter the mixed decimal number \(+12.75\) starting at location "\(m\)" in memory.

1. The location counter is set to "\(m\)."
2. Pressing \(N\) sets the computer control logic for number fill.
3. Pressing "+" determines that a binary one will be entered, eventually, into the sign position of \(A\).
4. Pressing the decimal integer "1" will:
   a. Put binary 1 (0001) into the \(F\)'s
   b. Multiply the contents of \(R\) (0000) by ten (1010) and place the product (0000) in \(R\).
   c. Add the \(F\)'s (0001) to \(R\) (via \(A\)) forming the sum "0001" (one) in \(R\)
d. Multiply B (initially "1") by ten thus forming 1010 in B. This step is not required for whole-number fill.
e. Copy R into A while preserving R.

5. Pressing the decimal integer "2" will:
   
a. Put "2" (0010) into the F's
b. Multiply the contents of R (0001) by ten (1010) and place the product (01010) in R.
c. Add the F's (0010)(two) to R (01010), (via A) forming the sum (01100) in R.
d. Multiply B (1010) by ten (1010)² - 1100100. This is not actually required for the whole-number fill.
e. The value in R is transferred to A leaving the binary equivalent of +12 in A.

6. Pressing the "." button will:
   
a. Enter the contents of A (+12) into memory at location "m" (specified in 1.) as +0......011005.*
b. Reset B to "1." The B-register contents were not really required for the whole-number fill.
c. Augment location counter by "1."
d. Clear R.

7. Pressing the fractional decimal numbers "75" will, in like manner to steps 4 and 5, put the binary equivalent of the integer 75 into the A-register and will put the binary equivalent of 10010 in the B-register.

8. Pressing the enter button will cause the A value to be divided by the B value or:

\[
\begin{array}{c|c}
75_{10} & 1001011 \\
10010 & 1100100 \\
\end{array}
\]

\[
.11 = 1100100 / 1001011.00
    110010 0
    0011001 00
    11001 00
\]

That is, 0.11 would be found in the A-register after the conversion and also in the memory location m + 1.0. 0.11, of course, is the

*S is synch bit
binary fraction that is equivalent to a decimal 0.75. After steps 1 through 8 have been accomplished, the mixed number +12.75 would appear in memory as:

Assumed Binary Point

<table>
<thead>
<tr>
<th>Location m</th>
<th>+0000 .... 01100 S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location m + 1</td>
<td>+1100 ........... 0 S</td>
</tr>
</tbody>
</table>

Number Button N. Pressing the N-button generates a -12 signal $N_b$. $N_b$, itself, turns the fill decimal number flip-flop ($N_0$) one-set:

\[ 1^n_0 = N_b C \]

Also, $F_0$ and 0 are turned zero-set:

\[
\begin{align*}
0^n_0 &= N_b C \\
0^0_0 &= N_b C
\end{align*}
\]

In addition, the $H$-counter and $E$-counter are reset to "0000" and "001," respectively.

Sign Digit. Depressing the sign key will:

1. Generate $TPP_k$ which one-sets $TPP$ (see Step 6)
2. $TPP$ sets $P_m$ and $P_{d3}$ (see Step 6)
3. $P_{d3}$ sets $P_0$ for 1 word-time

Legitimate characters, for number fill, are $\emptyset - 9$, +, -, and "."

\[
\begin{align*}
1^P_0 &= N_0 F_5 F'_3 T_{41a} C + N_0 F_5 F'_3 T_{41a} C + N_0 F_5 F'_3 T_{41a} C \\
0^P_0 &= P_0 E_{2'} T_{41a} C
\end{align*}
\]
4. Store sign in $A_{42}$

\[
1_{42}^a = F_3' F_1 N_0 P_0 E_2' E_1', C \text{ (if sign = "+")}
\]

covers

First

character

\[
0_{42}^a = F_3' F_1 N_0 P_0 E_2' E_1', C \text{ (if sign = "-")}
\]

covers

First

character

5. Clear $R$-register

\[
0^r_{41} = N_0 P_0 E_3' E_2' C
\]

\[
0^r_{40} = R_{41'} E_3' U_{41'} C
\]

\[
0^r_{41} = R_{2'} N_0 P_0 E_1' C
\]

6. Set $A_1$ to zero (synch)

\[
0^a_{1} = N_0 P_0 T_{41} E_2' C
\]

7. Set $B$-register to $I 0...0001 Y$

\[
G \quad N
\]

\[
N \quad C
\]

\[
H
\]

\[
1^b_{41} = N_0 P_0 E_3' E_2' T_2 C
\]

\[
0^b_{41} = N_0 P_0 E_3' E_2' P_1' C
\]

covers

$T_1, T_3...$
8. H and E-counters change to "0001" and "010" respectively

\[
\begin{align*}
0^h_4 &= U_1 C^r A^t_0 \\
0^h_3 &= U_1 C^r A^t_0 \\
0^h_2 &= U_1 C^r A^t_0 \\
1^h_1 &= H^r_1 R^r c^t_4 N^t_8
\end{align*}
\]

where \( N_8 \) is set by:

\[
1^n_8 = (T P P)' R^r c^t_3 T^t_4 0 C
\]

\[
\begin{align*}
0^e_3 &= E_3 2 1 0^t_4 1 C \\
1^e_2 &= E_2 E_1 P^t_0 T^t_4 1 C \\
0^e_1 &= E_1 P^t_0 T^t_4 1 a C
\end{align*}
\]

9. F-Register is cleared by \( Z_{ch} \)

\[
\begin{align*}
1^z_{ch} &= P^0 E^t_2 T^t_4 1 C \\
0^f_i &= Z_{ch} P^t_i C & i = 1-5
\end{align*}
\]

First Decimal Character. Depressing the first decimal character \((0 - 9)\) starts the following actions:

1. \( T P P_k \) sets \( T P P \)
2. \( T P P \) sets \( P_m \) and \( P_d^3 \)
3. \( P_d^3 \) sets \( P_0 \) for 3 word-times

\[
\begin{align*}
1^{p_0}_0 &= N F^6_1 F^5_4 P^t_d^3 T^t_4 1 a C + F_5 F^3_3 P_d^3 C + N_0 F^5 F^2_2 F^t_1 P_d^3 C \\
0^{p_0}_0 &= P^0 E^t_2 T^t_4 1 a C
\end{align*}
\]

4. E-counter advanced to "001" by \( P^t_0 T^t_4 1 \)
5. \( E_{20} \) turned on for 1 word-time

\[
1^e_{20} = \begin{array}{cccccccc}
N_0 & F_5 & F_4 & E_2 & P_3 & C+ & N_0 & F_5 & F_3 & E_2 & P_3 & C
\end{array}
\]

same gates as \( P_0 \)

\[
0^e_{20} = E_{20} T_{41a} I_{4} C
\]

6. Multiply \( R \) by Ten during \( E_{20} \)

a. General Approach

Assume that the binary number \( X \) is added to a left shifted (by two bit times) version of itself. We then have:

\[ X + 4X = 5X \]

If the resulting sum (5X) is further left-shifted one bit, we obtain:

\[ 5X + \text{delay} = 10X \]

It is this approach that will be used to generate \( R \) times "ten."

b. Details

The contents of \( R \) are delayed two bit-times (before entering the sum flip-flop, \( S \)) by \( A_2 \) and \( A_1 \). This information is added to the undelayed \( R \) contents and the sum is written back into the \( R \)-register. The last one-bit delay is provided by \( S \).

The following diagram and logic summarizes this action:
7. Adding F to R during E$_{30}$

E$_{30}$ is one-set for 1 word-time following E$_{20}$

---

*E$_{30}$ logic is discussed in operation 7.*
During E_{30}, the F-register (input digit) is added via the A-register to the R-register as follows:

```
<table>
<thead>
<tr>
<th>F_4</th>
<th>F_3</th>
<th>F_2</th>
<th>F_1</th>
</tr>
</thead>
</table>
```

```
E_30
```

```
\begin{align*}
A_1 & \rightarrow S \\
R_{41} & \rightarrow R_{40} \rightarrow R_1
\end{align*}
```

```
E_2
```

```
<table>
<thead>
<tr>
<th>B_{42}</th>
<th>B_{41}</th>
<th>B_2</th>
<th>B_1</th>
</tr>
</thead>
</table>
```

\[ B_1 = 1 \text{ for } E_{30} \]

The logic involved is:

\[ s = A_1 B_1 R_1 K C + A_1 R_1 K' C + A_1' B_1 R_1 K' C + A_1' R_1 K C \]

\[ 0s = A_1' R_1' K C + A_1' B_1 R_1 K C + A_1 R_1 K C + A_1' R_1 K' C \]
Also the F. register is being cleared during $E_{30}$:

$$f^4 = E_{30} C$$

8. Multiplying $B$ by "ten" during $E_{40}$

A similar process, to that described in 6, is used during $E_{40}$ to multiply the contents of $B$ by "ten."

The logic is:

$$1^{a_1} = A_2' N_0' P E_1' T_1' C$$

$$0^{a_1} = A_2' I_3' E_{30}' C + N_0' P E_2' T_41' C$$

(Synch)

$$1^{a_2} = B_1' E_{40} T_{41}' C$$

$$0^{a_2} = B_1' E_{40} T_{41}' C + A_3' E_{40} T_{41} C$$

$$1^{b_41} = S E_{40} C$$

$$0^{b_41} = S' E_{40} T_{41} a' C$$
The logic involved is:

\[
\begin{align*}
1^r_{42} &= R_2 E_{40} C \\
0^r_{42} &= R_2'E_{40} C \\

9. Copying R into A during E_{40}

\text{The logic involved is:}

\[
\begin{align*}
1^a_{41} &= R_{42} E_{40} T_{41} C + A_42 E_{40} T_{41} C \\
0^a_{41} &= R_{42}' E_{40} T_{41} C + A_{42}' E_{40} T_{41} C \\

1^a_2 &= A_3 E_{40} T_{41} C \\
0^a_2 &= A_3'E_{40} T_{41} C \\
0^a_1 &= N_0 P_0 T_{41} E_2 C \\
&\quad \text{(Synch)} \\
1^r_{42} &= R_2 E_{40} C \\
0^r_{42} &= R_2'E_{40} C \\
1^r_{41} &= R_{42} E_{40} C \\
0^r_{41} &= R_{42}' E_{40} C \\
1^r_{40} &= R_{41} E_{30}' U_{41}' C \\
1^r_{40} &= R_{41} E_{30}' U_{41}' C
\end{align*}
\]
10. Other operations of E₄₀

It is unknown (at this time), to the computer, whether or not the number in the A-register is to be filled as a fraction or a whole number. In order to be ready for a possible fractional entry, a "+" sign is forced into the B₄₁ flip-flop as follows:

₂₄₁ = E₄₀ T₄₁ C

At E₂, T₄₁, P₀ is terminated and A, B, and R recirculate

₀P₀ = P₀ E₂ T₄₁ C

The E-counter is reset to "010" at the end of E₄₀:

₀₈₁ = N₀ E₃ E₁ C

₁₈₂ = N₀ E₃ E₁ C

₀₈₃ = N₀ E₃ E₁ C

The computer is now ready for the ".", "Enter" process, or another decimal digit.

**VISUAL READOUT** (D₆ 'D₅ D₄ D₃ 'D₂ D₁ ')

**Displays Available**

The following may be selected for visual display on the control panel NIXIE register:
Contents of:  

<table>
<thead>
<tr>
<th>Register</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-Register</td>
<td>BCD, Command, or Octal</td>
</tr>
<tr>
<td>R-Register</td>
<td>BCD, Command, or Octal</td>
</tr>
<tr>
<td>X-Register</td>
<td>BCD, Command, or Octal</td>
</tr>
<tr>
<td>C-Register</td>
<td>BCD, Command, or Octal</td>
</tr>
<tr>
<td>Main Memory Location (M)</td>
<td>BCD, Command, or Octal</td>
</tr>
<tr>
<td>L - Loop Location (L)</td>
<td>BCD, Command, or Octal</td>
</tr>
<tr>
<td>V - Loop Location (V)</td>
<td>BCD, Command, or Octal</td>
</tr>
</tbody>
</table>

Register Selector Switch

The Register Selector switch, located on the control panel, is an 8-position switch with markings as follows:

![Register Selector Switch Diagram]

This switch alone determines the source of the display for the A, R, X, or C positions. The two M positions and the L or V positions require, in addition, the address of the location to be displayed. The address information is set into the control panel via the two 8-position channel selection rotary switches and/or the two 8-position sector selection rotary switches.
DISPLAY COMMAND (DIS, D6 'D5 D4 D3 D2 D1')

Displays Available

The display command can specify two readout formats: "command" and BCD. If the half-word indicator of the command is a "0", a command format is specified. If the half-word indicator is a "1", a decimal (BCD) format is specified.

The remainder of the "DIS" commands address gives the memory location of the information to be displayed.

Format Details

Command. The command format is as follows:

Decimal. The decimal format is as follows:
Channel and Sector Selector Switches

The figure following is of the channel and sector selector switches:

For the M positions of the register selector switch, all four channel and sector selector switches must be set to determine the required main memory location for display. For the L or V positions of the register selector switch, only the least significant sector selector switch need to be set.

Readout Format Buttons

The three readout format buttons, Decimal, Command, and Octal, determine the manual display format.

Format Details

Command. The Command format is as follows:
Decimal (BCD). The decimal format is as follows:

The BCD scheme is as follows:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>Not Used</td>
</tr>
<tr>
<td>1011</td>
<td>Not Used</td>
</tr>
<tr>
<td>1100</td>
<td>&quot;.&quot;</td>
</tr>
<tr>
<td>1101</td>
<td>&quot;Blank&quot;</td>
</tr>
<tr>
<td>1110</td>
<td>Not Used</td>
</tr>
<tr>
<td>1111</td>
<td>&quot;Terminate&quot;</td>
</tr>
</tbody>
</table>
Octal. The octal format is as follows:

![Octal Format Diagram](image)

Setting the Operation Code Register to $D_6' D_5' D_4' D_3' D_2' D_1'$

Depressing any one of the three readout buttons causes $R_b$ to become true. This, in turn, causes the $R_{c1}$ (readout timing) flip-flop to become one-set if it is $I_4$.

$$r_{c1} = R_b I_4 C$$

The operation code register is forced to the manual readout array $D_6' D_5' D_4' D_3' D_2' D_1'$ by $R_{c1}$ and $I_4$.

$$d_6 = R_{c1} C$$

$$d_5 = I_4 C$$

$$d_4 = R_{c1} C$$
Filling the B-Register with Manual Display Information

From Main Memory or High-Speed Loops

1. The computer monitors the control panel channel selector switches for use in displaying main memory or high-speed loop locations, the control panel channel selection information is placed in the channel register.

The channel register is first cleared of old information by:

\[ 0^d_i = R_c \bar{I}_4 \bar{C} + \]

where \( i = 1-6 \)

The two 8-position channel selector switches each use three contacts (J7, 8, 9 or J10, 11, 12) per position for this purpose. Therefore any arbitrary channel setting will generate six signals. For example if the switches are set to channel "258" then the "J" signal would be:

\[
\begin{align*}
J_{12} & \quad J_{11} & \quad J_{10} & \quad J_9 & \quad J_8 & \quad J_7 \\
0 & 1 & 0 & 1 & 0 & 1
\end{align*}
\]

where: 1 is -12.5 volts and 0 is 0 volt

The state of the \( J \) signals determine the new setting of the channel register as follows:
where \( i = 1-6 \)

2. Monitoring the control panel sector selector switches. In order to display a particular location of the main memory, or the L and V loops, a sector search must be made using, in part, the two sector selector switches.

Before the search can be made the system must go to the \( \text{I}_2 \text{I}_4 \) state:

\[
\begin{align*}
1'_{2} &= R_{c1} R_{b} ' \text{I}_2 ' T_{41a} C' \\
0'_{0} &= E_{3} J_{3} J_{2} J_{1} R_{c1} I_{2} T_{2} C' T_{41} C
\end{align*}
\]

The sector selector switches monitor the prime (or unprime) side of the sector counter flip-flops \( (S_{6}, \ldots, S_{1}) \) dependent on the sector count placed into the two switches. For example, if the sector selector switches (contacts \( J_{6}, 5, 4, 3, 2, 1 \)) specify sector "328" the sector counter array monitored is \( S_{6}', S_{5}, S_{4}', S_{3}', S_{2}, S_{1}' \). When the sector counter reaches the count of 011010 (this array is in the sector counter by \( T_{1} \) of sector 318) each \( J_{6}, 5, 4, 3, 2, 1 \) contact monitors -12.5 volts from the \( S \) outputs. These agreement signals are used to zero-set \( S_{0} \) at \( T_{2} \). If \( S_{0} \) is in its zero-set state by \( I_{2} T_{41} \), sector agreement has taken place and the next sector to pass the main memory read heads is the one requested by the control panel sector selector switches.

\[
\begin{align*}
1'_{0} &= I_{2} R_{c1} T_{1} C' \\
0'_{0} &= E_{3} J_{3} J_{2} J_{1} R_{c1} I_{2} T_{2} C' T_{41} C
\end{align*}
\]

The \( E_{3} \) signal provides the \( J_{6}, J_{5} \) and \( J_{4} \) signals:

\[
\begin{align*}
1'_{e3} &= J_{6} J_{5} J_{4} I_{2} R_{c1} T_{1} C' \\
0'_{e3} &= I_{2} R_{c1} \bar{T}_{41} C'
\end{align*}
\]

For main memory reading, \( D_{0} \) is one-set by:

\[
\begin{align*}
1'_{d0} &= D_{3} S_{3} L_{p1} ' I_{2} T_{41} C' D_{1} S_{1} L_{p1} ' I_{2} T_{41} C'
\end{align*}
\]
For high-speed loop reading, $D_{10}$ is one-set by:

$$d_{10} = S' p1 D_{10} ' I_c D_3 direction 41 C' S' p1 D_{10} ' I_c D_1 direction 41 C$$

3. Gating information into B-register. The B-register copies the main memory read flip-flop ($M_0$) if $D_0$ is one-set or the loop read flip-flop ($K_0$) if $D_{10}$ is one-set:

$$b_{41} = M_0 D_{12} C + K_0 D_{10} I_2 C$$
$$0_b = M_0 D_{12} C + K_0 D_{10} I_2 C$$

$B_{40}$ copies $B_{41}$ etc.

After one word-time, $D_0$ and $D_{10}$ are zero-set by:

$$0_{d0} = D_0 T_{41} C$$
$$0_{d10} = D_{10} T_{41} C$$

... and the state change flip-flop, $N_{11}$, is one-set for one bit time by:

$$1_{n11} = D_0 T_{40} C + D_{10} T_{40} C$$
$$0_{n11} = T_{41a} C$$

The $I_2$ state is then terminated by:

$$0_i = I_2 N_{11} C$$

From One-Word Registers

1. The register selection switch. The one-word registers A, R, C, or X may also be displayed by the proper placement of the register selector. No memory search is required for this display.

2. Gating information into the B-register. For the one-word registers, B copies the register specified by register selection switch signals $A_d$, $R_d$, $Z_d$, or $X_d$. 

214
The I₂ state is terminated by:

\[ 1^b_{41} = A_1 A_d I_2 R_{c1} C^+ X_1 X_d I_2 R_{c1} C \]

\[ + R_1 R_d I_2 R_{c1} C^+ Z_1 Z_d I_2 R_{c1} C \]

\[ 0^b_{41} = A_1' A_d I_2 R_{c1} C^+ X_1' X_d I_2 R_{c1} C^+ R_1' R_d I_2 R_{c1} C \]

\[ + Z_1' Z_d I_2 R_{c1} C \]

The I₂ state is terminated by:

\[ 1^n_{11} = M_d' R_{c1} T_{40} C \] (\( M_d \) is main memory or loop signal from register selector)

\[ 0^n_{11} = T_{41a} C \]

\[ 0^{i2} = I_2 N_{11} C \]

**Summary of I₂ I₄ State**

During the I₂ I₄ state the requested display information is gated into the B-register. After the gating is finished, the I₂ flip-flop is zero-set and the I₃ flip-flop one-set. We thus transfer to the I₃ I₄ state. The following timing diagram summarizes the sequencing of the I₂ I₄ state.
Timing Diagram For $I_2$ $I_4$ State For Manual Readout
Transferring the "B" Register to the "F" Register

General

Now that the requested readout information is in the "B" register, the $I_3 I_4$ state is started by:

$$I_3 = I_2 I_4 N_{11} C$$

During $I_3 I_4$, the information contained in "B" will be shifted character-wise into the output register (F1, 2, 3, 4) in the requested octal, B.C.D., or command format. The format request is made via the control panel "readout" buttons.

Format Control

The format request, from the momentary "readout" format buttons, (or the "DIS" command) is stored in flip-flops $F_{c1}$ and $F_{c2}$. The following table shows the possible arrays before $I_3 I_4$:

<table>
<thead>
<tr>
<th>$F_{c2}$</th>
<th>$F_{c1}$</th>
<th>Format Requested</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Octal ($O_f = 1$)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>B.C.D. ($D_f = 1$)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Command ($C_f = 1$)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Programed Readout in B.C.D. (1/2 word indicator was &quot;1&quot;)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Programed Readout in Command (1/2 word indicator was &quot;0&quot;)</td>
</tr>
</tbody>
</table>

The logic for the manual readout follows:

$$\begin{align*}
1'c1 &= 0_f C \\
0'c1 &= C_f C + I_4' R_{c2} U_1 C \\
1'c2 &= 0_f I_4 C + C_f I_4 C + A_{42}' T_x D_1 C \\
0'c2 &= D_f I_4 C + A_{42} T_x D_1 C
\end{align*}$$
NOTE

A<sub>42</sub> logic is involved in programed readout and B.C.D. manual readout.

Octal Format

"B" and "F" Configuration

In order to shift octal format characters into the "F" register, the following shift mechanism is used:

When T<sub>x</sub> = 0, the "B" register recirculates. When T<sub>x</sub> = 1, two lengthened loops are possible under the control of S<sub>x</sub>. After 41 clocks, the previous contents of "B" will be shifted 1-bit left if T<sub>x</sub> S<sub>x</sub>' = 1 or will be shifted 3-bits left if T<sub>x</sub> S<sub>x</sub>' = 1. The "F" and "B" logic follow:

\[
\begin{align*}
1^f 4 = & \begin{cases} 
\text{not involved} \\
0^f 4 = & \end{cases} \\
1^f 3 = & B_1 A_{42}' S_x' T_x M_2 N_{12}^i C \\
0^f 3 = & B_1 A_{42}' T_x M_2 C+ M_3 N_8 C \\
1^f 2 = & F_3 E_{30}' T_x N_{12}^i C \\
0^f 2 = & F_3 T_x C+ M_3 N_8 C
\end{align*}
\]
The rest of "B" copies left to right.

**NOTE**

A₄₂ is one-set for B.C.D. format only:

<table>
<thead>
<tr>
<th>1₄₂</th>
<th>0₄₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1₄₂</td>
<td>Fₖ₂ D₄ I₄ C</td>
</tr>
<tr>
<td>0₄₂</td>
<td>H₀ I₂ T₁ C</td>
</tr>
</tbody>
</table>

1. General. The requirements for Tₓ and Sₓ (for octal readout) are summarized in the following table:

<table>
<thead>
<tr>
<th>H Counter</th>
<th>Word Time of I₃I₄</th>
<th>Array Required (A₄₂ = 1)</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>T = 0</td>
<td>Recirculate &quot;B&quot;</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Sₓ Tₓ = 1</td>
<td>To shift sign bit into F₁</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Tₓ = 0</td>
<td>Recirculate &quot;B&quot; while sign is copied into sign NEON</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>Sₓ Tₓ = 1</td>
<td>Shift first octal character into F₃, F₂, F₁</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Tₓ = 0</td>
<td>Recirculate &quot;B&quot; while first octal character is copied into most significant NIXIE</td>
</tr>
<tr>
<td>H Counter</td>
<td>Word Time of I₃I₄</td>
<td>Array Required (A₄₂ = 1)</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------</td>
<td>--------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>Sₓ'Tₓ = 1</td>
<td>Shift second octal character into F₃, F₂, F₁</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tₓ = 0</td>
<td>Etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Etc.</td>
</tr>
<tr>
<td>13</td>
<td>28</td>
<td>Sₓ'Tₓ = 1</td>
<td>Shift last octal character into F₃, F₂, F₁</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>Tₓ = 0</td>
<td>Recirculate &quot;B&quot; while 13th octal bit is copied into 13th (left to right) NIXIE. 14 and 15 are blank.</td>
</tr>
<tr>
<td>14</td>
<td>30</td>
<td>Sₓ'Tₓ = 1</td>
<td>Shift garbage into F₃, F₂, F₁</td>
</tr>
</tbody>
</table>

2. "H" Counter (readout control flip-flop's H₄', 3', 2', 1')

The number of characters shifted through the output register is stored in the "H" counter. For octal format, 14 characters are displayed.

The counter determines the character count by summing the N₈ flip-flop outputs. N₈ is dependent on the R₉ flip-flop which, in turn, monitors Tₓ. These operations follow.

The "H" counter is an 8, 4, 2, 1 binary counter. The logic is listed below:

\[
\begin{align*}
1^h₁ &= H₁₁ R_{c₄}^1 N₈ C \\
0^h₁ &= H₁ N₈ C + U₁ C₉^1 A₀^1 C \\
1^h₂ &= H₂ H₁ N₈ C \\
0^h₂ &= H₂ H₁ N₈ C + U₁ C₉^1 A₀^1 C
\end{align*}
\]
The following table shows the counting process:

<table>
<thead>
<tr>
<th>H4</th>
<th>H3</th>
<th>H2</th>
<th>H1</th>
<th>Cleared by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>U1 Cx'A0' (T1 of U1)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>First N8 (T41 of U3)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Second N8 (T41 of U5)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Third N8 (T41 of U7)</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Rc3 Terminates Counting</td>
</tr>
</tbody>
</table>

The timing chart on the following page shows the interrelation of the Tx, Rc3, N8, Sx, N12 and I3 flip-flops:
Timing Diagram For $I_3 \cdot I_4$ State Of "Octal" Manual Readout
Figure 35. Filling the first Octal Character into "A"
(The other characters are handled similarly)
Command Format

"B" and "F" Configuration

The same loops used in the octal readout are used for command readout with, however, differences in logic of $S_x$ and $N_{12}$ that provide the variance in format. (See "B" and "F" configuration)

1. General. The following table lists the requirements for $S_x$ and $T_x$ to provide command format:

<table>
<thead>
<tr>
<th>&quot;H&quot; Counter</th>
<th>Word-Time of $I_3$</th>
<th>Array Required $(A_{42} = 1)$</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$T_x = 0$</td>
<td>Recirculate &quot;B&quot;</td>
</tr>
<tr>
<td>2</td>
<td>$A_{42} S_x T_x = 1$</td>
<td></td>
<td>Shift Sign into $F_1$</td>
</tr>
<tr>
<td>3</td>
<td>$T_x = 0$</td>
<td></td>
<td>Copy sign into sign neon</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>$A_{42} S_x T_x = 1$</td>
<td>Shift first octal group into $F$'s</td>
</tr>
<tr>
<td>5</td>
<td>$T_x = 0$</td>
<td></td>
<td>Copy first octal group into NIXIE</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>$A_{42} S_x T_x = 1$</td>
<td>Shift second octal group into $F$'s</td>
</tr>
<tr>
<td>7</td>
<td>$T_x = 0$</td>
<td></td>
<td>Copy second octal group into NIXIE</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>$A_{42} S_x T_x = 1$</td>
<td>Shift third octal group into $F$'s</td>
</tr>
<tr>
<td>9</td>
<td>$T_x = 0$</td>
<td></td>
<td>Copy third octal group into NIXIE</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>$A_{42} S_x T_x = 1$</td>
<td>Shift fourth octal group into $F$'s</td>
</tr>
<tr>
<td>11</td>
<td>$T_x = 0$</td>
<td></td>
<td>Copy fourth octal group into NIXIE</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>$A_{42} S_x T_x = 1$</td>
<td>Shift fifth octal group into $F$'s</td>
</tr>
<tr>
<td>'H' Counter</td>
<td>Word-Time of I₃</td>
<td>Array Required (A₄₂₁ = 1)</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>---------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>13</td>
<td>Tₓ = 0</td>
<td></td>
<td>Copy fifth octal group into NIXIE</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>A₄₂₁ Sₓ₁ Tₓ = 1</td>
<td>Shift sixth octal group into F's</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>Sₓ₁ Tₓ = 1</td>
<td>Shift 1/2 word indicator into F's</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>Tₓ = 0</td>
<td>Copy 1/2 word indicator into NIXIE</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>Sₓ₁ Tₓ = 1</td>
<td>Shift sign of second command into F's</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>Tₓ = 0</td>
<td>Copy sign into NIXIE</td>
</tr>
<tr>
<td>15</td>
<td>31</td>
<td>Sₓ₁ Tₓ = 1</td>
<td>Shift 1/2 word indicator into F's</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>Tₓ = 0</td>
<td>Copy 1/2 word indicator into NIXIE</td>
</tr>
<tr>
<td>16</td>
<td>33</td>
<td>Sₓ₁ Tₓ = 1</td>
<td>Shift garbage into F's</td>
</tr>
</tbody>
</table>


The following timing chart shows the interrelation of the Tₓ, Rₓ₃, N₈, Rₓ₂, Sₓ, and N₁₂ flip-flops. Note that only the logic for Sₓ and N₁₂ is changed from that used in the octal format case.

\[
\begin{align*}
S_x &= H_3 H_2 H_1' N_8 F_{c2} F_{c1}' C + I \bar{C} \\
0^{S_x} &= H_2' H_1' N_8 C \\
1^{N12} &= T_x R_{c2}' H_4' T_{40} D_{1}' C \\
0^{N12} &= T_{41a} C
\end{align*}
\]
Timing Diagram For $I_3$ $I_4$ State of Command Format Manual Readout
Binary-Coded-Decimal (BCD) Format

"B" and "F" Configuration

The loops used for B.C.D. format are as follows:

\[ A_{42} S'_{x} T_{x} \]

\[ S_{x} T_{x} \]

\[ F_{1} F_{2} F_{3} F_{4} \]

\[ T_{x}' \]

\[ T_{x} \]

NOTE

A_{42} is one-set only for B.C.D.,

\[ l^{a}_{42} = F_{c2'} D_{4} I_{4} C \]

1. General. The table on the following page lists the requirements for \( S_{x} \) and \( T_{x} \) to provide B.C.D. readout.
<table>
<thead>
<tr>
<th>&quot;H&quot; Counter</th>
<th>Word-Time of I₃</th>
<th>Array Required (A₄₂ = 1)</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Tₓ = 0</td>
<td>Recirculate &quot;B&quot;</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Sₓ Tₓ = 1</td>
<td>Shift sign into F₁</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Tₓ = 0</td>
<td>Copy sign into NEON</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>Sₓ' Tₓ = 1</td>
<td>Shift 4 bits (B.C.D. character) into F's</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Tₓ = 0</td>
<td>Copy character into NIXIE</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>Sₓ' Tₓ = 1</td>
<td>Shift last B.C.D. to &quot;F&quot; and one-set &quot;B&quot; register</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>Sₓ' Tₓ = 1</td>
<td>Copy into NIXIE</td>
</tr>
</tbody>
</table>

2. "H" Counter and Timing Diagram

a. General. The "H" counter operates in an identical manner to that of "octal" or "command" readout. See Octal Format-2. "H" counter.

The following Timing Chart shows the interrelation of Tₓ, Rₓ, N₈, N₁₂, Rₓ₁, Rₓ₂ and I₃ flip-flops.

b. Details of manual B.C.D. readout termination. In order to terminate the display after 9 characters (counting sign), the terminate code (1111) is internally forced to appear as the last (tenth) character shifted into the "F" register. This is accomplished as follows:

B₄₂ is one-set; 1ₜ₄₂ = \[
\begin{array}{c}
\text{R}_{c₁} \\
\text{I₄}
\end{array}
\]

and is not allowed to zero-set during I₄; 0ₜ₄₂ = I₄' R₁' Rₓ₂.

B₄₁ is allowed (for B.C.D. only) to copy B₄₂ after the eighth character

\[
\begin{array}{c}
\text{1ₜ₄₁ = B₄₂ A₄₂ Rₓ₂ H₄ Tₓ C} \\
\text{0ₜ₄₁ = B₄₂' A₄₂ Rₓ₂ H₄ Tₓ C}
\end{array}
\]
$I_3$ State For PROGRAMED B.C.D. Readout ("DIS")
After one word-time of B’s shifting, the "1" copied into B₁₄₁ has one-set the entire B-register. The ninth character was shifted into F₄, F₃, F₂, F₁ during this word-time. The ninth character is read out during the next word-time (9th R₉₃ pedestal). During the last word-time of I₃, the terminate code (111X) is shifted into the "F" register. The code is monitored by N₁₂ two clocks before, i.e., T₄₀, the shift is completed. At this time, the leading three coding bits are in F₄, B₂, and B₁. Thus, N₁₂ is one-set by:

\[ \text{l}^{n}_{12} = F_4 \quad B_2 \quad B_1 \quad T_x \quad T_{40} \quad C \]

in turn, I₃ is terminated by N₁₂ as follows:

\[ 0^3_3 = N_{12} \quad C \]

I₃ may also be terminated earlier by an actual (not forced by logic) terminate character (see table on previous page).

It must be emphasized that the internal binary array in locations "m" and "m+1" had to be previously processed by a programed routine in order to provide a useful readout. One does not readout the true decimal equivalent of a register(s) location(s) contents without first converting the contents via a routine. This is also true for manual readout.

By the use of the "Decimal Point" code, the "Blank" code and the "Terminate" code, several numbers may be displayed at the same time by means of judicious programing.

**SETTING THE CODE OPERATION REGISTER TO D₆ D₅ D₄ D₃ D₂ D₁’**

The operation code register is set during I₂ by the "DIS" command in exactly the same manner as in any command. See 9.1 of the "Description of the Logical Design of the RECOMP II."

**FILLING THE B AND R REGISTERS**

Filling the "B" Register with Information Specified by "m" of "DIS (m)"

The display command, "DIS", is an I₂, Type-3 and is handled, during I₂, in essentially the normal manner. See 9.2.2.2 of "Description of the Logical Design of the RECOMP II." The Timing Diagram on page 230 gives the special readout timing.
Filling the "R" Register with the "m + 1" Information for BCD Readout

One characteristic difference between manual and programmed readout is the use of two memory locations for programmed B.C.D. readout. During I₃ the second word is placed in the "R" register by:

\[ r'_{41} = M_r p l r\' D_6 D_4 I_3 I_4 D' U_1 C \]
\[ + K_0 L p l D_6 D_4 I_3 I_4 D' U_1 C \]

\[ r'_{41} = M_r p l D_6 D_4 I_3 I_4 D' U_1 C \]
\[ + K_0 L p l D_6 D_4 I_3 I_4 D' U_1 C \]

The remainder of "R" shifts so that at I₃ U₂ T₁ the "R" register has the contents of location "m + 1".

Transferring the "B" Register to the "F" Register

General

Now that the requested readout information is in the "B" (and "R" for B.C.D.) register, the I₃ state is started by:

\[ i_3 = I_2 N_1 I_1 C \]

During I₃, the information contained in "B" (and "R" for B.C.D.) will be shifted character-wise into the output register (F₄ 3 2 1) in the format requested by the command's 1/2-word indicator bit; 1 for B.C.D., 0 for command.

Format Control

The format request is stored in the Fc₁ and Fc₂ flip flops as follows:

<table>
<thead>
<tr>
<th>Fc₂</th>
<th>Fc₁</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>B.C.D.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Command</td>
</tr>
</tbody>
</table>
The logic involved follows:

\[
\begin{align*}
0^{c_1} &= A_{42} T \cdot D_1 \cdot C_1^+ \cdot I_4 \cdot R_{c_2} U, C \\
1^{c_2} &= A_{42} T_1 \cdot D_1 \cdot C \\
0^{c_2} &= A_{42} T_1 \cdot D_1 \\
1^{a_{42}} &= Z_1 I_2 I_4^+ N_9 C \\
0^{a_{42}} &= H I_2 T_1 C
\end{align*}
\]

The Z\(_1\) flip-flop at T\(_2\) (or T\(_{22}\)) of I\(_2\) contains the half-word indicator of the second command (or first). If the indicator is a "1", A\(_{42}\) is set to one and, in turn, F\(_1\) and F\(_4\) are zero-set. If the indicator is zero, A\(_{42}\) is left in its zero-set state.

**Command Format**

During I\(_3\), the logic and operation for programmed command readout is the same as for manual readout.

**BCD Format**

**B, F and R configuration**

During I\(_3\), sixteen characters (including sign) are read out. Nine originate in location "m" and seven in location "m + 1".

The loops used for programmed B.C.D. format are as follows:
The exact logic required, to mechanize the above diagram, follows:

\[
1^{r}_{41} = M_{r} L_{r} D_{r} D_{4} I_{3} I_{4} D_{1} U C + K_{0} L_{p} D_{p} D_{4} I_{3} I_{4} D_{1} U C + R_{1} M_{3} D_{2} U C
\]

\[
0^{r}_{41} = M' L_{r} D_{r} D_{4} I_{3} I_{4} D_{1} U C + K' L_{p} D_{p} D_{4} I_{3} I_{4} D_{1} U C + R_{1}' M_{3} D_{2} U C
\]

\[
1^{b}_{42} = R_{h} R_{c} N_{12} C
\]

\[
0^{b}_{42} = R_{1}' I_{4} R_{c} C
\]

\[
1^{b}_{41} = B_{1} T_{x} M_{0} M_{8} M_{10} I_{3} F_{1} H_{4} T_{x} F_{1} R_{c} T_{x} C + A_{42} R_{c} H_{4} T_{x} C
\]

\[
0^{b}_{41} = B_{1}' T_{x} M_{0} M_{8} M_{10} I_{3} F_{1} H_{4} T_{x} F_{1} R_{c} T_{x} C + A_{42}' R_{c} H_{4} T_{x} C
\]

\[
1^{f}_{4} = B_{1} A_{42} M_{2} N_{12} S_{x} T_{x} C
\]

\[
0^{f}_{4} = B_{1}' A_{42} T_{x} C
\]

\[
1^{f}_{1} = B_{1} S_{x} T_{x} N_{12} C F_{2} S_{x} T_{x} N_{12} C
\]

\[
0^{f}_{1} = B_{1}' S_{x} T_{x} C F_{2}' S_{x} T_{x} C
\]
1. General. The following table lists the requirements for $S_x$ and $T_x$ to provide B.C.D. programed readout.

<table>
<thead>
<tr>
<th>&quot;H&quot; Counter</th>
<th>Word-Time of $I_3$</th>
<th>Array Required ($A_{42} = 1$)</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$T_x = 0$</td>
<td>Recirculate &quot;B&quot;</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>$S_x T_x = 1$</td>
<td>Shift sign into $F_1$</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>$T_x = 0$</td>
<td>Copy sign into NEON</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>$S_x T_x = 1$</td>
<td>Shift 4 bits into $F$'s</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>$T_x = 0$</td>
<td>Copy character into NIXIE</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>$S_x T_x = 1$</td>
<td>Shift last B.C.D. character (of &quot;m&quot;) into $F$'s while &quot;B&quot; accepts &quot;R's&quot; &quot;m + 1&quot; information via $B_{42}$</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>$T_x = 0$</td>
<td>Copy last character (of &quot;m&quot;) into NIXIE</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>$S_x T_x = 1$</td>
<td>Shift first B.C.D. character (of &quot;m + 1&quot;) into $F$'s. (sign was lost in $B_{42}$)</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>$T_x = 0$</td>
<td>Copy last B.C.D. character into NIXIE</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
<td>$S_x T_x = 0$</td>
<td>Shift garbage into $F$'s</td>
</tr>
</tbody>
</table>

The timing chart on the following page shows the interrelation of $R_{c1}$, $R_{c2}$, $R_{c3}$, $S_x$, $T_x$, $N_8$, $N_{12}$ during $I_3$ for programed B.C.D.
PUNCH AND TYPEWRITER OUTPUT

Punch Character (PNC), Type Character (TYC), Punch and Type Character (PTC) Commands

General

The one-character output commands all are similar in operation, therefore, they will be discussed as a unit.

"PNC" (D6 D5 D4 D3 D2' D1'). A single character is punched in a 5-bit teletype code corresponding to the least significant five address bits (not counting 1/2 word indicator).

"TYC" (D6 D5 D4 D3' D2 D1'). The "Type Character" command substitutes typing for punching. The actual typed character depends on whether "LTRS." shift or "FIGS." shift has been previously called for.

"PTC" (D6 D5 D4 D3 D2 D1'). The "Punch and Type Character" command causes both punching and typing to occur.

"PNC" Operation

Setting M7. After I1, the command pair is in the Z-register and during I2 the operation code is placed in the operation code register (D6 D5 D4 D3 D2 D1).

\[ 1 \text{m7} = J \cdot D_6 D_5 D_4 I_2 N_{11} C \]

M7 is then set by:

\[ 0 \text{m7} = \text{TPP X p 10} \]

M7 is zero-set while the actual punching is being done.

Z and F Configuration. The flow of the code bits from Z to F during word 1 of I3, is shown on the following page:
The logic involved follows:

\[
\begin{align*}
& f_5 = Z_1 N_6 C \\
& 0'5 = Z_1' N_6 C \\
& f_i = F_i + 1 N_6 C \\
& 0'i = F_i' + 1 N_6 C \\
\end{align*}
\]

Where \( i = 1-4 \)

\( N_6 \) determines bit-times \( T_3 - T_7 \) (or \( T_{23} - T_{27} \)) which are the required code bit positions of the "PNC" command.

\[
\begin{align*}
& 1'n_6 = M_7 D_6 D_1' U_1 N_9 C \\
& 0'n_6 = P_2 P_1 C \\
\end{align*}
\]

"Punch" and "Tape Punch Pulse" Flip-Flops. The punch flip-flop one-sets by:

\[
1'x_9 = M_7 D_3 X_10' Q_5 (TPP)' T_{41} C
\]

\( Q_5 T_{41} \) determines a delay of 16 word-times in \( I_3 \).

The term \( (TPP)' \) inhibits \( X_9 \) while the Tape-Punch-Pulse flip-flop is one-set. TPP is one-set (for punching) by the closure of a contact \( (TPP_p) \) during the actual mechanical punching motion.

\[
1TPP = (TPP_p) X_{10}' C
\]

\( X_{10}' \) is necessary for the "PTC" command.
TPP is zero-set at the termination of the punch motion by the TPP
contact opening.

\[ 0^{TPP} = (TPP_{rm})' (TPP_{k})' (TPP_{t})' (TPP_{p})' (TPP_{rp})' [S_1 S_2 S_3 S_4] \]

The other terms were already in their prime states. The definitions of these terms follow:

- \( TPP_{rm} \) Tape punch pulse for mechanical reader
- \( TPP_{k} \) Tape punch pulse for keyboard
- \( TPP_{t} \) Tape punch pulse for typewriter
- \( TPP_{p} \) Tape punch pulse for punch
- \( TPP_{rp} \) Tape punch pulse for photo-reader
- \( S_{u1} \) "Set up" button #1
- \( S_{u2} \) "Set up" button #2
- \( S_{u3} \) "Set up" button #3
- \( S_{c} \) "Start Compute" button

Delay Flip-Flops \( P_{m} \) and \( P_{d3} \). Delay flip-flops will be as follows:

1. \( P_{m} \): The \( P_{m} \) flip-flop resets the Q-counter by forcing a zero into Q_6 at \( T_2 \).

\[
0^{Q_6} = P_{m} C
\]

\[
1^{P_{m}} = (TPP) J_{13}' A_n' P_5' P_4' P_3' P_2' P_1
\]

\( P_5' P_4' P_3' P_2' P_1 \) determines \( T_2 \)

\( J_{13}' \) indicates no ECHO check error

TPP indicates punch is in motion

\[
0^{P_{m}} = P_{d3} C + J_{13} C
\]
2. \textbf{P}_{d3}: The \textbf{P}_{d3} flip-flop indicates termination of the command and clears the output register (F's).

\begin{align*}
1^{p}_{d3} &= (\text{TPP})' T_0' P_m T_{40} C \\
0^{p}_{d3} &= P_{d3} T_{40} C \\
0^5 &= P_{d3} T_{40} C \\
0^4 &= D_5 P_1' P_{d3} T_{40} C \\
0^3 &= D_5 P_2' P_{d3} T_{40} C \\
0^2 &= D_5 P_3' P_{d3} T_{40} C \\
0^1 &= D_5 P_4' P_{d3} T_{40} C
\end{align*}

Where \( T_0' \triangleq \) Tape Reader Not on

Where \( P_0' P_{d3} \) limits coverage to non-fill use of \( P_{d3} \)

3. \textbf{ECHO Checking}: The output error flip-flop (\( J_{13} \)) will one-set and cause its light to flash if the requested punching does not take place. ECHO Checking contacts are on each punch. The contact signals (E's) are compared with the original code in F by \( J_{13} \).

\begin{align*}
1^{j}_{13} &= F_1 E_1' T_c+F_1 E_1 T_c+F_2 E_2' T_c+F_2 E_2 T_c+F_3 E_3' T_c + F_3' E_3 T_c+F_4 E_4' T_c+F_4 E_4 T_c+F_5 E_5' T_c+F_5 E_5 T_c \\
0^{j}_{13} &= T_{d3} C
\end{align*}

"TYC" Operation

Setting \( M_7 \). The same one-set logic is used as for "PNC". See "PNC" operation.

The zero-set logic is:

\begin{align*}
0^m_7 &= T_{bs} J_{13} C
\end{align*}

where "\( T_t \)" is the type ECHO timing pulse
Z and F Configuration. See "PNC" operation.

"Type" and "Tape Punch Pulse" Flip-Flops. \( X_{10} \) indicates a typing operation. Its logic follows:

\[
\begin{align*}
1^x_{10} &= M_7 D_2 D_1' Q_5 (\text{TPP})_t T_{41} C \\
0^x_{10} &= (\text{TPP}_t) C^+ T_d C
\end{align*}
\]

As for \( X_9 \), the \( Q_5 \) term allows a delay (in \( I_3 \)) of 16 word-times; the \( \text{TPP}_t \) signal is true during the actual typing motion. The logic for the \( \text{TPP} \) signal is:

\[
\begin{align*}
1_{\text{TPP}} &= \text{TPP}_t X_9 ' I_4 ' C \\
0_{\text{TPP}} &= \text{same as for punching. See "PNC" operation.}
\end{align*}
\]

Delay flip-flops \( P_m \) and \( P_{d3} \). See "PNC" operation.

**Echo Checking.** The error checking logic for typing is:

\[
\begin{align*}
1^j_{13} &= F_1 E_1' T C^+ F_2 F_4 K_1 T C F_2 E_4 T C F_2 K_2 T C F_3 E_4' T C \\
0^j_{13} &= T_d C^+ S_{ul} C
\end{align*}
\]

The \( E_t \)'s are the typing ECHO checking normally closed contacts.

The typing ECHO checking normally open contacts (\( K_k \)'s) receive their nomenclature from their use during keyboard or typewriter fill state. \( T_t \) is the type ECHO timing pulse. The zero-set logic for \( J_{13} \) is:

\[
\begin{align*}
0^j_{13} &= T_d C^+ S_{ul} C
\end{align*}
\]

**"PTC" Operation**

Setting \( M_7 \). The one-set logic for \( M_7 \) is identical to that of "TYC" or "PNC." See "PNC" operation.

\[
\begin{align*}
0^m_{7} &= T_{bs} J_{13} ' C + \text{TPP}_X p_{10} ' C
\end{align*}
\]
Z and F Configuration. See "PNC" operation.

"Punch," "Type," and "Tape Punch Pulse" Flip-Flops. For the "PTC" command TPP is controlled by the slower of the two output devices. The sequence, if the punch is faster than the typewriter, is:

1. $X_9$ and $X_{10}$ are one-set

\[
\begin{align*}
X_9 &= M_7 D_3 X_{10}' Q_5 (TPP)' T_{41} C \\
X_{10} &= M_7 D_2 D_1' Q_5 (TPP)' T_{41} C
\end{align*}
\]

2. TPP goes true

3. $X_9$ zero-sets

\[
0X_9 = (TPP_p) C
\]

4. TPP goes true

5. $X_{10}$ zero-sets

\[
0X_{10} = (TPP_t) C
\]

6. TPP flip-flop one-sets

\[
1TPP = (TPP_t) X_9' T_{41} C
\]

7. TPP goes false

8. TPP goes false

9. TPP zero-sets

\[
0TPP = (TPP_p)' (TPP_t)'
\]

Delay Flip-Flops, $P_m$ and $P_d3$. See "PNC" operation.

Echo Checking. Errors in either typing or punching will cause the output error light to flash and the computer to stop.
Timing of "PNC" Command

\[ x_9 = m_7 d_2 d_1 q_5 (TPP_p) t_{41} \]

\[ 0 x_9 = (TPP_p) + t_d \]

\[ x_{10} = m_7 d_2 d_1 q_5 (TPP_p) t_{41} \]

\[ 0 x_{10} = (TPP_p) + t_d \]

\[ P_m = (TPP_p) j_{13} a_{2} p_{2} p_{3} p_{4} + \]

\[ 0 P_m = p_{33} + t_{41} \]

\[ P_{d3} = (TPP_p) t_{0} p_{33} t_{40} + \]

\[ 0 P_{d3} = p_{33} t_{40} + \]

\[ TPP = (TPP_p) x_{10} + (TPP_p) x_9 t_4 + \]

\[ctp = (TPP_p) (TPP_p) (TPP_p) (TPP_p) (TPP_p) \]

\[ (TPP_p) \]

\[ -12V \]
Timing of "TYC" Command

\[ x_9 = M_2 D_3 D_1 Q_5 (T_{PP})' T_{41} \]
\[ 0 x_9 = (T_{PP_p}) + T_d \]

\[ x_{10} = M_7 D_2 D_5 Q_5 (T_{PP})' T_{41} \]
\[ 0 x_{10} = (T_{PP_b}) + T_d \]

\[ P_m = (T_{PP}) J_{13} A_n^p f_p p' p'' p''' \]
\[ 0 P_m = P_d + J_{13} \]

\[ F_{d3} = (T_{PP}) F_{0m T_{40}} \]
\[ 0 F_{d3} = P_{d3} T_{40} \]

\[ T_{PP_b} \]
\[ Bounce \]

\[ T_{PP_b} \]
\[ Bounce \]

\[ T_{PP} \]
\[ Bounce \]

\[ T_{PP} \]
\[ (T_{PP_p})' x_{10} + (T_{PP_b}) x_{21} \]

\[ 0 T_{PP} = (T_{PP_p})' (T_{PP_b})' (\ldots')' (\ldots')' (\ldots')' \]

\[ -12V \]
Timing of "TYC" Command

\[ T_{TP} = (t_{TP})(t_{TP}) + \ldots + (t_{TP}) \]

\[ T_{o} = (t_{TP}) + T_{d} \]
PUNCH WORD (PNW), TYPE WORD (TYW), AND PUNCH AND TYPE WORD (PTW) COMMANDS

General

In order to provide compatibility between the 8, 4, 2, 1 binary internal code and the teletype code used to drive the punch and typewriter, the output register must have binary-to-teletype conversion logic.

In other respects, the PNW, TYW, and PTW commands are similar to the Display Command ("DIS"). An "ECHO" check is made on each character.

Punch Word, "PNW" (D₆' D₅' D₄ D₃ D₂' D₁'). The punch word command, "PNW (m)" punches the requested information in the format specified by the half-word indicator, 1 for BCD, 0 for Command.

1. Command Format. Sixteen 5-bit characters are punched to represent the usual command format (sign, 6-octal, 1-binary, 1-binary, 6-octal, 1-binary).

2. Decimal (BCD) Format. The partial contents of memory positions, "m" and "m + 1" are punched out in BCD if the command half-word indicator is "1."

Up to sixteen 5-bit characters may be punched; 1 binary, and 15 BCD. The information from "m" and "m + 1" is apportioned in the same manner as for the "DIS" command (a leading "+" sign will not be punched; a leading "-" sign will be punched).

Type Word, "TYW" (Dₒ' D₅' D₄ D₃ D₂' D₁')

The same formats as above are available for typewriting.

Punch and Type Word, "PTW" (Dₒ' D₅' D₄ D₃ D₂' D₁')

The same formats as above are available for punching and typing

"PNW" Command

I₁ State. As for any command, the command pair is gated into the Z register during I₁.
I₂ State. The "PNW" command is an I₂, Type 3. Therefore, the address "m" of the command is located during I₂ and the contents of "m" are gated into the B-register. The operation code bits are also placed into the operation code register (D₆₋₁) during I₂.

I₃ State. During I₃ U₁ (for BCD) the contents of "m + 1" are gated into R. Also during I₃, the information to be punched out is shifted, character by character into the F-register (very much the same as for the "DIS" command). However, a binary-to-teletype code conversion must be accomplished on each character (after it is in the F-register) before the punch may be allowed to monitor F. This added step was not required for the Display command.

1. I₃ and N₁₂ Flip-Flops. I₃ is turned one-set by the usual logic:

\[ i'_3 = J_0 I'_2 N_{11} C \]

After the command is completed, I₃ is zero-set by:

\[ 0^i_3 = N_{12} C \]

...where N₁₂'s logic is:

\[
\begin{array}{c|c}
1^{n}_{12} & T \times R \times H \times T_{40} D_{1} D_{5}^{+} F_{4} B_{1} T \times T_{40} C \\
\hline
\text{Command Format} & \text{BCD Format} \\
0^{n}_{12} & T_{41a}
\end{array}
\]

2. M₇ Flip-Flop. In order to punch, type, or punch and type a word, the basic character operation is repeated a number of times dependent on the format.

M₇ is cycled for each character as follows:

\[
1^{m}_{7} = J_{0} D_{6}^{+} D_{5} D_{4} D_{1} I_{2} N_{11} C^{+} D_{5}^{+} T \times T_{41a} N_{12}^{+} C
\]

\[
\text{Initial One-Setting} \quad \text{Subsequent One-Settings}
\]
After the punching, typing, or punching and typing actually begins, the M7 flip-flop zero-sets.

\[ 0^m7 = T_{bs} 13'C+ TPP_p X_{10} '+ \]

"TYW" or "PNW" or "PTW" or "PTW"

3. A-Register. During I_3 of the PNW Command, the A-register recirculates.

\[
\begin{align*}
1^a41 &= A_1 M_0 \ M_1 \ M_2 \ M_3 \ M_4 \ M_{10} \ P_0 \ C \\
0^a41 &= A_1' M_0 \ M_1 \ M_2 \ M_3 \ M_4 \ M_{10} \ P_0 \ C
\end{align*}
\]

4. Shifting B into F. Command Format is as follows:

R recirculates by:

\[
\begin{align*}
1^r41 &= R_1 M_0 \ M_1 \ M_2 \ M_3 \ M_4 \ M_9 \ I_3 \ C \\
0^r41 &= R_1' M_0 \ M_1 \ M_2 \ M_3 \ M_4 \ M_9 \ I_3 \ C
\end{align*}
\]

B_41's logic is:

\[
\begin{align*}
1^b41 &= B_1 T_x M_0 \ M_8 \ M_{10} \ I_3 C+ F_1 R_{c2} T C+ F_1 H_4 T_C x \\
0^b41 &= B_1' T_x M_0 \ M_8 \ M_{10} \ I_3 C+ F_1' R_{c2} T C+ F_1' H_4 T_C x
\end{align*}
\]
The F-Register logic is:

| 1 f₁ = F₂ S'T N₁₂'C+ B₁ S'T N₁₂'C |
| 0 f₁ = F₂ S'T C+ B₁ S'T C+ D₅ P₃ T₄₁C |
| 1 f₂ = F₃ E₃₀'T N₁₂'C |
| 0 f₂ = F₃ T C+ D₅ P₃ T₄₁C |
| 1 f₃ = B₁ A₄₂'S'T M₂ N₁₂'C |
| 0 f₃ = B₁ A₄₂'T M₂ C+ D₅ P₃ T₄₁C |
| 1 f₄ = Not involved in shifting |
| 0 f₄ = Not involved in shifting |

A₄₂'s logic is:

| 1 a₄₂ = Won't one-set for "command" format |
| 0 a₄₂ = H₀'I₂ TₑC |

Rₖ₂'s logic is:

| 1 Rₖ₂ = M₇ D₆' U₁ C |
| 0 Rₖ₂ = A₄₂'H₃ H₂ H₁ N₈ C+ I₃'C |
The logic involved is:

\[ l_{41}^r = M_r L_1 D_6 D_4 I_3 I_4 D_1 U C + K_0 L_1 D_6 D_4 I_3 I_4 D_1 U C \]

\[ + R_1 M_0 M_1 M_2 M_3 M_4 M_9 I_3 C \]
5. Timing Logic. The following are timing logic operations:

a. H-Counter: The H-Counter is initially cleared by:

$$h_i = U_1 C' A' +$$

$$0_i = U_1 C' A'$$

i = 1-4

The H-Counter here will keep track of the number of times T_x becomes one-set (via N_8).
PNW, or TYW Commands (See "Display Command" passout for format timing)
b. \( S_x \): \( S_x \) is one-set by:

\[
S_x = I_2 \bar{C} + H_3 H_2 \bar{H}_{18} F_{2 c} F_{c1}
\]

Command Format

\( S_x \) will be one-set during the times that binary characters are being shifted into the F-register.

\( S_x \) is Zero-set by:

\[
0_x = H_2 \bar{H}_{18} F_{c1}
\]

c. \( T_x \): \( T_x \) is one-set by:

\[
1_x = M_7 D_6 U_{14} a C + D_5 \bar{P}_{d3} T_{14} a C
\]

Initially Subsequent Characters

...and zero-set after 1 word-time by:

\[
0_x = M_2 T_x T_{14} a C
\]

6. Sign Conversion. The following are sign conversion operations:

Command Format. In command format the sign (either "+" or "-") is a legitimate output character. Before conversion, the F-register will contain one of the following:

<table>
<thead>
<tr>
<th></th>
<th>( F_1 )</th>
<th>( F_2 )</th>
<th>( F_3 )</th>
<th>( F_4 )</th>
<th>( F_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The conversion should derive one of the following:

<table>
<thead>
<tr>
<th></th>
<th>( F_1 )</th>
<th>( F_2 )</th>
<th>( F_3 )</th>
<th>( F_4 )</th>
<th>( F_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
For the "+" conversion, only the F5 flip-flop must contain added logic. The following logic of F5 covers the sign position.

\[ f_5 = F_2' F_1 N_8 C \]

N8 here is a conversion timing flip-flop:

\[ l_8^n = M_2' D_5 T x T_41 N_12' C \]
\[ 0_8^n = N_8 C \]

For the "-" conversion, the bits in F2 and F1 require modification.

\[ f_2 = F_3' F_2' F_1' D_5' N_8 C + F_4' F_2' F_1' D_5' N_8 C \]
\[ f_1 = D_5' S x H_1' N_8 C \]

For BCD Format. In BCD, the sign is legitimate only if it is "-". "+" signs are not punched out. This allows the programmer the ability to program a BCD output longer than 15 characters without the usually meaningless inclusion of sign characters at the start of each word of output.

The "+" sign is used to generate a "FIGS" shift code and, therefore, negates the need of a separate "FIGS" shift before the "+PNW" command.

Before conversion, the "F" register contains one of the following:

<table>
<thead>
<tr>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
</tr>
</thead>
</table>
| 1  | 0  | 0  | 0  | 0  | for "+"
| 0  | 0  | 0  | 0  | 0  | for "-"
The conversion should derive one of the following:

<table>
<thead>
<tr>
<th>F₁</th>
<th>F₂</th>
<th>F₃</th>
<th>F₄</th>
<th>F₅</th>
</tr>
</thead>
</table>
| 1  | 1  | 0  | 1  | 1  | for "+"
| 1  | 1  | 0  | 0  | 0  | for "-"

The logic required for the "+" to "FIGS" conversion is:

\[ f_4 = A_{42} S F_1 N_{8a} C \]

- BCD Sign Same as \( N_8 \)
- Position is +

\[ f_2 = A_{42} S F_1 N_{8a} C \]

\[ f_1 = D_5 S H_1 N_8 C \]

Same gate as for "command"

The "-" conversion is identical to that for "command" format:

\[ f_2 = F_3 F_2 F_1 D_5 N_8 + F_4 F_2 F_1 D_5 N_8 C \]

\[ f_1 = D_5 S H_1 N_8 C \]
**"PNW", "TYW", and "PTW" Output Conversion**

**Binary-to-Teletype**

<table>
<thead>
<tr>
<th>Character (In &quot;FIGS&quot; Shift)</th>
<th>Binary (Before Conversion)</th>
<th>Teletype (After Conversion)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( F_5 ) ( F_4 ) ( F_3 ) ( F_2 ) ( F_1 )</td>
<td>( F_5 ) ( F_4 ) ( F_3 ) ( F_2 ) ( F_1 )</td>
</tr>
<tr>
<td>Ø</td>
<td>X 0 0 0 0</td>
<td>1 0 1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>X 0 0 0 1</td>
<td>1 0 1 1 1</td>
</tr>
<tr>
<td>2</td>
<td>X 0 0 1 0</td>
<td>1 0 0 1 1</td>
</tr>
<tr>
<td>3</td>
<td>X 0 0 1 1</td>
<td>0 0 0 0 1</td>
</tr>
<tr>
<td>4</td>
<td>X 0 1 0 0</td>
<td>0 1 0 1 0</td>
</tr>
<tr>
<td>5</td>
<td>X 0 1 0 1</td>
<td>1 0 0 0 0</td>
</tr>
<tr>
<td>6</td>
<td>X 0 1 1 0</td>
<td>1 0 1 0 1</td>
</tr>
<tr>
<td>7</td>
<td>X 0 1 1 1</td>
<td>0 0 1 1 1</td>
</tr>
<tr>
<td>8</td>
<td>X 1 0 0 0</td>
<td>0 0 1 1 0</td>
</tr>
<tr>
<td>9</td>
<td>X 1 0 0 1</td>
<td>1 1 0 0 0</td>
</tr>
<tr>
<td>&quot;.&quot;</td>
<td>X 1 1 0 0</td>
<td>1 1 1 0 0</td>
</tr>
<tr>
<td>Carriage Return</td>
<td>X 1 0 1 1</td>
<td>0 1 0 0 0</td>
</tr>
<tr>
<td>&quot;+&quot;</td>
<td>X 0 0 0 1</td>
<td>*1 1 0 1 1 (BCD)</td>
</tr>
<tr>
<td>&quot;-&quot;</td>
<td>X 0 0 0 0</td>
<td>**1 0 0 0 1 (Command)</td>
</tr>
<tr>
<td>&quot;Space&quot;</td>
<td>X 1 0 1 0</td>
<td>0 0 1 1 1</td>
</tr>
<tr>
<td>No Output</td>
<td>X 1 1 1 1</td>
<td>Causes Termination of ( I_3 )</td>
</tr>
</tbody>
</table>

\( X \) Will always be "0" initially and will not be affected by the shifting operations of \( F \)

* Causes "FIGS" shift

** Causes "+" shift
7. More Code Conversion. Code conversion examples follow:

The Problem

Of course the sign is not the only character that requires conversion for the "PNW", "TYW," and "PTW" commands.

Each character on the figure on the preceding page needs conversion. To derive the required logic, recourse to Veitch diagram techniques will be made.

$F_1$ Logic

First a four-variable Veitch diagram is constructed locating each character by its binary code:

Secondly, another four-variable Veitch is constructed using the following rules of procedure:

1. Place a 1' for each character (as defined on the last Veitch diagram) that requires $F_1$ to be one-set when going from binary to teletype coding.
2. Place a "1" for each character that may be one-set in going from binary to teletype coding.
3. Place a 0' for each character that requires $F_1$ to be zero-set.
4. Place a "0" for each character that $F_1$ may be zero-set.
The total conversion logic is shown below with the logic covering the primed 1's and 0's underlined.

\[ f_1 = F'_4 F'_2 N_8 D'_5 C^* + D'_5 H'_1 S N_8 C \]

for sign conversion

The zero-set logic is:

\[ f_1 = F'_3 F'_2 N_8 C + F'_4 N_8 C + D'_5 P d_3 T_4 C^* N_12 C \]

Clearing

**F₂ Logic**

The same process, as in "b", is used for F₂ logic:

\[ f_1 = F'_4 F'_2 N_8 C^* + D'_5 H'_1 S N_8 C \]

*N₈ (and N₈ₐ) here is a conversion timing flip-flop for outputs*
\[ f_2 = F' F' N D' H C + F' F' N D' S' C \]

\[
(H' S_x)' \text{ excludes sign positions}
\]

\[ + F' F' F' N D' C + F' F' F' N D' C \]

\[ f_2 = F' F' F' N D' C + F' F' F' N D' C + F' F' N D' C \]

\[
+ D' P d 3 T 4 C + N 12 C
\]

Clearing

\[
F_3 \text{ Logic}
\]

\[
\begin{array}{cccc}
0 & 0 & 1' & 0 \\
1 & 1 & & 0' \\
0' & 1 & 0' & \\
1 & 0 & 1' & 1' \\
\end{array}
\]

\[ f_3 = F' F' F' N D' H C + F' F' F' N D' S' C + F' F' F' N D' S' C \]

\[
(H' S_x)' \text{ excludes sign positions}
\]

\[ f_3 = F' F' F' N C + D' P d 3 T 4 C \]

Clear
The F5 flip-flop is, for the "PNW", "TYW", and "PTW" commands, in its zero-set state at the beginning of I3 and remains so (untouched by octal, binary, or B.C.D. shifts from "B" to "F") before conversion. During binary-to-teletype conversion, F5 may be set to 1 dependent on the character. Therefore, the following Veitch diagram assumes that F5 is "0" at the start of the conversion.
\[ f_5 = F'^4 F'^3 F'^2 N_{H, G} F'^4 F'^3 F'^2 N_{S, C} \]

\[(H'S_x)^' \text{ Excludes signs} \]

\[ + F'^4 F'^3 F'^2 N_{C} + F'^4 F'^3 F'^2 N_{C} + F'^3 N_{G} C \]

\[ f_5 = P_{d3} T_{40} C + D'_{p} T_{41} C + N_{12} C \]

*Clearing*
SYSTEM CHECKOUT PROCEDURES

GENERAL

Many procedures can be devised for checking or troubleshooting a computing system. But selecting the appropriate procedure can save much time in diagnosing a malfunction and isolating it to the faulty component.

The procedures given in this section, when used wherever possible in conjunction with test routines, will enable detection and isolation of most system malfunctions relatively quickly and easily. Procedures for correcting the malfunctions, if not presented in this section, are given in the Computer Service Manual and Input-Output Service Manual.

COMPUTER

Careful analysis of the indications of trouble will save time in isolating a computer malfunction, particularly when more than one unit or function is affected. At times an error in the program or routines may be causing the difficulty, though the computer may appear to be at fault. In addition to thorough analysis of the indications of trouble, the proper troubleshooting sequence may also save considerable time. Generally, however, the first steps will include checking the voltage levels and the controls, and running appropriate test routines.

Procedures for checking the voltage levels are given in the Computer Service Manual, for initially testing operation of the controls and running test routines in this and section , respectively, of this manual. These procedures should at least isolate the malfunction to a particular unit.

If the malfunction is in the computer or circuit portions of the other units, most rapid isolation can usually be achieved by using the system tester. This equipment enables flip-flop checks, static and dynamic checks of logic, set-reset control of flip-flops, and application of marginal conditions to voltage levels and clock.

Before making any circuit tests, however, the person performing them should familiarize himself with the circuit characteristics and functions by studying the logic data-flow diagrams and accompanying logic charts in this manual, and the circuit schematic and wire listings.
in the System Reference Schematics. For example, if the computer fails to execute a command, reference to the diagrams and charts mentioned will aid in isolating the difficulty by enabling testing of the circuitry used by that command. When making tests, the following precautions should be observed:

**CAUTION**

Boards containing transistors should never be pulled out while power is on. (Logic boards may be removed with power on.) Power to the computer should never be removed except with the switch at the control console. Metal objects, such as a screwdriver, should never be used on boards or wiring while power is on. The computer should never be moved while the memory disk is rotating; allow at least 15 minutes from power turnoff for complete memory deceleration before moving computer.

The following paragraphs outline checkout and troubleshooting procedures on the computer, some using the system tester as indicated (a functional description of the system tester and procedures for operating it are presented in the Test Equipment Service Manual).

**Power Turn-On Check**

1. Connect variac into outlet and adjust its output to 115 volts as monitored on voltmeter.
2. Connect computer power cable (attached to desk) to variac.
3. Depress Power On-Off switch on control console to On position and note the time elapsed until the Ready indicator lights (Ready indicator should light in 40 ± 10 seconds; Power indicator should light immediately upon depressing Power button).
4. Recheck line voltage with voltmeter to reassured that line voltage is 115 volts.

**Power Supply Checks**

The following procedures ascertain if power supply voltages are correct and tolerances are within percentage limits.

1. Attach digital voltmeter and note power supply voltages are monitored at System Tester test points (see table following for normal power supply levels and Computer Service Manual for adjustment procedures).
2. Check voltage percent levels at power supply voltmeter on rear of control console by turning voltmeter range switch to each of the positions and observing the reading at each position. Levels must read 100 ± 5 percent and the -100 voltage which should read within ±10 percent; percent voltage levels may be adjusted at the appropriate potentiometer settings on the control panel network board as outlined below:

Table 9. Power Supply Voltages

<table>
<thead>
<tr>
<th>Level</th>
<th>Value</th>
<th>Adjustable</th>
<th>Tolerance (plus/minus)</th>
<th>Ripple p-p (less than)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-18</td>
<td>-18</td>
<td>yes</td>
<td>2.5%</td>
<td>90 mv</td>
</tr>
<tr>
<td>-12</td>
<td>-12</td>
<td>yes</td>
<td>2.5%</td>
<td>60 mv</td>
</tr>
<tr>
<td>-6</td>
<td>-6</td>
<td>yes</td>
<td>2.5%</td>
<td>90 mv</td>
</tr>
<tr>
<td>+6</td>
<td>+6</td>
<td>yes</td>
<td>2.5%</td>
<td>30 mv</td>
</tr>
<tr>
<td>+75</td>
<td>+75</td>
<td>yes</td>
<td>2.5%</td>
<td>400 mv</td>
</tr>
<tr>
<td>-3</td>
<td>-2.5</td>
<td>no</td>
<td>0.5V</td>
<td></td>
</tr>
<tr>
<td>+.75</td>
<td>+.75</td>
<td>no</td>
<td>0.15V</td>
<td></td>
</tr>
<tr>
<td>-3B</td>
<td>-2.5</td>
<td>no</td>
<td>0.5V</td>
<td></td>
</tr>
<tr>
<td>+75R</td>
<td>+35</td>
<td>no</td>
<td>10.0V</td>
<td></td>
</tr>
<tr>
<td>-100</td>
<td>-100</td>
<td>no</td>
<td>10.0V</td>
<td>3.5 V</td>
</tr>
</tbody>
</table>

Figure 36. Control Panel Network Board
Location of Potentiometers On Control Panel Network Board

<table>
<thead>
<tr>
<th>Adjustment</th>
<th>Voltage</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>R 33</td>
<td>Line Trans</td>
<td>High Limit</td>
</tr>
<tr>
<td>R 34</td>
<td>+50</td>
<td>Adjusted before adjustment of R 47</td>
</tr>
<tr>
<td>R 35</td>
<td>-12</td>
<td></td>
</tr>
<tr>
<td>R 36</td>
<td>±6</td>
<td>Adjust for mean value</td>
</tr>
<tr>
<td>R 37</td>
<td>+.75</td>
<td></td>
</tr>
<tr>
<td>R 38</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>R 39</td>
<td>-18</td>
<td></td>
</tr>
<tr>
<td>R 40</td>
<td>-100</td>
<td>Adjustment dependent on voltage level</td>
</tr>
<tr>
<td>R 42</td>
<td>Line Trans</td>
<td>Low Limit</td>
</tr>
<tr>
<td>R 47</td>
<td>+75</td>
<td>Adjusted after adjustment of R 34</td>
</tr>
</tbody>
</table>

3. Check power supply ripple values with oscilloscope at System Tester test points, with the System Tester Mode switch in static position (see Normal Power Supply Table 8 for proper values).

4. Adjust variac until system is supplied with 108 volts ac as monitored on voltmeter.

5. Turn control pots on control panel network board (R33 and R42) counterclockwise all the way (see figure 36 for location of pots).

6. Depress Error Reset button on control console (Line Transient indicator on control console should extinguish).

7. Slowly turn R42 pot adjustment clockwise until Line Transient indicator lights.

8. Check voltmeter on rear of control console (reading should be ±3 percent of values given above).

9. Adjust variac up to 125 volts ac as monitored on voltmeter.

10. Slowly turn R 33 pot adjustment clockwise until Line Transient indicator lights.

11. Check voltmeter on rear of control console (reading should be ±3 percent of values given in table 8).

12. Readjust variac to 115 volts ac.
Static Computer Flip-Flop Check

This procedure checks for faulty flip-flops and whether they will toggle individually. The procedure can be varied in troubleshooting to the extent that only those flip-flops associated with the suspected malfunction, as indicated by the logic, need be individually one-and zero-set.

1. If not done previously, check to see if voltage levels are correct.
2. Connect System Tester and turn Marginal Test switch to Off position and Mode switch to Static position.
3. If not already done, apply power to computer at control console.
4. Place One-Set-Zero-Set-All switch in One-Set position and observe if all neon indicators on System Tester light (if a neon does not light, assuring that the neon isn't malfunctioning, replace the flip-flop concerned and repeat this step; if the flip-flop is not at fault check the Set lines within the computer).
5. Depress the One-Shot Clock button and observe if all neons remain lighted (if a neon extinguishes replace the flip-flop concerned and repeat this step).
6. Place All switch in Zero-Set position and observe if all neon indicators extinguish (if a neon remains lighted, replace the flip-flop concerned and repeat this step; if the flip-flop is not at fault check the Set lines within the computer).
7. Depress the One-Shot Clock button and observe if any neons light (if a neon lights, replace the flip-flop concerned).
8. Place Individual One-Set-Zero-Set switch to l-Preset position.
9. Depress microswitch of each flip-flop (term) on System Tester, beginning with A42 and ending with Zch and observe if all flip-flops come true by noting if all neon indicators light (indicator should remain lighted); if a neon extinguishes it indicates one of three possible conditions:

   a. The flip-flop has 0-set itself, indicating a faulty board.
   b. Another flip-flop has triggered the flip-flop indicated by the neon that extinguished, in which case, if (a) is eliminated as the cause set all flip-flops to 0-set state, then 1-set the flip-flop concerned and again 1-set each flip-flop individually, meanwhile observing when the neon for the flip-flop concerned extinguishes. When this neon extinguishes check for the possibility of an improper interconnection between the flip-flops concerned.
   c. The neon has burned out.

(Procedures for removing circuit boards are given in the Computer Service Manual).
10. After all neon s remain lighted from performing Steps 8 and 9, place individual switch to 0-Preset position.

11. Depress microswitch of each flip-flop on System Tester, beginning with A42 and ending with Zch; and observe if all flip-flops go false by noting if its neon indicator extinguishes (indicator should remain off); if a neon lights, it indicates one of three possible conditions:

   a. The flip-flop has 1-set itself, indicating a faulty board.
   b. Another flip-flop has triggered the flip-flop that lighted, in which case, if (a) is eliminated as the cause set all flip-flops to 1-set state, then 0-set the flip-flop concerned and again 0-set each flip-flop individually, meanwhile observing when the neon for the flip-flop concerned lights. When this neon lights, check for the possibility of an improper interconnection between the flip-flops concerned.

   (Procedures for removing circuit boards are given in the Computer Service Manual).

12. Upon completion of Static Computer Flip-Flop Check, turn off power to computer at control console.

One Shot of Logic Check

This check can be used to detect and correct major wiring errors and broken wires. The procedure is:

1. Insert specially wired logic board and attach meter.
2. Zero-set all flip-flops with System tester and set up logical proposition to be tested using System tester with $S_1$ in static position (observe if gate goes true as last term in proposition is set).
3. Check if each term can cause output to go false if it is not as stated in proposition under test.
4. After all of proposition has been checked, ohm out wiring between gate and set inputs to the flip-flop.

Example: one-shot $1^m = J_0 'D_6 D_5 'D_4 'I_2 N_11$ 120-9

Zero-set all

Turn $D_6$ on, (true) $J_0$ is already off (false)
Turn $I_0'$ on, $D_5 D_4$ are already off (false)
Turn $N_2$ on, gate should go true.
Turn $J_0'$ on, gate goes false.
Turn J₀ off, gate goes true.
Turn D₆ off, gate goes false.
Turn D₆ on, gate goes true.
Turn D₅ on, gate goes false.
Turn D₅ off, gate goes true.
Turn D₄ on, gate goes false.
Turn D₄ off, gate goes true.
Turn I₂ off, gate goes false.
Turn I₂ on, gate goes true.
Turn N₁₁ off, gate goes false.
Turn N₁₁ on, gate goes true.

Gate is now checked.

Manual Input-Output (Console Keyboard and Visual Readout) Check

Procedures for checking proper operation of the console keyboard and visual readout are given later in this section.

Clock Signal Check

CAUTION

The computer should not be moved while the memory disk is rotating; allow at least 15 minutes from power turnoff for complete memory deceleration before moving computer.

NOTE

See figure 40 for location of computer connectors.

To Check Clock Track Preamplifier Signal

1. Monitor preamplifier signal with oscilloscope at connector J525-11 of computer or through monitor box plugged into J525 (amplitude should be 4.5 ± 0.5 volts peak-to-peak).
2. If necessary, adjust at potentiometer on preamplifier plugged into connectors J439 and J539 of computer.

To Check Computer Clock Duty Cycle

1. Measure clock signal output with oscilloscope at J339-36 and J339-37 (final signal should be as shown in figure 37 or from computer...
manual; necessary adjustment should be made according to procedures in the Computer Service Manual).

Figure 37. Clock Signal Waveform

(1) 1.75 ±0.15 usec (measured at 10% point of wave shape)
(2) 4.62 ±0.5 usec (measured at 10% point of wave shape)
(3) Rise Time: less than 0.5 usec
(4) Fall Time: less than 2 usec
(5) Amplitude: 0 ±0.5V to -6.5 ±0.5V
(6) Jitter: 0.5 usec

Origin Pulse Check

CAUTION

The computer should never be moved while the memory disk is rotating; allow at least 15 minutes from point of turnoff for complete memory deceleration before moving computer.

NOTE

To check origin pulse amplitude:

1. Monitor pulse with oscilloscope at connector J525-10 of computer or through monitor box plugged into J525 (origin pulse occurs once every disk revolution; amplitude should be 4.5 ±0.5 volts peak-to-peak).

2. If necessary, adjust potentiometer on preamplifier board plugged into connectors J438 and J538 of computer.
To check origin pulse shape:

1. With oscilloscope, check origin waveshape as monitored at System Tester X flip-flop output (origin waveshape is available once every disk revolution; wave shape should be as shown in figure 38).
   2. If wave shape does not conform to standards given in figure below, memory should be returned to factory for adjustment.

To check origin pulse for noise and crosstalk:

Check for noise and crosstalk (in combination) on base line of origin track on oscilloscope (should be less than 500 millivolts).

(1) Pulse Width at 10 percent of magnitude must be less than 12 usec.
(2) Amplitude, excluding strobe, 4.5 ±0.5 volts.
(3) Pulse Width of Origin Wave shape: 6.37 usec ±0.65 usec (1 clock time measured at top of waveshape).

Figure 38. Origin Pulse and Waveshape
Sector Counter Signal Check

This procedure checks proper counting of the Sector Counter.

1. Connect System Tester.
2. Place Static-Dynamic switch in Static position.
3. Place One-Set–Zero-Set-All switch in 0-set position (check that all neon indicators are extinguished).
4. Set up any sector configuration (except 008 or 778) on flip-flops S6 through S0 by depressing individual microswitches to achieve binary indication of desired sector.
5. One-set G0 and T1 flip-flops (respective indicators should light).
6. Depress One-Shot Clock button (K9 and N9 indicators should light).
7. Depress One-Shot Clock button (N1 indicator should light).
8. Depress One-Shot Clock button six times (indicators S6 through S1 should light and the preceding one extinguish in succession).
9. Depress One-Shot Clock button (N1 indicator should extinguish and sector counter should advance one to next sector from that set up in step 4).

Digit Counter Check

This procedure checks correct counting of the Digit Counter.

1. Connect System Tester.
2. Place Static-Dynamic switch in Static position.
3. Place One-Set–Zero-Set-All switch in 0-set position (check that all neon indicators are extinguished).
4. Depress One-Shot Clock button 41 times; indicators should light as follows: T1 indicator lights on first depression of One-Shot Clock button; then digit (bit) counter indicators P6 through P1 should show binary representation of the decimal quantity signifying the number of times the One-Shot Clock button has been depressed through the 39th time; then T40 indicator lights on the 40th depression of the One-Shot Clock button; then both T41 indicators light on the 41st depression of the One-Shot Clock button.

Memory Signal Check

The following procedures enable checks of amplitude, pulsewidth, noise and crosstalk of all memory channels (additional procedures are given in the Computer Service Manual).
CAUTION

The computer should never be moved while the memory disk is rotating, allow at least 15 minutes from power turnoff for complete memory deceleration before moving computer.

NOTE

See figure 40 for location of connectors.

1. Set location counter to 0000.0.
2. Fill all of channel 00 with the pattern +2514340-7407601.
3. Connect test point at D₀ to ground at System Tester.
4. Select readout location 0077 with Channel and Sector Selector switches on control console.
5. Depress Command Readout button on control console.
6. Observe amplitude with oscilloscope at Connector J550-1 of computer (amplitude excluding strobe should be 4.5 ±0.5 volts average peak-to-peak).
7. If necessary, adjust amplitude at read switch plugged into J342 by adjusting the potentiometers (counting downward from the top beginning with 0).
8. Check pulse width at 10 percent of its magnitude (width should be less than 12 microseconds).
9. Disconnect ground from D₀ flip-flop and fill all of channel 00 with -000000000000000.
10. Reconnect D₀ to ground and select readout location 0077 as in Step 4.
11. Observe noise and crosstalk (in combination) with oscilloscope (should be less than 500 millivolts).
12. Disconnect ground from D₀.
13. Enter and verify Fast Memory Fill and One-Channel Zero routine with tape reader (see Test Routine section).
14. Place Sense Switch D up.
15. Depress Start 1 button (memory will load with +2514340-7407601 pattern; routine will halt automatically).
16. Reconnect D₀ to ground.
17. Select readout location 0177 with Channel and Sector Selector switches and depress Command Readout button.
18. Repeat steps 6 through 8.
19. Disconnect ground from D₀ and depress Start 2 button (computer is in a "copy from Y reader" state).
20. Depress numbers 01 on console keyboard and channel 01 will be filled with zero in all sectors through 0177.
21. After routine halts, reconnect Do to ground and repeat step 11.

22. Repeat steps 12, 15, 16, 17, 18, 19, 20, and 21 except as follows:

<table>
<thead>
<tr>
<th>When Testing Channel</th>
<th>Select Readout Location</th>
<th>And Depress Keyboard Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>0277</td>
<td>02</td>
</tr>
<tr>
<td>03</td>
<td>0377</td>
<td>03</td>
</tr>
<tr>
<td>04</td>
<td>0477</td>
<td>04</td>
</tr>
<tr>
<td>05</td>
<td>0577</td>
<td>05</td>
</tr>
<tr>
<td>06</td>
<td>0677</td>
<td>06</td>
</tr>
<tr>
<td>07</td>
<td>0777</td>
<td>07</td>
</tr>
</tbody>
</table>

23. To test remaining channels repeat steps 12, 15, 16, 17, 18, 19, 20, and 21 except as follows:

<table>
<thead>
<tr>
<th>When Testing Channels</th>
<th>Observe Amplitude At</th>
<th>Adjust Amplitude At</th>
<th>Select Corresponding Readout Locations</th>
<th>And Depress Corresponding Keyboard Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 through 17</td>
<td>J550-2</td>
<td>J343</td>
<td>1077, 1177, etc through 1777</td>
<td>10 through 17</td>
</tr>
<tr>
<td>20 through 27</td>
<td>J550-3</td>
<td>J344</td>
<td>2077, 2177, etc through 2777</td>
<td>20 through 27</td>
</tr>
<tr>
<td>30 through 37</td>
<td>J550-4</td>
<td>J345</td>
<td>3077, 3177, etc through 3777</td>
<td>30 through 37</td>
</tr>
<tr>
<td>40 through 47</td>
<td>J550-5</td>
<td>J346</td>
<td>4077, 4177, etc through 4777</td>
<td>40 through 47</td>
</tr>
<tr>
<td>50 through 57</td>
<td>J550-6</td>
<td>J347</td>
<td>5077, 5177, etc through 5777</td>
<td>50 through 57</td>
</tr>
<tr>
<td>60 through 67</td>
<td>J550-7</td>
<td>J348</td>
<td>6077, 6177, etc through 6777</td>
<td>60 through 67</td>
</tr>
<tr>
<td>When Testing</td>
<td>Observe Amplitude At</td>
<td>Adjust Amplitude At</td>
<td>Select Corresponding Readout Locations</td>
<td>And Depress Corresponding Keyboard Numbers</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------</td>
<td>---------------------</td>
<td>----------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Channels</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70 through 77</td>
<td>J550-8</td>
<td>J349</td>
<td>7077, 7177, etc</td>
<td>70 through 77</td>
</tr>
</tbody>
</table>

(1) Pulse width of positive pulse and negative pulse measured individually at 10 percent of its magnitude shall be less than:

- 10 usec, for loops and registers
- 12 usec, for main memory

(2) Amplitude, excluding strobe:

- 4.5 ±0.5 V, main memory
- 5.0 ±0.5 V, loops and registers

Figure 39. Strobe Waveshape for Normal Pulse
Loop and Register Signal Check

The following procedures enable checks of amplitude, pulse width, noise and crosstalk of all loops and registers (additional procedures are given in the Computer Service Manual).

CAUTION

The computer should never be moved while the memory disk is rotating; allow at least 15 minutes from power turnoff for complete memory deceleration before moving computer.

NOTE

See figure 40 for location of connectors.

1. Enter and verify Loop Load routine with tape reader, typewriter, or console keyboard.
2. Depress Start 1 button and allow computer to operate until all loops and registers (except X register) are filled (when computer halts, location counter indicator should read 2525.0).
3. Observe amplitude of loops L and V and registers A, B, R, and Z (C) with oscilloscope at connectors indicated below (amplitude excluding strobe should be 5 ±0.5 volts peak-to-peak).

<table>
<thead>
<tr>
<th>Register or Loop</th>
<th>Connector and Pin Number</th>
<th>Read Amplifier Adjustment Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>J525-1</td>
<td>J529</td>
</tr>
<tr>
<td>B</td>
<td>J525-2 or</td>
<td>J530</td>
</tr>
<tr>
<td>R</td>
<td>J525-3 through</td>
<td>J531</td>
</tr>
<tr>
<td>X</td>
<td>J525-4 monitor</td>
<td>J532</td>
</tr>
<tr>
<td>Z (C)</td>
<td>J525-5 box</td>
<td>J533</td>
</tr>
<tr>
<td>L2a*</td>
<td>J525-6 plugged</td>
<td>J534</td>
</tr>
<tr>
<td>L2*</td>
<td>J525-7 into</td>
<td>J535</td>
</tr>
<tr>
<td>V2a*</td>
<td>J525-8 J525</td>
<td>J536</td>
</tr>
<tr>
<td>V2*</td>
<td>J525-9</td>
<td>J537</td>
</tr>
</tbody>
</table>

*L and V loops each contain two read heads which transmit signals to separate read amplifiers.

4. If necessary, adjust amplitude at read amplifier board plugged into locations indicated in table in step 3.
5. **Depress Command button followed by Enter button** (this fills X register).

6. Repeat steps 3 and 4 on X register.


8. Observe noise and crosstalk (in combination) with oscilloscope (should be less than 500 millivolts).

9. If noise and crosstalk exceed 500 millivolts zero-set each register or loop in turn to determine if it is being picked up from any of the other registers or loops; this is accomplished by observing on the oscilloscope if there is a decrease in noise and crosstalk as the zero setting indicates need for head adjustment.

10. **Depress Start 1 button** to refill registers and loops with pattern in Loop Load routine.

11. Repeat steps 7 through 10 until all registers and loops have been checked for noise and crosstalk, except in step 7 zero-set a different register or loop each time (X register is filled as in Step 5; step 10 is not repeated after the last loop-and-register check has been made).

12. **Depress Start 1 button** and allow all loops and registers to fill with pattern in Loop Load routine.

13. Fill X register as in step 5.

14. Check pulse width of each register and loop one at a time at 10 percent of its magnitude (width should be less than 10 usec).
System Check Under Marginal Conditions

Adjustments used to establish marginal test conditions of table are as follows:

-18V  Static margin = R12
-6V   Static margin = R14
-12V  Static margin = R5
+6V   Static margin = R13
-12V  Dynamic margin = R10
+6V   Dynamic margin = R40

R12, R14, R5, R3, R10, R40 located on marginal test board of System Tester.

Clock jitter width = R4 on clock board No. 1 in computer.

Checking the computer under marginal conditions with the System Tester enables the location of beginning malfunctions. Both static and dynamic marginal conditions can be established in varying degrees. In the static mode, the System Tester disables the computer clock and the clock pulse is then initiated at will by depressing the One-Shot Clock switch. The dynamic mode enables running routines under computer clock control, but also with varying marginal conditions. Procedures are as follows:

1. Establish the marginal conditions desired by placing the Static-Dynamic Mode and Marginal Test switches on the System Tester to the appropriate position (see table 9).
2. Establish the marginal conditions necessary for the desired mode and degree of marginal condition by adjusting the potentiometers as specified above.
3. Make the operational check desired.
4. To test computer under most severe marginal conditions:
   a. Place Static-Dynamic switch in Dynamic position.
   b. Place Marginal Test switch on position five.
   c. Enter and verify any test routine.
   d. Place Operation switch in Single Step or Continuous position as desired.
<table>
<thead>
<tr>
<th>Switch Setting</th>
<th>Mode</th>
<th>Marginal Test</th>
<th>Action</th>
<th>Voltage Change</th>
<th>Clock Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_1$</td>
<td>$S_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td>OFF</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Static</td>
<td>No. 1</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Lowered by 10%</td>
</tr>
<tr>
<td>Static</td>
<td>No. 2</td>
<td>Lowered by 10%</td>
<td>None</td>
<td>None</td>
<td>Lowered by 10%</td>
</tr>
<tr>
<td>Static</td>
<td>No. 3</td>
<td>Lowered by 10%</td>
<td>None</td>
<td>Raised by 10%</td>
<td>Lowered by 10%</td>
</tr>
<tr>
<td>Static</td>
<td>No. 4</td>
<td>Lowered by 10%</td>
<td>Raised by 10%</td>
<td>Raised by 10%</td>
<td>Lowered by 10%</td>
</tr>
<tr>
<td>Static</td>
<td>No. 5</td>
<td>Lowered by 10%</td>
<td>Raised by 10%</td>
<td>Raised by 10%</td>
<td>Lowered by 10%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>OFF</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Dynamic</td>
<td>No. 1</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Lowered by 10%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>No. 2</td>
<td>Lowered by 10%</td>
<td>None</td>
<td>None</td>
<td>Lowered by 10%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>No. 3</td>
<td>Lowered by 10%</td>
<td>None</td>
<td>Jitter ±10%</td>
<td>Lowered by 10%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>No. 4</td>
<td>Lowered by 10%</td>
<td>Jitter ±10%</td>
<td>Jitter ±10%</td>
<td>Lowered by 10%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>No. 5</td>
<td>Lowered by 10%</td>
<td>Jitter ±10%</td>
<td>Jitter ±10%</td>
<td>Lowered by 10%</td>
</tr>
</tbody>
</table>

Table 9. Marginal Test Conditions
e. Start computer (if computer halts at any place in the routine other than where so directed, the fault will most likely be a component or circuit failure caused by the marginal operating condition).

f. Remove power from computer and replace component or circuit board that failed.

g. Rerun routine used in step c to assure satisfactory operation under this marginal condition.

CONTROL CONSOLE

Normally the control console requires relatively little maintenance, consisting principally of replacing indicator neon and visual readout NIXIE tubes. All controls and indicators, however, should be checked periodically to detect incipient malfunctions if possible and ensure proper operation. Troubleshooting of control console malfunctions will usually involve the visual readout controls and NIXIE tube.

Procedures for checking all controls and indicators are given on the following pages. If a real or beginning malfunction is detected during this preliminary checkout, it can be usually isolated by testing the connections and circuits involved. Detailed procedures for maintenance of the control console are presented in the Input-Output Service Manual.

Controls and Indicators

When power to the computer has been turned on, the neon indicators on the sloping portion of the control panel should initially display the following characteristics. Any indication contrary to that given below should be noted as it will aid in locating a malfunction, if one exists.

<table>
<thead>
<tr>
<th>Neon</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>should be a steady glow</td>
</tr>
<tr>
<td>READY</td>
<td>should be a steady glow</td>
</tr>
<tr>
<td>FILL SOURCE KEYBOARD</td>
<td>should be a steady glow; if not, depress FILL SOURCE KEYBOARD button.</td>
</tr>
<tr>
<td>HIGH TEMP</td>
<td>should be off</td>
</tr>
<tr>
<td>INTER-LOCK</td>
<td>should be off</td>
</tr>
<tr>
<td>VERIFY ERROR</td>
<td>should be off; if on, depress ERROR RESET button</td>
</tr>
</tbody>
</table>
Checking the fill mode indicators achieves two objectives: (1) a test on correct operation of the keys involved, and (2) verification of correct operation of the indicators. Procedures for checking the controls and indicators used in entering information in command and number fill modes and in setting the location counter are:

**Number Fill Mode.** Depress "N" key; NUMBER neon should light. Depress "+" key; "D" and "+" neons to left of visual readout should light. Depress CLEAR key to extinguish "D" and "+" neons.

Depress "-" key; "D" and "-" neons to left of visual readout should light.

**Command Fill Mode.** Depress "C" key; command neon should light. Depress "+" neons to left of visual readout should light. Depress CLEAR key to extinguish "C" and "+" neons.

Depress "-" key; "C" and "-" neons to left of visual readout should light. Depress CLEAR key to extinguish "C" and "-" neons.

**Location Selection.** Depress LOCATION key; LOCATION neon and "0" neon to left of visual readout should light. Depress CLEAR key; "0" neon should remain lighted. Select location 00001 by depressing corresponding number keys and ENTER; all LOCATION COUNTER neons should be off except HALF-WORD neon.

Continue check as follows: select location 11110 and ENTER; only number "1" neons in both CHANNEL and SECTOR sections of LOCATION COUNTER should light. Proceed with check by selecting locations 22220, 33330, 44440, 55550, 66660, and 77770, noting LOCATION COUNTER display after each selection by counting (octally) the numbers represented by the neons. For example, when location 55550 is selected, neons 4 and 1 should be lighted in both positions of each of the channel and sector portions of the location counter indicator.
Visual Readout Nixie Tubes and Decimal Number Keys

While the preceding Location Selection check will display the numbers entered on the visual readout, the NIXIE should be further tested. To conduct this further check, depress number 1 key at the keyboard 15 times to fill all 15 NIXIEs, observing that each in turn displays a complete configuration of the number entered. Depress Clear key to clear NIXIEs and enter number 2 in the same manner and making the same observation. Continue by displaying all numbers, including zero and decimal point, in this manner. A test on correct operation of the NIXIE under computer control is given in Section V; procedure for checking operation of the console keyboard is presented later in this section.

Keyboard Fill Source Button and Indicator

Proper functioning of the Keyboard Fill Source button and its indicator is best confirmed by checking them with the System tester. The procedure is:

1. Turn "$T_o$" flip-flop on System Tester on; Fill Source neon should extinguish.
2. Depress Fill Source button; neon should relight.

Error and Environment Indicators

Because these indicators are the principal sources of information regarding proper operation, they should be tested frequently to assure correct operation. Tests on the Verify, Output, and Overflow Error indicators are performed with the System Tester. The procedure on each test is:

Verify Error Indicator.

1. Set "$H_o$" and "$J_o$" on System tester to true (1-set) states; Verify Error neon should flash.
2. Set "$H_o$" and "$J_o$" to false (0-set).

Output Error Indicator.

1. Set "$J_{13}$" to true (1-set); Output Error neon should flash.

Overflow Error Indicator

2. Set "$J_{13}$" to false (0-set); "$J_o$" should be true and Overflow neon should flash.
3. Set "$J_o$" to false (0-set).
Interlock Indicator.

1. Release both doors on computer sufficiently to energize Interlock neon.
2. Push in one of interlock switches on computer door (Interlock neon should remain flashing).
3. Release interlock switch and push in other switch (Interlock neon should remain flashing).
4. Close and latch both doors (Interlock neon should extinguish).

Line Transient.

1. Connect and Adjust Variac until system is supplied with 108 V ac as monitored on voltmeter (Line Transient neon should light).
2. Depress Error Reset button (Line Transient neon should extinguish momentarily).
3. Adjust Variac to 115 V ac (Line Transient neon should extinguish).
4. Adjust Variac until system is supplied with 125 V ac as monitored on voltmeter (Line Transient neon should light).
5. Depress Error Reset button (Line Transient neon should extinguish momentarily).
6. Adjust Variac to 115 V ac (Line Transient neon should extinguish).

Compute Indicator and Start and Stop Buttons

Proper operation of the Compute indicator can be determined by running a known program already stored in memory or the following short routine:

L 00000 ENTER
C +0000030 +0100040 ENTER (L00000)
+6000030 +3600030 ENTER (L00010)
+570000000000000 ENTER (L00020)
N +0. ENTER (L00030)
+2. ENTER (L00040)

Depress START 2 button; COMPUTE neon should light and FILL SOURCE neon should extinguish. (The LOCATION COUNTER will display a continuous random lighting during the computation; all other neons should be extinguished except READY and ON. The visual readout will continuously display the changing addition as the computation proceeds.) Depress STOP button; computer will stop computing and COMPUTE neon will extinguish and FILL SOURCE neon will light. (The LOCATION COUNTER neons will display the location the Computer halted when computation was stopped.)
Start 1, Start 2, and Start 3 Buttons

Proper functioning of these buttons can be verified by the following procedure (correct operation of the keyboard in the various fill modes should be determined first to ensure proper entry):

1. Enter the following routine:

<table>
<thead>
<tr>
<th>Location</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.0</td>
<td>+0000000 +0000000</td>
</tr>
<tr>
<td>0002.0</td>
<td>+0000000 +0000000</td>
</tr>
<tr>
<td>0003.0</td>
<td>+0000000 +0000000</td>
</tr>
</tbody>
</table>

2. Place the Operation switch in Single-Command position.
3. Depress Start 1 button (location counter should read 0001.1).
4. Depress Start 2 button (location counter should read 0002.1).
5. Depress Start 3 button (location counter should read 0003.1).

Octal, Decimal, and Command Readout Buttons

The three readout buttons can be checked by the following procedure (the test can also serve as a check on proper functioning of the rotary switches by following the procedures outlined under comments):

1. Enter the following information:

<table>
<thead>
<tr>
<th>Location</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000.0</td>
<td>+2443360 -4700760</td>
</tr>
</tbody>
</table>

2. Readout memory location 0000.0 with command button; visual readout should display +244336004700760
3. Readout memory location 0000.0 with octal button; visual readout should display +2443361160174
4. Readout memory location 0000.0 with decimal button; visual readout should display +523789.0

Comments

Operation of the rotary switches can be checked by placing the information in other main memory locations and by transferring it to the loops and registers with the appropriate commands after entry and prior to readout. The information can be copied into any loop location by executing the Copy to L Loop and Copy to V Loop instructions. Execution of a Clear and Add command places the information in the Accumulator and Command register; execution of a Clear and Add command followed by the Exchange A and R instruction places it in the R
register; execution of a Clear and Add command followed by a Store A and Exchange A and X instruction places it in the X register.

**Single Command Switch**

Correct operation of the computer in Single Command operation can be ascertained with the following procedure:

1. Enter the following information:

<table>
<thead>
<tr>
<th>Location</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.0</td>
<td>+0000000 +0000000</td>
</tr>
<tr>
<td>0002.0</td>
<td>+0000000 +0000000</td>
</tr>
<tr>
<td>0003.0</td>
<td>+0000000 +0000000</td>
</tr>
</tbody>
</table>

3. Depress Start 1 button; location counter should read 0001.1.
4. Depress Start button; location counter should read 0002.0.
5. Depress Start button; location counter should read 0002.1.
6. Depress Start button; location counter should read 0003.0.
7. Depress Start button; location counter should read 0003.1.

**Preset Stop Switch**

The Preset Stop switch can be tested with the following procedure:

1. Enter the following routine:

<table>
<thead>
<tr>
<th>Location</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.0</td>
<td>+5700100 -0000000</td>
</tr>
<tr>
<td>0010.0</td>
<td>+5700201 -0000000</td>
</tr>
<tr>
<td>0020.0</td>
<td>+0000000 +570010</td>
</tr>
</tbody>
</table>

2. Depress Start 1 button.
3. Set Channel and Sector Selector switches to 0010 and place Preset Stop switch in 1st position to stop the computer after executing the first command in a word; routine should halt and location counter should read 0020.1.
4. Place Preset Stop switch in Off position.
5. Depress Start 1 button.
6. Set Channel and Sector Selector switches to 0020 and place Preset Stop switch in 2nd position to stop the computer after executing the second command in a word; routine should halt and location counter should read 0001.0.
7. Place Preset Stop switch in Off position.
8. Depress Start 1 button.
9. Set Channel and Sector Selector switches to 0001 and place Preset Stop switch in 2nd position; computer should continue operating.
10. Set Channel and Sector Selector switches to 0020 and place Preset Stop switch in 1st position; computer should continue operating.

**Transfer Stop Switch**

Proper operation of the Transfer Stop switch can be determined by the following procedure:

1. Enter the following routine:

<table>
<thead>
<tr>
<th>Location</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.0</td>
<td>+5700011 +5400020</td>
</tr>
<tr>
<td>0002.0</td>
<td>+5500021 +5600011</td>
</tr>
</tbody>
</table>

2. Place Transfer Stop switch up (on).
3. Depress Start 1 button; location counter should read 0001.1.
4. Depress Start button; location counter should read 0002.0.
5. Depress Start button; location counter should read 0002.1.
6. Depress Start button; location counter should read 0001.1.

**Sense Switches B, C, D**

Correct operation in both obeying and ignoring the settings of Sense Switches B, C, and D can be determined with the routine and procedures given below (the routine is changed as indicated according to the switch being tested; the procedures remain the same, however, except as stated):

1. Enter one of the following routines, according to the switch being tested (memory locations are the same for each routine):

<table>
<thead>
<tr>
<th>Location</th>
<th>Commands Switch B</th>
<th>Commands Switch C</th>
<th>Commands Switch D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.0</td>
<td>+5400051 +7700051</td>
<td>+5500051 +7700051</td>
<td>+5600051 +7700051</td>
</tr>
<tr>
<td>0002.0</td>
<td>+7700000 +5400040</td>
<td>+7700000 +5500040</td>
<td>+7700000 +5600040</td>
</tr>
<tr>
<td>0003.0</td>
<td>+7700040 +7700000</td>
<td>+7700040 +7700000</td>
<td>+7700040 +7700000</td>
</tr>
<tr>
<td>0004.0</td>
<td>+5400010 +7700010</td>
<td>+5500010 +7700010</td>
<td>+5600010 +7700010</td>
</tr>
<tr>
<td>0005.0</td>
<td>+7700000 +5400021</td>
<td>+7700000 +5500021</td>
<td>+7700000 +5600021</td>
</tr>
<tr>
<td>0006.0</td>
<td>+7700021 -0000000</td>
<td>+7700021 -0000000</td>
<td>+7700021 -0000000</td>
</tr>
</tbody>
</table>
2. Place Sense Switch B (C or D) up (tests if sense switch transfer instruction is ignored).
3. Place Operation switch in Continuous position.
4. Depress Start 1 button (computer should halt).
5. Read location counter and C register (readouts should be as follows):

<table>
<thead>
<tr>
<th>Location Counter</th>
<th>C Register Switch B</th>
<th>C Register Switch C</th>
<th>C Register Switch D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0005.1</td>
<td>+54000511770051</td>
<td>+55000511770051</td>
<td>+56000511770051</td>
</tr>
</tbody>
</table>

6. Depress Start button; when computer halts, read out location counter and C register (readouts should be as follows):

<table>
<thead>
<tr>
<th>Location Counter</th>
<th>C Register Switch B</th>
<th>C Register Switch C</th>
<th>C Register Switch D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0002.1</td>
<td>+7700021000000000</td>
<td>+7700021000000000</td>
<td>+7700021000000000</td>
</tr>
</tbody>
</table>

7. Start button; when computer halts, read out location counter and C register (readouts should be as follows):

<table>
<thead>
<tr>
<th>Location Counter</th>
<th>C Register Switch B</th>
<th>C Register Switch C</th>
<th>C Register Switch D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0004.0</td>
<td>+7700040177000000</td>
<td>+7700040177000000</td>
<td>+7700040177000000</td>
</tr>
</tbody>
</table>

8. Depress Start button; when computer halts, read out location counter and C register (readouts should be as follows):

<table>
<thead>
<tr>
<th>Location Counter</th>
<th>C Register Switch B</th>
<th>C Register Switch C</th>
<th>C Register Switch D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.0</td>
<td>+540001017700010</td>
<td>+550001017700010</td>
<td>+560001017700010</td>
</tr>
</tbody>
</table>

9. Place Sense Switch B (C or D) down (tests if sense switch transfer instruction is followed).
10. Depress Start 1 button; routine should run continuously, after it has run for a couple of minutes, depress Stop button.
11. Place Transfer Stop switch up (on).
12. Depress Start 1 button; when computer halts, read out location counter and X register (readouts should be as follows):

<table>
<thead>
<tr>
<th>Location Counter</th>
<th>X Register Switch B</th>
<th>X Register Switch C</th>
<th>X Register Switch D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0005.1</td>
<td>+0000000000000010</td>
<td>+0000000000000010</td>
<td>+0000000000000010</td>
</tr>
</tbody>
</table>

13. Set location counter to 0002.1.
14. Depress Start button; when computer halts, read out location counter and X register (readouts should be as follows):

<table>
<thead>
<tr>
<th>Location Counter</th>
<th>X Register Switch B</th>
<th>X Register Switch C</th>
<th>X Register Switch D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0004.0</td>
<td>+000000000000021</td>
<td>+000000000000021</td>
<td>+000000000000021</td>
</tr>
</tbody>
</table>

**Typewriter-Actuated Control and Indicators**

Most control operations carried on from the control console can also be conducted from the typewriter. The console indicators, however, provide the indication of the control activities performed from the typewriter and the internal computer operations. Proper operation of the typewriter as a control unit as well as correct indication by the console neons can be ascertained by performing the following check procedures:

Turn Compute-Manual switch on punch control panel to Computer position, then depress Fill lever on typewriter; Fill Source neon on control console should extinguish.

Depress Letters key and L key on typewriter; Location neon on control console should light (the Location neon should remain lighted when the Figures key and numeral keys on the typewriter selecting a location are depressed).

Depress Return key on typewriter (this sets the location counter); the Location Counter lights should remain lighted.

Depress Letters key and C key on typewriter; Command neon on control should light (the Command neon should remain lighted when the Figures number keys representing a pair of commands, and the Return keys are depressed; the Location Counter neons should step up to show the next location).

Depress the Letters key and N key; Number neon on control console should light (the Number neon should remain lighted when the Figures, plus, number keys representing a number, decimal point, and the Return keys on the typewriter are depressed; the Location Counter neons should step up to the next location when the decimal point and Return keys are depressed during mixed number input and when the Return key is depressed on integer or fraction input).

If a routine is stored in the computer, set the Location Counter to the location at which the routine begins and depress the $S$ key on the typewriter; the Compute neon on the control console should light and remain so until the computer stops or is halted.
To stop computation manually from the typewriter, depress the H key; the Compute neon on the control console should extinguish.

**INPUT-OUTPUT UNITS**

**Typewriter**

When checking or troubleshooting the typewriter, proper functioning of controls should be determined before attempting to locate the cause of incorrect operation, unless the area of malfunction is definitely established from a report of the difficulty.

Proper functioning of the various typewriter controls can be readily checked by operating them and observing the corresponding action. All controls and their functions are given in the Input-Output Service Manual; observing the performance of the functions will serve as a satisfactory preliminary check on operation of the controls and usually isolate the malfunction. More extensive checking, however, may be necessary on such malfunctions as weak impression, improper tab operation or carriage action, and main power faults. Weak impression may also be caused by weak drive output in the decoder and mechanical drive units, in which case all circuits and components associated with typing under computer control may require checking. Improper tab operation or carriage return may require checking the circuits carrying the signals from the computer, signal decoding, and the mechanical action involved in tab and carriage return operations. A routine to test typing with tab operation under computer control is given in the Test Routine Section. Main power faults will necessitate testing of the power supply circuits for faulty connections and components.

Incorrect entry and recording of information, or non-entry and non-recording, can usually be traced to malfunctions in the coder or decoder assemblies, respectively. The source of the trouble may, however, also be located in the power assembly supplying mechanical drive to the coder and decoder or in the circuitry transmitting the signals to and from the computer. Procedures for isolating these malfunctions are given in the Input-Output Service Manual.

Preliminary checking of the typewriter for correct entry of information into the computer involves principally a comparison of the characters recorded from the computer with those entered. This can be accomplished by running the test routine given in Section . Correct entry of numbers in command and number formats can also be verified by displaying the contents of a memory location into which this information has been entered from the typewriter. Proper input of alphanumeric
characters can be verified by entering the information and then record-
ing it with the typewriter or punch under computer control, or by enter-
ing information in a specific sequence and then reading it out in command format. A procedure for manually entering alphanumeric characters from the typewriter and reading them out in command format on the visual readout is outlined below. A routine for testing alphanumeric input from the typewriter under computer control is given in Section Before checking for accuracy of input, proper operation of the type-
writer in the various input modes should be confirmed by performing the steps outlined under Typewriter Actuated Indications and Control.

Depress Fill key on typewriter, then the characters L 00100, followed by Carriage Return.

Depress the following keys in the sequence indicated:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Carriage Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>RYRYRYRYRY</td>
<td>Carriage Return</td>
</tr>
<tr>
<td>01234567</td>
<td>Carriage Return</td>
</tr>
<tr>
<td>89ZAMXLC</td>
<td>Carriage Return</td>
</tr>
<tr>
<td>NSHDFGKV</td>
<td>Carriage Return</td>
</tr>
</tbody>
</table>

Read out the following memory locations on the visual readout, using the Command readout button (the display should read as indicated):

<table>
<thead>
<tr>
<th>Location</th>
<th>Information Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010.0</td>
<td>-525252 1 0 525252 1</td>
</tr>
<tr>
<td>0011.0</td>
<td>+327460 1 0 520523 1</td>
</tr>
<tr>
<td>0012.0</td>
<td>-330421 1 1</td>
</tr>
<tr>
<td>0013.0</td>
<td>-605504 1 0 672377 0</td>
</tr>
</tbody>
</table>

Initial checkout of the typewriter for correct recording of output consists mainly of comparing the information transmitted from the computer with that typed or typewriter action performed. This is most easily accomplished with test routines in which the output as it should be recorded is known. These test routines are given in the Test Routine Section. Correct output can also be checked by entering information in a specific memory location and then entering and executing one of the Type commands. These diagnostic checks are especially useful when the source of the malfunction is in the computer output circuitry rather than the typewriter. Malfunctions of incorrect decoding and recording by the typewriter are usually detected by the echo-checking circuits. These circuits verify the output with the signals sent from the computer. Any discrepancy causes the computer and typewriter to halt and the Output Error indicator on the control console to light. The character
recorded incorrectly can be determined by comparing its configuration with that displayed by the output register indicator on the punch. Output errors can be isolated by troubleshooting computer and typewriter circuits and components associated with this function.

Console Keyboard and Visual Readout

Console Keyboard

Should preliminary checkout of the various control console indicators and controls indicate a malfunction in the keyboard, the fault can be further isolated by the following procedure:

1. Enter the following routine from the console keyboard (correct entry of each pair of commands can be verified by reading out the Accumulator before depressing the Enter button and the memory location into which it was placed after depressing the Enter button):

<table>
<thead>
<tr>
<th>Location</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.0</td>
<td>+3600050 +7700020</td>
</tr>
<tr>
<td>0002.0</td>
<td>+3600060 +7700030</td>
</tr>
<tr>
<td>0003.0</td>
<td>+3600070 +7700040</td>
</tr>
<tr>
<td>0004.0</td>
<td>+3600100 +7700010</td>
</tr>
</tbody>
</table>

2. Enter the following numbers in Number Fill mode (correct keying of numbers can be verified by observing display on visual readout before depressing Clear or Enter button):

<table>
<thead>
<tr>
<th>Location</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0005.0</td>
<td>+8192</td>
</tr>
<tr>
<td>0006.0</td>
<td>-00048828125</td>
</tr>
<tr>
<td>0007.0</td>
<td>+15722656250</td>
</tr>
<tr>
<td>0010.0</td>
<td>-590864</td>
</tr>
</tbody>
</table>

3. Depress Start 1 button.
4. Read-out location 0002.0; display should be +00 00000 0 01 00000 (in command format).
5. Depress Start button.
6. Read out location 0003.0; display (in command format) should be -00 02000 0 00 00000.
7. Depress Start button.
8. Read out location 0004.0, display (in command format) should be +12 04000 0 00 00000.
9. Depress Start button.
10. Read out location 0001.0; display (in command format) should be 00 00000 1 10 10100.  
(The routine should repeat if the Start button continues to be depressed).

Visual Readout

If the control console checkout procedures indicate a malfunction in the visual readout, the fault can be further isolated by running the NIXIE Tube Test routine given in the Test Routine section. This test should define the tube assembly or its associated circuitry causing the difficulty. The exact source of the malfunction can then be determined by examining the circuit connections and components used by the malfunctioning tube assembly. Neon indicator faults can be readily traced by examining these circuits. If a fault within the computer appears to be the cause of the malfunction, it can be confirmed by entering known information into a specific memory location and then reading it out on both the visual readout and typewriter or punch.

Photoelectric Paper Tape Reader

Unless a malfunction in the paper tape reader is definitely known from an explanation of the trouble, it is often to advantage to first check the d-c voltages, then the controls and mechanical operation. Application of power to the reader upon turning on the Power On-Off circuit breaker and Motor switches can usually be checked simply by observing if the capstan rotates. If the capstan is not rotating, the trouble can be isolated by checking for circuit continuity. If the capstan rotates, and d-c voltages are correct, the exciter lamps and the photodiode window on the reader head should be checked to determine that the lamps are not blackened or burned out and that the window is clean. Also check the AGC board signals and adjustments. Proper functioning of the remaining switches and indicators can be determined with the following procedures:

Fill Button and Indicator

Insert any program tape into reader and depress Fill button. If tape fails to pass through reader head, check idler roller for extreme wear, switch and circuits for faulty connections and components, and insufficient pressure against capstan drum. The Fill indicator should light upon depressing the Fill button and extinguish when tape movement stops.
Verify Button and Indicator

Enter any program tape through reader, then re-enter same tape but by depressing Verify instead of Fill button. If tape fails to pass through reader, check switch and circuits for faulty connections and components. The Verify indicator should light upon depressing the Verify button and extinguish when tape movement stops.

Stop Switch

Enter any program tape through reader; before tape reaches end depress Stop switch. If tape fails to halt, check switch and circuit connections and control components.

Tape Advance Switch

Insert endless program tape in reader and depress Fill switch. After a short length of tape has passed through the reader, depress Stop switch, then the Tape Advance switch. If the tape fails to start at any point other than the beginning of the tape and to stop at slit at starting point, check switch and circuits for faulty connections and components.

Mechanical actions of the tape reader requiring checkout are usually indicated by faulty feeding of tape past the reader head. Indications include tape slippage on starting tape movement and excessive tape movement after a stop has been initiated. Cause of excessive tape slippage on starting can be determined by checking the idler roller for excessive wear, the solenoid engaging the idler roller for malfunction, the tape pressure spring for weakness, and the movement of the locking handle for faulty closing. If the tape only accelerates slowly, the drive belt between motor and capstan should be checked for slippage. Excessive tape movement after a stop has been given can be diagnosed by checking the solenoid engaging the idler roller for abnormal delay in action, and the brake for faulty braking action.

Incorrect reading by the tape reader can best be confirmed and the cause isolated by running test routines. The routines testing operation and correct reading of the tape reader are given in the Test Routine section. Correct reading may also be ascertained by entering a tape of known content, then displaying the contents of the memory locations into which the information was entered and comparing the readout with the information in the tape. If recording in memory is faulty, one work at a time may be entered and displayed in the accumulator.
Most punch malfunctions are attributable to failures of mechanical parts or wear to a degree requiring readjustment. But unless these conditions are definitely known to be the cause of the malfunction, it is advisable to first check the controls and test-run a tape to note any improper operation.

Proper application of power to the punch upon turning on the Power On-Off circuit breaker and Motor switches can be readily ascertained. If the punch remains inoperative, checking the power supply circuits will locate the fault. Proper functioning of the other switches and indicators can be determined by following the procedures given below:

**Computer-Manual Punch Switch**

In Computer position, place information in memory, then attempt to punch it into a blank tape with any of the Punch instructions. To test Manual position, strike a key on the typewriter. Though failure of the punch to respond in either instance may be caused by a malfunction elsewhere, the switch circuits should be tested before conducting other checkout or troubleshooting procedures. (See also Punch-External Switch below.)

**Punch-External Switch**

Unless the system includes a plotter or other external output device, only the Punch position need be checked. This switch should be in the Punch position when the Computer-Manual Punch Switch is tested. Because failure of the punch to operate may be caused by malfunction in either switch, the Punch position should be tested simultaneously with the two positions of the Computer-Manual Punch switch.

**Tape Advance Switch**

Place a section of blank tape in punch and depress Tape Advance button. If tape fails to advance, check connections and circuits associated with tape advance process.

**Output Register Indicator**

Because this indicator must always display the correct code transmitted from the computer to an output unit, it should be checked frequently to assure correct functioning. To test the indicator, enter a program which will transmit a series of the same character to the
punch at a time, the first series consisting of a character with a hole punched in only one channel, the second series of a character punching in only another channel, and continuing until a series has been punched in all channels. (This procedure may be repeated, if desired, with a program containing series of characters punching in combinations of channels.) Observe the output register indicator while the punch is recording, if further checking is necessary to determine correct output, visually inspect tape punched. If the indicator malfunctions, inspect neon, connections, and circuits to output register. If further checking is necessary, use the System Tester to monitor the output register.

Tape punch mechanical malfunctions are usually due to misadjustment caused by usage and consequent wear and misalignment. Procedures for disassembly and adjustment are given in the Input-Output Service Manual. Indications of maladjustment include improper tape feeding, poor quality punching, and double punching. A test to confirm existence of double punching is given in the test routines. Incorrect punching, if not definitely known, can best be detected and the cause located by running the test routines presented in the Test Routine section. Malfunctions can be further isolated and the cause determined by checking circuits and components; procedures for accomplishing this are given in the Input-Output manual.

TAPE PREPARATION PROCESS

With Computer Off

Determining correct operation of the typewriter and tape punch as a tape preparation system can be accomplished with the following procedure:

1. Turn on power at desk circuit breaker switch, typewriter, and tape punch.
3. Depress each typewriter character and function key having a teletype code equivalent in Letters Shift and Figure Shift modes (any sequence of key depression is satisfactory but one should be selected that will permit easy comparison between character typed and punched).
4. Visually inspect tape to determine if character configurations punched correspond to character typed and if configuration punched is legitimate code.
If the test above indicates a malfunction, the source can be isolated by following the checkout and troubleshooting procedures for each unit and by checking the connections, circuits, and components associated with the tape preparation process.

**With Computer Operating**

Proper functioning of the typewriter and tape punch in tape preparation while the computer is operating can be determined with the following procedure (perform check with computer off as given above before testing with computer operating):

1. Enter any routine that will operate in a loop and start computer.
2. After computer has operated for a few seconds, depress Stop button.
4. Depress Start button on computer.
5. Turn Power switch on punch to on.
6. Depress Tape Advance button on punch and hold for a few seconds (computer operation should not be affected).
7. Depress following keys on typewriter in succession: FIGS 1 2 3 4 5 6 7 8 9 0 - s $ ! & HALT ' ( ) + / : ; ? , . LINEFEED BLANK TAB RETURN (computer operation should not be affected).
8. Depress Letters Shift key followed by keys listed in Step 7 except FIGS (computer operation should not be affected).
9. Turn Punch-External switch to External position.
10. Depress typewriter keys as in Steps 7 and 8 (punch should not operate and computer operation should not be affected).

If computer operation should be affected, check circuits from typewriter to computer for wrong or faulty connections and malfunctions.
TEST ROUTINES

The test routines are a quick, effective means of determining proper operation of all or a part of a computing system. Frequently, it also can aid materially in locating a malfunction, particularly when the routine provides indication of the area of malfunction or records information which can be compared with known, correct results.

Most test routines check only a portion of a system. This is because (1) a routine testing operation of a system completely would require considerable time to run and (2) usually the unit at fault is known. Thus, while a few routines check the system extensively, the majority test one unit such as the memory or several units in combination such as the typewriter and tape punch.

Selecting the most appropriate routine to locate a malfunction requires a complete understanding of the theory of operation of the system. For example, if the punch records the wrong character the error could be caused by faulty execution of the Punch Character command rather than the punch recording incorrectly. Consequently the Type Character, Type/Punch Character, and Punch Character test routines might locate the difficulty more quickly than the Input-Output and Memory Test routine.

Interpreting a routine halt caused by a computer malfunction necessitates detailed knowledge of the system logic. When the logic is understood, the malfunction can usually be readily isolated to a particular circuit or component, particularly if the System tester is used in conjunction with the test routine to check the flip-flops and gates involved in the halt.

The test routines in this section, together with those among the tests in Section IV, enable a relatively easy check of the entire system both in preventive maintenance and trouble shooting. Test tapes are available for all except the very short routines. These, as well as the long routines if the tape reader is inoperative, can be readily entered from the typewriter or console keyboard. Operating instructions are included with each routine.
COMMAND TEST ROUTINE

Purpose

To check execution of all computer commands under marginal or normal operating conditions. (Operation as described in Steps 1 through 5 below tests all commands except input-output and HTR, TMI, TPL, TOV, TSB, TSC, and TSD; operation as described in Steps 6 through 8 tests the transfer commands mentioned.)

Operating Instructions

1. If routine is to be run under marginal conditions, connect System Tester and set marginal test switch on System Tester to 5; if routine is to be run under normal operating conditions, operate computer without System Tester connected.
2. Enter and verify Command Test routine with tape reader.
3. Place all Sense switches on Control Console in down position.
4. Depress Start 1 button. (Visual readout will display on right side of readout the number of times the routine has run successfully (see comments following 8).)
5. After routine has run the desired number of times, depress Stop button.
6. Set location counter to 2077.0
7. Depress Start button. (Routine should halt at location 2101.1 and display +0000000-0000510.)
8. Continue as indicated below:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Correct Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depress Start button</td>
<td>Routine halts at location 2103.1 and displays +0000000-0000530</td>
</tr>
<tr>
<td>Depress Start button</td>
<td>Routine halts at location 2104.0 and displays +0000000-0000540</td>
</tr>
<tr>
<td>Depress Start button</td>
<td>Routine halts at location 2104.1 and displays +0000000-0000550</td>
</tr>
</tbody>
</table>

5-2
<table>
<thead>
<tr>
<th><strong>Operation</strong></th>
<th><strong>Correct Result</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Depress Start button</td>
<td>Routine halts at location 2105.0 and displays +0000000-0000560</td>
</tr>
<tr>
<td>Place Sense Switch B up; depress Start button</td>
<td>Routine halts at location 2103.1 and displays +0000000-0000530</td>
</tr>
<tr>
<td>Keep Sense Switch B up; depress Start button</td>
<td>Routine halts at location 2105.0 and displays +0000000-0000560</td>
</tr>
<tr>
<td>Place Sense Switches B and C up; depress Start button</td>
<td>Routine halts at location 2101.1 and displays +0000000-0000510</td>
</tr>
<tr>
<td>Keep Sense Switches B and C up; depress Start button</td>
<td>Routine halts at location 2103.1 and displays +0000000-0000530</td>
</tr>
<tr>
<td>Keep Sense Switches B and C up; depress Start button</td>
<td>Routine halts at location 2105.0 and displays +0000000-0000560</td>
</tr>
<tr>
<td>Place Sense Switches B, C, and D up; depress Start button</td>
<td>Routine halts at location 2102.0 and displays +0000000-0000520</td>
</tr>
<tr>
<td>Keep Sense Switches B, C, and D up; depress Start button</td>
<td>Routine halts at location 2103.1 and displays +0000000-0000530</td>
</tr>
<tr>
<td>Keep Sense Switches B, C, and D up; depress Start button</td>
<td>Routine halts at location 2101.1 and displays +0000000-0000510</td>
</tr>
</tbody>
</table>
COMMENTS

If the routine halts, the location counter will be reset to the location from which the transfer command came. The command on which the malfunction occurred will be found two locations prior to that indicated; this command can be displayed on the visual readout by selecting the proper location with the rotary selector switches and depressing the command readout button. This routine can be used in its entirety or in two separate portions: Steps 1, 2, 3, 6, 7, and 8.

The command test routine printout is shown on the subsequent pages.
<table>
<thead>
<tr>
<th>L00000</th>
<th>+2525250+2525250</th>
<th>L00600</th>
<th>+0304220+0305220</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>+0003000+6004000</td>
<td>1.</td>
<td>+5000620+5702100</td>
</tr>
<tr>
<td>2.</td>
<td>+0305000+500031</td>
<td>2.</td>
<td>+0003010+2203000</td>
</tr>
<tr>
<td>3.</td>
<td>+5702100+0303000</td>
<td>3.</td>
<td>+3504230+0305230</td>
</tr>
<tr>
<td>4.</td>
<td>+0040300+6004010</td>
<td>4.</td>
<td>+5000550+5702100</td>
</tr>
<tr>
<td>5.</td>
<td>+0305010+5000061</td>
<td>5.</td>
<td>+0004240+0305240</td>
</tr>
<tr>
<td>6.</td>
<td>+5702100+0203000</td>
<td>6.</td>
<td>+5000670+5702100</td>
</tr>
<tr>
<td>7.</td>
<td>+0040420+0305020</td>
<td>7.</td>
<td>+003010+2303000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L00100</td>
<td>+5000110+5702100</td>
<td>L00700</td>
<td>+3504250+0305250</td>
</tr>
<tr>
<td>1.</td>
<td>+3003000+5003000</td>
<td>1.</td>
<td>+5000710+5702100</td>
</tr>
<tr>
<td>2.</td>
<td>+3504030+0305030</td>
<td>2.</td>
<td>+0004260+0305260</td>
</tr>
<tr>
<td>3.</td>
<td>+5000140+5702100</td>
<td>3.</td>
<td>+5000740+5702100</td>
</tr>
<tr>
<td>4.</td>
<td>+0040400+0305040</td>
<td>4.</td>
<td>+000320+2503200</td>
</tr>
<tr>
<td>5.</td>
<td>+0001060+5702100</td>
<td>5.</td>
<td>+6004270+0305270</td>
</tr>
<tr>
<td>6.</td>
<td>+3003000+5003020</td>
<td>6.</td>
<td>+5000770+5702100</td>
</tr>
<tr>
<td>7.</td>
<td>+3504050+0305050</td>
<td>7.</td>
<td>+003000+3303000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L00200</td>
<td>+5000210+5702100</td>
<td>L01000</td>
<td>+6004300+0305300</td>
</tr>
<tr>
<td>1.</td>
<td>+0004060+0305060</td>
<td>1.</td>
<td>+5001020+5702100</td>
</tr>
<tr>
<td>2.</td>
<td>+5000230+5702100</td>
<td>2.</td>
<td>+0003060+4000170</td>
</tr>
<tr>
<td>3.</td>
<td>+3403000+0603000</td>
<td>3.</td>
<td>+6004310+0305310</td>
</tr>
<tr>
<td>4.</td>
<td>+3504070+0305070</td>
<td>4.</td>
<td>+5001050+5702100</td>
</tr>
<tr>
<td>5.</td>
<td>+5000260+5702100</td>
<td>5.</td>
<td>+000300+4100170</td>
</tr>
<tr>
<td>6.</td>
<td>+0001010+0305100</td>
<td>6.</td>
<td>+6004320+0305320</td>
</tr>
<tr>
<td>7.</td>
<td>+5000300+5702100</td>
<td>7.</td>
<td>+5001100+5702100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L00300</td>
<td>+3003000+0703170</td>
<td>L01100</td>
<td>+1504330+6004340</td>
</tr>
<tr>
<td>1.</td>
<td>+3504110+0305110</td>
<td>1.</td>
<td>+0305340+5001121</td>
</tr>
<tr>
<td>2.</td>
<td>+5000330+5702100</td>
<td>2.</td>
<td>+5702100+0034330</td>
</tr>
<tr>
<td>3.</td>
<td>+0004120+0305120</td>
<td>3.</td>
<td>+0305330+5001141</td>
</tr>
<tr>
<td>4.</td>
<td>+5000350+5702100</td>
<td>4.</td>
<td>+5702100+0033000</td>
</tr>
<tr>
<td>5.</td>
<td>+0003000+1003000</td>
<td>5.</td>
<td>+3504350+4300000</td>
</tr>
<tr>
<td>6.</td>
<td>+3504130+0305130</td>
<td>6.</td>
<td>+3504370+0305370</td>
</tr>
<tr>
<td>7.</td>
<td>+5000460+5702100</td>
<td>7.</td>
<td>+5001200+5702100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L00400</td>
<td>+0004140+0305140</td>
<td>L01200</td>
<td>+0004350+0305350</td>
</tr>
<tr>
<td>1.</td>
<td>+5000420+5702100</td>
<td>1.</td>
<td>+5001220+5702100</td>
</tr>
<tr>
<td>2.</td>
<td>+0003000+1303000</td>
<td>2.</td>
<td>+0004360+0305360</td>
</tr>
<tr>
<td>3.</td>
<td>+6004150+0305150</td>
<td>3.</td>
<td>+5001210+5702100</td>
</tr>
<tr>
<td>4.</td>
<td>+5000450+5702100</td>
<td>4.</td>
<td>+0004400+0305400</td>
</tr>
<tr>
<td>5.</td>
<td>+1504160+0004170</td>
<td>5.</td>
<td>+5001260+5702100</td>
</tr>
<tr>
<td>6.</td>
<td>+0004160+0305160</td>
<td>6.</td>
<td>+4103220+3504410</td>
</tr>
<tr>
<td>7.</td>
<td>+5000500+5702100</td>
<td>7.</td>
<td>+0305410+5001301</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L00500</td>
<td>+0004170+0305170</td>
<td>L01300</td>
<td>+5702100+0034420</td>
</tr>
<tr>
<td>1.</td>
<td>+5000520+2702100</td>
<td>1.</td>
<td>+00305420+5001321</td>
</tr>
<tr>
<td>2.</td>
<td>+0003010+2003000</td>
<td>2.</td>
<td>+5702100+3003000</td>
</tr>
<tr>
<td>3.</td>
<td>+6004200+0305200</td>
<td>3.</td>
<td>+4500000+3504430</td>
</tr>
<tr>
<td>4.</td>
<td>+5000550+5702100</td>
<td>4.</td>
<td>+0305430+5001351</td>
</tr>
<tr>
<td>5.</td>
<td>+0003010+2103000</td>
<td>5.</td>
<td>+5702100+0004440</td>
</tr>
<tr>
<td>6.</td>
<td>+3504210+0305210</td>
<td>6.</td>
<td>+0305440+5001371</td>
</tr>
<tr>
<td>7.</td>
<td>+5000600+5702100</td>
<td>7.</td>
<td>+5702100+4403400</td>
</tr>
<tr>
<td>Command</td>
<td>Test Routine Printout (Cont)</td>
<td>Command</td>
<td>Test Routine Printout (Cont)</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
<td>---------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>LO1400</td>
<td>+6503500+0003500</td>
<td>LO2200</td>
<td>+6504300+6504000</td>
</tr>
<tr>
<td>1.</td>
<td>+303400+5001421</td>
<td>1.</td>
<td>+6504500+6504600</td>
</tr>
<tr>
<td>2.</td>
<td>+5702100+0003510</td>
<td>2.</td>
<td>+6504700+6510500</td>
</tr>
<tr>
<td>3.</td>
<td>+303410+5001441</td>
<td>3.</td>
<td>+6503700+700010</td>
</tr>
<tr>
<td>4.</td>
<td>+5702100+0003520</td>
<td>4.</td>
<td>+6507400+0210300</td>
</tr>
<tr>
<td>5.</td>
<td>+303420+5001461</td>
<td>5.</td>
<td>+0000000-0000321</td>
</tr>
<tr>
<td>6.</td>
<td>+5702100+0003530</td>
<td>6.</td>
<td>+0000000-1161671</td>
</tr>
<tr>
<td>7.</td>
<td>+303430+5001501</td>
<td>7.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>LO1500</td>
<td>+5702100+0003540</td>
<td>LO2300</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+303440+5001521</td>
<td>1.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>2.</td>
<td>+5702100+0003550</td>
<td>2.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+303450+5001541</td>
<td>3.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>4.</td>
<td>+5702100+0003550</td>
<td>4.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+303460+5001561</td>
<td>5.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>6.</td>
<td>+5702100+0003570</td>
<td>6.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>7.</td>
<td>+303470+5001591</td>
<td>7.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>LO1600</td>
<td>+5702100+6603600</td>
<td>LO2400</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+6703700+0003700</td>
<td>1.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>2.</td>
<td>+5702100+0003710</td>
<td>2.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+303510+5001631</td>
<td>3.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>4.</td>
<td>+5702100+0003750</td>
<td>4.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+303450+5001651</td>
<td>5.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>6.</td>
<td>+5702100+0003770</td>
<td>6.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>7.</td>
<td>+303470+5001691</td>
<td>7.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>LO1700</td>
<td>+303530+5001711</td>
<td>LO2500</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+5702100+5001740</td>
<td>1.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>2.</td>
<td>+303640+5001731</td>
<td>2.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+5702100+0003750</td>
<td>3.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>4.</td>
<td>+303650+5001751</td>
<td>4.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+5702100+0003760</td>
<td>5.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>6.</td>
<td>+303660+5001771</td>
<td>6.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>7.</td>
<td>+5702100+0003770</td>
<td>7.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>LO2000</td>
<td>+303670+5002011</td>
<td>LO2600</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+5702100+0002040</td>
<td>1.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>2.</td>
<td>+362050+0002040</td>
<td>2.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+5702100+0002160</td>
<td>3.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>4.</td>
<td>+303000+0000000</td>
<td>4.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+000000+0000010</td>
<td>5.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>6.</td>
<td>+000000+0000030</td>
<td>6.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>7.</td>
<td>+000000+00000370</td>
<td>7.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>LO2100</td>
<td>+1502240+6022500</td>
<td>LO2700</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+340000+6022850</td>
<td>1.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>2.</td>
<td>+062250+4202151</td>
<td>2.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+400010+4100010</td>
<td>3.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>4.</td>
<td>+420214+3600000</td>
<td>4.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+530070+7700900</td>
<td>5.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>6.</td>
<td>+640230+6504000</td>
<td>6.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>7.</td>
<td>+6504100+6504200</td>
<td>7.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>Command Test Routine Printout (Cont)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LO3000</strong></td>
<td>+0012250+6567651</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+0000000-0001000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+6010000+3610000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+0000000+5710000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+0000010+0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LO3100</strong></td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LO3200</strong></td>
<td>+0000000-1161000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>-5263431+0370170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+2420000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+0000000-000120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+0010170-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LO3300</strong></td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>-0000000-0000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LO3400</strong></td>
<td>+1235210-7325010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+4502640+714761</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+7563241+6321761</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+6574130-5746210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+1563241-3736310</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+3514340+5321721</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+6401650-7503450</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+6324511+5417510</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LO3500</strong></td>
<td>+1235210-7325010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+4502640+714761</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+7563241+6321761</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+6574130-5746210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+1563241-3736310</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+3514340+5321721</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+6401650-7503450</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+6324511+5417510</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** The above table represents a portion of the command test routine printout and includes hexadecimal values and operations. The numbers are likely part of a larger data set, possibly related to computer programming or electronic testing. The specific context or application is not provided in the image.
<table>
<thead>
<tr>
<th>Command Test Routine Printout (Cont)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L04400</strong> -0000000-0000000</td>
</tr>
<tr>
<td>1. -0000000-0000000</td>
</tr>
<tr>
<td>2. -0000000-0000000</td>
</tr>
<tr>
<td>3. -0000000-0000000</td>
</tr>
<tr>
<td>4. -0000000-0000000</td>
</tr>
<tr>
<td>5. -0000000-0000000</td>
</tr>
<tr>
<td>6. -0000000-0000000</td>
</tr>
<tr>
<td>7. -0000000-0000000</td>
</tr>
<tr>
<td><strong>L04500</strong> -0000000-0000000</td>
</tr>
<tr>
<td>1. -0000000-0000000</td>
</tr>
<tr>
<td>2. -0000000-0000000</td>
</tr>
<tr>
<td>3. -0000000-0000000</td>
</tr>
<tr>
<td>4. -0000000-0000000</td>
</tr>
<tr>
<td>5. -0000000-0000000</td>
</tr>
<tr>
<td>6. -0000000-0000000</td>
</tr>
<tr>
<td>7. -0000000-0000000</td>
</tr>
<tr>
<td><strong>L04600</strong> -0000000-0000000</td>
</tr>
<tr>
<td>1. -0000000-0000000</td>
</tr>
<tr>
<td>2. -0000000-0000000</td>
</tr>
<tr>
<td>3. -0000000-0000000</td>
</tr>
<tr>
<td>4. -0000000-0000000</td>
</tr>
<tr>
<td>5. -0000000-0000000</td>
</tr>
<tr>
<td>6. -0000000-0000000</td>
</tr>
<tr>
<td>7. -0000000-0000000</td>
</tr>
<tr>
<td><strong>L04700</strong> -0000000-0000000</td>
</tr>
<tr>
<td>1. -0000000-0000000</td>
</tr>
<tr>
<td>2. -0000000-0000000</td>
</tr>
<tr>
<td>3. -0000000-0000000</td>
</tr>
<tr>
<td>4. -0000000-0000000</td>
</tr>
<tr>
<td>5. -0000000-0000000</td>
</tr>
<tr>
<td><strong>L05000</strong> +0012250+5567651</td>
</tr>
<tr>
<td>1. +0024521+5357530</td>
</tr>
<tr>
<td>2. -0012250+5567651</td>
</tr>
<tr>
<td>3. +5125651+3726000</td>
</tr>
<tr>
<td>4. +0000000-0000741</td>
</tr>
<tr>
<td>5. +6676471+4346000</td>
</tr>
<tr>
<td>6. -0000000+570740</td>
</tr>
<tr>
<td>7. -5125651+3726000</td>
</tr>
<tr>
<td><strong>L05100</strong> +0000000-0000741</td>
</tr>
<tr>
<td>1. +5243261+4000000</td>
</tr>
<tr>
<td>2. +000000-1161671</td>
</tr>
<tr>
<td>3. +000011-5321471</td>
</tr>
<tr>
<td>4. +23211130-5123341</td>
</tr>
<tr>
<td>5. +000000-5321471</td>
</tr>
<tr>
<td>6. -0000000-0000000</td>
</tr>
<tr>
<td>7. -0000000-0000440</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
TRANSFER COMMANDS TEST ROUTINE

Purpose

To test correct operation of computer in following and ignoring all transfer commands.

Operating Instructions

1. Enter and verify Transfer Commands Test routine with tape reader.
2. Place Operation switch in Continuous position.
3. Place Sense Switch B up and Sense Switches C and D down.
4. Depress Start 3 button (first routine loop stops automatically; loop can be repeated by again depressing Start 3 button - see Comments).
5. Place Sense Switches B, C, and D down.
6. Depress Start button (second routine loop routine runs continuously for a period of time, then automatically enters third routine loop; third loop runs continuously for a period of time, then halts automatically - see Comments).
7. Place Sense Switches B, C, and D up.
8. Depress Start button (fourth routine loop runs continuously for a period of time, then automatically enters fifth routine loop; after routine has run in fifth loop for a short while, halt computer by depressing Stop button - see Comments).

Comments

First routine loop tests the HTR command; computer should halt at location 0003.0 and display seven decimal points on visual readout; error in operation is indicated by display on visual readout of +0000000-0000040 or any operating sequence other than as specified.

Second routine loop tests TRA command; visual readout display should continuously change, starting at +400000-000000 and counting down to +0010000-0000000; error is indicated by any operating sequence other than as specified or any of the halts and displays as given below:

<table>
<thead>
<tr>
<th>Halt at Location</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>0011.1</td>
<td>+0000000-0000141</td>
</tr>
<tr>
<td>0015.1</td>
<td>+0000000-0000151</td>
</tr>
<tr>
<td>0020.0</td>
<td>+0000000-0000200</td>
</tr>
<tr>
<td>0023.0</td>
<td>+0000000-0000230</td>
</tr>
</tbody>
</table>
Third Routine Loop Tests The Following:

Transfer on TSB, TSC, TSD with sense switches in down position.
Transfer on TMI with a negative number in the accumulator.
Transfer ignored on TPL with a negative number in the accumulator
Transfer ignored on TOV.
Transfer to Trapping Mode on a negative command.

Visual readout display should continuously change, starting at
-40000000-0000000 and counting down to -0010000-0000000; after displaying last number computer should halt at location 0041.0; error is indicated by any operating sequence other than as specified or any of the halts and displays as given below:

<table>
<thead>
<tr>
<th>Halt at Location</th>
<th>Display</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0027.1</td>
<td>+0001000+7700271</td>
<td>Ignoring TSB command instead of executing it</td>
</tr>
<tr>
<td>0031.0</td>
<td>+0001000+7700310</td>
<td>Ignoring TSC command instead of executing it</td>
</tr>
<tr>
<td>0032.1</td>
<td>+0001000+7700321</td>
<td>Ignoring TSD command instead of executing it</td>
</tr>
<tr>
<td>0036.1</td>
<td>+0001000+7700361</td>
<td>Ignoring TMI command instead of executing it</td>
</tr>
<tr>
<td>0035.1</td>
<td>+0001000+7700351</td>
<td>Executing TPL command instead of ignoring it</td>
</tr>
<tr>
<td>0035.0</td>
<td>+0001000+7700350</td>
<td>Executing TOV command instead of ignoring it</td>
</tr>
<tr>
<td>0026.1</td>
<td>+0000000-0000261</td>
<td>Ignoring negative sign of command instead of entering Trapping Mode</td>
</tr>
</tbody>
</table>

Fourth Routine Loop Tests The Following:

Ignoring TSB, TSC, TSD with sense switches in up position.
Ignoring TMI and TZE with a positive, non-zero number in accumulator.
Transfer on TPL with a positive, non-zero number in accumulator.
Visual readout display should continuously change, starting at -0000000-0000000 and counting to +7770000-0000000; after displaying last number, computer should automatically enter fifth routine loop; error is indicated by any operating sequence other than as specified or any of the halts and displays as given below:

<table>
<thead>
<tr>
<th>Halt at Location</th>
<th>Display</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0041.0</td>
<td>+0001000+7700410</td>
<td>Executing TSB command instead of ignoring it</td>
</tr>
<tr>
<td>0041.1</td>
<td>+0001000+7700411</td>
<td>Executing TSC command instead of ignoring it</td>
</tr>
<tr>
<td>0042.0</td>
<td>+0001000+7700420</td>
<td>Executing TSD command instead of ignoring it</td>
</tr>
<tr>
<td>0043.0</td>
<td>+0001000+7700430</td>
<td>Executing TMI command instead of ignoring it</td>
</tr>
<tr>
<td>0045.0</td>
<td>+0001000+7700450</td>
<td>Executing TZB command instead of ignoring it</td>
</tr>
<tr>
<td>0045.1</td>
<td>+0001000+7700451</td>
<td>Ignoring TPL command instead of executing it</td>
</tr>
</tbody>
</table>

Fifth Routine Loop Tests The Following:

Transfer on TOV when overflow occurs.
Transfer on TZE with a zero in the accumulator.

Overflow error indicator on control console should flash on and off; visual readout should display a positive zero; error is indicated by any operating sequence other than as specified or any of the halts and displays as given below:

<table>
<thead>
<tr>
<th>Halt at Location</th>
<th>Display</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0051.0</td>
<td>+0001000+7700510</td>
<td>Ignoring TOV command instead of executing it</td>
</tr>
<tr>
<td>0053.1</td>
<td>+0001000+7700531</td>
<td>Ignoring TZE command instead of executing it</td>
</tr>
</tbody>
</table>
## Transfer Commands Test Routine Printout

| LO0000 | +1501000+4200071 | LO0500 | +0100650+0100650 |
|        | +0300600+5000060 | 1.     | +5300521+1501000  |
| 2.     | +3600700+5700070 | 2.     | +5700001+6001040  |
| 3.     | +5400100+3600611 | 3.     | +3601040+5000500  |
| 4.     | +7700030+3606630 | 4.     | +1501000+5700001  |
| 5.     | +7700030+7700030 | 5.     | -0000000-0000000 |
| 6.     | +0001000+5700271 | 6.     | -0000000-0000000 |
| 7.     | +0001000+7700000 | 7.     | -0000000-0000000 |

| L00100 | +0000050+6001010 | L00600 | +0000000-0000261 |
| 1.     | +0001010+5700130 | 1.     | +6314630-6317000 |
| 2.     | +3600720+7700111 | 2.     | +0000000-0000040 |
| 3.     | +1501000+0100640 | 3.     | +0010000-0000000 |
| 4.     | +4201514+3601010 | 4.     | +0000000-0000060 |
| 5.     | +0001000+5700171 | 5.     | -0000000-0000000 |
| 6.     | +3600730+7700151 | 6.     | -0000000-0000000 |
| 7.     | +7700151+0300630 | 7.     | -0000000-0000000 |

| L00200 | +7002111+3600740 | L00700 | -0000000-0000000 |
| 1.     | +7700200+0001010 | 1.     | -0000000-0000000 |
| 2.     | +5000270+3300360 | 2.     | +0000000-0000011 |
| 3.     | +7001100+3600750 | 3.     | +0000000-0000151 |
| 4.     | +7700230+0000000 | 4.     | +0000000-0000200 |
| 5.     | -0000000-0000000 | 5.     | +0000000-0000230 |
| 6.     | +0200650+3600600 | 6.     | +0000000-0000000 |
| 7.     | +5700260+5400310 | 7.     | -0000000-0000000 |

| L00300 | +1501000+5700001 | L00800 | -0000000-0000000 |
| 1.     | +5000321+1501000 | 1.     | -0000000-0000000 |
| 2.     | +5700001+500310 | 2.     | +0000000-0000011 |
| 3.     | +1501000+5700001 | 3.     | +0000000-0000151 |
| 4.     | +3501000+0100630 | 4.     | +0000000-0000200 |
| 5.     | +5000073+5200000 | 5.     | +0000000-0000230 |
| 6.     | +5000401+5100261 | 6.     | +0000000-0000000 |
| 7.     | +1501000+5700001 | 7.     | -0000000-0000000 |

| L00400 | +7700371+7700410 | L00900 | -0000000-0000000 |
| 1.     | +5400000+5500000 | 1.     | -0000000-0000000 |
| 2.     | +5500000+0100630 | 2.     | +0000000-0000011 |
| 3.     | +5000073+5200000 | 3.     | +0000000-0000151 |
| 4.     | +5000401+5100261 | 4.     | +0000000-0000200 |
| 5.     | +1501000+5700001 | 5.     | +0000000-0000230 |
| 6.     | -0000000-0000750 | 6.     | +0000000-0000000 |
| 7.     | -0000000-0000000 | 7.     | -0000000-0000000 |
GENERAL MACHINE TEST ROUTINE 1

PURPOSE

To test for proper execution of miscellaneous commands in RECOMP II instruction repertoire, particularly the logic commands.

OPERATING INSTRUCTIONS

1. Enter and verify General Machine Test Routine 1 with tape reader.
   2. Prepare typewriter for operation.
   3. Depress Start 3 button (computer should operate about ten minutes, then halt at location 0004.0 - see Comments).
   4. When last display is observed, depress Stop button on control console.

COMMENTS

Following numbers should be displayed in order as computer executes program:

+2525250+2525250
-0025250-0025250
+0000000-0000000
+0000000-0001650
+2525250+2525250
+2525250+2525240
+2525250+2525200
etc, losing one bit at a time and ending at
+0000000-0000000

Error is indicated by typeout of first location of the test which failed or any operating sequence other than as specified.

The General Machine Test Routine I Printout is shown on the subsequent pages.
<table>
<thead>
<tr>
<th>LO0000</th>
<th>+1503770+3303700</th>
<th>LO0600</th>
<th>+4000000+6402700</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>+0303710+5200031</td>
<td>1.</td>
<td>+0077600+0302700</td>
</tr>
<tr>
<td>2.</td>
<td>+1013710+5703000</td>
<td>2.</td>
<td>+5000060+3000000</td>
</tr>
<tr>
<td>3.</td>
<td>+5700040+7700031</td>
<td>3.</td>
<td>+00000610+0100730</td>
</tr>
<tr>
<td>4.</td>
<td>+0002700+6004000</td>
<td>4.</td>
<td>+6000610+0300740</td>
</tr>
<tr>
<td>5.</td>
<td>+0004000+6004010</td>
<td>5.</td>
<td>+5100610+4600000</td>
</tr>
<tr>
<td>6.</td>
<td>+5000000+0304010</td>
<td>6.</td>
<td>+3002000+6000010</td>
</tr>
<tr>
<td>7.</td>
<td>+4000000+3604010</td>
<td>7.</td>
<td>+6504000+6404000</td>
</tr>
<tr>
<td>LO0100</td>
<td>+5000110+3000000</td>
<td>LO0700</td>
<td>+0000670+0100760</td>
</tr>
<tr>
<td>1.</td>
<td>+0000570+0100150</td>
<td>1.</td>
<td>+0000670+0300770</td>
</tr>
<tr>
<td>2.</td>
<td>+6000050+4200061</td>
<td>2.</td>
<td>+5100610+5701000</td>
</tr>
<tr>
<td>3.</td>
<td>+4200071+0300160</td>
<td>3.</td>
<td>+0000010-000010</td>
</tr>
<tr>
<td>4.</td>
<td>+5100050+700200</td>
<td>4.</td>
<td>+0077000-000000</td>
</tr>
<tr>
<td>5.</td>
<td>+0000010-000010</td>
<td>5.</td>
<td>+0077000-0302700</td>
</tr>
<tr>
<td>6.</td>
<td>+0004771-000000</td>
<td>6.</td>
<td>+0001000-000010</td>
</tr>
<tr>
<td>7.</td>
<td>-0000000-000010</td>
<td>7.</td>
<td>+6512000-000000</td>
</tr>
<tr>
<td>LO0200</td>
<td>+6002700+6004000</td>
<td>LO1000</td>
<td>+4000000+6602700</td>
</tr>
<tr>
<td>1.</td>
<td>+0004000+1504010</td>
<td>1.</td>
<td>+0077000+0302700</td>
</tr>
<tr>
<td>2.</td>
<td>+1500330+0304010</td>
<td>2.</td>
<td>+5001030-3000000</td>
</tr>
<tr>
<td>3.</td>
<td>+5000241-3000000</td>
<td>3.</td>
<td>+001010+011130</td>
</tr>
<tr>
<td>4.</td>
<td>+4000000+3604010</td>
<td>4.</td>
<td>+6001010+031140</td>
</tr>
<tr>
<td>5.</td>
<td>+000210+010310</td>
<td>5.</td>
<td>+5101010+4600000</td>
</tr>
<tr>
<td>6.</td>
<td>+600210+420021</td>
<td>6.</td>
<td>+0001150+6001010</td>
</tr>
<tr>
<td>7.</td>
<td>+4200241+030320</td>
<td>7.</td>
<td>+6704000+6604000</td>
</tr>
<tr>
<td>LO0300</td>
<td>+5100210+5704000</td>
<td>LO1100</td>
<td>+0001070+0101160</td>
</tr>
<tr>
<td>1.</td>
<td>+0000010-000010</td>
<td>1.</td>
<td>+6001070+0301170</td>
</tr>
<tr>
<td>2.</td>
<td>+0004770-000000</td>
<td>2.</td>
<td>+5101010+5701200</td>
</tr>
<tr>
<td>3.</td>
<td>-0000000-000000</td>
<td>3.</td>
<td>+0000010-00010</td>
</tr>
<tr>
<td>4.</td>
<td>-0000000-000000</td>
<td>4.</td>
<td>+0100000-000000</td>
</tr>
<tr>
<td>5.</td>
<td>-0000000-000000</td>
<td>5.</td>
<td>+0077000+0302700</td>
</tr>
<tr>
<td>6.</td>
<td>-0000000-000000</td>
<td>6.</td>
<td>+0001000-000100</td>
</tr>
<tr>
<td>7.</td>
<td>-0077771-0077771</td>
<td>7.</td>
<td>+6712000-000000</td>
</tr>
<tr>
<td>LO0400</td>
<td>+3000360+6004000</td>
<td>LO1200</td>
<td>+3002700+3504000</td>
</tr>
<tr>
<td>1.</td>
<td>+0000400+0100550</td>
<td>1.</td>
<td>+3004000+0302700</td>
</tr>
<tr>
<td>2.</td>
<td>+4200401+0300570</td>
<td>2.</td>
<td>+5001230-3000000</td>
</tr>
<tr>
<td>3.</td>
<td>+5100400+0002700</td>
<td>3.</td>
<td>+4300000+0302710</td>
</tr>
<tr>
<td>4.</td>
<td>+3300370+6004000</td>
<td>4.</td>
<td>+5001250-3000000</td>
</tr>
<tr>
<td>5.</td>
<td>+0004000+4204011</td>
<td>5.</td>
<td>+0001210+010130</td>
</tr>
<tr>
<td>6.</td>
<td>+6204010+0304011</td>
<td>6.</td>
<td>+6001210+0301310</td>
</tr>
<tr>
<td>7.</td>
<td>+5000500-3000000</td>
<td>7.</td>
<td>+5101210+5701400</td>
</tr>
<tr>
<td>LO0500</td>
<td>+3604010+0004600</td>
<td>LO1300</td>
<td>+0000020-0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+4200450+0100550</td>
<td>1.</td>
<td>+3004770-0000000</td>
</tr>
<tr>
<td>2.</td>
<td>+5000460+4200500</td>
<td>2.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+4200451+0300560</td>
<td>3.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>4.</td>
<td>+5100450+700600</td>
<td>4.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+0000010-000010</td>
<td>5.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>6.</td>
<td>+4204770-0000000</td>
<td>6.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>7.</td>
<td>+3300370+6004770</td>
<td>7.</td>
<td>-0000000-0000000</td>
</tr>
<tr>
<td>L01400</td>
<td>+3002700+4300000</td>
<td>L02200</td>
<td>+0002700+3302420</td>
</tr>
<tr>
<td>--------+-----------------+--------+-----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+4300000+0302700</td>
<td>1.</td>
<td>+6004000+0002700</td>
</tr>
<tr>
<td>2.</td>
<td>+5001430+3000000</td>
<td>2.</td>
<td>+0400010+010010</td>
</tr>
<tr>
<td>3.</td>
<td>+4300000+0302710</td>
<td>3.</td>
<td>+0304000+5002241</td>
</tr>
<tr>
<td>4.</td>
<td>+5001450+3000000</td>
<td>4.</td>
<td>-3000000+3604000</td>
</tr>
<tr>
<td>5.</td>
<td>+6004000+3604000</td>
<td>5.</td>
<td>+0002400+0302430</td>
</tr>
<tr>
<td>6.</td>
<td>+0001510+0301520</td>
<td>6.</td>
<td>+0602400+0502211</td>
</tr>
<tr>
<td>7.</td>
<td>+6001510+5201400</td>
<td>7.</td>
<td>+0002410+6002400</td>
</tr>
</tbody>
</table>

| L01500 | +0001520+5701650 | L02300 | +0002220+0102440 |
|--------+-----------------+--------+-----------------|
| 1.     | +7700000-0000000 | 1.     | +0602220+6002340|
| 2.     | +0100000-0000000 | 2.     | +0302450+5202500|
| 3.     | -0000000-0000000 | 3.     | +0302420+4000000|
| 4.     | -0000000-0000000 | 4.     | +0400010+410010 |
| 5.     | -0000000-0000000 | 5.     | +0602400+5702200|
| 6.     | -0000000-0000000 | 6.     | -0000000-0000000|
| 7.     | -0000000-0000000 | 7.     | -0000000-0000000|

| L01600 | +1577770+6004000 | L02400 | +7700000-0000000 |
|--------+-----------------+--------+-----------------|
| 1.     | +0301660+5001621 | 1.     | +7700000-0000000|
| 2.     | -3000000+3604000 | 2.     | +7777771+7777770|
| 3.     | +0001670+0301700 | 3.     | +0100000-0000000|
| 4.     | +6001670+4000000 | 4.     | +000010+000010 |
| 5.     | +5201600+5702000 | 5.     | +0400050+0000000|
| 6.     | +0000000+0001650 | 6.     | -0000000-0000000|
| 7.     | +7700000-0000000 | 7.     | -0000000-0000000|

| L01700 | +0100000-0000000 | L02500 | +3002700+4500000 |
|--------+-----------------+--------+-----------------|
| 1.     | -0000000-0000000 | 1.     | +3504000+4000010|
| 2.     | -0000000-0000000 | 2.     | +4500000+4300000|
| 3.     | -0000000-0000000 | 3.     | +0304010+0102650|
| 4.     | -0000000-0000000 | 4.     | +0502550+3000000|
| 5.     | -0000000-0000000 | 5.     | +4300030+0304000|
| 6.     | -0000000-0000000 | 6.     | +0400000+400010 |
| 7.     | -0000000-0000000 | 7.     | +0502500+3000000|

| L02000 | +0002070+3302700 | L02600 | +0002510+0102660 |
|--------+-----------------+--------+-----------------|
| 1.     | +3302700+0302700 | 1.     | +6002510+4202561|
| 2.     | +5002030+0300000 | 2.     | +4000010+420251 |
| 3.     | +3302700+500291  | 3.     | +4100010+0302670|
| 4.     | -3000000+0002100 | 4.     | +5102500+7700040|
| 5.     | +0302110+6002100 | 5.     | +0000300-000001 |
| 6.     | +5202000+5702200 | 6.     | +0000000-000010 |
| 7.     | +7777771+777771  | 7.     | +3504000+4000470|

<p>| L02100 | +7700000-0000000 | L02700 | +2525250+2525250 |
|--------+-----------------+--------+-----------------|
| 1.     | +0100000-0000000 | 1.     | +2525250+2525250|
| 2.     | -0000000-0000000 | 2.     | +2525250+2525250|
| 3.     | -0000000-0000000 | 3.     | +2525250+2525250|
| 4.     | -0000000-0000000 | 4.     | +2525250+2525250|
| 5.     | -0000000-0000000 | 5.     | +2525250+2525250|
| 6.     | -0000000-0000000 | 6.     | +2525250+2525250|
| 7.     | -0000000-0000000 | 7.     | +2525250+2525250|</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L03000</strong></td>
<td><strong>L03600</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+4000030+0103720</td>
<td>1.</td>
<td>+0000001+7777771</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+4203041+0003730</td>
<td>2.</td>
<td>+0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+5703200+7200060</td>
<td>3.</td>
<td>+6004100+0077620</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+7200140+7200040</td>
<td>4.</td>
<td>+6004110+0077630</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+4000000+0000000</td>
<td>5.</td>
<td>+6004120+0077640</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+5703200+0003040</td>
<td>6.</td>
<td>+6004130+0077650</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+103740+4203041</td>
<td>7.</td>
<td>+6004140+0077660</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L03100</td>
<td>L03700</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+5703041+0003750</td>
<td>1.</td>
<td>+0000000+0002410</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+5703200+7200150</td>
<td>2.</td>
<td>+0000000+0013000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+4000000+0103760</td>
<td>3.</td>
<td>-0370250+5302420</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+4203151+4000000</td>
<td>4.</td>
<td>+0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+4000000+1213361</td>
<td>5.</td>
<td>-0200221+0044540</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+7703160+0000000</td>
<td>6.</td>
<td>+0000000+0000261</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+7703160+0000930</td>
<td>7.</td>
<td>-0200221+0044540</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L03200</td>
<td>L13000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+1503770+0103610</td>
<td>1.</td>
<td>-0054140-2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+4203321+0003770</td>
<td>2.</td>
<td>-0054010+2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+4000120+4100910</td>
<td>3.</td>
<td>-0120220-0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+4203250+3303600</td>
<td>4.</td>
<td>-0054011-2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+400050+4000000</td>
<td>5.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+7200040+5003261</td>
<td>6.</td>
<td>-2164110+5547620</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+5703230+0003770</td>
<td>7.</td>
<td>-0163310-2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L03300</td>
<td>L13100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+4100170+4203310</td>
<td>1.</td>
<td>-0163370-2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+7202220+7211300</td>
<td>2.</td>
<td>-0151300+2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+4000000+4570311</td>
<td>3.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+6003210+6403021</td>
<td>4.</td>
<td>-0350650+2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+6003330+0103430</td>
<td>5.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+6003330+5703111</td>
<td>6.</td>
<td>-0050761+2017270</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+3677600+7703000</td>
<td>7.</td>
<td>-0240660+5011000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L03400</td>
<td>L13200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+0000010+0000000</td>
<td>1.</td>
<td>-0017300-2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+0077071+0077071</td>
<td>2.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+0000000+0000000</td>
<td>3.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+6004160+5703171</td>
<td>4.</td>
<td>-0350650+2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+3677610+7703060</td>
<td>5.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+3604200+7703000</td>
<td>6.</td>
<td>-0050761+2017270</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+0077610+5707610</td>
<td>7.</td>
<td>-0240660+5011000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L03500</td>
<td>L13200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+0077071+0077071</td>
<td>1.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+6004100+0077620</td>
<td>2.</td>
<td>-0032421+5547620</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+6004111+0077630</td>
<td>3.</td>
<td>-0034421+2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+6004120+0077640</td>
<td>4.</td>
<td>-0153160+2000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+6004134+0077650</td>
<td>5.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+6004140+0077660</td>
<td>6.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+6004150+0077670</td>
<td>7.</td>
<td>-0000000+0000000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
General Machine Test Routine I Printout (Cont)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L13600</td>
<td>-0000000-0000000</td>
<td>1.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GENERAL MACHINE TEST II

Purpose

To test for proper operation of all arithmetical commands.

Operating Instructions

Explanations of the arithmetic routines located in the memory are discussed in the following table.

<table>
<thead>
<tr>
<th>Sense Switch</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Up; display.</td>
</tr>
<tr>
<td>B</td>
<td>Down; type.</td>
</tr>
<tr>
<td>C</td>
<td>Up; leapfrog.</td>
</tr>
<tr>
<td>C</td>
<td>Down; New constants after running 12 tests.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>Beginning of type-out linkage.</td>
</tr>
<tr>
<td>00010</td>
<td>Initializing: depress START 1.</td>
</tr>
<tr>
<td>00020</td>
<td>Type-out linkage.</td>
</tr>
<tr>
<td>through</td>
<td></td>
</tr>
<tr>
<td>00070</td>
<td></td>
</tr>
<tr>
<td>00200</td>
<td>Pseudo-random constants generator:</td>
</tr>
<tr>
<td></td>
<td>Constants are generated by routine in location 0034-0045, and modified and stored by location 0012-0031, given proper format to prevent overflow, i.e., constants used in square root tests are positive.</td>
</tr>
<tr>
<td>00320</td>
<td>If SENSE switch &quot;B&quot; is down, constants will be typed out. (L00331 is exit.)</td>
</tr>
<tr>
<td>00340</td>
<td>Constants generator: constants are stored as follows:</td>
</tr>
<tr>
<td></td>
<td>Fixed point, for tests other than square root (test 8), L7770 and 7771 (i.e. two constants).</td>
</tr>
</tbody>
</table>
Floating point, except square root: L7772 and 7773 (mantissa and exponent, first constant); L7774 and 7775 (mantissa and exponent, second constant).

Square root, always positive. Fixed point, L7654; floating point, L7655 and 7656.

Types out constants in command format and initializes the alphanumeric type-out loop for labeling constants.

Types out constants in command format and initializes the alphanumeric type-out loop for labeling constants.

Floating point, except square root: L7772 and 7773 (mantissa and exponent, first constant); L7774 and 7775 (mantissa and exponent, second constant).

Square root, always positive. Fixed point, L7654; floating point, L7655 and 7656.

Types out constants in command format and initializes the alphanumeric type-out loop for labeling constants.

Types out constants in command format and initializes the alphanumeric type-out loop for labeling constants.

BCD labels used to identify tests in error type-out or display.

Test 1: Note that tests are "leapfrogged" through the memory, so that this test will subsequently be located at location 02000, 03000, 04000, etc., to 74000, after which it will be returned to 01000.

Test 2: Also note that in all fixed point tests the accumulator should contain a + or - zero immediately before the command "+5077000." If not, the program jumps to 77030 and types or displays the error.

Test 3

Test 4

Test 5

Test 6

Instruction beginning at L76520 will cause a jump to Test 7, which begins at sector 00 of the next channel.

Contains labels to be typed out in alpha for the constants generator routine.
Test 7

Test 8

Test 9. Tests 9 through 12 are floating point tests. In these cases it is possible to obtain a "1" in bit position 1, instead of a zero, due to normalization of round-off. If this occurs, the program will test the mantissa; anything but a normalized "1" will cause indication of a mantissa error. If it is a normalized "1," the exponent will be tested. It should differ from the exponent of the first floating point constant by exactly 458 or 3710 as a result of denormalization and normalization. Different values will cause the indication of floating point exponent errors.

Test 10

Test 11

Test 12

Jump to L7600, which links to the constants generator when SENSE switch "C" is down and to the leapfrog loop when that switch is up. If "C" is up, jump to constants generator. If not, continue through leapfrog.
This page intentionally left blank
General Machine Test Routine II Printout

L0000  +1577670+570020  L00600  +5241571+0000000
  1.  +5700111+0000000  1.  +5242571+0000000
  2.  +5777600+6400000  2.  +5243571+0000000
  3.  +5777631+4277651  3.  +5244571+0000000
  4.  +5777660+0000000  4.  +5245571+0000000
  5.  +4000240+277261  5.  +5246571+0000000
  6.  +0077650+777240  6.  +5247571+0000000
  7.  +0077650+777240  7.  +5250571+0000000

L00100 +3377540+0376710  L00700  +5251571+0000000
  1.  +4200331+5700340  1.  +5020571+0000000
  2.  +607700+5700340  2.  +5021571+0000000
  3.  +6077110+0177700  3.  +5022571+0000000
  4.  +5300150+5700160  4.  +3001710+1276550
  5.  +0377710+5700000  5.  +3001730+1276560
  6.  +5700340+4500000  6.  +7200370+5700330
  7.  +607720+5700340  7.  +0000000+000001

L00200 +3376640+5077730  L01000  +0077700+0177710
  1.  +5700340+5076550  1.  +0377710+0377700
  2.  +577740+5076550  2.  +5577100+0607760
  3.  +5376640+5077750  3.  +0000600+5777030
  4.  +5700340+5076540  4.  +0010000+0010000
  5.  +5200261+0276540  5.  +0010000+0010000
  6.  +6076540+5700340  6.  +0006000+5777030
  7.  +6076550+500310  7.  +0006000+5777030

L00300 +0276550+5076550  L01100  +0277700+0177710
  1.  +0077730+6076560  1.  +5077000+0607660
  2.  +5400460+7200370  2.  +0006100+5777030
  3.  +7200100+5701000  3.  +0006100+5777030
  4.  +1577650+0107700  4.  +0006100+5777030
  5.  +4200450+0076570  5.  +0006100+5777030
  6.  +1776600+3376610  6.  +0006100+5777030
  7.  +6076620+0076600  7.  +0006100+5777030

L00400 +6076570+1300000  L01200  +0077700+1177710
  1.  +100250+0176620  1.  +607760+2277710
  2.  +6077630+6076660  2.  +0377700+5077700
  3.  +3376650+5000450  3.  +607760+0000620
  4.  +0077630+5000450  4.  +5777030+0000000
  5.  +0277630+500311  5.  +5777030+0000000
  6.  -3001610-3001630  6.  +5777030-0000000
  7.  +1277700-3001650  7.  +5777030-0000000

L00500 +1277710-3001670  L01300  +0077700+1377710
  1.  -3001710+1277720  1.  +0377760+3376670
  2.  -3001730+1277730  2.  +5077000+0607660
  3.  -3001670+3001710  3.  +000630+5777030
  4.  +1277740+3001730  4.  +000630+5777030
  5.  +1277750+3001750  5.  +000630+5777030
  6.  -3001630+1276540  6.  +000630+5777030
  7.  -3001670+5700740  7.  +000630+5777030
<table>
<thead>
<tr>
<th>LO1400</th>
<th>+0077700+1177710</th>
<th>LO2200</th>
<th>+3077720+0477720</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>+23777710+03777700</td>
<td>1.</td>
<td>+0677720+0677720</td>
</tr>
<tr>
<td>2.</td>
<td>+5077000+06777600</td>
<td>2.</td>
<td>+5077000+35777620</td>
</tr>
<tr>
<td>3.</td>
<td>+0000640+5777030</td>
<td>3.</td>
<td>+0000700+5777340</td>
</tr>
<tr>
<td>4.</td>
<td>+0000640+5777030</td>
<td>4.</td>
<td>+0000700+5777340</td>
</tr>
<tr>
<td>5.</td>
<td>+0000640+5777030</td>
<td>5.</td>
<td>+0000700+5777340</td>
</tr>
<tr>
<td>6.</td>
<td>+0000640+5777030</td>
<td>6.</td>
<td>+0000700+5777340</td>
</tr>
<tr>
<td>7.</td>
<td>+0000640+5777030</td>
<td>7.</td>
<td>+0000700+5777340</td>
</tr>
<tr>
<td>LO1500</td>
<td>+0077700+1177710</td>
<td>LO2300</td>
<td>+3477720+0477720</td>
</tr>
<tr>
<td>1.</td>
<td>+2077710+06777700</td>
<td>1.</td>
<td>+5077000+35777620</td>
</tr>
<tr>
<td>2.</td>
<td>+3076670+0077760</td>
<td>2.</td>
<td>+0000710+5777340</td>
</tr>
<tr>
<td>3.</td>
<td>+23777710+03777700</td>
<td>3.</td>
<td>+0000710+5777340</td>
</tr>
<tr>
<td>4.</td>
<td>+5077000+06777600</td>
<td>4.</td>
<td>+0000710+5777340</td>
</tr>
<tr>
<td>5.</td>
<td>+0000650+5777030</td>
<td>5.</td>
<td>+0000710+5777340</td>
</tr>
<tr>
<td>6.</td>
<td>+0000650+5777030</td>
<td>6.</td>
<td>+0000710+5777340</td>
</tr>
<tr>
<td>7.</td>
<td>+0000650+5777030</td>
<td>7.</td>
<td>+0000710+5777340</td>
</tr>
<tr>
<td>LO1600</td>
<td>+5777520+0000300</td>
<td>LO2400</td>
<td>+3077720+0777740</td>
</tr>
<tr>
<td>1.</td>
<td>-0000171-7303021</td>
<td>1.</td>
<td>+0577740+0677720</td>
</tr>
<tr>
<td>2.</td>
<td>-0000100-1614021</td>
<td>2.</td>
<td>+5077000+35777620</td>
</tr>
<tr>
<td>3.</td>
<td>-0000400-6467220</td>
<td>3.</td>
<td>+0000720+5777340</td>
</tr>
<tr>
<td>4.</td>
<td>-0001300+0332040</td>
<td>4.</td>
<td>+0000720+5777340</td>
</tr>
<tr>
<td>5.</td>
<td>-0001711-4332040</td>
<td>5.</td>
<td>+0000720+5777340</td>
</tr>
<tr>
<td>6.</td>
<td>-0000800-0000000</td>
<td>6.</td>
<td>+0000720+5777340</td>
</tr>
<tr>
<td>7.</td>
<td>-0000171-4154500</td>
<td>7.</td>
<td>+0000720+5777340</td>
</tr>
<tr>
<td>LO1700</td>
<td>-0000020+3206640</td>
<td>LO2500</td>
<td>+4776500+3577760</td>
</tr>
<tr>
<td>1.</td>
<td>-00001716033151</td>
<td>1.</td>
<td>+777760+0676550</td>
</tr>
<tr>
<td>2.</td>
<td>-0000040-0000000</td>
<td>2.</td>
<td>+5077000+35777620</td>
</tr>
<tr>
<td>3.</td>
<td>-00001714332171</td>
<td>3.</td>
<td>+0000730+5777340</td>
</tr>
<tr>
<td>4.</td>
<td>-0000001+6665640</td>
<td>4.</td>
<td>+0000730+5777340</td>
</tr>
<tr>
<td>5.</td>
<td>-0000171-4055620</td>
<td>5.</td>
<td>+0000730+5777340</td>
</tr>
<tr>
<td>6.</td>
<td>-0000050+4304020</td>
<td>6.</td>
<td>+0000730+5777340</td>
</tr>
<tr>
<td>7.</td>
<td>-00001714154500</td>
<td>7.</td>
<td>+0000730+5777340</td>
</tr>
<tr>
<td>LO2000</td>
<td>+0077700+1177710</td>
<td>LO2600</td>
<td>+5776000+0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+0077700+1177710</td>
<td>1.</td>
<td>-0000171+6033020</td>
</tr>
<tr>
<td>2.</td>
<td>+2177710+06777700</td>
<td>2.</td>
<td>-0000000+0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+3076670+0077760</td>
<td>3.</td>
<td>-0000171-0755420</td>
</tr>
<tr>
<td>4.</td>
<td>+23777710+03777700</td>
<td>4.</td>
<td>-0000000+0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+5077000+06777600</td>
<td>5.</td>
<td>+1514010+2446550</td>
</tr>
<tr>
<td>6.</td>
<td>+0000660+5777030</td>
<td>6.</td>
<td>+272540+6063460</td>
</tr>
<tr>
<td>7.</td>
<td>+0000660+5777030</td>
<td>7.</td>
<td>+370730+0104620</td>
</tr>
<tr>
<td>LO2100</td>
<td>+2576540+6077770</td>
<td>LO2700</td>
<td>+513711+5406270</td>
</tr>
<tr>
<td>1.</td>
<td>+1377770+0376540</td>
<td>1.</td>
<td>+0672420+7263110</td>
</tr>
<tr>
<td>2.</td>
<td>+3376670+5077000</td>
<td>2.</td>
<td>+1116021+7104130</td>
</tr>
<tr>
<td>3.</td>
<td>+6077600+0000670</td>
<td>3.</td>
<td>+1422561+1132710</td>
</tr>
<tr>
<td>4.</td>
<td>+5777030+0000000</td>
<td>4.</td>
<td>+2030670+5443660</td>
</tr>
<tr>
<td>5.</td>
<td>+0000700+5777340</td>
<td>5.</td>
<td>+2566500+7332350</td>
</tr>
<tr>
<td>6.</td>
<td>+0000700+5777340</td>
<td>6.</td>
<td>+3510551+443210</td>
</tr>
<tr>
<td>7.</td>
<td>+0000700+5777340</td>
<td>7.</td>
<td>+4665341+3331530</td>
</tr>
<tr>
<td>L76000</td>
<td>+1577600+5500100</td>
<td>L76700</td>
<td>+0000000+0000000</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>1.</td>
<td>+0076000+3377540</td>
<td>1.</td>
<td>+0000000+0001000</td>
</tr>
<tr>
<td>2.</td>
<td>+0377550+5176070</td>
<td>2.</td>
<td>+0000000+0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+0076000+3376660</td>
<td>3.</td>
<td>+0000000+00171-7240660</td>
</tr>
<tr>
<td>4.</td>
<td>+4276001+0076130</td>
<td>4.</td>
<td>+0000000+0526620</td>
</tr>
<tr>
<td>5.</td>
<td>+0376600+4276131</td>
<td>5.</td>
<td>+0000000+0301070</td>
</tr>
<tr>
<td>6.</td>
<td>+0775600+6077570</td>
<td>6.</td>
<td>+0000000+1541000</td>
</tr>
<tr>
<td>7.</td>
<td>+6400000+6500000</td>
<td>7.</td>
<td>+0457521+0601530</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L76100</th>
<th>+0076000+0176450</th>
<th>L77000</th>
<th>+1577770+33777130</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>+6076000+3376720</td>
<td>1.</td>
<td>+0176450+4277001</td>
</tr>
<tr>
<td>2.</td>
<td>+5076130+5776070</td>
<td>2.</td>
<td>+3677770+5702500</td>
</tr>
<tr>
<td>3.</td>
<td>+6400000+6500000</td>
<td>3.</td>
<td>+1577610+33777130</td>
</tr>
<tr>
<td>4.</td>
<td>+0076130+0176450</td>
<td>4.</td>
<td>+0000000+4277111</td>
</tr>
<tr>
<td>5.</td>
<td>+0676130+3376720</td>
<td>5.</td>
<td>+5477140+0077610</td>
</tr>
<tr>
<td>6.</td>
<td>+0576710+5776130</td>
<td>6.</td>
<td>+0177120+6077610</td>
</tr>
<tr>
<td>7.</td>
<td>+0077560+5276240</td>
<td>7.</td>
<td>+3677761+7777100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L76200</th>
<th>+0076000+3376660</th>
<th>L77100</th>
<th>+3677600+7777111</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>+4276000+0076130</td>
<td>1.</td>
<td>+7200100+5777500</td>
</tr>
<tr>
<td>2.</td>
<td>+0076600+4276130</td>
<td>2.</td>
<td>+1000000+6317000</td>
</tr>
<tr>
<td>3.</td>
<td>+0377560+0777570</td>
<td>3.</td>
<td>+0000000+8777700</td>
</tr>
<tr>
<td>4.</td>
<td>+0076130+3376460</td>
<td>4.</td>
<td>+3077170+3077210</td>
</tr>
<tr>
<td>5.</td>
<td>+0000204+4200331</td>
<td>5.</td>
<td>+1277611+1277600</td>
</tr>
<tr>
<td>6.</td>
<td>+0076130+3376740</td>
<td>6.</td>
<td>+7200370+5777110</td>
</tr>
<tr>
<td>7.</td>
<td>+4100000+6077630</td>
<td>7.</td>
<td>+0000171+0522540</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L76300</th>
<th>+0076130+3376500</th>
<th>L77200</th>
<th>+0000050+2063020</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>+0076730+0176510</td>
<td>1.</td>
<td>+0000010+0454020</td>
</tr>
<tr>
<td>2.</td>
<td>+6077630+0076070</td>
<td>2.</td>
<td>+0000037+7777711</td>
</tr>
<tr>
<td>3.</td>
<td>+3376470+4100010</td>
<td>3.</td>
<td>+0000000+5700001</td>
</tr>
<tr>
<td>4.</td>
<td>+0676640+0767070</td>
<td>4.</td>
<td>+6477600+5777750</td>
</tr>
<tr>
<td>5.</td>
<td>+3376400+0176640</td>
<td>5.</td>
<td>+010770+4277231</td>
</tr>
<tr>
<td>6.</td>
<td>+0765100+6077640</td>
<td>6.</td>
<td>+0000000+3000000</td>
</tr>
<tr>
<td>7.</td>
<td>+0076000+0076000</td>
<td>7.</td>
<td>+4100000+4277300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L76400</th>
<th>+7700331+5702000</th>
<th>L77300</th>
<th>+7200331+1000500</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>+3076730+1277631</td>
<td>1.</td>
<td>+3377220+5077321</td>
</tr>
<tr>
<td>2.</td>
<td>+3076750+1277641</td>
<td>2.</td>
<td>+5777271+1300000</td>
</tr>
<tr>
<td>3.</td>
<td>+5700321+0000000</td>
<td>3.</td>
<td>+5077231+5777720</td>
</tr>
<tr>
<td>4.</td>
<td>+4276401+5776401</td>
<td>4.</td>
<td>+1577660+3377130</td>
</tr>
<tr>
<td>5.</td>
<td>+0001000+0001000</td>
<td>5.</td>
<td>+0176450+4277501</td>
</tr>
<tr>
<td>6.</td>
<td>+0077000+0000000</td>
<td>6.</td>
<td>+3077620+3377740</td>
</tr>
<tr>
<td>7.</td>
<td>+0077000+0000000</td>
<td>7.</td>
<td>+5077420+6077600</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L76500</th>
<th>+0007000+0000000</th>
<th>L77400</th>
<th>+5077420+6077600</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>+5000531+0000000</td>
<td>1.</td>
<td>+0077660+5777030</td>
</tr>
<tr>
<td>2.</td>
<td>+1577600+3377540</td>
<td>2.</td>
<td>+4300000+0177520</td>
</tr>
<tr>
<td>3.</td>
<td>+0767700+5776410</td>
<td>3.</td>
<td>+0377730+5077501</td>
</tr>
<tr>
<td>4.</td>
<td>+6723150+5377721</td>
<td>4.</td>
<td>+6077600+0077530</td>
</tr>
<tr>
<td>5.</td>
<td>+5522311+6663471</td>
<td>5.</td>
<td>+5477530+0077660</td>
</tr>
<tr>
<td>6.</td>
<td>-0000000-0000000</td>
<td>6.</td>
<td>+5777030+0000000</td>
</tr>
<tr>
<td>7.</td>
<td>+5547170+1644760</td>
<td>7.</td>
<td>+3177771+7777771</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L76600</th>
<th>+0550040+5327774</th>
<th>L77500</th>
<th>+4000000+5702300</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>+0000000+7777771</td>
<td>1.</td>
<td>+3002510+5777410</td>
</tr>
<tr>
<td>2.</td>
<td>+0000000+5327774</td>
<td>2.</td>
<td>+0000000+0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+3105231+0024020</td>
<td>3.</td>
<td>+0000000+0000000</td>
</tr>
<tr>
<td>4.</td>
<td>+0000000+000371</td>
<td>4.</td>
<td>+0000000+0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+0001000+0000000</td>
<td>5.</td>
<td>+0000000+0076000</td>
</tr>
<tr>
<td>6.</td>
<td>+0074000+0074000</td>
<td>6.</td>
<td>+0075000+0075000</td>
</tr>
<tr>
<td>7.</td>
<td>+7777771+7777770</td>
<td>7.</td>
<td>+7777771+7777771</td>
</tr>
</tbody>
</table>
TYPE CHARACTER AND PUNCH/TYPE CHARACTER TEST ROUTINE

Purpose

To check proper operation of typewriter and punch under computer control with Type Character and Punch and Type Character commands.

Operating Instructions

1. Fill and verify Fill Memory with Random Numbers routine.
2. Set location counter to 7757.0 and depress Start button and allow routine entered in Step 1 to run until it stops.
3. Fill and verify Type Character and Punch/Type Character Test routine.
4. Place Sense Switch B up.
5. Prepare typewriter for operation and depress Letters Shift key on typewriter.
7. Depress Start 1 button (unless there is a malfunction, the computer halts automatically after the typewriter has typed and performed all functions from code 00 through 37^8, with 16 operations of each character in Letters Shift mode).
8. Depress Figures Shift key on typewriter.
9. Depress Start 1 button (unless there is a malfunction, the computer halts automatically after the typewriter has typed and performed all functions from code 00 through 37^8, with 16 operations of each code, in Figures Shift mode).
10. Prepare punch for operation.
11. Place Sense Switch C up.
12. Depress Start 2 button (unless there is a malfunction, the computer halts after the typewriter and punch have recorded all characters and functions from code 00 through 37^8, with 16 operations of each code).
13. Remove punched tape from punch and visually check to confirm that all codes have been punched correctly and in proper sequence.

Comments

To cause the typewriter to continue typing, place Sense Switch B down; to cause the typewriter and punch to continue typing and punching, place Sense Switch C down.

The octal character of each character is displayed on the visual readout as the character is typed and punched.
**Type Character and Punch/Type Character Test Routine**

<table>
<thead>
<tr>
<th>L00010</th>
<th>+5710440-000000</th>
<th>L10500</th>
<th>+6010070+5710560</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>+5710510-000000</td>
<td>1.</td>
<td>+6010430+6010200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.</td>
<td>+6010210+6010220</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.</td>
<td>+6010230+6010240</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.</td>
<td>+6010250+6010260</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.</td>
<td>+6010270+5710600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.</td>
<td>+0010400+4210370</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.</td>
<td>+4210371+5710000</td>
</tr>
<tr>
<td>L10000</td>
<td>+7200000+7200000</td>
<td>L10600</td>
<td>+0010400+4210410</td>
</tr>
<tr>
<td>1.</td>
<td>+7200000+7200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+7200000+7200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+7200000+7200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+7200000+7200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+7200000+7200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+7200000+7200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+7200000+7200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L10100</td>
<td>+7200100+0010370</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+6010000+6010000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+6010020+6010030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+6010040+6010050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+6010060+6010070</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+0110400+6010370</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+3510000+5410300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+5710620-000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L10200</td>
<td>+7600000+7600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+7600000+7600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+7600000+7600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+7600000+7600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+7600000+7600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+7600000+7600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+7600000+7600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+7600000+7600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L10300</td>
<td>+7200100+0010410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+6010200+6010210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+6010220+6010230</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+6010240+6010250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+6010260+6010270</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+0110400+6010410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+3610200+5710660</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+7200010+7200010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L10400</td>
<td>+0000010-0000010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>+7600010+7600010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>+7200000+7200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+7600000+7600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+0010420+6010000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+6010010+6010020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+6010030+6010040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+6010050+6010060</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"C" UP

TO HALT

L10700  +0010710+5710200

1.      +7700010-0000000
2.      +7700020-0000000
3.      +7700030-0000000
4.      +7700040-0000000
5.      +5510200+5300000
6.      +0010200+0310720
7.      +5010710+5710200

"B" UP

TO HALT

L77570  +6L10000+6677700

L77600  +5477640+5577670

L77700  +0117724+277670

1.      +3677670+5777670
2.      +400001000000000
Fill Memory With Random Numbers Test Routine Printout

L77570 +6477600+6677700
L77600 +0077710+1177700
 1. +4100010+6077710
 2. +4300000+6001010
 3. +0077620+0177670
 4. +6077620+3677620
 5. +0077620+0377730
 6. +5077740+5777600
 7. +0000000-0000010
L77700 -5252521-5252521
 1. -5260110-0664160
 2. +2525250-0000000
 3. +4300000+6077560
 4. +7777600-0000000

START: L 77570
TYPE CHARACTER, TYPE/PUNCH CHARACTER, AND PUNCH CHARACTER CONSECUTIVELY TEST ROUTINE

Purpose

To check operation of the typewriter and punch under computer control with Type Character, Type and Punch Character, and Punch Character commands.

Operating Instructions

1. Fill and verify the Type Character, Type and Punch Character, and Punch Character Consecutively Test routine.
2. Prepare typewriter for operation and depress Letters Shift key on typewriter.
3. Place Sense Switches B and C up (causes typing only).
4. Set location counter to 7757.0.
5. Depress Start button (unless there is a malfunction, the routine will continue to run until stopped manually; the routine should be allowed to run at least one minute).
6. Prepare punch for operation.
7. Set location counter to 7760.0.
8. Place Sense Switch B down (causes typing and punching).
9. Depress Start button (unless there is a malfunction, the routine will run until stopped manually; the routine should be allowed to run at least one minute).
10. Remove punched tape from punch and visually inspect to determine if punched correctly (see Comments below).
11. Place Sense Switches B up and C down (causes punching only).
12. Set location counter to 7760.0.
13. Depress Start button (unless there is a malfunction, the routine will run until stopped manually; the routine should be allowed to run at least one minute).
14. Remove punched tape from punch and visually inspect to determine if punched correctly (see Comments below).

Comments

On type only, the typewriter should record the following:

EA SIU
DRJNFCKTZLWHYPQOBG. /;EA SIU
DRJNFCKTZLWHYPQOBG. /;EA SIU
etc.
On punch and type, the same will be recorded as on type only (the binary configurations in the tape corresponding to the octal numbers 00, 01, 02, etc. through 36, except 33 and 37 which are not punched).

On punch only, the same will be recorded by the punch as on punch and type.

Type, Punch/Type, Punch Characters Consecutively
Test Routine Printout

| L 77570 | +6477600+6677700 |
| L 77600 | +5477640+5577670 |
| 1.    | +7200000+0077610 |
| 2.    | +0177720+4277610 |
| 3.    | +3677610+5777610 |
| 4.    | +7600000+0077640 |
| 5.    | +0177720+4277640 |
| 6.    | +3677640+5777640 |
| 7.    | +7400000+0077670 |
| L 77700 | +0177720+4277670 |
| 1.    | +3677670+5777670 |
| 2.    | +00000100000000 |

**TAB TEST AND TYPE WORD WITH TAB TEST ROUTINE**

**Purpose**

To check the tab operation of the typewriter under computer control with the Type Word command.

**Operating Instructions**

1. Prepare typewriter for operation and set Tab Over-ride switch on typewriter to permit tabulating.
2. Set tabs on typewriter to cause three tab operations, the first of 30 to 40 spaces from the left-hand margin and the remaining two any place before the right-hand margin and carriage return.
3. Enter and verify Tab Test and Type Word with Tab Test routine.
4. Depress Start 3 button.
5. After typewriter has completed approximately 12 operations of three tabs and a carriage return each, depress Stop button on computer.
6. Set tabs on typewriter to cause three tabulations across the page, each sufficiently long to contain one computer word in command format (16 characters).

7. Depress Start 2 button.

8. After several minutes of operations, depress Stop button.

Comments

In Step 7 typewriter records whatever is in memory, beginning with channel 00, in three columns across the page, then performs a carriage return and repeats the procedure, each time from different memory locations.

Tab Test And Type Word With Tab Test Routine Printout

LOC. 1100: START TAB TEST

LOC. 12000: START TYPE WORD WITH TAB.

L12000 +7200330+1200000
1. +7200100+1200010
2. +7200100+1200020
3. +7200370+7200100
4. +0012000+0112110
5. +4212001+0012010
6. +0112110+4212011

L11000 +7200330+7200100
1. +7200100+7200100
2. +7200370+7200100
3. +5711000-000000

L12100 +4212021+5712012
1. +0000000-000030
2. +3612020+5712000

ALPHA DUMP TEST ROUTINE

Purpose

To check operation of the tape punch by recording the contents of specified locations in memory in alphanumeric format under computer control.

Operating Instructions

1. Enter and verify Alpha Dump Test routine.
2. Set location counter to 5000.0.
3. Enter following command from control console keyboard:

+0000000+0050000 or +00XXXX0+00YYYY0
where X = beginning location from which dump is to be made
Y = location (plus 1) at which dump is to end
(X and Y locations can be 0000 to 5000 and 5042 to 7756).

5. Depress Tape Advance button on punch until at least 18 inches
   of leader have been punched.
6. Typewriter punch the following:

   L 00000 Carriage Return
   F

8. Check to make certain location counter is set at 5001.0.
9. If display is desired, place Sense Switch C up (see Comments
   below).
10. Depress Start button (routine runs until it halts automatically).
11. Depress Tape Advance button to run off at least 12 inches of
    leader and remove tape from punch.
12. Near end of punched information, cut out 15 sprocket holes with
    small knife.
13. Glue ends of tape together, with three sprocket holes over
    lapping.
14. Place tape in reader and depress Verify button on reader.
15. Set location counter to 2000.0.
16. Fill location 2000.0 with zeros from console keyboard.
17. Re-run tape in Verify mode (Verification process should stop
    and Verify Error neon on control console should light when verification
    attempt fails at location 2000.0).

Comments

Console readout display indicates location being dumped in first
address portion of readout panel.
START 1.50010  Alpha Dump Test Routine

IN 1.50000 FILL THE FOLLOWING:

+00Xxxx0+00Yyyyy0

X = START DUMP AT LOCATION

Y = END DUMP LOCATION PLUS 1

SENSE SWITCH C:

UP = DISPLAY 1.7700

DOWN = NO DISPLAY

COMMAND MEMORY DUMP (PUNCH) TEST ROUTINE

Purpose

To check operation of punch by recording contents of memory on paper tape in command format under computer control.

Operating Instructions

1. Enter Command Memory Dump (Punch) Test routine.
2. Set location counter to 7757.0.
3. Depress Start button (routine will halt computer after three channels of information have been punched).
4. Remove punched tape and verify through reader.
Purpose

To check proper entry of numbers with paper tape reader and correct type-out of the same numbers in both command and decimal formats.

Operating Instructions

1. Fill and verify Enter Numbers Through Photoreader Test routine.
2. Fill and verify Type Word (Command and Decimal) Test routine.
3. Depress Start button (typewriter should type out results as listed below).
L70000

N+12345678901.
+98765432109.
+23456789.
+144.
+275.
-52678975.
+654389.
+2.
-76555.
+1.
+2.
+12.
-123.
+123.
+123.
+123.
+0.

TYPED RESULTS OF WHICH SHOULD BE:

+0133761+4070321 05.
+1337670-0712261 2.
+0000130-5730121 0002.
+000000-0001100 00000001+566666
+2146311-3146311 4666666666.
-4155561+8554071 -86.
+0000000+1770321 000013.
+1463141+1463150 3333333333.
-6777540+2665350 -3.
+0000000-0000001 000000000000.
+0000000-0000010 000000000000.
+0000000-0000060 000000000000.
-0000000-0000751 -00000000000000.
+0000000-0000751
+0000000-0011510 000000000006.
+0000000-0110341 00000000000000.

508-T-6 325
DOUBLE-PUNCHING TEST ROUTINE

Purpose

To test operation of punch against double-punching with Punch Character and Punch and Type Character commands.

Operating Instructions

1. Enter Double-Punching Test routine from control console keyboard:

   L 40000 Enter
   C  +7400000+7400370 Enter
       +5740000+0000000 Enter

2. Set location counter to 40000.
3. Depress Start button.
4. After routine has run approximately three minutes, depress Stop button.
5. Inspect tape for double punches (see Comments below).
6. Change routine as follows:

   L 40000 Enter
   C  +7600000+7600370 Enter

7. Depress Start button.
8. After routine has run approximately three minutes, depress Stop button.
9. Inspect tape for double punches (see Comments below).

Comments

With both the punch character and punch and type character commands, the punch should record a blank code (sprocket hole only) followed by a Letters Shift code (sprocket hole and five character code holes), then continue to repeat this configuration (blank, Letters Shift, blank, Letters Shift, etc.).

INPUT-OUTPUT AND MEMORY TEST ROUTINE

Purpose

To (1) check operation of paper tape reader, tape punch, and typewriter, and (2) simultaneously check the memory for incorrect recording and reading and memory changing information.

5-18
Operating Instructions

1. Enter and verify Input-Output and Memory Test routine.
2. Punch leader sufficiently long to loop back to and insert in photoreader (leave tape slack).
3. Turn Compute-Manual switch on punch to Compute position.
4. Set location counter to 2516.0.
5. Depress Start button (routine runs until malfunction occurs or it has operated through entire memory).

Comments

Routine causes punching leader of first 17 words of routine; after these have been punched and typed, routine causes reading and verification of one word at a time, each reading and verification being followed by typing and punching of another word.

During the typing and punching of the first 17 words, the visual readout displays the location being recorded on the right side of the readout; when reading and verification commences, the visual readout displays the location just verified in the first half command portion of the readout. If the routine stops after reading a word and the Compute neon is extinguished, either (1) the reader has failed in verifying the information punched, or (2) the reader has malfunctioned by permitting too many locations to pass through the reader, or (3) the reader did not read the proper character. If the routine halts and the Compute neon remains lighted, the typewriter or tape punch malfunctioned.

Input-Output and Memory Test Routine Printout

L000010 +5765160-0000000
L65000 +7200330+7200100
1. +7400150+1600000
2. +7400050+0065040
3. +7400000+0165030
4. +5365030+0065010
5. +0165140+4265011
6. +3665010+3365150
7. +5065001+7365100

L65200 +0000000-0000000
L65100 +0300000+5065111
1. +7765111+0065100
2. +0165140+4265100
3. +3665100+5765001
4. +0000010-0000010
5. +0000000-0077600
6. +0065200+4265011
7. +4265100+5765000
<table>
<thead>
<tr>
<th>Type Word: Command and Decimal Routine Printout</th>
</tr>
</thead>
<tbody>
<tr>
<td>L77570  +6477600+6677700</td>
</tr>
<tr>
<td>L77600  +7200330+1260000</td>
</tr>
<tr>
<td>1.     +7200100+1260001</td>
</tr>
<tr>
<td>2.     +7200370+7200100</td>
</tr>
<tr>
<td>3.     +0077600+0177700</td>
</tr>
<tr>
<td>4.     +4277601+0077610</td>
</tr>
<tr>
<td>5.     +0177700+4277611</td>
</tr>
<tr>
<td>6.     +0377710+5077720</td>
</tr>
<tr>
<td>7.     +5777600-0000000</td>
</tr>
<tr>
<td>L77700  +0000000-0000010</td>
</tr>
<tr>
<td>1.     +7200100+1260201</td>
</tr>
<tr>
<td>2.     +7777600-0000000</td>
</tr>
</tbody>
</table>
MEMORY TEST ROUTINE

Purpose

To check accuracy of writing and reading processes by writing various patterns in each sector and verifying the readout from each.

Operating Instructions

1. Enter and verify Memory Test routine.
2. Place Sense Switch B up (causes typewriter to print Memory OK. if routine runs successfully).
3. Depress Start 1 button (unless there is a malfunction, routine will halt automatically).

Comments

Malfunction causes computer to stop after displaying number of faulty location on console visual readout.

Memory Test Routine Printout

<table>
<thead>
<tr>
<th>Memory Location</th>
<th>Data Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO0000</td>
<td>-5263431+0370170</td>
</tr>
<tr>
<td>0+3600620+7700101</td>
<td>-6254341+0370170</td>
</tr>
<tr>
<td>+0000600+6000620</td>
<td>+6400600+6600500</td>
</tr>
<tr>
<td>+0000620+4200030</td>
<td>+3677650+7777620</td>
</tr>
<tr>
<td>+6000170+40100510</td>
<td>+3077590+3501000</td>
</tr>
<tr>
<td>+6000620+3000630</td>
<td>+0001000+0301010</td>
</tr>
<tr>
<td>+5100020+0000620</td>
<td>+5077550+7777550</td>
</tr>
<tr>
<td>+0300610+4200070</td>
<td>+5277650+7777560</td>
</tr>
<tr>
<td>+0300170+5000101</td>
<td>+0077620+0177740</td>
</tr>
<tr>
<td>-0000000+0000620</td>
<td>+1277621+0077630</td>
</tr>
<tr>
<td>+0300610+6000620</td>
<td>+0177460+0077630</td>
</tr>
<tr>
<td>+3600070+5200061</td>
<td>+0377170+377630</td>
</tr>
<tr>
<td>+4100500+6000620</td>
<td>+5177650+3400591</td>
</tr>
<tr>
<td>+6400020+6600030</td>
<td>+7200370+5700560</td>
</tr>
<tr>
<td>+5777500+0000000</td>
<td>+0000020+0000020</td>
</tr>
<tr>
<td>+2525250+0000000</td>
<td>+2514340+7407601</td>
</tr>
<tr>
<td>+0000170+0000000</td>
<td>+2514340+7407601</td>
</tr>
<tr>
<td>+6701000+0077600</td>
<td>+0077560+0377570</td>
</tr>
<tr>
<td>+6701000+0077600</td>
<td>+0001170+0000000</td>
</tr>
<tr>
<td>+6701000+0077600</td>
<td>+0001000+0000000</td>
</tr>
<tr>
<td>+0000620+0100610</td>
<td>+0060000+0000000</td>
</tr>
<tr>
<td>+0000620+3000640</td>
<td>+0100170+0000000</td>
</tr>
<tr>
<td>+5000000+5700140</td>
<td>+0006000+0000000</td>
</tr>
<tr>
<td>+7777500+0777500</td>
<td>+3677630+7700130</td>
</tr>
<tr>
<td>+0001000+0000000</td>
<td>+7200340+7200010</td>
</tr>
<tr>
<td>-5263431+0370170</td>
<td>+7200340+7200030</td>
</tr>
<tr>
<td>-5263431+0370170</td>
<td>+7200120+7200050</td>
</tr>
<tr>
<td>-5263431+0370170</td>
<td>+7200040+7200030</td>
</tr>
<tr>
<td>-5263431+0370170</td>
<td>+7200170+7200530</td>
</tr>
<tr>
<td>-5263431+0370170</td>
<td>+7200340+7200100</td>
</tr>
<tr>
<td>-5263431+0370170</td>
<td>+5700130+0000000</td>
</tr>
</tbody>
</table>
NIXIE TUBE TEST ROUTINE

Purpose

To check accuracy of readout of the Nixie tubes under computer control with the Display command.

Operating Instructions

1. Set location counter to 7757.0.
2. Enter and verify Nixie Tube Test routine.
3. Place Transfer Stop switch up.
4. Set location counter to 7757.0.
5. Depress Start button once (all Nixie tubes should display numeral 0).
6. Depress Start button twice (all Nixie tubes should display numeral 1).
7. Repeat Step 6 to display, in order, each numeral through 9.
8. Depress Start button six times (all decimal point indicators should light).
9. Place Transfer Stop switch down.
10. Depress Stop button once after routine has run approximately one minute.

Comments

Permitting routine to run continuously checks whether continuous operation will cause an indicator to malfunction after warm-up.
Nixie Tube Test Routine Printout

L 7757.0 +6477600+0000000
7760.0 +4100500+6077650
7761.0 +6077660+3677651
7762.0 +0077650+0177640
7763.0 +5377600+5777601
7764.0 +0421040+0421040

READ TAPE TEST ROUTINE

Purpose

To determine correct reading of tape under computer control with the RDZ command.

Operating Instructions

1. Prepare a tape as follows:
   a. C+7654321+7654321 Carriage Return
      -6543211+7351240 Carriage Return
   b. 12 sprocket holes (for sufficient memory access time)
   c. N +8192.
   d. 12 sprocket holes (for sufficient memory access time)
   e. L 00011 Carriage Return
   f. S
   g. 12 sprocket holes (for sufficient tape leader to allow proper starting and stopping of reader)
   h. FABCEFEGH Carriage Return
   i. IJKLMNOPQ Carriage Return
   j. 12 sprocket holes (for sufficient memory access time)
   k. L 00111 Carriage Return
   l. S

2. Enter the program given in the routine printout via the tape reader, typewriter, or console keyboard.

3. Splice tape prepared in (1) above to form a continuous loop sufficiently long to easily fit around reader head; place loop to read start of tape first.

4. Place Operation switch in Continuous position.

5. Depress Start 1 button (first length of tape should feed, then stop and immediately resume).

6. If tape runs through successfully several times, depress Halt button.
Comments

Computer should not halt nor display during test; if either occurs, the information read improperly can be determined by comparing the contents of memory locations 0050, 0051, 0052, 0060, and 0061 with the information on the tape.

Read Tape Test Routine Printout

L00010  +7300500+0000500
  2.  +0300170+5000040
  3.  +3600500+7700040
  4.  +0000510+0300200
  5.  +5000061+3600510
  6.  +7700061+0000520
  7.  +0300210+5000110
L00100  +3600520+7700110
  1.  +7300600+0000600
  2.  +0300220+5000140
  3.  +3600600+7700140
  4.  +0000610+0300230
  5.  +5000161+3600610
  6.  +7700010+5700010
  7.  +7654321+7654321
L00200  -6543211+7351240
  1.  +0000000-0100000
  2.  -1713441-0556520
  3.  -3133710+6146130

COMPUTER-CONTROLLED TYPEWRITER INPUT TEST ROUTINE

Purpose

To check proper operation of typewriter on computer-controlled input with the RDY command.

Operating Instructions

1. Enter the program given in the routine printout via the tape reader or control console.
2. Place Operation switch in Continuous position.
3. Depress Start 1 button (compute indicator should flash on and off).
4. Depress the following keys on the typewriter:
   a. C+7654321+7654321 Carriage Return
   b. -6543211+7351240 Carriage Return
   c. N+8192.
   d. L00011 Carriage Return
   e. S (Compute indicator should flash on and off; if visual readout displays anything, typewriter has entered information improperly and should be checked.)

5. After Compute indicator extinguished and if no display was obtained, depress the following keys on the typewriter:
   a. FABCDEFGH Carriage Return
   b. IJKLMNOPQ Carriage Return
   c. L00111 Carriage Return
   d. S (Compute indicator should flash on and off; if visual readout displays anything, typewriter has entered improperly and should be checked.)

Computer-Controlled Typewriter Input Test Routine Printout

`COMPUTER-CONTROLLED TYPEWRITER ALPHANUMERIC INPUT TEST ROUTINE`

Purpose

To check accuracy of input of alphanumeric information from typewriter under computer control with the RDY command.

Operating Instructions

1. Enter the routine given in the routine printout with the tape reader or from the control console.
2. Depress Start 1 button (routine should halt immediately).
3. Depress following typewriter keys (DO NOT DEPRESS TYPEWRITER FILL SWITCH):

Carriage Return Carriage Return Carriage
Return Letter Shift THIS (pause)

Space IS space RECO (pause: Place Sense
Switch B up)

MP Period Carriage Return Blank Blank Blank
Blank (After depressing the last Blank the
computer should immediately begin typing the
message entered: THIS IS RECOMP. and
then Carriage Return).

4. Depress Halt button on control console.

Computer-Controlled Typewriter Alphanumeric Input Test Routine

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L03060</td>
<td>+403100+4100500</td>
<td></td>
<td>L50200</td>
<td>+7200000+4277710</td>
</tr>
<tr>
<td>7.</td>
<td>+777600-000000</td>
<td></td>
<td>1.</td>
<td>+7200000+0775000</td>
</tr>
<tr>
<td>L03100</td>
<td>+717600+6020000</td>
<td></td>
<td>2.</td>
<td>+7200000+0150300</td>
</tr>
<tr>
<td>1.</td>
<td>+007600+0177630</td>
<td></td>
<td>3.</td>
<td>+7200000+7200000</td>
</tr>
<tr>
<td>2.</td>
<td>+677600+5777600</td>
<td></td>
<td>4.</td>
<td>+4277600+7200000</td>
</tr>
<tr>
<td>3.</td>
<td>+0000000-000010</td>
<td></td>
<td>5.</td>
<td>+4100000+7200000</td>
</tr>
<tr>
<td>L50000</td>
<td>+0020000-0000000</td>
<td></td>
<td>6.</td>
<td>+3577600+7200000</td>
</tr>
<tr>
<td>1.</td>
<td>+0050000+4250100</td>
<td></td>
<td>7.</td>
<td>+5477600+5750330</td>
</tr>
<tr>
<td>2.</td>
<td>+6450100+6650200</td>
<td></td>
<td>L50300</td>
<td>+0000010-0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+777600-0000000</td>
<td></td>
<td>1.</td>
<td>+0000200-0000000</td>
</tr>
<tr>
<td>4.</td>
<td>+777600-0000000</td>
<td></td>
<td>2.</td>
<td>-1737940+7400000</td>
</tr>
<tr>
<td>5.</td>
<td>+777600-0000000</td>
<td></td>
<td>3.</td>
<td>+0077600+4250340</td>
</tr>
<tr>
<td>6.</td>
<td>+777600-0000000</td>
<td></td>
<td>4.</td>
<td>+0020150+4505060</td>
</tr>
<tr>
<td>7.</td>
<td>+777600-0000000</td>
<td></td>
<td>5.</td>
<td>+777600-0000000</td>
</tr>
<tr>
<td>L50100</td>
<td>+0020000+6050320</td>
<td></td>
<td>6.</td>
<td>+7750010-0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+4000160+5177621</td>
<td></td>
<td>7.</td>
<td>+4000050+4277741</td>
</tr>
<tr>
<td>2.</td>
<td>+0150310+4277700</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>+4277730+050320</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>+4100010+4277730</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>+4277761+4000530</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>+4277720+4277751</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>+4000050+4277741</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PUNCH WORD TEST ROUTINE

PURPOSE

To test operation of punch under computer control with PNW command in command and binary-command and binary-coded-decimal formats.

OPERATING INSTRUCTIONS

1. Enter the following routine with tape reader, typewriter, or console keyboard:

<table>
<thead>
<tr>
<th>Location</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.0</td>
<td>+7200330+1400030</td>
</tr>
<tr>
<td>0002.0</td>
<td>+1400041+5700010</td>
</tr>
<tr>
<td>0003.0</td>
<td>+7654321+7654321</td>
</tr>
<tr>
<td>0004.0</td>
<td>-0022150-2547000</td>
</tr>
<tr>
<td>0005.0</td>
<td>+4234671-7400000</td>
</tr>
</tbody>
</table>

2. Prepare punch for operation.
3. Place Operation switch in continuous position.
4. Depress Start 1 button (see Comments).
5. After routine has run desired number of times, depress stop button.
Comments

The following tape should be punched each time the routine runs:

Start of Tape

PUNCH AND TYPE WORD TEST ROUTINE

Purpose

To test operation of typewriter and punch under computer control with PTW command in command and binary-coded-decimal formats.

OPERATING INSTRUCTIONS

1. Enter the following routine with the tape reader, typewriter, or console keyboard:

<table>
<thead>
<tr>
<th>Location</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.0</td>
<td>+7200370+7200100</td>
</tr>
<tr>
<td>0002.0</td>
<td>+7200330+1600040</td>
</tr>
<tr>
<td>0003.0</td>
<td>+1600051+5700010</td>
</tr>
<tr>
<td>0004.0</td>
<td>+7654321+7654321</td>
</tr>
<tr>
<td>0005.0</td>
<td>-0022150-2547000</td>
</tr>
<tr>
<td>0006.0</td>
<td>+4234021-4760000</td>
</tr>
</tbody>
</table>

2. Prepare typewriter and punch for operation.
3. Place Operation switch in Continuous position.
4. Depress Start 1 button (see Comments):
5. After routine has run desired number of times depress Stop button.

Comments

The following information should be typed each time the routine runs:

+7654321+7654321-0123456789.0
The following tape should be punched each time the routine runs:

Start of Tape

TRAPPING MODE TEST ROUTINE

Purpose

To test correct operation of the Trapping Mode from various memory locations.

Operating Instructions

1. Enter the following routine with the tape reader, typewriter, or console keyboard:

<table>
<thead>
<tr>
<th>Location</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000.0</td>
<td>+7700000-000000</td>
</tr>
<tr>
<td></td>
<td>The above command in location 0000.0 must always accompany either of the commands below entered from any other location</td>
</tr>
<tr>
<td>0001.0</td>
<td>+0000000-000000</td>
</tr>
<tr>
<td>0002.0</td>
<td>-0000000+000000</td>
</tr>
<tr>
<td>Any other</td>
<td>The commands at 0001.0 or 0002.0</td>
</tr>
</tbody>
</table>

2. Place Operation switch in Continuous position.
3. Depress Start 1 button (computer should halt and location counter should read 0000.0).
4. Depress Start 2 button (computer should halt and location counter should read 0000.0).
5. If any other location is selected, set location counter to location in which it is desired to place the pair of commands at 0001.0 or 0002.0 and enter the pair chosen.
6. Set location counter to location chosen in Step 5.
7. Depress Start button to start routine at location chosen in Step 5. (computer should halt and location counter should read 0000.0).
LOOP LOAD ROUTINE

Purpose

To fill L and V loops and A, B, C, and R registers with pattern in memory locations 0003.0 through 0013.0 of this routine for checking loops and registers for amplitude, crosstalk and noise, and pulse width.

Operating Instructions

1. Enter and verify Loop Load routine with tape reader, typewriter, or console keyboard.
2. Depress Start 1 button (computer halts automatically).

Loop Load Routine Printout

<table>
<thead>
<tr>
<th>Location</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L00010</td>
<td>+64000030+66000030</td>
</tr>
<tr>
<td>2</td>
<td>+30000030+57000030</td>
</tr>
<tr>
<td>3</td>
<td>+7725250-0005250</td>
</tr>
<tr>
<td>4</td>
<td>+7725250-0005250</td>
</tr>
<tr>
<td>5</td>
<td>+7725250-0005250</td>
</tr>
<tr>
<td>6</td>
<td>+7725250-0005250</td>
</tr>
<tr>
<td>7</td>
<td>+7725250-0005250</td>
</tr>
<tr>
<td>L00100</td>
<td>+7725250-0005250</td>
</tr>
<tr>
<td>1</td>
<td>+7725250-0005250</td>
</tr>
<tr>
<td>2</td>
<td>+7725250-0005250</td>
</tr>
<tr>
<td>3</td>
<td>+7725250-0005250</td>
</tr>
</tbody>
</table>

FAST MEMORY FILL AND ONE CHANNEL ZERO ROUTINE

Purpose

To fill main memory with pattern in memory locations 0040.0 through 0047.0 of this routine and load any channel selected (except 00) with all zeros for checking memory for amplitude, crosstalk and noise, and pulse width.

Operating Instructions

See instructions for checking main memory in Section IV.
### Fast Memory Fill and One-Channel Zero Routine Printout

<table>
<thead>
<tr>
<th>Location</th>
<th>Instruction</th>
<th>Location</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO0000</td>
<td>+5600020+7700010</td>
<td>LO0600</td>
<td>+5241571+0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+5700260+000000</td>
<td>1.</td>
<td>+5242571+0000000</td>
</tr>
<tr>
<td>2.</td>
<td>+7177610+4100120</td>
<td>2.</td>
<td>+5243571+0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+4200101+4100500</td>
<td>3.</td>
<td>+5244571+0000000</td>
</tr>
<tr>
<td>4.</td>
<td>+7177610+3300220</td>
<td>4.</td>
<td>+5245571+0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+4100700+300100</td>
<td>5.</td>
<td>+5246571+0000000</td>
</tr>
<tr>
<td>6.</td>
<td>+4200101+300210</td>
<td>6.</td>
<td>+5247571+0000000</td>
</tr>
<tr>
<td>7.</td>
<td>+4200161+3600200</td>
<td>7.</td>
<td>+5250571+0000000</td>
</tr>
<tr>
<td>LO0100</td>
<td>+0000140+6000000</td>
<td>LO0700</td>
<td>+5251571+0000000</td>
</tr>
<tr>
<td>1.</td>
<td>+0000100+0100150</td>
<td>1.</td>
<td>+5020571+0000000</td>
</tr>
<tr>
<td>2.</td>
<td>+4200101+000000</td>
<td>2.</td>
<td>+5021571+0000000</td>
</tr>
<tr>
<td>3.</td>
<td>+5700170+000000</td>
<td>3.</td>
<td>+5022571+0000000</td>
</tr>
<tr>
<td>4.</td>
<td>-3000000-000010</td>
<td>4.</td>
<td>-3001710+1276550</td>
</tr>
<tr>
<td>5.</td>
<td>+0000000-000010</td>
<td>5.</td>
<td>-3001730+1276560</td>
</tr>
<tr>
<td>6.</td>
<td>+0000140+6000000</td>
<td>6.</td>
<td>+7200370+5700330</td>
</tr>
<tr>
<td>7.</td>
<td>+0300160+5002001</td>
<td>7.</td>
<td>+0000000-000001</td>
</tr>
<tr>
<td>LO0200</td>
<td>+5700100+7700010</td>
<td>LO1000</td>
<td>+6157741+7556460</td>
</tr>
<tr>
<td>1.</td>
<td>+0000000+0000100</td>
<td>1.</td>
<td>+0000000+0000100</td>
</tr>
<tr>
<td>2.</td>
<td>+0000000-000031</td>
<td>2.</td>
<td>+0000000-000031</td>
</tr>
<tr>
<td>3.</td>
<td>+5777030+0077700</td>
<td>3.</td>
<td>+5777030+0077700</td>
</tr>
<tr>
<td>4.</td>
<td>+1777720+377720</td>
<td>4.</td>
<td>+1777720+377720</td>
</tr>
<tr>
<td>5.</td>
<td>+0377700+577700</td>
<td>5.</td>
<td>+0377700+577700</td>
</tr>
<tr>
<td>6.</td>
<td>+6400000+0077700</td>
<td>6.</td>
<td>+6400000+0077700</td>
</tr>
<tr>
<td>7.</td>
<td>+5777600-0000000</td>
<td>7.</td>
<td>+5777600-0000000</td>
</tr>
<tr>
<td>LO0300</td>
<td>+0077670+0777610</td>
<td>LO1030</td>
<td>+0077670+0777610</td>
</tr>
<tr>
<td>1.</td>
<td>+6710000+0177660</td>
<td>1.</td>
<td>+6710000+0177660</td>
</tr>
<tr>
<td>2.</td>
<td>+0677610+0377650</td>
<td>2.</td>
<td>+0677610+0377650</td>
</tr>
<tr>
<td>3.</td>
<td>+5000000+0077610</td>
<td>3.</td>
<td>+5000000+0077610</td>
</tr>
<tr>
<td>4.</td>
<td>+5777610-0000000</td>
<td>4.</td>
<td>+5777610-0000000</td>
</tr>
<tr>
<td>5.</td>
<td>+5776000+0177660</td>
<td>5.</td>
<td>+5776000+0177660</td>
</tr>
<tr>
<td>6.</td>
<td>+0000100-000000</td>
<td>6.</td>
<td>+0000100-000000</td>
</tr>
<tr>
<td>7.</td>
<td>+6710000+0177660</td>
<td>7.</td>
<td>+6710000+0177660</td>
</tr>
<tr>
<td>LO0400</td>
<td>+2514340+7407601</td>
<td>LO0500</td>
<td>+1277710-3001970</td>
</tr>
<tr>
<td>1.</td>
<td>+2514340+7407601</td>
<td>1.</td>
<td>+3001710+1277720</td>
</tr>
<tr>
<td>2.</td>
<td>+2514340+7407601</td>
<td>2.</td>
<td>+3001730+1277730</td>
</tr>
<tr>
<td>3.</td>
<td>+2514340+7407601</td>
<td>3.</td>
<td>+3001750+1277750</td>
</tr>
<tr>
<td>4.</td>
<td>+2514340+7407601</td>
<td>4.</td>
<td>+1277740-3001730</td>
</tr>
<tr>
<td>5.</td>
<td>+2514340+7407601</td>
<td>5.</td>
<td>+1277750-3001750</td>
</tr>
<tr>
<td>6.</td>
<td>+2514340+7407601</td>
<td>6.</td>
<td>+3001790+1276540</td>
</tr>
<tr>
<td>7.</td>
<td>+2514340+7407601</td>
<td>7.</td>
<td>+3001670+5700740</td>
</tr>
</tbody>
</table>

**To zero a channel:**

Depress start 2. Strike two keyboard characters of the channel desired to be zeroed. (from channel 9100 and up.)

**To load memory with pattern:**

Depress start 1 and memory will load with pattern found in locations 004000 thru 004700.
## APPENDIX I. DEFINITION OF LOGIC TERMS

### FUNCTIONAL COMPONENT LISTING

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>E 0</td>
<td>Add-Subtract Indicator Flip-Flop</td>
</tr>
<tr>
<td>E 1-3</td>
<td>Input Sequence Counter</td>
</tr>
<tr>
<td>E 20</td>
<td>&quot;Number Fill&quot; State Flip-Flop</td>
</tr>
<tr>
<td>E 30</td>
<td>&quot;Number Fill&quot; State Flip-Flop</td>
</tr>
<tr>
<td>E 40</td>
<td>&quot;Number Fill&quot; State Flip-Flop</td>
</tr>
<tr>
<td>E p1-p5 &amp; E'p1-p5</td>
<td>&quot;Punch&quot; Echo Signals</td>
</tr>
<tr>
<td>E t1-t5 &amp; E't1-t5</td>
<td>&quot;Type&quot; Echo Signals</td>
</tr>
<tr>
<td>F 0</td>
<td>Fill Order Counter Flip-Flop</td>
</tr>
<tr>
<td>F 1</td>
<td>Input-Output Register</td>
</tr>
<tr>
<td>F 2</td>
<td>Input-Output Register</td>
</tr>
<tr>
<td>F 3</td>
<td>Input-Output Register</td>
</tr>
<tr>
<td>F 4</td>
<td>Input-Output Register</td>
</tr>
<tr>
<td>F 5</td>
<td>Input-Output Register</td>
</tr>
<tr>
<td>F c1, c2</td>
<td>Format Control Flip-Flop</td>
</tr>
<tr>
<td>G 0</td>
<td>Half-Word Indication Flip-Flop</td>
</tr>
<tr>
<td>G 1, 2</td>
<td>Location Counter Logic</td>
</tr>
<tr>
<td>G 3-10</td>
<td>Location Counter Logic</td>
</tr>
<tr>
<td>G 11, 12</td>
<td>Location Counter Logic</td>
</tr>
<tr>
<td>H 0</td>
<td>Transfer of Control-Verify Input Flip-Flop</td>
</tr>
<tr>
<td>H 1-4</td>
<td>Readout Control</td>
</tr>
<tr>
<td>H c</td>
<td>&quot;Stop Compute&quot; Button</td>
</tr>
<tr>
<td>I 1</td>
<td>State Flip-Flop</td>
</tr>
<tr>
<td>I 2, 3</td>
<td>State Flip-Flop</td>
</tr>
<tr>
<td>I 4</td>
<td>State Flip-Flop</td>
</tr>
<tr>
<td>I 5</td>
<td>Input Buffer Fill Flip-Flop</td>
</tr>
<tr>
<td>J 0</td>
<td>Overflow and Error Flip-Flop</td>
</tr>
<tr>
<td>J 1-6</td>
<td>Sector Agreement Signals for Readout</td>
</tr>
<tr>
<td>J 7-12</td>
<td>Channel Set Signals for Readout</td>
</tr>
<tr>
<td>J 13</td>
<td>Output Error Flip-Flop</td>
</tr>
<tr>
<td>K 0</td>
<td>Loop Read Flip-Flop</td>
</tr>
<tr>
<td>K 1-5</td>
<td>Input Channel Switches 1 through 5</td>
</tr>
<tr>
<td>K a</td>
<td>Computation Carry Flip-Flop</td>
</tr>
<tr>
<td>K f</td>
<td>Keyboard Fill Button</td>
</tr>
<tr>
<td>K g</td>
<td>Order Counter Carry Flip-Flop</td>
</tr>
<tr>
<td>K p5</td>
<td>Photoreader Input Channels 1 thru 5</td>
</tr>
<tr>
<td>K q</td>
<td>Computation Cycle Counter Carry Flip-Flop</td>
</tr>
<tr>
<td>K s</td>
<td>Sector Counter Carry Flip-Flop</td>
</tr>
<tr>
<td>Code</td>
<td>Name</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>L 1-2</td>
<td>L-Memory Loop</td>
</tr>
<tr>
<td>L 2a</td>
<td>L-Loop 4-Word Read Flip-Flop</td>
</tr>
<tr>
<td>L 41</td>
<td>L-Memory Loop</td>
</tr>
<tr>
<td>L b</td>
<td>Fill &quot;Location&quot; Button</td>
</tr>
<tr>
<td>L p1</td>
<td>Loop-Memory Selection Flip-Flop</td>
</tr>
<tr>
<td>L p2</td>
<td>L-V Loop Selection Flip-Flop</td>
</tr>
<tr>
<td>L r4</td>
<td>L-Loop Read Amplifier Output from 4-word Read Head</td>
</tr>
<tr>
<td>L r8</td>
<td>L-Loop Read Amplifier Output from 8-word Read Head</td>
</tr>
<tr>
<td>L w</td>
<td>L-Loop Write Amplifier Input</td>
</tr>
<tr>
<td>M 0-2</td>
<td>Operation-Type Flip-Flops</td>
</tr>
<tr>
<td>M 3-5</td>
<td>Operation-Type Flip-Flops</td>
</tr>
<tr>
<td>M 6-10</td>
<td>Operation-Type Flip-Flops</td>
</tr>
<tr>
<td>M d</td>
<td>Display Selection Switch Setting for Memory</td>
</tr>
<tr>
<td>M r</td>
<td>Memory Read Flip-Flop</td>
</tr>
<tr>
<td>M ra</td>
<td>Memory Read Power Amplifier</td>
</tr>
<tr>
<td>M rb</td>
<td>Memory Read Power Amplifier</td>
</tr>
<tr>
<td>M rc</td>
<td>Memory Read Power Amplifier</td>
</tr>
<tr>
<td>M rd</td>
<td>Memory Read Power Amplifier</td>
</tr>
<tr>
<td>M re</td>
<td>Memory Read Power Amplifier</td>
</tr>
<tr>
<td>M rf</td>
<td>Memory Read Power Amplifier</td>
</tr>
<tr>
<td>M rg</td>
<td>Memory Read Power Amplifier</td>
</tr>
<tr>
<td>M rh</td>
<td>Memory Read Power Amplifier</td>
</tr>
<tr>
<td>M r0-r63</td>
<td>Memory Read Amplifier Outputs from Channels 0 to 63 respectively</td>
</tr>
<tr>
<td>M s</td>
<td>Memory Sync Flip-Flop</td>
</tr>
<tr>
<td>M w1</td>
<td>Main Memory Write Flip-Flop</td>
</tr>
<tr>
<td>M w2</td>
<td>Main Memory Write Flip-Flop</td>
</tr>
<tr>
<td>N 0</td>
<td>Fill Decimal Number Flip-Flop</td>
</tr>
<tr>
<td>N 1</td>
<td>Sector Counter Timing Flip-Flop</td>
</tr>
<tr>
<td>N 2</td>
<td>Operation Code Timing Flip-Flop</td>
</tr>
<tr>
<td>N 4</td>
<td>Location Counter Timing Flip-Flop</td>
</tr>
<tr>
<td>N 5</td>
<td>Channel Register Timing Flip-Flop</td>
</tr>
<tr>
<td>N 6</td>
<td>Output Timing Flip-Flop</td>
</tr>
<tr>
<td>N 7</td>
<td>PS Order Flip-Flop</td>
</tr>
<tr>
<td>N 8</td>
<td>Readout Control</td>
</tr>
<tr>
<td>N 8a</td>
<td>Readout Timing Flip-Flop</td>
</tr>
<tr>
<td>N 9</td>
<td>Sector Counter Timing Flip-Flop</td>
</tr>
<tr>
<td>N 11</td>
<td>State Change Flip-Flop</td>
</tr>
<tr>
<td>N 12</td>
<td>State Change Flip-Flop</td>
</tr>
<tr>
<td>N b</td>
<td>Fill &quot;Numbers&quot; Button</td>
</tr>
<tr>
<td>Code</td>
<td>Name</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>O 0</td>
<td>Fill Orders Flip-Flop</td>
</tr>
<tr>
<td>O 1</td>
<td>Fill Orders Flip-Flop</td>
</tr>
<tr>
<td>O f</td>
<td>Octal Setting of Format Switch</td>
</tr>
<tr>
<td>P 0</td>
<td>Legitimate Code Flip-Flop</td>
</tr>
<tr>
<td>P 1-6</td>
<td>Digit Counter</td>
</tr>
<tr>
<td>P d3</td>
<td>Delay Flip-Flop</td>
</tr>
<tr>
<td>P m</td>
<td>Delay Flip-Flop</td>
</tr>
<tr>
<td>P ma</td>
<td>Delay Flip-Flop</td>
</tr>
<tr>
<td>Q 0</td>
<td>Computation Cycle Counter</td>
</tr>
<tr>
<td>Q 1-6</td>
<td>Computation Cycle Counter</td>
</tr>
<tr>
<td>R 1-2</td>
<td>R-Register</td>
</tr>
<tr>
<td>R 40</td>
<td>R-Register</td>
</tr>
<tr>
<td>R 41</td>
<td>R-Register</td>
</tr>
<tr>
<td>R 42</td>
<td>R-Register</td>
</tr>
<tr>
<td>R 43</td>
<td>R-Register</td>
</tr>
<tr>
<td>R b</td>
<td>Readout Activation Button</td>
</tr>
<tr>
<td>R c1-c4</td>
<td>Readout Timing Flip-Flops</td>
</tr>
<tr>
<td>R d</td>
<td>Display Selection Switch Setting for R-Register</td>
</tr>
<tr>
<td>R gm</td>
<td>Mechanical Tape Reader On Button</td>
</tr>
<tr>
<td>R gp</td>
<td>Photoelectric Tape Reader On Button</td>
</tr>
<tr>
<td>R sm</td>
<td>Mechanical Tape Reader Off Button</td>
</tr>
<tr>
<td>R sp</td>
<td>Photoelectric Reader Off Button</td>
</tr>
<tr>
<td>R r</td>
<td>R-Register Read Amplifier Output</td>
</tr>
<tr>
<td>R w</td>
<td>R-Register Write Amplifier Inputs</td>
</tr>
<tr>
<td>S</td>
<td>Sum Flip-Flop</td>
</tr>
<tr>
<td>S 0</td>
<td>Memory Read Sector Selection Flip-Flop</td>
</tr>
<tr>
<td>S 1-6</td>
<td>Sector Counter</td>
</tr>
<tr>
<td>S 7</td>
<td>Loop Read Sector Selection Flip-Flop</td>
</tr>
<tr>
<td>S 8</td>
<td>Loop Write Sector Selection Flip-Flop</td>
</tr>
<tr>
<td>S 11</td>
<td>Memory Write Sector Selection Flip-Flop</td>
</tr>
<tr>
<td>S c</td>
<td>&quot;Start Compute&quot; Button</td>
</tr>
<tr>
<td>S p</td>
<td>Preset Stop Switch On</td>
</tr>
<tr>
<td>S sB</td>
<td>Sense Switch for TSB Command</td>
</tr>
<tr>
<td>S sC</td>
<td>Sense Switch for TSC Command</td>
</tr>
<tr>
<td>S sD</td>
<td>Sense Switch for TSD Command</td>
</tr>
<tr>
<td>S x</td>
<td>Readout Timing Flip-Flop</td>
</tr>
<tr>
<td>S u1-3</td>
<td>Set-up Buttons 1-3</td>
</tr>
<tr>
<td>S0</td>
<td>&quot;Single Order&quot; Positions of Operation Switch</td>
</tr>
<tr>
<td>S0'</td>
<td>&quot;Continuous&quot; Position of Operation Switch</td>
</tr>
<tr>
<td>Code</td>
<td>Name</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>T 0</td>
<td>Tape Reader On Flip-Flop</td>
</tr>
<tr>
<td>T 1</td>
<td>Sync Bit Time Flip-Flop</td>
</tr>
<tr>
<td>T 2</td>
<td>Least significant bit time flip-flop</td>
</tr>
<tr>
<td>T 40</td>
<td>Most significant bit time flip-flop</td>
</tr>
<tr>
<td>T 41, 41a</td>
<td>Sign Bit Time Flip-Flops</td>
</tr>
<tr>
<td>T d</td>
<td>Initial Time Delay and Error Reset Button</td>
</tr>
<tr>
<td>T g</td>
<td>Typewriter On Button</td>
</tr>
<tr>
<td>T s</td>
<td>Typewriter Off Button</td>
</tr>
<tr>
<td>T p</td>
<td>&quot;Punch&quot; Echo Timing Pulse</td>
</tr>
<tr>
<td>T t</td>
<td>&quot;Type&quot; Echo Timing Pulse</td>
</tr>
<tr>
<td>T x</td>
<td>Readout Timing Flip-Flop</td>
</tr>
<tr>
<td>TPP</td>
<td>Tape Punch Pulse Flip-Flop</td>
</tr>
<tr>
<td>TPP k</td>
<td>Tape Punch Pulse for Keyboard</td>
</tr>
<tr>
<td>TPP p</td>
<td>Tape Punch Pulse for Punch</td>
</tr>
<tr>
<td>TPP rm</td>
<td>Tape Punch Pulse for Mechanical Reader</td>
</tr>
<tr>
<td>TPP rp</td>
<td>Tape Punch Pulse for Photoelectric Reader</td>
</tr>
<tr>
<td>TPP t</td>
<td>Tape Punch Pulse for Typewriter</td>
</tr>
<tr>
<td>TPP 5</td>
<td>Input Sampling Signal = (TPP) P5</td>
</tr>
<tr>
<td>U 1-2</td>
<td>Computation Word Time Flip-Flops</td>
</tr>
<tr>
<td>U 41</td>
<td>Last Divide Word Time Flip-Flop</td>
</tr>
<tr>
<td>V 1-2</td>
<td>V-Memory Loop</td>
</tr>
<tr>
<td>V 2a</td>
<td>V-loop 4-word Read Flip-Flop</td>
</tr>
<tr>
<td>V 41</td>
<td>V-Memory Loop</td>
</tr>
<tr>
<td>V gm</td>
<td>Mechanical Tape Reader &quot;Verify&quot; Button</td>
</tr>
<tr>
<td>V gp</td>
<td>Photoelectric Reader &quot;Verify&quot; Button</td>
</tr>
<tr>
<td>V r4</td>
<td>V-Loop Read Amplifier Output from 4-word Read Head</td>
</tr>
<tr>
<td>V r8</td>
<td>V-Loop Read Amplifier Output from 8-word Read Head</td>
</tr>
<tr>
<td>V w</td>
<td>V-Loop Write Amplifier Input</td>
</tr>
<tr>
<td>W 0-63</td>
<td>Memory Write Amplifiers</td>
</tr>
<tr>
<td>X 0</td>
<td>Origin Flip-Flop</td>
</tr>
<tr>
<td>X 1, 2</td>
<td>X-Register</td>
</tr>
<tr>
<td>X 9</td>
<td>Punch Flip-Flop</td>
</tr>
<tr>
<td>X 10</td>
<td>Type Flip-Flop</td>
</tr>
<tr>
<td>X 20</td>
<td>Blinker Flip-Flop</td>
</tr>
<tr>
<td>X 40</td>
<td>X-Register</td>
</tr>
<tr>
<td>X 41</td>
<td>X-Register</td>
</tr>
<tr>
<td>X or</td>
<td>Origin Pulse Amplifier</td>
</tr>
</tbody>
</table>
FUNCTIONAL COMPONENT LISTING (CONT)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xd</td>
<td>X-Register Display Selection Switch Setting</td>
</tr>
<tr>
<td>Xr</td>
<td>X-Register Read Amplifier Output</td>
</tr>
<tr>
<td>Xw</td>
<td>X-Register Write Amplifier Input</td>
</tr>
<tr>
<td>Y 0-12</td>
<td>&quot;Preset Stop&quot; Bit Agreement Signals</td>
</tr>
<tr>
<td>Z 1, 2</td>
<td>Order Register</td>
</tr>
<tr>
<td>Z 41</td>
<td>Order Register</td>
</tr>
<tr>
<td>Z ch</td>
<td>Input Zeroing-Keyboard Stop Flip-Flop</td>
</tr>
<tr>
<td>Zd</td>
<td>Display Selection Switch Setting for Instruction Register</td>
</tr>
<tr>
<td>Zr</td>
<td>Z-Register Read Amplifier Output</td>
</tr>
<tr>
<td>Zw</td>
<td>Z-Register Write Amplifier Input</td>
</tr>
</tbody>
</table>

DEFINITION OF FUNCTIONAL COMPONENTS

A₀ Inhibit Fill Flip-Flop:

(a) Used in decimal fill to prevent decimal point from being interpreted as an ENTER code following a plus or minus sign:*  

```
<table>
<thead>
<tr>
<th>i</th>
<th>1 1</th>
<th>0 0</th>
<th>d3</th>
<th>41</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>C</td>
</tr>
</tbody>
</table>
```

(b) Prevents an ENTER code after a decimal point from turning on I₅ a second time. I₅ is first turned on by a decimal point code, and places the number in the X-register for storage in memory. It is the habit of programmers to hit the ENTER button at the end of a fill process. The ENTER button would ordinarily turn on I₅ a second time if not prevented by A₀. Turning on I₅ a second time would put random information into the X-register instead of the number to be filled.

---

*Decimal point normally ENTERS in decimal fill but will only enter if N₀ is true (fill decimal number flip-flop). A plus or minus sign turns on A₀. A₀ prevents I₁ from turning on. Therefore A₀ prevents the decimal point code from entering information into the memory when the decimal point code follows a plus or minus sign.
A1, 2, 3, 39, 40, 41

These six flip-flops form the two ends of the accumulator register; the remaining 35 bits of information which constitute a word are stored on the memory disk. With appropriate timing this register shifts from the high-numbered to low-numbered end. Information is entered into the accumulator register via A41 and the output is taken from A1. This register is used to hold intermediate and final results of all arithmetic operations.

A42 Logical Function of Flip-Flops and Registers

A flip-flop used as an auxiliary to the A-register and utilized for certain operations as follows:
During left-shift operation it is added to the A-register:

\[ 1^a_{42} = A_1 M_4 D_2 T_{41a} ' C \]

During floating multiply while adding exponents, \( A_2 \) recirculates to \( A_{42} \):

\[
\begin{align*}
1^a_{42} &= A_2 M_{10} C \\
0^a_{42} &= A'_2 M_{10} C
\end{align*}
\]

During square root operation it copies the sum flip-flop:

\[ 1^a_{42} = S M_2 C \]

Stores sign bit in decimal fill:

\[ 1^a_{42} = F'_3 F_1 N_0 P_0 E_2 \ E_1 \]

Store address; sets \( N_7 \) to select first or second address to be stored; \( A_{42} \) RIGHT, second address; \( A'_{42} \) LEFT, first address.

copies \( G_0 \) information from \( Z \)-register:

\[ 1^a_{42} = Z'_{12} I_4 ' N_9 C \]

Determines decimal command readout.

\( A_n \) Delay Flip-Flop (Alphanumeric Input)

Used in alphanumeric fill for read-Y (typewriter) and read-Z (photoreader) instructions:

Allows alphanumeric input 1 to 8 five-bit characters from keyboard:

\[ 1^n = M_7 D_6 D_3 ' D_1 L_1 C \]

\( A^n \) is also one-set by an \( F \) code from typewriter or tape reader when alphanumeric fill is desired from this source:

\[ 1^n = F'_5 F_4 F_3 F_2 ' F_1 C_1 t \]
A. Delay Flip-Flop

This flip-flop is one-set for alphanumeric fill. After 8 characters have been struck, it remains one-set while the format is changed to command fill in order to enter the information (the alphanumeric flip-flop, $A_n$, is zero-set to receive the enter code). This flip-flop then functions to reset the input format to alphanumeric.

B₁, 2, 40, 41

These four flip-flops form the two ends of the B-register while the remaining 37 bits of information which constitute a word are stored on the memory disk. With appropriate timing this register shifts one bit at a time from the high-numbered end to the low-numbered end. Information is entered into this register via B₄₁ and the output is taken from B₁. This register is used to hold the operand and/or intermediate results of arithmetic operations.

B₄₂

This flip-flop is used in division to hold the sign of the quotient. It is used as an extension of the B-register during floating-multiply, floating-division, and square root.

B. Photoreader Clutch-Brake Flip-Flop

This flip-flop is used to energize and de-energize the solenoid on the photoreader; hence the use of this flip-flop is to engage the clutch or apply the brake to the tape in the photoreader.

C₁ - C₆ Channel Register Flip-Flops

This register consists of a series of flip-flops. Its purpose is to indicate which one of the 64 channels in memory to read from or write into.

C. Clock Jitter Flip-Flop

This flip-flop is used to delay the clock pulse by as much as 25 percent of the clock pulse width during marginal testing. The effect of this flip-flop is to provide an uneven clock pulse rate for the marginal testing.
C Input Counter Reset Flip-Flop

This flip-flop is used to set the H-counter to a count of 0001 and the E-counter to a count of 100 during input and output operation.

C Code Conversion Time Flip-Flop

Because the typewriter and photoreader accept standard teletype code, it is necessary to convert the teletype code to the binary representation which is used internally in this machine. This flip-flop provides the one-bit timing for the necessary conversion.

D Memory Read Gating Flip-Flop

This flip-flop is used to gate a command pair from memory into the Z-register. D0 is one-set if S0' is true at T41. D0 remains on for one complete word time (bit times 1 through 41) after which T41 resets it to zero.

D - Operational Code Register

This register consists of a series of flip-flops D1 - D6. Its purpose is to hold the command that is to be executed. By means of the binary configuration in this register, gates are set up in the computer to allow the operation desired.

D Loop Gating Flip-Flop

D10 allows the command pair read from the loop into K0 to be gated into the Z-register as required. D0 is one-set for a loop address if S7' indicates that the desired word will appear under one of the read heads during the next word time. D10 remains on for one complete word time (bit times 1-41) at which time T41 resets it to zero.

D Memory Write Gating Flip-Flop

This flip-flop is used to gate information into the memory write flip-flops. From there it enters the memory write amplifiers and is written into memory. D11 comes on only when sector agreement has occurred and stays on for one word time after which T40 resets it to zero.
E Add-Subtract Indicator Flip-Flop

This flip-flop is used to establish whether absolute addition or absolute subtraction is required when addition or subtraction is to be performed; that is, whether to add or subtract the two quantities algebraically.

E1-3 Input Sequence Counter

This counter counts the number of times the P0 flip-flop is one-set, hence this counter counts the number of legitimate characters being entered.

E20 Number Fill State Flip-Flop

During number fill it becomes necessary to multiply the contents of the R-register by ten. This flip-flop provides the necessary timing to accomplish this multiplication.

E30 Number Fill State Flip-Flop

This flip-flop provides the timing logic for adding the contents of the F-register to the contents of the R-register during number fill. This flip-flop also provides the timing for clearing the F-register.

E40 Number Fill State Flip-Flop

This flip-flop provides the timing logic for multiplying the contents of the B-register by ten during number fill. This flip-flop also includes the timing for copying the contents of the R-register into the A-register. Another function of this flip-flop is to reset the E-counter to 010.

F Fill Order Counter Flip-Flop

This flip-flop is used to establish the input format to receive a location and set the location counter.

F1 - F5 Input-Output Register

These five flip-flops constitute the input-output register.

(a) During input mode three sources provide information: typewriter, photoreader, and keyboard. The first two devices generate a teletype or Baudot code (see figure ) which is accepted
by the input register flip-flops $F_1$, $F_2$, $F_3$, $F_4$, $F_5$. They are set
in accordance with the Baudot code sent to them. At a certain bit
time later the state of these flip-flops is changed by the following
equations to effect a translation to a corresponding binary code used
internally in the computer.

<table>
<thead>
<tr>
<th>$f_1$</th>
<th>$f_1 = F_5 F_3' F_2' C_t C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_2$</td>
<td>$f_2 = F_5' F_2' C_t + F_1' F_2' F_1 C_t$</td>
</tr>
<tr>
<td>$f_3$</td>
<td>$f_3 = F_5' F_4' C_t + F_5' F_4' F_1 C_t$</td>
</tr>
<tr>
<td>$f_4$</td>
<td>$f_4 = F_5' F_3' F_2' C_t C + F_5' F_3' F_2' F_1 C_t$</td>
</tr>
<tr>
<td>$f_5$</td>
<td>$f_5 = F_5' F_4' F_2' C_t C + F_5' F_4' F_3' F_2' C_t C$</td>
</tr>
</tbody>
</table>

The translation process from Baudot to binary is accomplished by
specifically turning on or off $F$ flip-flops as required for the transla-
tion with a configuration of the $F$ flip-flops initially set by the tele-
type code input in combination with $C_t$ (code conversion-time flip-
flop) (figure ). In terms of the computer time base the $F$ flip-flops
are first set by the incoming information in Baudot code.

The $F$ flip-flops are again set in a translation process to a binary con-
figuration corresponding to the particular Baudot character (for in-
ternal logic of the computer which functions by use of the binary code).

$F_{c1}$, $F_{c2}$ format control flip-flop

The four possible combinations of these two flip-flops is used to
determine the format of the information to be output.
G₀ Half-Word Indicator Flip-Flop

This flip-flop is used to specify the first or second command depending on whether it is zero-set or one-set, respectively. When referring to a location which contains data (in contrast to a command), a zero is used to indicate the data is in command format and a one is used to indicate the data is in BCD format and occupies the location specified plus the following sector (two successive words).

G₁ - G₁₂ Location Counter

The location counter must contain the desired memory location to read from or write into. G₇ - G₁₂ determines the memory channel; G₁ - G₆ determines the sector.

H₀ Transfer of Control-Verify Input Flip-Flop

During the verify mode of operation this flip-flop is initially set to zero and then a comparison is made between the information previously recorded in memory and the information currently being read by the photoreader. Disagreement results in setting this flip-flop which activates the verify error light. This flip-flop is also used to reset the location counter (K₈ and G₀) when a transfer of control is indicated.

H₁-₄ Readout Control Flip-Flop

This four-flip-flop counter counts the number of characters shifted through the output register by counting the times the N₈ flip-flop is one-set (which is once per character).

I₁ Computer State Flip-Flop

This flip-flop is utilized for timing. During the time I₁ is true, the desired command pair is located and transferred from the memory to the Z-register. Since a full word was taken from memory, this function is not necessary for the second command in a pair.

I₂ Computer State Flip-Flop

This flip-flop is utilized for timing. During the time I₂ is true, the word information referred to by the address of the command is located and transferred from memory to the B-register.
I₃  Computer State Flip-Flop

This flip-flop is utilized for timing. During the time I₃ is true, the operation designated by the command is executed. Also at this time the location counter is augmented to its succeeding value unless a transfer-of-control command is encountered.

I₄  Computer State Flip-Flop

This flip-flop is utilized for timing. I₄ becomes true if the computer encounters a stop command or error. Therefore, I₄ is known as the noncompute flip-flop. During I₄ input-output operations such as computer fill, verify, or readout may be performed.

I₅  Computer State Flip-Flop

This flip-flop is utilized for timing. When a word has been filled into the accumulator and the ENTER button is depressed, I₅ becomes true and allows A₁ through A₄₁ to be copied into X₁ through X₄₁, and then the word enters the memory.

During floating square root, I₅ becomes true if the exponent does not have to be denormalized. If I₅ does not become true, then the exponent needs denormalizing.

J₀  Overflow Error Flip-Flop

This flip-flop is one-set when a number exceeding the capacity of the machine is called for; that is, when an attempt is made to enter too large a number or when the results of an arithmetic operation exceed absolute one. This flip-flop activates the overflow error light.

J₁₃  Output Error Flip-Flop

During output operations the computer directs a signal to the output device which in turn transmits to the computer a signal signifying the character being output. If the two signals are not identical, this flip-flop is set to activate the output error light.

K₀  Loop Read Flip-Flop

This flip-flop copies bit by bit the information being gated from the loop memory and acts as a buffer between the loop memory and the register to which the information is being transferred.
K Computation Carry Flip-Flop

This flip-flop is used in the logic to determine whether a carry or borrow arithmetic condition has been made in addition and subtraction.

K Carry Flip-Flop for Location Counter

This flip-flop comes on at bit time 2 and stays on until a zero is shifted into $G_1$ during $N_4$ timing. This is done to provide the one input to the counter, counting it up to the next location.

K Carry Flip-Flop for Computation Cycle Counter

This flip-flop injects the "one" which makes the $Q$-counter a one-input adder. It serves also to inject the zero to the $Q$-counter the first word time following the execution of a command or arithmetic operation.

K Carry Flip-Flop for Sector Counter

This flip-flop comes on at bit time 2 and stays on until a zero is shifted into $S_1$ during $N_1$ timing. This is done to provide the one input to the sector counter.

L L-Loop Memory Flip-Flops

These flip-flops are used to delay the information by three bit times in the recirculation path to compensate for the difference in spacing between the write head and the two read heads. $L_2$ acts as a buffer between the memory and the $K_0$ flip-flop.

L L-Loop 4-Word Read Flip-Flop

This flip-flop is used as a buffer between the four-word read head and the $K_0$ flip-flop.

L Loop-Memory Selection Flip-Flop

This flip-flop selects whether the information will be gated out of the main memory or from one of the high-speed loops.

L L-V-Loop Selection Flip-Flop

If the high-speed loop memory is indicated, this flip-flop is used to determine whether the L-loop or V-loop is indicated.
$M_0$ Operation-Type Flip-Flop

This flip-flop is used to specify the 00, 01, 02, and 03 code commands. It is used to decrease the number of inputs to "and" gates in the logic.

$M_1$ Operation-Type Flip-Flop

This flip-flop is used to specify the command codes 10 through 17 inclusive.

$M_2$ Operation-Type Flip-Flop

This flip-flop is used to specify the command codes 20 through 27 inclusive.

$M_3$ Operation-Type Flip-Flop

This flip-flop is used to specify the command codes 30 through 37 inclusive.

$M_4$ Operation-Type Flip-Flop

This flip-flop is used to specify the command codes 40 through 47 inclusive.

$M_5$ Operation-Type Flip-Flop

This flip-flop is used to specify the command codes 50 through 57 inclusive.

$M_6$ Operation-Type Flip-Flop

This flip-flop is used to specify the command codes 60 through 67.

$M_7$ Operation-Type Flip-Flop

This flip-flop is used to specify the command codes 70 through 77.

$M_8$ Operation-Type Flip-Flop

This flip-flop is used to specify the command codes 04, 06, 44, and 46.
M₉ Operation-Type Flip-Flop

This flip-flop is used to specify the command codes 05 and 07.

M₁₀ Operation-Type Flip-Flop

This flip-flop identifies the operation of adding or subtracting the exponents and storing the resultant exponent in the X-register during floating multiplication and division. This operation takes place in two word times.

Mᵣ Memory Read Flip-Flop

This flip-flop is used to copy the information being gated from main memory. It acts as a buffer between the main memory and the register to which the information is being directed.

Mₘ₁ Main Memory Write Flip-Flop

This flip-flop is used to write a binary "0" in main memory when it is zero-set.

Mₘ₂ Main Memory Write Flip-Flop

This flip-flop is used to write a binary "1" in main memory when it is zero-set.

N₀ Fill Decimal Number Flip-Flop

This flip-flop establishes the input format so that the computer will accept a decimal integer and transform it to a four-place binary equivalent.

N₁ Sector Counter Timing Flip-Flop

This flip-flop comes on at bit times 3 or 23 and stays on through bit times 8 or 28 respectively, depending on the half-word indicated. During this time it allows the sector counter to shift and augment one count. It is also a timing gate interval for N₅ and sector selector flip-flops.
$N_2$ Operation Code Timing Flip-Flop

This flip-flop comes on at bit times 14 or 34 and stays on through bit times 19 or 39 respectively, depending on the half-word indicated. During these times it allows the operational code register to receive new information from the order register and to shift out the old.

$N_4$ Location Counter Timing Flip-Flop

This flip-flop comes on at bit times 3 or 23 and stays on through bit times 14 or 34 respectively depending on the half-word indicated. During these times it allows shifting and recirculating of the location counter.

$N_5$ Channel Register Timing Flip-Flop

This flip-flop comes on at bit times 7 or 27 and stays on through bit times 14 or 34 respectively, depending on the half-word indicated. During this time it allows the channel register to receive new information from the location counter and to shift out the old. It is also a gate for the operation code timing flip-flop $N_2$.

$N_6$ Output Timing Flip-Flop

This flip-flop comes on at bit times 3 or 23 and stays on through bit times 7 or 27 respectively, depending on the half-word indicated. During these times it allows shifting of the input and output register ($F_1 - F_5$).

$N_7$ Partial Substitution Order Flip-Flop

This flip-flop is used for the three one-word store commands, STO (store word), STA (store address), and SAX (store contents of the accumulator and exchange contents of the A- and X-registers). The use of this flip-flop is to provide the timing necessary to gate the information from $A_2$ to $B_1$ from which it is gated into memory.

$N_8$ Readout Control Flip-Flop

This flip-flop provides conversion timing in the output register to convert from the internal representation of the computer to the tele-type code. In turn, this flip-flop provides the input to the H-counter; that is, the H-counter counts the number of times the $N_8$ flip-flop is one-set.
N₈a Readout Timing Flip-Flop

This flip-flop is similar to N₈ except that it is used in BCD format output. (In BCD, the sign is legitimate only if it is minus. Plus signs are not punched or typed. The plus sign is used to generate a figs shift code.)

N₉ Sector Counter Timing Flip-Flop (Intermediate Timing Flip-Flop)

This flip-flop is turned on at bit times 2 or 22 depending on which half of the word is indicated. It is used as a timing gate interval for N₁, N₄, and N₆ timing flip-flops.

N₁₁ State Change Flip-Flop

This flip-flop is used to terminate the I₂, command interpretation state. The logic one-sets this flip-flop when the necessary interpretation of the command to be executed has been completed. This flip-flop becomes one-set during the last bit time of a word time. The following word time, it zero-sets both itself and the I₂ flip-flop which terminates the command interpretation state.

N₁₂ State Change Flip-Flop

This flip-flop is used to terminate the I₃, command execution state. During the last bit time of the last word time in the execution of an arithmetic operation or command, this flip-flop is one-set. The following bit time, as with N₁₁, it zero-sets both itself and the I₃ flip-flop simultaneously which terminates the command execution state.

O₀ Fill Orders (Command Fill) Flip-Flop

This flip-flop is used in conjunction with logic to establish the command format-type of input; that is, the first character which is entered is treated as the sign which is a binary bit, the next six characters are converted to octal, the next character is treated as the half-word indicator and fills as a binary bit, the next character is treated as the sign of the second command and enters as a binary bit, the next six characters are filled as octal information, and finally the last character is the half-word indicator of the second command filling the last binary bit.
O₁ Fill Orders Flip-Flop

When this flip-flop is one-set, the input information accepted by the computer is an octal character. When this flip-flop is zero-set, the computer accepts only binary information.

P₀ Legitimate Code Flip-Flop

This flip-flop is set for one word time if the character being input is a legitimate character for the format selected.

P₁ - P₆ Young Digit Counter

This counter is used to count the number of binary bits in each word. This counter identifies each of the 41 bits within a given word and furnishes appropriate signals for timing control.

P₃ Delay Flip-Flop

This flip-flop provides a delay for determining whether the character being input is legitimate. During output operation this flip-flop indicates termination of the command and clears the F (output) register.

P₅ Delay Flip-Flop

This flip-flop is used to reset the Q-counter to zero during input-output operation, other than alphanumeric.

P₆ Delay Flip-Flop

Operation of this flip-flop is similar to that of P₅ except this flip-flop is used during alphanumeric input-output.

Q₀ Computation Cycle Counter Flip-Flop

This flip-flop is used during floating addition and subtraction to determine the sign of the exponent and whether or not the exponent is zero.

Q₁ - Q₆ Computation Cycle Counter Flip-Flop

These six flip-flops constitute a one-input adder which counts the word times during the execution of commands in order that the operation being performed will be terminated at the proper time.
R-Register Flip-Flops

These flip-flops form the two ends of the R-register while the remainder of the register is contained on the memory disk. R42 and R43 are not normally part of this register, but act to extend the register under special conditions.

Readout Timing Flip-Flops

Rc1 is used to initiate a readout. Rc2 is used in timing the shifting of information from the R-register and the B-register. Rc3 is used in shifting the information into the nixies. Rc4 is used to clear the previous display in the nixies.

Sum-Difference Flip-Flop

During addition and subtraction, this flip-flop creates the sum or difference.

Memory Read Sector Selection Flip-Flop

This flip-flop is one-set, if during sector comparison, sector agreement is not reached. At the end of each word, T41 zero-sets S0. Therefore, if the next word provides us with sector agreement, S0 remains zero-set. This provides D0 or K0 with the necessary terms to start gating information from memory.

Sector Counter

This counter is used to count the number of word sectors of which there are 64 per revolution, always indicating the sector address of the next word to pass under a given read head. Such a counter is sometimes called a one-input adder.

Loop Read Sector Selection Flip-Flop

These two flip-flops are used to determine whether the 4-word or 8-word read head is to be used for reading from the loops. The configuration of these flip-flops is dependent upon the sector comparison.

Memory Write Sector Selection Flip-Flop

This flip-flop is one-set, if during sector comparison, sector agreement is not reached. At the end of each word T41 zero-sets S11.
Therefore, if the next word provides sector agreement, \( S_{11} \) remains zero-set, indicating that the desired sector is the next to appear under the write head.

\[ S_x \text{ Readout Timing Flip-Flop} \]

This flip-flop is used to determine whether to shift a binary bit (when it is one-set) or whether to shift an octal character (when it is zero-set) during output operation.

\[ T_0 \text{ Tape Reader "On" Flip-Flop} \]

This flip-flop is used to indicate to the computer that the photoelectric tape reader is turned on.

\[ T_{1, 2, 40, 41, 41a} \text{ Digit Time Flip-Flop} \]

These flip-flops are used throughout the computer logic as a timing reference. The digit counter could be used, but to save on logic terms the \( T \) flip-flops were used. They provide a term to represent the beginning (\( T_1 \)) and a term to represent the end (\( T_{41} \)) of each word.

\[ T_x \text{ Readout Timing Flip-Flop} \]

During the time this flip-flop is one-set, information is shifted into the \( F \) (output) register. When this flip-flop is false during output, information is copied into the nixies.

\[ \text{TPP Tape Punch Pulse Flip-Flop} \]

This flip-flop is used to indicate to the computer that either the punch or typewriter is in the process of punching or typing.

\[ U_{1, 2, 41} \text{ Computation Cycle Time Flip-Flops} \]

During the execution of commands certain word times appear in numerous logic gates. \( U_1 \) and \( U_2 \) represent the first and second word times, respectively, of the execution of commands because of the importance of these word times. \( U_{41} \) is used to represent the last word time during the execution of a command.

\[ V_{1, 2, 2a, 41} \text{ V-Loop Flip-Flops} \]

Identical to the \( L \)-flip-flops of the same number.
X₀ Origin Flip-Flop

This flip-flop is used to initialize the Young digit counter and the sector counter. It also provides us with a reference point on the memory disk. \( X₀ = X₀r \)

X₁ - X₄₁ Register

This register has several uses. First, it is used as a buffer register for information coming from an input device. Second, it is used for the exponent portion of a floating-point number when the number is coming from memory.

X₉ Punch Flip-Flop

This flip-flop is for initializing. When the punch coding solenoids have copied the output register \( (F₁ - F₅) \) X₉ becomes true. This activates the punch clutch magnet and initializes the mechanics of punching.

X₁₀ Type Flip-Flop

This flip-flop is for initializing. When the typewriter coding solenoids have copied the output register \( (F₁ - F₅) \) X₁₀ becomes true. This activates the typewriter clutch magnet and initializes the mechanics of typing.

X₂₀ Blinker Flip-Flop

This flip-flop is used to blink the error-trouble indicator lights.

X₀r Origin Pulse Amplifier

X₀r is the amplifier used to amplify the origin pulse from the memory disk, and supply a signal to the origin flip-flop X₀.

Z₁ - Z₄₁ Order Register

This register is comprised of 41 bit positions, 38 of which are contained on the memory disk. The other 3 are held in flip-flop's Z₁, Z₂, and Z₄₁. A read and write head is used as a recirculating register. Its purpose is to excerpt from memory the selected command word.
This flip-flop is used to zero the $F$ output register after each character is entered. This flip-flop is one-set by the $E$-counter in order to accomplish this purpose.
## APPENDIX II. LOGIC EQUATIONS

### INDEX FOR RECOMP II LOGICAL EQUATIONS

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 0</td>
<td>Inhibit Fill Flip-Flop</td>
<td>477</td>
</tr>
<tr>
<td>A 1-3</td>
<td>A Register</td>
<td>400-3</td>
</tr>
<tr>
<td>A 39</td>
<td>A Register</td>
<td>403</td>
</tr>
<tr>
<td>A 40</td>
<td>A Register</td>
<td>404</td>
</tr>
<tr>
<td>A 41</td>
<td>A Register</td>
<td>405-8</td>
</tr>
<tr>
<td>A 42</td>
<td>A Register</td>
<td>409</td>
</tr>
<tr>
<td>A-d</td>
<td>A Register Display Selection Switch Setting</td>
<td>456</td>
</tr>
<tr>
<td>A n</td>
<td>Delay Flip-Flops</td>
<td>403</td>
</tr>
<tr>
<td>A r</td>
<td>A Register Read Power Amplifier Output</td>
<td>457</td>
</tr>
<tr>
<td>A w</td>
<td>A Register Write Amplifier Input</td>
<td>403</td>
</tr>
<tr>
<td>B 1-2</td>
<td>B Register</td>
<td>410</td>
</tr>
<tr>
<td>B 40</td>
<td>B Register</td>
<td>411</td>
</tr>
<tr>
<td>B 41</td>
<td>B Register</td>
<td>412-4</td>
</tr>
<tr>
<td>B 42</td>
<td>B Register</td>
<td>415-6</td>
</tr>
<tr>
<td>B c</td>
<td>Photoreader Clutch-Brake Flip-Flop</td>
<td>478</td>
</tr>
<tr>
<td>B r</td>
<td>B Register Read Power Amplifier Output</td>
<td>410</td>
</tr>
<tr>
<td>B w</td>
<td>B Register Write Amplifier Input</td>
<td>411</td>
</tr>
<tr>
<td>C 1</td>
<td>Channel Register</td>
<td>371</td>
</tr>
<tr>
<td>C 2,3,4</td>
<td>Channel Register</td>
<td>372-3</td>
</tr>
<tr>
<td>C 5,6</td>
<td>Channel Register</td>
<td>373-4</td>
</tr>
<tr>
<td>C b</td>
<td>Fill Commands Button</td>
<td>375</td>
</tr>
<tr>
<td>C f</td>
<td>Command Setting of Format Switch</td>
<td>376</td>
</tr>
<tr>
<td>C j</td>
<td>Clock Jitter Flip-Flop</td>
<td>377</td>
</tr>
<tr>
<td>C r</td>
<td>Input Counter Reset Flip-Flop</td>
<td>378</td>
</tr>
<tr>
<td>C t</td>
<td>Code Conversion Time Flip-Flop</td>
<td>379</td>
</tr>
<tr>
<td>D 0</td>
<td>Memory Read Gating Flip-Flop</td>
<td>390</td>
</tr>
<tr>
<td>D 1,2</td>
<td>Operation Code Register</td>
<td>375-6</td>
</tr>
<tr>
<td>D 3,4,5</td>
<td>Operation Code Register</td>
<td>377-8</td>
</tr>
<tr>
<td>D 6</td>
<td>Operation Code Register</td>
<td>378</td>
</tr>
<tr>
<td>D 10</td>
<td>Loop Gating Flip-Flop</td>
<td>391</td>
</tr>
<tr>
<td>D 11</td>
<td>Memory Write Gating Flip-Flop</td>
<td>391</td>
</tr>
<tr>
<td>D f</td>
<td>Decimal Setting of Format Switch</td>
<td>391</td>
</tr>
<tr>
<td>D r</td>
<td>Reader Delay Flip-Flop</td>
<td>448</td>
</tr>
</tbody>
</table>
Index for RECOMP II Logical Equations (Cont)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E 0</td>
<td>Add-Subtract Indicator Flip-Flop</td>
<td>444-5</td>
</tr>
<tr>
<td>E 1-3</td>
<td>Input Sequence Counter</td>
<td>474-5</td>
</tr>
<tr>
<td>E 20</td>
<td>&quot;Number Fill&quot; State Flip-Flop</td>
<td>476</td>
</tr>
<tr>
<td>E 30</td>
<td>&quot;Number Fill&quot; State Flip-Flop</td>
<td>476</td>
</tr>
<tr>
<td>E 40</td>
<td>&quot;Number Fill&quot; State Flip-Flop</td>
<td>476</td>
</tr>
<tr>
<td>E pl-p5 and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E'tl-t5</td>
<td>&quot;Punch&quot; Echo Signals</td>
<td></td>
</tr>
<tr>
<td>E tl-t5 and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E'ttl-t5</td>
<td>&quot;Type&quot; Echo Signals</td>
<td></td>
</tr>
<tr>
<td>F 0</td>
<td>Fill Order Counter Flip-Flop</td>
<td>460</td>
</tr>
<tr>
<td>F 1</td>
<td>Input-Output Register</td>
<td>462-3</td>
</tr>
<tr>
<td>F 2</td>
<td>Input-Output Register</td>
<td>464-5</td>
</tr>
<tr>
<td>F 3</td>
<td>Input-Output Register</td>
<td>466-7</td>
</tr>
<tr>
<td>F 4</td>
<td>Input-Output Register</td>
<td>468-9</td>
</tr>
<tr>
<td>F 5</td>
<td>Input-Output Register</td>
<td>470-1</td>
</tr>
<tr>
<td>F cl, c2</td>
<td>Format Control Flip-Flop</td>
<td>472</td>
</tr>
<tr>
<td>G 0</td>
<td>Half-Word Indication Flip-Flop</td>
<td>451</td>
</tr>
<tr>
<td>G 1, 2</td>
<td>Location Counter Logic</td>
<td>385</td>
</tr>
<tr>
<td>G 3-10</td>
<td>Location Counter Logic</td>
<td>385-6</td>
</tr>
<tr>
<td>G 11, 12</td>
<td>Location Counter Logic</td>
<td>386-7</td>
</tr>
<tr>
<td>H 0</td>
<td>Transfer of Control-Verify Input Flip-Flop</td>
<td>446</td>
</tr>
<tr>
<td>H 1-4</td>
<td>Readout Control</td>
<td>452</td>
</tr>
<tr>
<td>H c</td>
<td>STOP COMPUTE Button</td>
<td></td>
</tr>
<tr>
<td>I 1</td>
<td>State Flip-Flop</td>
<td>436</td>
</tr>
<tr>
<td>I 2, 3</td>
<td>State Flip-Flop</td>
<td>437</td>
</tr>
<tr>
<td>I 4</td>
<td>State Flip-Flop</td>
<td>438</td>
</tr>
<tr>
<td>I 5</td>
<td>Input Buffer Fill Flip-Flop</td>
<td>457</td>
</tr>
<tr>
<td>J 0</td>
<td>Overflow and Error Flip-Flop</td>
<td>447</td>
</tr>
<tr>
<td>J 1-6</td>
<td>Sector Agreement Signals for Readout</td>
<td></td>
</tr>
<tr>
<td>J 7-12</td>
<td>Channel Set Signals for Readout</td>
<td></td>
</tr>
<tr>
<td>J 13</td>
<td>Output Error Flip-Flop</td>
<td>449-0</td>
</tr>
<tr>
<td>K 0</td>
<td>Loop Read Flip-Flop</td>
<td>435</td>
</tr>
<tr>
<td>K 1-5</td>
<td>Input Channel Switches 1 through 5</td>
<td></td>
</tr>
<tr>
<td>K a</td>
<td>Computation Carry Flip-Flop</td>
<td>443</td>
</tr>
<tr>
<td>K f</td>
<td>KEYBOARD FILL button</td>
<td></td>
</tr>
</tbody>
</table>
### Index for RECOMP II Logical Equations (Cont)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kg</td>
<td>Order Counter Carry Flip-Flop</td>
<td>387</td>
</tr>
<tr>
<td>K p1-p5</td>
<td>Photoreader Input Channels 1 through 5</td>
<td>383</td>
</tr>
<tr>
<td>K q</td>
<td>Computation Cycle Counter Carry Flip-Flop</td>
<td>370</td>
</tr>
<tr>
<td>K s</td>
<td>Sector Counter Carry Flip-Flop</td>
<td></td>
</tr>
<tr>
<td>L 1-2</td>
<td>L Memory Loop</td>
<td>428</td>
</tr>
<tr>
<td>L 2a</td>
<td>L Loop 4-Word Read Flip-Flop</td>
<td>428</td>
</tr>
<tr>
<td>L 41</td>
<td>L Memory Loop</td>
<td>429</td>
</tr>
<tr>
<td>L b</td>
<td>L Memory Loop</td>
<td>429</td>
</tr>
<tr>
<td>L p1</td>
<td>Loop-Memory Selection Flip-Flop</td>
<td>431</td>
</tr>
<tr>
<td>L p2</td>
<td>L-V Loop Selection Flip-Flop</td>
<td>431</td>
</tr>
<tr>
<td>L r4</td>
<td>L Loop Read Amplifier Output from 4-Word Read Head</td>
<td>429</td>
</tr>
<tr>
<td>L r8</td>
<td>L Loop Read Amplifier Output from 8-Word Read Head</td>
<td>429</td>
</tr>
<tr>
<td>L w</td>
<td>L Loop Write Amplifier Input</td>
<td>429</td>
</tr>
<tr>
<td>M 0-2</td>
<td>Operation Type Flip-Flops</td>
<td>395</td>
</tr>
<tr>
<td>M 3-5</td>
<td>Operation Type Flip-Flops</td>
<td>396</td>
</tr>
<tr>
<td>M 6-10</td>
<td>Operation Type Flip-Flops</td>
<td>398</td>
</tr>
<tr>
<td>M d</td>
<td>Display Selection Switch Setting for Memory</td>
<td></td>
</tr>
<tr>
<td>M r</td>
<td>Memory Read Flip-Flop</td>
<td>496</td>
</tr>
<tr>
<td>M ra</td>
<td>Memory Read Power Amplifier</td>
<td>492</td>
</tr>
<tr>
<td>M rb</td>
<td>Memory Read Power Amplifier</td>
<td>492</td>
</tr>
<tr>
<td>M rc</td>
<td>Memory Read Power Amplifier</td>
<td>493</td>
</tr>
<tr>
<td>M rd</td>
<td>Memory Read Power Amplifier</td>
<td>493</td>
</tr>
<tr>
<td>M re</td>
<td>Memory Read Power Amplifier</td>
<td>494</td>
</tr>
<tr>
<td>M rf</td>
<td>Memory Read Power Amplifier</td>
<td>494</td>
</tr>
<tr>
<td>M rg</td>
<td>Memory Read Power Amplifier</td>
<td>495</td>
</tr>
<tr>
<td>M rh</td>
<td>Memory Read Power Amplifier</td>
<td>495</td>
</tr>
<tr>
<td>M r0-r63</td>
<td>Memory Read Amplifier Outputs from Channels 0-63 Respectively</td>
<td>496</td>
</tr>
<tr>
<td>M s</td>
<td>Memory Sync Flip-Flop</td>
<td></td>
</tr>
<tr>
<td>M w1</td>
<td>Main Memory Write Flip-Flop</td>
<td>479</td>
</tr>
<tr>
<td>M w2</td>
<td>Main Memory Write Flip-Flop</td>
<td>480</td>
</tr>
<tr>
<td>N 0</td>
<td>Fill Decimal Number Flip-Flop</td>
<td>458</td>
</tr>
<tr>
<td>N 1</td>
<td>Sector Counter Timing Flip-Flop</td>
<td>370</td>
</tr>
<tr>
<td>N 2</td>
<td>Operation Code Timing Flip-Flop</td>
<td>379</td>
</tr>
<tr>
<td>N 4</td>
<td>Location Counter Timing Flip-Flop</td>
<td>388</td>
</tr>
<tr>
<td>N 5</td>
<td>Channel Register Timing Flip-Flop</td>
<td>374</td>
</tr>
</tbody>
</table>
Index for RECOMP II Logical Equations (Cont)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>N 6</td>
<td>Output Timing Flip-Flop</td>
<td>392</td>
</tr>
<tr>
<td>N 7</td>
<td>PS Order Flip-Flop</td>
<td>451</td>
</tr>
<tr>
<td>N 8</td>
<td>Readout Control</td>
<td>453</td>
</tr>
<tr>
<td>N 8a</td>
<td>Readout Timing Flip-Flop</td>
<td>455</td>
</tr>
<tr>
<td>N 9</td>
<td>Sector Counter Timing Flip-Flop</td>
<td>370</td>
</tr>
<tr>
<td>N 11</td>
<td>State Change Flip-Flop</td>
<td>439</td>
</tr>
<tr>
<td>N 12</td>
<td>State Change Flip-Flop</td>
<td>440-1</td>
</tr>
<tr>
<td>N b</td>
<td>Fill NUMBERS Button</td>
<td></td>
</tr>
<tr>
<td>O 0</td>
<td>Fill Orders Flip-Flop</td>
<td>459</td>
</tr>
<tr>
<td>O 1</td>
<td>Fill Orders Flip-Flop</td>
<td>459</td>
</tr>
<tr>
<td>O f</td>
<td>Octal Setting of Format Switch</td>
<td></td>
</tr>
<tr>
<td>P 0</td>
<td>Legitimate Code Flip-Flop</td>
<td>461</td>
</tr>
<tr>
<td>P 1-6</td>
<td>Digit Counter</td>
<td>366-7</td>
</tr>
<tr>
<td>P d3</td>
<td>Delay Flip-Flop</td>
<td>456</td>
</tr>
<tr>
<td>P m</td>
<td>Delay Flip-Flop</td>
<td>456</td>
</tr>
<tr>
<td>P ma</td>
<td>Delay Flip-Flop</td>
<td>457</td>
</tr>
<tr>
<td>Q 0</td>
<td>Computation Cycle Counter</td>
<td>384</td>
</tr>
<tr>
<td>Q 1-6</td>
<td>Computation Cycle Counter</td>
<td>380-3</td>
</tr>
<tr>
<td>R 1-2</td>
<td>R Register</td>
<td>417-8</td>
</tr>
<tr>
<td>R 40</td>
<td>R Register</td>
<td>419</td>
</tr>
<tr>
<td>R 41</td>
<td>R Register</td>
<td>420-2</td>
</tr>
<tr>
<td>R 42</td>
<td>R Register</td>
<td>424</td>
</tr>
<tr>
<td>R 43</td>
<td>R Register</td>
<td>424</td>
</tr>
<tr>
<td>R b</td>
<td>Readout Activation Button</td>
<td></td>
</tr>
<tr>
<td>R c1-c4</td>
<td>Readout Timing Flip-Flops</td>
<td>453-4</td>
</tr>
<tr>
<td>R d</td>
<td>Display Selection Switch Setting for R Register</td>
<td></td>
</tr>
<tr>
<td>R gm</td>
<td>Mechanical Tape Reader ON Button</td>
<td></td>
</tr>
<tr>
<td>R gp</td>
<td>Photoelectric Tape Reader ON Button</td>
<td></td>
</tr>
<tr>
<td>R sm</td>
<td>Mechanical Tape Reader OFF Button</td>
<td></td>
</tr>
<tr>
<td>R sp</td>
<td>Photoelectric Reader OFF Button</td>
<td></td>
</tr>
<tr>
<td>R r</td>
<td>R Register Read Amplifier Output</td>
<td>419</td>
</tr>
<tr>
<td>R w</td>
<td>R Register Write Amplifier Inputs</td>
<td>419</td>
</tr>
<tr>
<td>S</td>
<td>Sum Flip-Flop</td>
<td>442</td>
</tr>
<tr>
<td>S 0</td>
<td>Memory Read Sector Selection Flip-Flop</td>
<td>389</td>
</tr>
<tr>
<td>S 1-6</td>
<td>Sector Counter</td>
<td>368-9</td>
</tr>
<tr>
<td>S 7</td>
<td>Loop Read Sector Selection Flip-Flop</td>
<td>389</td>
</tr>
</tbody>
</table>
### Index for RECOMP II Logical Equations (Cont)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>S 8</td>
<td>Loop Write Sector Selection Flip-Flop</td>
<td>390</td>
</tr>
<tr>
<td>S 11</td>
<td>Memory Write Sector Selection Flip-Flop</td>
<td>390</td>
</tr>
<tr>
<td>S c</td>
<td>START COMPUTE Button</td>
<td></td>
</tr>
<tr>
<td>S p</td>
<td>Preset Stop Switch ON</td>
<td></td>
</tr>
<tr>
<td>S sA</td>
<td>Sense Switch for SPA Command</td>
<td></td>
</tr>
<tr>
<td>S sB</td>
<td>Sense Switch for SPB Command</td>
<td></td>
</tr>
<tr>
<td>S sC</td>
<td>Sense Switch for SPC Command</td>
<td></td>
</tr>
<tr>
<td>S sD</td>
<td>Sense Switch for SPD Command</td>
<td></td>
</tr>
<tr>
<td>S x</td>
<td>Readout Timing Flip-Flop</td>
<td>455</td>
</tr>
<tr>
<td>S ul-3</td>
<td>Setup Buttons 1-3</td>
<td></td>
</tr>
<tr>
<td>SO</td>
<td>&quot;single Order&quot; Position of Operation Switch</td>
<td></td>
</tr>
<tr>
<td>SO'</td>
<td>&quot;Continuous&quot; Position of Operation Switch</td>
<td></td>
</tr>
<tr>
<td>T 0</td>
<td>Tape Reader ON Flip-Flop</td>
<td>478</td>
</tr>
<tr>
<td>T 1</td>
<td>Sync Bit Time Flip-Flop</td>
<td>393</td>
</tr>
<tr>
<td>T 2</td>
<td>LS Bit Time Flip-Flop</td>
<td>393</td>
</tr>
<tr>
<td>T 40</td>
<td>MS Bit Time Flip-Flop</td>
<td>393</td>
</tr>
<tr>
<td>T41, 41a</td>
<td>Sign Bit Time Flip-Flops</td>
<td>393</td>
</tr>
<tr>
<td>T d</td>
<td>Initial Time Delay and Error Reset Button</td>
<td></td>
</tr>
<tr>
<td>T g</td>
<td>Typewriter ON Button</td>
<td></td>
</tr>
<tr>
<td>T s</td>
<td>Typewriter OFF Button</td>
<td></td>
</tr>
<tr>
<td>T p</td>
<td>&quot;Punch&quot; Echo Timing Pulse</td>
<td></td>
</tr>
<tr>
<td>T t</td>
<td>&quot;Type&quot; Echo Timing Pulse</td>
<td></td>
</tr>
<tr>
<td>T x</td>
<td>Readout Timing Flip-Flop</td>
<td>454</td>
</tr>
<tr>
<td>TPP</td>
<td>Tape Punch Pulse Flip-Flop</td>
<td>455</td>
</tr>
<tr>
<td>TPP k</td>
<td>Tape Punch Pulse for Keyboard</td>
<td></td>
</tr>
<tr>
<td>TPP p</td>
<td>Tape Punch Pulse for Punch</td>
<td></td>
</tr>
<tr>
<td>TPP rm</td>
<td>Tape Punch Pulse for Mechanical Reader</td>
<td></td>
</tr>
<tr>
<td>TPP rp</td>
<td>Tape Punch Pulse for Photoelectric Reader</td>
<td></td>
</tr>
<tr>
<td>TPP t</td>
<td>Tape Punch Pulse for Typewriter</td>
<td></td>
</tr>
<tr>
<td>TPP 5</td>
<td>Input Sampling Signal = (TPP)P5</td>
<td></td>
</tr>
<tr>
<td>U -12</td>
<td>Computation Word Time Flip-Flop</td>
<td>394</td>
</tr>
<tr>
<td>U 41</td>
<td>Last Divide Word Time Flip-Flop</td>
<td>394</td>
</tr>
<tr>
<td>V 1-2</td>
<td>V Memory Loop</td>
<td>432</td>
</tr>
<tr>
<td>V 2a</td>
<td>V Loop 4-Word Read Flip-Flop</td>
<td>432</td>
</tr>
<tr>
<td>V 41</td>
<td>V Memory Loop</td>
<td>433-4</td>
</tr>
<tr>
<td>V gm</td>
<td>Mechanical Tape Reader VERIFY Button</td>
<td></td>
</tr>
<tr>
<td>V gp</td>
<td>Photoelectric Reader VERIFY Button</td>
<td></td>
</tr>
<tr>
<td>V r4</td>
<td>V Loop Read Amplifier Output from 4-Word</td>
<td>432</td>
</tr>
</tbody>
</table>

Read Head
<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>V r8</td>
<td>V Loop Read Amplifier Output from 8-Word Read Head</td>
<td>432</td>
</tr>
<tr>
<td>V w</td>
<td>V Loop Write Amplifier Input</td>
<td>432</td>
</tr>
<tr>
<td>W 0-63</td>
<td>Memory Write Amplifiers</td>
<td>481-491</td>
</tr>
<tr>
<td>X 0</td>
<td>Origin Flip-Flop</td>
<td>366</td>
</tr>
<tr>
<td>X 1, 2</td>
<td>X Register</td>
<td>425</td>
</tr>
<tr>
<td>X 9</td>
<td>Punch Flip-Flop</td>
<td>392</td>
</tr>
<tr>
<td>X 10</td>
<td>Type Flip-Flop</td>
<td>392</td>
</tr>
<tr>
<td>X 20</td>
<td>Blinker Flip-Flop</td>
<td>456</td>
</tr>
<tr>
<td>X 40</td>
<td>X Register</td>
<td>425</td>
</tr>
<tr>
<td>X 41</td>
<td>X Register</td>
<td>426-8</td>
</tr>
<tr>
<td>X or</td>
<td>Origin Pulse Amplifier</td>
<td>366</td>
</tr>
<tr>
<td>X d</td>
<td>X Register Display Selection Switch Setting</td>
<td></td>
</tr>
<tr>
<td>X r</td>
<td>X Register Read Amplifier Output</td>
<td>425</td>
</tr>
<tr>
<td>X w</td>
<td>X Register Write Amplifier Input</td>
<td>425</td>
</tr>
<tr>
<td>Y 0-12</td>
<td>&quot;Preset Stop&quot; Bit Agreement Signals</td>
<td></td>
</tr>
<tr>
<td>Z 1, 2</td>
<td>Order Register</td>
<td>379</td>
</tr>
<tr>
<td>Z 41</td>
<td>Order Register</td>
<td>380</td>
</tr>
<tr>
<td>Z ch</td>
<td>Input Zeroing-Keyboard Stop Flip-Flop</td>
<td>473</td>
</tr>
<tr>
<td>Z d</td>
<td>Display Selection Switch Setting for Instruction Register</td>
<td></td>
</tr>
<tr>
<td>Z r</td>
<td>Z Register Read Amplifier Output</td>
<td>379</td>
</tr>
<tr>
<td>Z w</td>
<td>Z Register Write Amplifier Input</td>
<td>379</td>
</tr>
</tbody>
</table>
RECOMP II LOGIC EQUATIONS

ORIGIN FLIP-FLOPS

\[ X_0 = \overline{X_0} \]
\[ \overline{X_0} = X_0' \]

\[ X_{OR} \] ORIGIN PULSE AMPLIFIER

YOUNG DIGIT COUNTER

\[ P_1 = P_5' P_1' X_0' + P_6' P_5' P_4' P_3' X_0' \]
\[ = P_5' P_1 + X_0 \]
\[ = P_5' P_1 + X_0 \]

\[ P_2 = P_1' X_0' \]
\[ = P_1' + X_0 \]
YOUNG DIGIT COUNTER (Cont)

\[ P_3 \quad 1P_3 = P_2 \]
\[ 0P_3 = P'_2 \]
\[ P_4 \quad 1P_4 = P_3 \times \bar{X}_0 \]
\[ 0P_4 = P'_3 \]
\[ + \bar{X}_0 \]
\[ P_5 \quad 1P_5 = P_4 \times \bar{X}_0 \]
\[ 0P_5 = P'_4 \]
\[ + \bar{X}_0 \]
\[ P_6 \quad 1P_6 = P'_6 P_5 P'_4 P'_3 P'_2 \]
\[ 0P_6 = T41_a \]

CIRCUIT BOARD LOCATION

264-19
263-19
265-18
263-34
263-34
261-18
260-34
260-34
159-9
259-19
SECTOR COUNTER

\[ S_1 \quad 1S_1 = S_2 \ N_1 \]
\[ oS_1 = S'_2 \ N_1 \]
\[ + X_o \]
\[ 402-18 \]

\[ S_2 \quad 1S_2 = S_3 \ N_1 \]
\[ oS_2 = S'_3 \ N_1 \]
\[ + X_o \]
\[ 203-18 \]

\[ S_3 \quad 1S_3 = S_4 \ N_1 \]
\[ oS_3 = S'_4 \ N_1 \]
\[ + X_o \]
\[ 204-18 \]
<table>
<thead>
<tr>
<th>SECTOR COUNTER (Cont)</th>
<th>CIRCUIT BOARD LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_4$ $S_4 = S_5 N_1$</td>
<td>405-18</td>
</tr>
<tr>
<td>$S_4$ $S_5 N_1$</td>
<td>206-34</td>
</tr>
<tr>
<td>$S_5$ $S_5 = S_6 N_1$</td>
<td>208-18</td>
</tr>
<tr>
<td>$S_5$ $S_6 N_1$</td>
<td>408-18</td>
</tr>
<tr>
<td>$S_6$ $S_6 = S_1 K_5 \cdot N_1$</td>
<td>222-12</td>
</tr>
<tr>
<td>$S_6$ $S_1 K_5 N_1$</td>
<td>418-12</td>
</tr>
<tr>
<td>$S_6$ $S_1 K_5 N_1$</td>
<td>420-19</td>
</tr>
</tbody>
</table>
SECTOR COUNTER (Cont)

\[ K_S \]
\[ k_S = T_1 \]
\[ k_S = S_1' \times N_1 \]

SECTOR COUNTER TIMING FLIP-FLOPS

\[ N_1 \]
\[ n_1 = N_q \]
\[ n_1 = P_2 \times P_3 \]

\[ N_q \]
\[ n_q = G_0 \times T_1 \]
\[ + G_0' \times P_5 \times P_4' \times P_3' \times P_2' \times P_6' \]
\[ n_a = N_q \]
\[ C_1 = C_2 \cdot N_5 \]
\[ + J_7 R_{D1} \cdot D_3' \]
\[ + M_8 U_1' R_{41} X_{41} I_{5}' T_{2}' \]
\[ + M_8 U_1' R_{41} X_{41} T_{2}' I_{5}' \]

\[ C_1' = C_2' \cdot N_5 \]
\[ + R_b I_4 \]
\[ + M_8 T_{41} \]
\[ + M_8 U_1' C_{2}' T_{40} \]
\[ + M_8 U_1' C_{3}' T_{40} \]
\[ + M_8 U_1' Q_0 Q_1' T_{40} \]

206-12
206-12
103-9
104-9
406-34
406-34
405-27
405-27
406-24
506-9
CHANNEL REGISTER (Cont)

\[ C_2 \quad _1 C_2 = C_3 N_5 \]
\[ + J_8 R_{c1} D_3' \]
\[ + A_2 M_8 U_1' \]
\[ + M_6 \]
\[ _0 C_2 = C_3 N_5 \]
\[ + R_b I_4 \]
\[ + M_8 T_{41} \]

\[ C_3 \quad _1 C_3 = C_4 N_5 \]
\[ + J_9 R_{c1} D_3' \]
\[ + B_2 M_8 U_1' \]
\[ _0 C_3 = C_4 N_5 \]
\[ + R_b I_4 \]
\[ + M_8 T_{41a} \]

CIRCUIT BOARD LOCATION

415-12

415-12

415-10

416-18

217-18

216-34

216-34

467-12

467-12

467-10

267-34

267-34

267-18
CHANNEL REGISTER (Cont)

\[ C_4 \cdot C_4 = C_5 \cdot N_5 \]
\[ + J_{10} \cdot R_{c1} \cdot D_3 \]
\[ + M_8 \cdot U_1 \cdot T_1 \]

\[ 263-10 \]
\[ 264-12 \]
\[ 264-12 \]

\[ 0 \cdot C_4 = C_5' \cdot N_5 \]
\[ + R_b \cdot I_4 \]
\[ + M_8 \cdot Q_2 \cdot X_1' \cdot T_1' \cdot U_1' \]
\[ + M_8 \cdot Q_2' \cdot X_1 \cdot T_1' \cdot U_1' \]
\[ + M_8 \cdot T_4' \]

\[ 264-34 \]
\[ 264-34 \]
\[ 164-25 \]
\[ 164-25 \]
\[ 268-34 \]

\[ C_5 \cdot C_5 = C_6 \cdot N_5 \]
\[ + J_{11} \cdot R_{c1} \cdot D_3 \]

\[ 477-18 \]
\[ 477-10 \]

\[ 0 \cdot C_5 = C_6' \cdot N_5 \]
\[ + R_b \cdot I_4 \]

\[ 277-34 \]
\[ 277-34 \]
CHANNEL REGISTER (Cont)

\[ c_6 = G_1 I_1 N_5 \]
\[ + Z_1 I_2 N_5 \]
\[ + J_{12} R_{c1} D_3' \]
\[ c_6' = G_1' I_1 N_5 \]
\[ + Z_1' I_2 N_5 \]
\[ + R_b I_4 \]

CHANNEL REGISTER TIMING FLIP-FLOPS

\[ n_5 = D_0' D_{10} D_{11} R_{c1} I_3' N_1 P_5 \]
\[ n_5' = P_5 P_4 P_3 \]

CIRCUIT BOARD LOCATION

265-12
464-12
464-12
464-10
176-18
276-10
\[ D_1 \cdot d_1 = D_2 \cdot N_2 \]

\[ + N_0 \cdot C_r \]

\[ + M_{8} \cdot U_1 \cdot C_{1} \cdot T_{41} \]

\[ + M_{10} \]

\[ + I_{5} \cdot I_{3} \cdot T_{41} \]

\[ + B_{42} \cdot I_{2} \cdot I_{4} \cdot T_{40} \]

\[ o \cdot d_1 = D_{2} \cdot N_{2} \]

\[ + R_{c1} \]

\[ + N_{12} \]

\[ + M_{9} \cdot M_{4} \]

\[ + M_{6} \cdot D_{6} \]

CIRCUIT BOARD LOCATION

508-T-6
OPERATION CODE REGISTER (Cont)

\[ D_2 \begin{align*}
1d_2 &= D_3 N_2 \\
&\quad + R_{c1} \\
&\quad + B_{42} I_4' I_2 T_{40} \\
0d_2 &= D_3' N_2 \\
&\quad + M_4 D_3' T_{41a} \\
&\quad + N_0 C_\lambda' \\
&\quad + M_0 \\
&\quad + M_1 U_2 \\
\end{align*} \]
### OPERATION CODE REGISTER (Cont)

<table>
<thead>
<tr>
<th>$D_3$</th>
<th>$d_3 = D_4 \cdot N_2$</th>
<th>211-18</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+ $M_3 \cdot D_2'$</td>
<td>213-18</td>
</tr>
<tr>
<td></td>
<td>+ $J_0 \cdot I_2 \cdot I_4' \cdot T_{40}$</td>
<td>211-24</td>
</tr>
<tr>
<td></td>
<td>+ $B_{42} \cdot I_4' \cdot I_2$</td>
<td>209-24</td>
</tr>
<tr>
<td></td>
<td>+ $U_2 \cdot N_4 \cdot M_7$</td>
<td>212-24</td>
</tr>
<tr>
<td></td>
<td>+ $M_1 \cdot T_{41}$</td>
<td>209-34</td>
</tr>
<tr>
<td>$o_3$</td>
<td>$d_3 = D_4 \cdot N_2$</td>
<td>212-34</td>
</tr>
<tr>
<td></td>
<td>+ $I_4$</td>
<td>212-34</td>
</tr>
<tr>
<td></td>
<td>+ $M_{10} \cdot D_2 \cdot U_2 \cdot T_{41}$</td>
<td>213-24</td>
</tr>
<tr>
<td></td>
<td>+ $M_6 \cdot D_6'$</td>
<td>213-10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_4$</th>
<th>$d_4 = D_5 \cdot N_2$</th>
<th>277-18</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+ $R_{C1}$</td>
<td>277-19</td>
</tr>
<tr>
<td></td>
<td>+ $B_{42} \cdot I_2 \cdot T_{40}$</td>
<td>277-10</td>
</tr>
<tr>
<td>$o_4$</td>
<td>$d_4 = D_5^t \cdot N_2$</td>
<td>479-18</td>
</tr>
<tr>
<td></td>
<td>+ $R_{C1} \cdot M_3' \cdot I_4$</td>
<td>479-18</td>
</tr>
</tbody>
</table>
### Operation Code Register (Cont)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Formula</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_5$</td>
<td>$d_5 = D_6 N_2 B_{42}' + I_4 + M_2 T_x$</td>
<td>266-12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>266-12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>264-18</td>
</tr>
<tr>
<td>$d_5 = D_6' N_2 + B_{42} I_4' I_2$</td>
<td>466-18</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>566-9</td>
</tr>
<tr>
<td>$D_6$</td>
<td>$d_6 = Z_2 N_2 + B_{42} I_4' I_2 T_{40}$</td>
<td>474-18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>468-27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>475-34</td>
</tr>
<tr>
<td>$d_6 = Z_2' N_2 + I_4 P_{d3}$</td>
<td>475-34</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>475-19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>475-19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>475-18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>475-19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>475-18</td>
</tr>
</tbody>
</table>
OPERATION CODE TIMING FLIP-FLOPS

\[ N_2 \quad 1n_2 = I_2 I_4' N_5 P_4 P_2 P_1' \]

\[ 0n_2 = P_2' P_1' \]

ORDER REGISTER

\[ Z_1 \quad 1z_1 = Z_2 M_5' + Z_4 B_{42} \]

\[ 0z_1 = \overline{Z_2} + M_5 B_{42} \]

\[ Z_2 \quad 1z_2 = Z_r \]

\[ 0z_2 = \overline{Z_r} \]

\[ \{ Z_r, \overline{Z_r}' \} \quad Z \text{ CHANNEL READ POWER AMPLIFIER OUTPUTS} \]

\[ Z_w \quad 1z_w = Z_{41} \]

\[ 0z_w = \overline{Z_{41}} \quad Z \text{ CHANNEL WRITE AMPLIFIER INPUTS} \]
ORDER REGISTER (Cont)

\[ Z_{41} = M_2 D_0 I_1 \]
\[ + Z_1 I'_1 \]
\[ + K_0 D_{10} I_1 \]

230-12

\[ 0 Z_{41} = M_2' D_0 I_1 \]
\[ + Z_1' I'_1 \]
\[ + K_0' D_{10} I_1 \]

231-12

COMPUTATION CYCLE COUNTER

\[ Q_1 = Q_2 N_1 M_8' \]
\[ + M_8 X_{41} T_1' \]

412-12

\[ 0 Q_1 = Q_2' N_1 M_8' \]
\[ + I_5 T_{41} \]
\[ + M_8 T_1 \]

413-12
COMPUTATION CYCLE COUNTER (Cont)

\[ q_2 = Q_3 N_1 M_8 \]
\[ + q_1 ' M_8 T_{40} \]
\[ + X_{41} M_8 U_1 T_1 \]
\[ oq_2 = Q_3 N_1 M_8 \]
\[ + I_5 T_{41} \]
\[ q_3 = Q_4 N_1 M_8 \]
\[ + M_8 I_5 U_1 C_2 T_{40} \]
\[ oq_3 = Q_4 N_1 M_8 \]
\[ + U_1 T_{41} \]
\[ (+ M_{10} T_{41})^* \]

417-10
418-27
418-27
418-10
420-18
115-25
115-25
212-10
213-12
213-12
COMPUTATION CYCLE COUNTER (Cont)

\[ Q_4 = Q_5 N_1 M_8^' \]
\[ + M_8 M_0 I_5 V_1 T_{41} \]
\[ 505-25 \]

\[ Q_4 = Q_5 N_1 M_8^' \]
\[ + M_8 Q_5 U_2 T_{41} \]
\[ + M_8 U_1 C_1 T_{41} \]
\[ 204-10 \]
\[ 104-25 \]

\[ Q_5 = Q_6 N_1 M_8^' \]
\[ + M_8 D_6 X_2 R_2 U_2 \]
\[ + M_8 D_6 U_2 X_2 Q_5 T_{40} \]
\[ 515-25 \]
\[ 511-18 \]

\[ Q_5 = Q_6 N_1 M_8^' \]
\[ + M_8 X_2 R_2 U_2 \]
\[ + M_8 U_2 R_2 Q_5 T_{40} \]
\[ 214-10 \]
\[ 214-24 \]
\[ 104-18 \]

NOTE: * Gates In Parentheses Are Redundant
COMPUTATION CYCLE COUNTER (Cont)

\[ Q_6 = Q_1 K_q' I_3 P_m' N_1 \]
\[ + Q_1 K_q I_3 P_m' N_1 \]
\[ + Q_1 K_q' N_1 \]
\[ + Q_1 K_q N_1 \]
\[ + I_3' \]
\[ + P_m \]
\[ + M_8 \]

\[ K_q \]
\[ Q_6 = T_1 M_0' M_q' M_10' \]
\[ + T_1 M_0' M_q' \]
\[ + 457-27 \]

\[ Q_6 = Q_1' N_1 \]
\[ 457-18 \]
COMPUTATION CYCLE COUNTER (Cont)

\[ Q_0 = Q_1 Z_1 I_4 D_3 N_1 + Q_1 Z_1 I_4 D_3 N_1 + D_3 T_1 + R_{9P} I_4 + V_{9P} I_4 + M_1 D_2 \]

\[ \text{CIRCUIT BOARD LOCATION} \]

157-25

257-10

257-27

257-27

258-18

256-34

257-24

256-12

256-12

256-34
ORDER COUNTER LOGIC

\[ G_1 \quad 1\!9_1 = G_2 \quad N_4 \]
\[ + F_5 \quad F_4 \quad F_2 \quad F_1 \quad I_4 \quad P_{d3} \quad T_{41a} \quad T_0' \]
\[ 0\!g_1 = G_2' \quad N_4 \]
\[ 263-18 \]

\[ G_2 \quad 1\!9_2 = G_3 \quad N_4 \]
\[ + F_5' \quad F_4' \quad F_3' \quad F_2 \quad I_4 \quad P_{d3} \quad T_{41a} \quad T_0' \]
\[ 0\!g_2 = G_3' \quad N_4 \]
\[ 259-10 \]

\[ G_3 \quad 1\!9_3 = G_4 \quad N_4 \]
\[ 0\!g_3 = G_4' \quad N_4 \]
\[ 273-18 \]

\[ G_4 \quad 1\!9_4 = G_5 \quad N_4 \]
\[ 0\!g_4 = G_5' \quad N_4 \]
\[ 272-18 \]

\[ G_5 \quad 1\!9_5 = G_6 \quad N_4 \]
\[ 0\!g_5 = G_6' \quad N_4 \]
\[ 271-18 \]
<table>
<thead>
<tr>
<th>ORDER COUNTER LOGIC (Cont)</th>
<th>( G_6 )</th>
<th>( G_7 )</th>
<th>( G_8 )</th>
<th>( G_9 )</th>
<th>( G_{10} )</th>
<th>( G_{11} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g_6 = G_7 \ N_4 )</td>
<td>( g_7 = G_8 \ N_4 )</td>
<td>( g_8 = G_9 \ N_4 )</td>
<td>( g_9 = G_{10} \ N_4 )</td>
<td>( g_{10} = G_{11} \ N_4 )</td>
<td>( g_{11} = G_{12} \ N_4 )</td>
<td></td>
</tr>
<tr>
<td>( g_6 = G_7 \ N_4 )</td>
<td>( g_7 = G_8 \ N_4 )</td>
<td>( g_8 = G_9 \ N_4 )</td>
<td>( g_9 = G_{10} \ N_4 )</td>
<td>( g_{10} = G_{11} \ N_4 )</td>
<td>( g_{11} = G_{12} \ N_4 )</td>
<td></td>
</tr>
<tr>
<td>( 270-18 )</td>
<td>( 470-18 )</td>
<td>( 269-18 )</td>
<td>( 469-18 )</td>
<td>( 453-18 )</td>
<td>( 452-18 )</td>
<td></td>
</tr>
<tr>
<td>( 460-18 )</td>
<td>( 460-18 )</td>
<td>( 259-18 )</td>
<td>( 459-18 )</td>
<td>( 254-18 )</td>
<td>( 253-18 )</td>
<td></td>
</tr>
</tbody>
</table>
ORDER COUNTER LOGIC (Cont)

\[ G_{12} = G_1 K_9 F_0' U_i N_4 T P P' + G_1 K_9 U_i N_4 + G_1 K_9 U_i N_4 + Z_1 U_2 N_4 + A_1 F_0 N_4 + G_1 K_9 F_0' U_i N_4 S_{ub} + I_4 (S_{u1} + S_{u2} + S_{u3}) \]

\[ K_9 = D_{11} I_1 T_1 + G_0 I_3 I_4' U_i T_1 + M_2' J_0' H_0 I_2 I_4 T_1 + (T_0' D_{11} I_1 T_1) \]

\[ 0 K_9 = G_1' N_4 + T_{410} \]

CIRCUIT BOARD LOCATION

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>520-18</td>
<td></td>
</tr>
<tr>
<td>420-27</td>
<td></td>
</tr>
<tr>
<td>420-27</td>
<td></td>
</tr>
<tr>
<td>419-10</td>
<td></td>
</tr>
<tr>
<td>521-18</td>
<td></td>
</tr>
<tr>
<td>118-9</td>
<td></td>
</tr>
<tr>
<td>219-27</td>
<td></td>
</tr>
<tr>
<td>219-27</td>
<td></td>
</tr>
<tr>
<td>219-10</td>
<td></td>
</tr>
<tr>
<td>220-18</td>
<td></td>
</tr>
<tr>
<td>558-25</td>
<td></td>
</tr>
<tr>
<td>558-25</td>
<td></td>
</tr>
<tr>
<td>558-9</td>
<td></td>
</tr>
<tr>
<td>458-24</td>
<td></td>
</tr>
<tr>
<td>258-34</td>
<td></td>
</tr>
<tr>
<td>258-34</td>
<td></td>
</tr>
</tbody>
</table>
ORDER COUNTER LOGIC (Cont)

\[ N_4 = H_0 I_4 U_2 N_9 \]
\[ + I_3 M_9 M_9 U_1 T_2 \]
\[ + F_5 F_0 I_1 T_1 \]
\[ + M_2 H_0 I_2 I_4 T_2 \]
\[ + I_1 N_9 F_0 \]
\[ + I_4 (S_{u1} + S_{u2} + S_{u3}) \]
\[ + M_7 M_9 T_2 \]

\[ n_4 = P_5 P_4 P_3 \]
SECTOR SELECTION FLIP-FLOPS

\[
S_0 = G_1 S_1 I_2 I_3' N_1 + G_1' S_1 I_2' I_3' N_1 + Z_1 S_1 I_1' I_4' N_1 + Z_1' S_1 I_1 I_4 N_1 + I_2 R_{c1} T_1
\]

\[
S_0 = T_{41} + E_3 J_3 J_2 J_1 R_{c1} I_2 T_2
\]

\[
S_7 = S_0 D_{10}' M_3' N_1 P_4 P_1' + I_4 I_2' + D_6 D_5 I_2 T_{40}
\]

\[
S_7 = T_{410}
\]
SECTOR SELECTION FLIP-FLOPS (Cont)

\[ S_8 \]
\[ S_8 = S_0 \, D_{10} \, M_3 \, N_1 \, P_2 \]
\[ S_8 = S_7 \, T_{41} \]
\[ + I_1 \, D_{10} \, T_{41} \]
\[ + N_{12} \]

\[ S_{11} \]
\[ S_{11} = S_0 \, N_1 \]
\[ + S_0 \, P_6 \, P_5 \, P_4 \]
\[ + J_0 \]
\[ S_{11} = T_{41} \]

MEMORY READ GATING FLIP-FLOPS

\[ D_0 \]
\[ D_0 = S_0 \, L_{P_1} \, I_1 \, I_4 \, T_{41} \]
\[ + D_3 \, S_0 \, L_{P_1} \, I_2 \, T_{41} \]
\[ + D_1 \, S_0 \, L_{P_1} \, I_2 \, T_{41} \]
\[ + S_0 \, L_{P_1} \, H_0 \, I_1 \, T_{41} \]
\[ + D_6 \, D_4 \, S_0 \, L_{P_1} \, I_2 \, T_{41} \]

\[ D_0 = D_0 \, T_{41} \]
LOOP GATING FLIP-FLOPS

\[ D_{10} = S_7 L_{P1} D_{10} I_3 D_3 T_{41} \]
\[ + S_8 L_{P1} D_{10} M_6 D_3 T_{41} \]
\[ + S_7 L_{P1} D_{10} I_3 D_1 T_{41} \]
\[ + S_7 L_{P1} D_{10} I_3 D_4 T_{41} + I_3 M_6 D_5 T_s \]
\[ o_d_{10} = D_{10} T_{41} \]

MEMORY WRITE GATING FLIP-FLOPS

\[ D_{11} = S_{11} F_0 H_0 I_1 I_4 T_{40} \]
\[ + S_{11} D_6 D_5 D_4 D_3 D_1 I_2 T_{40} \]
\[ + S_{11} M_6 D_3 T_{40} \]
\[ + S_{11} H_0 D_6 D_5 D_4 D_1 I_2 T_{40} \]
\[ o_d_{11} = D_{11} I_1 T_{40} \]
\[ + D_{11} M_6 D_3 T_{40} \]
\[ + M_3 U_2 T_{40} \]
\[ + D_3 Q_4 T_{40} + T_d \]

CIRCUIT BOARD LOCATION

503-18
504-18
505-18
505-9
403-24
403-18

116-9
115-18
212-24
116-18
217-27
217-27
217-12
217-12
217-19
OUTPUT TIMING FLIP-FLOPS

\[ N_{6} \quad 1 n_{6} = M_{7} D_{1} U_{1} N_{9} D_{6} \]
\[ 0 n_{6} = P_{2} P_{1} \]

PUNCH FLIP-FLOPS

\[ X_{9} \quad 1 x_{9} = M_{7} D_{3} X_{10} Q_{5} (T P P) T_{41} \]
\[ 0 x_{9} = (T P P) + T_{d} \]

\[ 105-9 \]
\[ 205-34 \]

TYPE FLIP-FLOPS

\[ X_{10} \quad 1 x_{10} = M_{7} D_{2} D_{1} Q_{5} (T P P) T_{41} \]
\[ 0 x_{10} = (T P P) + T_{d} \]

\[ 507-9 \]
\[ 409-34 \]
DIGIT TIME FLIP-FLOPS

\[ T_1 \quad 1t_1 = T_{41a} \]
\[ 0t_1 = T_1 \]

\[ T_2 \quad 1t_2 = T_1 \]
\[ 0t_2 = T_2 \]

\[ T_{40} \quad 1t_{40} = P_6 P_5 P_3 P_2 P_1 \]
\[ 0t_{40} = T_{40} \]

\[ T_{41} \quad 1t_{41} = T_{40} \]
\[ 0t_{41} = T_{41} \]

\[ T_{41a} \quad 1t_{41a} = T_{40} \]
\[ 0t_{41a} = T_{41a} \]

CIRCUIT BOARD LOCATION

\[ 267-19 \]
\[ 467-19 \]

\[ 478-18 \]
\[ 478-19 \]

\[ 564-9 \]
\[ 463-19 \]

\[ 415-19 \]
\[ 215-18 \]

\[ 266-19 \]
\[ 265-19 \]
COMPUTATION CYCLE TIME FLIP-FLOPS

$U_1 \quad 1u_1 = I_3'$

$+ M_8 \ U_1 'C_i T_{41a}$

$+ M_0 \ U_2 T_{41a}$

$+ M_q M_1 M_2 U_2 T_{41a}$

$+ M_q B_{42} Q_6 T_{41a}$

$0u_1 = I_3 \ U_1 I_5 'T_{41a}$

$272-19$

$273-27$

$273-27$

$173-25$

$173-25$

$272-24$

$U_2 \quad 1u_2 = I_3 \ U_1 T_{41} I_5 'N_{12} 'A_t'$

$0u_2 = U_2 T_{41}$

$508-9$

$409-18$

$U_{41} \quad 1u_{41} = M_2 D_i Q_6 Q_4 Q_1 T_{41a}$

$+ M_2 D_i Q_6 Q_4 Q_1 T_{41a}$

$0u_{41} = U_{41} T_{41a}$

$576-18$

$576-9$

$476-18$
OPERATION TYPE FLIP-FLOPS

\[ M_0 \]
\[ m_0 = J_0 D'_6 D'_5 D'_4 D'_3 I_2 N_{11} + M_8 U'_1 C'_1 T_{41a} I'_5 M'_0 \]
\[ o m_0 = N_{12} + T_d + M_0 U_2 T_{41a} \]
\[ 151-18 \]

\[ M_1 \]
\[ m_1 = J_0 D'_6 D'_5 D'_4 D'_1 I_2 N_{11} + M_{10} D_2 U_2 T_{41a} \]
\[ o m_1 = N_{12} + T_d + M_1 B_{42} T_{41a} + N_7 T_{41a} \]
\[ 167-18 \]

CIRCUIT BOARD LOCATION

151-9
452-19
452-12
452-12
167-18
267-24
267-12
267-12
267-10
263-18
OPERATION TYPE FLIP-FLOPS (Cont)

\[ M_2 \quad m_2 = J_0 D'_6 D'_5 D'_4 I_2 I'_4 N_{11} \]
\[ + F_5 F_4 F_3 F_2 A'_6 N_6 P_{d3} T_{41a} \quad 553-18 \]
\[ + M_{10} D'_2 U_2 T_{41a} \quad 455-27 \]
\[ + I_5 I'_3 T_{41a} \quad 455-27 \]
\[ 0 m_2 = T_d \]
\[ + U_{d1} T_{41a} \quad 455-34 \]

\[ M_3 \quad m_3 = J'_0 D'_6 D'_5 D'_4 I_2 N_{11} \]
\[ + R_{c1} I_2 N_{11} \quad 413-10 \]
\[ 0 m_3 = N_{12} \]
\[ + T_d \quad 213-34 \]
OPERATION TYPE FLIP-FLOPS (Cont)

M₄

\[ 1 m₄ = J'₀ D₆ D₅' D₄' I₂ N₁₁ \]
\[ + D₅' D₄' D₃ I₂ N₁₁ J₀' \]
\[ + M₀ M₈ U₂ T₄₁ \]
\[ + M₉ B₄₂ Q₆ T₄₁ \]
\[ 0 m₄ = N₁₂ \]
\[ + M₄ D₂ D₃' T₄₁ \]
\[ + T_d \]
\[ + M₈ M₀' Q₅ U₂ T₄₁ \]
\[ + M₈ M₀' U₁' C₁' T₄₁ \]
\[ + M₉ U₂ T₄₁ \]
\[ + I₅ T₄₁ \]

M₅

\[ 1 m₅ = J'₀ D₆ D₅' D₄' I₂ N₁₁ \]
\[ 0 m₅ = N₁₂ \]
\[ + T_d \]
\[ + D₃' U₂ T₄₀ \]

CIRCUIT BOARD LOCATION

120-9
121-9
220-24
219-24
222-34
222-27
222-34
119-18
122-9
222-10
222-27
169-18
270-19
269-19
274-10
OPERATION TYPE FLIP-FLOPS (Cont)

\[ M_6 \]
\[ I_{m_6} = J_0' D_6 D_5 D_4' I_2 N_{11} \]
\[ + M_4 D_3' D_2' D_1' T_{41a} \]
\[ + M_1 N_7 T_{41a} \]
\[ \theta m_6 = N_{12} \]
\[ + I_4 \]
\[ 167-9 \]

\[ M_7 \]
\[ I_{m_7} = J_0' D_6 D_5 D_4 I_2 N_{11} \]
\[ + J_0' D_6' D_5' D_4' D_1' I_2 N_{11} \]
\[ + D_5' T_x T_{41a} N_{12} \]
\[ + J_{13} I_4' \]
\[ \theta m_7 = N_{12} \]
\[ + I_4 \]
\[ + (TBS) J_{13}' \]
\[ + TPPp X_10' \]
\[ 154-9 \]

\[ M_8 \]
\[ I_{m_8} = D_5' D_3 D_4' D_1' I_2 N_{11} J_0' \]
\[ \theta m_8 = M_0 U_2 T_{41} \]
\[ + I_5 T_{41} \]
\[ + T_d \]
\[ 514-18 \]

398

508-T-6
OPERATION TYPE FLIP-FLOPS (Cont)

\[ m_9 = D_6 D_5 D_3 D_1 I_2 N_{11} J_0 D_4 \]
\[ + M_7 D_6 U_1 \]
\[ 578-18 \]
\[ 478-10 \]
\[ o m_9 = B_{42} Q_6 T_{41a} \]
\[ + M_7 U_2 \]
\[ + T_d \]
\[ 475-12 \]
\[ 475-12 \]
\[ 472-19 \]

\[ m_{10} = M_9 M_4 U_2 T_{41a} \]
\[ 459-24 \]
\[ o m_{10} = M_{10} U_2 T_{41a} \]
\[ + T_d \]
\[ 459-10 \]
\[ 459-19 \]
A REGISTER

\[ A_1 \times a_1 = A_2 \times I_3 \times E_{20} \times E_{30} \times E_{40} \times T_{41} \]

\[ + A_2 \times I_3 \times M_2 \times M_{10} \times T_{41} \]

\[ + A_2 \times M_2 \times E_{30} \times U_{41} \]

\[ + A_3 \times D_1 \times U_{41} \]

\[ + R_2 \times D_1 \times U_{41} \times T_{41} \]

\[ + A_2 \times N_0 \times P_0 \times E_1 \times T_1 \]

\[ + R_2 \times M_{10} \]

\[ + F_1 \times E_{30} \]

502-9

503-25

502-25

402-27

402-27

502-25

402-12

402-12

0^a_1 (Next Page)
A REGISTER (Cont)

\[ a_1 = A_2 I_3 E_{30} \]
\[ + A_2 I_3 M_2 M_{10} \]
\[ + A_2 M_2 E_{30} U_{41} \]
\[ + A_3 D_1 U_{41} \]
\[ + R_2 D_1 U_{41} \]
\[ + N_0 P_0 T_{41} E_{2} \]
\[ + F_1 E_{30} \]
\[ + I_2 T_{41} \]
\[ + R_2 M_{10} \]
\[ + U_{41} T_{41} \]
\[ + M_{10} T_{41} \]

CIRCUIT BOARD LOCATION

508-T-6
A REGISTER (Cont)

\[ A_2 \cdot a_2 = A_3 I_3' E_{20}' E_{40}' \]
\[ + A_3 I_3' M_2' \]
\[ + A_3 M_2' T_{41}' \]
\[ + R_{42} M_2 D_2 U_{41}' T_{41}' \]
\[ + A_3 U_{41} \]
\[ + R_{1} E_{20} I_4 \]
\[ + B_1 E_{40} T_{41}' \]
\[ + A_3 E_{40} T_{41} \]
\[ a_2 = A_3 I_3' E_{20}' E_{40}' \]
\[ + A_3 I_3' M_2' \]
\[ + A_3 M_2' T_{41}' \]
\[ + A_3 U_{41} \]
\[ + R_{1} E_{20} I_4 \]
\[ + B_1 E_{40} T_{41}' \]
\[ + A_3 E_{40} T_{41} \]
A REGISTER (Cont)

\[ A_3 \quad 1a_3 = A_r \quad 529 \]
\[ 0a_3 = A_r' \quad 529 \]

\[ A_r \]
\[ A_r' \quad \text{A CHANNEL READ POWER AMPLIFIER OUTPUTS} \]

\[ A_w \quad 1a_w = A_{39} \]
\[ 0a_w = A_{39}' \quad \text{A CHANNEL WRITE AMPLIFIER INPUTS} \]

\[ A_{39} \quad 1a_{39} = A_{40} M_1 \quad 226-34 \]
\[ + A_{40} U_1 \quad 226-34 \]
\[ + A_{40} T_1 \quad 226-18 \]
\[ + S \ M_1 U_1' T_1' \quad 129-9 \]
\[ 0a_{39} = A_40 M_1' \quad 225-34 \]
\[ + A_{40}' U_1 \quad 225-34 \]
\[ + A_{40}' T_1 \quad 225-18 \]
\[ + S' M_1 U_1' T_1' \quad 225-24 \]
A REGISTER (Cont)

\[ a_{40} = A_{41} M_0' M_1' U_{41} \]

\[ + A_{41} M_0 U_1 \]

\[ + S M_0 U_2 \]

\[ + A_{41} M_1 U_1 \]

\[ + M_1 D_2 U_1 T_{41} \]

\[ + K_a M_1 U_1' \]

\[ + A_{41} D_1' U_{41} \]

\[ + S D_1 U_{41} \]

\[ 0 a_{40} = A_{41}' M_0' M_1' U_{41} \]

\[ + A_{41}' M_0 U_1 \]

\[ + S' M_0 U_2 \]

\[ + A_{41}' M_1 T_{41}' \]

\[ + K_a' M_1 U_1' \]

\[ + A_{41}' D_1' U_{41} \]

\[ + S' D_1 U_{41} \]

\[ + A_{41}' M_1 D_3 U_1' \]

\[ 419-24 \]

\[ 419-12 \]

\[ 419-12 \]

\[ 420-12 \]

\[ 420-24 \]

\[ 420-12 \]

\[ 421-12 \]

\[ 421-12 \]

\[ 220-27 \]

\[ 220-27 \]

\[ 220-10 \]

\[ 220-12 \]

\[ 220-12 \]

\[ 221-12 \]

\[ 221-12 \]

\[ 123-18 \]
A REGISTER (Cont)

\[ A_{41} = A_1 I_3 P_0 \]

\[ + B_1 M_0 D_1 \]

\[ + S M_0 J_0 U_2 T_1 \]

\[ + S' M_0 J_0 U_2 T_1 \]

\[ + R_{43} M_1 U_2 T_1 \]

\[ + S M_2 T_x U_{41} T_1 \]

\[ + R_{41} U_{41} T_1 \]

\[ + R_{42} D_1 U_{41} \]

\[ + A_1 B_1 M_3 D_3 D_1 \]

\[ + B_1 M_3 D_2 D_1 \]

\[ + R_1 M_3 D_3 D_1 \]

\[ + A_1 M_3 D_2 D_1 \]

\[ + A_1 M_4 D_2 U_1 \]

\[ + A_1 M_4 U_1 T_{41a} \]

\[ 279-27 \]

\[ 277-27 \]

\[ 174-25 \]

\[ 174-25 \]

\[ 278-27 \]

\[ 177-25 \]

\[ 278-27 \]

\[ 279-12 \]

\[ 177-25 \]

\[ 277-27 \]

\[ 279-24 \]

\[ 279-27 \]

\[ 278-24 \]

\[ 276-27 \]

1\(^{a}{41}\) (Cont Next Page)
A REGISTER (Cont)

1^a 41 (Cont)

+ A_2 \cdot M_4 \cdot D_1 \cdot U_1 \cdot T_1 \cdot T_{40} \cdot T_{41a}
+ A_42 \cdot M_4 \cdot D_1 \cdot U_1 \cdot T_{41a}
+ A_1 \cdot M_4 \cdot D_3 \cdot D_2 \cdot D_1
+ R_1 \cdot M_4 \cdot D_3 \cdot D_2 \cdot D_1
+ U_41 \cdot T_x
+ M_1 \cdot D_3 \cdot U_1 \cdot X_1

+ F_1 \cdot N_0 \cdot P_0 \cdot T_1
+ R_{42} \cdot E_40 \cdot T_{41a}
+ A_{42} \cdot E_40 \cdot T_{41a}
+ A_{42} \cdot M_2 \cdot T_x \cdot U_1 \cdot U_41
+ A_1 \cdot M_4 \cdot D_3 \cdot U_1
+ A_{42} \cdot M_{10}

0^a 41 (Next Page)
A REGISTER (Cont)

\[ 0a_{41} = A_1 I_3 P_0 \]
\[ + B_1 M_0 D_1 \]
\[ + M_2 E_20 U_{41} T_1 \]
\[ + S' M_2 T_x U_{41} \]
\[ + R_{41} U_{41} T_1 \]
\[ + R_{42} D_1 U_{41} \]
\[ + B_1 M_3 D_1 D_3 \]
\[ + R_1 M_3 D_3 D_1 \]
\[ + A_1 M_3 D_2 \]
\[ + A_1 M_4 D_2 U_1 \]
\[ + A_1 M_4 U_1 T_{41a} \]
\[ + M_4 D_1 T_1 \]
\[ + U_1 M_4 D_1 T_{40} \]
\[ + A_2 M_4 D_1 U_1 T_{41a} \]
\[ + A_4 M_4 D_1 U_1 T_{41a} \]

CIRCUIT BOARD
LOCATION

0a_{41} (Cont Next Page)
A REGISTER (Cont)

0^a41 (Cont)

\[ + A_1 M_4 D_3 D_2 D_1 \]
\[ + R_1 M_4 D_3 D_2 D_1 \]
\[ + F_1 N_0 P_0 T_1' \]
\[ + R_{42} E_{40} T_{41a} \]
\[ + A_{42} E_{40} T_{41a} \]
\[ + M_1 U_1 X_1' \]
\[ + A_1 M_4 D_3 U_1 \]
\[ + M_0 U_1 T_1 \]
\[ + M_1 D_3 T_1 \]
\[ + A_{42} M_{10} \]
\[ + A_1 M_0 M_1 M_2 M_3 M_4 M_{10} P_0' \]
\[ + A_{42} M_2 T_x U_{41} \]
\[ + B_1 M_3 D_2 D_1 \]
\[ + N_0 P_0 E_2' E_3' \]

CIRCUIT BOARD LOCATION

579-25

579-25

479-24

477-12

476-27

471-12

478-24

478-27

477-12

479-34

579-18

477-24

476-24

475-24

408

508-T-6
A REGISTER (Cont)

\[
A_{42} = A_i M_4 D_2' T_{41a}' + F_3' F_1 N_0 P_0 E_2' E_1 + Z_1 I_2 I_4' N_q + A_2 M_{10} + S M_2 + F_{c2'} D_4 I_4
\]

\[
o A_{42} = A_i M_4 D_2' + M_4 T_{41a} + F_3' F_1' N_0 P_0 E_2' E_1 + H_0' I_2 T_1 + A_2' M_{10} + S' M_2
\]

GIRCUIT BOARD LOCATION

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>158-25</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>158-18</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>158-25</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>258-12</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>258-12</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>258-10</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>458-12</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>458-34</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>558-18</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>458-12</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>458-34</td>
<td>A_{42} expression</td>
</tr>
<tr>
<td>458-18</td>
<td>A_{42} expression</td>
</tr>
</tbody>
</table>
B REGISTER

\[ b_1 = B_2 M_{10} N_T E_{30} \]
\[ + E_{20} \]
\[ + X_2 M_{10} \]
\[ + A_2 N_T \]

\[ b_1 = B_2 M_{10} N_T E_{20} E_{30} \]
\[ + E_{30} I_4 T_{41a} \]
\[ + X_2 M_{10} \]
\[ + A_2 N_T \]

\[ b_2 = B_2 \]
\[ b_2 = B_2' \]

\[ b_2 = B_2' \]

B CHANNEL READ POWER AMPLIFIER OUTPUTS

CIRCUIT BOARD LOCATION

178-9
278-19
278-34
278-34
177-9
277-12
277-12
278-18
530
530
B REGISTER (Cont)

\[ B_w \quad 1 b_w = B_{40} \quad \{ \quad B \text{ CHANNEL WRITE AMPLIFIER INPUTS} \quad \}
\]

\[ 0 b_w = B_{40} \]

\[ B_{40} \quad 1 b_{40} = B_{41} M_0' M_3' \]

\[ + S M_0 D_1 \]

\[ + B_{41} D_3' D_2' D_1' \]

\[ + B_{41} M_0 D_2 D_1' \]

\[ + B_{41} M_3 D_2 \]

\[ + B_{41} M_3 D_3 D_2' \]

\[ 224 - 27 \]

\[ 225 - 27 \]

\[ 224 - 27 \]

\[ 224 - 27 \]

\[ 226 - 27 \]

\[ 226 - 27 \]

\[ 227 - 27 \]

\[ + S' M_0 D_1 \]

\[ + B_{41} D_3' D_2' D_1' \]

\[ + B_{41} M_0 D_2 D_1' \]

\[ + B_{41} M_3 D_2 \]

\[ + B_{41} M_3 D_3 D_2' \]

\[ 227 - 27 \]

\[ 224 - 24 \]

\[ 224 - 27 \]

\[ 125 - 18 \]

\[ 228 - 27 \]

\[ 228 - 27 \]
B REGISTER (Cont)

\[ b_{41} = M_2 D_0 I_2 \]
\[ + K_0 D_{10} I_2 \]
\[ + B_1 T_x M_0 M_8 M_{10} I_3 \]
\[ + T_x B_1 R_{cl} I_4 N_0 P_0 \]
\[ + N_0 P_0 E_3 E_2 T_2 \]
\[ + B_{42} N_0 P_0 E_2 \]
\[ + S E_{40} \]
\[ + E_{40} T_{41a} \]
\[ + A_1 A_d I_2 R_{cl} \]
\[ + X_1 X_d I_2 R_{cl} \]
\[ + R_1 R_d I_2 R_{cl} \]
\[ + Z_1 Z_d I_2 R_{cl} \]
\[ + F_1 H_4 T_x \]
\[ + F_1 R_{c2} T_x \]
B REGISTER (Cont)

\[ 0_{41}^b = M_n \cdot D_0 \cdot I_2 \]
\[ + K_0 \cdot D_{10} \cdot I_2 \]
\[ + B_1 \cdot M_0 \cdot M_4 \cdot M_{10} \cdot I_3 \cdot T_x \]
\[ + B_1 \cdot R_{cl} \cdot I_4 \cdot P_0 \cdot T_x \]
\[ + N_0 \cdot P_0 \cdot E_3 \cdot E'_2 \cdot P'_1 \]
\[ + N_0 \cdot P_0 \cdot E_2 \]
\[ + S \cdot E_{40} \cdot T_{41a} \]
\[ + X_1 \cdot X_d \cdot I_2 \cdot R_{cl} \]
\[ + A_1 \cdot A_d \cdot I_2 \cdot R_{cl} \]
B REGISTER (Cont)

$0^{b}41$ (Cont)

$+ R_{d}^{'} R_{d} I_{2} R_{c1}$

$+ Z_{d}^{'} Z_{d} I_{2} R_{c1}$

$+ F_{1}^{'} H_{4}^{'} T_{x}$

$+ F_{1}^{'} R_{c2}^{'} T_{x}$

$+ B_{42}^{'} A_{42}^{'} R_{c2}^{'} H_{4}^{'} T_{x}$

$+ B_{1}^{'} M_{8}^{'} Q_{4}^{'} M_{0}^{'}$  

$+ B_{2}^{'} M_{8}^{'} Q_{4}^{'} T_{41a}^{'}$  

$+ B_{1}^{'} M_{8}^{'} Q_{4}^{'} T_{41a}$

$+ M_{8}^{'} Q_{4}^{'} T_{1}$

$+ M_{8}^{'} Q_{4}^{'} T_{40}$

$+ B_{42}^{'} M_{10}$
B REGISTER (Cont)

\[ b_{42} = A_{40} M_2 M_q' U_1 T_2 \]

\[ + R_1 H_4 R_{c2} N_{12} \]

\[ + Z_1 G_0 I_2 P_6' P_5' P_4' P_3' P_2' \]

\[ + Z_4 G_0' T_1 I_2 H_0' \]

\[ + B_2 N_0 P_0 E_2 T_{41}' \]

\[ + I_4 R_{c1} \]

\[ + B_2 M_{10} T_{41}' \]

\[ + M_1 M_q Q_6 Q_3 Q_2 Q_1 T_{41} \]

\[ + M_q Q_6 Q_4 Q_1 T_{41} \]

\[ + M_1 D_3 U_1 \]

[109-25]

[208-24]

[110-18]

[109-25]

[110-25]

[210-34]

[110-25]

[109-18]

[110-9]

[210-24]

0\textsuperscript{b}42 (Next Page)
B REGISTER (Cont)

\[ o_{bn2} = I_4 R_1 R_{c2} + B_2 N_0 P_0 E_2 + B_2 M_{10} + M_9 B_{42} T_{41} + N_1 I_3 T_{41} + N_{12} + M_7 U_1 \]

CIRCUIT BOARD LOCATION

- 510-25
- 510-25
- 412-18
- 410-12
- 410-12
- 412-19
- 405-12
R REGISTER

\[ R_1 \cdot r_1 = R_2 I_3' P_0' + M_0 T_1 + R_2 M_1 U_1 + R_2 M_1 T_{41a} + M_2 U_1 + R_2 M_0' M_1' M_2' M_{10} I_3 + R_2 N_0 P_0 E_2 + E_{40} T_1 + M_{10} T_1 + M_2 T_x U_{41} T_{41a} + R_{42} M_2 U_{41} T_x T_1' \]

CIRCUIT BOARD LOCATION
R REGISTER (Cont)

\[ o_{r_1} = R_2 I_3 P_0 \]
\[ + M_0 T_{40} \]
\[ + M_1 R_2 T_{41a} \]
\[ + D_1 U_{41} \]
\[ + R_2^t M_1 U_1 \]
\[ + E_0 U_{41} \]
\[ + U_{41} T_{41a} \]
\[ + R_{42} M_2 T_1 T_1 T_{41a} \]
\[ + R_2^t M_0 M_1 M_2 M_{10} I_3 \]
\[ + R_2^t N_0 P_0 E_3 \]
\[ + E_{40} T_{41a} \]
\[ + M_{10} T_{40} \]
\[ + E_0^t M_2 T_1 T_1 \]

\[ R_2 \]
\[ 1_{r_2} = R_r \]
\[ 0_{r_2} = R_r^t \]

CIRCUIT BOARD LOCATION

418 508-T-6
R REGISTER (Cont)

\[ R \text{CH~NNE'L} R \text{R} \]

<table>
<thead>
<tr>
<th>Circuit Board Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>R CHANNEL READ POWER AMPLIFIER OUTPUTS</td>
</tr>
<tr>
<td>R CHANNEL WRITE AMPLIFIER INPUTS</td>
</tr>
</tbody>
</table>

\[ R_w \]

\[ R_w \]

\[ R_{40} \]

1) \[ R_{40} = R_{41} E_{30} U_{41}^{'} + S U_{41} T_{x}^{'} M_{q}^{'} + S E_{30} I_{4} + R_{41} T_{x} + R_{41} M_{q} \]

2) \[ R_{40} = R_{41} E_{30} U_{41}^{'} + S U_{41} T_{x}^{'} M_{q}^{'} + S E_{30} I_{4} + R_{41} T_{x} + R_{41} M_{q} \]
\[ R_{41} = R_1 I_3 P_0 \]

\[ + R_{42} M_1 Q_3 \]
\[ + A_2 M_1 D_3 \]
\[ + R_2 M_1 B_{42} U_1 T_{40} T_{41a} \]
\[ + A_3 M_1 B_{42} T_{40} \]
\[ + A_{41} M_1 B_{42} T_{41a} \]
\[ + R_{42} M_2 U_{41} \]
\[ + B_{42} T_X U_{41} M_9 \]
\[ + R_1 M_3 D_3 \]
\[ + M_4 L_{P1} D_6 D_4 I_3 I_4 D_1 U_1 \]
\[ + K_0 L_{P1} D_6 D_4 I_3 I_4 D_1 U_1 \]
\[ + A_1 M_3 D_3 D_1 \]
\[ + R_1 M_3 D_2 U_1 \]
\[ + R_1 M_4 D_3 D_2 \]
\[ + R_1 M_4 D_3 D_1 \]

\[ (\text{Cont Next Page}) \]
R REGISTER (Cont)

1\textsuperscript{r}41 (Cont)

+ A_1 M_4 D_3 D_2 D_1 ~ 169-9
+ R_1 K_a M_4 D_3 T_41a ~ 166-25
+ R'_1 K_a M_4 D_3 T_41a ~ 166-25
+ E_0 M_4 D_3 T_41a ~ 265-24

+ S E_{20} I_4 ~ 266-10
+ R_{42} E_{40} ~ 266-18
( + R_1 M_3 I_4 ) ~ 264-27
+ X_1 M_0 Q_3 ~ 263-27

+ M_9 M_{10} B_{42} X_1 M'_4 ~ 165-25
+ R_1 M'_0 M'_1 M'_2 M'_3 M'_4 M'_9 I_3 ~ 164-18
+ X_2 T_X U_{41} T_{40} T_{41a} ~ 165-25
+ X_1 T_X U_{41} T_{41a} ~ 263-27
+ R_1 M_1 D_3 U_1 ~ 263-24

0\textsuperscript{r}41 (Next Page)
R REGISTER (Cont)

\[ R_{41}^{r} = R_{1}^{r} I_{3}^{r} P_{0}^{r} + R_{42}^{r} M_{0} Q_{3}^{r} + A_{2}^{r} M_{1} D_{3}^{r} + R_{2}^{r} M_{1} B_{42}^{r} U_{1}^{r} T_{40}^{r} T_{41a}^{r} + A_{3}^{r} M_{1} B_{42}^{r} T_{40} \]
\[ + M_{10} + A_{41}^{r} M_{1} B_{42}^{r} T_{41a} + R_{42}^{r} M_{2} U_{41}^{r} + B_{42}^{r} T_{x}^{r} U_{41}^{r} M_{q}^{r} + R_{1}^{r} M_{3} D_{3}^{r} + M_{p}^{r} L_{p1}^{r} D_{6}^{r} D_{4}^{r} I_{3}^{r} I_{4}^{r} D_{1}^{r} U_{1}^{r} + K_{p}^{r} L_{p1}^{r} D_{6}^{r} D_{4}^{r} I_{3}^{r} I_{4}^{r} D_{1}^{r} U_{1}^{r} + A_{1}^{r} M_{3} D_{3}^{r} D_{1}^{r} + R_{1}^{r} M_{3} D_{2}^{r} U_{1}^{r} + R_{1}^{r} M_{4} D_{3}^{r} D_{2}^{r} \]

0\[r_{41}\] (Cont Next Page)
R REGISTER (Cont)

0r41 (Cont)

+ $X_1' M_0 Q_3$

+ $R_1' M_4 D_3' D_1'$

+ $A_1' M_4 D_3' D_2 D_1$

+ $R_1' K_a M_4 D_3 T_{41a}$

+ $R_1' K_0 M_4 D_3 T_{41a}^*$

+ $E_0' M_4 D_3 T_{41a}$

+ $N_0 P_0 E_3' E_2'$

+ $S' E_{20} I_4$

+ $R_{42} E_{40}$

(+ $R_1' M_3 I_4$)

+ $M_9 B_{42} X_1' M_4'$

+ $R_1' M_0 M_1' M_2 M_3' M_4 M_9' I_3$

+ $X_2' T_X U_{41} T_{41a}$

+ $X_1' T_X U_{41} T_{41a}$

+ $R_1' M_1 D_3 U_1$

CIRCUIT BOARD
LOCATION

463-27
564-25
564-25
565-25
565-25
466-24
465-24
465-12
465-18
464-27
464-24
567-18
463-24
463-27
967-24

508-T-6

423
R REGISTER (Cont)

\[ R_{42} = R_2 M_0 + R_{43} M_2 + R_2 E_{40} \]
\[ 0 R_{42} = R_2 M_0 + R_{43} M_2 + R_2 E_{40} \]

\[ R_{43} = R_2 T_x T_{40} T_{41a}^{'(M4)} \]
\[ + A_1 B_1 I_3 M_4 T_{41a}^{'(M4)} T_x U_1 \]
\[ + A_1 B_1 I_3 M_4 T_{41a}^{'(M4)} U_1 \]
\[ + E_0 K_a U_1 M_4 T_{41a}^{'(M4)} \]
\[ + A_1 K_a U_1 M_4 T_{41a}^{'(M4)} \]
\[ + R_2 U_1 T_{41a}^{'(M4)} \]
\[ 0 R_{43} = R_2 T_{41a}^{'(M4)} \]
\[ + M_1 E_0 K_a T_{41a}^{'(M4)} \]
\[ + A_1 K_a U_1 T_{41a}^{'(M4)} \]
\[ + T_{40} \]
**X REGISTER**

\[ X_1 \quad 1 \cdot X_1 = X_2 \]
\[ 0 \cdot X_1 = X_2' \]

\[ X_2 \quad 1 \cdot X_2 = X_\lambda \]
\[ 0 \cdot X_2 = X_\lambda' \]

\[ X_\lambda \]
\[ X_\lambda' \]

\[ X_{40} \quad 1 \cdot X_{40} = X_{41} \quad M_{10}' \quad 228-34 \]
\[ + S \quad M_{10} \quad 228-34 \]
\[ 0 \cdot X_{40} = X_{41}' \quad M_{10}' \quad 227-34 \]
\[ + S' \quad M_{10} \quad 227-34 \]
X REGISTER (Cont)

\[ X_{41} = M_{X2} L_{P1} M_{4} U_{1} \]
\[ + K_{0} L_{P1} M_{4} U_{1} \]
\[ + X_{1} M_{B} C_{4} Q_{4} T_{41} \]
\[ + X_{1} M_{B} C_{4} Q_{4} T_{41} \]
\[ + M_{B} Q_{2} Q_{4} T_{41} \]
\[ + X_{1} Q_{4} U_{1} M_{10} \]
\[ + X_{1} M_{q} M_{10} M_{4} \]
\[ + S M_{10} U_{2} T_{1} \]
\[ + A_{1} I_{1} I_{5} \]
\[ + X_{1} M_{q} M_{q} I_{i} U_{1} \]
\[ + M_{4} U_{1} Q_{2} \]
\[ + G_{1} M_{5} U_{1} N_{4} \]
\[ + G_{0} M_{5} T_{2} \]
\[ + X_{1} D_{5} I_{5} \]
X REGISTER (Cont)

1*41 (Cont)

+ A 1 M 1 D 3 U 1
+ X 1 T x
+ M 3 T 41
+ X 1 I 3 I 5

0*41 = M 2 L 51 M 4 U 1 Q 2
+ K 0 L 51 M 4 U 1 Q 2
+ X 1 M 8 C 4 Q 4 T 41
+ X 1 M 8 C 4 Q 4 T 41
+ M 8 Q 2 Q 4 T 41
+ X 1 Q 4 U 1 M 10
+ X 1 M 9 M 10 M 4
+ M 10 U 1
+ A 1 I 1 I 5
+ X 1 M 8 M 9 I 5 U 1

0*41 (Cont Next Page)
X REGISTER (Cont)

\[ \begin{align*}
0^{\times}41 \text{ (Cont) } & + G_1' N_4' U_1 M_5 & 221-27 \\
& + M_5' G_0' T_2 & 221-27 \\
& + X_1' D_5' I_5' & 223-10 \\
& + A_1' M_1 D_3' U_1 & 223-27 \\
& + X_1' T_x & 223-18 \\
& + M_5 U_1 N_4' P_2 & 223-24 \\
& + X_1' I_3' I_5' & 123-9
\end{align*} \]

L MEMORY LOOP

\[ \begin{align*}
L_1 & \quad 1l_1 = L_2 & 215-19 \\
& \quad 0l_1 = L_2' & 216-19 \\
L_2 & \quad 1l_2 = L_{r8} & 535 \\
& \quad 0l_2 = L_{r8}' & 535 \\
L_{2a} & \quad 1l_{2a} = L_{r4} & \quad \\
& \quad 0l_{2a} = L_{r4}' & \quad 
\end{align*} \]
L MEMORY LOOP (Cont)

L Channel Read Power Amplifier Outputs From 8-word Read Head
\[ L_{r8} \]
\[ L_{r8'} \]

L Channel Read Power Amplifier Outputs From 4-word Read Head
\[ L_{r4} \]
\[ L_{r4'} \]

L Channel Write Amplifier Inputs
\[ L_w \]
\[ 1^w = L_{41} \]
\[ 0^w = L_{41'} \]

\[ L_{41} = M_2 M_6 D_3 D_2 D_i T_i \]
\[ + A_1 M_3 D_3 D_1 L_{p1} L_{p2}' T_i \]
\[ + B_1 M_6 D_i' L_{p2} D_{10} \]
\[ + L_1 M_3' M_6' \]
\[ + L_1 M_3 D_i' \]
\[ + L_1 D_2 \]
\[ + L_1 D_i' L_{p1} \]
\[ + L_1 D_{10} \]
\[ + L_1 D_3' L_{p2} \]

0'41 (Next Page)
L MEMORY LOOP (Cont)

\[ 0 \ell_{41} = M_6 \ \ M_6 \ \ D_3 \ \ D_2 \ \ D_1 \]

\[ + A_1 \ M_3 \ D_3 \ D_1 \ L_{P1} \ \ L_{P2} \]  
\[ + B_1 \ M_6 \ D_3 \ L_{P2} \ \ D_{10} \]  
\[ + L_1 \ M_3 \ M_6 \]  
\[ + L_1 \ M_3 \ D_1 \]  
\[ + L_1 \ D_2 \]  
\[ + L_1 \ D_1 \ L_{P1} \]  
\[ + L_1 \ D_1 \ L_{P2} \]  
\[ + L_1 \ D_3 \ D_{10} \]  
\[ + L_1 \ D_3 \ L_{P2} \]  
\[ + T_1 \]  

CIRCUIT BOARD LOCATION

124-25
124-18
124-25
125-25
125-25
224-34
225-12
224-12
225-12
224-12
224-34
L MEMORY LOOP (Cont)

\[ L_{p1} \quad l_{p1} = D_0 \cdot D_{II} \cdot I_3 \cdot R_{c1} \cdot T_1 + (V_d + L_d) I_2 \cdot R_{c1} \]

\[ 0 \cdot l_{p1} = G_1 \cdot I_1 \cdot N_5 + Z_1 \cdot I_1 \cdot N_5 + M_d \cdot I_2 \cdot R_{c1} \]

\[ L_{p2} \quad l_{p2} = G_1 \cdot N_1 \cdot P_5 \cdot P_2' \cdot I_1 + Z_1 \cdot N_1 \cdot P_5 \cdot P_2' \cdot I_1 \cdot I_4 + V_d \cdot I_4 \]

\[ 0 \cdot l_{p2} = G_1 \cdot N_1 \cdot P_5 \cdot I_1 + Z_1 \cdot N_1 \cdot P_5 \cdot I_1 \cdot I_4 + L_d \cdot I_4 \]
V MEMORY LOOP

\[ V_1 \quad 1 V_1 = V_2 \]
\[ 0 V_1 = V_2' \]

\[ V_2 \quad 1 V_2 = V_{R8} \]
\[ 0 V_2 = V_{R8}' \]

\[ V_{R8} \]
\[ V_{R8}' \]

\[ V_{W} \quad 1 V_{W} = V_{41} \]
\[ 0 V_{W} = V_{41}' \]
V MEMORY LOOP (Cont)

\[ V_{41} = M_\lambda M_6 D_2 D_1 T_1 \]
\[ + A_1 M_3 D_3 D_1 L_{P1} L_{P2} T_1' \]
\[ + B_1 M_6 D_3' L_{P2} D_{10} \]
\[ + V_1 M_3' M_6' \]
\[ + V_1 M_3' D_1' \]
\[ + V_1 M_6' D_3' D_2' \]
\[ + V_1 D_1' L_{P1}' \]
\[ + V_1 D_1' L_{P2}' \]
\[ + V_1 D_3' D_{10}' \]
\[ + V_1 D_3' L_{P2}' \]

129-25
131-18
129-25
229-27
229-27
128-25
229-12
229-12
128-25
231-10

0^V_{41} (Next Page)
V MEMORY LOOP (Cont)

\[ 0V_{41} = M_2 \cdot M_6 \cdot D_2 \cdot D_1 \]

\[ + A_1 \cdot M_3 \cdot D_3 \cdot L_{P1} \cdot L_{P2} \cdot D_1 \]

\[ + B_1 \cdot M_6 \cdot D_3 \cdot L_{P2} \cdot D_{10} \]

\[ + V_1 \cdot M_3 \cdot M_6 \]

\[ + V_1 \cdot M_3 \cdot D_1 \]

\[ + V_1 \cdot M_6 \cdot D_3 \cdot D_2 \]

\[ + V_1 \cdot D_1 \cdot L_{P1} \]

\[ + V_1 \cdot D_1 \cdot L_{P2} \]

\[ + V_1 \cdot D_3 \cdot D_{10} \]

\[ + V_1 \cdot D_3 \cdot L_{P2} \]

\[ + T_1 \]

CIRCUIT BOARD LOCATION

130-25

130-9

131-9

130-25

131-25

131-25

230-27

230-27

231-27

231-27

231-19
V MEMORY LOOP (Cont)

\[ K_0 = L_{2a} L_{P2} S_7 S_8 \]
\[ + L_2 L_{P2} S_8 \]
\[ + V_{2a} L_{P2} S_7 S_8 \]
\[ + V_2 L_{P2} S_8 \]
\[ 0 K_0 = L_{2a} L_{P2} S_7 S_8 \]
\[ + L_2 L_{P2} S_8 \]
\[ + V_{2a} L_{P2} S_7 S_8 \]
\[ + V_2 L_{P2} S_8 \]

163-25
263-12
163-25
263-12
563-25
463-12
563-25
463-12
STATE FLIP-FLOPS

\[ I_1, i_1 = Z_{CH}^1 G_0 N_{12} \]
\[ + Z_{CH}^1 M_5 N_{12} \]
\[ + N_0 N_{12} \]
\[ + F_4 F_3 F_2 F_1 N_0 A_0 P_{d3} T_4 \]
\[ + F_5 F_4 F_3 F_2 C_h P_{d3} N_0 A_1 \]
\[ + E_3 E_2 E_1 A_n A_t P_0 T_{41} \]
\[ + F_5' F_4' F_2 F_0 I_{4} P_{d3} \]
\[ + F_5' F_4' F_2 I_{4} G_0 A_n P_{d3} \]

\[ o i_1 = I_1 D_0 T_{41a} \]
\[ + I_1 D_{10} T_{41a} \]
\[ + D_{11} T_{40} \]
\[ + I_1 F_0 T_{41a} \]
\[ + T_d \]
STATE FLIP-FLOPS (Cont)

\[ I_2 = Z_{CH} G_0 M_5 N_0 N_{12} \]
\[ + I_1 D_0 T_{41a} \]
\[ + I_1 I_4 D_{10} T_{41a} \]
\[ + R_{c1} R_b I_2 T_{41a} \]
\[ + F_5 F_4 F_2 I_4 F_0 G_0 A_n P_{d3} \]
\[ o\cdot i_2 = I_2 N_{11} \]
\[ + T_d \]
\[ + I_2 H_o T_{41} \]

\[ I_3 = J_o I_2 N_{11} \]
\[ + I_2 I_4 N_{11} \]
\[ + F_5 F_4 F_3 F_2 N_0 A_0 P_{d3} T_{41} \]
\[ o\cdot i_3 = N_{12} \]
\[ + M_7 D_1 U_2 T_{41} \]
\[ + M_2 H_o I_3 I_4 T_{41} \]
\[ + T_d \]
\[ + J_{13} \]
STATE FLIP-FLOPS (Cont)

\[ I_4 = M_7 D_1 U_2 T_{41} + Z_{CH} N_{12} + J_0 I_2 N_{11} + T_d + J_{13} \]

\[ 0 I_4 = F_5' F_4' F_2 A_n' P_{d3} + E_3 E_2 E_1' A_n A_t' P_0 T_{41} \]
STATE CHANGE FLIP-FLOPS

\[ n_{11} = B_{42} T_{40} I_{4}' + D_0 T_{40} + D_{10} T_{40} + S_0' D_6 D_5 D_1 ' T_{40} + S_8' L_{pl} D_4 D_3 D_1 T_{40} + S_{11} ' D_4 D_3 D_1 T_{40} + D_6 D_3 ' D_2 ' T_{40} + D_6 D_5 ' D_1 T_{40} + D_6 D_4 T_{40} + J_0 T_{40} I_{4}' + M_d ' R_{cl} T_{40} + S_{11} ' D_6 D_1 T_{40} + D_2 ' D_4 D_3 T_{40} D_1 \]

State Change Flip-Flops

\[ n_{11} = T_{410} \]
STATE CHANGE FLIP-FLOPS (Cont)

\[ N_{12} = M_0 D_1 T_{40} \]

\[ + M_0 M_8 U_2 T_{40} \]

\[ + M_1 M_9 Q_6 Q_4 T_{40} \]

\[ + U_{41} M_q T_{40} \]

\[ + M_3 D_1 D_3 T_{40} \]

\[ + T_x R_{c2} H_4 T_{40} D_1 \]

\[ + M_3 D_1 U_2 T_{40} \]

\[ + T_x F_{c2} F_{c1} H_4 H_3 H_2 T_{40} \]

\[ + F_4 B_2 B_1 T_x T_{40} \]

\[ + Q_0' M_4 D_3' D_2 T_{40} \]

\[ + M_4 D_2 D_1 T_{40} D_3' \]

\[ + A_1 M_8 M_9 M_4 D_3 U_1 T_{40} \]

\[ + A_{42} M_4 M_8 M_9 D_3 U_1 T_{40} \]

\[ + (H_0 M_4 T_{40}) \]

\[ + M_5 U_2 T_{40} \]

\[ n_{12} (Cont \ Next \ Page) \]
STATE CHANGE FLIP-FLOPS (Cont)

\[ \text{On}_{12} = T_{410} \]

<table>
<thead>
<tr>
<th>Expression</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ M_6 \ D_3 \ Q_4 \ T_{40}</td>
<td>474-27</td>
</tr>
<tr>
<td>+ T_{40} \ S_{5e} \ M_5 \ D_2 \ D_1 \ D_3</td>
<td>575-1</td>
</tr>
<tr>
<td>+ M_6 \ D_3^' \ D_{10} \ T_{40}</td>
<td>274-27</td>
</tr>
<tr>
<td>+ M_6 \ D_3^' \ D_{11} \ T_{40}</td>
<td>574-25</td>
</tr>
<tr>
<td>+ I_4^' \ I_3 \ D_6 \ P_{d3} \ T_{40}</td>
<td>574-25</td>
</tr>
<tr>
<td>+ T_{40} \ S_{5d} \ M_5 \ D_2 \ D_1^' \ D_3</td>
<td>574-2</td>
</tr>
<tr>
<td>+ T_{40} \ S_{5b} \ M_5 \ D_2^' \ D_1^' \ D_3</td>
<td>174-9</td>
</tr>
<tr>
<td>+ M_7 \ D_3^' \ D_2 \ D_1 \ T_{40}</td>
<td>174-25</td>
</tr>
<tr>
<td>+ M_3 \ D_2^' \ D_1 \ T_{40}</td>
<td>174-25</td>
</tr>
</tbody>
</table>

CIRCUIT BOARD LOCATION

508-T-6

441
SUM FLIP-FLOPS

\[ S_{1s} = A_1 B_1 R_1 K_a + A_1 B_1' K'_a + A_1 R_1' K_a + A_1 B_1 R_1 K_a' + A_1 B_1' K_a' + A_1 R_1' K_a' \]

\[ S_2 = A_1 B_1' K_a + A_1 R_1' K_a + A_1 B_1 R_1 K_a + A_1 B_1' K_a + A_1 R_1' K_a + A_1 B_1 R_1 K_a' \]
COMPUTATION CARRY FLIP-FLOPS

\[ K_a = E_0 A_1 B_1 R_1 M_4 T_{41} \]
\[ + E_0' A_1' B_1' R_1' M_4' T_{41}' \]
\[ + R_{43} D_1 U_{41} T_1 \]
\[ + M_4 U_1 T_1 \]

\[ \bar{k}_a = E_0 A_1 B_1 M_4 T_1 \]
\[ + E_0 A_1' R_1' M_4' \]
\[ + E_0 A_1 B_1 M_4 \]
\[ + E_0 A_1 R_1 M_4 \]
\[ + E_0 R_1 M_4 D_1 T_1 \]
\[ + E_0 R_1 M_4 D_1 T_1' \]
\[ + E_0 R_1 M_4 D_1 T_1 \]
\[ + T_{41} \]
\[ + E_0 R_1 M_4 D_1 T_1' \]
\[ + E_0 R_1 M_4 D_1 T_1 \]

CIRCUIT BOARD LOCATION

512-18
512-9
411-27
411-27
111-25
114-25
113-25
114-25
112-25
111-25
212-19
113-25
112-25
ADD-SUBTRACT INDICATOR FLIP-FLOPS

\[ E_0 = K_a' M_0 T_{41a} \]

\[ + A_1' K_a M_2 U_1' T_{41a} \]

\[ + M_2' T_x' U_2' T_1 \]

\[ + Q_0 M_4 M_4' T_{40} \]

\[ + U_{41} \]

\[ + R_{41} M_4 D_3 T_1 \]

\[ + N_5 \]

\[ + Q_0 M_8 M_0' T_{40} \]

\[ + M_9 M_2' U_2' T_{41a} \]

\[ + K_a' M_{10} T_{41a} \]

\[ + M_8 M_0' T_{41a} \]
ADD-SUBTRACT INDICATOR FLIP-FLOPS (Cont)

\[ e_0 = A_{41} B_{41} M_0 D_2^1 U_1 T_1 \]

\[ + A_{41}^1 B_{41} M_0 D_2^1 U_1 T_1 \]

\[ + A_{41} B_{41} M_0 D_2 \]

\[ + A_{41}^1 B_{41} M_0 D_2 \]

\[ + M_2 U_1 \]

\[ + A_1 K_a M_2 T_{41a} \]

\[ + R_{41} M_4 D_3 T_1 \]

\[ + Q_0 M_4 D_3 D_1 T_{40} \]

\[ + X_{41} R_{41} M_{10} D_2 U_1 T_1 \]

\[ + X_{41}^1 R_{41} M_{10} D_2 U_1 T_1 \]

\[ + X_{41} R_{41} M_{10} D_2^1 U_1 T_1 \]

\[ + X_{41}^1 R_{41} M_{10} D_2^1 U_1 T_1 \]

\[ + F_a \]

\[ 155-18 \]

\[ 555-9 \]

\[ 556-9 \]

\[ 155-9 \]

\[ 456-18 \]

\[ 455-24 \]

\[ 556-25 \]

\[ 556-25 \]

\[ 555-18 \]

\[ 156-1 \]

\[ 157-1 \]

\[ 158-1 \]

\[ 257-19 \]
TRANSFER OF CONTROL-VERIFY INPUT FLIP-FLOPS

\[ H_0 = M_4 M_8 M_9 D_3 U_1 T_1 \]
\[ + A_{41} M_5 D_2 U_1 T_1 \]
\[ + A_{41} M_5 U_1 D_1 T_1 \]
\[ + (J_0 M_5 D_2 D_1 T_1) \]
\[ + V_{gm} I_4 \]
\[ + M_7 D_1 T_2 \]
\[ + M_5 D_3 U_1 \]
\[ + V_{gp} I_4 \]

\[ \theta h_0 = A_1 M_4 T_1 \]
\[ + A_1 M_5 D_3 D_2 D_1 T_1 T_4 T_4 \]
\[ + J_0 T_0 I_4 \]
\[ + I_3 I_4 \]
\[ + R_{cl} J_0 \]
\[ + U_2 N_4 \]
\[ + A_n A_t \]
\[ + N_{12} I_4 \]
ERROR FLIP-FLOPS

\[ J_0 \quad j_0 = D_1 E_0 K_a M_0 U_1 T_{41} \]
\[ + K_a' M_2 U_1 T_{41} \]
\[ + K_a D_1 U_{41} T_{41} \]
\[ + X_2 Z_2' H_0 I_2 T_{41}' \]
\[ + X_2' Z_2 H_0 I_2 \]
\[ + R_b T_o \]
\[ + M_2 E_{30} U_1 F_3' \]
\[ + J_{t3} \]
\[ + E_{40} T_{41} S \]
\[ J_0 = T_d \]
\[ + D_6 D_5' D_4' D_3' D_2 D_1 I_2 T_{40} \]
READER DELAY FLIP-FLOPS

\[ D_R = d_R = F_5' F_4 F_3' F_2 F_0' A_0' P_m I_1 \]
\[ + F_5 F_3 F_2' F_1' A_0' P_m I_1 \]
\[ + F_5' F_4' F_3 F_2 F_1' P_m I_1 \]
\[ + M_2 N_{12} \]
\[ + (R_{Sm} + R_{Sp}) T_{pp}^5 \]
\[ + N_{11} J_0 \]
\[ + E_3 E_2 E_1' P_{ma} A_t' P_{d3} \]
\[ o_{dR} = (R_{gm} + T_g) \]
\[ + V_{gm} \]
\[ + N_{12} \]
\[ + R_{gp} \]
\[ + V_{gp} \]
\[ + K_g I_t \]
\[ + T_o' \]

173-18
172-18
171-18
271-34
271-34
214-18
172-4
473-34
473-34
472-34
471-34
471-34
472-34
470-19
OUTPUT ERROR FLIP-FLOPS

\[ J_{13} = F_1 E_{t1} T_t + F_1' K_{1k} T_t + F_2 E_{t2} T_t + F_2' K_{2k} T_t + F_3 E_{t3} T_t + F_3' K_{3k} T_t + F_4 E_{t4} T_t + F_4' K_{4k} T_t + F_5 E_{t5} T_t + F_5' K_{5k} T_t + F_1 E_{p1} T_p + F_1' E_{p1} T_p + F_2 E_{p2} T_p + F_2' E_{p2} T_p + F_3 E_{p3} T_p \]

251-27

255-27

255-27

255-12

254-12

254-12

254-27

254-27

255-12

253-12

253-12

253-27

253-27

252-12
OUTPUT ERROR FLIP-FLOPS (Cont)

$$1^j_{13} \text{ (Cont)}$$

$$+ F_3' E_{P3} T_p$$  \hspace{1cm} 252-12
$$+ F_4' E_{P4} T_p$$  \hspace{1cm} 252-27
$$+ F_4' E_{P4} T_p$$  \hspace{1cm} 252-27
$$+ F_5' E_{P5} T_p$$  \hspace{1cm} 251-12
$$+ F_5' E_{P5} T_p$$  \hspace{1cm} 251-12

$$o_{j13} = T_d$$  \hspace{1cm} 254-19
$$+ S_{u1}$$  \hspace{1cm} 253-19
HALF-WORD INDICATOR FLIP-FLOPS

\[ G_0 = G_0' I_4' N_{12} \]
\[ + Z_1' H_0' I_4' U_2' N_9 \]
\[ + G_0' M_7' I_4 \]
\[ + G_1' F_0' N_4 \]
\[ \alpha_0 = G_0' I_4' N_{12} \]
\[ + Z_1' H_0' I_4' U_2' T_2 \]
\[ + G_0' M_7' I_4 \]
\[ + G_1' F_0' N_4 \]
\[ + I_4' (S_{u1} + S_{u2} + S_{u3}) \]

PS ORDER FLIP-FLOPS

\[ N_7' = A_{42}' I_2' N_{11}' D_6' D_3' \]
\[ + A_{42}' M_4' D_3' P_6' P_5' P_3' P_1' \]
\[ + I_2' N_{11}' D_6' D_5 \]
\[ + I_2' N_{11}' D_4' D_3' D_1 \]
\[ \alpha_n = N_7' T_{41} \]
\[ + P_4' P_3' P_2' M_4' \]

216-27
114-9
216-27
215-10
215-12
115-9
215-12
216-12
216-12
119-9
120-18
218-24
123-25
218-10
216-24

CIRCUIT BOARD LOCATION
READOUT CONTROL

\[ H_1 \cdot h_1 = H_1 \cdot R_4 \cdot N_8 + R_4 \cdot H_1 \cdot H_2 \cdot H_3 \cdot F \cdot T \]
\[ \quad o \cdot h_1 = H_1 \cdot N_8 + U_1 \cdot C_2 \cdot A_0 \]

\[ H_2 \cdot h_2 = H_2 \cdot H_1 \cdot N_8 \]
\[ \quad o \cdot h_2 = H_2 \cdot H_1 \cdot N_3 + U_1 \cdot C_2 \cdot A_0 \]

\[ H_3 \cdot h_3 = H_3 \cdot H_2 \cdot H_1 \cdot N_8 \]
\[ \quad o \cdot h_3 = H_3 \cdot H_2 \cdot H_1 \cdot N_8 + U_1 \cdot C_2 \cdot A_0 \]

\[ H_4 \cdot h_4 = H_4 \cdot H_3 \cdot H_2 \cdot H_1 \cdot N_8 \]
\[ \quad o \cdot h_4 = H_4 \cdot H_3 \cdot H_2 \cdot H_1 \cdot N_8 + U_1 \cdot C_2 \cdot A_0 + N_{12} \]
READOUT CONTROL (Cont)

\[ N_8 \quad n_8 = M_2 D_5 T_x T_{41} N_{12} \]

\[ + (TPP') R_{c3} T_{40} \]

\[ o_n_8 = N_8 \]

READOUT TIMING

\[ R_{c1} \quad r_{c1} = J_0 D_6 D_5 D_4 D_2 D_1 I_2 N_{11} \]

\[ + R_b I_4 \]

\[ o_r_{c1} = R_{c2} \]

\[ + T_d \]

\[ R_{c2} \quad r_{c2} = R_{c1} I_3 \]

\[ + M_7 D_6' U_1 \]

\[ o_r_{c2} = H_4 N_8 \]

\[ + I_3' \]

\[ + A_{42}' H_3 H_2 H_1 N_8 \]

CIRCUIT BOARD LOCATION

508-T-6 453
READOUT TIMING (Cont)

\[ R_{c3} \]
\[ 1 \times R_{c3} = M_3 T_x T_{41} N_{12} \]
\[ + (TPP) P_m T_0' I_4 F_5 \]
\[ 0 \times R_{c3} = (TPP)' R_{c3} T_{40} \]

\[ R_{c4} \]
\[ 1 \times R_{c4} = R_{c1} \]
\[ + T_0' C_h A_0' I_4 P_{d3} F_5 \]
\[ 0 \times R_{c4} = R_{c1}' T_{40} \]

\[ T_x \]
\[ 1 \times T_x = M_3 D_2 D'_1 Q_1 T_{41a} \]
\[ + M_7 D'_6 U_1 T_{41a} \]
\[ + D'_5 P_{d3} T_{41a} \]
\[ + D'_6 D_5 D'_4 D_3 I_2 N_{11} J_0' \]
\[ + I_5 I_3 T_{41a} \]
\[ 0 \times T_x = M_2 T_x T_{41a} \]
\[ + N_{12} \]
READOUT TIMING (Cont)

\[ N_{8a} = M_2 D_5 T_x T_41 N_{12} \]
\[ n_{8a} = N_{8a} \]
\[ S_x = H_3 H_2 H_1 N_8 F_{c2} F_{c1} + I_2 \]
\[ s_x = H_2 H_1 N_8 + M_2 \]

TAPE PUNCH PULSE FLIP-FLOPS

\[ T_{PP} = (T_{PP}) X_{10}' + T_0 Q_0 P_5' P_4 P_3' P_2' P_i \]
\[ + (S_u + T_{PP}) I_4 P_5' P_4 P_3' P_2' P_i + T_{PP} X_4' I_4' \]
\[ t_{PP} = (T_{PP})' (T_{PP}) (T_{PP}) (T_{PP}) [S_{u1} S_{u2} S_{u3} S_c'] (T_{PP})' \]

CIRCUIT BOARD LOCATION

174-9
274-19
111-9
211-19
209-12
209-12
457-12
559-18
556-18
457-12
156-18
DELAY FLIP-FLOPS

\[
P_m = (TPP) J_{13}' A_n' P_5' P_4' P_3' P_2' P_1 \\
0 P_m = P_{d3} \\
+ J_{13} \\
+ C_t \\
\]

\[
P_{d3} = (TPP)' T_0' P_m T_{40} \\
+ C_t \\
+ (TPP)' P_{ma} T_{40} \\
0 P_{d3} = P_{d3} T_{40} \\
\]

\[
X_{20} = X_{20}' X_0 \\
0 X_{20} = X_{20} X_0 \\
\]

\[
A_n = M_7 D_6 D_3' D_1' L_{p1} \\
+ F_5' F_4' F_3' F_2' F_1' C_t \\
+ A_t I_5 \\
0 A_n = M_7' T_0' \\
+ E_3' E_2' E_1' A_t P_0 T_{41} \\
+ T_4 \\
\]

CIRCUIT BOARD LOCATION

\[
564-18 \\
466-19 \\
466-34 \\
466-34 \\
418-24 \\
419-19 \\
415-24 \\
219-18 \\
256-18 \\
255-18 \\
577-9 \\
578-9 \\
474-12 \\
275-18 \\
177-18 \\
279-19 \\
456 \\
508-T-6
DELAY FLIP-FLOPS (Cont)

\[ a_t = (A_n P_m) + F_5' F_4' F_3' F_2' F_1' C_T \]
\[ o_a_t = A_n' P_{d3} T_{40} + T_d \]

\[ P_{ma} = (TPP) A_n P_5' P_4' P_3' P_2' P_1 \]
\[ o_P_{ma} = P_{d3} + F_5' F_4' F_3' F_2' F_1' A_n' Q_0 \]

INPUT BUFFER FILL FLIP-FLOPS

\[ i_5 = F_0' T_1 I_1 P_{d3} F_4 F_3 F_1' A_n' \]
\[ + M_3' I_4 N_{12} \]
\[ + M_3' L_{P1} M_8 D_6 T_2 \]
\[ + K_3' L_{P1} M_8 D_6 T_2 \]
\[ + M_8 D_6 I_5' T_{41a} \]
\[ o_i_5 = I_5 T_{41a} \]
FILL DECIMAL NUMBER FLIP-FLOPS

\[ n_0 = F_5' F_4 F_3 F_2' F_1' C_t + N_b C_b M_3 I_4 \]
\[ + n_0 = L_b + F_5 F_4' F_3' F_2' F_1' T_0 P_m + C_b + F_5' F_4 F_3 F_2' F_1' P_m + I_4' + M_3 + F_5' F_4 F_3' F_2 F_1' P_m \]

CIRCUIT BOARD LOCATION

516-9
416-24
416-34
515-18
416-34
515-9
417-34
417-34
519-9
FILL ORDERS FLIP-FLOPS

\[ 0_0 = F_5' F_4 F_3 F_2 F_1' C_t \]
\[ + C_b C_r' M_3' I_4 \]
\[ + A_t A_n' \]
\[ 0_0 = L_b \]
\[ + F_5 F_4' F_3' F_2 P_m T_0 F_1' \]
\[ + N_n \]
\[ + F_5' F_4 F_3 F_2' P_m \]
\[ + I_4' \]
\[ + M_3 \]
\[ + A_t I_5 \]

\[ 0_1 = N_0' P_o E_3 O_1' F_4' \]
\[ + O_1' N_0' P_o E_2 F_4' \]
\[ + A_n P_o O_1' \]
\[ 0_0 = O_1 T_{41c} \]
FILL ORDER COUNTER FLIP-FLOP

\[ F_0 \quad f_0 = F_5 F_4' F_3' F_2 F_1' C_t + M_7 I_4 A_n' + L_b C_h' M_3' I_4 T_1 + E_6 M_8 N_{12} + M_5 N_{12} Z_{ch} \]

\[ G_0 = F_5' F_4 F_3' P_m + N_b + I_4' N_{12}' + M_3' N_{12}' + (S_{u1} + S_{u2} + S_{u3}) + C_b \]
LEGITIMATE CODE FLIP-FLOP

\[ P_0 \quad iP_0 = F_0 F_5 F_4' P_{d3} \]

\[ + O_0 F_5 F_4' P_{d3} \quad 169-25 \]

\[ + O_0 F_5 F_3' F_2 P_{d3} \quad 170-25 \]

\[ + N_0 F_5 F_4' P_{d3} T_{41a} \quad 169-25 \]

\[ + N_0 F_5 F_3' P_{d3} \quad 269-24 \]

\[ + P_{d3} F_1' F_2' F_5 N_0 \quad 170-25 \]

\[ + A_n P_{d3} P_{ma} \quad 271-12 \]

\[ oP_0 = N_0' P_0 T_{41a} \quad 470-12 \]

\[ + P_0 E_2' T_{41a} \quad 470-12 \]

CIRCUIT BOARD LOCATION
INPUT-OUTPUT REGISTER

\[ F_1 \quad f_1 = K_1 \cdot I_4 \cdot (TPP) \cdot P_5 \]
\[ + F_5 \cdot F_3 \cdot F_2 \cdot C_t \]
\[ + A_1 \cdot N_0 \cdot O_1 \cdot P_0 \cdot T_1 \]
\[ + F_2 \cdot O_1 \]
\[ + F_2 \cdot E_3 \]
\[ + B_1 \cdot S_x \cdot T_x \cdot N_1 \]
\[ + F_2 \cdot S_x \cdot T_x \cdot N_1 \]
\[ + F_2 \cdot N_6 \]
\[ + D_5 \cdot S_x \cdot N_8 \cdot H_1 \]
\[ + F_4 \cdot F_2 \cdot N_8 \cdot D_5 \]
\[ + S_{u1} \cdot I_4 \]
\[ + S_{u3} \cdot I_4 \]

\(0^f1\) (Next Page)
\[ o_f_1 = F'_3 F'_2 C_t + F'_4 F'_3 F'_2 C_t + A'_1 N'_0 O'_1 P_0 T'_1 + F'_2 O_1 + F'_2 E_{30} + B'_1 S'_x T'_x + F'_2 S'_x T'_x + F'_2 N'_6 + Z_{ch} I'_3 + M'_3 N'_8 + D'_5 P'_0 P_{d3} T_{40} + N_{12} + D'_5 P_{d3} T_{41} + F'_3 F'_2 N_{8a} + F'_4 N_{8a} \]
INPUT-OUTPUT REGISTER (Cont)

\[ F_2 \cdot f_2 = K_2 \cdot I_4 \cdot (TPP) \cdot P_5 \]

\[ + F'_5 \cdot F'_2 \cdot C_t \]

\[ + F'_4 \cdot F'_2 \cdot F_1 \cdot C_t \]

\[ + F_3 \cdot O_1 \]

\[ + F_3 \cdot E_{30} \cdot M'_2 \]

\[ + F_3 \cdot E_{30} \cdot T_x \cdot N_{12}' \]

\[ + F_3 \cdot N_6 \]

\[ + S_c \cdot I_4 \]

\[ + F'_4 \cdot F'_3 \cdot F'_2 \cdot N_8 \cdot D'_5 \cdot H_1 \]

\[ + F'_3 \cdot F'_1 \cdot N_8 \cdot D'_5 \cdot F'_2 \]

\[ + F'_4 \cdot F'_2 \cdot F'_1 \cdot N_8 \cdot D'_5 \]

\[ + S_x \cdot F'_4 \cdot F'_3 \cdot F'_2 \cdot N_8 \cdot D'_5 \]

\[ + I_4 \cdot (S_{u1} + S_{u2} + S_{u3}) \]

\[ + N_{8a} \cdot A_{42} \cdot S_x \cdot F_1 \]
\[ a_{f_2} = F_2 F_1' C_t \]
\[ + F_4 F_1 C_t \]
\[ + F_5 F_3 F_2 C_t \]
\[ + F_3' O_1 \]
\[ + F_3' E_{30} \]
\[ + F_3' T_x \]
\[ + F_3' N_6 \]
\[ + N_{12} \]
\[ + Z_{CH} I_3' \]
\[ + M_3 N_8 \]
\[ + D_5 P_0' P_{d3} T_{40} \]
\[ + D_5' P_{d3} T_{41} \]
\[ + F_3' F_2 F_1 N_{8a} \]
\[ + F_3 F_2 F_1' N_{8a} \]
\[ + M_2 E_{30} \]
\[ + N_{8a} F_2 F_4 \]
INPUT-OUTPUT REGISTER (Cont)

\[ F_3 = K_3 \cdot I_4 \cdot (TPP)P_5 \]

\[ + S_c \cdot I_4 \]

\[ + F_5' \cdot F_4 \cdot C_t \]

\[ + F_4' \cdot F_5' \cdot F_2' \cdot F_1' \cdot C_t \]

\[ + A_1 \cdot A_n' \cdot O_1 \]

\[ + F_4 \cdot E_{30} \]

\[ + B_2 \cdot M_2 \cdot T_x \cdot N_{12}' \]

\[ + B_1 \cdot A_{42}' \cdot S_x' \cdot T_x \cdot M_2' \cdot N_{12}' \]

\[ + F_4 \cdot A_{42} \cdot T_x \cdot N_{12}' \]

\[ + F_4' \cdot F_3' \cdot F_2' \cdot N_8 \cdot D_5' \cdot H_1 \]

\[ + F_4' \cdot N_6 \]

\[ + F_4' \cdot F_3' \cdot F_2' \cdot N_8 \cdot S_x' \cdot D_5' \]

\[ + F_4 \cdot F_1' \cdot N_8 \cdot D_5' \]

\[ + S_{u1} \cdot I_4 \]

\[ + F_4 \cdot A_n \cdot O_1 \]

0f3 (Next Page)
\[ o f_3 = F_5 F_2 C_t \]
\[ + F_5' F_4' F_1' C_t \]
\[ + A_1' A_n' O_1 \]
\[ + F_4' E_{30} N_0 \]
\[ + B_1' A_{42} T_x M_2' \]
\[ + N_{12} \]
\[ + F_4' A_{42} M_2' T_x \]
\[ + F_4' N_6 \]
\[ + Z_{ch} I_3' \]
\[ + M_3' N_8 \]
\[ + D_5' P_0' P_{d3} T_{40} \]
\[ + D_5' P_{d3} T_{41} \]
\[ + F_4' F_3 F_2' N_{80} \]
\[ + B_2' M_2 T_x \]
\[ + F_4' A_n O_1 \]

508-T-6

CIRCUIT BOARD LOCATION

INPUT-OUTPUT REGISTER (Cont)
INPUT-OUTPUT REGISTER (Cont)

\[ F_4 = K_4 \cdot I_4 \cdot (TPP) \cdot P_5 \]

\[ + F_5 \cdot F_3 \cdot F_2 \cdot F_1 \cdot C_t \]

\[ + F_5' \cdot F_4' \cdot F_1' \cdot C_t \]

\[ + F_5' \cdot F_4' \cdot F_3' \cdot F_2 \cdot C_t \]

\[ + B_1 \cdot A_{42} \cdot M_2 \cdot N_{12} \cdot S_x \cdot T_x \]

\[ + F_5' \cdot N_6 \]

\[ + F_3 \cdot F_2' \cdot F_1' \cdot N_8 \cdot D_5' \]

\[ + F_5 \cdot O_1 \cdot A_N \]

\[ + N_{9a} \cdot A_{42} \cdot S_x \cdot F_1 \]

\[ 207-12 \]

\[ 106-25 \]

\[ 107-25 \]

\[ 106-25 \]

\[ 106-9 \]

\[ 207-12 \]

\[ 107-25 \]

\[ 207-10 \]

\[ 206-24 \]
\[ of_4 = F_4 F_2 C_t \]
\[ + E_{30} \]
\[ + B'_1 A_{42} T_x \]
\[ + F'_5 N_6 \]
\[ + Z_{CH} I'_3 \]
\[ + M_3 N_B \]
\[ + D_5 P'_0 P_{d3} T_{40} \]
\[ + D'_5 P_{d3} T_{41} \]
\[ + N_{12} \]
\[ + F'_3 F'_1 N_{8A} \]
\[ + F'_5 O_1 \]
\[ + A_0 P_{d3} F_3 T_0 \]

\[ \text{CIRCUIT BOARD LOCATION} \]

\[ 407-12 \]
\[ 407-19 \]
\[ 406-12 \]
\[ 407-34 \]
\[ 407-34 \]
\[ 407-12 \]
\[ 406-27 \]
\[ 406-27 \]
\[ 408-19 \]
\[ 406-12 \]
\[ 406-18 \]
\[ 408-24 \]
INPUT-OUTPUT REGISTER (Cont)

\[ F_5 \]
\[ f_5 = K_5 I_4 \underbrace{(T'}_5 P \overline{P}) P_5 \]
\[ + F_4' F_3' F_1 C_t \]
\[ + F_4' F_3 F_2 C_t \]
\[ + F_5' F_4 F_3' F_1' C_t \]
\[ + Z_1 N_6 \]
\[ + F_4' F_3' F_2' N_8 H_1 \]
\[ + F_2 F_1' N_8 F_4' \]
\[ + F_2' F_1 N_8 \]
\[ + F_4 F_3 N_8 \]
\[ + F_4' F_3' F_2' N_8 S_x' \]
\[ + A_1 O_1 \]

423-12
423-27
422-27
523-9
423-34
523-25
424-24
423-27
422-27
523-25
423-34

0f5 (Next Page)
INPUT-OUTPUT REGISTER (Co-:)

\[ o_5 = F_4 F_3' F_1 C_t \]
\[ + F_4 F_3 F_2 C_t \]
\[ + F_4' F_3 F_2' F_1' C_t \]
\[ + F_5 F_3' F_2 F_1' C_t \]
\[ + Z_1' N_6 \]
\[ + N_{12} \]
\[ + P_{d3} T_{40} \]
\[ + Z_{CH} I_3' \]
\[ + D_5' P_{d3} T_{41} \]
\[ + A_1' O_1 A_N \]
<table>
<thead>
<tr>
<th>Format Control Flip-Flop</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_{c1} )</td>
<td>( f_{c1} = O_f )</td>
</tr>
<tr>
<td></td>
<td>( o_f = A_{42} T_x D_i )</td>
</tr>
<tr>
<td></td>
<td>+ ( C_f )</td>
</tr>
<tr>
<td></td>
<td>+ ( N_0 R_c' )</td>
</tr>
<tr>
<td></td>
<td>+ ( I_4 )</td>
</tr>
<tr>
<td></td>
<td>+ ( R_c U_1 )</td>
</tr>
<tr>
<td>( F_{c2} )</td>
<td>( f_{c2} = A_{42} T_x D_i )</td>
</tr>
<tr>
<td></td>
<td>+ ( O_f )</td>
</tr>
<tr>
<td></td>
<td>+ ( C_f )</td>
</tr>
<tr>
<td></td>
<td>+ ( D_f I_a )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>403-19</td>
<td></td>
</tr>
<tr>
<td>404-10</td>
<td></td>
</tr>
<tr>
<td>404-19</td>
<td></td>
</tr>
<tr>
<td>405-10</td>
<td></td>
</tr>
<tr>
<td>402-10</td>
<td></td>
</tr>
<tr>
<td>456-10</td>
<td></td>
</tr>
<tr>
<td>456-34</td>
<td></td>
</tr>
<tr>
<td>456-34</td>
<td></td>
</tr>
<tr>
<td>456-12</td>
<td></td>
</tr>
<tr>
<td>456-12</td>
<td></td>
</tr>
</tbody>
</table>
INPUT ZEROING-KEYBOARD STOP FLIP-FLOP

\[ z_{ch} = N_0 P_0 T_{41} \]
\[ + P_0 E_2 T_{41} \]
\[ + M_3 I_4 \]
\[ + T_d I_4 \]
\[ + (S_0 + H_c) I_4 T_{41} \]
\[ + E_3 Y_8 Y_1 Y_0 I_3 U_1 T_2 Y_2 \]
\[ + M_5 S_{50c} \]
\[ o z_{ch} = J_{13} \]
\[ + z_{ch} I_3 T_{41}' \]

CIRCUIT BOARD LOCATION

211-10
212-12
211-34
211-34
212-12
111-18
214-18
209-19
210-10
INPUT SEQUENCE COUNTER

\[ E_1 \quad e_1 = E_1' P_0 T_{41a} \]
\[ + C_h' \]
\[ + A_0 P_{d3} F_4 F_3 \]
\[ o e_1 = E_1' P_0 T_{41a} \]
\[ + N_0 E_3 E_1 \]
\[ + F_0 E_3' T_{40} \]
\[ + Z_2 N_q A_n U_2 \]

\[ E_2 \quad e_2 = E_2' E_1 P_0 T_{41} \]
\[ + N_0 E_3 E_1 \]
\[ + S_p Y_{12} Y_{11} Y_{10} Y_q A_n' I_4' T_{41} \]
\[ + Z_2' A_n U_2 N_1 P_4' P_3' \]
\[ o e_2 = E_2' E_1 P_0 T_{41} \]
\[ + C_h' I_2' \]
\[ + P_{d3} N_0 F_3 F_4 F_5 \]
INPUT SEQUENCE COUNTER (Cont)

\[
E_3 \quad _1 e_3 = E_3' E_2 E_1 P_0 T_{41} \\
+ F_0 E_3' T_{40} E_1 R_{c1} \\
+ E_2 Y_7 Y_6 Y_5 Y_4 Y_3 T_1 I_4' \\
+ J_6 J_5 J_4 I_2 R_{c1} T_1 \\
+ Z_2' A_n U_2 N_1 P_5' P_3 \\
- \\
\]

\[
\quad 5 e_3 = E_3 E_2 E_1 P_0 T_{41} \\
+ N_0 E_3 E_1 \\
+ I_2 R_{c1} T_{41} \\
+ C_4 I_2' T_1' \\
\]

CIRCUIT BOARD LOCATION

119-25
119-25
121-18
122-18
114-18
523-18
422-10
422-12
422-12

508-T-6
"NUMBER FILL" STATE FLIP-FLOPS

\[ E_{20} = N_0 F_5 F_4' E_2 P_{d3} \]
\[ + N_0 F_5 F_3' E_2 P_{d3} \]
\[ + M_2 T_x U_{41} \]
\[ o e_{20} = E_{20} T_{41a} I_4 \]
\[ + N_{12} \]

\[ E_{30} = E_{20} T_{41a} N_0 \]
\[ + M_2 E_{20} U_{41} T_{40} \]
\[ o e_{30} = I_4 E_{30} T_{41a} \]
\[ + M_2 T_1 \]
\[ + Q_6 Q_4 Q_1 T_{41a} \]

\[ E_{40} = E_{30} N_0 T_{41a} \]
\[ o e_{40} = E_{40} T_{41a} \]
INPUT COUNTER RESET FLIP-FLOPS

\[ C_r = F_0 \cdot T_2 ' + O_0 \cdot T_2 ' + N_0 \cdot T_2 ' + A_n \cdot U_2 + A_n \cdot A_t \]
\[ o_c_r = A'_n \cdot F_4 \cdot F_3 \cdot P_d \cdot T_2 + F'_o \cdot O'_o \cdot N'_o \cdot A_n' \cdot I'_3 + R_b + N_{12} \]

INHIBIT FILL FLIP-FLOP

\[ A_o = N_o \cdot F_5 \cdot F_4 \cdot F_3 \cdot F_2 \cdot F'_1 \cdot P_d \cdot T_1 + F'_5 \cdot F_4 \cdot F'_3 \cdot F_2 \cdot N_o \cdot P_d \cdot T_1 \]
\[ o_a_o = F_5 \cdot P_d \cdot T_4_{10} + I'_4 + R_b + T_1 \]
TAPE READER "ON" FLIP-FLOPS

\[ T_o \quad t_o = (R_{gm} + T_g) I_4 \]  
\[ + V_{gm} I_4 \]  
\[ + M_7 D_3' I_4 \]  
\[ + R_{gp} I_4 \]  
\[ + V_{gp} I_4 \]  
\[ o t_o = K_f \]  
\[ + F_5 F_4' F_3 F_2' F_1' C_t \]  
\[ + I_4' \]  
\[ + T_d \]  
\[ + (R_{sm} + R_{sp}) \]

PHOTOREADER CLUTCH-BRAKE FLIP-FLOPS

\[ B_c \quad b_c = T_o Q_o D_{r'} \]  
\[ o b_c = T_o' \]  
\[ + Q_o' \]  
\[ + D_{r'} \]
<table>
<thead>
<tr>
<th>CODE CONVERSION TIME FLIP-FLOP</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_t$ 1 $C_t = (TPP)' T_0 P_6 P_3 P_2 P_1 P'$</td>
<td>113-18</td>
</tr>
<tr>
<td>$0 C_t = C_t$</td>
<td>213-19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAIN MEMORY WRITE FLIP-FLOPS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{\text{wl}}$ 1 $m_{\text{wl}} = A_2 M_3 T_4$</td>
<td>226-10</td>
</tr>
<tr>
<td>$+ B_2 M_6 D_3 T_4$</td>
<td>227-24</td>
</tr>
<tr>
<td>$+ L_2 M_6 D_2 D_1 T_4$</td>
<td>126-9</td>
</tr>
<tr>
<td>$+ V_2 M_6 D_2 D_1 T_4$</td>
<td>127-9</td>
</tr>
<tr>
<td>$+ X_2 I_1 T_4$</td>
<td>227-10</td>
</tr>
<tr>
<td>$+ D_1$</td>
<td>227-19</td>
</tr>
<tr>
<td>$0 m_{\text{wl}} = A_2' M_3 D_1$</td>
<td>224-10</td>
</tr>
<tr>
<td>$+ B_2' M_6 D_3' D_1$</td>
<td>226-24</td>
</tr>
<tr>
<td>$+ L_2' M_6 D_2' D_1 D_1$</td>
<td>124-9</td>
</tr>
<tr>
<td>$+ V_2' M_6 D_2' D_1 D_1$</td>
<td>125-9</td>
</tr>
<tr>
<td>$+ X_2' I_1 D_1$</td>
<td>225-10</td>
</tr>
<tr>
<td>$+ D_1 T_4$</td>
<td>227-18</td>
</tr>
</tbody>
</table>


MAIN MEMORY WRITE FLIP-FLOPS (Cont)

\[ M_{w2} = A_2 M_3 \]
\[ + B_2 M_6 D_3 \]
\[ + L_2 M_6 D_2 D_1 \]
\[ + V_2 M_6 D_2 D_1 \]
\[ + X_2 I_1 \]
\[ + T_{41} \]

\[ m_{w2} = A_2 M_3 D_{11} T_{41} \]
\[ + B_2 M_6 D_3 D_{11} T_{41} \]
\[ + L_2 M_6 D_2 D_1 D_{11} T_{41} \]
\[ + V_2 M_6 D_2 D_1 D_{11} T_{41} \]
\[ + X_2 I_1 D_{11} T_{41} \]
MEMORY WRITE AMPLIFIERS

\[ \begin{align*}
1w_0 &= \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{w2}^o \\
o_{w0} &= \left( \quad \right) M_{w1}' \\
1w_1 &= \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{w2}^o \\
o_{w1} &= \left( \quad \right) M_{w1}' \\
1w_2 &= \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{w2}^o \\
o_{w2} &= \left( \quad \right) M_{w1}' \\
1w_3 &= \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{w2}^o \\
o_{w3} &= \left( \quad \right) M_{w1}' \\
1w_4 &= \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{w2}^o \\
o_{w4} &= \left( \quad \right) M_{w1}' \\
1w_5 &= \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{w2}^o \\
o_{w5} &= \left( \quad \right) M_{w1}'
\end{align*} \]
MEMORY WRITE AMPLIFIERS (Cont)

\[
\begin{align*}
1w_6 &= (C_6' C_5' C_4' C_3' C_2' C_1') M_{w2}' \\
oc_6 &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w1}' \\
1w_7 &= (C_6' C_5' C_4' C_3 C_2 C_1') M_{w2}' \\
oc_7 &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w1}' \\
1w_8 &= (C_6' C_5' C_4 C_3' C_2' C_1') M_{w2}' \\
oc_8 &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w1}' \\
1w_9 &= (C_6' C_5' C_4 C_3' C_2 C_1') M_{w2}' \\
oc_9 &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w1}' \\
1w_{10} &= (C_6' C_5' C_4 C_3' C_2 C_1') M_{w2}' \\
oc_{10} &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w1}' \\
1w_{11} &= (C_6' C_5' C_4 C_3' C_2 C_1') M_{w2}' \\
oc_{11} &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w1}'
\end{align*}
\]
MEMORY WRITE AMPLIFIERS (Cont)

\[ I_{W_{12}} = (c_6' c_5' c_4 c_3 c_2 c_1') M_{w_2} \\
O_{W_{12}} = (\quad " \quad) M_{w_1} \]

\[ I_{W_{13}} = (c_6' c_5' c_4 c_3 c_2 c_1') M_{w_2} \\
O_{W_{13}} = (\quad " \quad) M_{w_1} \]

\[ I_{W_{14}} = (c_6' c_5' c_4 c_3 c_2 c_1') M_{w_2} \\
O_{W_{14}} = (\quad " \quad) M_{w_1} \]

\[ I_{W_{15}} = (c_6' c_5' c_4 c_3 c_2 c_1') M_{w_2} \\
O_{W_{15}} = (\quad " \quad) M_{w_1} \]

\[ I_{W_{16}} = (c_6' c_5' c_4' c_3 c_2 c_1') M_{w_2} \\
O_{W_{16}} = (\quad " \quad) M_{w_1} \]

\[ I_{W_{17}} = (c_6' c_5' c_4' c_3 c_2' c_1') M_{w_2} \\
O_{W_{17}} = (\quad " \quad) M_{w_1} \]
MEMORY WRITE AMPLIFIERS (Cont)

\[
1w_{18} = (c_6' c_5' c_4' c_3' c_2' c_1') M_{w2}
\]

\[
o_{w18} = (\text{" } \text{" } \text{" } \text{" } \text{" } \text{" } ) M_{w1}
\]

\[
1w_{19} = (c_6' c_5' c_4' c_3' c_2' c_1') M_{w2}
\]

\[
o_{w19} = (\text{" } \text{" } \text{" } \text{" } \text{" } \text{" } ) M_{w1}
\]

\[
1w_{20} = (c_6' c_5' c_4' c_3' c_2' c_1') M_{w2}
\]

\[
o_{w20} = (\text{" } \text{" } \text{" } \text{" } \text{" } \text{" } ) M_{w1}
\]

\[
1w_{21} = (c_6' c_5' c_4' c_3' c_2' c_1') M_{w2}
\]

\[
o_{w21} = (\text{" } \text{" } \text{" } \text{" } \text{" } \text{" } ) M_{w1}
\]

\[
1w_{22} = (c_6' c_5' c_4' c_3' c_2' c_1') M_{w2}
\]

\[
o_{w22} = (\text{" } \text{" } \text{" } \text{" } \text{" } \text{" } ) M_{w1}
\]

\[
1w_{23} = (c_6' c_5' c_4' c_3' c_2' c_1') M_{w2}
\]

\[
o_{w23} = (\text{" } \text{" } \text{" } \text{" } \text{" } \text{" } ) M_{w1}
\]
MEMORY WRITE AMPLIFIERS (Cont)

\[ \begin{align*}
I_{W24} &= (C_6' C_5' C_4' C_3' C_2' C_1') M_{w2}' \\
O_{W24} &= ( \quad ) M_{w1}' \\
I_{W25} &= (C_6' C_5' C_4' C_3' C_2' C_1') M_{w2}' \\
O_{W25} &= ( \quad ) M_{w1}' \\
I_{W26} &= (C_6' C_5' C_4' C_3' C_2' C_1') M_{w2}' \\
O_{W26} &= ( \quad ) M_{w1}' \\
I_{W27} &= (C_6' C_5' C_4' C_3' C_2' C_1') M_{w2}' \\
O_{W27} &= ( \quad ) M_{w1}' \\
I_{W28} &= (C_6' C_5' C_4' C_3' C_2' C_1') M_{w2}' \\
O_{W28} &= ( \quad ) M_{w1}' \\
I_{W29} &= (C_6' C_5' C_4' C_3' C_2' C_1') M_{w2}' \\
O_{W29} &= ( \quad ) M_{w1}'
\end{align*} \]
MEMORY WRITE AMPLIFIERS (Cont)

\[
\begin{align*}
W_{30} &= (C_6' C_5 C_4 C_3 C_2 C_1') M_{w2}' \\
0W_{30} &= (2^{n-1} 0^{n-1}) M_{w1}' \\
W_{31} &= (C_6' C_5 C_4 C_3 C_2 C_1') M_{w2}' \\
0W_{31} &= (2^{n-1} 0^{n-1}) M_{w1}' \\
W_{32} &= (C_6' C_5 C_4 C_3 C_2 C_1') M_{w2}' \\
0W_{32} &= (2^{n-1} 0^{n-1}) M_{w1}' \\
W_{33} &= (C_6' C_5 C_4 C_3 C_2 C_1') M_{w2}' \\
0W_{33} &= (2^{n-1} 0^{n-1}) M_{w1}' \\
W_{34} &= (C_6' C_5 C_4 C_3 C_2 C_1') M_{w2}' \\
0W_{34} &= (2^{n-1} 0^{n-1}) M_{w1}' \\
W_{35} &= (C_6' C_5 C_4 C_3 C_2 C_1') M_{w2}' \\
0W_{35} &= (2^{n-1} 0^{n-1}) M_{w1}'
\end{align*}
\]
MEMORY WRITE AMPLIFIERS (Cont)

\[ i_{W36} = (C_6 C'_5 C'_4 C'_3 C'_2 C'_1) M_{W2} \]
\[ o_{W36} = (\text{""}) M_{W1} \]

\[ i_{W37} = (C_6 C'_5 C'_4 C'_3 C'_2 C'_1) M_{W2} \]
\[ o_{W37} = (\text{""}) M_{W1} \]

\[ i_{W38} = (C_6 C'_5 C'_4 C'_3 C'_2 C'_1) M_{W2} \]
\[ o_{W38} = (\text{""}) M_{W1} \]

\[ i_{W39} = (C_6 C'_5 C'_4 C'_3 C'_2 C'_1) M_{W2} \]
\[ o_{W39} = (\text{""}) M_{W1} \]

\[ i_{W40} = (C_6 C'_5 C'_4 C'_3 C'_2 C'_1) M_{W2} \]
\[ o_{W40} = (\text{""}) M_{W1} \]

\[ i_{W41} = (C_6 C'_5 C'_4 C'_3 C'_2 C'_1) M_{W2} \]
\[ o_{W41} = (\text{""}) M_{W1} \]
MEMORY WRITE AMPLIFIERS (Cont)

\[ W_{42} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w2} \]
\[ W_{42} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w1} \]

\[ W_{43} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w2} \]
\[ W_{43} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w1} \]

\[ W_{44} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w2} \]
\[ W_{44} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w1} \]

\[ W_{45} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w2} \]
\[ W_{45} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w1} \]

\[ W_{46} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w2} \]
\[ W_{46} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w1} \]

\[ W_{47} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w2} \]
\[ W_{47} = (C_6 C_5' C_4 C_3' C_2 C_1') M_{w1} \]
MEMORY WRITE AMPLIFIERS (Cont)

\[ \begin{align*}
1W_{48} &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w2} \\
0W_{48} &= (\quad) M_{w1}
\end{align*} \]

\[ \begin{align*}
1W_{49} &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w2} \\
0W_{49} &= (\quad) M_{w1}
\end{align*} \]

\[ \begin{align*}
1W_{50} &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w2} \\
0W_{50} &= (\quad) M_{w1}
\end{align*} \]

\[ \begin{align*}
1W_{51} &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w2} \\
0W_{51} &= (\quad) M_{w1}
\end{align*} \]

\[ \begin{align*}
1W_{52} &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w2} \\
0W_{52} &= (\quad) M_{w1}
\end{align*} \]

\[ \begin{align*}
1W_{53} &= (C_6 C_5 C_4 C_3 C_2 C_1) M_{w2} \\
0W_{53} &= (\quad) M_{w1}
\end{align*} \]

508-T-6 489
MEMORY WRITE AMP LiFiERS (Cont)

\[ I_{W54} = \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{W2} \]

\[ O_{W54} = \left( \right) M_{W1} \]

\[ I_{W55} = \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{W2} \]

\[ O_{W55} = \left( \right) M_{W1} \]

\[ I_{W56} = \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{W2} \]

\[ O_{W56} = \left( \right) M_{W1} \]

\[ I_{W57} = \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{W2} \]

\[ O_{W57} = \left( \right) M_{W1} \]

\[ I_{W58} = \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{W2} \]

\[ O_{W58} = \left( \right) M_{W1} \]

\[ I_{W59} = \left( C_6 C_5 C_4 C_3 C_2 C_1 \right) M_{W2} \]

\[ O_{W59} = \left( \right) M_{W1} \]
MEMORY WRITE AMPLIFIERS (Cont)

\[ I_{W60} = (C_6 C_5 C_4 C_3 C_2 C_1) M_{w2} \]

\[ 0_{W60} = (\quad \quad \quad ) M_{w1} \]

\[ I_{W61} = (C_6 C_5 C_4 C_3 C_2' C_1') M_{w2} \]

\[ 0_{W61} = (\quad \quad \quad ) M_{w1} \]

\[ I_{W62} = (C_6 C_5 C_4 C_3 C_2 C_1') M_{w2} \]

\[ 0_{W62} = (\quad \quad \quad ) M_{w1} \]

\[ I_{W63} = (C_6 C_5 C_4 C_3 C_2 C_1) M_{w2} \]

\[ 0_{W63} = (\quad \quad \quad ) M_{w1} \]
MEMORY READ POWER AMPLIFIERS

\[ M_{\alpha a} \]

\[ M_{\beta a} = \left[ (C_3' C_2' C_1') M_{\lambda 0} + (C_3' C_2' C_1') M_{\lambda 1} + (C_3' C_2' C_1') M_{\lambda 2} \right. \]
\[ + (C_3' C_2' C_1') M_{\lambda 3} + (C_3' C_2' C_1') M_{\lambda 4} + (C_3' C_2' C_1') M_{\lambda 5} \]
\[ + (C_3' C_2' C_1') M_{\lambda 6} + (C_3' C_2' C_1') M_{\lambda 7} + M_s \]

\[ M_{\alpha b} \]

\[ M_{\beta b} = \left[ (C_3' C_2' C_1') M_{\lambda 8} + (C_3' C_2' C_1') M_{\lambda 9} + (C_3' C_2' C_1') M_{\lambda 10} \right. \]
\[ + (C_3' C_2' C_1') M_{\lambda 11} + (C_3' C_2' C_1') M_{\lambda 12} + (C_3' C_2' C_1') M_{\lambda 13} \]
\[ + (C_3' C_2' C_1') M_{\lambda 14} + (C_3' C_2' C_1') M_{\lambda 15} \]
\[ + (C_3' C_2' C_1') M_{\lambda 16} + (C_3' C_2' C_1') M_{\lambda 17} + M_s \]
MEMORY READ POWER AMPLIFIERS (Cont)

\[ M_{\text{ac}} \]

\[ m_{\text{ac}} = [(C'_3 C'_2 C'_1) M_{\text{ac}16} + (C'_3 C'_2 C'_1) M_{\text{ac}17} + (C'_3 C'_2 C'_1) M_{\text{ac}18} \\
+ (C'_3 C'_2 C'_1) M_{\text{ac}19} + (C'_3 C'_2 C'_1) M_{\text{ac}20} + (C'_3 C'_2 C'_1) M_{\text{ac}21} \\
+ (C'_3 C'_2 C'_1) M_{\text{ac}22} + (C'_3 C'_2 C'_1) M_{\text{ac}23}] M_s + (C'_3 C'_2 C'_1) M_{\text{ac}24} + (C'_3 C'_2 C'_1) M_{\text{ac}25} + M_s \]

\[ M_{\text{ad}} \]

\[ m_{\text{ad}} = [(C'_3 C'_2 C'_1) M_{\text{ad}14} + (C'_3 C'_2 C'_1) M_{\text{ad}25} + (C'_3 C'_2 C'_1) M_{\text{ad}26} \\
+ (C'_3 C'_2 C'_1) M_{\text{ad}27} + (C'_3 C'_2 C'_1) M_{\text{ad}28} + (C'_3 C'_2 C'_1) M_{\text{ad}29} \\
+ (C'_3 C'_2 C'_1) M_{\text{ad}30} + (C'_3 C'_2 C'_1) M_{\text{ad}31}] M_s + (C'_3 C'_2 C'_1) M_{\text{ad}34} + (C'_3 C'_2 C'_1) M_{\text{ad}35} + M_s \]
MEMORY READ POWER AMPLIFIERS (Cont)

\[ M_{\text{re}} \]

\[ 1_{M_{\text{re}}} = \left[ (C_3' C_2' C_1') M_{\lambda 32} + (C_3' C_2' C_1') M_{\lambda 33} + (C_3' C_2' C_1') M_{\lambda 34} \\
+ (C_3 C_2 C_1') M_{\lambda 35} + (C_3 C_2 C_1') M_{\lambda 36} + (C_3 C_2 C_1') M_{\lambda 37} \\
+ (C_3 C_2 C_1') M_{\lambda 38} + (C_3 C_2 C_1') M_{\lambda 39} \right] M_5 \]

\[ 0_{M_{\text{re}}} = \left[ (C_3' C_2' C_1') M_{\lambda 32} + (C_3' C_2' C_1') M_{\lambda 33} + (C_3' C_2' C_1') M_{\lambda 34} \\
+ (C_3' C_2' C_1') M_{\lambda 35} + (C_3' C_2' C_1') M_{\lambda 36} + (C_3' C_2' C_1') M_{\lambda 37} \\
+ (C_3' C_2' C_1') M_{\lambda 38} + (C_3' C_2' C_1') M_{\lambda 39} + M_5 \right] \]

\[ M_{\text{rf}} \]

\[ 1_{M_{\text{rf}}} = \left[ (C_3' C_2' C_1') M_{\lambda 40} + (C_3' C_2' C_1') M_{\lambda 41} + (C_3' C_2' C_1') M_{\lambda 42} \\
+ (C_3' C_2' C_1') M_{\lambda 43} + (C_3' C_2' C_1') M_{\lambda 44} + (C_3' C_2' C_1') M_{\lambda 45} \\
+ (C_3' C_2' C_1') M_{\lambda 46} + (C_3' C_2' C_1') M_{\lambda 47} \right] M_5 \]

\[ 0_{M_{\text{rf}}} = \left[ (C_3' C_2' C_1') M_{\lambda 40} + (C_3' C_2' C_1') M_{\lambda 41} + (C_3' C_2' C_1') M_{\lambda 42} \\
+ (C_3' C_2' C_1') M_{\lambda 43} + (C_3' C_2' C_1') M_{\lambda 44} + (C_3' C_2' C_1') M_{\lambda 45} \\
+ (C_3' C_2' C_1') M_{\lambda 46} + (C_3' C_2' C_1') M_{\lambda 47} + M_5 \right] \]
MEMORY READ POWER AMPLIFIERS (Cont)

\[ m_{\text{Lg}} = \left[ (C_3' C_2' C_i') M_{\text{Lg}48} + (C_3' C_2' C_i') M_{\text{Lg}49} + (C_3' C_2' C_i') M_{\text{Lg}50} \right. \\
+ (C_3' C_2' C_i') M_{\text{Lg}51} + (C_3' C_2' C_i') M_{\text{Lg}52} + (C_3' C_2' C_i') M_{\text{Lg}53} \\
+ (C_3' C_2' C_i') M_{\text{Lg}54} + (C_3' C_2' C_i') M_{\text{Lg}55} \left. \right] M_s \]

\[ o_{\text{Lg}} = \left[ (C_3' C_2' C_i') M_{\text{Lg}48} + (C_3' C_2' C_i') M_{\text{Lg}49} + (C_3' C_2' C_i') M_{\text{Lg}50} \right. \\
+ (C_3' C_2' C_i') M_{\text{Lg}51} + (C_3' C_2' C_i') M_{\text{Lg}52} + (C_3' C_2' C_i') M_{\text{Lg}53} \\
+ (C_3' C_2' C_i') M_{\text{Lg}54} + (C_3' C_2' C_i') M_{\text{Lg}55} + M_s \left. \right] \]

\[ m_{\text{Lh}} = \left[ (C_3' C_2' C_i') M_{\text{Lh}56} + (C_3' C_2' C_i') M_{\text{Lh}57} + (C_3' C_2' C_i') M_{\text{Lh}58} \right. \\
+ (C_3' C_2' C_i') M_{\text{Lh}59} + (C_3' C_2' C_i') M_{\text{Lh}60} + (C_3' C_2' C_i') M_{\text{Lh}61} \\
+ (C_3' C_2' C_i') M_{\text{Lh}62} + (C_3' C_2' C_i') M_{\text{Lh}63} \left. \right] M_s \]

\[ o_{\text{Lh}} = \left[ (C_3' C_2' C_i') M_{\text{Lh}56} + (C_3' C_2' C_i') M_{\text{Lh}57} + (C_3' C_2' C_i') M_{\text{Lh}58} \right. \\
+ (C_3' C_2' C_i') M_{\text{Lh}59} + (C_3' C_2' C_i') M_{\text{Lh}60} + (C_3' C_2' C_i') M_{\text{Lh}61} \\
+ (C_3' C_2' C_i') M_{\text{Lh}62} + (C_3' C_2' C_i') M_{\text{Lh}63} + M_s \left. \right] \]
MEMORY SYNCHRONIZATION FLIP-FLOP

\[ M_5 \quad 1m_5 = T_{40} \]
\[ o_m_5 = T_{41a} \]

CLOCK JITTER FLIP-FLOP

\[ C_j \quad 1c_j = C_j \]
\[ o_c_j = C_j \]

MEMORY READ FLIP-FLOP

\[ M_r \quad 1m_r = C_6' C_5' C_4' M_{ra} \]
\[ + C_6' C_5' C_4 M_{rb} \]
\[ + C_6' C_5 C_4' M_{rc} \]
\[ + C_6 C_5 C_4 M_{rd} \]
\[ + C_6 C_5' C_4' M_{re} \]
\[ + C_6 C_5' C_4 M_{rf} \]
\[ + C_6 C_5 C_4 M_{rg} \]
\[ + C_6 C_5 C_4 M_{rh} \]

Next Page

CIRCUIT BOARD LOCATION

458-19

456-19

421-19

220-19

154-25

154-25

153-25

153-25

152-25

152-25

252-24

153-9

508-T-6
\[ \alpha M_R = C_6' C_5' C_4' M_{R_a} + C_6' C_5' C_4 M_{R_b} + C_6' C_5 C_4' M_{R_c} + C_6' C_5 C_4 M_{R_d} + C_6 C_5' C_4' M_{R_e} + C_6 C_5' C_4 M_{R_f} + C_6 C_5 C_4' M_{R_g} + C_6 C_5 C_4 M_{R_h} \]

554-25
554-25
552-9
552-25
553-25
553-25
452-24
453-24

508-T-b
APPENDIX III. CIRCUIT BOARD CHARTS

COMPUTER CONNECTOR SIGNAL CHART

The computer connector signal chart represents each cable receptacle, plug receptacle, and terminal board located in the computer assembly, excluding the signal circuit board receptacles and the power circuit board receptacles in the power supply assembly. The signal found at each pin of these receptacles is identified. (See figure .)

Each square represents a pin or a jack on the receptacle, and the letter designation or the combination letter and number designation within individual squares represents the signal or voltage found at that pin or jack. These designations are those used for the computer logic equations, or are defined in the computer wire list.

CIRCUIT BOARD CONNECTOR SIGNAL CHARTS

Six circuit board connector signal charts are required to represent each plug-in circuit board receptacle mounted on the two side castings of the computer assembly. Again, the signal charts also identify each signal found at each pin or jack of these receptacles, with individual squares representing individual pins or jacks. Each connector signal chart illustrates one complete row of 40 connectors. There are six such rows of connectors, three on each side casting; thus, there are six charts. (See figures 41 through 47.)

Two of the rows of connectors on each side casting are made up of double-receptacle connectors, and one is made up of single-receptacle connectors. Therefore, there are actually five rows of receptacles on each side: number designations 101-140 through 501-540 (side No. 1) and 141-180 through 541-580 (side No. 2). Thus, a 3-digit number specifies exactly one receptacle. For example, receptacle 129 specifies the top receptacle in the twenty-ninth position of side No. 1.

The layout of each signal chart is discussed in the following paragraphs.
100-200 SERIES-SIDE NO. 1

This chart is arranged to facilitate signal checking of the logic network boards plugged into receptacles 102 and 202 through 131 and 231. All the pins on the particular board are represented by one horizontal row of squares. The pin numbers for each logic network board are printed at the top of the chart and grouped into individual gates on the boards. The output pin numbers are raised above the input pin numbers for each gate. The number at the top of each gate grouping indicates the number of diodes used in that gate.

The second part of this chart illustrates the signals found at each pin of the eight write switches plugged into side No. 1.

300 SERIES-SIDE NO. 1

This chart is arranged to facilitate signal checking of the flip-flop boards plugged into side No. 1. As for all other signal charts, one horizontal row of squares represents all the pins on a particular circuit board. The pin numbers along the top of the chart are grouped into four sets of input-output signals (four flip-flops on a board) and one set of voltages. Also the signal found at each pin of the write amplifier circuit boards and the clock power amplifier boards is shown.

400-500 SERIES-SIDE NO. 1

This chart is arranged similarly to the 100-200 series chart for side No. 1. This arrangement facilitates signal checking of the logic network boards plugged into 402 and 502 through 424 and 524. Also, the signals found at each pin of 3 computer network boards and the 11 nonmain memory read amplifier boards are illustrated.

100-200 SERIES-SIDE NO. 2

This chart is arranged similarly to the 100-200 series chart for side No. 1. This arrangement facilitates signal checking of the logic network boards plugged into 151 and 251 through 179 and 279. Also, the signals found at each pin of the other three computer network boards are illustrated.

300 SERIES-SIDE NO. 2

This chart is arranged similarly to the 300 series chart for side No. 1. This arrangement facilitates signal checking of the flip-flop
boards plugged into side No. 2. Also, the signals found at each pin of the eight read switching circuit boards are illustrated.

400-500 SERIES-SIDE NO. 2

This chart is arranged similarly to the 100-200 series chart for side No. 1. This arrangement facilitates signal checking of the logic network boards plugged into 452 and 552 through 479 and 579 (except that receptacles 462 and 562 comprise a spare connector).
Figure 42. Circuit Board Connector Signal Chart No. 1
Figure 43. Circuit Board Connector Signal Chart No. 2
Figure 44. Circuit Board Connector Signal Chart No. 3
Figure 45. Circuit Board Connector Signal Chart No. 4
Figure 46. Circuit Board Connector Signal Chart No. 5